# ALGORITMI DE ÎNVĂŢARE PENTRU REŢELE NEURONALE CLIFFORD

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea Politehnica Timişoara
în domeniul CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI
de către

**ing. Călin-Adrian Popa**

| | |
|---|---|
| Conducător ştiinţific: | prof.univ.dr.ing. Ştefan Holban |
| Referenţi ştiinţifici: | prof.univ.dr.ing. Gavril Toderean |
| | prof.univ.dr. Alexandru Cicortaş |
| | prof.univ.dr.ing. Ionel Jian |

Ziua susţinerii tezei: 29.05.2015

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor
12. Ingineria sistemelor
13. Inginerie energetică
14. Calculatoare şi tehnologia informaţiei
15. Ingineria materialelor
16. Inginerie şi Management
17. Arhitectură
18. Inginerie civilă și instalații

Universitatea Politehnica Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul Şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.



România, 300159 Timişoara, Bd. Republicii 9,
Tel./fax 0256 403823
e-mail: editura@edipol.upt.ro

# Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Calculatoare al Universităţii Politehnica Timişoara.

Mulţumiri deosebite se cuvin conducătorului de doctorat, prof.dr.ing. Ştefan Holban, care m-a introdus pentru prima dată în tainele inteligenţei artificiale, şi mai ales ale reţelelor neuronale. Fără încrederea constantă pe care a avut-o în mine de la început, chiar în momentele în care nu se întrevedea o direcţie clară, nu aş fi reuşit.

Dedic cu toată dragostea şi recunoştinţa această teză familiei mele, pentru sprijinul şi încurajarea pe care mi le-au oferit pe tot parcursul elaborării ei.

Nu în ultimul rând, aş vrea să-I mulţumesc lui Dumnezeu pentru că mi-a dat inspiraţia, capacitatea şi puterea de a duce la bun sfârşit activitatea de doctorat.

Timişoara, mai 2015 Călin-Adrian Popa

Familiei mele

Popa, Călin-Adrian

**Algoritmi de învăţare pentru reţele neuronale Clifford**

Teze de doctorat ale UPT, Seria 14, Nr. 28, Editura Politehnica, 2015, 232 pagini, 22 figuri, 16 tabele.

ISSN: 2069-8216 ISSN-L: 2069-8216

ISBN: 978-606-554-965-4

Cuvinte cheie: reţele neuronale, algoritmul backpropagation, reţele neuronale cu valori complexe, reţele neuronale Clifford, reţele neuronale cu valori matrici pătratice, reţele neuronale cu valori matrici antisimetrice, medierea rotaţiilor

Rezumat,

În prezenta teză, am extins cei mai cunoscuţi algoritmi de învăţare din cadrul reţelelor neuronale de tip feedforward cu valori reale în domeniul **reţelelor neuronale cu valori complexe**, şi anume metodele gradient îmbunătăţite (quickprop, resilient backpropagation, delta-bar-delta şi SuperSAB), metodele gradienţilor conjugaţi (cu actualizări Hestenes-Stiefel, Polak-Ribiere, Fletcher-Reeves şi Dai-Yuan, cu reporniri Powell-Beale şi cu actualizări Hestenes-Stiefel şi Polak-Ribiere pozitive), metoda gradienţilor conjugaţi scalaţi, metoda Newton (cu algoritmul de calcul al hessianei şi a produsului hessianei cu un vector), metodele quasi-Newton (a actualizării de rang unu, Davidon-Fletcher-Powell, Broyden-Fletcher-Goldfarb-Shanno şi one step secant) şi metoda Levenberg-Marquardt. Am prezentat rezultate experimentale pentru aplicaţii sintetice, respectiv problema XOR, problema XOR extinsă, două probleme de aproximare a unor funcţii complexe complet şi trei probleme de aproximare a unor funcţii complexe divizat, şi pentru aplicaţii din lumea reală, respectiv egalizarea canalelor liniare şi neliniare, predicţia seriilor de timp liniare şi neliniare şi predicţia direcţiei şi vitezei vântului, care au arătat o îmbunătăţire de până la şase ordine de mărime în termeni de eroare medie pătratică faţă de metoda gradient clasică.

Ca o generalizare a reţelelor neuronale cu valori complexe, am dedus aceiaşi algoritmi pentru **reţelele neuronale Clifford**.

În această teză, am introdus de asemenea un **algoritm de încorporare** (embedding algorithm) ce permite calculul mediei pe varietăţi diferenţiabile, cu aplicaţie la medierea rotaţiilor. Am prezentat rezultate experimentale folosind funcţii de tip $L^p$, ceea ce constituie o noutate în literatură, şi în particular am făcut comparaţia între cazul $L^2$ şi cazul $L^4$.

Am introdus, în premieră după cunoştinţele noastre, unele generalizări ale reţelelor neuronale Clifford, şi anume **reţelele neuronale cu valori matrici pătratice**, definite pe algebra matricilor cu operaţiile naturale de adunare şi înmulţire, şi respectiv **reţelele neuronale cu valori matrici antisimetrice**, adică reţele neuronale pe algebra asociată grupului de rotaţii. Rezultatele experimentale, pe trei probleme de aproximare a unor funcţii sintetice în cazul reţelelor neuronale cu valori matrici pătratice, respectiv pe două probleme de aproximare a unor funcţii sintetice şi pe transformări geometrice (translaţia, scalarea şi rotaţia) în cazul reţelelor neuronale cu valori matrici antisimetrice sunt promiţătoare pentru viitorul acestor tipuri de reţele.

# Cuprins

# Listă de figuri

# Listă de tabele

# Capitolul 1

# Introducere

## 1.1 Motivaţie

Reţelele neuronale cu valori complexe sunt reţele neuronale ale căror intrări, ponderi, bias-uri şi ieşiri sunt numere complexe. Apărute pentru prima dată în anii 1970, reţelele neuronale cu valori complexe au început practic să se dezvolte ca domeniu de sine stătător din anii 1990, interesul pentru acest tip de reţele crescând constant în ultimii ani, devenind astfel în prezent un subiect activ de cercetare. Dovada acestui fapt este aceea că unele dintre cele mai prestigioase conferinţe din domeniul reţelelor neuronale au avut sesiuni speciale de reţele neuronale cu valori complexe. Dintre acestea, menţionăm International Joint Conference on Neural Networks (IJCNN), organizată de IEEE Computational Intelligence Society (IEEE CIS) şi International Neural Network Society (INNS), International Conference on Neural Information Processing (ICONIP), promovată de Asia-Pacific Neural Network Assembly (APNNA) şi organizată de Japanese Neural Network Society (JNNS), şi International Conference on Artificial Neural Networks (ICANN) organizată de European Neural Network Society (ENNS). O altă dovadă a importanţei crescânde a domeniului este dedicarea numărului din septembrie 2014 al IEEE Transactions on Neural Newtorks and Learning Systems şi a numărului din aprilie 2008 al Intenational Journal of Neural Systems reţelelor neuronale cu valori complexe. De asemenea, s-a constituit Task Force on Complex-Valued Neural Networks, în cadrul Institute of Electrical and Electronics Engineers (IEEE), Computational Intelligence Society (CIS), Neural Network Technical Committee (NNTC), care are în prezent mai mult de 40 de membri activi în întreaga lume.

Domeniul de aplicare al reţelelor cu valori complexe s-a lărgit considerabil în ultimii ani, acesta ajungând azi să includă aplicaţii în ingineria undelor electromagnetice, luminoase şi ultrasonice cum ar fi observări terestre şi ambientale cu sisteme radar aflate în sateliţi sau în aer, imagistică de securitate în aeroporturi, gări şi alte mijloace de transport în comun, diagnoză medicală şi monitorizarea utilizând fenomene ondulatorii coerente. Alte domenii aflate în expansiune sunt procesarea adaptivă a imaginilor în domeniul frecvenţă şi procesarea semnalelor timp-secvenţială. Pentru o detaliere a multora dintre aceste aplicaţii, a se vedea, spre exemplu, cărţile [118] şi [119] care, fiind foarte recente, rezumă cele mai noi progrese făcute în domeniu.

Ideea de bază din spatele acestui tip de reţele neuronale este aceea că există unele semnale care apar natural sub formă complexă, iar tratarea lor presupunea până acum procesarea separată a părţilor reală şi imaginară ale semnalului, pierzându-se astfel legătura naturală care există între ele. Însă reţelele neuronale cu valori complexe tratează semnalele complexe ca întreg, posedând astfel capacităţi mult mai bune în comparaţie cu reţelele neuronale clasice, cu valori reale.

În multe aplicaţii inginereşti, cum ar fi studiul plăcilor tectonice sau modelarea continuă dependentă de secvenţă a ADN-ului, rezultatele experimentale sunt date sub forma unor rotaţii tridimensionale care însă sunt destul de afectate de zgomot. Din acest motiv, se vrea o metodă de a reduce efectele acestui zgomot prin medierea acestor date experimentale pentru a obţine rezultate cât mai aproape de adevăr. Se naşte astfel problema medierii pe grupul rotaţiilor, adică a matricilor pătratice ortogonale. Deoarece acest grup nu este un spaţiu euclidian, ci doar o varietate diferenţiabilă, problema mediei mai multor puncte din acest spaţiu nu are o soluţie evidentă, deoarece, de exemplu media aritmetică calculată cu adunarea naturală a matricilor nu este la rândul ei o matrice ortogonală. Prin urmare trebuie găsite alte metode de a afla media mai multor matrici ortogonale.

Astfel de tehnici au fost dezvoltate în literatură, şi ele se bazează pe aplicaţia exponenţială, care duce matricile ortogonale în algebra matricilor antisimetrice, unde se poate face media unor matrici, revenindu-se apoi în grupul matricilor ortogonale. Există o serie de articole care folosesc această tehnică, spre exemplu [2], [22], [83], [84], [100], [213]. Aplicaţia exponenţială poate fi apoi înlocuită cu noţiunea de retractă dezvoltată în [2], [3], [82]. Problemele de mediere pe diferite varietăţi vin din aplicaţii din viaţa reală şi au fost studiate în [1], [2], [111], [227], [243], [262] folosind noţiunile de derivată covariantă şi geometria geodezicelor pe varietăţi riemanniene.

Toate aceste contribuţii însă sunt foarte dificile din punct de vedere computaţional şi se pretează doar anumitor tipuri de medii, pentru care calculele pot fi făcute mai uşor. Însă dacă luăm în considerare medii definite de funcţii de cost arbitrare, atunci aceste metode nu mai pot fi aplicate pentru găsirea unei soluţii la problema medierii.

## 1.2 Contribuţia proprie

Din motivele prezentate în secţiunea anterioară, am decis că un subiect interesant de cercetare ar fi **reţelele neuronale cu valori complexe**, în particular cele de tip feedforward sau perceptron multistrat, cum au mai fost denumite.

O caracteristică a cercetării în acest domeniu o constituie încercarea de a extinde din domeniul real în domeniul complex diferiţi algoritmi şi arhitecturi de reţele neuronale. Studiind literatura de specialitate, după cunoştinţa noastră, dintre algoritmii de învăţare cei mai cunoscuţi pentru reţelele neuronale de tip feedforward cu valori reale, doar metoda gradient (a se vedea, spre exemplu [71, 184]), metoda gradient cu moment [143], algoritmul resilient backpropagation [141] şi algoritmul Levenberg-Marquardt [11] au fost extinşi în domeniul complex.

Ţinând cont de acest fapt, am hotărât să realizăm extinderea şi a altor algoritmi din domeniul real în domeniul complex, cum ar fi *metodele gradient îmbunătăţite*, *metodele gradienţilor conjugaţi*, *metoda gradienţilor conjugaţi scalaţi*, *metoda Newton cu calculul exact al hessianei* şi al *produsului hessianei cu un vector*, şi *metodele quasi-Newton*. Am folosit un tip de calcul special creat pentru derivarea în planul complex, şi anume *calculul Wirtinger*, sau *calculul* $\mathbb{CR}$.

Cu aceasta, am construit practic un *cadru* pentru reţelele neuronale cu valori complexe, folosind MATLAB, unde am creat un pachet pentru acest tip de reţele care este analogul pachetului disponibil în MATLAB pentru reţelele neuronale cu valori reale, aflat în Neural Network Toolbox, unde sunt disponibili pentru a fi folosiţi la antrenare toţi algoritmii enumeraţi mai sus. Acest pachet pentru reţelele neuronale de tip feedforward cu valori complexe poate fi folosit într-o gamă largă de aplicaţii ale reţelelor neuronale cu valori complexe, pornind de la probleme de aproximare a unor funcţii, la egalizarea cu canal liniar sau neliniar folosită în modularea analogică a semnalelor,

la predicţia seriilor de timp liniare şi neliniare, cu aplicaţii în procesarea semnalelor, precum şi la predicţia direcţiei şi vitezei vântului.

Rezultatele experimentale pe toate aceste tipuri de aplicaţii enumerate mai sus arată o îmbunătăţire semnificativă a performanţelor reţelelor neuronale feedforward cu valori complexe în comparaţie cu metoda gradient clasică de antrenare a acestor reţele, dar şi în comparaţie cu alte arhitecturi şi reţele folosite în literatură pentru a învăţa aceste probleme. În funcţie de algoritm şi de problemă, se constată îmbunătăţiri chiar şi cu patru ordine de mărime în termeni de eroare medie pătratică, ceea ce constituie o creştere semnificativă a performanţei.

Aceste experimente mai relevă faptul că performanţa este puternic dependentă de caracteristicile problemei de rezolvat, astfel încât unii algoritmi sunt mai eficienţi pe probleme de aproximare de funcţii, iar alţii pe egalizare şi predicţie de semnale. Acesta este şi motivul pentru care nu ne-am limitat la deducerea şi implementarea unei singure clase de algoritmi, ci am hotărât, pornind de la ideea din MATLAB unde sunt prezenţi aproape toţi aceşti algoritmi, chiar şi în ultima versiune MATLAB R2015a, că ei trebuie puşi toţi la dispoziţia celor interesaţi de aplicaţiile reţelelor neuronale cu valori complexe. Doar experimente făcute pe fiecare problemă în parte vor putea releva care algoritm are cea mai bună performanţă pe respectiva problemă. În manualul de utilizare al Neural Network Toolbox pentru MATLAB R2015a, există experimente făcute pe diverse probleme care arată că fiecare algoritm are anumite caracteristici care sunt de dorit într-o anumită aplicaţie.

Astfel, intenţia a fost de a pune la dispoziţia celor interesaţi o paletă largă de algoritmi pentru ca aceştia să poată alege, în urma experimentelor făcute cu pachetul creat, algoritmul cu cea mai bună performanţă şi cele mai bune caracteristici necesare în aplicaţiile lor de interes.

Ca o generalizare a reţelelor neuronale cu valori complexe, am dedus *aceiaşi* algoritmi pentru **reţelele neuronale Clifford**. Datorită interesului din ultimii ani pentru acest tip de reţele, am considerat oportun să prezentăm aceleaşi metode de învăţare şi pentru acest tip de reţele, ţinând cont de faptul că doar metoda gradient (a se vedea, spre exemplu [211]) a mai fost dedusă în altă parte în literatură, restul algoritmilor apărând aici pentru prima dată. Calculul diferenţial folosit este cel cu derivate parţiale reale, care se poate particulariza şi pentru reţelele neuronale cu valori complexe, dând formule diferite pentru algoritmi, care însă sunt echivalente cu cele deduse folosind calculul Wirtinger, sau calculul $\mathbb{CR}$.

Algebrele Clifford sau algebrele geometrice au multe aplicaţii în fizică şi inginerie, şi este natural că reţele neuronale cu valori în acest tip de algebră au apărut. Datorită legăturii strânse dintre algebrele Clifford şi geometrie, reţelele neuronale definite pe aceste algebre vor fi capabile să proceseze diferite obiecte geometrice şi să aplice diferite modele geometrice datelor. Aceasta va oferi, în viitor, o modalitate de a rezolva multe probleme care apar în design-ul sistemelor inteligente. Contribuţia noastră vine să preîntâmpine aceste aplicaţii, prin punerea la dispoziţia celor interesaţi a deducerii teoretice a principalilor algoritmi care au avut performanţe semnificative în cazul real.

Principala particularizare a acestui tip de reţele o constituie *reţelele neuronale cu valori cuaternionice*. Prin urmare, algoritmii mai sus propuşi se pot adapta uşor pentru algebra Clifford a cuaternionilor, de dimensiune $4$. Reţelele neuronale cu valori cuaternionice au apărut iniţial ca o generalizare a reţelelor neuronale cu valori complexe, însă curând au început să aibă aplicaţii interesante, de la predicţia seriilor de tip haotice, la problema parităţii pe $4$ biţi, şi nu în ultimul rând la procesarea semnalelor cu valori cuaternionice. Analog cu semnalele care apar natural în formă complexă, există semnale care apar natural în formă cuaternionică, astfel că ultimii trei ani au cunoscut practic o explozie a cercetării semnalelor cu valori cuaternionice, această teorie fiind în

plină dezvoltare, motiv pentru care aceşti algoritmi ar putea avea aplicaţii interesante în viitor. Deoarece algebra cuaternionilor are dimensiune $4$, obiectele tridimensionale pot fi reprezentate folosind cuaternioni, şi aplicaţii ale acestui tip de reţele au apărut în procesarea imaginilor tridimensionale, de asemenea.

Algoritmii dezvoltaţi vor putea fi implementaţi pentru a crea un *cadru* pentru reţelele neuronale cu valori cuaternionice în MATLAB, analog celui deja creat pentru reţelele cu valori complexe. Pornind de la deducerea teoretică a acestor algoritmi, şi de la implementarea făcută pentru reţelele neuronale cu valori complexe, acest cadru se poate implementa foarte uşor, nepunând niciun fel de probleme pentru cei interesaţi. Ca o contribuţie viitoare, avem în vedere crearea unui astfel de cadru, şi folosirea lui pe cele mai cunoscute benchmark-uri din domeniu, cum am făcut pentru reţelele cu valori complexe.

Tot din motivele prezentate în secţiunea anterioară, am decis de asemenea că un subiect interesant de cercetare ar fi găsirea unui algoritm de **mediere a matricilor ortogonale** care să se poată folosi în orice situaţie, pentru orice funcţie de cost, şi care să presupună calcule mai directe, care pot fi inclusiv automatizate cu ajutorul calculatorului, ceea ce nu se poate întâmpla în cazul abordărilor clasice din domeniu.

Astfel, algoritmul nostru, numit *algoritm de încorporare*, presupune liftarea funcţiei de cost iniţiale la o varietate diferenţiabilă care poate fi scufundată într-o varietate a riemanniană (spaţiu euclidian) şi construirea unui câmp de vectori definit pe spaţiul ambient a cărui restricţie la varietatea scufundată este câmpul de vectori gradient al funcţiei de cost liftate. Avantajul acestei metode este acela că ne permite să facem calcule în coordonate carteziene în loc să folosim coordonate locale şi derivate covariante pe varietatea iniţială. Astfel, pot fi calculate medii date de orice funcţie de cost, indiferent de cât de complicată este expresia ei, pentru că algoritmul presupune doar calcule în coordonate carteziene.

O altă parte a contribuţiei noastre o constituie introducerea, în premieră după cunoştinţele noastre, a **reţelelor neuronale cu valori matriciale**, care reprezintă *generalizări* ale reţelelor neuronale Clifford. În aceste reţele neuronale intrările, ieşirile, ponderile şi bias-urile sunt matrici. Am decis să discutăm două cazuri, şi anume **reţelele neuronale cu valori matrici pătratice**, care sunt considerate pe algebra matricilor cu operaţiile naturale de adunare şi înmulţire, şi respectiv **reţele neuronale cu valori matrici antisimetrice**, adică reţele neuronale pe algebra asociată grupului de rotaţii.

În ambele cazuri, este introdusă *metoda gradient* de antrenare a reţelelor neuronale de tip feedforward. Ideea acestor reţele vine de la faptul că algoritmul de mediere propus poate fi folosit pentru a genera seturi de date de antrenare pentru reţele neuronale, care pot astfel învăţa medii de orice fel, permiţându-ne să găsim media unor rotaţii fără a mai trece măcar prin calculele presupuse de algoritmul de încorporare propus. Având valorile din grup, ele pot fi trecute în algebra asociată, date reţelei neuronale, iar rezultatele trecute înapoi în grup.

Datorită gradului lor de generalitate, se poate prevedea că reţelele neuronale cu valori matrici pătratice vor avea multe aplicaţii în viitor la rezolvarea unor probleme la care reţelele neuronale cu valori reale au eşuat sau au avut performanţe slabe. Ele pot fi folosite pentru a rezolva ecuaţii matriciale, pentru a calcula mediile unor matrici, sau pentru a învăţa orice funcţii definite pe algebra matricilor. Reţelele neuronale cu valori matrici pătratice ar putea avea aplicaţii interesante în computer vision, procesarea de imagini, şi toate celelalte domenii legate de transformări geometrice ale obiectelor, dar şi pe unele probleme $n$-dimensionale.

Reţelele neuronale cu valori matrici antisimetrice ar putea avea aplicaţii interesante în computer vision şi toate celelalte domenii legate de transformări geometrice ale obiectelor tridimensionale, dar ar putea avea performanţe mai bune şi pe unele probleme $n$-dimensionale decât soluţiile disponibile în prezent.

Rezultatele experimentale ale metodei gradient pentru reţelele feedforward cu valori matrici pătratice pe trei probleme de aproximare de funcţii, inclusiv inversa matricială, dau motive de optimism privind viitorul acestor reţele. În cazul reţelelor neuronale cu valori matrici antisimetrice, am folosit două probleme de aproximare de funcţii, precum şi trei transformări geometrice: translaţia, scalarea şi rotaţia, cu rezultate foarte bune.

Paleta de aplicaţii posibile ale acestor două tipuri de reţele este largă, şi sperăm ca, pe viitor, acest prim pas făcut să dea naştere unei teorii a reţelelor neuronale cu valori matriciale, asemenea celor care s-au dezvoltat sau sunt în curs de dezvoltare pentru reţelele neuronale cu valori complexe şi cu valori cuaternionice. Ca o contribuţie viitoare, intenţionăm să dezvoltăm şi alte tipuri de reţele cu valori matriciale pe lângă cele feedforward, care le vor mări şi mai mult domeniul de aplicabilitate.

## 1.3 Structura tezei

Astfel, teza este structurată după cum urmează:

- Capitolul 2 face o prezentare sintetică a **reţelelor neuronale cu valori complexe**, a **reţelelor neuronale cu valori hiperbolice**, a **reţelelor neuronale cu valori cuaternionice** şi a **reţelelor neuronale cu valori în algebre Clifford**, începând cu descrierea respectivelor sisteme de numere, şi continuând cu principalele caracteristici ale reţelelor feedforward de tipurile respective.

- Capitolul 3 prezintă câte o secţiune pentru fiecare dintre metodele de învăţare nou propuse pentru **reţelele neuronale cu valori complexe**, începând cu clasica *metodă gradient*, continuând cu *metodele gradient îmbunătăţite*, *metodele gradienţilor conjugaţi*, *metoda gradienţilor conjugaţi scalaţi*, *metoda Newton cu calculul exact al hessianei* şi al *produsului hessianei cu un vector*, *metodele quasi-Newton* şi terminând cu *metoda Levenberg-Marquardt*. Pentru a deduce aceşti algoritmi, am folosit *calculul Wirtinger*, sau *calculul* $\mathbb{CR}$, care este utilizat pe larg în literatura de specialitate pentru a extinde algoritmi de învăţare din planul real în planul complex.

- Deoarece pentru **reţelele neuronale Clifford** nu există ceva asemănător calculului Wirtinger, am folosit calculul obişnuit cu derivate parţiale pentru a deduce *aceiaşi algoritmi* ca mai sus şi pentru acest tip de reţele în Capitolul 4. Datorită generalităţii acestui tip de reţele, algoritmii deduşi se pot particulariza uşor pentru *reţele neuronale cu valori hiperbolice*, pentru *reţele neuronale cu valori cuaternionice*, şi chiar pentru reţele neuronale cu valori complexe, fiind o alternativă de exprimare a algoritmilor deduşi în capitolul precedent.

- Capitolul 5 prezintă *metoda gradient* pentru antrenarea **reţelelor neuronale** de tip feedforward **cu valori vectori tridimensionali**, **cu valori vectori $n$-dimensionali**, şi respectiv **cu valori în algebra matricilor pătratice** şi **în algebra matricilor antisimetrice**. În acelaşi capitol se prezintă *algoritmul de încorporare* dezvoltat pentru a uşura calculul mediilor date de diferite funcţii de cost pe varietăţi riemanniene, şi în particular **calculul mediilor pe grupul matricilor ortogonale**.

- Capitolul 6 prezintă **rezultatele experimentale** ale aplicării algoritmilor propuşi pentru cele mai cunoscute benchmark-uri folosite în literatura de specialitate, atât pentru aplicaţii sintetice, cât şi pentru aplicaţii din lumea reală, în cadrul *reţelelor neuronale cu valori complexe*. Aplicaţiile sintetice includ problema XOR, problema XOR extinsă şi aproximarea a două funcţii complexe complet şi a trei funcţii complexe divizat. Aplicaţiile din lumea reală sunt egalizarea cu canal liniar şi respectiv neliniar, predicţia seriilor de timp liniare şi respectiv neliniare, precum şi predicţia direcţiei şi vitezei vântului. Tot în acest capitol sunt prezentate aplicaţii sintetice pe probleme de aproximare de funcţii ale metodei gradient pentru *reţelele neuronale cu valori matrici pătratice*, respectiv *cu valori matrici antisimetrice*, precum şi rezultatele experimentale ale *medierii pe grupul matricilor ortogonale*.
- În final, Capitolul 7 prezintă **concluziile** prezentei teze.

# Capitolul 2

# Reţele neuronale Clifford

Acest capitol face o introducere în domeniul reţelelor neuronale Clifford, prezentând o istorie succintă a domeniului, principalele contribuţii din literatură, cele mai importante aplicaţii şi caracteristicile generale ale algebrelor numerelor complexe, hiperbolice, cuaternionice şi Clifford.

Astfel, Secţiunea 2.1 este dedicată reţelelor neuronale cu valori complexe. După o scurtă introducere privind apariţia acestui tip de reţele, se discută cele mai importante contribuţii teoretice din domeniu, iar apoi se evidenţiază principalele aplicaţii ale acestor reţele. Apoi, după o scurtă istorie a numerelor complexe, se prezintă trei definiţii echivalente ale mulţimii acestor numere, cu sublinierea caracteristicilor calculului diferenţial în planul complex. Secţiunea se încheie cu deducerea algoritmului backpropagation pentru metoda gradient de învăţare a unei reţele neuronale de tip feedforward cu valori complexe, deoarece prezenta teză se ocupă doar de reţele feedforward.

Secţiunea 2.2 prezintă reţelele neuronale cu valori hiperbolice, şi începe cu evidenţierea principalelor contribuţii care vizează acest tip de reţele. Datorită faptului că algebra numerelor hiperbolice este tot de dimensiune doi, aplicaţiile sunt mult mai puţine decât cele din cadrul reţelelor cu valori complexe, însă este posibil ca ele să apară pe viitor. Din acest motiv, după această scurtă introducere se trece la definirea echivalentă a numerelor hiperbolice, în acelaşi fel cu a numerelor complexe. Acelaşi algoritm backpropagation este evidenţiat pentru o reţea cu valori hiperbolice de tip feedforward.

Următoarea secţiune a acestui capitol, Secţiunea 2.3, am dedicat-o reţelelor neuronale cu valori cuaternionice. Dintre toate reţelele de tip Clifford care au fost discutate în literatura de specialitate, reţelele cuaternionice sunt cele mai populare, bineînţeles după reţelele cu valori complexe. Studiul în aceste reţele s-a intensificat foarte mult, motiv pentru care prezentarea contribuţiilor şi aplicaţiilor celor mai importante se face cu citarea unor lucrări de dată foarte recentă. Aceeaşi schemă a celor trei definiţii echivalente este aplicată şi pentru mulţimea numerelor cuaternionice, şi acelaşi algoritm este prezentat pentru antrenarea unei reţele de tip feedforward, acesta fiind perfect analog celui din cazurile complex şi hiperbolic.

Apoi, Secţiunea 2.4 introduce reţelele neuronale Clifford, care dau şi titlul prezentei teze, şi care reprezintă o generalizare $2^n$-dimensională a reţelelor reale, complexe, hiperbolice şi cuaternionice. După prezentarea principalelor contribuţii din domeniul acestor reţele, se trece la construirea progresivă a definiţiei algebrelor Clifford. Odată definite algebrele Clifford, se continuă cu prezentarea algoritmului backpropagation generalizat pentru reţelele neuronale Clifford, care reprezintă, de fapt, o extindere a aceluiaşi algoritm din cadrul reţelelor cu valori complexe, hiperbolice şi cuaternionice.

În fine, Secţiunea 2.5 prezintă concluziile acestui capitol. El face o descriere a stării actuale a domeniului reţelelor de tip Clifford şi prezintă cel mai cunoscut algoritm pentru antrenarea reţelelor feedforward, şi anume metoda gradient, care va constitui baza pentru capitolele următoare, unde sunt deduşi alţi algoritmi de antrenare a acestor reţele.

## 2.1 Reţele neuronale cu valori complexe

În anul 1971, Aizenberg et al. din Uniunea Sovietică (Rusia de azi) au avut ideea de a extinde valorile binare $0$ şi $1$ la mai multe valori situate pe cercul unitate din planul complex [9]. Ei au propus de asemenea o implementare bazată pe modularea puls poziţie (PPM) în care au codificat multiplele valori ale fazei în secvenţa de timp a pulsului $1$ în unitatea de timp. Această idee este compatibilă cu aplicaţiile moderne din domeniul undelor electromagnetice, sugerând existenţa unei legături strânse între timp şi fază.

În cazul de mai sus, semnalele de ieşire cu valori multiple sunt plasate în puncte discrete pe cercul unitate. Această clasă de reţele neuronale cu valori complexe sunt uneori numite *reţele neuronale fazoriale*, în care amplitudinea este fixată ca fiind egală cu unitatea, deşi fazorul nu este neapărat un vector cu amplitudine fixată în planul complex. Grupul lui Aizenberg a publicat o carte despre reţelele neuronale cu valori multiple în anul 2000 [8]. Activitatea acestui grup continuă şi astăzi, când, deşi inventatorul iniţial al reţelelor cu valori multiple, Naum Aizenberg, nu mai este în viaţă, fiul său, Igor Aizenberg continuă cercetarea în acest domeniu.

În prelucrarea adaptivă a semnalelor în sisteme radar, de comunicaţii şi alte aplicaţii, semnalele cu valori complexe apar inevitabil şi ele trebuie prelucrate. În cadrul procesării liniare, Widrow et al. din Statele Unite ale Americii a propus algoritmul celor mai mici pătrate complex (complex least mean squares, CMLS) în 1975 [264]. Deoarece acest algoritm este de tip gradient cu o eroare de tip pătratic, şi este de asemenea şi liniar, dinamica lui este clară. Practic ceea ce se întâmplă este că transpusa vectorilor şi matricilor din procesarea semnalelor din domeniul real este înlocuită cu transpusa hermitiană (sau, altfel spus, transpusa conjugată) în domeniul complex. Această operaţie se regăseşte pe larg în ştiinţă şi tehnologie, ca de exemplu în mecanica cuantică.

În anul 1988, Noest din Olanda a propus memoria asociativă cu valori multiple, sau fazorială [205] şi a obţinut capacitatea de memorare [204]. El a descoperit că, într-o anumită condiţie, capacitatea normalizată a memoriei asociative fazoriale este $\pi/4$, puţin mai mare decât capacitatea memoriei asociative binare, care este $2/\pi$.

Apoi, în anii 1991 şi 1992, au fost prezentate metoda gradient şi algoritmul backpropagation de învăţare, de către mai mulţi cercetători în mod independent. În 1991, Leung & Haykin au considerat o reţea neuronală cu valori complexe care are o funcţie de activare de tip sigmoid complexă complet

$$f(z) = \frac{1}{1 + e^{-z}}.$$

Ei au dezvoltat algoritmul de învăţare pentru o astfel de reţea folosind derivate parţiale ale părţilor reale şi imaginare ale funcţiilor care apar în cadrul metodei gradient [163].

În 1992, Benvenuto & Piazza au considerat o reţea neuronală cu valori complexe a cărei funcţii de activare este de tip sigmoid complexă divizat

$$f(z) = \frac{1}{1 + e^{-\mathrm{Re}z}} + i\frac{1}{1 + e^{-\mathrm{Im}z}},\ i = \sqrt{-1}.$$

Aceasta este una dintre cele mai simple extensii ale funcţiei de activare sigmoid din domeniul real. Avantajul ei este că putem de asemenea să obţinem algoritmul back-propagation ca o simplă extensie a aceluiaşi algoritm de la reţelele neuronale cu valori reale [39].

Articolul lui Nitta [191] a propus de asemenea o reţea neuronală cu valori complexe cu funcţie de activare complexă divizat şi a studiat dinamica acesteia tratând separat părţile reală şi imaginară. El a transformat imaginile bidimensionale formate din puncte în planul complex pentru a discuta capacitatea de generalizare bidimensională a acestui tip de reţea.

Există şi alte articole care au introdus metoda gradient şi algoritmul backpropagation pentru reţele neuronale cu funcţii de activare complexe divizat. De exemplu, scopul articolelor lui Kim & Guest [143] şi Birx & Pipenberg [43] este implementarea optică.

În acest context, articolul lui Georgiou & Kutsougeras din 1992 [90] discută ce clasă de funcţii de activare este cea mai potrivită, după ce face la început o analiză teoretică a funcţiilor de activare complexe divizat. Presupunând o implementare folosind circuite analogice, ei au propus o funcţie a cărei amplitudine se saturează, câtă vreme faza rămâne neschimbată, şi anume

$$f(z) = \frac{z}{c + |z|/r},$$

unde $c$ şi $r$ sunt parametri reali care determină caracteristica saturării. Luând în considerare semnalele sau undele din circuitele electronice, o astfel de funcţie de activare reprezintă o alegere naturală.

În acelaşi an, Hirose a propus o memorie asociativă care foloseşte o funcţie de activare de tip amplitudine-fază, independent de grupul lui Georgiou, şi a demonstrat aducerea din memorie netedă (smooth recall) în domeniul complex [114]. Această propunere va da naştere unor implementări fizice cum sunt dispozitivele cuantice, în viitor.

Tot în 1992, Hirose a propus metoda gradient şi algoritmul backpropagation pentru reţele neuronale cu valori complexe care au funcţii de activare de tip amplitudine-fază, ca cea de mai sus [113], şi a demonstrat eficienţa lor. Ele au fost deduse folosind derivate parţiale în direcţiile amplitudinii şi fazei, care sunt compatibile cu acest tip de funcţie de activare. O caracteristică importantă a algoritmului backpropagation este aceea că nu semnalele de eroare sunt propagate înapoi, ci din contră semnalele de antrenare sunt propagate înapoi prin straturile reţelei, ceea ce îl face compatibil cu implementări cu unde de lumină şi unde electromagnetice.

Se poate observa că anul 1992 a fost fructuos în ceea ce priveşte cercetarea în domeniul reţelelor neuronale cu valori complexe. Takeda & Kishigami au raportat teoria şi experimentele pe o memorie asociativă cu valori complexe folosind unde de lumină [248]. Ei au observat faptul că expresia matematică a câmpului undei de lumină într-un rezonator este identică cu cea a unei memorii asociative în domeniul complex. Succesul experimentelor lor a deschis o nouă direcţie de cercetare. Pentru a construi o teorie care să fie eficace în lumea reală, este important ca aceasta să fie dezvoltată în conformitate cu realitatea fizică. Ei au reuşit de asemenea să prezinte un exemplu excelent al faptului că un fenomen fizic de bază reprezintă o realizare a unei reţele neuronale cu valori complexe.

Urmând ideile legate de reţelele cu funcţii de activare de tip amplitudine-fază, grupul lui Hirose a publicat mai multe articole despre reţelele neuronale bazate pe unde de lumină coerente (coherent lightwave neural networks), inclusiv un articol care descrie implementarea lor fizică folosind detecţia homodină şi detaliile algoritmului

backpropagation de tip amplitudine-fază [115], precum şi alte trei lucrări care prezintă analiza numerică a caracteristicilor de generalizare în domeniul frecvenţei [122, 123, 121].

Mai mult, ei au propus realizarea comportării dependentă de frecvenţa purtătoare (carrier-frequency-dependent behavior) în reţele neuronale bazate pe unde electromagnetice [122, 120]. Această idee conduce nu numai la controlul comportării reţelei neuronale prin modularea frecvenţei purtătoare, ci şi la apariţia voinţei în combinaţie cu un feedback al semnalelor de ieşire către controlere. Se realizează apoi multiplexarea în domeniul frecvenţă în vasta bandă de frecvenţă optică. Aceste avantaje îşi au originea în manipularea precisă dependentă de frecvenţă a semnalelor prin folosirea informaţiei de fază în reţelele neuronale cu valori complexe.

Aspectele teoretice ale reţelelor neuronale cu valori complexe au fost discutate din mai multe puncte de vedere. Hanna & Mandic au analizat dinamica metodei gradient de învăţare într-un filtru cu un singur strat având funcţia de activare de tip sigmoid complexă divizat sau de tip amplitudine-fază [106]. Ei au investigat de asemenea ieşirile rezultate în urma procesului de filtrare [104]. Hanna & Mandic [105] au discutat efectul reutilizării datelor în învăţarea reţelelor neuronale cu valori complexe. În plus, multe articole au făcut analize ale rezultatelor unor algoritmi de învăţare, ca de exemplu Casasent & Natarajan [60], Nitta [191, 196], Goh & Mandic [93, 96, 97], Fiori [81] şi Xu et al. [268]. Există de asemenea evaluări ale erorii perceptronilor de către Yang et al. [272] şi reducerea numărului de neuroni de pe stratul ascuns de către Kobayashi [151].

Caracteristicile funcţiilor de activare au fost discutate, spre exemplu de Kim & Adali [148, 146], Jankowski et al. [137]. În ceea ce priveşte memoriile asociative cu valori complexe, există mai multe articole privind funcţiile energie şi dinamica: Kuroe et al. [155], Nemoto & Kubono [181], Lee & Wang [162], Aoki & Kosugi [15], Lee [160, 159, 161], Takahashi [247], Muezzinoglu et al. [179]. Dinamica reţelelor neuronale cu valori complexe cu funcţii de activare complexe divizat a fost analizată când aceste reţele au fost folosite pentru transformări geometrice în planul complex de către Nitta [193, 195, 197]. Analiza componentelor principale (principal component analysis, PCA) şi analiza componentelor independente (independent component analysis, ICA) au fost cercetate din diferite puncte de vedere de către Zhang & Ma [280], Rattan & Hsieh [224], Uncini & Piazza [259], Anemuller et al. [14], Li & Adali [164, 166], Novey & Adali [206]. Reţelele oscilatoare (oscillatory networks) sunt de asemenea strâns legate de reţelele cu valori complexe, şi ele au fost studiate de către Burwick [58, 59].

Reţelele neuronale cu valori cuaternionice şi Clifford au fost propuse şi analizate de către Pearson & Bisset [212], Arena et al. [17], Isokawa et al. [131], Kusamichi et al. [157], Buchholz [53, 48], Byro-Corrochano & Arana-Daniel [34], şi alţii.

În ceea ce priveşte aplicaţiile reţelelor neuronale cu valori complexe, se poate spune că exisă multe aplicaţii în domeniile microundelor şi a undelor milimetrice, ca design-ul adaptiv bazat pe reţele neuronale cu valori complexe a antenelor patch, estimarea direcţiei de sosire a undelor (direction of arrival, DoA) folosind reţele neuronale cu valori complexe propusă de Yang et al. [274], formarea adaptivă a razelor (beamforming) în antenele array de către Yang & Bose [271], Yamaki & Hirose [270], sau radare penetrante ale solului (ground penetrating radars, GPRs) pentru detecţia minelor antipersoană de către Hara & Hirose [107], Masuyama & Hirose [175], Masuyama et al. [176].

În analiza neurofiziologică, dinamica neuronului de tip Nagumo-Sato cu valori complexe a fost folosită pentru a reprezenta activităţile neuronale pe secvenţe de timp şi

apariţia comportării haotice de către Nemoto & Saito [182]. În bioinformatică şi procesarea imaginilor din bioinformatică, expresia genetică a fost analizată pentru clasificarea stadiilor expresiei genetice folosind reţele neuronale cu valori complexe de către Aizenberg et al. [6].

În memoriile asociative, mai multe modalităţi interesante au fost propuse pentru a manipula date transformate în domeniul frecvenţei de către Aizenberg & Butakoff [5], Aizenberg et al. [7] şi Tanaka & Aihara [249].

Egalizatoarele adaptive şi filtrele cu valori complexe au fost propuse în multe articole, spre exemplu de You & Hong [276], Gan et al. [86], Jianping et al [139], Park & Jeong [208], Deng & Yang [75] şi Chen et al. [68].

Există de asemenea un număr mare de articole în domeniul procesării secvenţelor de timp cu valori complexe, ca de exemplu Chakravarthy & Ghosh [63], unde s-a discutat încorporarea secvenţelor de tipare în oscilaţii. Reţelele neuronale cu valori complexe au o comportare dinamică mai stabilă în comparaţie cu cele cu valori reale când sunt de tipul reţele neuronale recurente, cum ar fi memorii asociative cu secvenţe de timp (time-sequential associative memories) [116, 124, 125]. Stabilitatea dinamică este aplicabilă şi în alte domenii. De exemplu, memorarea şi extragerea melodiilor (muzicale) a fost realizată de către Kinouchi & Hagiwara [149]. Alte metode de învăţare pentru predicţia secvenţelor de timp au fost de asemenea propuse în Rajagopal & Kak [183], care arată un algoritm cu învăţare rapidă.

Una dintre aplicaţiile recente este controlul optimal dependent de trafic (traffic-dependent optimal control) al semafoarelor conectate prin drumuri, propus şi analizat de către Nishikawa & Kuroe [183]. Ei au considerat dinamica de învăţare a unui mare număr de oscilatoare (semafoare) conectate prin drumuri pentru a face maşinile să treacă mai repede. Reţelele neuronale cu valori complexe au fost de asemenea aplicate pentru a calcula inversa matricilor de către Song & Yam [242], care este o aplicaţie matematică. În ceea ce priveşte cuaternionii, avem o reţea neuronală cu valori cuaternionice aplicată în învăţarea în spaţiul tridimensional color RGB, pentru a transforma imagini color, propusă de Isokawa et al. [131].

O varietate de aplicaţii hardware ale reţelelor neuronale cu valori complexe a fost propusă folosind unde de lumină [142], unde electromagnetice, în particular unde electrice de înaltă frecvenţă, unde de electroni, şi unde sonice sau ultrasonice. Cum procesarea de înaltă frecvenţă necesită operarea la viteze mari, reţelele neuronale analogice sunt preferabile reţelelor neuronale digitale sau pulsate. Deşi reţelele neuronale analogice pot procesa semnalele rapid, precizia scăzută a fost de obicei o mare problemă. Însă, progrese recente în componente ale circuitelor reuşesc să depăşească acest neajuns.

Dovadă a interesului pentru domeniul reţelelor neuronale cu valori complexe îl reprezintă numeroasele cărţi de autor şi colective care au apărut, cu mai mare frecvenţă, mai ales în ultimii ani. Astfel avem cărţile de autor:

- „Neural Networks in Multidimensional Domains” de P. Arena, L. Fortuna, G. Muscato, M.G. Xibilia (Springer, 1998) [18],
- „Multi-Valued and Universal Binary Neurons – Theory, Learning and Applications” de I. Aizenberg, N. Aizenberg, J. Vandewalle (Kluwer Academic Publishers, 2000) [8],
- „Complex-Valued Neural Networks” de A. Hirose (Springer, 2006, 2012) [118],
- „Complex Valued Nonlinear Adaptive Filters – Noncircularity, Widely Linear and Neural Models” de D.P. Mandic & S.L. Goh (Wiley, 2009) [171],

- „Complex-Valued Neural Networks with Multi-Valued Neurons” de I. Aizenberg (Springer, 2011) [4],
- „Supervised Learning with Complex-Valued Neural Networks” de S. Suresh, N. Sundararajan, R. Savitha (Springer, 2013) [245],

şi cărţile colective:

- „Complex-Valued Neural Networks: Theories and Applications” editată de A. Hirose (World Scientific, 2003) [117],
- „Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters” editată de T. Nitta (Information Science Publishing, 2009) [201].
- „Complex-Valued Neural Networks: Advances and Applications” editată de A. Hirose (Wiley, 2013) [119].

În cele ce urmează, vom face o scurtă prezentare a istoriei numerelor complexe, urmărind în principal referinţele [171] şi [118].

Nevoia de a introduce numere complexe a apărut în contextul încercării mai multor matematicieni de a demonstra Teorema Fundamentală a Algebrei care spune că „Orice polinom de gradul $n$ cu coeficienţi reali are exact $n$ rădăcini complexe”. În secolul al XVI-lea, Niccolo Tartaglia şi Girolamo Cardano au căutat să găsească formule pentru calculul rădăcinilor polinoamelor de gradul trei şi patru.

Numerele complexe au fost introduse de către Girolamo Cardano în lucrarea sa „Ars Magna” din 1545, folosindu-le ca un mijloc de a găsi rădăcinile reale ale ecuaţiei cubice

$$ax^3 + bx^2 + cx + d = 0,$$

pe care, folosind substituţia $y = x - \frac{1}{3}b$, a transformat-o în ecuaţia redusă (i.e. fără termenul pătratic):

$$y^3 + \alpha y + \beta = 0.$$

Această ecuaţie poate fi rezolvată folosind celebra formulă descoperită de Scipione del Ferro şi Niccolo Tartaglia, care astăzi este însă cunoscută sub numele de formula lui Cardano:

$$y = \sqrt[3]{-\frac{\beta}{2} + \sqrt{\frac{\beta^2}{4} + \frac{\alpha^3}{27}}} + \sqrt[3]{-\frac{\beta}{2} - \sqrt{\frac{\beta^2}{4} + \frac{\alpha^3}{27}}}.$$

Rafael Bombelli a analizat de asemenea rădăcinile polinoamelor cubice cu ajutorul formulei de mai sus. Astfel, încercând să rezolve ecuaţia

$$y^3 - 15y - 4 = 0,$$

el a arătat că $(2 + \sqrt{-1}) + (2 - \sqrt{-1}) = 4$. Într-adevăr $y = 4$ este o soluţie corectă, însă pentru a găsi această soluţie reală, trebuie făcute calcule cu numere complexe. În anul 1572, în lucrarea sa „Algebra”, Bombelli a introdus pentru prima dată simbolul $\sqrt{-1}$ şi a stabilit primele reguli pentru operaţii cu numere complexe.

În 1806, Jean Robert Argand, încercând să demonstreze Teorema Fundamentală a Algebrei, a dat prima dată o interpretare geometrică a numerelor complexe ca puncte în plan. El a interpretat unitatea imaginară $\sqrt{-1}$ ca fiind o rotaţie cu $90^\circ$, introducând planul Argand (sau diagrama Argand), în care $\pm\sqrt{-1}$ reprezintă unitatea pe o dreaptă perpendiculară pe axa reală. Notaţia şi terminologia folosite astăzi au rămas aproximativ la fel. Un număr complex

$$z = x + iy,$$

se reprezintă ca un vector în planul complex. Argand a numit valoarea $\sqrt{x^2+y^2}$ modul, iar Karl Friedrich Gauss a fost cel care a introdus termenul de număr complex şi notaţia $i=\sqrt{-1}$. Gauss a folosit de asemenea numere complexe în demonstraţiile sale la Teorema Fundamentală a Algebrei, iar, în 1831, nu numai că a asociat numărului complex $z=x+iy$ un punct $(x,y)$ din plan, ci, mai mult, a introdus regulile pentru adunarea şi înmulţirea acestor numere. Mare parte din terminologia folosită astăzi a fost introdusă de Gauss, de Augustin-Louis Cauchy care a introdus termenul de conjugat şi de Hermann Hankel care, în 1867, a introdus termenul de coeficient de direcţie pentru $\cos\theta+i\sin\theta$.

Între timp însă, Leonhard Euler a folosit numerele complexe în calcule în mod intuitiv, însă corect. Se crede că până în 1728 el a cunoscut relaţia transcendentală $i\log i=-\frac{\pi}{2}$. Formulele lui Euler apar în cartea lui sub forma:

$$\cos x=\frac{e^{ix}+e^{-ix}}{2},\ \sin x=\frac{e^{ix}-e^{-ix}}{2i}.$$

Se crede de asemenea că, în 1749, Euler interpreta deja numerele complexe ca fiind puncte în plan. El a descris un număr $x$ aparţinând cercului unitate ca fiind $x=\cos g+i\sin g$, unde $g$ este un arc al cercului.

Acum vom face o prezentare a mulţimii numerelor complexe, privită din mai multe perspective. În primul rând perspectiva pur complexă, în care mulţimea numerelor complexe este definită ca fiind

$$\mathbb{C}:=\{x+iy|x,y\in\mathbb{R},i=\sqrt{-1}\}.$$

Dacă $z=x+iy\in\mathbb{C}$, $x$ se numeşte *partea reală* a numărului complex $z$, iar $y$ se numeşte *partea imaginară* a lui $z$. Vom nota cu $z^R$ partea reală a lui $z$ şi cu $z^I$ partea imaginară a lui $z$, deci vom putea scrie

$$z=z^R+iz^I,\forall z\in\mathbb{C}.$$

Pentru orice $z_1=x_1+iy_1$, $z_2=x_2+iy_2\in\mathbb{C}$, definim *adunarea* prin

$$z_1+z_2:=(x_1+x_2)+i(y_1+y_2),$$

*înmulţirea* prin

$$z_1\cdot z_2:=(x_1x_2-y_1y_2)+i(x_1y_2+x_2y_1),$$

*inversul* lui $z_1$ prin

$$z_1^{-1}:=\frac{x_1}{x_1^2+y_1^2}-i\frac{y_1}{x_1^2+y_1^2},$$

*conjugatul* lui $z_1$ prin

$$\overline{z_1}:=x_1-iy_1,$$

şi *modulul* lui $z_1$ prin

$$|z_1|:=\sqrt{z_1\overline{z_1}}=\sqrt{x_1^2+y_1^2}.$$

Fiecare număr complex $z=x+iy\in\mathbb{C}$ se poate scrie sub forma

$$z=r(\cos\theta+i\sin\theta),$$

unde $r=|z|:=\sqrt{x^2+y^2}$ este modulul lui $z$, iar $\theta:=\arg z$ se numeşte argumentul numărului complex $z$. Ţinând cont de formulele lui Euler, date mai sus, avem că

$$z=re^{i\theta},$$

ceea ce înseamnă că $z$ se poate scrie şi sub forma

$$z = (r, \theta),$$

numită reprezentarea în coordonate polare a numărului complex $z$.

O altă modalitate de definire a mulţimii numerelor complexe este prin identificarea ei cu perechile ordonate de numere reale, astfel

$$\mathbb{C} := \{(x, y) | x, y \in \mathbb{R}\}.$$

Acum, pentru orice $z_1 = (x_1, y_1)$, $z_2 = (x_2, y_2) \in \mathbb{C}$, definim adunarea prin

$$z_1 + z_2 := (x_1 + x_2, y_1 + y_2),$$

înmulţirea prin

$$z_1 \cdot z_2 := (x_1 x_2 - y_1 y_2, x_1 y_2 + x_2 y_1),$$

şi inversul lui $z_1$ prin

$$z_1^{-1} := \left( \frac{x_1}{x_1^2 + y_1^2}, -\frac{y_1}{x_1^2 + y_1^2} \right).$$

În acest caz se face identificarea $1 \equiv (1, 0)$ şi $i = \sqrt{-1} \equiv (0, 1)$, caz în care se vede că avem $z = x + iy \equiv (x, y)$. Prin această identificare, fiecărui număr complex $z = x + iy$ îi corespunde punctul $(x, y)$ din plan, obţinând astfel reprezentarea geometrică a numărului complex $z$.

O a treia reprezentare posibilă a mulţimii numerelor complexe este ca o subalgebră a algebrei matricilor pătratice de ordinul doi cu elemente reale, notată $\mathcal{M}_2(\mathbb{R})$. Astfel, avem că

$$\mathbb{C} := \left\{ \begin{pmatrix} x & -y \\ y & x \end{pmatrix} \middle| \, x, y \in \mathbb{R} \right\}.$$

Acum avem identificarea $1 \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ şi $i = \sqrt{-1} \equiv \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, şi prin urmare numărul complex $z = x + iy$ se identifică cu matricea $\begin{pmatrix} x & -y \\ y & x \end{pmatrix}$.

Adunarea a două numere complexe $z_1 = \begin{pmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{pmatrix}$ şi $z_2 = \begin{pmatrix} x_2 & -y_2 \\ y_2 & x_2 \end{pmatrix}$ se defineşte ca fiind adunarea matricilor

$$z_1 + z_2 := \begin{pmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{pmatrix} + \begin{pmatrix} x_2 & -y_2 \\ y_2 & x_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 & -(y_1 + y_2) \\ y_1 + y_2 & x_1 + x_2 \end{pmatrix},$$

iar înmulţirea se defineşte ca fiind înmulţirea matricilor

$$z_1 \cdot z_2 := \begin{pmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{pmatrix} \begin{pmatrix} x_2 & -y_2 \\ y_2 & x_2 \end{pmatrix} = \begin{pmatrix} x_1 x_2 - y_1 y_2 & -(x_1 y_2 + x_2 y_1) \\ x_1 y_2 + x_2 y_1 & x_1 x_2 - y_1 y_2 \end{pmatrix}.$$

Inversul numărului complex $z = \begin{pmatrix} x & -y \\ y & x \end{pmatrix}$ este inversa matricii care îl reprezintă pe $z$

$$z^{-1} := \begin{pmatrix} x & -y \\ y & x \end{pmatrix}^{-1} = \frac{1}{x^2 + y^2} \begin{pmatrix} x & y \\ -y & x \end{pmatrix},$$

conjugatul lui $z$ este transpusa matricii $z$

$$\overline{z} := \begin{pmatrix} x & -y \\ y & x \end{pmatrix}^T = \begin{pmatrix} x & y \\ -y & x \end{pmatrix},$$

iar modulului lui $z$ este rădăcina pătrată a determinantului matricii $z$

$$|z| := \sqrt{\det \begin{pmatrix} x & -y \\ y & x \end{pmatrix}} = \sqrt{x^2 + y^2}.$$

Scriind în coordonate polare, avem că

$$\begin{pmatrix} x & -y \\ y & x \end{pmatrix} = r \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

Acum putem face o scurtă introducere în problematica funcţiilor definite pe mulţimea numerelor complexe. Fie $f : \mathbb{C} \to \mathbb{C}$ o funcţie complexă cu valori complexe. Pornind de la identificarea $\mathbb{C} \simeq \mathbb{R}^2$, funcţia $f$ se poate scrie sub forma

$$f(x + iy) = u(x, y) + iv(x, y),$$

unde $u, v : \mathbb{R}^2 \to \mathbb{R}$ sunt funcţii de două variabile reale cu valori reale.

*Limita funcţiei* $f$ în punctul $z_0$ este acel număr complex $L$ care satisface: pentru orice $\varepsilon > 0$, există $\delta > 0$, astfel încât, pentru orice $z \in \mathbb{C}$ care satisface $|z - z_0| < \delta$, avem că $|f(z) - L| < \varepsilon$. Notăm

$$\lim_{z \to z_o} f(z) = L.$$

Se poate observa imediat că

$$\lim_{z \to z_o} f(z) = \lim_{(x,y) \to (x_0,y_0)} u(x, y) + i \lim_{(x,y) \to (x_0,y_0)} v(x, y),$$

ceea ce ne arată că $f$ este continuă în $z_0$ (şi, mai general, pe $\mathbb{C}$) dacă şi numai dacă $u$ şi $v$ sunt continue în $(x_0, y_0)$ (sau, mai general, pe $\mathbb{R}^2$).

Putem acum defini *derivata complexă* a lui $f$ în punctul $z_0$ ca fiind

$$f'(z_0) = \lim_{z \to z_o} \frac{f(z) - f(z_0)}{z - z_0}.$$

Se pot defini *derivatele parţiale* ale lui $f$ în punctul $z_0$ prin

$$\frac{\partial f}{\partial x}(z_0) := \lim_{h \to 0} \frac{f(z_0 + h) - f(z_0)}{h},$$

$$\frac{\partial f}{\partial y}(z_0) := \lim_{h \to 0} \frac{f(z_0 + ih) - f(z_0)}{h}.$$

Observăm că dacă există $f'(z_0)$, atunci trebuie să avem

$$f'(z_0) = \frac{\partial f}{\partial x}(z_0) = -i \frac{\partial f}{\partial y}(z_0). \tag{2.1.1}$$

Pe de altă parte, putem scrie că

$$\begin{aligned}\frac{\partial f}{\partial x}(z_0) &= \lim_{h\to 0}\frac{u(x_0+h,y_0)-u(x_0,y_0)}{h}+i\lim_{h\to 0}\frac{v(x_0+h,y_0)-v(x_0,y_0)}{h}\\ &= \frac{\partial u}{\partial x}(x_0,y_0)+i\frac{\partial v}{\partial x}(x_0,y_0),\end{aligned}$$

şi analog

$$\begin{aligned}\frac{\partial f}{\partial y}(z_0) &= \lim_{h\to 0}\frac{u(x_0,y_0+h)-u(x_0,y_0)}{h}+i\lim_{h\to 0}\frac{v(x_0,y_0)-v(x_0,y_0+h)}{h}\\ &= \frac{\partial u}{\partial y}(x_0,y_0)+i\frac{\partial v}{\partial y}(x_0,y_0).\end{aligned}$$

Din relaţia (2.1.1), rezultă că, pentru ca $f'(z_0)$ să existe, trebuie ca

$$\frac{\partial u}{\partial x}(x_0,y_0)+i\frac{\partial v}{\partial x}(x_0,y_0)=-i\frac{\partial u}{\partial y}(x_0,y_0)+\frac{\partial v}{\partial y}(x_0,y_0),$$

adică

$$\begin{cases}\frac{\partial u}{\partial x}(x_0,y_0) = & \frac{\partial v}{\partial y}(x_0,y_0)\\ \frac{\partial v}{\partial x}(x_0,y_0) = & -\frac{\partial u}{\partial y}(x_0,y_0)\end{cases},$$

care poartă denumirea de *ecuaţiile Cauchy-Riemann*.

Dacă funcţia $f$ este derivabilă complex în fiecare punct $z_0$ al unei mulţimi deschise $U\subset\mathbb{C}$, se spune că $f$ este *olomorfă* pe $U$. Spunem că $f$ este olomorfă în punctul $z_0$ dacă este olomorfă pe o vecinătate a lui $z_0$. O funcţie olomorfă a cărui domeniu este întreg planul complex se numeşte *funcţie întreagă*.

Acum, Teorema lui Liouville ne spune că orice funcţie olomorfă pe $\mathbb{C}$ (întreagă) şi mărginită este constantă. Acest fapt va avea consecinţe importante în domeniul reţelelor neuronale, deoarece ideal am vrea ca funcţiile de activare să fie atât mărginite cât şi derivabile complex pe toată mulţimea numerelor complexe. Cum acest deziderat nu poate fi atins decât de funcţiile constante, care nu sunt acceptabile ca funcţii de activare, un compromis este necesar. Unii autori aleg funcţii de activare mărginite, dar care nu sunt derivabile complex, iar alţii aleg funcţii de activare derivabile complex, care însă au puncte în care sunt nemărginite.

O altă problemă ridicată de analiza complexă este aceea că funcţiile de eroare folosite în reţelele neuronale au valori reale. După cum se poate observa uşor, singurele funcţii $f:\mathbb{C}\to\mathbb{R}$ derivabile complex sunt constantele. Prin urmare, pentru deducerea algoritmilor de învăţare pentru reţelele neuronale cu valori complexe, nu vom putea utiliza derivata complexă a funcţiei de eroare, deoarece aceasta nu există. Singurele derivate care există pentru o astfel de funcţie sunt derivatele parţiale $\frac{\partial f}{\partial x}$ şi $\frac{\partial f}{\partial y}$, şi respectiv derivatele parţiale ale componentelor $u$ şi $v$ ale funcţiei $f$, adică $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial v}{\partial x}$, $\frac{\partial v}{\partial y}$.

Vom discuta acum reţelele neuronale cu valori complexe, şi vom începe cu neuronul complex, pe care îl vom compara cu un neuron real. Presupunem că avem o reţea neuronală cu două intrări reale $x_1$ şi $x_2$ şi două ieşiri reale $y_1$ şi $y_2$, fără straturi ascunse şi cu funcţia de activare $g:\mathbb{R}\to\mathbb{R}$, $g(x)=\frac{1}{1+e^{-x}}$. Avem că

$$y_1=\frac{1}{1+e^{-(w_{11}x_1+w_{12}x_2)}},$$

$$y_2 = \frac{1}{1 + e^{-(w_{21}x_1 + w_{22}x_2)}},$$

unde $(w_{ij})_{1 \leq i,j \leq 2}$ sunt ponderile reale care leagă neuronul $i$ din stratul $2$ cu neuronul $j$ din stratul $1$.

Presupunem acum că avem o reţea neuronală formată dintr-un singur neuron de intrare cu valori complexe $x$ şi un singur neuron de ieşire cu valori complexe $y$. Singura pondere complexă a reţelei este cea care leagă neuronul $x$ de neuronul $y$, pe care o vom nota cu $w$. Dacă alegem funcţia de activare $g : \mathbb{C} \to \mathbb{C}$, $g(x+iy) = \frac{1}{1+e^{-x}} + i\frac{1}{1+e^{-y}}$, avem că

$$y_1 = \frac{1}{1 + e^{-(w_1 x_1 - w_2 x_2)}},$$

$$y_2 = \frac{1}{1 + e^{-(w_2 x_1 + w_1 x_2)}},$$

ceea ce ne arată că reţeaua cu valori complexe, de fapt neuronul complex, este o particularizare a reţelei cu valori reale, pentru

$$w_{11} = w_1,\ w_{12} = -w_2,\ w_{21} = w_2,\ w_{22} = w_1,$$

ceea ce înseamnă că, dacă pentru reţeaua reală spaţiul de căutare al ponderilor este $\mathbb{R}^4$, pentru reţeaua complexă acest spaţiu se reduce la $\mathbb{R}^2$. Aceasta face ca reţelele cu valori complexe să nu aibă rezultate mai bune decât reţelele reale pentru orice aplicaţie, ci doar pentru acele aplicaţii care sunt formulate mai natural în numere complexe, şi pentru care căutarea într-un spaţiu mai mare al ponderilor nu este benefică din perspectiva performanţei. Practic, reţelele neuronale cu valori complexe se pretează acelor probleme care au o formulare complexă, şi care până la apariţia reţelelor complexe au fost rezolvate despărţindu-se părţile reală şi imaginară ale fiecărei intrări complexe, pierzând astfel legătura naturală care există între ele.

Metoda gradient a fost una dintre primele metode de învăţare extinse din domeniul real în domeniul complex, marcând practic naşterea domeniului reţelelor neuronale cu valori complexe. Articolele clasice, care practic simultan realizează această extindere sunt [71, 143, 37, 38, 163, 39, 42, 90, 113, 20, 43, 89, 184], publicate în anii 1990-1993. Toate împreună marchează punctul de pornire pentru acest nou domeniu de cercetare, care apoi se extinde atât din punct de vedere teoretic, cât şi din punctul de vedere al aplicaţiilor.

În cadrul teoriei, se demonstrează în [20, 21, 145, 147, 148] că perceptronul multistrat poate aproxima oricât de bine orice funcţie complexă, acesta fiind un rezultat important pentru reţelele neuronale cu valori reale care rămâne valabil şi în cazul reţelelor neuronale cu valori complexe. Au fost descoperite apoi caracteristici specifice perceptronului complex, cum ar fi aceea că frontierele de decizie ale neuronului complex se intersectează ortogonal, ceea ce dă, de fapt, puterea neuronului complex asupra neuronului real, a se vedea [187, 189, 192, 193, 195, 194].

În cele ce urmează, vom realiza o prezentare unitară a metodei gradient pentru reţelele de tip feedforward cu valori complexe, care nu ţine cont de faptul că funcţiile de activare sunt complexe complet (tratează numărul complex ca un întreg), sau complexe divizat (tratează separat părţile reală şi imaginară ale numărului complex). Particularizând rezultatele obţinute, regăsim formulele clasice deduse în literatura de specialitate pentru metoda gradient complexă divizat (spre exemplu în [143]) şi respectiv pentru metoda gradient complexă complet (spre exemplu în [144]).

Să presupunem că avem o reţea neuronală cu valori complexe de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este

stratul de ieşire, iar straturile notate cu $\{2,\dots,L-1\}$ sunt straturi ascunse. Funcţia de eroare $E:\mathbb{C}^N\to\mathbb{R}$ pentru o asemenea reţea este definită prin

$$E(\mathbf{w})=\frac{1}{2}\sum_{i=1}^{c}(y_i^L-t_i)(\overline{y_i^L-t_i}), \tag{2.1.2}$$

unde $\mathbf{y}^L=(y_i^L)_{1\le i\le c}$ reprezintă *ieşirile* reţelei, $\mathbf{t}=(t_i)_{1\le i\le c}$ reprezintă *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{w}$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei.

Dacă notăm cu $w_{jk}^l$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l\in\{2,\dots,L\}$, putem defini *pasul de actualizare* al ponderii $w_{jk}^l$ în epoca $t$ ca fiind

$$\Delta w_{jk}^l(t)=w_{jk}^l(t+1)-w_{jk}^l(t).$$

Cu această notaţie, pentru *metoda gradient*, *regula de actualizare* pentru ponderea $w_{jk}^l$ este

$$\Delta w_{jk}^l(t)=-\varepsilon\left(\frac{\partial E}{\partial w_{jk}^{l,R}}(t)+i\frac{\partial E}{\partial w_{jk}^{l,I}}(t)\right),$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi am notat cu $a^R$ partea reală a numărului complex $a$, şi cu $a^I$ partea imaginară a numărului complex $a$, i.e. $a=a^R+ia^I$, $i=\sqrt{-1}$.

Scris sub formă vectorială, pasul de actualizare devine

$$\Delta\mathbf{w}(t)=\mathbf{w}(t+1)-\mathbf{w}(t),$$

unde $\mathbf{w}\in\mathbb{C}^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient se poate scrie acum în formă vectorială, astfel:

$$\Delta\mathbf{w}(t)=-\varepsilon\nabla E(t),$$

unde $\nabla E(t):=\nabla E(\mathbf{w}(t))\in\mathbb{C}^N$, este gradientul funcţiei $E$, ale cărui componente $\frac{\partial E}{\partial w_{jk}^{l,R}}(\mathbf{w}(t))+i\frac{\partial E}{\partial w_{jk}^{l,I}}(\mathbf{w}(t))$ au fost scrise prescurtat mai sus sub forma $\frac{\partial E}{\partial w_{jk}^{l,R}}(t)+i\frac{\partial E}{\partial w_{jk}^{l,I}}(t)$. Prin urmare, pentru a minimiza funcţia de eroare $E$, avem nevoie să calculăm gradientul $\nabla E$, adică derivatele parţiale de forma $\frac{\partial E}{\partial w_{jk}^{l,R}}$, $\frac{\partial E}{\partial w_{jk}^{l,I}}$.

Pentru aceasta, facem următoarele notaţii

$$s_j^l:=\sum_k w_{jk}^l x_k^{l-1}+w_{j0}^l, \tag{2.1.3}$$

$$y_j^l:=G^l(s_j^l), \tag{2.1.4}$$

unde $G^l$ este *funcţia de activare* a stratului $l\in\{2,\dots,L\}$, $\mathbf{x}^1=(x_k^1)_{1\le k\le d}$ sunt intrările reţelei, şi avem că $x_k^l=y_k^l$, $\forall l\in\{2,\dots,L-1\}$, $\forall k$.

Vom calcula mai întâi actualizările pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta w_{jk}^L=-\varepsilon\left(\frac{\partial E}{\partial w_{jk}^{L,R}}+i\frac{\partial E}{\partial w_{jk}^{L,I}}\right).$$

Folosind regula înlănţuirii, putem scrie următoarele relaţii:

$$\frac{\partial E}{\partial w_{jk}^{L,R}} = \frac{\partial E}{\partial s_j^{L,R}} \frac{\partial s_j^{L,R}}{\partial w_{jk}^{L,R}} + \frac{\partial E}{\partial s_j^{L,I}} \frac{\partial s_j^{L,I}}{\partial w_{jk}^{L,R}},$$
$$\frac{\partial E}{\partial w_{jk}^{L,I}} = \frac{\partial E}{\partial s_j^{L,R}} \frac{\partial s_j^{L,R}}{\partial w_{jk}^{L,I}} + \frac{\partial E}{\partial s_j^{L,I}} \frac{\partial s_j^{L,I}}{\partial w_{jk}^{L,I}}.$$

Aplicând din nou regula înlănţuirii pentru $\partial E/\partial s_j^{L,R}$ şi $\partial E/\partial s_j^{L,I}$, avem că

$$\begin{aligned}\frac{\partial E}{\partial s_j^{L,R}} &= \frac{\partial E}{\partial y_j^{L,R}} \frac{\partial y_j^{L,R}}{\partial s_j^{L,R}} + \frac{\partial E}{\partial y_j^{L,I}} \frac{\partial y_j^{L,I}}{\partial s_j^{L,R}} \\ &= (y_j^{L,R} - t_j^R) \frac{\partial G^{L,R}(s_j^L)}{\partial s_j^{L,R}} \\ &\quad + (y_j^{L,I} - t_j^I) \frac{\partial G^{L,I}(s_j^L)}{\partial s_j^{L,R}},\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial s_j^{L,I}} &= \frac{\partial E}{\partial y_j^{L,R}} \frac{\partial y_j^{L,R}}{\partial s_j^{L,I}} + \frac{\partial E}{\partial y_j^{L,I}} \frac{\partial y_j^{L,I}}{\partial s_j^{L,I}} \\ &= (y_j^{L,R} - t_j^R) \frac{\partial G^{L,R}(s_j^L)}{\partial s_j^{L,I}} \\ &\quad + (y_j^{L,I} - t_j^I) \frac{\partial G^{L,I}(s_j^L)}{\partial s_j^{L,I}},\end{aligned}$$

unde am ţinut seama de notaţia (2.1.4) şi de expresia pentru funcţia de eroare (2.1.2). Din (2.1.3), deducem că

$$\frac{\partial s_j^{L,R}}{\partial w_{jk}^{L,R}} = x_k^{L-1,R}, \frac{\partial s_j^{L,R}}{\partial w_{jk}^{L,I}} = -x_k^{L-1,I},$$
$$\frac{\partial s_j^{L,I}}{\partial w_{jk}^{L,R}} = x_k^{L-1,I}, \frac{\partial s_j^{L,I}}{\partial w_{jk}^{L,I}} = x_k^{L-1,R},$$

şi folosind notaţia $\delta_j^L := \partial E/\partial s_j^L$, obţinem expresia pentru actualizarea dorită:

$$\Delta w_{jk}^L = -\varepsilon \delta_j^L \overline{x_k^{L-1}}.$$

Acum, vom calcula actualizarea pentru o pondere arbitrară $w_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, putem scrie că

$$\Delta w_{jk}^l = -\varepsilon \left( \frac{\partial E}{\partial w_{jk}^{l,R}} + i \frac{\partial E}{\partial w_{jk}^{l,I}} \right),$$

iar apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial w_{jk}^{l,R}} = \frac{\partial E}{\partial s_j^{l,R}}\frac{\partial s_j^{l,R}}{\partial w_{jk}^{l,R}} + \frac{\partial E}{\partial s_j^{l,I}}\frac{\partial s_j^{l,I}}{\partial w_{jk}^{l,R}},$$
$$\frac{\partial E}{\partial w_{jk}^{l,I}} = \frac{\partial E}{\partial s_j^{l,R}}\frac{\partial s_j^{l,R}}{\partial w_{jk}^{L,I}} + \frac{\partial E}{\partial s_j^{l,I}}\frac{\partial s_j^{l,I}}{\partial w_{jk}^{l,I}}.$$

Aplicând din nou regula înlănţuirii, obţinem că

$$\begin{aligned}\frac{\partial E}{\partial s_j^{l,R}} &= \sum_m \frac{\partial E}{\partial s_m^{l+1,R}}\frac{\partial s_m^{l+1,R}}{\partial s_j^{l,R}} \\ &\quad + \sum_m \frac{\partial E}{\partial s_m^{l+1,I}}\frac{\partial s_m^{l+1,I}}{\partial s_j^{l,R}},\end{aligned} \tag{2.1.5}$$

$$\begin{aligned}\frac{\partial E}{\partial s_j^{l,I}} &= \sum_m \frac{\partial E}{\partial s_m^{l+1,R}}\frac{\partial s_m^{l+1,R}}{\partial s_j^{l,I}} \\ &\quad + \sum_m \frac{\partial E}{\partial s_m^{l+1,I}}\frac{\partial s_m^{l+1,I}}{\partial s_j^{l,I}},\end{aligned} \tag{2.1.6}$$

unde sumele sunt luate după toţi neuronii $m$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite conexiuni. Apoi

$$\frac{\partial s_m^{l+1,R}}{\partial s_j^{l,R}} = \frac{\partial s_m^{l+1,R}}{\partial y_j^{l,R}}\frac{\partial y_j^{l,R}}{\partial s_j^{l,R}} + \frac{\partial s_m^{l+1,R}}{\partial y_j^{l,I}}\frac{\partial y_j^{l,I}}{\partial s_j^{l,R}},$$

şi relaţiile analoage. Din nou din (2.1.3), avem

$$\frac{\partial s_m^{l+1,R}}{\partial y_j^{l,R}} = w_{mj}^{l+1,R}, \frac{\partial s_m^{l+1,R}}{\partial y_j^{l,I}} = -w_{mj}^{l+1,I},$$
$$\frac{\partial s_m^{l+1,I}}{\partial y_j^{l,R}} = w_{mj}^{l+1,I}, \frac{\partial s_m^{l+1,I}}{\partial y_j^{l,I}} = w_{mj}^{l+1,R}.$$

Acum, punând toate cele de mai sus împreună, relaţiile (2.1.5) şi (2.1.6) devin

$$\begin{aligned}\frac{\partial E}{\partial s_j^{l,R}} &= \left(\sum_m \overline{w_{mj}^{l+1}}\delta_m^{l+1}\right)^R \frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} \\ &\quad + \left(\sum_m \overline{w_{mj}^{l+1}}\delta_m^{l+1}\right)^I \frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}},\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial s_j^{l,I}} &= \left(\sum_m \overline{w_{mj}^{l+1}}\delta_m^{l+1}\right)^R \frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}} \\ &\quad + \left(\sum_m \overline{w_{mj}^{l+1}}\delta_m^{l+1}\right)^I \frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}}.\end{aligned}$$

În fine, dacă notăm $\delta_j^l := \partial E/\partial s_j^l$, şi folosim (2.1.3) pentru a calcula

$$\frac{\partial s_j^{l,R}}{\partial w_{jk}^{l,R}} = x_k^{l-1,R}, \frac{\partial s_j^{l,R}}{\partial w_{jk}^{l,I}} = -x_k^{l-1,I},$$

$$\frac{\partial s_j^{l,I}}{\partial w_{jk}^{l,R}} = x_k^{l-1,I}, \frac{\partial s_j^{l,I}}{\partial w_{jk}^{l,I}} = x_k^{l-1,R},$$

obţinem

$$\Delta w_{jk}^l = -\varepsilon \delta_j^l \overline{x_k^{l-1}}, \forall l \in \{2, \ldots, L-1\}.$$

În concluzie, avem că

$$\Delta w_{jk}^l = -\varepsilon \delta_j^l \overline{x_k^{l-1}}, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left(\sum_m \overline{w_{mj}^{l+1}} \delta_m^{l+1}\right)^R \left(\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}}\right) \\ + \left(\sum_m \overline{w_{mj}^{l+1}} \delta_m^{l+1}\right)^I \left(\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}}\right), & l \leq L-1 \\ (y_j^{l,R} - t_j^R)\left(\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}}\right) \\ +(y_j^{l,I} - t_j^I)\left(\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}}\right), & l = L \end{cases}.$$

Observăm că aceste relaţii nu depind de tipul funcţiei de activare $G^l$: aceasta poate să fie *complexă complet* (tratează numărul complex ca un întreg) sau *complexă divizat* (tratează separat părţile reală şi imaginară ale numărului complex).

Spre exemplu, dacă $G^l(z) = \tanh z$, adică $G^l$ este complexă complet, avem că

$$\begin{aligned} \frac{\partial G^{l,R}(z)}{\partial z^R} + i\frac{\partial G^{l,R}(z)}{\partial z^I} &= \frac{\partial (\tanh z)^R}{\partial z^R} + i\frac{\partial (\tanh z)^R}{\partial z^I} \\ &= \frac{\partial}{\partial z^R}\left(\frac{\sinh 2z^R}{\cosh 2z^R + \cos 2z^I} + i\frac{\sin 2z^I}{\cosh 2z^R + \cos 2z^I}\right)^R \\ &\quad + i\frac{\partial}{\partial z^I}\left(\frac{\sinh 2z^R}{\cosh 2z^R + \cos 2z^I} + i\frac{\sin 2z^I}{\cosh 2z^R + \cos 2z^I}\right)^R \\ &= \frac{\partial}{\partial z^R}\left(\frac{\sinh 2z^R}{\cosh 2z^R + \cos 2z^I}\right) + i\frac{\partial}{\partial z^I}\left(\frac{\sinh 2z^R}{\cosh 2z^R + \cos 2z^I}\right) \\ &= \frac{2(1 + \cosh 2z^R \cos 2z^I)}{(\cosh 2z^R + \cos 2z^I)^2} + i\frac{2\sinh 2z^R \sin 2z^I}{(\cosh 2z^R + \cos 2z^I)^2}, \end{aligned}$$

şi analog pentru $\frac{\partial G^{l,I}(z)}{\partial z^R} + i\frac{\partial G^{l,I}(z)}{\partial z^I}$.

Dacă $G^l$ este complexă divizat, de exemplu $G^l(z) = G^l(z^R + iz^I) = \tanh z^R + i \tanh z^I$, atunci

$$\begin{aligned}\frac{\partial G^{l,R}(z)}{\partial z^R}+i\frac{\partial G^{l,R}(z)}{\partial z^I} &= \frac{\partial(\tanh z^R+i\tanh z^I)^R}{\partial z^R}+i\frac{\partial(\tanh z^R+i\tanh z^I)^R}{\partial z^I}\\ &= \frac{\partial(\tanh z^R)}{\partial z^R}+i\frac{\partial(0)}{\partial z^I}\\ &= 1-\tanh^2 z^R,\end{aligned}$$

şi analog pentru $\frac{\partial G^{l,I}(z)}{\partial z^R}+i\frac{\partial G^{l,I}(z)}{\partial z^I}$.

## 2.2 Reţele neuronale cu valori hiperbolice

Reţelele neuronale cu valori hiperbolice au fost introduse în [52], unde a fost discutată varianta feedforward a acestora. Apoi, diverse experimente au fost făcute utilizând acelaşi tip de reţele cu valori hiperbolice în teza de doctorat [48]. În [203], Nitta demonstrează că, spre deosebire de reţelele neuronale cu valori complexe, care au frontiere de decizie care se intersectează ortogonal, frontierele de decizie ale reţelelor cu valori hiperbolice se pot intersecta sub orice unghi, fapt care le oferă o flexibilitate sporită şi deschide calea unor aplicaţii pe viitor.

Reţelele cu valori hiperbolice de tip Hopfield au fost introduse de către Kuroe et al. în [156], iar apoi, mult mai detaliat, în [152]. În fine, Hitzer discută în [126] diverse posibilităţi pentru funcţiile de activare ale reţelelor cu valori hiperbolice de tip feedforward. Cercetarea în acest subdomeniu al reţelelor multidimensionale este încă la început, existând speranţe ca ele să poată fi aplicate în probleme pentru a da rezultate mai bune decât, spre exemplu, reţelele neuronale cu valori complexe, şi cu atât mai mult decât reţelele neuronale clasice cu valori reale.

Vom prezenta mulţimea numerelor hiperbolice privită din trei perspective. În primul rând perspectiva pur hiperbolică, în care mulţimea numerelor hiperbolice este definită ca fiind

$$\mathbb{B}:=\{x+uy|x,y\in\mathbb{R},u^2=1,u\neq\pm1\},$$

unde $u$ reprezintă unitatea imaginară hiperbolică.

Pentru orice $z_1=x_1+uy_1$, $z_2=x_2+uy_2\in\mathbb{B}$ definim *adunarea* prin

$$z_1+z_2:=(x_1+x_2)+u(y_1+y_2),$$

*înmulţirea* prin

$$z_1\cdot z_2:=(x_1x_2+y_1y_2)+u(x_1y_2+x_2y_1),$$

*inversul* lui $z_1$ prin

$$z_1^{-1}:=\frac{x_1}{x_1^2-y_1^2}-u\frac{y_1}{x_1^2-y_1^2},\ x_1\neq\pm y_1,$$

*conjugatul* lui $z_1$ prin

$$\overline{z_1}:=x_1-uy_1,$$

şi *modulul* lui $z_1$ prin

$$|z_1|:=\sqrt{|z_1\overline{z_1}|}=\sqrt{|x_1^2-y_1^2|}.$$

Un alt mod de a defini mulţimea numerelor hiperbolice este prin identificarea ei cu perechile ordonate de numere reale, astfel:

$$\mathbb{B}:=\{(x,y)|x,y\in\mathbb{R}\}.$$

Acum, pentru orice $z_1 = (x_1, y_1)$, $z_2 = (x_2, y_2) \in \mathbb{B}$, definim adunarea prin

$$z_1 + z_2 := (x_1 + x_2, y_1 + y_2),$$

înmulţirea prin

$$z_1 \cdot z_2 := (x_1x_2 + y_1y_2, x_1y_2 + x_2y_1),$$

şi inversul lui $z_1$ prin

$$z_1^{-1} := \left( \frac{x_1}{x_1^2 - y_1^2}, -\frac{y_1}{x_1^2 - y_1^2} \right), \ x_1 \neq \pm y_1.$$

Dacă facem identificarea $1 \equiv (1, 0)$ şi $u \equiv (0, 1)$ în acest caz, se poate vedea că avem $z = x + uy \equiv (x, y)$. Prin această identificare, fiecărui număr hiperbolic $z = x + uy$ îi corespunde punctul $(x, y)$ din plan, ceea ce constituie reprezentarea geometrică a numărului hiperbolic $z$.

O a treia definiţie echivalentă a mulţimii numerelor hiperbolice este scrierea ei ca o subalgebră a algebrei matricilor pătratice de ordinul doi cu elemente reale $\mathcal{M}_2(\mathbb{R})$. Astfel, avem că

$$\mathbb{B} := \left\{ \begin{pmatrix} x & y \\ y & x \end{pmatrix} \middle| \, x, y \in \mathbb{R} \right\}.$$

Acum avem identificarea $1 \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ şi $u \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, şi prin urmare numărul hiperbolic $z = x + uy$ se identifică cu matricea $\begin{pmatrix} x & y \\ y & x \end{pmatrix}$.

Adunarea a două numere hiperbolice $z_1 = \begin{pmatrix} x_1 & y_1 \\ y_1 & x_1 \end{pmatrix}$ şi $z_2 = \begin{pmatrix} x_2 & y_2 \\ y_2 & x_2 \end{pmatrix}$ se defineşte ca fiind adunarea matricilor

$$z_1 + z_2 := \begin{pmatrix} x_1 & y_1 \\ y_1 & x_1 \end{pmatrix} + \begin{pmatrix} x_2 & y_2 \\ y_2 & x_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 & y_1 + y_2 \\ y_1 + y_2 & x_1 + x_2 \end{pmatrix},$$

iar înmulţirea se defineşte ca fiind înmulţirea matricilor

$$z_1 \cdot z_2 := \begin{pmatrix} x_1 & y_1 \\ y_1 & x_1 \end{pmatrix} \begin{pmatrix} x_2 & y_2 \\ y_2 & x_2 \end{pmatrix} = \begin{pmatrix} x_1x_2 + y_1y_2 & x_1y_2 + x_2y_1 \\ x_1y_2 + x_2y_1 & x_1x_2 + y_1y_2 \end{pmatrix}.$$

Inversul numărului hiperbolic $z = \begin{pmatrix} x & y \\ y & x \end{pmatrix}$ este inversa matricii care îl reprezintă pe $z$

$$z^{-1} := \begin{pmatrix} x & y \\ y & x \end{pmatrix}^{-1} = \frac{1}{x^2 - y^2} \begin{pmatrix} x & -y \\ -y & x \end{pmatrix}, \ x \neq \pm y,$$

conjugatul lui $z$ este matricea dată prin

$$\overline{z} := \begin{pmatrix} x & -y \\ -y & x \end{pmatrix},$$

iar modulului lui $z$ este radicalul modulului determinantului matricii $z$

$$|z| := \sqrt{\left| \det \begin{pmatrix} x & y \\ y & x \end{pmatrix} \right|} = \sqrt{|x^2 - y^2|}.$$

Să presupunem acum că avem o reţea neuronală cu valori hiperbolice de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile $\{2,\dots,L-1\}$ sunt straturi ascunse. Funcţia de eroare $E:\mathbb{B}^N\to\mathbb{R}$ a acestei reţele este definită prin

$$E(\mathbf{w})=\frac{1}{2}\sum_{i=1}^{c}\left([y_i^L-t_i]_0^2+[y_i^L-t_i]_1^2\right), \tag{2.2.1}$$

unde $\mathbf{y}^L=(y_i^L)_{1\le i\le c}$ sunt *ieşirile* reţelei, $\mathbf{t}=(t_i)_{1\le i\le c}$ sunt *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{w}$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei.

Notând cu $w_{jk}^l$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l\in\{2,\dots,L\}$, putem defini *pasul de actualizare* al ponderii $w_{jk}^l$ în epoca $t$ ca fiind

$$\Delta w_{jk}^l(t)=w_{jk}^l(t+1)-w_{jk}^l(t).$$

Astfel, pentru *metoda gradient*, *regula de actualizare* pentru ponderea $w_{jk}^l$ este

$$\Delta w_{jk}^l(t)=-\varepsilon\left(\frac{\partial E}{\partial[w_{jk}^l]_0}(t)+u\frac{\partial E}{\partial[w_{jk}^l]_1}(t)\right),$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi am notat cu $[a]_0$ partea reală a numărului hiperbolic $a$, şi cu $[a]_1$ partea imaginară a numărului hiperbolic $a$, i.e. $a=[a]_0+u[a]_1$, $u^2=1$.

În scriere vectorială, pasul de actualizare devine

$$\Delta\mathbf{w}(t)=\mathbf{w}(t+1)-\mathbf{w}(t),$$

unde $\mathbf{w}\in\mathbb{B}^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient se poate scrie acum vectorial, astfel:

$$\Delta\mathbf{w}(t)=-\varepsilon\nabla E(t),$$

unde $\nabla E(t):=\nabla E(\mathbf{w}(t))\in\mathbb{B}^N$, este gradientul funcţiei $E$, ale cărui componente $\frac{\partial E}{\partial[w_{jk}^l]_0}(\mathbf{w}(t))+u\frac{\partial E}{\partial[w_{jk}^l]_1}(\mathbf{w}(t))$ au fost scrise prescurtat mai sus sub forma $\frac{\partial E}{\partial[w_{jk}^l]_0}(t)+u\frac{\partial E}{\partial[w_{jk}^l]_1}(t)$. Prin urmare, pentru a minimiza funcţia de eroare $E$, avem nevoie să calculăm gradientul $\nabla E$, adică derivatele parţiale de forma $\frac{\partial E}{\partial[w_{jk}^l]_0}$, $\frac{\partial E}{\partial[w_{jk}^l]_1}$.

Facem mai întâi următoarele notaţii

$$s_j^l:=\sum_k w_{jk}^l x_k^{l-1}+w_{j0}^l, \tag{2.2.2}$$

$$y_j^l:=G^l(s_j^l), \tag{2.2.3}$$

unde $G^l$ este *funcţia de activare* a stratului $l\in\{2,\dots,L\}$, $\mathbf{x}^1=(x_k^1)_{1\le k\le d}$ sunt intrările reţelei, şi avem că $x_k^l=y_k^l$, $\forall l\in\{2,\dots,L-1\}$, $\forall k$.

Vom calcula mai întâi actualizarea pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta w_{jk}^L=-\varepsilon\left(\frac{\partial E}{\partial[w_{jk}^L]_0}+u\frac{\partial E}{\partial[w_{jk}^L]_1}\right),$$

Folosind regula înlănţuirii, putem scrie:

$$\frac{\partial E}{\partial [w_{jk}^L]_0} = \frac{\partial E}{\partial [s_j^L]_0}\frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_0} + \frac{\partial E}{\partial [s_j^L]_1}\frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_0},$$
$$\frac{\partial E}{\partial [w_{jk}^L]_1} = \frac{\partial E}{\partial [s_j^L]_0}\frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_1} + \frac{\partial E}{\partial [s_j^L]_1}\frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_1}.$$

Aplicând din nou regula înlănţuirii pentru $\partial E/\partial [s_j^L]_0$ şi $\partial E/\partial [s_j^L]_1$, avem că

$$\begin{aligned}\frac{\partial E}{\partial [s_j^L]_0} &= \frac{\partial E}{\partial [y_j^L]_0}\frac{\partial [y_j^L]_0}{\partial [s_j^L]_0} + \frac{\partial E}{\partial [y_j^L]_1}\frac{\partial [y_j^L]_1}{\partial [s_j^L]_0} \\ &= [y_j^L - t_j]_0 \frac{\partial [G^L(s_j^L)]_0}{\partial [s_j^L]_0} \\ &\quad + [y_j^L - t_j]_1 \frac{\partial [G^L(s_j^L)]_1}{\partial [s_j^L]_0},\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial [s_j^L]_1} &= \frac{\partial E}{\partial [y_j^L]_0}\frac{\partial [y_j^L]_0}{\partial [s_j^L]_1} + \frac{\partial E}{\partial [y_j^L]_1}\frac{\partial [y_j^L]_1}{\partial [s_j^L]_1} \\ &= [y_j^L - t_j]_0 \frac{\partial [G^L(s_j^L)]_0}{\partial [s_j^L]_1} \\ &\quad + [y_j^L - t_j]_1 \frac{\partial [G^L(s_j^L)]_1}{\partial [s_j^L]_1},\end{aligned}$$

unde am ţinut seama de notaţia (2.2.3) şi de expresia pentru funcţia de eroare (2.2.1). Din (2.2.2), rezultă că

$$\frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_0} = [x_k^{L-1}]_0, \frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_1} = [x_k^{L-1}]_1,$$
$$\frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_0} = [x_k^{L-1}]_1, \frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_1} = [x_k^{L-1}]_0,$$

şi folosind notaţia $\delta_j^L := \partial E/\partial s_j^L$, obţinem expresia pentru actualizarea dorită:

$$\Delta w_{jk}^L = -\varepsilon \delta_j^L x_k^{L-1}.$$

Acum, vom calcula actualizarea pentru o pondere arbitrară $w_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, expresia actualizării este

$$\Delta w_{jk}^l = -\varepsilon \left( \frac{\partial E}{\partial [w_{jk}^l]_0} + u \frac{\partial E}{\partial [w_{jk}^l]_1} \right),$$

iar apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial [w_{jk}^l]_0} = \frac{\partial E}{\partial [s_j^l]_0}\frac{\partial [s_j^l]_0}{\partial [w_{jk}^l]_0} + \frac{\partial E}{\partial [s_j^l]_1}\frac{\partial [s_j^l]_1}{\partial [w_{jk}^l]_0},$$

$$\frac{\partial E}{\partial [w_{jk}^l]_1} = \frac{\partial E}{\partial [s_j^l]_0}\frac{\partial [s_j^l]_0}{\partial [w_{jk}^l]_1} + \frac{\partial E}{\partial [s_j^l]_1}\frac{\partial [s_j^l]_1}{\partial [w_{jk}^l]_1}.$$

Aplicând din nou regula înlănţuirii, putem scrie că

$$\begin{aligned}\frac{\partial E}{\partial [s_j^l]_0} &= \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_0}\frac{\partial [s_m^{l+1}]_0}{\partial [s_j^l]_0} \\ &\quad + \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_1}\frac{\partial [s_m^{l+1}]_1}{\partial [s_j^l]_0},\end{aligned} \tag{2.2.4}$$

$$\begin{aligned}\frac{\partial E}{\partial [s_j^l]_1} &= \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_0}\frac{\partial [s_m^{l+1}]_0}{\partial [s_j^l]_1} \\ &\quad + \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_1}\frac{\partial [s_m^{l+1}]_1}{\partial [s_j^l]_1},\end{aligned} \tag{2.2.5}$$

unde sumele sunt luate după toţi neuronii $m$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite conexiuni. Apoi

$$\frac{\partial [s_m^{l+1}]_0}{\partial [s_j^l]_0} = \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_0}\frac{\partial [y_j^l]_0}{\partial [s_j^l]_0} + \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_1}\frac{\partial [y_j^l]_1}{\partial [s_j^l]_0},$$

şi relaţiile analoage. Din nou din (2.2.2), avem

$$\frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_0} = [w_{mj}^{l+1}]_0, \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_1} = [w_{mj}^{l+1}]_1,$$

$$\frac{\partial [s_m^{l+1}]_1}{\partial [y_j^l]_0} = [w_{mj}^{l+1}]_1, \frac{\partial [s_m^{l+1}]_1}{\partial [y_j^l]_1} = [w_{mj}^{l+1}]_0.$$

Acum, punând toate cele de mai sus împreună, relaţiile (2.2.4) şi (2.2.5) devin

$$\begin{aligned}\frac{\partial E}{\partial [s_j^l]_0} &= \left[\sum_m w_{mj}^{l+1}\delta_m^{l+1}\right]_0 \frac{\partial [G^l(s_j^l)]_0}{\partial [s_j^l]_0} \\ &\quad + \left[\sum_m w_{mj}^{l+1}\delta_m^{l+1}\right]_1 \frac{\partial [G^l(s_j^l)]_1}{\partial [s_j^l]_0},\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial [s_j^l]_1} &= \left[\sum_m w_{mj}^{l+1}\delta_m^{l+1}\right]_0 \frac{\partial [G^l(s_j^l)]_0}{\partial [s_j^l]_1} \\ &\quad + \left[\sum_m w_{mj}^{l+1}\delta_m^{l+1}\right]_1 \frac{\partial [G^l(s_j^l)]_1}{\partial [s_j^l]_1}.\end{aligned}$$

În fine, dacă notăm $\delta_j^l := \partial E/\partial s_j^l$, şi folosim (2.2.2) pentru a calcula

$$\frac{\partial [s_j^l]_0}{\partial [w_{jk}^l]_0} = [x_k^{l-1}]_0, \frac{\partial [s_j^l]_0}{\partial [w_{jk}^l]_1} = [x_k^{l-1}]_1,$$
$$\frac{\partial [s_j^l]_1}{\partial [w_{jk}^l]_0} = [x_k^{l-1}]_1, \frac{\partial [s_j^l]_1}{\partial [w_{jk}^l]_1} = [x_k^{l-1}]_0,$$

obţinem

$$\Delta w_{jk}^l = -\varepsilon \delta_j^l x_k^{l-1}, \forall l \in \{2, \ldots, L-1\}.$$

În concluzie, avem că

$$\Delta w_{jk}^l = -\varepsilon \delta_j^l x_k^{l-1}, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left[\sum_m w_{mj}^{l+1} \delta_m^{l+1}\right]_0 \left(\frac{\partial [G^l(s_j^l)]_0}{\partial [s_j^l]_0} + u \frac{\partial [G^l(s_j^l)]_0}{\partial [s_j^l]_1}\right) \\ + \left[\sum_m w_{mj}^{l+1} \delta_m^{l+1}\right]_1 \left(\frac{\partial [G^l(s_j^l)]_1}{\partial [s_j^l]_0} + u \frac{\partial [G^l(s_j^l)]_1}{\partial [s_j^l]_1}\right), & l \leq L-1 \\ [y_j^L - t_j]_0 \left(\frac{\partial [G^l(s_j^l)]_0}{\partial [s_j^l]_0} + u \frac{\partial [G^l(s_j^l)]_0}{\partial [s_j^l]_1}\right) \\ + [y_j^L - t_j]_1 \left(\frac{\partial [G^l(s_j^l)]_1}{\partial [s_j^l]_0} + u \frac{\partial [G^l(s_j^l)]_1}{\partial [s_j^l]_1}\right), & l = L \end{cases}.$$

## 2.3 Reţele neuronale cu valori cuaternionice

Reţelele neuronale cu valori cuaternionice au fost introduse în [19] şi [190], în primul rând ca o generalizare în $4$ dimensiuni a reţelelor cu valori complexe. Cartea [18] este una dintre primele în care aceste reţele apar, cu aplicaţii în predicţia seriilor de timp haotice şi în robotică. Capabilităţile acestor reţele pentru predicţia seriilor de timp haotice şi a aproximării de funcţii cu valori cuaternionice au fost discutate în [16], şi respectiv [17]. Un tip nou de reţea neuronală cu valori cuaternionice, numită reţea neuronală spinor, a fost introdusă în [53], bazat pe faptul că algebra cuaternionilor nu este comutativă asemenea algebrelor numerelor complexe şi hiperbolice. Isokawa et al. arată în [131] că reţelele neuronale feedforward cu valori cuaternionice obţin corect transformări geometrice pentru o problemă de compresie de imagini color, pe când o reţea neuronală cu valori reale eşuează. Apoi, Nitta demonstrează în [198] că un singur neuron cuaternionic poate învăţa problema parităţii pe $4$ biţi, care altfel necesită o reţea neuronală reală cu cel puţin un strat ascuns pentru a fi învăţată. Tot ca o aplicaţie a acestor reţele, se arată în [157] faptul că o astfel de reţea poate fi folosită pentru a extrage informaţii de culoare dintr-o imagine întunecată, care poate fi folosită pentru aplicaţii în care este necesară vederea nocturnă color. Datorită arhitecturii reţelei, la intrare i se dă imaginea întunecată, iar la ieşire i se dă imaginea color normală, iar reţeaua este capabilă să extragă informaţia de culoare din imaginea întunecată.

În ultimii ani însă, cele mai multe aplicaţii ale reţelelor neuronale cu valori cuaternionice au fost în procesarea semnalelor. Astfel, în [132], Isokawa et al. propune pentru prima dată o memorie asociativă cu valori cuaternionice. Buchholz & Le Bihan discută în [50] clasificarea semnalelor polarizate folosind reţele de tip feedforward cu valori cuaternionice. Apoi, Took & Mandic [251] şi respectiv Took et al. [255] propun algoritmul celor mai mici pătrate (LMS) cu valori cuaternionice, care este practic

o reţea neuronală. Urmează reţelele neuronale feedforward cu valori cuaternionice care sunt propuse şi folosite în [253] la aplicaţii din domeniul procesării de semnale. O problemă practică unde aceste reţele au fost folosite cu succes este aceea a predicţiei direcţiei şi vitezei vântului, în trei dimensiuni de data aceasta, spre deosebire de problema similară în două dimensiuni care a fost tratată folosind reţele cu valori complexe, a se vedea, spre exemplu [135, 254, 256]. Un nou algoritm de învăţare pentru reţele de tip feedforward cu valori cuaternionice a fost propus în [252], şi un nou tip de reţea, asemănător celor care folosesc funcţii de activare complexe divizat, în [65]. S-a trecut apoi la reţele recurente cu valori cuaternionice, folosite tot pentru probleme din domeniul procesării de semnale, de exemplu în [136]. Che Ujang et al. fac în [66, 67] o sinteză a tot ce se cunoaşte până în acest moment despre reţelele cuaternionice folosite în aplicaţii din domeniul procesării de semnale.

Dovada faptului că cercetarea în domeniu este abia la început este aceea că articole foarte recente introduc pentru prima oară anumite tipuri de reţele cu valori cuaternionice, care au deja aplicaţii practice. Astfel, filtrele Kalman sunt introduse în [134], un nou tip de reţea Hopfield cu valori continue este introdusă în [260], reţele cuaternionice cu auto-organizare sunt folosite în [138] pentru modelarea articulaţiilor anatomice, reţele cuaternionice neuro-fuzzy au rezultate promiţătoare pe probleme de clasificare cu valori reale [110] şi reţelele de tip echo state cu valori cuaternionice apar pentru prima dată în [265]. Vechimea de sub un an a tuturor acestor articole dovedeşte faptul că teoria reţelelor cu valori cuaternionice abia acum începe să se contureze, şi se întrevede o dezvoltare asemănătoare cu a teoriei reţelelor cu valori complexe, în următorii ani.

Vom face o prezentare a mulţimii numerelor cuaternionice din mai multe perspective. În primul rând perspectiva pur cuaternionică, în care mulţimea numerelor cuaternionice este definită ca fiind

$$\mathbb{H} := \{x + iy + jz + kt | x, y, z, t \in \mathbb{R}, i^2 = j^2 = k^2 = ijk = -1\},$$

unde $i, j, k$ reprezintă unităţile imaginare cuaternionice.

Pentru orice $q_1 = x_1 + iy_1 + jz_1 + kt_1$, $q_2 = x_2 + iy_2 + jz_2 + kt_2 \in \mathbb{H}$ definim *adunarea* prin

$$q_1 + q_2 := (x_1 + x_2) + i(y_1 + y_2) + j(z_1 + z_2) + k(t_1 + t_2),$$

*înmulţirea* prin

$$\begin{aligned} q_1 \cdot q_2 &:= (x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2) + i(x_1y_2 + x_1y_2 + z_1t_2 - t_1z_2) \\ &+ j(x_1z_2 + x_1z_2 - y_1t_2 + t_1y_2) + k(x_1t_2 + x_1t_2 + y_1z_2 - z_1y_2) \end{aligned}$$

*inversul* lui $q_1$ prin

$$q_1^{-1} := \frac{x_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2} - i\frac{y_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2} - j\frac{z_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2} - k\frac{t_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2},$$

*conjugatul* lui $q_1$ prin

$$\overline{q_1} := x_1 - iy_1 - jz_1 - kt_1,$$

şi *modulul* lui $q_1$ prin

$$|q_1| := \sqrt{q_1\overline{q_1}} = \sqrt{x_1^2 + y_1^2 + z_1^2 + t_1^2}.$$

Un alt mod de a defini mulţimea numerelor cuaternionice este prin identificarea ei cu perechile ordonate de numere reale, astfel

$$\mathbb{H} := \{(x, y, z, t) | x, y, z, t \in \mathbb{R}\}.$$

Acum, pentru orice $q_1 = (x_1, y_1, z_1, t_1)$, $q_2 = (x_2, y_2, z_2, t_2) \in \mathbb{H}$, adunarea este dată prin

$$q_1 + q_2 := (x_1 + x_2, y_1 + y_2, z_1 + z_2, t_1 + t_2),$$

înmulţirea prin

$$\begin{aligned} q_1 \cdot q_2 \quad := \quad & (x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2, x_1y_2 + x_1y_2 + z_1t_2 - t_1z_2, \\ & x_1z_2 + x_1z_2 - y_1t_2 + t_1y_2, x_1t_2 + x_1t_2 + y_1z_2 - z_1y_2) \end{aligned}$$

şi inversul lui $q_1$ prin

$$q_1^{-1} := \left( \frac{x_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2}, \frac{-y_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2}, \frac{-z_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2}, \frac{-t_1}{x_1^2 + y_1^2 + z_1^2 + t_1^2} \right).$$

În acest caz se face identificarea $1 \equiv (1,0,0,0)$, $i \equiv (0,1,0,0)$, $j \equiv (0,0,1,0)$, $k \equiv (0,0,0,1)$, de unde este clar că avem $q = x + iy + jz + kt \equiv (x, y, z, t)$. Prin această identificare, fiecărui număr cuaternionic $q = x + iy + jz + kt$ îi corespunde punctul $(x, y, z, t)$ din spaţiul cvadridimensional, ceea ce constituie reprezentarea geometrică a numărului cuaternionic $q$.

O a treia reprezentare posibilă a mulţimii numerelor cuaternionice este ca o subalgebră a algebrei matricilor pătratice de ordinul patru cu elemente reale, notată $\mathcal{M}_4(\mathbb{R})$. Astfel, avem că

$$\mathbb{H} := \left\{ \begin{pmatrix} x & y & z & t \\ -y & x & -t & z \\ -z & t & x & -y \\ -t & -z & y & x \end{pmatrix} \middle| \, x, y, z, t \in \mathbb{R} \right\}.$$

Acum avem identificarea

$$1 \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; i \equiv \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$j \equiv \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \; k \equiv \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

şi prin urmare numărul cuaternionic $q = x + iy + jz + kt$ se identifică cu matricea $\begin{pmatrix} x & y & z & t \\ -y & x & -t & z \\ -z & t & x & -y \\ -t & -z & y & x \end{pmatrix}$.

Adunarea a două numere cuaternionice $q_1 = \begin{pmatrix} x_1 & y_1 & z_1 & t_1 \\ -y_1 & x_1 & -t_1 & z_1 \\ -z_1 & t_1 & x_1 & -y_1 \\ -t_1 & -z_1 & y_1 & x_1 \end{pmatrix}$ şi $q_2 =$

$\begin{pmatrix} x_2 & y_2 & z_2 & t_2 \\ -y_2 & x_2 & -t_2 & z_2 \\ -z_2 & t_2 & x_2 & -y_2 \\ -t_2 & -z_2 & y_2 & x_2 \end{pmatrix}$ se defineşte ca fiind adunarea matricilor

$$\begin{aligned} q_1 + q_2 &:= \begin{pmatrix} x_1 & y_1 & z_1 & t_1 \\ -y_1 & x_1 & -t_1 & z_1 \\ -z_1 & t_1 & x_1 & -y_1 \\ -t_1 & -z_1 & y_1 & x_1 \end{pmatrix} + \begin{pmatrix} x_2 & y_2 & z_2 & t_2 \\ -y_2 & x_2 & -t_2 & z_2 \\ -z_2 & t_2 & x_2 & -y_2 \\ -t_2 & -z_2 & y_2 & x_2 \end{pmatrix} \\ &= \begin{pmatrix} x_1 + x_2 & y_1 + y_2 & z_1 + z_2 & t_1 + t_2 \\ -(y_1 + y_2) & x_1 + x_2 & -(t_1 + t_2) & z_1 + z_2 \\ -(z_1 + z_2) & t_1 + t_2 & x_1 + x_2 & -(y_1 + y_2) \\ -(t_1 + t_2) & -(z_1 + z_2) & y_1 + y_2 & x_1 + x_2 \end{pmatrix}, \end{aligned}$$

iar înmulţirea este dată de înmulţirea matricilor

$$\begin{aligned} q_1 \cdot q_2 &:= \begin{pmatrix} x_1 & y_1 & z_1 & t_1 \\ -y_1 & x_1 & -t_1 & z_1 \\ -z_1 & t_1 & x_1 & -y_1 \\ -t_1 & -z_1 & y_1 & x_1 \end{pmatrix} \begin{pmatrix} x_2 & y_2 & z_2 & t_2 \\ -y_2 & x_2 & -t_2 & z_2 \\ -z_2 & t_2 & x_2 & -y_2 \\ -t_2 & -z_2 & y_2 & x_2 \end{pmatrix} \\ &\begin{pmatrix} x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2 & x_1y_2 + x_1y_2 + z_1t_2 - t_1z_2 \\ -(x_1y_2 + x_1y_2 + z_1t_2 - t_1z_2) & x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2 \\ -(x_1z_2 + x_1z_2 - y_1t_2 + t_1y_2) & x_1t_2 + x_1t_2 + y_1z_2 - z_1y_2 \\ -(x_1t_2 + x_1t_2 + y_1z_2 - z_1y_2) & -(x_1z_2 + x_1z_2 - y_1t_2 + t_1y_2) \end{pmatrix} \\ &\begin{pmatrix} x_1z_2 + x_1z_2 - y_1t_2 + t_1y_2 & x_1t_2 + x_1t_2 + y_1z_2 - z_1y_2 \\ -(x_1t_2 + x_1t_2 + y_1z_2 - z_1y_2) & x_1z_2 + x_1z_2 - y_1t_2 + t_1y_2 \\ x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2 & -(x_1y_2 + x_1y_2 + z_1t_2 - t_1z_2) \\ x_1y_2 + x_1y_2 + z_1t_2 - t_1z_2 & x_1x_2 - y_1y_2 - z_1z_2 - t_1t_2 \end{pmatrix}. \end{aligned}$$

Inversul numărului cuaternionic $q = \begin{pmatrix} x & y & z & t \\ -y & x & -t & z \\ -z & t & x & -y \\ -t & -z & y & x \end{pmatrix}$ este inversa matricii care îl reprezintă pe $q$

$$q^{-1} := \begin{pmatrix} x & y & z & t \\ -y & x & -t & z \\ -z & t & x & -y \\ -t & -z & y & x \end{pmatrix}^{-1} = \frac{1}{x^2 + y^2 + z^2 + t^2} \begin{pmatrix} x & -y & -z & -t \\ y & x & t & -z \\ z & -t & x & y \\ t & z & -y & x \end{pmatrix},$$

conjugatul lui $q$ este transpusa matricii $q$

$$\overline{q} := \begin{pmatrix} x & y & z & t \\ -y & x & -t & z \\ -z & t & x & -y \\ -t & -z & y & x \end{pmatrix}^T = \begin{pmatrix} x & -y & -z & -t \\ y & x & t & -z \\ z & -t & x & y \\ t & z & -y & x \end{pmatrix},$$

iar modului lui $q$ este radicalul de ordinul $4$ al determinantului matricii $q$

$$|q| := \sqrt[4]{\det \begin{pmatrix} x & y & z & t \\ -y & x & -t & z \\ -z & t & x & -y \\ -t & -z & y & x \end{pmatrix}} = \sqrt{x^2 + y^2 + z^2 + t^2}.$$

Să presupunem acum că avem o reţea neuronală cu valori cuaternionice de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile numerotate $\{2, \ldots, L-1\}$ sunt straturi ascunse. Funcţia de eroare $E : \mathbb{H}^N \to \mathbb{R}$ pentru o atare reţea este dată prin

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{c}(y_i^L - t_i)(\overline{y_i^L - t_i}), \tag{2.3.1}$$

unde $\mathbf{y}^L = (y_i^L)_{1\leq i\leq c}$ reprezintă *ieşirile* reţelei, $\mathbf{t} = (t_i)_{1\leq i\leq c}$ reprezintă *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{w}$ reprezintă vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei.

Dacă notăm cu $w_{jk}^l$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l \in \{2, \ldots, L\}$, putem defini *pasul de actualizare* al ponderii $w_{jk}^l$ în epoca $t$ ca fiind

$$\Delta w_{jk}^l(t) = w_{jk}^l(t+1) - w_{jk}^l(t).$$

*Regula de actualizare* pentru ponderea $w_{jk}^l$ este, pentru *metoda gradient*, dată de expresia

$$\Delta w_{jk}^l(t) = -\varepsilon\left(\frac{\partial E}{\partial [w_{jk}^l]_0}(t) + i\frac{\partial E}{\partial [w_{jk}^l]_1}(t) + j\frac{\partial E}{\partial [w_{jk}^l]_2}(t) + k\frac{\partial E}{\partial [w_{jk}^l]_3}(t)\right),$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi am notat $a = [a]_0 + i[a]_1 + j[a]_2 + k[a]_3$, $i^2 = j^2 = k^2 = ijk = -1$.

Scris sub formă vectorială, pasul de actualizare devine

$$\Delta \mathbf{w}(t) = \mathbf{w}(t+1) - \mathbf{w}(t),$$

unde $\mathbf{w} \in \mathbb{H}^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient se scrie în formă vectorială, astfel:

$$\Delta \mathbf{w}(t) = -\varepsilon \nabla E(t),$$

unde $\nabla E(t) := \nabla E(\mathbf{w}(t)) \in \mathbb{H}^N$, este gradientul funcţiei $E$, ale cărui componente $\frac{\partial E}{\partial [w_{jk}^l]_0}(\mathbf{w}(t)) + i\frac{\partial E}{\partial [w_{jk}^l]_1}(\mathbf{w}(t)) + j\frac{\partial E}{\partial [w_{jk}^l]_2}(\mathbf{w}(t)) + k\frac{\partial E}{\partial [w_{jk}^l]_3}(\mathbf{w}(t))$ au fost scrise prescurtat mai sus sub forma $\frac{\partial E}{\partial [w_{jk}^l]_0}(t) + i\frac{\partial E}{\partial [w_{jk}^l]_1}(t) + j\frac{\partial E}{\partial [w_{jk}^l]_2}(t) + k\frac{\partial E}{\partial [w_{jk}^l]_3}(t)$. Prin urmare, pentru a calcula actualizările ponderilor, avem nevoie să calculăm gradientul $\nabla E$, adică derivatele parţiale de forma $\frac{\partial E}{\partial [w_{jk}^l]_0}$, $\frac{\partial E}{\partial [w_{jk}^l]_1}$, $\frac{\partial E}{\partial [w_{jk}^l]_2}$, $\frac{\partial E}{\partial [w_{jk}^l]_3}$.

Pentru aceasta, vom face următoarele notaţii

$$s_j^l := \sum_k w_{jk}^l x_k^{l-1} + w_{j0}^l, \tag{2.3.2}$$

$$y_j^l := G^l(s_j^l), \tag{2.3.3}$$

unde $G^l$ este *funcţia de activare* a stratului $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1\leq k\leq d}$ sunt intrările reţelei, şi avem că $x_k^l = y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$.

Începem cu actualizarea pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta w_{jk}^L = -\varepsilon\left(\frac{\partial E}{\partial [w_{jk}^L]_0} + i\frac{\partial E}{\partial [w_{jk}^L]_1} + j\frac{\partial E}{\partial [w_{jk}^L]_2} + k\frac{\partial E}{\partial [w_{jk}^L]_3}\right).$$

Din regula înlănţuirii, putem scrie următoarele relaţii:

$$\frac{\partial E}{\partial [w_{jk}^L]_0} = \frac{\partial E}{\partial [s_j^L]_0}\frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_0} + \frac{\partial E}{\partial [s_j^L]_1}\frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_0} + \frac{\partial E}{\partial [s_j^L]_2}\frac{\partial [s_j^L]_2}{\partial [w_{jk}^L]_0} + \frac{\partial E}{\partial [s_j^L]_3}\frac{\partial [s_j^L]_3}{\partial [w_{jk}^L]_0},$$

$$\frac{\partial E}{\partial [w_{jk}^L]_1} = \frac{\partial E}{\partial [s_j^L]_0}\frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_1} + \frac{\partial E}{\partial [s_j^L]_1}\frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_1} + \frac{\partial E}{\partial [s_j^L]_2}\frac{\partial [s_j^L]_2}{\partial [w_{jk}^L]_1} + \frac{\partial E}{\partial [s_j^L]_3}\frac{\partial [s_j^L]_3}{\partial [w_{jk}^L]_1},$$

şi analoagele. Aplicând aceeaşi regulă din nou pentru $\partial E/\partial [s_j^L]_0$, $\partial E/\partial [s_j^L]_1$, $\partial E/\partial [s_j^L]_2$ şi $\partial E/\partial [s_j^L]_3$, avem că

$$\begin{aligned}
\frac{\partial E}{\partial [s_j^L]_0} &= \frac{\partial E}{\partial [y_j^L]_0}\frac{\partial [y_j^L]_0}{\partial [s_j^L]_0} + \frac{\partial E}{\partial [y_j^L]_1}\frac{\partial [y_j^L]_1}{\partial [s_j^L]_0} + \frac{\partial E}{\partial [y_j^L]_2}\frac{\partial [y_j^L]_2}{\partial [s_j^L]_0} + \frac{\partial E}{\partial [y_j^L]_3}\frac{\partial [y_j^L]_3}{\partial [s_j^L]_0} \\
&= [y_j^L - t_j]_0\frac{\partial [G^L(s_j^L)]_0}{\partial [s_j^L]_0} + [y_j^L - t_j]_1\frac{\partial [G^L(s_j^L)]_1}{\partial [s_j^L]_0} \\
&+ [y_j^L - t_j]_2\frac{\partial [G^L(s_j^L)]_2}{\partial [s_j^L]_0} + [y_j^L - t_j]_3\frac{\partial [G^L(s_j^L)]_3}{\partial [s_j^L]_0},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial [s_j^L]_1} &= \frac{\partial E}{\partial [y_j^L]_0}\frac{\partial [y_j^L]_0}{\partial [s_j^L]_1} + \frac{\partial E}{\partial [y_j^L]_1}\frac{\partial [y_j^L]_1}{\partial [s_j^L]_1} + \frac{\partial E}{\partial [y_j^L]_2}\frac{\partial [y_j^L]_2}{\partial [s_j^L]_1} + \frac{\partial E}{\partial [y_j^L]_3}\frac{\partial [y_j^L]_3}{\partial [s_j^L]_1} \\
&= [y_j^L - t_j]_0\frac{\partial [G^L(s_j^L)]_0}{\partial [s_j^L]_1} + [y_j^L - t_j]_1\frac{\partial [G^L(s_j^L)]_1}{\partial [s_j^L]_1} \\
&+ [y_j^L - t_j]_2\frac{\partial [G^L(s_j^L)]_0}{\partial [s_j^L]_1} + [y_j^L - t_j]_3\frac{\partial [G^L(s_j^L)]_1}{\partial [s_j^L]_1},
\end{aligned}$$

şi cele analoage, unde am ţinut seama de notaţia (2.3.3) şi de expresia pentru funcţia de eroare (2.3.1). Din (2.3.2) şi

$$\begin{aligned}
w_{jk}^L x_k^{L-1} &= ([w_{jk}^L]_0[x_k^{L-1}]_0 - [w_{jk}^L]_1[x_k^{L-1}]_1 - [w_{jk}^L]_2[x_k^{L-1}]_2 - [w_{jk}^L]_3[x_k^{L-1}]_3) \\
&+ i([w_{jk}^L]_0[x_k^{L-1}]_1 + [w_{jk}^L]_1[x_k^{L-1}]_0 + [w_{jk}^L]_2[x_k^{L-1}]_3 - [w_{jk}^L]_3[x_k^{L-1}]_2) \\
&+ j([w_{jk}^L]_0[x_k^{L-1}]_2 + [w_{jk}^L]_2[x_k^{L-1}]_0 - [w_{jk}^L]_1[x_k^{L-1}]_3 + [w_{jk}^L]_3[x_k^{L-1}]_1) \\
&+ k([w_{jk}^L]_0[x_k^{L-1}]_3 + [w_{jk}^L]_3[x_k^{L-1}]_0 + [w_{jk}^L]_1[x_k^{L-1}]_2 - [w_{jk}^L]_2[x_k^{L-1}]_1),
\end{aligned}$$

rezultă că

$$\frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_0} = [x_k^{L-1}]_0, \frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_1} = -[x_k^{L-1}]_1, \frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_2} = -[x_k^{L-1}]_2, \frac{\partial [s_j^L]_0}{\partial [w_{jk}^L]_3} = -[x_k^{L-1}]_3,$$

$$\frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_0} = [x_k^{L-1}]_1, \frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_1} = [x_k^{L-1}]_0, \frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_2} = [x_k^{L-1}]_3, \frac{\partial [s_j^L]_1}{\partial [w_{jk}^L]_3} = -[x_k^{L-1}]_2,$$

şi relaţiile analoage, şi folosind notaţia $\delta_j^L := \partial E/\partial s_j^L$, obţinem expresia pentru actualizarea dorită:

$$\Delta w_{jk}^L = -\varepsilon \delta_j^L \overline{x_k^{L-1}}.$$

Acum, trecem la actualizarea pentru o pondere arbitrară $w_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, aceasta este

$$\Delta w_{jk}^l = -\varepsilon \left( \frac{\partial E}{\partial [w_{jk}^l]_0} + i \frac{\partial E}{\partial [w_{jk}^l]_1} + j \frac{\partial E}{\partial [w_{jk}^l]_2} + k \frac{\partial E}{\partial [w_{jk}^l]_3} \right),$$

iar apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial [w_{jk}^l]_0} = \frac{\partial E}{\partial [s_j^l]_0} \frac{\partial [s_j^l]_0}{\partial [w_{jk}^l]_0} + \frac{\partial E}{\partial [s_j^l]_1} \frac{\partial [s_j^l]_1}{\partial [w_{jk}^l]_0} + \frac{\partial E}{\partial [s_j^l]_2} \frac{\partial [s_j^l]_2}{\partial [w_{jk}^l]_0} + \frac{\partial E}{\partial [s_j^l]_3} \frac{\partial [s_j^l]_3}{\partial [w_{jk}^l]_0},$$

$$\frac{\partial E}{\partial [w_{jk}^l]_1} = \frac{\partial E}{\partial [s_j^l]_0} \frac{\partial [s_j^l]_0}{\partial [w_{jk}^l]_1} + \frac{\partial E}{\partial [s_j^l]_1} \frac{\partial [s_j^l]_1}{\partial [w_{jk}^l]_1} + \frac{\partial E}{\partial [s_j^l]_2} \frac{\partial [s_j^l]_2}{\partial [w_{jk}^l]_1} + \frac{\partial E}{\partial [s_j^l]_3} \frac{\partial [s_j^l]_3}{\partial [w_{jk}^l]_1},$$

şi cele analoage.

Din nou regula înlănţuirii ne dă

$$\begin{aligned} \frac{\partial E}{\partial [s_j^l]_0} &= \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_0} \frac{\partial [s_m^{l+1}]_0}{\partial [s_j^l]_0} + \frac{\partial E}{\partial [s_m^{l+1}]_1} \frac{\partial [s_m^{l+1}]_1}{\partial [s_j^l]_0} \\ &\quad + \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_2} \frac{\partial [s_m^{l+1}]_2}{\partial [s_j^l]_0} + \frac{\partial E}{\partial [s_m^{l+1}]_3} \frac{\partial [s_m^{l+1}]_3}{\partial [s_j^l]_0}, \end{aligned} \tag{2.3.4}$$

$$\begin{aligned} \frac{\partial E}{\partial [s_j^l]_1} &= \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_0} \frac{\partial [s_m^{l+1}]_0}{\partial [s_j^l]_1} + \frac{\partial E}{\partial [s_m^{l+1}]_1} \frac{\partial [s_m^{l+1}]_1}{\partial [s_j^l]_1} \\ &\quad + \sum_m \frac{\partial E}{\partial [s_m^{l+1}]_2} \frac{\partial [s_m^{l+1}]_2}{\partial [s_j^l]_1} + \frac{\partial E}{\partial [s_m^{l+1}]_3} \frac{\partial [s_m^{l+1}]_3}{\partial [s_j^l]_1}, \end{aligned} \tag{2.3.5}$$

şi analog pentru $\frac{\partial E}{\partial [s_j^l]_2}$ şi $\frac{\partial E}{\partial [s_j^l]_3}$, unde sumele sunt luate după toţi neuronii $m$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite conexiuni. Apoi

$$\frac{\partial [s_m^{l+1}]_0}{\partial [s_j^l]_0} = \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_0} \frac{\partial [y_j^l]_0}{\partial [s_j^l]_0} + \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_1} \frac{\partial [y_j^l]_1}{\partial [s_j^l]_0} + \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_2} \frac{\partial [y_j^l]_2}{\partial [s_j^l]_0} + \frac{\partial [s_m^{l+1}]_0}{\partial [y_j^l]_3} \frac{\partial [y_j^l]_3}{\partial [s_j^l]_0},$$

şi relaţiile analoage. Folosind (2.3.2) şi

$$\begin{aligned} w_{mj}^{l+1} y_j^l &= ([w_{mj}^{l+1}]_0 [y_j^l]_0 - [w_{mj}^{l+1}]_1 [y_j^l]_1 - [w_{mj}^{l+1}]_2 [y_j^l]_2 - [w_{mj}^{l+1}]_3 [y_j^l]_3) \\ &+ i([w_{mj}^{l+1}]_0 [y_j^l]_1 + [w_{mj}^{l+1}]_1 [y_j^l]_0 + [w_{mj}^{l+1}]_2 [y_j^l]_3 - [w_{mj}^{l+1}]_3 [y_j^l]_2) \\ &+ j([w_{mj}^{l+1}]_0 [y_j^l]_2 + [w_{mj}^{l+1}]_2 [y_j^l]_0 - [w_{mj}^{l+1}]_1 [y_j^l]_3 + [w_{mj}^{l+1}]_3 [y_j^l]_1) \\ &+ k([w_{mj}^{l+1}]_0 [y_j^l]_3 + [w_{mj}^{l+1}]_3 [y_j^l]_0 + [w_{mj}^{l+1}]_1 [y_j^l]_2 - [w_{mj}^{l+1}]_2 [y_j^l]_1) \end{aligned}$$

avem

$$\frac{\partial[s_m^{l+1}]_0}{\partial[y_j^l]_0}=[w_{mj}^{l+1}]_0,\frac{\partial[s_m^{l+1}]_0}{\partial[y_j^l]_1}=-[w_{mj}^{l+1}]_1,\frac{\partial[s_m^{l+1}]_0}{\partial[y_j^l]_2}=-[w_{mj}^{l+1}]_2,\frac{\partial[s_m^{l+1}]_0}{\partial[y_j^l]_3}=-[w_{mj}^{l+1}]_3,$$

$$\frac{\partial[s_m^{l+1}]_1}{\partial[y_j^l]_0}=[w_{mj}^{l+1}]_1,\frac{\partial[s_m^{l+1}]_1}{\partial[y_j^l]_1}=[w_{mj}^{l+1}]_0,\frac{\partial[s_m^{l+1}]_1}{\partial[y_j^l]_2}=-[w_{mj}^{l+1}]_3,\frac{\partial[s_m^{l+1}]_1}{\partial[y_j^l]_3}=[w_{mj}^{l+1}]_2,$$

şi celelalte relaţii analoage. Acum, punând toate ecuaţiile de mai sus împreună, relaţiile (2.3.4) şi (2.3.5) devin

$$\begin{aligned}\frac{\partial E}{\partial[s_j^l]_0} &= \left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_0\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_0}+\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_1\frac{\partial[G^l(s_j^l)]_1}{\partial[s_j^l]_0}\\ &+\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_2\frac{\partial[G^l(s_j^l)]_2}{\partial[s_j^l]_0}+\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_3\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_0},\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial[s_j^l]_1} &= \left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_0\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_1}+\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_1\frac{\partial[G^l(s_j^l)]_1}{\partial[s_j^l]_1}\\ &+\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_2\frac{\partial[G^l(s_j^l)]_2}{\partial[s_j^l]_1}+\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_3\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_1},\end{aligned}$$

şi analoagele.

În fine, dacă notăm $\delta_j^l:=\partial E/\partial s_j^l$ şi folosim (2.3.2) pentru a calcula

$$\frac{\partial[s_j^l]_0}{\partial[w_{jk}^l]_0}=[x_k^{l-1}]_0,\frac{\partial[s_j^l]_0}{\partial[w_{jk}^l]_1}=-[x_k^{l-1}]_1,\frac{\partial[s_j^l]_0}{\partial[w_{jk}^l]_2}=-[x_k^{l-1}]_2,\frac{\partial[s_j^l]_0}{\partial[w_{jk}^l]_3}=-[x_k^{l-1}]_3,$$

$$\frac{\partial[s_j^l]_1}{\partial[w_{jk}^l]_0}=[x_k^{l-1}]_1,\frac{\partial[s_j^l]_1}{\partial[w_{jk}^l]_1}=[x_k^{l-1}]_0,\frac{\partial[s_j^l]_1}{\partial[w_{jk}^l]_2}=[x_k^{l-1}]_3,\frac{\partial[s_j^l]_1}{\partial[w_{jk}^l]_3}=-[x_k^{l-1}]_2,$$

şi celelalte, obţinem

$$\Delta w_{jk}^l=-\varepsilon\delta_j^l\overline{x_k^{l-1}},\forall l\in\{2,\dots,L-1\}.$$

Prin urmare, avem că

$$\Delta w_{jk}^l=-\varepsilon\delta_j^l\overline{x_k^{l-1}},\forall l\in\{2,\dots,L\},$$

unde

$$\delta_j^l=\begin{cases}\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_0\left(\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_0}+i\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_1}+j\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_2}+k\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_3}\right)+\dots\\ +\left[\sum_m\overline{w_{mj}^{l+1}}\delta_m^{l+1}\right]_3\left(\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_0}+i\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_1}+j\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_2}+k\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_3}\right), & l\le L-1\\ [y_j^L-t_j]_0\left(\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_0}+i\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_1}+j\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_2}+k\frac{\partial[G^l(s_j^l)]_0}{\partial[s_j^l]_3}\right)+\dots\\ +[y_j^L-t_j]_3\left(\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_0}+i\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_1}+j\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_2}+k\frac{\partial[G^l(s_j^l)]_3}{\partial[s_j^l]_3}\right), & l=L\end{cases}.$$

## 2.4 Reţele neuronale Clifford

În ultimul timp, s-au făcut cercetări în încercarea de a generaliza reţelele neuronale bazate pe numere complexe şi pe cuaternioni, aceste cercetări concretizându-se în ceea ce poartă denumirea de reţele neuronale bazate pe algebre Clifford. Algebrele Clifford reprezintă generalizarea naturală a numerelor complexe şi a cuaternionilor, fiind astfel alegerea naturală pentru folosirea în cadrul teoriei reţelelor neuronale. Reţelele Clifford de tip feedforward au fost introduse de Pearson în teza sa [210], precum şi în articolele [211, 212].

Bayro-Corrochano et al. au introdus reţelele Clifford cu auto-organizare în [27] şi [28]. În lucrarea [29], Bayro-Corrochano & Sommer fac o prezentare generală a acestor reţele, introduc reţelele Clifford de tip RBF, şi prezintă experimente în preprocesarea geometrică incorporată, recunoaşterea de obiecte tridimensionale şi recunoaşterea geometriei în procese haotice. În [127], sunt prezentate diferite aplicaţii ale reţelelor Clifford: procesarea imaginilor color, predicţia locaţiei vehiculelor şi analiza biometrică bazată pe verificarea automată a semnăturilor. Articolul [30] dezvoltă SVM-urile Clifford, cu noi experimente în învăţarea proceselor haotice şi în clasificarea multivectorilor. Buchholz & Sommer [51] discută neuronul Clifford în contextul învăţării transformărilor geometrice, ceea ce îl face promiţător pentru folosirea în aplicaţii. După ce face o prezentare sintetică a domeniului în [24], Bayro-Corrochano prezintă aplicaţii ale SVM-urilor Clifford în vederea automată tridimensională în [31]. O aplicaţie interesantă şi inovativă apare în [223], reţelele Clifford fiind folosite pentru recunoaşterea de caractere.

Un rezumat al aplicaţiilor reţelelor Clifford este făcut în [32], cu accent pe problemele din domeniul recunoaşterii de tipare şi a determinării obiectelor tridimensionale pornind de la imagini bidimensionale (pose estimation). Apoi, în [25] şi în [26], se continuă cercetarea în domeniul SVM-urilor Clifford, şi se introduce un nou tip de neuron, şi anume neuronul geometric conformal. Aplicarea reţelelor Clifford pentru predicţia seriilor de timp haotice a fost făcută în [275]. Un tutorial despre reţelele feedforward cu valori Clifford este [56], unde sunt prezentate şi alte aplicaţii. Un studiu asupra ratelor de învăţare optime pentru neuronul Clifford este [55], unde apare pentru prima dată şi hessiana unui astfel de neuron. Se face de asemenea şi o discuţie amănunţită a suprafeţelor de eroare pentru acest tip de neuroni, cu experimente care să ilustreze rezultatele teoretice.

O altă lucrare generală de prezentare detaliată a stării domeniului este [54], unde sunt introduse reţelele Clifford de tip spinor, o generalizare a reţelelor cu valori cuaternionice de tip spinor, bazate pe noncomutativitatea algebrelor Clifford de dimensiune mai mare decât $2$. Neuronii Clifford de tip versor sunt introduşi în [49], unde sunt date de asemenea formule de actualizare pentru aceşti neuroni care sunt independente de sistemul de coordonate ales. Un alt tutorial despre reţelele pe algebra geometrică, cum mai sunt denumite reţelele Clifford, este [246], unde sunt evidenţiate aplicaţii în special în reprezentarea şi calculele cu date geometrice multidimensionale. SVM-urile Clifford recurente sunt introduse pentru prima dată de către Bayro-Corrochano et al. în [33], iar reţelele Clifford de tip Hopfield de către Vallejo & Bayro-Corrochano în [261]. Acest ultim tip de reţele, a mai fost discutat în [156], iar reţelele Clifford recurente au fost introduse de către Kuroe în [154]. În fine, una dintre cele mai recente aplicaţii ale reţelelor Clifford de tip feedforward este într-o problemă de control a unui robot şi este discutată în [72].

În cele ce urmează, urmărind îndeosebi teza lui Buchholz [48], cât şi capitole din cartea [241], vom face o introducere în recent dezvoltata teorie a reţelelor neuronale bazate pe algebre Clifford.

Motivaţia introducerii acestui tip de reţele este una dublă. Pe de o parte, există ideea clară de a generaliza din planul real, în planul complex, hiperbolic, cuaternionic şi în final în planul Clifford, deoarece aceste algebre sunt extensiile naturale ale numerelor reale şi complexe. Pe de altă parte, pentru a vedea dacă extinderea reţelelor neuronale la algebre multidimensionale nu rezultă într-o reprezentare mai compactă a unor spaţii de semnale, dând astfel un model mai puternic şi cu performanţe mai bune decât cel clasic.

Ideea generală a acestor reţele, menţionată şi în cazul reţelelor cu valori complexe şi cuaternionice, este aceea de a putea procesa semnale multidimensionale care ar putea apărea în diverse domenii folosind mai puţini neuroni Clifford în loc de mai mulţi neuroni reali, ţinând cont şi de forma naturală în care apar aceste semnale. În anumite situaţii, este evident din contextul problemei care dintre algebrele Clifford trebuie folosită, însă în alte cazuri alegerea s-ar putea să nu fie atât de clară. Din acest motiv, trebuie făcute experimente cu diferite algebre Clifford, de exemplu de aceeaşi dimensiune, pentru a vedea care este cea mai potrivită pentru problema în cauză.

Aceasta constituie o motivaţie în plus pentru a studia acest tip de reţele. Exact după cum se experimentează cu diverse arhitecturi de reţele, la fel s-ar putea experimenta cu diverse configuraţii ale algebrei pe care este definită reţeaua neuronală, pentru a o alege pe cea care are cele mai bune performanţe şi cele mai potrivite caracteristici pentru respectiva aplicaţie.

Pentru început, vom introduce succesiv noţiunile necesare definirii unei algebre Clifford.

O *algebră reală* este un spaţiu liniar real $(\mathcal{A}, +, \cdot)$ înzestrat cu un produs biliniar $\otimes$, $\otimes : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$, $(a, b) \longmapsto a \otimes b$. O algebră reală este deci perechea $((\mathcal{A}, +, \cdot), \otimes)$.

Deoarece vom folosi doar algebre reale, în continuare vom folosi termenul simplu de algebră cu înţelesul de algebră reală. De asemenea, dacă nu există pericol de confuzie, vom nota $a \otimes b$ cu $ab$.

O algebră $((\mathcal{A}, +, \cdot), \otimes)$ se numeşte:

(i) *asociativă*, dacă are loc $(a \otimes b) \otimes c = a \otimes (b \otimes c), \forall a, b, c \in \mathcal{A}$.

(ii) *comutativă*, dacă are loc $a \otimes b = b \otimes a, \forall a, b, c \in \mathcal{A}$.

(iii) *cu element neutru*, dacă $\exists 1 \in \mathcal{A}$, astfel încât are loc $a \otimes 1 = 1 \otimes a = a, \forall a \in \mathcal{A}$.

Biliniaritatea produsului unei algebre are două consecinţe importante, date mai jos. Prima va fi folosită mult în cele de mai jos.

**Propoziţia 1.** *Pentru orice algebră $((\mathcal{A}, +, \cdot), \otimes)$, produsul $\otimes$ este unic determinat de produsele dintre elementele unei baze a lui $\mathcal{A}$.*

A doua specifică legătura dintre algebre şi inele.

**Propoziţia 2.** *Orice algebră $((\mathcal{A}, +, \cdot), \otimes)$ este distributivă conform definiţiei, sau altfel spus, $(\mathcal{A}, +, \otimes)$ este un inel.*

Două algebre finit dimensionale $\mathcal{A}$ şi $\mathcal{B}$ sunt izomorfe dacă sunt izomorfe ca inele, şi scriem $\mathcal{A} \simeq \mathcal{B}$, adică există o aplicaţie bijectivă $\varphi : \mathcal{A} \to \mathcal{B}$, care satisface, pentru orice $a, b \in \mathcal{A}$, relaţiile:

(i) $\varphi(a +_{\mathcal{A}} b) = \varphi(a) +_{\mathcal{B}} \varphi(b)$,

(ii) $\varphi(a \otimes_{\mathcal{A}} b) = \varphi(a) \otimes_{\mathcal{B}} \varphi(b)$.

Se poate defini şi un produs tensorial de algebre:

Fie $\mathcal{A}$ o algebră asociativă cu element neutru. Dacă există două subalgebre $\mathcal{B}$ şi $\mathcal{C}$ a lui $\mathcal{A}$ astfel încât:

(i) $b \otimes c = c \otimes b, \forall b \in \mathcal{B}, \forall c \in \mathcal{C}$,

(ii) $\mathcal{A}$ este generată ca algebră de $\mathcal{B}$ şi $\mathcal{C}$,

(iii) $\dim \mathcal{A} = \dim \mathcal{B} \cdot \dim \mathcal{C}$,

atunci $\mathcal{A}$ este *produsul tensorial* dintre $\mathcal{B}$ şi $\mathcal{C}$, şi scriem $\mathcal{A} = \mathcal{B} \otimes \mathcal{C}$.

Fie $X$ un spaţiu liniar real înzestrat cu un produs scalar, adică cu o formă simetrică biliniară $F : X \times X \to \mathbb{R}$, $(a, b) \longmapsto a \cdot b$. Atunci aplicaţia $Q : X \to \mathbb{R}$, $a \longmapsto a \cdot a$ se numeşte *forma pătratică* a lui $F$. Mai mult, perechea $(X, Q)$ se numeşte *spaţiu pătratic real*.

Se observă că $Q$ este unic determinată de către $F$ şi viceversa, deoarece

$$F(a, b) = \frac{1}{2}(Q(a + b) - Q(a) - Q(b)). \tag{2.4.1}$$

Orice spaţiu pătratic real finit dimensional posedă o bază specială, după cum rezultă din propoziţia de mai jos:

**Propoziţia 3.** *Fie $(X, Q)$ un spaţiu pătratic real $n$-dimensional. Atunci există o bază $\{e_1, \ldots, e_n\}$ a lui $(X, Q)$ şi numerele unic determinate $p, q, r \in \{0, \ldots, n\}$, astfel încât, pentru orice $i, j \in \{1, \ldots, n\}$, următoarele condiţii sunt îndeplinite:*

(i) $Q(e_i) = \begin{cases} 1 & 1 \leq i \leq p \\ -1 & p + 1 \leq i \leq p + q \\ 0 & p + q + 1 \leq i \leq p + q + r = n \end{cases}$,

(ii) $Q(e_i + e_j) - Q(e_i) - Q(e_j) = 0$.

O bază care are proprietăţile de mai sus se numeşte *bază ortonormată* a lui $(X, Q)$, iar tripletul $(p, q, r)$ se numeşte *signatura* lui $(X, Q)$.

Se poate face o diferenţiere între spaţiile pătratice pornind de la valorile lui $r$. Un spaţiu pătratic real finit dimensional $(X, Q)$ se numeşte *degenerat* dacă există $a \in X$ astfel încât $Q(a) = 0$, şi *nedegenerat* în caz contrar. Se poate demonstra că dacă $r = 0$, atunci spaţiul este nedegenerat, iar dacă $r \neq 0$, atunci el este degenerat.

Particularizând, pentru $X = \mathbb{R}^{p+q+r}$, obţinem *spaţiul pătratic standard.*

Pentru orice $p, q, r \in \{0, \ldots, n\}$ definim produsul scalar

$$F : \mathbb{R}^{p+q+r} \times \mathbb{R}^{p+q+r} \to \mathbb{R}, (a, b) \longmapsto \sum_{i=1}^{p} a_i b_i - \sum_{i=p+1}^{p+q} a_i b_i.$$

Spaţiul pătratic real standard corespunzător acestui produs scalar este $(\mathbb{R}^{p+q+r}, Q)$, care va fi notat $\mathbb{R}^{p,q,r}$.

Suntem acum pregătiţi să dăm definiţia unei algebre Clifford.

*Algebra universală Clifford* $C\ell_{p,q,r}$ a lui $\mathbb{R}^{p,q,r}$ este unica algebră asociativă $2^{p+q+r}$-dimensională cu element unitate $1_{\mathbb{R}}$, care satisface următoarele proprietăţi:

(i) $C\ell_{p,q,r}$ este generată ca algebră de către subspaţiile sale distincte $\mathbb{R}$ şi $\mathbb{R}^{p,q,r}$,

(ii) $x \otimes_{p,q,r} x = xx = Q(x) \cdot 1_{\mathbb{R}}, \forall x \in \mathbb{R}^{p,q,r}$.

Deoarece fiecare algebră Clifford este o algebră reală asociativă prin definiţie, are loc următoarea teoremă importantă:

**Teorema 1.** *Orice algebră Clifford este izomorfă cu o algebră de matrici.*

Deoarece vom folosi doar algebrele universale, $C\ell_{p,q,r}$, acest atribut nu va mai fi menţionat. De asemenea, dacă $r = 0$, vom folosi notaţiile mai scurte $C\ell_{p,q}$ şi $\otimes_{p,q}$.

În cele ce urmează vom specifica o bază a algebrelor $C\ell_{p,q,r}$. Să notăm

$$\mathcal{I} = \{\{i_1, \ldots, i_s\} \in \mathcal{P}(\{1, \ldots, n\}) | 1 \leq i_1 < \cdots < i_s \leq n\}, \tag{2.4.2}$$

unde $\mathcal{P}(\{1, \ldots, n\})$ este mulţimea părţilor lui $\{1, \ldots, n\}$, iar $n = p + q + r$.

Acum, dacă $\{e_1, \ldots, e_n\}$ este o bază ortonormată a lui $\mathbb{R}^{p,q,r}$, atunci, pentru orice $I \in \mathcal{I}$ definim

$$e_I := e_{i_1} \cdots e_{i_s}, \tag{2.4.3}$$

unde între elementele bazei avem produsul $\otimes_{p,q,r}$, scris prescurtat prin juxtapunere. În particular, $e_\emptyset := 1_\mathbb{R}$. În cele ce urmează, când nu este pericol de confuzie, vom nota $e_I = e_{\{i_1,\ldots i_s\}}$ cu $e_{i_1 \cdots i_s}$. Putem demonstra acum următoarea propoziţie:

**Propoziţia 4.** *Dacă $\{e_1, \ldots, e_n\}$ este o bază ortonormată a lui $\mathbb{R}^{p,q,r}$, atunci $\{e_I | I \in \mathcal{I}\}$, unde $\mathcal{I}$ este definit ca mai sus, este o bază pentru algebra Clifford $C\ell_{p,q,r}$. În particular, orice element din $C\ell_{p,q,r}$, numit multivector, poate fi scris ca*

$$x = \sum_{I \in \mathcal{I}} x_I e_I, \tag{2.4.4}$$

*unde $x_I \in \mathbb{R}, \forall I \in \mathcal{I}$.*

Se observă uşor că baza este formată din $2^n = 2^{p+q+r}$ vectori, ceea ce este consistent cu faptul că algebrele Clifford $C\ell_{p,q,r}$ au dimensiune $2^{p+q+r}$. Acei vectori pentru care $|I| = k$, unde $k \in \{0, \ldots, n\}$ se numesc $k$-vectori şi generează subspaţii ale lui $C\ell_{p,q,r}$.

Dacă $\{e_1, \ldots, e_n\}$ este o bază ortonormată a lui $\mathbb{R}^{p,q,r}$, atunci, pentru orice $k \in \{0, \ldots, n\}$, mulţimea

$$\{e_I | I \in \mathcal{I}, |I| = k\} \tag{2.4.5}$$

generează un subspaţiu liniar al algebrei $C\ell_{p,q,r}$, notat $C\ell^k_{p,q,r}$, ale cărui elemente se numesc $k$-*vectori*.

Este clar că

$$C\ell_{p,q,r} = \bigoplus_{k=0}^{n} C\ell^k_{p,q,r}, \tag{2.4.6}$$

unde am notat prin $\bigoplus$ suma directă de subspaţii. Este evident şi că $C\ell^0_{p,q,r} = \mathbb{R}$ şi $C\ell^1_{p,q,r} = \mathbb{R}^{p,q,r}$. Fiecare subspaţiu $C\ell^k_{p,q,r}$ este de dimensiune $\mathrm{C}^k_n$. Toate subspaţiile de dimensiune pară ale lui $C\ell_{p,q,r}$ formează o subalgebră:

**Propoziţia 5.** *Suma directă definită prin*

$$C\ell^+_{p,q,r} := \bigoplus_{k=0}^{\left[\frac{n}{2}\right]} C\ell^{2k}_{p,q,r} \tag{2.4.7}$$

*este o subalgebră a lui $C\ell_{p,q,r}$ de dimensiune $2^{n-1}$, numită subalgebra pară a lui $C\ell_{p,q,r}$.*

Putem defini, pentru orice $k \in \{0, \ldots, n\}$, următorul *operator de gradare*:

$$\langle \cdot \rangle_k : C\ell_{p,q,r} \to C\ell^k_{p,q,r}, x \longmapsto \sum_{\substack{I \in \mathcal{I} \\ |I|=k}} x_I e_I. \tag{2.4.8}$$

Folosind acest operator, orice multivector din $C\ell_{p,q,r}$ poate fi scris ca

$$x = \sum_{k=0}^{n} \langle x \rangle_k . \tag{2.4.9}$$

Un automorfism $f$ de algebră $\mathcal{A}$ este *involuţie* dacă şi numai dacă $f^2 = 1_{\mathcal{A}}$.

Pentru algebrele Clifford nedegenerate $C\ell_{p,q}$ se pot defini trei automorfisme importante, care sunt şi involuţii:

(i) Inversiune $\widetilde{\cdot} : C\ell_{p,q} \to C\ell_{p,q}, x \longmapsto \sum_{k=0}^{n} (-1)^k \langle x \rangle_k$ ,

(ii) Reversiune $\widehat{\cdot} : C\ell_{p,q} \to C\ell_{p,q}, x \longmapsto \sum_{k=0}^{n} (-1)^{\frac{k(k-1)}{2}} \langle x \rangle_k$ ,

(ii) Conjugare $\bar{\cdot} : C\ell_{p,q} \to C\ell_{p,q}, x \longmapsto \sum_{k=0}^{n} (-1)^{\frac{k(k+1)}{2}} \langle x \rangle_k$ .

În cele ce urmează vom introduce reţele neuronale cu valori din $C\ell_{p,q}$. Să presupunem că avem o reţea neuronală cu valori Clifford de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile notate cu $\{2, \ldots, L-1\}$ sunt straturi ascunse. Funcţia de eroare $E : (C\ell_{p,q})^N \to \mathbb{R}$ pentru o asemenea reţea este definită prin

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{c} ||y_i^L - t_i||^2, \tag{2.4.10}$$

unde $||y||$ reprezintă norma euclidiană a lui $y$. $\mathbf{y}^L = (y_i^L)_{1 \le i \le c} \in (C\ell_{p,q})^c$ sunt *ieşirile* reţelei, $\mathbf{t} = (t_i)_{1 \le i \le c} \in (C\ell_{p,q})^c$ sunt *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{w} \in (C\ell_{p,q})^N$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei.

Dacă notăm cu $w_{jk}^l \in C\ell_{p,q}$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l \in \{2, \ldots, L\}$, putem defini *pasul de actualizare* al ponderii $w_{jk}^l$ în epoca $t$ ca fiind

$$\Delta w_{jk}^l(t) = w_{jk}^l(t+1) - w_{jk}^l(t).$$

Cu această notaţie, pentru *metoda gradient*, *regula de actualizare* pentru ponderea $w_{jk}^l$ este

$$\Delta w_{jk}^l(t) = -\varepsilon \sum_{A \in \mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A}(t) e_A,$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, unde $\frac{\partial E}{\partial [w_{jk}^l]_A}(t)$ reprezintă derivata parţială a funcţiei $E$ în raport cu fiecare componentă $[w_{jk}^l]_A$ a numărului Clifford $w_{jk}^l$, cu $A \in \mathcal{I}$.

Scris sub formă vectorială, pasul de actualizare devine

$$\Delta \mathbf{w}(t) = \mathbf{w}(t+1) - \mathbf{w}(t),$$

unde $\mathbf{w} \in (C\ell_{p,q})^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient devine acum:

$$\Delta \mathbf{w}(t) = -\varepsilon \nabla E(t),$$

unde $\nabla E(t) := \nabla E(\mathbf{w}(t)) \in (C\ell_{p,q})^N$, este gradientul funcţiei $E$, ale cărui componente $\sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A}(\mathbf{w}(t))e_A$ au fost scrise prescurtat mai sus sub forma $\sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A}(t)e_A$. Prin urmare, pentru a minimiza funcţia de eroare $E$ pentru metoda gradient, avem nevoie să calculăm derivatele parţiale de forma $\frac{\partial E}{\partial [w_{jk}^l]_A}$, pentru $A \in \mathcal{I}$.

Pentru aceasta, facem notaţiile

$$s_j^l := \sum_k w_{jk}^l \otimes_{p,q} x_k^{l-1}, \tag{2.4.11}$$

$$y_j^l := G^l(s_j^l), \tag{2.4.12}$$

unde (2.4.11) ne spune că înmulţirea de la reţelele neuronale cu valori reale este înlocuită cu înmulţirea Clifford, $G^l$ este *funcţia de activare* a stratului $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1\le k\le d} \in (C\ell_{p,q})^d$ sunt intrările reţelei, şi avem că $x_k^l = y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$. Funcţia de activare este considerată a fi definită pe componente. De pildă, pentru numărul Clifford $x = \sum_{I\in\mathcal{I}} [x]_I e_I \in C\ell_{p,q}$, un exemplu de funcţie de activare este funcţia tangentă hiperbolică pe componente definită prin

$$G\left(\sum_{I\in\mathcal{I}} [x]_I e_I\right) = \sum_{I\in\mathcal{I}} (\tanh[x]_I) e_I.$$

Vom începe cu actualizarea pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, care este dată prin

$$\Delta w_{jk}^L(t) = -\varepsilon \sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w_{jk}^L]_A}(t) e_A.$$

Folosind regula înlănţuirii, putem scrie următorul set de relaţii:

$$\frac{\partial E}{\partial [w_{jk}^L]_A} = \sum_{B\in\mathcal{I}} \frac{\partial E}{\partial [s_j^L]_B} \frac{\partial [s_j^L]_B}{\partial [w_{jk}^L]_A}, \tag{2.4.13}$$

pentru orice $A \in \mathcal{I}$.

Ne vom ocupa mai întâi de $\frac{\partial [s_j^L]_B}{\partial [w_{jk}^L]_A}$. Astfel, avem, $\forall A, B \in \mathcal{I}$, că

$$\frac{\partial [s_j^L]_B}{\partial [w_{jk}^L]_A} = \frac{\partial [\sum_k w_{jk}^L \otimes_{p,q} x_k^{L-1}]_B}{\partial [w_{jk}^L]_A} = \frac{\partial [w_{jk}^L \otimes_{p,q} x_k^{L-1}]_B}{\partial [w_{jk}^L]_A}. \tag{2.4.14}$$

Folosind faptul că

$$w_{jk}^L \otimes_{p,q} x_k^{L-1} = \sum_{C,D\in\mathcal{I}} [w_{jk}^L]_C [x_k^{L-1}]_D e_C e_D, \tag{2.4.15}$$

conchidem că ecuaţia (2.4.14) se poate scrie

$$\begin{aligned}\frac{\partial[w_{jk}^L \otimes_{p,q} x_k^{L-1}]_B}{\partial[w_{jk}^L]_A} &= \frac{\partial[\sum_{C,D\in\mathcal{I}}[w_{jk}^L]_C[x_k^{L-1}]_D e_C e_D]_B}{\partial[w_{jk}^L]_A} \\ &= \frac{\partial(\sum_{\substack{C,D\in\mathcal{I}\\ \kappa_{C,D}e_Ce_D=e_B}}\kappa_{C,D}[w_{jk}^L]_C[x_k^{L-1}]_D)}{\partial[w_{jk}^L]_A} \\ &= \kappa_{A,D}[x_k^{L-1}]_D, \text{ unde } \kappa_{A,D}e_Ae_D = e_B, \kappa_{A,D}\in\{\pm 1\}.\end{aligned}$$

Am obţinut prin urmare

$$\frac{\partial[s_j^L]_B}{\partial[w_{jk}^L]_A} = \kappa_{A,D}[x_k^{L-1}]_D, \text{ unde } \kappa_{A,D}e_Ae_D = e_B,$$

de unde relaţia (2.4.13) se poate scrie sub forma

$$\frac{\partial E}{\partial[w_{jk}^L]_A} = \sum_{\substack{B\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_B}} \frac{\partial E}{\partial[s_j^L]_B}\kappa_{A,D}[x_k^{L-1}]_D. \tag{2.4.16}$$

Mai departe, cu notaţia $\delta_j^L := \frac{\partial E}{\partial s_j^L}$, avem din regula înlănţuirii că

$$[\delta_j^L]_B = \frac{\partial E}{\partial[s_j^L]_B} = \sum_{E\in\mathcal{I}}\frac{\partial E}{\partial[y_j^L]_E}\frac{\partial[y_j^L]_E}{\partial[s_j^L]_B},$$

$\forall B\in\mathcal{I}$, sau, ţinând seama de notaţia (2.4.12) şi de expresia pentru funcţia de eroare (2.4.10), că

$$\begin{aligned}[\delta_j^L]_B &= \sum_{E\in\mathcal{I}}([y_j^L]_E - [t_j]_E)\frac{\partial[G^L(s_j^L)]_E}{\partial[s_j^L]_B} \\ &= ([y_j^L]_B - [t_j]_B)\frac{\partial[G^L(s_j^L)]_B}{\partial[s_j^L]_B},\end{aligned}$$

$\forall B\in\mathcal{I}$, deoarece $[G^L(s_j^L)]_E$ depinde de $[s_j^L]_B$ doar pentru $E=B$, ceea ce înseamnă că $\frac{\partial[G^L(s_j^L)]_E}{\partial[s_j^L]_B} = 0, \forall E\neq B$. Dacă notăm cu $\odot$ înmulţirea pe componente a două numere Clifford, relaţia de mai sus ne dă

$$\delta_j^L = (y_j^L - t_j)\odot\frac{\partial G^L(s_j^L)}{\partial s_j^L}, \tag{2.4.17}$$

unde $\frac{\partial G^L(s_j^L)}{\partial s_j^L}$ este numărul Clifford al derivatei pe componente a funcţiei de activare $G^L$. De exemplu, dacă $x=\sum_{I\in\mathcal{I}}[x]_I e_I$, atunci

$$\frac{\partial G(x)}{\partial x} = \sum_{I\in\mathcal{I}}(\operatorname{sech}^2[x]_I)e_I,$$

cu funcţia $G$ definită ca în exemplul de mai sus. În final, din ecuaţia (2.4.16), obţinem expresia pentru actualizarea dorită:

$$\Delta w_{jk}^L(t) = -\varepsilon \delta_j^L \otimes_{p,q} (x_k^{L-1})^*,$$

unde numărul Clifford $\delta_j^L \in C\ell_{p,q}$ este dat de relaţia (2.4.17), şi $x^*$ reprezintă conjugatul Clifford al numărului Clifford $x$, operaţia de conjugare fiind cea definită mai sus.

Acum, vom calcula actualizarea pentru o pondere arbitrară $w_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, aceasta este dată prin

$$\Delta w_{jk}^l(t) = -\varepsilon \sum_{A \in \mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A} e_A,$$

iar apoi, regula înlănţuirii ne dă

$$\frac{\partial E}{\partial [w_{jk}^l]_A} = \sum_{B \in \mathcal{I}} \frac{\partial E}{\partial [s_j^l]_B} \frac{\partial [s_j^l]_B}{\partial [w_{jk}^l]_A}, \forall A \in \mathcal{I}. \tag{2.4.18}$$

Aplicând din nou regula înlănţuirii, obţinem

$$\frac{\partial E}{\partial [s_j^l]_B} = \sum_r \sum_{C \in \mathcal{I}} \frac{\partial E}{\partial [s_r^{l+1}]_C} \frac{\partial [s_r^{l+1}]_C}{\partial [s_j^l]_B}, \forall B \in \mathcal{I}, \tag{2.4.19}$$

unde suma se ia după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite legături. Apoi, cu aceeaşi regulă, avem că

$$\frac{\partial [s_r^{l+1}]_C}{\partial [s_j^l]_B} = \sum_{D \in \mathcal{I}} \frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} \frac{\partial [y_j^l]_D}{\partial [s_j^l]_B}, \forall B, C \in \mathcal{I}.$$

Ca mai sus, avem, $\forall A, B \in \mathcal{I}$, că

$$\frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} = \frac{\partial [\sum_j w_{rj}^{l+1} \otimes_{p,q} y_j^l]_C}{\partial [y_j^{(l)}]_D} = \frac{\partial [w_{rj}^{l+1} \otimes_{p,q} y_j^l]_C}{\partial [y_j^{(l)}]_D}. \tag{2.4.20}$$

Din

$$w_{rj}^{l+1} \otimes_{p,q} y_j^l = \sum_{E,F \in \mathcal{I}} [w_{rj}^{l+1}]_E [y_j^l]_F e_E e_F, \tag{2.4.21}$$

rezultă că ecuaţia (2.4.20) se poate scrie

$$\begin{aligned} \frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} &= \frac{\partial [\sum_{E,F \in \mathcal{I}} [w_{rj}^{l+1}]_E [y_j^l]_F e_E e_F]_C}{\partial [y_j^l]_D} \\ &= \frac{\partial (\sum_{\substack{E,F \in \mathcal{I} \\ \kappa_{E,F} e_E e_F = e_C}} \kappa_{E,F} [w_{rj}^{l+1}]_E [y_j^l]_F)}{\partial [y_j^l]_D} \\ &= \kappa_{E,D} [w_{rj}^{l+1}]_E, \text{ unde } \kappa_{E,D} e_E e_D = e_C. \end{aligned}$$

Deci

$$\frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} = \kappa_{E,D} [w_{rj}^{l+1}]_E, \text{ unde } \kappa_{E,D} e_E e_D = e_C.$$

şi apoi

$$\begin{aligned}\frac{\partial[s_r^{l+1}]_C}{\partial[s_j^l]_B} &= \sum_{\substack{D\in\mathcal{I}\\ \kappa_{E,D}e_Ee_D=e_C}} \kappa_{E,D}[w_{rj}^{l+1}]_E\frac{\partial[G^l(s_j^l)]_D}{\partial[s_j^l]_B}\\ &= \kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B},\ \text{unde } \kappa_{E,B}e_Ee_B=e_C,\end{aligned}$$

$\forall B,C\in\mathcal{I}$, unde din nou am ţinut cont de faptul că $\frac{\partial[G^l(s_j^l)]_D}{\partial[s_j^l]_B}=0,\forall D\neq B$. Revenind înapoi în ecuaţia (2.4.19), avem

$$\begin{aligned}\frac{\partial E}{\partial[s_j^l]_B} &= \sum_r\sum_{\substack{C\in\mathcal{I}\\ \kappa_{E,B}e_Ee_B=e_C}}\frac{\partial E}{\partial[s_r^{l+1}]_C}\kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B}\\ &= \sum_r\left(\sum_{\substack{C\in\mathcal{I}\\ \kappa_{E,B}e_Ee_B=e_C}}\kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial E}{\partial[s_r^{l+1}]_C}\right)\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B}\\ &= \sum_r[(w_{rj}^{l+1})^*\otimes_{p,q}\delta_r^{l+1}]_B\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B},\end{aligned}$$

$\forall B\in\mathcal{I}$, unde, din nou, $\delta_j^l:=\frac{\partial E}{\partial s_j^l}$. Acum, putem scrie relaţia de mai sus sub forma

$$\delta_j^l=\left(\sum_r(w_{rj}^{l+1})^*\otimes_{p,q}\delta_r^{l+1}\right)\odot\frac{\partial G^l(s_j^l)}{\partial s_j^l},\tag{2.4.22}$$

unde $\odot$ este înmulţirea pe componente a două numere Clifford, ca mai sus.

Apoi, ţinând cont că

$$\frac{\partial[s_j^l]_B}{\partial[w_{jk}^l]_A}=\kappa_{A,D}[x_k^{l-1}]_D,\ \text{unde } \kappa_{A,D}e_Ae_D=e_B,$$

relaţia (2.4.18) devine

$$\begin{aligned}\frac{\partial E}{\partial[w_{jk}^l]_A} &= \sum_{\substack{B\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_B}}\frac{\partial E}{\partial[s_j^l]_B}\kappa_{A,D}[x_k^{l-1}]_D\\ &= [\delta_j^l\otimes_{p,q}(x_k^{l-1})^*]_A,\end{aligned}$$

unde $\delta_j^l:=\frac{\partial E}{\partial s_j^l}$, ca mai sus.

În final am obţinut o formulă de calcul pentru actualizarea ponderii $w_{jk}^l$ asemănătoare cu cea din primul caz tratat, şi anume

$$\Delta w_{jk}^l(t)=-\varepsilon\delta_j^l\otimes_{p,q}(x_k^{l-1})^*.$$

Avem deci următoarea formulă pentru actualizarea ponderii $w_{jk}^l$:

$$\Delta w_{jk}^l(t) = -\varepsilon \delta_j^l \otimes_{p,q} (x_k^{l-1})^*, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left(\sum_r (w_{rj}^{l+1})^* \otimes_{p,q} \delta_r^{l+1}\right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l \leq L-1 \\ (y_j^l - t_j) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l = L \end{cases}.$$

## 2.5 Concluzii

Prezentul capitol este dedicat introducerii în stadiul actual al cercetării în domeniul reţelelor neuronale Clifford de tip feedforward. Astfel,

- Secţiunea 2.1 prezintă **reţelele neuronale cu valori complexe**. După o scurtă introducere în istoria reţelelor neuronale cu valori complexe şi o prezentare a progreselor făcute în acest domeniu în ultimii ani, precum şi a aplicaţiilor acestui tip de reţele, am făcut o descriere a mulţimii numerelor complexe privită din mai multe perspective. Apoi am discutat principalele probleme ale calculului diferenţial pe această mulţime şi am arătat diferenţa dintre neuronul complex şi neuronul real. În final, am dedus algoritmul backpropagation pentru reţelele neuronale cu valori complexe de tip feedforward, folosind derivate parţiale reale.
- Secţiunea 2.2 introduce **reţelele neuronale cu valori hiperbolice**. O generalizare diferită de dimensiune doi a mulţimii numerelor reale, mulţimea numerelor hiperbolice a fost introdusă prin mai multe definiţii echivalente, după evidenţierea principalelor contribuţii în domeniul reţelelor neuronale hiperbolice. Apoi am prezentat algoritmul backpropagation pentru reţelele neuronale cu valori hiperbolice de tip feedforward, folosind singurul tip de calcul diferenţial posibil pe mulţimea numerelor hiperbolice în acest context, şi anume cel cu derivate parţiale reale.
- În Secţiunea 2.3, am făcut o prezentare succintă a **reţelelor neuronale cu valori cuaternionice**. Mulţimea numerelor cuaternionice este o generalizare de dimensiune patru a mulţimii numerelor reale, fiind prima dintre mulţimile discutate în care înmulţirea definită nu este comutativă. La început, am prezentat cele mai importante contribuţii în domeniul reţelelor cu valori cuaternionice, evidenţiind de asemenea şi aplicaţiile acestui tip de reţele. Şi cuaternionii pot fi definiţi în mai multe feluri echivalente, pe care le-am prezentat în această secţiune. Apoi am introdus reţelele neuronale cu valori cuaternionice de tip feedforward, pentru care am arătat deducerea completă a algoritmului de învăţare backpropagation. Calculul diferenţial a presupus patru derivate parţiale diferite pentru fiecare funcţie definită pe mulţimea numerelor cuaternionice, câte una pentru fiecare din cele patru componente reale ale unui astfel de număr.
- În fine, Secţiunea 2.4 prezintă generalizarea tuturor acestor tipuri de reţele, şi anume reţelele neuronale cu valori în algebre Clifford, sau **reţelele neuronale Clifford**. Datorită faptului că definiţia unei algebre Clifford este mai dificilă decât în cazurile discutate anterior, am prezentat pas cu pas etapele necesare construirii unei astfel de algebre. După ce am definit algebra Clifford, am enumerat succint contribuţiile şi aplicaţiile existente la ora actuală în acest domeniu, iar apoi am introdus reţelele neuronale Clifford de tip feedforward. Algoritmul backpropagation a fost dedus şi pentru aceste reţele, folosind tot calculul diferenţial real, şi fiind o generalizare a tuturor algoritmilor de acest tip prezentaţi în secţiunile anterioare.

# Capitolul 3

# Algoritmi de învăţare pentru reţele neuronale cu valori complexe

Prezentul capitol este dedicat deducerii complete a algoritmilor pentru antrenarea reţelelor neuronale de tip feedforward cu valori complexe. Motivul pentru care au fost introduşi toţi aceşti algoritmi este acela că s-a demonstrat în cazul real că nu există un algoritm sau o clasă de algoritmi care să fie mai bun decât toţi ceilalţi indiferent de aplicaţie. Astfel, punem la dispoziţia celor interesaţi o paletă largă de algoritmi, din care, în urma experimentelor, se pot alege cei mai potriviţi şi cu cea mai bună performanţă pentru o anumită problemă.

Secţiunea 3.1 face o introducere în calculul diferenţial Wirtinger sau calculul diferenţial $\mathbb{CR}$, care este standardul în literatură pentru lucrul cu reţele neuronale cu valori complexe. Sunt prezentate pas cu pas, toate noţiunile care vor fi folosite pe parcursul capitolului pentru a deduce algoritmii de învăţare propuşi.

Secţiunea 3.2 prezintă metoda gradient folosind algoritmul de tip backpropagation, însă de această dată bazat pe calculul Wirtinger, spre deosebire de Secţiunea 2.1, care a prezentat acelaşi algoritm folosind derivate parţiale reale.

Următoarea secţiune, Secţiunea 3.3, introduce metodele gradient îmbunătăţite: metoda gradient cu moment, metoda quickprop, metoda resilient backpropagation, metoda delta-bar-delta şi metoda SuperSAB. Cu excepţia primei şi a celei de-a treia, toate aceste metode sunt deduse aici pentru prima oară. Aceşti algoritmi constituie o excepţie faţă de toţi ceilalţi din acest capitol, deoarece ei nu pot fi deduşi folosind calculul Wirtinger, ci doar derivatele parţiale reale.

După această clasă simplă de algoritmi, Secţiunea 3.4 este dedicată metodei gradienţilor conjugaţi. Prima subsecţiune explică metoda gradienţilor conjugaţi liniară, care stă la baza celei de a doua subsecţiuni, unde sunt deduse expresii pentru metoda gradienţilor conjugaţi cu actualizări Hestenes-Stiefel, metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere, metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves, metoda gradienţilor conjugaţi cu actualizări Dai-Yuan, variaţiile algoritmilor Hestenes-Stiefel şi Polak-Ribiere şi metoda gradienţilor conjugaţi cu reporniri Powell-Beale.

În continuare, Secţiunea 3.5 introduce o variaţiune a metodei gradienţilor conjugaţi, şi anume metoda gradienţilor conjugaţi scalaţi. Aceasta propune două îmbunătăţiri ale metodei mai sus menţionate, care se vor dovedi foarte importante în cadrul experimentelor.

Cea mai importantă metodă de ordinul doi, metoda Newton, este prezentată în Secţiunea 3.6. Deoarece această metodă presupune calculul explicit al hessianei, un algoritm pentru efectuarea acestui calcul a fost propus într-o subsecţiune a acestei secţiuni, cu o particularizare pentru o reţea cu trei straturi în subsecţiunea următoare.

O simplificare a metodei porneşte de la observaţia că nu este necesar atât calculul hessianei, cât calculul produsului hessianei cu un vector. Astfel, ultima subsecţiune prezintă un algoritm pentru efectuarea acestui calcul.

Deoarece metoda Newton este foarte intensivă computaţional, Secţiunea 3.7 introduce o clasă foarte importantă de metode de ordinul doi, care se bazează pe o aproximare a hessianei, şi anume metodele quasi-Newton. Astfel, sunt deduse progresiv metoda metoda actualizării de rang unu, metoda Davidon-Fletcher-Powell, metoda Broyden-Fletcher-Goldfarb-Shanno şi metoda one step secant.

Pentru completitudine, deşi a mai apărut şi în altă parte în literatură, Secţiunea 3.8 prezintă metoda de învăţare Levenberg-Marquardt.

În final, Secţiunea 3.9 este dedicată concluziilor acestui capitol.

## 3.1 Introducere în calculul Wirtinger sau calculul $\mathbb{CR}$

Ideile principale ale acestui tip de calcul sunt prezentate în cele ce urmează (a se vedea, spre exemplu, [153]).

Să presupunem că avem o funcţie $f : \mathbb{C} \to \mathbb{R}$, care în cazul nostru va fi o funcţie de eroare pe care vrem s-o minimizăm. Pentru această funcţie, fie $g : \mathbb{R}^2 \to \mathbb{R}$ astfel încât $g(x, y) = f(z)$, unde $z = x + iy$, $i = \sqrt{-1}$, şi $h : \mathbb{C} \times \mathbb{C} \to \mathbb{R}$, astfel încât $h(z, \overline{z}) = f(z)$, unde $\overline{z}$ este conjugatul complex al lui $z$. De exemplu, dacă $f_0(z) = z\overline{z} = |z|^2 = x^2 + y^2$, atunci $g_0(x, y) = x^2 + y^2$, şi $h_0(z, \overline{z}) = z\overline{z}$, unde în acest ultim caz $z$ şi $\overline{z}$ sunt variabile independente.

Funcţia $f$ nu este derivabilă complex datorită faptului că are valori reale, singurele funcţii derivabile complex sau olomorfe cu valori reale fiind cele constante. Prin urmare, $f$ nu poate avea decât derivate parţiale în funcţie de părţile reală şi respectiv imaginară ale argumentului complex. Din felul în care a fost definită funcţia $g$, rezultă că

$$\frac{\partial f}{\partial z^R}(z) = \frac{\partial g}{\partial x}(x, y),$$
$$\frac{\partial f}{\partial z^I}(z) = \frac{\partial g}{\partial y}(x, y),$$

$\forall z = z^R + iz^I = x + iy \in \mathbb{C}$. Din nou din $g(x, y) = f(z)$, rezultă că orice punct de extrem local pentru $g$ este un punct de extrem local pentru $f$. Deci, pentru a minimiza funcţia $f$ este suficient să minimizăm funcţia $g$.

Pentru funcţia $h$, definim $\mathbb{R}$-*derivata parţială* prin

$$\frac{\partial h}{\partial z}(z, \overline{z}) := \frac{1}{2}\left(\frac{\partial g}{\partial x}(x, y) - i\frac{\partial g}{\partial y}(x, y)\right) = \frac{1}{2}\left(\frac{\partial f}{\partial z^R}(z) - i\frac{\partial f}{\partial z^I}(z)\right),$$

şi $\mathbb{R}$-*derivata parţială complexă*, sau $\overline{\mathbb{R}}$-*derivata parţială* prin

$$\frac{\partial h}{\partial \overline{z}}(z, \overline{z}) := \frac{1}{2}\left(\frac{\partial g}{\partial x}(x, y) + i\frac{\partial g}{\partial y}(x, y)\right) = \frac{1}{2}\left(\frac{\partial f}{\partial z^R}(z) + i\frac{\partial f}{\partial z^I}(z)\right),$$

$\forall z = z^R + iz^I = x + iy \in \mathbb{C}$. De exemplu, avem că $\frac{\partial h_0}{\partial z} = \frac{1}{2}(2x - 2iy) = \overline{z}$ şi $\frac{\partial h_0}{\partial \overline{z}} = \frac{1}{2}(2x + 2iy) = z$, de unde se poate vedea că în cazul $\mathbb{R}$-derivatei parţiale îl tratăm pe $\overline{z}$ ca o constantă şi derivăm în funcţie de $z$, iar pentru $\overline{\mathbb{R}}$-derivata parţială îl tratăm pe $z$ ca o constantă şi derivăm în funcţie de $\overline{z}$.

Să trecem acum la cazul $N$-dimensional. Presupunem că avem o funcţie $F : \mathbb{C}^N \to \mathbb{R}$, care va fi tot o funcţie de eroare în cazul nostru, şi pe care vrem să o minimizăm. Pentru această funcţie, fie $G : \mathbb{R}^{2N} \to \mathbb{R}$ astfel încât $G(\mathbf{x}, \mathbf{y}) = F(\mathbf{z})$,

unde $\mathbf{z} = (z_1, \ldots, z_N)^T \in \mathbb{C}^N$, $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$, $\mathbf{y} = (y_1, \ldots, y_N)^T \in \mathbb{R}^N$ şi $\mathbf{z} = \mathbf{x} + i\mathbf{y}$, adică $z_j = x_j + iy_j$, $\forall j \in \{1, \ldots, N\}$, şi $H : \mathbb{C}^N \times \mathbb{C}^N \to \mathbb{R}$, astfel încât $H(\mathbf{z}, \overline{\mathbf{z}}) = F(\mathbf{z})$, unde $\overline{\mathbf{z}} = (\overline{z_1}, \ldots, \overline{z_N})^T \in \mathbb{C}^N$ este conjugatul complex al vectorului $\mathbf{z} = (z_1, \ldots, z_N)^T \in \mathbb{C}^N$, adică $\overline{z_j}$ este conjugatul complex al lui $z_j$, $\forall j \in \{1, \ldots, N\}$.

În acest caz, avem relaţiile

$$\frac{\partial F}{\partial z_j^R}(\mathbf{z}) = \frac{\partial G}{\partial x_j}(\mathbf{x}, \mathbf{y}),$$

$$\frac{\partial F}{\partial z_j^I}(\mathbf{z}) = \frac{\partial G}{\partial y_j}(\mathbf{x}, \mathbf{y}),$$

$\forall z_j = z_j^R + iz_j^I = x_j + iy_j \in \mathbb{C}, \forall j \in \{1, \ldots, N\}$.

Pentru funcţia $H$, $\mathbb{R}$-derivata parţială în funcţie de $z_j$ este dată de

$$\frac{\partial H}{\partial z_j}(\mathbf{z}, \overline{\mathbf{z}}) = \frac{1}{2}\left(\frac{\partial G}{\partial x_j}(\mathbf{x}, \mathbf{y}) - i\frac{\partial G}{\partial y_j}(\mathbf{x}, \mathbf{y})\right) = \frac{1}{2}\left(\frac{\partial F}{\partial z_j^R}(\mathbf{z}) - i\frac{\partial F}{\partial z_j^I}(\mathbf{z})\right),$$

şi $\mathbb{R}$-derivata parţială complexă, sau $\overline{\mathbb{R}}$-derivata parţială în funcţie de $\overline{z_j}$ de

$$\frac{\partial H}{\partial \overline{z_j}}(\mathbf{z}, \overline{\mathbf{z}}) = \frac{1}{2}\left(\frac{\partial G}{\partial x_j}(\mathbf{x}, \mathbf{y}) + i\frac{\partial G}{\partial y_j}(\mathbf{x}, \mathbf{y})\right) = \frac{1}{2}\left(\frac{\partial F}{\partial z_j^R}(\mathbf{z}) + i\frac{\partial F}{\partial z_j^I}(\mathbf{z})\right),$$

$\forall z_j = z_j^R + iz_j^I = x_j + iy_j \in \mathbb{C}, \forall j \in \{1, \ldots, N\}$.

Vom defini acum operatorul gradient pentru funcţia $G$ prin

$$\frac{\partial G}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial G}{\partial x_1}(\mathbf{x}, \mathbf{y}), \ldots, \frac{\partial G}{\partial x_N}(\mathbf{x}, \mathbf{y})\right),$$

$$\frac{\partial G}{\partial \mathbf{y}}(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial G}{\partial y_1}(\mathbf{x}, \mathbf{y}), \ldots, \frac{\partial G}{\partial y_N}(\mathbf{x}, \mathbf{y})\right),$$

$$\frac{\partial G}{\partial (\mathbf{x}, \mathbf{y})}(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial G}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}), \frac{\partial G}{\partial \mathbf{y}}(\mathbf{x}, \mathbf{y})\right),$$

$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. Pentru funcţia $H$, operatorul gradient se defineşte prin

$$\frac{\partial H}{\partial \mathbf{z}}(\mathbf{z}, \overline{\mathbf{z}}) = \left(\frac{\partial H}{\partial z_1}(\mathbf{z}, \overline{\mathbf{z}}), \ldots, \frac{\partial H}{\partial z_N}(\mathbf{z}, \overline{\mathbf{z}})\right),$$

$$\frac{\partial H}{\partial \overline{\mathbf{z}}}(\mathbf{z}, \overline{\mathbf{z}}) = \left(\frac{\partial H}{\partial \overline{z_1}}(\mathbf{z}, \overline{\mathbf{z}}), \ldots, \frac{\partial H}{\partial \overline{z_N}}(\mathbf{z}, \overline{\mathbf{z}})\right),$$

$$\frac{\partial H}{\partial (\mathbf{z}, \overline{\mathbf{z}})}(\mathbf{z}, \overline{\mathbf{z}}) = \left(\frac{\partial H}{\partial \mathbf{z}}(\mathbf{z}, \overline{\mathbf{z}}), \frac{\partial H}{\partial \overline{\mathbf{z}}}(\mathbf{z}, \overline{\mathbf{z}})\right),$$

$\forall \mathbf{z} \in \mathbb{C}^N$.

În cele ce urmează, vom lucra cu o funcţie $E : \mathbb{C}^N \to \mathbb{R}$, care va fi funcţia de eroare. Prin abuz de notaţie, vom defini $E : \mathbb{R}^{2N} \to \mathbb{R}$, $E(\overset{\mathbb{R}}{\mathbf{z}}) = E(\mathbf{z})$, unde $\overset{\mathbb{R}}{\mathbf{z}} = (\mathbf{x}^T, \mathbf{y}^T)^T \in \mathbb{R}^{2N}$, $\mathbf{z} = (z_1, \ldots, z_N)^T \in \mathbb{C}^N$, $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$, $\mathbf{y} = (y_1, \ldots, y_N)^T \in \mathbb{R}^N$ şi $\mathbf{z} = \mathbf{x} + i\mathbf{y}$, adică $z_j = x_j + iy_j$, $\forall j \in \{1, \ldots, N\}$, şi $E : \mathbb{C}^{2N} \to \mathbb{R}$, astfel încât $E(\overset{\mathbb{C}}{\mathbf{z}}) = E(\mathbf{z})$, unde $\overset{\mathbb{C}}{\mathbf{z}} = (\mathbf{z}^T, \overline{\mathbf{z}}^T)^T \in \mathbb{C}^{2N}$, iar $\overline{\mathbf{z}} = (\overline{z_1}, \ldots, \overline{z_N})^T \in \mathbb{C}^N$ este conjugatul

complex al vectorului $\mathbf{z} = (z_1, \ldots, z_N)^T \in \mathbb{C}^N$, adică $\overline{z_j}$ este conjugatul complex al lui $z_j$, $\forall j \in \{1, \ldots, N\}$. Prin urmare, discriminarea între cele trei funcţii se va face în funcţie de argument: $\mathbf{z} \in \mathbb{C}^N$, $\overset{\mathbb{R}}{\mathbf{z}} \in \mathbb{R}^{2N}$, $\overset{\mathbb{C}}{\mathbf{z}} \in \mathbb{C}^{2N}$, pe care le vom considera ca fiind convenţii generale pentru orice vector: lui $\mathbf{z} \in \mathbb{C}^N$ îi corespunde vectorul părţilor reale şi al părţilor complexe $\overset{\mathbb{R}}{\mathbf{z}} \in \mathbb{R}^{2N}$, şi vectorul complex şi complex conjugat $\overset{\mathbb{C}}{\mathbf{z}} \in \mathbb{C}^{2N}$.

Acum putem scrie dezvoltarea în serie Taylor pentru funcţiile $E(\overset{\mathbb{R}}{\mathbf{z}})$ şi $E(\overset{\mathbb{C}}{\mathbf{z}})$:

$$E(\overset{\mathbb{R}}{\mathbf{z}} + \Delta\overset{\mathbb{R}}{\mathbf{z}}) = E(\overset{\mathbb{R}}{\mathbf{z}}) + \frac{\partial E}{\partial \overset{\mathbb{R}}{\mathbf{z}}}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\overset{\mathbb{R}}{\mathbf{z}} + \frac{1}{2}\Delta\overset{\mathbb{R}}{\mathbf{z}}^T \frac{\partial^2 E}{\partial \overset{\mathbb{R}}{\mathbf{z}} \partial \overset{\mathbb{R}}{\mathbf{z}}^T}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\overset{\mathbb{R}}{\mathbf{z}},$$

$$E(\overset{\mathbb{C}}{\mathbf{z}} + \Delta\overset{\mathbb{C}}{\mathbf{z}}) = E(\overset{\mathbb{C}}{\mathbf{z}}) + \frac{\partial E}{\partial \overset{\mathbb{C}}{\mathbf{z}}}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\overset{\mathbb{C}}{\mathbf{z}} + \frac{1}{2}\Delta\overset{\mathbb{C}}{\mathbf{z}}^H \frac{\partial^2 E}{\partial \overset{\mathbb{C}}{\mathbf{z}} \partial \overset{\mathbb{C}}{\mathbf{z}}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\overset{\mathbb{C}}{\mathbf{z}}.$$

Pentru funcţia $E(\mathbf{z})$, dezvoltarea în serie Taylor se poate scrie doar în funcţie de gradienţii definiţi pentru $E(\overset{\mathbb{R}}{\mathbf{z}})$ şi $E(\overset{\mathbb{C}}{\mathbf{z}})$:

$$\begin{aligned} E(\mathbf{z}+\Delta\mathbf{z}) &= E(\mathbf{z}) + \frac{\partial E}{\partial \mathbf{z}}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\mathbf{z} + \frac{\partial E}{\partial \overline{\mathbf{z}}}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\overline{\mathbf{z}} + \frac{1}{2}\Delta\mathbf{z}^H \frac{\partial^2 E}{\partial \mathbf{z}\partial \mathbf{z}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\mathbf{z} \\ &+ \frac{1}{2}\Delta\mathbf{z}^H \frac{\partial^2 E}{\partial \overline{\mathbf{z}}\partial \mathbf{z}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\overline{\mathbf{z}} + \frac{1}{2}\Delta\overline{\mathbf{z}}^H \frac{\partial^2 E}{\partial \mathbf{z}\partial \overline{\mathbf{z}}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\mathbf{z} + \frac{1}{2}\Delta\overline{\mathbf{z}}^H \frac{\partial^2 E}{\partial \overline{\mathbf{z}}\partial \overline{\mathbf{z}}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\overline{\mathbf{z}} \\ &= E(\mathbf{z}) + 2\left(\frac{\partial E}{\partial \mathbf{z}}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\mathbf{z}\right)^R + \left(\Delta\mathbf{z}^H \frac{\partial^2 E}{\partial \mathbf{z}\partial \mathbf{z}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\mathbf{z} + \Delta\mathbf{z}^H \frac{\partial^2 E}{\partial \overline{\mathbf{z}}\partial \mathbf{z}^H}(\overset{\mathbb{C}}{\mathbf{z}})\Delta\overline{\mathbf{z}}\right)^R, \end{aligned}$$

unde $\overset{\mathbb{C}}{\mathbf{z}} = (\mathbf{z}^T, \overline{\mathbf{z}}^T)^T$, şi

$$\begin{aligned} E(\mathbf{z}+\Delta\mathbf{z}) &= E(\mathbf{z}) + \frac{\partial E}{\partial \mathbf{x}}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\mathbf{x} + \frac{\partial E}{\partial \mathbf{y}}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\mathbf{y} + \frac{1}{2}\Delta\mathbf{x}^T \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{x}^T}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\mathbf{x} \\ &+ \frac{1}{2}\Delta\mathbf{y}^T \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{y}^T}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T \frac{\partial^2 E}{\partial \mathbf{y}\partial \mathbf{x}^T}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\mathbf{y} + \frac{1}{2}\Delta\mathbf{y}^T \frac{\partial^2 E}{\partial \mathbf{y}\partial \mathbf{y}^T}(\overset{\mathbb{R}}{\mathbf{z}})\Delta\mathbf{y}, \end{aligned}$$

unde $\overset{\mathbb{R}}{\mathbf{z}} = (\mathbf{x}^T, \mathbf{y}^T)^T$ şi $\mathbf{z} = \mathbf{x} + i\mathbf{y}$.

O altă proprietate importantă pe care o vom folosi este regula înlănţuirii:

$$\frac{\partial h_1(h_2)}{\partial z} = \frac{\partial h_1}{\partial h_2}\frac{\partial h_2}{\partial z} + \frac{\partial h_1}{\partial \overline{h_2}}\frac{\partial \overline{h_2}}{\partial z},$$

$$\frac{\partial h_1(h_2)}{\partial \overline{z}} = \frac{\partial h_1}{\partial h_2}\frac{\partial h_2}{\partial \overline{z}} + \frac{\partial h_1}{\partial \overline{h_2}}\frac{\partial \overline{h_2}}{\partial \overline{z}}.$$

## 3.2 Metoda gradient

În cele ce urmează, vom realiza o prezentare unitară a metodei gradient pentru reţelele de tip feedforward cu valori complexe folosind calculul Wirtinger, care nu ţine cont de faptul că funcţiile de activare sunt complexe complet (tratează numărul complex ca un întreg), sau complexe divizat (tratează separat părţile reală şi imaginară ale numărului complex), acesta reprezentând primul element de noutate al tezei. Deosebirea faţă de primul capitol o constituie folosirea $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatelor parţiale în locul derivatelor parţiale obişnuite în raport cu părţile reală şi imaginară. Particularizând rezultatele obţinute, regăsim formulele clasice deduse în literatura de specialitate pentru

metoda gradient complexă divizat (spre exemplu în [143]) şi respectiv pentru metoda gradient complexă complet (spre exemplu în [144]).

Să considerăm o reţea neuronală cu valori complexe care are $L$ straturi notate $\{1,\ldots,L\}$, unde $1$ este stratul de intrare, $L$ este stratul de ieşire, iar restul sunt straturi ascunse. Pentru această reţea, avem următoarea funcţie de eroare:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{c}(y_i^L - t_i)(\overline{y_i^L - t_i}),$$

unde $\mathbf{w}$ reprezintă vectorul tuturor ponderilor şi bias-urilor reţelei, $\mathbf{y}^L = (y_i^L)_{1\leq i\leq c}$ reprezintă ieşirile reţelei, şi $\mathbf{t} = (t_i)_{1\leq i\leq c}$ reprezintă ieşirile dorite ale reţelei. Am notat cu $E : \mathbb{C}^N \to \mathbb{R}$ funcţia de eroare care depinde de cele $N$ ponderi şi bias-uri ale reţelei. Pentru această funcţie, fie $E : \mathbb{C}^{2N} \to \mathbb{R}$ astfel încât $E(\overset{c}{\mathbf{w}}) = E(\mathbf{w})$, $\overset{c}{\mathbf{w}} = (\mathbf{w}^T, \overline{\mathbf{w}}^T)^T$ şi $E : \mathbb{R}^{2N} \to \mathbb{R}$, $E(\overset{\mathbb{R}}{\mathbf{w}}) = E(\mathbf{w})$, $\overset{\mathbb{R}}{\mathbf{w}} = ((\mathbf{w}^R)^T, (\mathbf{w}^I)^T)^T$. Prin urmare, pentru $E$ putem aplica $\mathbb{R}$-derivata parţială şi $\overline{\mathbb{R}}$-derivata parţială în funcţie de fiecare pondere şi bias al reţelei, pe care le-am definit în secţiunea anterioară.

Dacă notăm cu $w_{jk}^l$ ponderea dintre neuronul $j$ din stratul $l$ şi neuronul $k$ din stratul $l-1$, putem scrie actualizarea pentru ponderea $w_{jk}^l$ ca fiind

$$\Delta w_{jk}^l = -\varepsilon\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathbb{R}}{\mathbf{w}}) - i\varepsilon\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathbb{R}}{\mathbf{w}}) = -2\varepsilon\frac{\partial E}{\partial \overline{w_{jk}^l}}(\overset{c}{\mathbf{w}}).$$

Din această relaţie deducem că este suficient să calculăm $\overline{\mathbb{R}}$-derivata parţială a lui $E$ în funcţie de $\overline{w_{jk}^l}$ pentru a obţine actualizarea dorită a ponderii $w_{jk}^l$. Pentru acest calcul, facem următoarele notaţii:

$$s_j^l := \sum_k w_{jk}^l x_k^{l-1} + w_{j0}^l, \tag{3.2.1}$$

$$y_j^l := G^l(s_j^l), \tag{3.2.2}$$

unde $G^l : \mathbb{C} \to \mathbb{C}$ reprezintă funcţia de activare a stratului $l \in \{2,\ldots,L\}$, $\mathbf{x}^1 = (x_k^1)_{1\leq k\leq d}$ sunt intrările reţelei şi $x_k^l = y_k^l$, $\forall l \in \{2,\ldots,L-1\}$, $\forall k$. Funcţia de activare poate fi o funcţie complexă complet sau complexă divizat, după cum am arătat în Secţiunea 2.1.

Pentru că în Secţiunea 2.1 am dedus formula de actualizare pentru metoda gradient folosind derivate parţiale clasice, vom deduce din nou aici metoda gradient folosind calculul $\mathbb{CR}$.

Mai întâi, vom începe cu stratul de ieşire $L$, pentru care putem scrie regula înlănţuirii

$$\begin{aligned}\frac{\partial E}{\partial \overline{w_{jk}^L}} &= \frac{\partial E}{\partial s_j^L}\frac{\partial s_j^L}{\partial \overline{w_{jk}^L}} + \frac{\partial E}{\partial \overline{s_j^L}}\frac{\partial \overline{s_j^L}}{\partial \overline{w_{jk}^L}}\\ &= \frac{\partial E}{\partial \overline{s_j^L}}\overline{x_k^{L-1}},\end{aligned}$$

unde $\frac{\partial s_j^L}{\partial \overline{w_{jk}^L}} = 0$, deoarece $s_j^L$ nu depinde de $\overline{w_{jk}^L}$. Aplicând în continuare regula înlănţuirii, avem

$$\begin{aligned}\frac{\partial E}{\partial \overline{s_j^L}} &= \frac{\partial E}{\partial y_j^L}\frac{\partial y_j^L}{\partial \overline{s_j^L}} + \frac{\partial E}{\partial \overline{y_j^L}}\frac{\partial \overline{y_j^L}}{\partial \overline{s_j^L}} \\ &= \frac{1}{2}(\overline{y_j^L - t_j})\frac{\partial G^L(s_j^L)}{\partial \overline{s_j^L}} + \frac{1}{2}(y_j^L - t_j)\frac{\partial \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}},\end{aligned}$$

şi notând $\delta_j^l := \frac{\partial E}{\partial \overline{s_j^L}}$, avem că

$$\frac{\partial E}{\partial \overline{w_{jk}^l}} = \delta_j^L \overline{x_k^{L-1}},$$

unde

$$\delta_j^L = \frac{1}{2}(\overline{y_j^L - t_j})\frac{\partial G^L(s_j^L)}{\partial \overline{s_j^L}} + \frac{1}{2}(y_j^L - t_j)\frac{\partial \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}}.$$

Pentru un strat ascuns $l \in \{2, \ldots, L-1\}$, aplicarea regulii înlănţuirii ne dă

$$\begin{aligned}\frac{\partial E}{\partial \overline{w_{jk}^l}} &= \frac{\partial E}{\partial s_j^l}\frac{\partial s_j^l}{\partial \overline{w_{jk}^l}} + \frac{\partial E}{\partial \overline{s_j^l}}\frac{\partial \overline{s_j^l}}{\partial \overline{w_{jk}^l}} \\ &= \frac{\partial E}{\partial \overline{s_j^l}}\overline{x_k^{l-1}},\end{aligned}$$

deoarece $\frac{\partial s_j^l}{\partial \overline{w_{jk}^l}} = 0$, şi atunci

$$\frac{\partial E}{\partial \overline{s_j^l}} = \sum_r \left( \frac{\partial E}{\partial s_r^{l+1}}\frac{\partial s_r^{l+1}}{\partial \overline{s_j^l}} + \frac{\partial E}{\partial \overline{s_r^{l+1}}}\frac{\partial \overline{s_r^{l+1}}}{\partial \overline{s_j^l}} \right), \tag{3.2.3}$$

unde suma este luată după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite conexiuni.

Folosind din nou regula înlănţuirii, avem că

$$\begin{aligned}\frac{\partial s_r^{l+1}}{\partial \overline{s_j^l}} &= \frac{\partial s_r^{l+1}}{\partial y_j^l}\frac{\partial y_j^l}{\partial \overline{s_j^l}} + \frac{\partial s_r^{l+1}}{\partial \overline{y_j^l}}\frac{\partial \overline{y_j^l}}{\partial \overline{s_j^l}} \\ &= w_{rj}^{l+1}\frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}},\end{aligned}$$

$$\begin{aligned}\frac{\partial \overline{s_r^{l+1}}}{\partial \overline{s_j^l}} &= \frac{\partial \overline{s_r^{l+1}}}{\partial y_j^l}\frac{\partial y_j^l}{\partial \overline{s_j^l}} + \frac{\partial \overline{s_r^{l+1}}}{\partial \overline{y_j^l}}\frac{\partial \overline{y_j^l}}{\partial \overline{s_j^l}} \\ &= \overline{w_{rj}^{l+1}}\frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}},\end{aligned}$$

pentru că din nou avem că $\frac{\partial s_r^{l+1}}{\partial y_j^l} = \frac{\partial \overline{s_r^{l+1}}}{\partial y_j^l} = 0$. Dacă notăm $\delta_j^l := \frac{\partial E}{\partial \overline{s_j^l}}$, $\forall l \in \{2, \ldots, L-1\}$, ecuaţia (3.2.3) devine

$$\frac{\partial E}{\partial \overline{s_j^l}} = \sum_r \left( \delta_r^{l+1} \overline{w_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \overline{\delta_r^{l+1}} w_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right),$$

şi în final

$$\frac{\partial E}{\partial \overline{w_{jk}^l}} = \delta_j^l \overline{x_k^{l-1}}, \forall l \in \{2, \ldots, L-1\}.$$

unde

$$\delta_j^l = \sum_r \left( \delta_r^{l+1} \overline{w_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \overline{\delta_r^{l+1}} w_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right).$$

În concluzie,

$$\Delta w_{jk}^l = -2\varepsilon \delta_j^l \overline{x_k^{l-1}}, \forall l \in \{2, \ldots, L\}, \tag{3.2.4}$$

unde

$$\delta_j^l = \begin{cases} \sum_r \left( \delta_r^{l+1} \overline{w_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \overline{\delta_r^{l+1}} w_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right), & l \leq L-1 \\ \frac{1}{2}(y_j^l - t_j) \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \frac{1}{2}(\overline{y_j^l - t_j}) \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}}, & l = L \end{cases}. \tag{3.2.5}$$

Alternativ, mai putem scrie, înlocuind $2\delta_j^l$ cu $\delta_j^l$, regula de actualizare sub forma

$$\Delta w_{jk}^l = -\varepsilon \delta_j^l \overline{x_k^{l-1}}, \forall l \in \{2, \ldots, L\}, \tag{3.2.6}$$

unde

$$\delta_j^l = \begin{cases} \left( \sum_m \overline{w_{mj}^{l+1}} \delta_m^{l+1} \right)^R \left( \frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} + i \frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}} \right) \\ + \left( \sum_m \overline{w_{mj}^{l+1}} \delta_m^{l+1} \right)^I \left( \frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}} + i \frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}} \right), & l \leq L-1 \\ (y_j^{l,R} - t_j^R) \left( \frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} + i \frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}} \right) \\ + (y_j^{l,I} - t_j^I) \left( \frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}} + i \frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}} \right), & l = L \end{cases}.$$

## 3.3 Metode gradient îmbunătăţite

Una dintre cele mai simple clase de algoritmi care au o performanţă mai bună decât metoda gradient clasică în domeniul real este aşa-numita clasă a *metodelor gradient îmbunătăţite*. Algoritmii din această clasă cresc performanţa metodei gradient făcând diferite modificări ad hoc algoritmului original. Există o literatură vastă care prezintă astfel de algoritmi, din care am ales unele dintre cele mai cunoscute, robuste şi bine fundamentate teoretic metode de optimizare pentru a le extinde în domeniul complex. Acesta este un prim pas făcut în direcţia găsirii unor algoritmi de optimizare a funcţiei de eroare care să aibă performanţe mai bune decât clasica metodă gradient. În cele ce urmează vom prezenta metodele gradient îmbunătăţite pentru reţele cu valori complexe, ca în articolul nostru [215].

### 3.3.1 Metoda gradient cu moment

Cu notaţiile din secţiunea precedentă, regula de actualizare pentru *metoda gradient cu moment* este

$$\begin{aligned}\Delta w_{jk}^{l}(t) &= -\varepsilon\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) - i\varepsilon\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) + \mu\Delta w_{jk}^{l}(t-1)\\ &= -2\varepsilon\frac{\partial E}{\partial \overline{w_{jk}^{l}}}(\overset{c}{\mathbf{w}}(t)) + \mu\Delta w_{jk}^{l}(t-1)\end{aligned} \tag{3.3.1}$$

unde parametrul $\mu$ reprezintă *momentul*, şi este, la fel ca rata de învăţare $\varepsilon$, un număr real. Ideea din spatele acestei modificări a metodei gradient clasice este de a da o anumită inerţie mişcării în spaţiul ponderilor pentru a netezi posibilele oscilaţii ale unor paşi succesivi din metoda gradient. Pentru reţelele cu valori complexe, această metodă a fost discutată într-unul dintre primele articole pe acest subiect, şi anume [143].

### 3.3.2 Metoda quickprop

Să presupunem că avem o funcţie $f$ definită în planul real, al cărei minim dorim să-l găsim. Pentru aceasta, trebuie să căutăm un $x$ astfel încât $f'(x) = 0$. O metodă pentru a face acest lucru, în ipoteza că $f$ este convexă, este să-l calculăm pe $x$ iterativ, folosind metoda Newton:

$$x(t+1) = x(t) + \Delta x(t),$$

unde

$$\Delta x(t) = -\frac{f'(x(t))}{f''(x(t))}. \tag{3.3.2}$$

Dacă nu putem sau nu dorim să calculăm a doua derivată a lui $f$, putem să o înlocuim printr-o aproximare de tip secantă:

$$f''(x(t)) \simeq \frac{f'(x(t)) - f'(x(t-1))}{x(t) - x(t-1)}. \tag{3.3.3}$$

Înlocuind această expresie cu a doua derivată în (3.3.2), avem

$$\Delta x(t) = \frac{f'(x(t))}{f'(x(t-1)) - f'(x(t))}\Delta x(t-1).$$

Folosind formula de mai sus pentru actualizarea ponderilor într-o reţea neuronală cu valori complexe, unde funcţia $f$ este funcţia de eroare $E$, avem, pentru o anumită pondere $w_{jk}^{l}$ definită ca mai sus, următoarea regulă de actualizare:

$$\Delta w_{jk}^{l,R}(t) = \frac{\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t))}{\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) - \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1))}\Delta w_{jk}^{l,R}(t-1). \tag{3.3.4}$$

$$\Delta w_{jk}^{l,I}(t) = \frac{\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t))}{\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) - \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1))}\Delta w_{jk}^{l,I}(t-1). \tag{3.3.5}$$

Această schemă de actualizare este cunoscută sub numele de algoritmul *quickprop*, şi a fost pentru prima dată propus de către Fahlman în [80]. Expresia de actualizare este exact aceeaşi în cazul complex ca în cazul real, fiind o aplicare pe componente a aceleaşi formule de actualizare. Pornirea algoritmului necesită doar un singur pas făcut cu metoda gradient, pentru a calcula pasul de actualizare în prima epocă. Euristica din spatele algoritmului este aceea că pasul de actualizare aproximează funcţia de eroare în raport cu fiecare pondere printr-un polinom de gradul doi, pentru care formula (3.3.3) este exactă. Această aproximare presupune că fiecare pondere este independentă de toate celelalte şi că polinomul de gradul doi are un minim, şi nu un maxim. De asemenea, o limită a valorii maxime a pasului de actualizare trebuie impusă pentru a evita probleme în cazul în care numitorul din (3.3.4) devine foarte mic.

### 3.3.3 Metoda RPROP

Pentru a elimina influenţa potenţial negativă a mărimii derivatei parţiale asupra pasului de actualizare, Riedmiller şi Braun [226], au conceput algoritmul *resilient backpropagation algorithm*, sau prescurtat *RPROP*. Ideea de bază a algoritmului este de a înlocui derivata parţială din formula de actualizare a ponderilor, şi de a folosi doar semnul ei pentru a calcula o cantitate notată $\Delta_{jk}^{l}$, care va fi apoi folosită pentru actualizarea ponderii $w_{jk}^{l}$. Pentru că suntem în planul complex, trebuie să tratăm părţile reală şi imaginară separat. Astfel, formulele pentru calcularea cantităţilor $\Delta_{jk}^{l}$ sunt:

$$\Delta_{jk}^{l,R}(t) = \begin{cases} \eta^{+}\Delta_{jk}^{l,R}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \eta^{-}\Delta_{jk}^{l,R}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ \Delta_{jk}^{l,R}(t-1) & \text{altfel} \end{cases},$$

$$\Delta_{jk}^{l,I}(t) = \begin{cases} \eta^{+}\Delta_{jk}^{l,I}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \eta^{-}\Delta_{jk}^{l,I}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ \Delta_{jk}^{l,I}(t-1) & \text{altfel} \end{cases}.$$

Aceste formule depind de parametrii reali $\eta^{+}$ şi $\eta^{-}$, care trebuie să satisfacă

$$0 < \eta^{-} < 1 < \eta^{+},$$

şi sunt de obicei folosiţi cu valorile $\eta^{-} = 0.5$ şi $\eta^{+} = 1.2$. Aceste formule spun că de fiecare dată când părţile reală sau imaginară ale derivatei parţiale a funcţiei de eroare în raport cu ponderea $w_{jk}^{l}$ îşi schimbă semnul, ceea ce indică faptul că ultima actualizare a fost prea mare şi că algoritmul a sărit peste un minim local, cantităţile $\Delta_{jk}^{l,R}$ şi $\Delta_{jk}^{l,I}$ sunt micşorate cu un factor de $\eta^{-}$. Dacă acele derivate parţiale au acelaşi semn, aceste cantităţi sunt mărite cu un factor de $\eta^{+}$, pentru a accelera convergenţa.

Odată ce am calculat valorile lui $\Delta_{jk}^{l,R}$ şi $\Delta_{jk}^{l,I}$ , putem scrie formulele pentru actualizarea părţilor reală şi imaginară ale ponderii $w_{jk}^{l}$:

$$\Delta w_{jk}^{l,R}(t) = \begin{cases} -\Delta_{jk}^{l,R}(t), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \Delta_{jk}^{l,R}(t), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ 0, & \text{altfel} \end{cases},$$

$$\Delta w_{jk}^{l,I}(t) = \begin{cases} -\Delta_{jk}^{l,I}(t), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \Delta_{jk}^{l,I}(t), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ 0, & \text{altfel} \end{cases}.$$

Se poate vedea că regula de actualizare este simplă: dacă derivata este pozitivă, ceea ce înseamnă o creştere a erorii, ponderea este micşorată cu $\Delta_{jk}^{l,R}$ sau $\Delta_{jk}^{l,I}$, iar dacă derivata este negativă, aceste valori sunt adunate la partea reală respectiv imaginară a pasului de actualizare a ponderii $w_{jk}^{l}$. La începutul algoritmului trebuie să dăm o valoare pentru cantităţile $\Delta_{jk}^{l,R}(0)$ şi $\Delta_{jk}^{l,I}(0)$, iar această valoare o luăm ca fiind $0.07$.

În contextul reţelelor neuronale cu valori complexe, acest algoritm a fost propus pentru rezolvarea problemei egalizării transmisiilor de date GSM în articolul [141]. Acesta a adus astfel atât o contribuţie teoretică prin extinderea algoritmului în domeniul complex, cât şi una aplicativă, folosind reţelele neuronale de tip feedforward cu valori complexe în domeniul egalizării semnalelor de date GSM.

### 3.3.4 Metoda delta-bar-delta

O altă cale de a îmbunătăţi performanţa metodei gradient este de a introduce o rată de învăţare diferită pentru fiecare pondere a reţelei, şi de a concepe o procedură pentru actualizarea acestor rate de învăţare pe parcursul antrenării reţelei. Metoda gradient devine în acest caz

$$\Delta w_{jk}^{l,R}(t) = -\varepsilon_{jk}^{l,R}(t)\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)),$$

$$\Delta w_{jk}^{l,I}(t) = -\varepsilon_{jk}^{l,I}(t)\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)),$$

unde fiecare rată de învăţare $\varepsilon_{jk}^{l}(t)$ depinde de ponderea $w_{jk}^{l}$ şi de epoca $t$. Dintru început, facem observaţia că rata de învăţare $\varepsilon$ este un număr real în metoda gradient pentru reţele cu valori complexe, în vreme ce în acest nou algoritm, rata de învăţare este un număr complex. Formulele pentru actualizarea părţilor reală şi imaginară ale fiecărei rate de învăţare $\varepsilon_{jk}^{l}(t)$ sunt:

$$\varepsilon_{jk}^{l,R}(t) = \begin{cases} \kappa + \varepsilon_{jk}^{l,R}(t-1), & \text{dacă } \frac{\partial \bar{E}}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \eta^{-}\varepsilon_{jk}^{l,R}(t-1), & \text{dacă } \frac{\partial \bar{E}}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ \varepsilon_{jk}^{l,R}(t-1) & \text{altfel} \end{cases},$$

$$\varepsilon_{jk}^{l,I}(t) = \begin{cases} \kappa + \varepsilon_{jk}^{l,I}(t-1), & \text{dacă } \frac{\partial \bar{E}}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \eta^{-}\varepsilon_{jk}^{l,I}(t-1), & \text{dacă } \frac{\partial \bar{E}}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ \varepsilon_{jk}^{l,I}(t-1) & \text{altfel} \end{cases},$$

unde am făcut notaţiile

$$\frac{\partial \bar{E}}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) = (1-\theta)\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) + \theta\frac{\partial \bar{E}}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)),$$

$$\frac{\partial \bar{E}}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) = (1-\theta)\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) + \theta\frac{\partial \bar{E}}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)).$$

Parametrul $\eta^-$ trebuie să satisfacă relaţia $0 < \eta^- < 1$, şi este de obicei considerat, ca în algoritmul RPROP, ca fiind $0.5$. Am luat de asemenea $\kappa = 10^{-6}$ şi $\theta = 0.7$ în experimentele noastre, dar aceste valori sunt de obicei determinate empiric.

Ideea de bază a acestei metode este următoarea: dacă două derivate parţiale consecutive, mai precis părţile lor reale şi respectiv imaginare, au acelaşi semn, atunci rata de învăţare este mărită cu o constantă de valoare mică $\kappa$ pentru a accelera învăţarea. O schimbare a semnului părţilor reale şi respectiv imaginare ale acestor derivate parţiale arată că s-a trecut peste minimul local, ceea ce înseamnă că pasul de învăţare a fost prea mare. În consecinţă, rata de învăţare este micşorată prin înmulţire cu un factor pozitiv subunitar $\eta^-$. Pentru a asigura convergenţa, derivata parţială anterioară $\frac{\partial E}{\partial w_{jk}^{l}}(t-1)$ a fost înlocuită cu media ponderată exponenţial între derivata parţială curentă şi cea anterioară, având pe $\theta$ ca bază a acestei medii şi epoca în calitate de exponent. Această medie ponderată a fost notată cu $\frac{\partial \bar{E}}{\partial w_{jk}^{l,R}}(t)$ şi $\frac{\partial \bar{E}}{\partial w_{jk}^{l,I}}(t)$, şi a fost definită mai sus.

Algoritmul pe care tocmai l-am descris a fost propus de Jacobs în [133], şi este numit algoritmul *delta-bar-delta*, deoarece în acel articol, autorul a notat derivata parţială $\frac{\partial E}{\partial w_{jk}^{l}}(t)$ cu $\delta$ (delta), şi media ponderată $\frac{\partial \bar{E}}{\partial w_{jk}^{l}}(t)$ cu $\bar{\delta}$ (delta-bar).

### 3.3.5 Metoda SuperSAB

Ultimul algoritm din seria metodelor gradient îmbunătăţite se numeşte *SuperSAB* şi a fost propus de Tollenaere în [250]. Este foarte asemănător cu algoritmul delta-bar-delta, în aceea că pentru fiecare pondere a reţelei este folosită o rată de învăţare diferită. Prin urmare, ca în cazul algoritmului anterior, regula de actualizare este

$$\Delta w_{jk}^{l,R}(t) = -\varepsilon_{jk}^{l,R}(t)\frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)),$$

$$\Delta w_{jk}^{l,I}(t) = -\varepsilon_{jk}^{l,I}(t)\frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)),$$

cu excepţia faptului că aici rata de învăţare $\varepsilon_{jk}^{l}(t)$ corespunzătoare ponderii $w_{jk}^{l}$ şi epocii $t$ are o formulă de actualizare diferită. De fapt, se poate spune că algoritmul SuperSAB este o combinaţie între algoritmii RPROP şi delta-bar-delta, pentru că formulele pentru actualizarea părţilor reală şi imaginară ale lui $\varepsilon_{jk}^{l}(t)$ sunt:

$$\varepsilon_{jk}^{l,R}(t) = \begin{cases} \eta^+\varepsilon_{jk}^{l,R}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \eta^-\varepsilon_{jk}^{l,R}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,R}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ \varepsilon_{jk}^{l,R}(t-1) & \text{altfel} \end{cases},$$

$$\varepsilon_{jk}^{l,I}(t) = \begin{cases} \eta^+\varepsilon_{jk}^{l,I}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) > 0 \\ \eta^-\varepsilon_{jk}^{l,I}(t-1), & \text{dacă } \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t-1)) \times \frac{\partial E}{\partial w_{jk}^{l,I}}(\overset{\mathcal{R}}{\mathbf{w}}(t)) < 0 \\ \varepsilon_{jk}^{l,I}(t-1) & \text{altfel} \end{cases},$$

unde parametrii $\eta^+$ şi $\eta^-$ trebuie să satisfacă

$$0 < \eta^- < 1 < \eta^+,$$

exact ca în algoritmul resilient backpropagation. Datorită acestei asemănări, am luat în experimentele noastre aceleaşi valori pentru aceşti parametri ca în cazul algoritmului RPROP, şi anume $\eta^- = 0.5$ şi $\eta^+ = 1.2$.

Ideea este destul de uşor de văzut: în loc să adunăm un termen la rata de învăţare dacă derivatele parţiale au acelaşi semn, facem o înmulţire cu un factor pozitiv supraunitar $\eta^+$ a acesteia, ca în cazul algoritmului RPROP. În acest caz, ultimele două derivate parţiale au fost luate în considerare cu valorile iniţiale, fără a folosi media ponderată, ca pentru algoritmul delta-bar-delta.

Toţi aceşti algoritmi sunt strategii de adaptare locală, în care pentru actualizarea fiecărei ponderi este folosită informaţie care este proprie doar acelei ponderi, fără a avea o cunoaştere globală a întregii stări a reţelei, aşa cum fac strategiile de adaptare globală. Deşi aceste strategii folosesc mai puţină informaţie, şi operaţiile de actualizare se fac mai uşor şi mai repede, pentru multe aplicaţii s-a dovedit că ele au o performanţă mai bună decât strategiile globale.

Una dintre principalele diferenţe între primul algoritm şi următorii patru, este aceea că regula de actualizare (3.3.1) depinde de numărul complex ca întreg, şi nu separat de părţile reală şi imaginară ca regulile de actualizare ale celorlalţi patru algoritmi. Motivul principal al acestei diferenţe este că aceşti algoritmi fac presupuneri despre valoarea derivatei parţiale a funcţiei de eroare în raport cu ponderea care este actualizată, presupuneri care implică realizarea de comparaţii. Este cunoscut că, spre deosebire de numerele reale, numerele complexe nu au o ordonare naturală, şi prin urmare nu există un analog al inegalităţilor de numere reale între numere complexe. În consecinţă, pentru quickprop, RPROP, delta-bar-delta şi SuperSAB, trebuie să dăm reguli de actualizare diferite pentru părţile reală şi imaginară ale fiecărei ponderi, care depind de presupuneri făcute asupra părţilor reală şi respectiv imaginară ale derivatei parţiale. Cu excepţia acestui fapt, putem observa că regulile de actualizare arată exact la fel în domeniul complex ca în domeniul real, şi nu pun niciun fel de probleme de implementare.

## 3.4 Metoda gradienţilor conjugaţi

Metodele gradienţilor conjugaţi sunt un caz particular dintr-o clasă mai largă de *algoritmi de căutare liniară*. Pentru a minimiza funcţia de eroare a unei reţele neuronale, sunt necesari mai mulţi paşi prin spaţiul valorilor ponderilor pentru a găsi acea valoare a ponderilor pentru care minimul funcţiei de eroare este atins. Fiecare pas este determinat de o *direcţie de căutare* şi de un număr real care ne spune cât de mult trebuie să ne deplasăm în acea direcţie. În metoda gradient clasică, direcţia de căutare este aceea a gradientului cu semn schimbat, iar numărul real este rata de învăţare. În cazul general, putem considera o anumită direcţie de căutare, şi apoi putem determina minimul funcţiei de eroare în acea direcţie, obţinând în acest fel acel număr real care ne spune cât de mult trebuie să ne deplasăm în direcţia de căutare. Acesta reprezintă *algoritmul de căutare liniară*, şi se află la baza unei familii de metode care au o performanţă mai bună decât metoda gradient. Deducerea algoritmilor gradienţilor conjugaţi prezentată mai jos, urmăreşte în principal referinţa [46], care la rândul ei urmăreşte referinţa [140], unul dintre primele articole în care se aplică aceşti algoritmi pentru antrenarea reţelelor neuronale, un altul fiind [64], şi este făcută ca în articolul nostru [217].

Să presupunem că avem o reţea neuronală cu valori complexe care are funcţia de eroare notată cu $E$, şi un vector al ponderilor $N$-dimensional notat cu $\mathbf{w}$. Trebuie să găsim vectorul ponderilor notat prin $\overset{c}{\mathbf{w}}^*$ care minimizează funcţia $E(\overset{c}{\mathbf{w}})$, unde $\overset{c}{\mathbf{w}} = (\mathbf{w}^T, \overline{\mathbf{w}}^T)^T \in \mathbb{C}^{2N}$. Să presupunem că facem paşi iterativi prin spaţiul ponderilor pentru a găsi valoarea lui $\overset{c}{\mathbf{w}}^*$, sau o foarte bună aproximare a ei. Mai mult, să presupunem că la pasul $k$ în iteraţie, vrem ca direcţia de căutare să fie $\overset{c}{\mathbf{p}}_k$, unde $\overset{c}{\mathbf{p}}_k$ este evident un vector $2N$-dimensional. În acest caz, următoarea valoare pentru vectorul ponderilor este dată prin

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k + \lambda_k \overset{c}{\mathbf{p}}_k,$$

unde parametrul $\lambda_k$ este un număr real care ne spune cât de mult în direcţia $\overset{c}{\mathbf{p}}_k$ vrem să mergem, ceea ce înseamnă că $\lambda_k$ trebuie ales pentru a minimiza funcţia

$$E(\lambda) = E(\overset{c}{\mathbf{w}}_k + \lambda \overset{c}{\mathbf{p}}_k).$$

Aceasta este o funcţie reală de o variabilă reală, ceea ce înseamnă că minimul ei este atins când

$$\frac{\partial E(\lambda)}{\partial \lambda} = \frac{\partial E(\overset{c}{\mathbf{w}}_k + \lambda \overset{c}{\mathbf{p}}_k)}{\partial \lambda} = 0.$$

Folosind regula înlănţuirii, putem scrie că

$$\begin{aligned}
\frac{\partial E(\overset{c}{\mathbf{w}}_k + \lambda \overset{c}{\mathbf{p}}_k)}{\partial \lambda} &= \sum_{j=1}^{N} \frac{\partial E(\overset{c}{\mathbf{w}}_k + \lambda \overset{c}{\mathbf{p}}_k)}{\partial (w_{j,k} + \lambda p_{j,k})} \frac{\partial (w_{j,k} + \lambda p_{j,k})}{\partial \lambda} \\
&\quad + \sum_{j=1}^{N} \frac{\partial E(\overset{c}{\mathbf{w}}_k + \lambda \overset{c}{\mathbf{p}}_k)}{\partial \overline{(w_{j,k} + \lambda p_{j,k})}} \frac{\partial \overline{(w_{j,k} + \lambda p_{j,k})}}{\partial \lambda} \\
&= \sum_{j=1}^{N} \frac{\partial E(\overset{c}{\mathbf{w}}_{k+1})}{\partial w_{j,k+1}} p_{j,k} + \frac{\partial E(\overset{c}{\mathbf{w}}_{k+1})}{\partial \overline{w_{j,k+1}}} \overline{p_{j,k}} \\
&= \langle \nabla E(\overset{c}{\mathbf{w}}_{k+1}), \overset{c}{\mathbf{p}}_k \rangle, \qquad (3.4.1)
\end{aligned}$$

unde $\langle \overset{c}{\mathbf{a}}, \overset{c}{\mathbf{b}} \rangle$ este *produsul scalar euclidian* din $\mathbb{C}^{2N}$, dat prin

$$\langle \overset{c}{\mathbf{a}}, \overset{c}{\mathbf{b}} \rangle = \langle \begin{pmatrix} \mathbf{a} \\ \overline{\mathbf{a}} \end{pmatrix}, \begin{pmatrix} \mathbf{b} \\ \overline{\mathbf{b}} \end{pmatrix} \rangle = \overset{c}{\mathbf{a}}^H \overset{c}{\mathbf{b}} = \sum_{j=1}^{N} a_j \overline{b_j} + \overline{a_j} b_j,$$

pentru orice $\overset{c}{\mathbf{a}}, \overset{c}{\mathbf{b}} \in \mathbb{C}^{2N}$, şi gradientul $\nabla E(\overset{c}{\mathbf{w}}_{k+1})$ este definit prin

$$\nabla E(\overset{c}{\mathbf{w}}) := \left( \frac{\partial E}{\partial \overset{c}{\mathbf{w}}} (\overset{c}{\mathbf{w}}) \right)^H.$$

Prin urmare, dacă notăm $\mathbf{g}(\overset{c}{\mathbf{w}}) := \nabla E(\overset{c}{\mathbf{w}})$, atunci (3.4.1) poate fi scrisă sub forma

$$\langle \mathbf{g}(\overset{c}{\mathbf{w}}_{k+1}), \overset{c}{\mathbf{p}}_k \rangle = 0, \qquad (3.4.2)$$

ceea ce arată că vectorii $\mathbf{g}(\overset{c}{\mathbf{w}}_{k+1})$ şi $\overset{c}{\mathbf{p}}_k$ sunt ortogonali în $\mathbb{C}^{2N}$.

Următoarea direcţie de căutare $\overset{c}{\mathbf{p}}_{k+1}$ este aleasă astfel încât componenta gradientului paralelă cu direcţia de căutare anterioară $\overset{c}{\mathbf{p}}_k$ rămâne zero. În consecinţă, avem că

$$\langle \mathbf{g}(\overset{c}{\mathbf{w}}_{k+1} + \lambda \overset{c}{\mathbf{p}}_{k+1}), \overset{c}{\mathbf{p}}_k \rangle = 0.$$

Din dezvoltarea în serie Taylor până la gradul întâi, avem că

$$\mathbf{g}(\overset{c}{\mathbf{w}}_{k+1} + \lambda \overset{c}{\mathbf{p}}_{k+1}) = \mathbf{g}(\overset{c}{\mathbf{w}}_{k+1}) + \nabla \mathbf{g}(\overset{c}{\mathbf{w}}_{k+1})^H \lambda \overset{c}{\mathbf{p}}_{k+1},$$

şi apoi, dacă ţinem cont de (3.4.2), obţinem că

$$\lambda \langle \nabla \mathbf{g}(\overset{c}{\mathbf{w}}_{k+1})^H \overset{c}{\mathbf{p}}_{k+1}, \overset{c}{\mathbf{p}}_k \rangle = 0,$$

ceea ce este echivalent cu

$$\langle \mathbf{H}^H(\overset{c}{\mathbf{w}}_{k+1}) \overset{c}{\mathbf{p}}_{k+1}, \overset{c}{\mathbf{p}}_k \rangle = 0,$$

sau, mai departe, cu

$$\langle \overset{c}{\mathbf{p}}_{k+1}, \mathbf{H}(\overset{c}{\mathbf{w}}_{k+1}) \overset{c}{\mathbf{p}}_k \rangle = 0, \tag{3.4.3}$$

unde am notat prin $\mathbf{H}(\overset{c}{\mathbf{w}}_{k+1})$ hessiana funcţiei de eroare $E(\overset{c}{\mathbf{w}})$, deoarece $\mathbf{g}(\overset{c}{\mathbf{w}}) = \nabla E(\overset{c}{\mathbf{w}})$, şi deci $\nabla \mathbf{g}(\overset{c}{\mathbf{w}}) = \frac{\partial^2 E}{\partial \overset{c}{\mathbf{w}}^H \partial \overset{c}{\mathbf{w}}}(\overset{c}{\mathbf{w}}) = \frac{\partial^2 E}{\partial \overset{c}{\mathbf{w}} \partial \overset{c}{\mathbf{w}}^H}(\overset{c}{\mathbf{w}})$ este hessiana.

Direcţiile de căutare care satisfac ecuaţia (3.4.3) se numesc *direcţii conjugate*. Metoda direcţiilor conjugate construieşte direcţiile de căutare $\overset{c}{\mathbf{p}}_k$, astfel încât fiecare nouă direcţie este conjugată cu toate cele anterioare.

### 3.4.1 Metoda gradienţilor conjugaţi liniară

În continuare, vom explica metoda gradienţilor conjugaţi liniară. Pentru aceasta, să considerăm o funcţie de eroare de forma

$$E(\overset{c}{\mathbf{w}}) = E_0 + \mathbf{b}^H \overset{c}{\mathbf{w}} + \frac{1}{2} \overset{c}{\mathbf{w}}^H \mathbf{H} \overset{c}{\mathbf{w}}, \tag{3.4.4}$$

unde $\mathbf{b}$ şi $\mathbf{H}$ sunt constante, şi matricea $\mathbf{H}$ este presupusă a fi pozitiv definită. Gradientul acestei funcţii este dat de

$$\mathbf{g}(\overset{c}{\mathbf{w}}) = \mathbf{b} + \mathbf{H} \overset{c}{\mathbf{w}}. \tag{3.4.5}$$

Vectorul ponderilor $\overset{c}{\mathbf{w}}^*$ care minimizează funcţia $E(\overset{c}{\mathbf{w}})$ satisface ecuaţia

$$\mathbf{b} + \mathbf{H} \overset{c}{\mathbf{w}}^* = 0. \tag{3.4.6}$$

După cum am văzut mai devreme din ecuaţia (3.4.3), o mulţime de $2N$ vectori nenuli $\{\overset{c}{\mathbf{p}}_1, \overset{c}{\mathbf{p}}_2, \ldots, \overset{c}{\mathbf{p}}_{2N}\} \subset \mathbb{C}^{2N}$ se zice a fi *conjugată faţă de matricea pozitiv definită* $\mathbf{H}$ dacă şi numai dacă

$$\langle \overset{c}{\mathbf{p}}_i, \mathbf{H} \overset{c}{\mathbf{p}}_j \rangle = 0, \forall i \neq j. \tag{3.4.7}$$

Este uşor de văzut că în aceste condiţii, mulţimea de $2N$ vectori este de asemenea liniar independentă, ceea ce înseamnă că aceşti vectori formează o bază în $\mathbb{C}^{2N}$. Dacă începem din punctul iniţial notat $\overset{c}{\mathbf{w}}_1$ şi vrem să găsim valoarea lui $\overset{c}{\mathbf{w}}^*$ care minimizează

funcţia de eroare dată în (3.4.4), luând în considerare remarcile anterioare, putem scrie că

$$\overset{c}{\mathbf{w}}^{*} - \overset{c}{\mathbf{w}}_1 = \sum_{i=1}^{2N} \alpha_i \overset{c}{\mathbf{p}}_i. \tag{3.4.8}$$

Acum, dacă punem

$$\overset{c}{\mathbf{w}}_k = \overset{c}{\mathbf{w}}_1 + \sum_{i=1}^{k-1} \alpha_i \overset{c}{\mathbf{p}}_i, \tag{3.4.9}$$

atunci (3.4.8) poate fi scrisă în formă iterativă, astfel

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k, \tag{3.4.10}$$

ceea ce înseamnă că valoarea lui $\overset{c}{\mathbf{w}}^{*}$ poate fi determinată în cel mult $2N$ paşi pentru funcţia de eroare (3.4.4), folosind iteraţia de mai sus. Mai trebuie să determinăm parametrii reali $\alpha_k$ care ne spun cât de mult trebuie să avansăm în fiecare dintre cele $2N$ direcţii conjugate $\overset{c}{\mathbf{p}}_k$.

Pentru aceasta, înmulţim ecuaţia (3.4.8) cu $\mathbf{H}$ la stânga, şi luăm produsul scalar euclidian din $\mathbb{C}^{2N}$ cu $\overset{c}{\mathbf{p}}_k$. Ţinând cont de ecuaţia (3.4.6), obţinem că

$$-\langle \overset{c}{\mathbf{p}}_k, \mathbf{b} + \mathbf{H}\overset{c}{\mathbf{w}}_1 \rangle = \sum_{i=1}^{N} \alpha_i \langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_i \rangle.$$

Dar, pentru că direcţiile $\overset{c}{\mathbf{p}}_k$ sunt conjugate faţă de matricea $\mathbf{H}$, avem din (3.4.7) că $\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_i \rangle = 0, \forall i \neq k$, şi prin urmare ecuaţia de mai sus dă următoarea expresie pentru calculul lui $\alpha_k$:

$$\alpha_k = -\frac{\langle \overset{c}{\mathbf{p}}_k, \mathbf{b} + \mathbf{H}\overset{c}{\mathbf{w}}_1 \rangle}{\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_k \rangle}. \tag{3.4.11}$$

Acum, dacă înmulţim ecuaţia (3.4.9) cu $\mathbf{H}$ la stânga şi luăm acelaşi produs scalar cu $\overset{c}{\mathbf{p}}_k$, avem că:

$$\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{w}}_k \rangle = \langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{w}}_1 \rangle + \sum_{i=1}^{k-1} \alpha_i \langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_i \rangle,$$

sau, ţinând cont că $\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_i \rangle = 0, \forall i \neq k$, avem că

$$\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{w}}_k \rangle = \langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{w}}_1 \rangle,$$

şi astfel relaţia (3.4.11) pentru calculul lui $\alpha_k$ devine:

$$\alpha_k = -\frac{\langle \overset{c}{\mathbf{p}}_k, \mathbf{b} + \mathbf{H}\overset{c}{\mathbf{w}}_k \rangle}{\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_k \rangle} = -\frac{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_k \rangle}{\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_k \rangle}, \tag{3.4.12}$$

unde $\overset{c}{\mathbf{g}}_k := \mathbf{g}(\overset{c}{\mathbf{w}}_k) = \mathbf{b} + \mathbf{H}\overset{c}{\mathbf{w}}_k$, după cum rezultă din relaţia (3.4.5).

În final, mai trebuie doar să construim direcţiile mutual conjugate $\overset{c}{\mathbf{p}}_k$. Pentru aceasta, prima direcţie este iniţializată cu gradientul cu semn schimbat al funcţiei de

eroare în punctul iniţial $\overset{c}{\mathbf{w}}_1$, i.e. $\overset{c}{\mathbf{p}}_1 = -\overset{c}{\mathbf{g}}_1$. Avem următoarea regulă de actualizare pentru direcţiile conjugate:

$$\overset{c}{\mathbf{p}}_{k+1} = -\overset{c}{\mathbf{g}}_{k+1} + \beta_k \overset{c}{\mathbf{p}}_k. \tag{3.4.13}$$

Luând produsul scalar cu $\mathbf{H}\overset{c}{\mathbf{p}}_k$, şi impunând condiţia de conjugare $\langle \overset{c}{\mathbf{p}}_{k+1}, \mathbf{H}\overset{c}{\mathbf{p}}_k \rangle = 0$, obţinem că

$$\beta_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \mathbf{H}\overset{c}{\mathbf{p}}_k \rangle}{\langle \overset{c}{\mathbf{p}}_k, \mathbf{H}\overset{c}{\mathbf{p}}_k \rangle}. \tag{3.4.14}$$

Se poate arăta uşor prin inducţie că aplicarea repetată a relaţiilor (3.4.13) şi (3.4.14), dă o mulţime de direcţii mutual conjugate faţă de matricea pozitiv definită $\mathbf{H}$.

### 3.4.2 Metoda gradienţilor conjugaţi neliniară

Până acum, am tratat cazul unei funcţii de eroare polinomiale de gradul doi care are o matrice hessiană pozitiv definită $\mathbf{H}$. Dar în aplicaţii practice, funcţia de eroare poate fi departe de a fi polinomială de gradul doi, şi astfel expresiile pentru calculul lui $\alpha_k$ şi $\beta_k$ pe care le-am dedus mai sus, s-ar putea să nu fie la fel de precise ca în cazul polinomial. Mai mult, aceste expresii au nevoie de calculul explicit al matricii hessiane $\mathbf{H}$ pentru fiecare pas al algoritmului, pentru că hessiana este constantă doar în cazul unei funcţii de eroare polinomiale de gradul doi. Acest calcul este dificil de făcut, şi ar trebui să fie evitat. În cele ce urmează, vom deduce expresii pentru $\alpha_k$ şi $\beta_k$ care nu necesită calculul explicit al matricii hessiane, şi nici măcar nu presupun această matrice ca fiind pozitiv definită.

În primul rând, să luăm în considerare expresia pentru $\alpha_k$ , dată în (3.4.12). Datorită relaţiei iterative (3.4.10), putem să înlocuim calculul explicit al lui $\alpha_k$ cu o *căutare liniară inexactă* care minimizează $E(\overset{c}{\mathbf{w}}_{k+1}) = E(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k)$, i.e. o minimizare liniară de-a lungul direcţiei $\overset{c}{\mathbf{p}}_k$, începând de la punctul $\overset{c}{\mathbf{w}}_k$. În experimentele noastre, am folosit metoda secţiunii de aur, care s-a demonstrat că are convergenţă liniară, de exemplu în [167, 47].

Acum, să ne îndreptăm atenţia către $\beta_k$. Din (3.4.5), avem că

$$\overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k = \overset{c}{\mathbf{H}}_k(\overset{c}{\mathbf{w}}_{k+1} - \overset{c}{\mathbf{w}}_k) = \alpha_k \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k, \tag{3.4.15}$$

unde $\overset{c}{\mathbf{H}}_k := \mathbf{H}(\overset{c}{\mathbf{w}}_k)$, şi deci expresia (3.4.14) devine:

$$\beta_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k \rangle}{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k \rangle}. \tag{3.4.16}$$

Aceasta este cunoscută ca fiind expresia de actualizare *Hestenes-Stiefel*, a se vedea [112].

Pentru a prelucra mai departe această expresie, trebuie să luăm produsul scalar euclidian din $\mathbb{C}^{2N}$ al ecuaţiei (3.4.15) cu $\overset{c}{\mathbf{p}}_k$ şi să ţinem cont de expresia (3.4.12) pentru $\alpha_k$ obţinând astfel

$$\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_{k+1} \rangle = 0. \tag{3.4.17}$$

Luând produsul scalar dintre (3.4.13) şi $\overset{c}{\mathbf{g}}_{k+1}$, şi folosind ecuaţia de mai sus, rezultă că

$$\langle \overset{c}{\mathbf{p}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} \rangle = -\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} \rangle. \tag{3.4.18}$$

Acum, inserând relaţiile (3.4.17) şi (3.4.18) în expresia (3.4.16), obţinem expresia de actualizare *Polak-Ribiere* (a se vedea [214]):

$$\beta_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k \rangle}{\langle \overset{c}{\mathbf{g}}_k, \overset{c}{\mathbf{g}}_k \rangle}. \tag{3.4.19}$$

În continuare, luăm produsul scalar euclidian din $\mathbb{C}^{2N}$ al ecuaţiei (3.4.15) cu $\overset{c}{\mathbf{p}}_i$ , $i < k$, pentru a obţine

$$\langle \overset{c}{\mathbf{p}}_i, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k \rangle = \alpha_k \langle \overset{c}{\mathbf{p}}_i, \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k \rangle = 0, \tag{3.4.20}$$

unde am luat în considerare faptul că $\overset{c}{\mathbf{p}}_i$ şi $\overset{c}{\mathbf{p}}_k$ sunt direcţii conjugate, pentru $i < k$. Ecuaţiile (3.4.20) şi (3.4.17) pot fi folosite pentru a demonstra prin inducţie că avem

$$\langle \overset{c}{\mathbf{p}}_i, \overset{c}{\mathbf{g}}_k \rangle = 0, \forall i < k \leq N. \tag{3.4.21}$$

Din expresia pentru $\overset{c}{\mathbf{p}}_{k+1}$ dată în ecuaţia (3.4.13), deducem că $\overset{c}{\mathbf{p}}_k$ este dat de o combinaţie liniară a tuturor vectorilor gradient anteriori, şi anume putem scrie că

$$\overset{c}{\mathbf{p}}_i = -\overset{c}{\mathbf{g}}_i + \sum_{j=1}^{k-1} \gamma_j \overset{c}{\mathbf{g}}_j. \tag{3.4.22}$$

Luând produsul scalar al ecuaţiei (3.4.22) cu $\overset{c}{\mathbf{p}}_i$, $i < k$, şi folosind ecuaţia (3.4.21), obţinem următoarea relaţie:

$$\langle \overset{c}{\mathbf{g}}_i, \overset{c}{\mathbf{g}}_k \rangle = \sum_{j=1}^{k-1} \gamma_j \langle \overset{c}{\mathbf{g}}_j, \overset{c}{\mathbf{g}}_k \rangle, \forall i < k \leq N.$$

Pornind de la faptul că $\overset{c}{\mathbf{p}}_1 = -\overset{c}{\mathbf{g}}_1$, şi de la ecuaţia (3.4.21), obţinem prin inducţie că

$$\langle \overset{c}{\mathbf{g}}_i, \overset{c}{\mathbf{g}}_k \rangle = 0, \forall i < k \leq N. \tag{3.4.23}$$

În particular, avem că $\langle \overset{c}{\mathbf{g}}_k, \overset{c}{\mathbf{g}}_{k+1} \rangle = 0$, şi astfel expresia (3.4.19) devine:

$$\beta_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} \rangle}{\langle \overset{c}{\mathbf{g}}_k, \overset{c}{\mathbf{g}}_k \rangle}. \tag{3.4.24}$$

Această expresie este cunoscută ca formula de actualizare *Fletcher-Reeves,* a se vedea [225].

În fine, utilizând (3.4.23), şi anume faptul că $\langle \overset{c}{\mathbf{g}}_k, \overset{c}{\mathbf{g}}_{k+1} \rangle = 0$, în ecuaţia (3.4.16), obţinem următoarea formulă de actualizare:

$$\beta_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} \rangle}{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k \rangle},$$

care este cunoscută sub numele de expresia de actualizare *Dai-Yuan*, a se vedea [73].

Există două variaţii ale formulelor de actualizare Hestenes-Stiefel şi Polak-Ribiere, a căror convergenţă s-a demonstrat în [103, 91] chiar dacă este folosită o căutare liniară inexactă pentru a determina valoarea lui $\alpha_k$. Dacă notăm cu $\beta_k^{HS}$, şi respectiv

cu $\beta_k^{PR}$, valorile paşilor de actualizare ale celor doi algoritmi, expresiile pentru aceşti paşi care garantează convergenţa sunt

$$\beta_k^{HS+} = \max\{0, \beta_k^{HS}\},$$

şi respectiv

$$\beta_k^{PR+} = \max\{0, \beta_k^{PR}\}.$$

Dacă funcţia de eroare $E$ este pătratică (polinom de gradul doi), atunci metoda gradienţilor conjugaţi va găsi garantat minimul în cel mult $2N$ paşi. Dar, în general, funcţia de eroare poate fi departe de a fi pătratică, şi astfel algoritmul va avea nevoie de mai mult de $2N$ paşi pentru a se apropia de minim. De-a lungul acestor paşi, conjugarea direcţiilor de căutare tinde să se deterioreze, şi astfel este o practică comună aceea de a reporni algoritmul cu gradientul cu semn schimbat

$$\overset{c}{\mathbf{p}}_k = -\overset{c}{\mathbf{g}}_k,$$

după $2N$ paşi.

Un algoritm de repornire mai sofisticat, propus de Powell în [222], urmând o idee a lui Beale din [35], este de a reporni dacă există puţină ortogonalitate rămasă între gradientul curent şi gradientul anterior. Pentru a testa aceasta, verificăm faptul că are loc următoarea inegalitate:

$$|\langle \overset{c}{\mathbf{g}}_k, \overset{c}{\mathbf{g}}_{k+1}\rangle| \geq 0.2\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1}\rangle.$$

Regula de actualizare (3.4.13) pentru direcţia de căutare $\overset{c}{\mathbf{p}}_k$ este de asemenea schimbată în

$$\overset{c}{\mathbf{p}}_{k+1} = -\overset{c}{\mathbf{g}}_{k+1} + \beta_k \overset{c}{\mathbf{p}}_k + \gamma_k \overset{c}{\mathbf{p}}_t,$$

unde coeficienţii $\beta_k$ şi $\gamma_k$ sunt calculaţi în felul următor:

$$\beta_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k\rangle}{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k\rangle}$$

$$\gamma_k = \frac{\langle \overset{c}{\mathbf{g}}_{k+1}, \overset{c}{\mathbf{g}}_{t+1} - \overset{c}{\mathbf{g}}_t\rangle}{\langle \overset{c}{\mathbf{p}}_t, \overset{c}{\mathbf{g}}_{t+1} - \overset{c}{\mathbf{g}}_t\rangle},$$

iar $\overset{c}{\mathbf{p}}_t$ este o direcţie descendentă de repornire arbitrar aleasă. Aceste expresii sunt deduse în acelaşi fel cu cele de mai sus. Metoda gradienţilor conjugaţi cu aceste caracteristici este numită metoda *Powell-Beale*.

## 3.5 Metoda gradienţilor conjugaţi scalaţi

După cum am văzut mai sus, în aplicaţiile din lumea reală, matricea hessiană poate fi departe de a fi pozitiv definită. Datorită acestui fapt, Møller a propus în [178], metoda gradienţilor conjugaţi scalaţi care foloseşte metoda *model trust region* cunoscută de la algoritmul Levenberg-Marquardt, combinată cu metoda gradienţilor conjugaţi. În cele ce urmează, vom prezenta această metodă pentru reţele neuronale cu valori complexe, ca în articolul nostru [221].

Pentru a asigura pozitiv definirea, trebuie să adăugăm la matricea hessiană o constantă pozitivă suficient de mare $\lambda_k$, înmulţită cu matricea identitate. Cu această modificare, formula pentru pasul de actualizare dată în (3.4.12), devine

$$\alpha_k = -\frac{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_k \rangle}{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k \rangle + \lambda_k \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{p}}_k \rangle}. \tag{3.5.1}$$

Notăm numitorul din (3.5.1) cu $\delta_k := \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k \rangle + \lambda_k \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{p}}_k \rangle$. Pentru o hessiană pozitiv definită, avem că $\delta_k > 0$. Dar dacă $\delta_k \leq 0$, atunci va trebui să creştem valoarea lui $\delta_k$ pentru a o face pozitivă. Fie $\overline{\delta_k}$ noua valoare a lui $\delta_k$, şi, corespunzător, fie $\overline{\lambda_k}$ noua valoare a lui $\lambda_k$. Este clar că avem următoarea relaţie

$$\overline{\delta_k} = \delta_k + (\overline{\lambda_k} - \lambda_k) \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{p}}_k \rangle, \tag{3.5.2}$$

şi, pentru ca $\overline{\delta_k} > 0$, trebuie să avem $\overline{\lambda_k} > \lambda_k - \delta_k / \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{p}}_k \rangle$. Møller în [178] alege pentru $\overline{\lambda_k}$ valoarea

$$\overline{\lambda_k} = 2 \left( \lambda_k - \frac{\delta_k}{\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{p}}_k \rangle} \right),$$

şi astfel expresia pentru noua valoare a lui $\delta_k$ dată în (3.5.2), devine

$$\overline{\delta_k} = -\delta_k + \lambda_k \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{p}}_k \rangle = -\langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k \rangle,$$

care este acum pozitivă şi va fi folosită în (3.5.1) pentru a calcula valoarea lui $\alpha_k$.

O altă problemă semnalată mai sus este aproximarea pătratică pentru funcţia de eroare $E$. Algoritmul gradienţilor conjugaţi scalaţi rezolvă această problemă prin considerarea unui parametru de comparaţie definit prin

$$\Delta_k = \frac{E(\overset{c}{\mathbf{w}}_k) - E(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k)}{E(\overset{c}{\mathbf{w}}_k) - E_Q(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k)}, \tag{3.5.3}$$

unde $E_Q(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k)$ reprezintă aproximarea pătratică locală a funcţiei de eroare într-o vecinătate a punctului $\overset{c}{\mathbf{w}}_k$, dată prin

$$E_Q(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k) = E(\overset{c}{\mathbf{w}}_k) + \alpha_k \overset{c}{\mathbf{g}}_k^H \overset{c}{\mathbf{p}}_k + \frac{1}{2} \alpha_k^2 \overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k,$$

sau

$$E_Q(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k) = E(\overset{c}{\mathbf{w}}_k) + \alpha_k \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_k \rangle + \frac{1}{2} \alpha_k^2 \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{p}}_k \rangle. \tag{3.5.4}$$

Putem vedea cu uşurinţă că $\Delta_k$ măsoară cât de bună este această aproximare pătratică. Introducând relaţia (3.5.4) în relaţia (3.5.3), şi ţinând cont de expresia (3.4.12) pentru $\alpha_k$, avem că

$$\Delta_k = \frac{2(E(\overset{c}{\mathbf{w}}_k) - E(\overset{c}{\mathbf{w}}_k + \alpha_k \overset{c}{\mathbf{p}}_k))}{\alpha_k \langle \overset{c}{\mathbf{p}}_k, \overset{c}{\mathbf{g}}_k \rangle}.$$

Valoarea lui $\lambda_k$ este apoi actualizată în felul următor:

$$\lambda_{k+1} = \begin{cases} \lambda_k/2, & \text{dacă } \Delta_k > 0.75 \\ 4\lambda_k, & \text{dacă } \Delta_k < 0.25 \\ \lambda_k, & \text{altfel} \end{cases}, \tag{3.5.5}$$

pentru a asigura o mai bună aproximare pătratică.

Prin urmare, există două stadii ale actualizării lui $\lambda_k$: una pentru a ne asigura că $\delta_k > 0$, şi una corespunzătoare validităţii aproximării pătratice. Cele două stadii sunt aplicate succesiv după fiecare actualizare a ponderilor.

## 3.6 Metoda Newton

Să presupunem că avem o reţea neuronală cu valori complexe, a cărei funcţie de eroare este $E : \mathbb{C}^{2N} \to \mathbb{R}$. Dacă notăm $\mathbf{g}(\overset{c}{\mathbf{w}}) := \nabla E(\overset{c}{\mathbf{w}})$, gradientul funcţiei $E$, şi cu $\mathbf{H}(\overset{c}{\mathbf{w}}) := \nabla^2 E(\overset{c}{\mathbf{w}})$, hessiana funcţiei $E$, atunci presupunând că $\overset{c}{\mathbf{w}} \in \mathbb{C}^{2N}$ este vectorul $2N$-dimensional al tuturor ponderilor şi bias-urilor reţelei şi ale conjugatelor lor, putem găsi minimul funcţiei de eroare folosind *metoda lui Newton* care presupune calculul iterativ al vectorului $\overset{c}{\mathbf{w}}^*$ care minimizează funcţia $E$ folosind următoarea regulă de actualizare

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k - \mathbf{H}^{-1}(\overset{c}{\mathbf{w}}_k)\mathbf{g}(\overset{c}{\mathbf{w}}_k),$$

sau, echivalent, dacă notăm $\overset{c}{\mathbf{H}}_k := \mathbf{H}(\overset{c}{\mathbf{w}}_k)$ şi respectiv $\overset{c}{\mathbf{g}}_k := \mathbf{g}(\overset{c}{\mathbf{w}}_k)$,

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k - \overset{c}{\mathbf{H}}_k^{-1} \overset{c}{\mathbf{g}}_k.$$

Această regulă de actualizare face o presupunere esenţială asupra matricii hessiane, şi anume aceea că ea trebuie să fie pozitiv definită. Dacă această condiţie nu este îndeplinită, atunci se foloseşte formula de actualizare modificată

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k - (\overset{c}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2N})^{-1} \overset{c}{\mathbf{g}}_k,$$

unde $\lambda_k$ este o constantă care se alege astfel încât matricea $\overset{c}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2N}$ să fie pozitiv definită.

Deoarece am dat în cadrul metodei gradient un algoritm de calculare a gradientului funcţiei de eroare a unei reţele neuronale cu valori complexe, pentru a putea implementa metoda Newton, mai trebuie să găsim o modalitate de calcul a hessianei funcţiei de eroare. Din acest motiv, restul acestei secţiuni este dedicat deducerii complete a unei metode de calcul a hessianei funcţiei de eroare a unei reţele neuronale cu valori complexe, şi respectiv a produsului hessianei cu un vector. Pentru această deducere, s-a folosit calculul Wirtinger, descris în prima secţiune a acestui capitol. Prima subsecţiune dă procedura generală de calcul a hessianei, a doua prezintă o aplicaţie a acestei proceduri generale pe o reţea cu valori complexe cu un singur strat ascuns, iar ultima este dedicată calculului produsului dintre matricea hessiană şi un vector oarecare.

Chiar dacă este complicată din punct de vedere computaţional, metoda Newton este cea mai exactă dintre metodele de optimizare de ordinul doi folosite pentru găsirea minimului funcţiei de eroare, motiv pentru care am decis s-o includem în studiul nostru despre algoritmii de învăţare pentru reţelele neuronale cu valori complexe. În plus, hessiana are şi alte aplicaţii în domeniul reţelelor neuronale, cum ar fi spre exemplu diferiţi algoritmi de antrenare a reţelelor, în reantrenarea rapidă a reţelei după o mică modificare în datele de antrenare [44], pentru identificarea celor mai puţin semnificative ponderi ca o bază pentru tehnici de simplificare a reţelelor [158], pentru îmbunătăţirea vitezei algoritmilor de antrenare [36], pentru estimarea bayesiană a parametrilor de regularizare [168], pentru asignarea de probabilităţi diferitelor soluţii ale reţelei, şi multe altele. Prezentarea noastră urmăreşte în principal primul articol în care a fost calculată exact hessiana pentru reţelele neuronale cu valori reale, şi anume [45], care a fost apoi reluat în cartea clasică [46].

### 3.6.1 Calculul hessianei

Vom prezenta algoritmul pentru calculul matricii hessiane ca în articolul nostru [216]. Elementele hessianei sunt $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatele parţiale de ordinul doi ale funcţiei de

eroare $E(\overset{\mathcal{C}}{\mathbf{w}})$ faţă de oricare două ponderi sau bias-uri ale reţelei. Să notăm cu $w_{mn}^{l_1}$ şi $w_{jk}^{l_2}$ doi astfel de parametri ai reţelei neuronale cu valori complexe, şi să presupunem de asemenea că $2 \leq l_1 \leq l_2 \leq L$. Pentru aceşti doi parametri, avem patru $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivate parţiale de ordinul doi:

$$\frac{\partial^2 E}{\partial w_{mn}^{l_1} \partial w_{jk}^{l_2}}, \quad \frac{\partial^2 E}{\partial \overline{w_{mn}^{l_1}} \partial w_{jk}^{l_2}}, \quad \frac{\partial^2 E}{\partial w_{mn}^{l_1} \partial \overline{w_{jk}^{l_2}}}, \quad \frac{\partial^2 E}{\partial \overline{w_{mn}^{l_1}} \partial \overline{w_{jk}^{l_2}}}.$$

Deoarece calculele sunt asemănătoare, vom da o metodă pentru a calcula $\overline{\mathbb{R}}$-derivata parţială de ordinul doi a funcţiei $E$ în raport cu $\overline{w_{mn}^{l_1}}$ şi $\overline{w_{jk}^{l_2}}$.

Începem din nou cu regula înlănţuirii

$$\begin{aligned} \frac{\partial^2 E}{\partial \overline{w_{mn}^{l_1}} \partial \overline{w_{jk}^{l_2}}} &= \frac{\partial s_m^{l_1}}{\partial \overline{w_{mn}^{l_1}}} \frac{\partial^2 E}{\partial s_m^{l_1} \partial \overline{w_{jk}^{l_2}}} + \frac{\partial \overline{s_m^{l_1}}}{\partial \overline{w_{mn}^{l_1}}} \frac{\partial^2 E}{\partial \overline{s_m^{l_1}} \partial \overline{w_{jk}^{l_2}}} \\ &= \overline{x_n^{l_1-1}} \frac{\partial^2 E}{\partial \overline{s_m^{l_1}} \partial \overline{w_{jk}^{l_2}}}, \end{aligned}$$

unde avem că $\frac{\partial s_m^{l_1}}{\partial \overline{w_{mn}^{l_1}}} = 0$. Acum, folosind (3.2.4), putem scrie că

$$\begin{aligned} \frac{\partial^2 E}{\partial \overline{s_m^{l_1}} \partial \overline{w_{jk}^{l_2}}} &= \frac{\partial (\delta_j^{l_2} \overline{x_k^{l_2-1}})}{\partial \overline{s_m^{l_1}}} \\ &= \frac{\partial \delta_j^{l_2}}{\partial \overline{s_m^{l_1}}} \overline{x_k^{l_2-1}} + \frac{\partial \overline{x_k^{l_2-1}}}{\partial \overline{s_m^{l_1}}} \delta_j^{l_2}, \end{aligned} \tag{3.6.1}$$

şi, din regula înlănţuirii, că

$$\begin{aligned} \frac{\partial \overline{x_k^{l_2-1}}}{\partial \overline{s_m^{l_1}}} &= \frac{\partial s_k^{l_2-1}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{x_k^{l_2-1}}}{\partial s_k^{l_2-1}} + \frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{x_k^{l_2-1}}}{\partial \overline{s_k^{l_2-1}}} \\ &= \frac{\partial s_k^{l_2-1}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial s_k^{l_2-1}} + \frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial \overline{s_k^{l_2-1}}} \\ &= h_{km}^{l_2-1,l_1} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial s_k^{l_2-1}} + g_{km}^{l_2-1,l_1} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial \overline{s_k^{l_2-1}}}, \end{aligned}$$

unde am notat $h_{km}^{l_2-1,l_1} := \frac{\partial s_k^{l_2-1}}{\partial \overline{s_m^{l_1}}}$ şi $g_{km}^{l_2-1,l_1} := \frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{s_m^{l_1}}}$. Dacă notăm în continuare $b_{jm}^{l_2,l_1} := \frac{\partial \delta_j^{l_2}}{\partial \overline{s_m^{l_1}}}$, ecuaţia (3.6.1) devine

$$\frac{\partial^2 E}{\partial \overline{s_m^{l_1}} \partial \overline{w_{jk}^{l_2}}} = b_{jm}^{l_2,l_1} \overline{x_k^{l_2-1}} + h_{km}^{l_2-1,l_1} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial s_k^{l_2-1}} \delta_j^{l_2} + g_{km}^{l_2-1,l_1} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial \overline{s_k^{l_2-1}}} \delta_j^{l_2}.$$

Acum trebuie să găsim o metodă de a calcula $h_{km}^{l_2-1,l_1}$, $g_{km}^{l_2-1,l_1}$, şi $b_{jm}^{l_2,l_1}$. Începem cu primele două. Dacă $l_2 = l_1$, avem că $h_{km}^{l_2-1,l_1} = g_{km}^{l_2-1,l_1} = 0$, deoarece $s_k^{l_2-1}$ şi $\overline{s_k^{l_2-1}}$ nu depind de $\overline{s_m^{l_1}}$. Dacă $l_2 - 1 = l_1$, $h_{km}^{l_2-1,l_1} = 0$, $g_{km}^{l_2-1,l_1} = \delta_{km}$, deoarece $s_k^{l_2-1}$, $\overline{s_k^{l_2-1}}$, şi $\overline{s_m^{l_1}}$ sunt acum pe acelaşi strat, şi $\delta_{km}$ este simbolul delta al lui Kronecker: $\delta_{km} = 1$ dacă $k = m$ şi $\delta_{km} = 0$ dacă $k \neq m$. Pentru $l_2 \geq l_1 + 2$, (în care caz trebuie să avem $L \geq 4$) putem să aplicăm din nou regula înlănţuirii

$$\begin{aligned}\frac{\partial s_k^{l_2-1}}{\partial \overline{s_m^{l_1}}} &= \sum_r \left( \frac{\partial s_r^{l_2-2}}{\partial \overline{s_m^{l_1}}} \frac{\partial s_k^{l_2-1}}{\partial s_r^{l_2-2}} + \frac{\partial \overline{s_r^{l_2-2}}}{\partial \overline{s_m^{l_1}}} \frac{\partial s_k^{l_2-1}}{\partial \overline{s_r^{l_2-2}}} \right) \\ &= \sum_r \left( h_{rm}^{l_2-2,l_1} w_{kr}^{l_2-1} \frac{\partial G^{l_2-2}(s_r^{l_2-2})}{\partial s_r^{l_2-2}} + g_{rm}^{l_2-2,l_1} w_{kr}^{l_2-1} \frac{\partial G^{l_2-2}(s_r^{l_2-2})}{\partial \overline{s_r^{l_2-2}}} \right),\end{aligned}$$

unde am folosit faptul că

$$\begin{aligned}\frac{\partial s_k^{l_2-1}}{\partial s_r^{l_2-2}} &= \frac{\partial s_k^{l_2-1}}{\partial y_r^{l_2-2}} \frac{\partial y_r^{l_2-2}}{\partial s_r^{l_2-2}} + \frac{\partial s_k^{l_2-1}}{\partial \overline{y_r^{l_2-2}}} \frac{\partial \overline{y_r^{l_2-2}}}{\partial s_r^{l_2-2}} \\ &= w_{kr}^{l_2-1} \frac{\partial G^{l_2-2}(s_r^{l_2-2})}{\partial s_r^{l_2-2}},\end{aligned}$$

şi

$$\begin{aligned}\frac{\partial s_k^{l_2-1}}{\partial \overline{s_r^{l_2-2}}} &= \frac{\partial s_k^{l_2-1}}{\partial y_r^{l_2-2}} \frac{\partial y_r^{l_2-2}}{\partial \overline{s_r^{l_2-2}}} + \frac{\partial s_k^{l_2-1}}{\partial \overline{y_r^{l_2-2}}} \frac{\partial \overline{y_r^{l_2-2}}}{\partial \overline{s_r^{l_2-2}}} \\ &= w_{kr}^{l_2-1} \frac{\partial G^{l_2-2}(s_r^{l_2-2})}{\partial \overline{s_r^{l_2-2}}},\end{aligned}$$

iar suma se ia după toţi neuronii $r$ din stratul $l_2 - 2$ care trimit conexiuni către neuronul $k$ din stratul $l_2 - 1$.

Similar, avem că

$$\begin{aligned}\frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{s_m^{l_1}}} &= \sum_r \left( \frac{\partial s_r^{l_2-2}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{s_k^{l_2-1}}}{\partial s_r^{l_2-2}} + \frac{\partial \overline{s_r^{l_2-2}}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{s_r^{l_2-2}}} \right) \\ &= \sum_r \left( h_{rm}^{l_2-2,l_1} \overline{w_{kr}^{l_2-1}} \frac{\partial \overline{G^{l_2-2}(s_r^{l_2-2})}}{\partial s_r^{l_2-2}} + g_{rm}^{l_2-2,l_1} \overline{w_{kr}^{l_2-1}} \frac{\partial \overline{G^{l_2-2}(s_r^{l_2-2})}}{\partial \overline{s_r^{l_2-2}}} \right),\end{aligned}$$

unde, ca mai sus, am folosit

$$\begin{aligned}\frac{\partial \overline{s_k^{l_2-1}}}{\partial s_r^{l_2-2}} &= \frac{\partial \overline{s_k^{l_2-1}}}{\partial y_r^{l_2-2}} \frac{\partial y_r^{l_2-2}}{\partial s_r^{l_2-2}} + \frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{y_r^{l_2-2}}} \frac{\partial \overline{y_r^{l_2-2}}}{\partial s_r^{l_2-2}} \\ &= \overline{w_{kr}^{l_2-1}} \frac{\partial \overline{G^{l_2-1}(s_r^{l_2-2})}}{\partial s_r^{l_2-2}},\end{aligned}$$

şi

$$\begin{aligned}\frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{s_r^{l_2-2}}} &= \frac{\partial \overline{s_k^{l_2-1}}}{\partial y_r^{l_2-2}} \frac{\partial y_r^{l_2-2}}{\partial \overline{s_r^{l_2-2}}} + \frac{\partial \overline{s_k^{l_2-1}}}{\partial \overline{y_r^{l_2-2}}} \frac{\partial \overline{y_r^{l_2-2}}}{\partial \overline{s_r^{l_2-2}}} \\ &= \overline{w_{kr}^{l_2-1}} \frac{\partial \overline{G^{l_2-2}(s_r^{l_2-2})}}{\partial \overline{s_r^{l_2-2}}}.\end{aligned}$$

În concluzie, am obţinut următoarele relaţii de recurenţă pentru calculul lui $h_{km}^{l_2-1,l_1}$ şi $g_{km}^{l_2-1,l_1}$:

$$h_{km}^{l_2-1,l_1} = \sum_r \left( h_{rm}^{l_2-2,l_1} w_{kr}^{l_2-1} \frac{\partial G^{l_2-2}(s_r^{l_2-2})}{\partial s_r^{l_2-2}} + g_{rm}^{l_2-2,l_1} w_{kr}^{l_2-1} \frac{\partial G^{l_2-2}(s_r^{l_2-2})}{\partial \overline{s_r^{l_2-2}}} \right), \quad (3.6.2)$$

$$g_{km}^{l_2-1,l_1} = \sum_r \left( h_{rm}^{l_2-2,l_1} \overline{w_{kr}^{l_2-1}} \frac{\partial \overline{G^{l_2-2}(s_r^{l_2-2})}}{\partial s_r^{l_2-2}} + g_{rm}^{l_2-2,l_1} \overline{w_{kr}^{l_2-1}} \frac{\partial \overline{G^{l_2-2}(s_r^{l_2-2})}}{\partial \overline{s_r^{l_2-2}}} \right). \quad (3.6.3)$$

Acum, să ne îndreptăm atenţia către $b_{jm}^{l_2,l_1}$. Tratăm mai întâi cazul în care $l_2 \leq L-1$. Din formula pentru $\delta_j^l$ dată în ecuaţia (3.2.5), avem că

$$\begin{aligned}\frac{\partial \delta_j^{l_2}}{\partial \overline{s_m^{l_1}}} &= \sum_r \left( \frac{\partial \delta_r^{l_2+1}}{\partial \overline{s_m^{l_1}}} \overline{w_{rj}^{l_2+1}} \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_j^{l_2}}} + \delta_r^{l_2+1} \frac{\partial \overline{w_{rj}^{l_2+1}}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_j^{l_2}}} \right. \\ &+ \delta_r^{l_2+1} \overline{w_{rj}^{l_2+1}} \frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}} \partial \overline{s_j^{l_2}}} + \frac{\partial \overline{\delta_r^{l_2+1}}}{\partial \overline{s_m^{l_1}}} w_{rj}^{l_2+1} \frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}}} \\ &+ \left. \overline{\delta_r^{l_2+1}} \frac{\partial w_{rj}^{l_2+1}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_j^{l_2}}} + \overline{\delta_r^{l_2+1}} w_{rj}^{l_2+1} \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}} \partial \overline{s_j^{l_2}}} \right) \\ &= \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_j^{l_2}}} \sum_r b_{rm}^{l_2+1,l_1} \overline{w_{rj}^{l_2+1}} + \frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}} \partial \overline{s_j^{l_2}}} \sum_r \delta_r^{l_2+1} \overline{w_{rj}^{l_2+1}} \\ &+ \frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}}} \sum_r a_{rm}^{l_2+1,l_1} w_{rj}^{l_2+1} + \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}} \partial \overline{s_j^{l_2}}} \sum_r \overline{\delta_r^{l_2+1}} w_{rj}^{l_2+1},\end{aligned}$$

unde am notat $a_{jm}^{l_2,l_1} := \frac{\partial \overline{\delta_j^{l_2}}}{\partial \overline{s_m^{l_1}}}$, iar suma se ia după toţi neuronii $r$ din stratul $l_2+1$ către care neuronul $j$ din stratul $l_2$ trimite conexiuni, şi $\frac{\partial \overline{w_{rj}^{l_2+1}}}{\partial \overline{s_m^{l_1}}} = \frac{\partial w_{rj}^{l_2+1}}{\partial \overline{s_m^{l_1}}} = 0$. Aceasta înseamnă că trebuie să găsim o formulă pentru a calcula $a_{jm}^{l_2,l_1}$, de asemenea. Ca în cazul lui $b_{jm}^{l_2,l_1}$, putem scrie că

$$\begin{aligned}
\frac{\partial \overline{\delta_j^{l_2}}}{\partial \overline{s_m^{l_1}}} &= \sum_r \left( \frac{\partial \overline{\delta_r^{l_2+1}}}{\partial \overline{s_m^{l_1}}} w_{rj}^{l_2+1} \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} + \overline{\delta_r^{l_2+1}} \frac{\partial w_{rj}^{l_2+1}}{\partial \overline{s_m^{l_1}}} \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} \right. \\
&+ \overline{\delta_r^{l_2+1}} w_{rj}^{l_2+1} \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}} \partial s_j^{l_2}} + \frac{\partial \delta_r^{l_2+1}}{\partial \overline{s_m^{l_1}}} \overline{w_{rj}^{l_2+1}} \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial s_j^{l_2}} \\
&+ \left. \delta_r^{l_2+1} \frac{\partial \overline{w_{rj}^{l_2+1}}}{\partial \overline{s_m^{l_1}}} \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial s_j^{l_2}} + \delta_r^{l_2+1} \overline{w_{rj}^{l_2+1}} \frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}} \partial s_j^{l_2}} \right) \\
&= \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} \sum_r a_{rm}^{l_2+1,l_1} w_{rj}^{l_2+1} + \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}} \partial s_j^{l_2}} \sum_r \overline{\delta_r^{l_2+1}} w_{rj}^{l_2+1} \\
&+ \frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial s_j^{l_2}} \sum_r b_{rm}^{l_2+1,l_1} w_{rj}^{l_2+1} + \frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}} \partial s_j^{l_2}} \sum_r \delta_r^{l_2+1} \overline{w_{rj}^{l_2+1}}. \qquad (3.6.4)
\end{aligned}$$

Expresiile pentru $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatele parţiale de ordinul doi ale funcţiilor de activare rezultă uşor din aplicarea regulii înlănţuirii:

$$\begin{aligned}
\frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}} \partial \overline{s_j^{l_2}}} &= \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2} \partial \overline{s_j^{l_2}}} \frac{\partial s_j^{l_2}}{\partial \overline{s_m^{l_1}}} + \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}} \partial \overline{s_j^{l_2}}} \frac{\partial \overline{s_j^{l_2}}}{\partial \overline{s_m^{l_1}}} \\
&= \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2} \partial \overline{s_j^{l_2}}} h_{jm}^{l_2,l_1} + \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}} \partial \overline{s_j^{l_2}}} g_{jm}^{l_2,l_1},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}} \partial s_j^{l_2}} &= \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2} \partial s_j^{l_2}} \frac{\partial s_j^{l_2}}{\partial \overline{s_m^{l_1}}} + \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}} \partial s_j^{l_2}} \frac{\partial \overline{s_j^{l_2}}}{\partial \overline{s_m^{l_1}}} \\
&= \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2} \partial s_j^{l_2}} h_{jm}^{l_2,l_1} + \frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}} \partial s_j^{l_2}} g_{jm}^{l_2,l_1},
\end{aligned}$$

şi cele analoage pentru $\frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}} \partial \overline{s_j^{l_2}}}$ şi $\frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}} \partial s_j^{l_2}}$.

În continuare, avem de tratat cazul în care $l_2 = L$. Atunci, din expresia (3.2.5), rezultă că

$$\begin{aligned}
\frac{\partial \delta_j^{l_2}}{\partial \overline{s_m^{l_1}}} &= \frac{1}{2}\frac{\partial (y_j^{l_2}-t_j)}{\partial \overline{s_m^{l_1}}}\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_j^{l_2}}} + \frac{1}{2}(y_j^{l_2}-t_j)\frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}\partial \overline{s_j^{l_2}}} \\
&+ \frac{1}{2}\frac{\partial \overline{(y_j^{l_2}-t_j)}}{\partial \overline{s_m^{l_1}}}\frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}}} + \frac{1}{2}\overline{(y_j^{l_2}-t_j)}\frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}\partial \overline{s_j^{l_2}}} \\
&= \frac{1}{2}\frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}}\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_j^{l_2}}} + \frac{1}{2}(y_j^{l_2}-t_j)\frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}\partial \overline{s_j^{l_2}}} \\
&+ \frac{1}{2}\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}}\frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}}} + \frac{1}{2}\overline{(y_j^{l_2}-t_j)}\frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}\partial \overline{s_j^{l_2}}},
\end{aligned} \tag{3.6.5}$$

şi

$$\begin{aligned}
\frac{\partial \overline{\delta_j^{l_2}}}{\partial \overline{s_m^{l_1}}} &= \frac{1}{2}\frac{\partial \overline{(y_j^{l_2}-t_j)}}{\partial \overline{s_m^{l_1}}}\frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} + \frac{1}{2}\overline{(y_j^{l_2}-t_j)}\frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}\partial s_j^{l_2}} \\
&+ \frac{1}{2}\frac{\partial (y_j^{l_2}-t_j)}{\partial \overline{s_m^{l_1}}}\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial s_j^{l_2}} + \frac{1}{2}(y_j^{l_2}-t_j)\frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}\partial s_j^{l_2}} \\
&= \frac{1}{2}\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}}\frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} + \frac{1}{2}\overline{(y_j^{l_2}-t_j)}\frac{\partial^2 G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}\partial s_j^{l_2}} \\
&+ \frac{1}{2}\frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}}\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial s_j^{l_2}} + \frac{1}{2}(y_j^{l_2}-t_j)\frac{\partial^2 \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}\partial s_j^{l_2}}.
\end{aligned}$$

Expresiile pentru calculul $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatelor parţiale de ordinul doi ale funcţiilor de activare sunt la fel ca cele de mai sus. Prin urmare, trebuie doar să mai dăm expresii pentru calculul $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatelor parţiale ale funcţiilor de activare folosite în expresiile de mai sus. Acestea sunt deduse prin aplicarea regulii înlănţuirii:

$$\begin{aligned}
\frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_m^{l_1}}} &= \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}}\frac{\partial s_j^{l_2}}{\partial \overline{s_m^{l_1}}} + \frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}}}\frac{\partial \overline{s_j^{l_2}}}{\partial \overline{s_m^{l_1}}} \\
&= \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} h_{jm}^{l_2,l_1} + \frac{\partial G^{l_2}(s_j^{l_2})}{\partial \overline{s_j^{l_2}}} g_{jm}^{l_2,l_1},
\end{aligned}$$

şi cele analoage pentru $\frac{\partial \overline{G^{l_2}(s_j^{l_2})}}{\partial \overline{s_m^{l_1}}}$.

Se poate observa că expresiile pentru $h_{km}^{l_2-1,l_1}$ şi $g_{km}^{l_2-1,l_1}$ sunt calculate prin propagare înainte, iar expresiile pentru $b_{jm}^{l_2,l_1}$ şi $a_{jm}^{l_2,l_1}$ sunt calculate prin propagare înapoi. În fine, punând totul împreună, obţinem următoarea formulă pentru o componentă arbitrară a matricii hessiane a funcţiei de eroare $E(\overset{c}{\mathbf{w}})$:

$$\begin{aligned}\frac{\partial^2 E}{\partial \overline{w_{mn}^{l_1}} \partial \overline{w_{jk}^{l_2}}} &= \overline{x_n^{l_1-1}} b_{jm}^{l_2,l_1} \overline{x_k^{l_2-1}} + \overline{x_n^{l_1-1}} h_{km}^{l_2-1,l_1} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial s_k^{l_2-1}} \delta_j^{l_2} \\ &+ \overline{x_n^{l_1-1}} g_{km}^{l_2-1,l_1} \frac{\partial \overline{G^{l_2-1}(s_k^{l_2-1})}}{\partial \overline{s_k^{l_2-1}}} \delta_j^{l_2}. \end{aligned} \tag{3.6.6}$$

### 3.6.2 Calculul hessianei pentru o reţea neuronală cu trei straturi

În această subsecţiune, vom considera cazul special al unei reţele cu un singur strat ascuns, pentru care $L = 3$. Vom nota prin $i, i'$ neuronii din stratul de intrare, prin $j, j'$ neuronii din stratul ascuns, şi prin $k, k'$ neuronii din stratul de ieşire. Cu aceste convenţii, ecuaţiile (3.2.1) şi (3.2.2) devin

$$s_j^2 := \sum_i w_{ji}^2 x_i^1 + w_{j0}^2,$$

$$x_j^2 := G^2(s_j^2),$$

$$s_k^3 := \sum_j w_{kj}^3 x_j^2 + w_{k0}^3,$$

$$x_k^3 := G^3(s_k^3).$$

Expresia (3.6.6) pentru $\overline{\mathbb{R}}$-derivata parţială de ordinul doi a funcţiei de eroare $E(\overset{c}{\mathbf{w}})$, în cazul $l_1 = l_2 = 3$, este

$$\frac{\partial^2 E}{\partial \overline{w_{k'j'}^3} \partial \overline{w_{kj}^3}} = \overline{x_{j'}^2} \frac{\partial \delta_k^3}{\partial \overline{s_{k'}^3}} \overline{x_j^2},$$

deoarece $h_{jk'}^{2,3} = g_{jk'}^{2,3} = 0$. Acum, mai trebuie doar să calculăm $\frac{\partial \delta_k^3}{\partial \overline{s_{k'}^3}}$ folosind ecuaţia (3.6.5):

$$\begin{aligned}\frac{\partial \delta_k^3}{\partial \overline{s_{k'}^3}} &= \frac{1}{2} \frac{\partial G^3(s_k^3)}{\partial \overline{s_{k'}^3}} \frac{\partial \overline{G^3(s_k^3)}}{\partial \overline{s_k^3}} + \frac{1}{2}(y_k^3 - t_k) \frac{\partial^2 \overline{G^3(s_k^3)}}{\partial \overline{s_{k'}^3} \partial \overline{s_k^3}} \\ &+ \frac{1}{2} \frac{\partial \overline{G^3(s_k^3)}}{\partial \overline{s_{k'}^3}} \frac{\partial G^3(s_k^3)}{\partial \overline{s_k^3}} + \frac{1}{2}(\overline{y_k^3 - t_k}) \frac{\partial^2 G^3(s_k^3)}{\partial \overline{s_{k'}^3} \partial \overline{s_k^3}} \\ &= \delta_{k'k} \left[ \frac{\partial G^3(s_k^3)}{\partial \overline{s_k^3}} \frac{\partial \overline{G^3(s_k^3)}}{\partial \overline{s_k^3}} + \frac{1}{2}(y_k^3 - t_k) \frac{\partial^2 \overline{G^3(s_k^3)}}{\partial \overline{s_k^3}^2} + \frac{1}{2}(\overline{y_k^3 - t_k}) \frac{\partial^2 G^3(s_k^3)}{\partial \overline{s_k^3}^2} \right], \end{aligned}$$

unde am folosit expresia (3.2.5) pentru $\delta_k^3$, şi $\delta_{k'k}$ este simbolul delta al lui Kronecker definit mai sus. Expresia finală pentru $\overline{\mathbb{R}}$-derivata parţială de ordinul doi este

$$\begin{aligned}\frac{\partial^2 E}{\partial\overline{w^3_{k'j'}}\partial\overline{w^3_{kj}}} &= \overline{x^2_{j'}x^2_j}\delta_{k'k}\left[\frac{\partial G^3(s^3_k)}{\partial\overline{s^3_k}}\frac{\partial\overline{G^3(s^3_k)}}{\partial\overline{s^3_k}}\right.\\ &+ \left.\frac{1}{2}(y^3_k - t_k)\frac{\partial^2\overline{G^3(s^3_k)}}{\partial^2\overline{s^3_k}} + \frac{1}{2}(\overline{y^3_k - t_k})\frac{\partial^2\overline{G^3(s^3_k)}}{\partial^2\overline{s^3_k}}\right].\end{aligned}$$

Acum, considerăm cazul în care $l_1 = l_2 = 2$. Avem de asemenea $h^{1,2}_{ij'} = g^{1,2}_{ij'} = 0$, deci expresia (3.6.6) devine

$$\frac{\partial^2 E}{\partial\overline{w^2_{j'i'}}\partial\overline{w^2_{ji}}} = \overline{x^1_{i'}}\frac{\partial\delta^2_j}{\partial\overline{s^2_{j'}}}\overline{x^1_i}.$$

Trebuie să calculăm $\frac{\partial\delta^2_j}{\partial\overline{s^2_{j'}}}$, şi pentru aceasta vom folosi formula (3.6.4):

$$\begin{aligned}\frac{\partial\delta^2_j}{\partial\overline{s^2_{j'}}} &= \frac{\partial\overline{G^2(s^2_j)}}{\partial\overline{s^2_j}}\sum_r b^{3,2}_{rj'}\overline{w^3_{rj}} + \frac{\partial^2\overline{G^2(s^2_j)}}{\partial\overline{s^2_{j'}}\partial\overline{s^2_j}}\sum_r \delta^3_r\overline{w^3_{rj}}\\ &+ \frac{\partial G^2(s^2_j)}{\partial\overline{s^2_j}}\sum_r a^{3,2}_{rj'}w^3_{rj} + \frac{\partial^2 G^2(s^2_j)}{\partial\overline{s^2_{j'}}\partial\overline{s^2_j}}\sum_r\overline{\delta^3_r}w^3_{rj}.\end{aligned}$$

Relaţia de mai sus arată că avem de asemenea nevoie de expresii pentru $a^{3,2}_{rj'}$ şi $b^{3,2}_{rj'}$. Folosind din nou ecuaţia (3.6.5), avem că

$$\begin{aligned}\frac{\partial\delta^3_r}{\partial\overline{s^2_{j'}}} &= \frac{1}{2}\frac{\partial G^3(s^3_r)}{\partial\overline{s^2_{j'}}}\frac{\partial\overline{G^3(s^3_r)}}{\partial\overline{s^3_r}} + \frac{1}{2}(y^3_r - t_r)\frac{\partial^2\overline{G^3(s^3_r)}}{\partial\overline{s^2_{j'}}\partial\overline{s^3_r}}\\ &+ \frac{1}{2}\frac{\partial\overline{G^3(s^3_r)}}{\partial\overline{s^2_{j'}}}\frac{\partial G^3(s^3_r)}{\partial\overline{s^3_r}} + \frac{1}{2}(\overline{y^3_r - t_r})\frac{\partial^2 G^3(s^3_r)}{\partial\overline{s^2_{j'}}\partial\overline{s^3_r}},\end{aligned}$$

şi

$$\begin{aligned}\frac{\partial\overline{\delta^3_r}}{\partial\overline{s^2_{j'}}} &= \frac{1}{2}\frac{\partial\overline{G^3(s^3_r)}}{\partial\overline{s^2_{j'}}}\frac{\partial G^3(s^3_r)}{\partial s^3_r} + \frac{1}{2}(\overline{y^3_r - t_r})\frac{\partial^2 G^3(s^3_r)}{\partial\overline{s^2_{j'}}\partial s^3_r}\\ &+ \frac{1}{2}\frac{\partial G^3(s^3_r)}{\partial\overline{s^2_{j'}}}\frac{\partial\overline{G^3(s^3_r)}}{\partial s^3_r} + \frac{1}{2}(y^3_r - t_r)\frac{\partial^2\overline{G^3(s^3_r)}}{\partial\overline{s^2_{j'}}\partial s^3_r},\end{aligned}$$

unde avem următoarele expresii:

$$\frac{\partial G^3(s^3_r)}{\partial\overline{s^2_{j'}}} = \frac{\partial G^3(s^3_r)}{\partial s^3_r}h^{3,2}_{rj'} + \frac{\partial G^3(s^3_r)}{\partial\overline{s^3_r}}g^{3,2}_{rj'},$$

şi analog pentru $\frac{\partial\overline{G^3(s^3_r)}}{\partial\overline{s^2_{j'}}}$, şi

$$\frac{\partial^2 G^3(s_r^3)}{\partial \overline{s_r^3}\partial \overline{s_{j'}^2}} = \frac{\partial^2 G^3(s_r^3)}{\partial s_r^3 \partial \overline{s_r^3}} h_{rj'}^{3,2} + \frac{\partial^2 G^3(s_r^3)}{\partial \overline{s_r^3}\partial \overline{s_r^3}} g_{rj'}^{3,2},$$

$$\frac{\partial^2 G^3(s_r^3)}{\partial s_r^3 \partial \overline{s_{j'}^2}} = \frac{\partial^2 G^3(s_r^3)}{\partial s_r^3 \partial s_r^3} h_{rj'}^{3,2} + \frac{\partial^2 G^3(s_r^3)}{\partial \overline{s_r^3}\partial s_r^3} g_{rj'}^{3,2},$$

şi cele analoage pentru $\frac{\partial^2 \overline{G^3(s_r^3)}}{\partial \overline{s_r^3}\partial \overline{s_{j'}^2}}$ şi $\frac{\partial^2 \overline{G^3(s_r^3)}}{\partial s_r^3 \partial \overline{s_{j'}^2}}$.

În fine, expresiile pentru $h_{rj'}^{3,2}$ şi $g_{rj'}^{3,2}$ pot fi deduse folosind ecuaţiile (3.6.2) şi (3.6.3):

$$\begin{aligned} h_{rj'}^{3,2} &= \sum_j \left( h_{jj'}^{2,2} w_{rj}^2 \frac{\partial G^2(s_j^2)}{\partial s_j^2} + g_{jj'}^{2,2} w_{rj}^2 \frac{\partial G^2(s_j^2)}{\partial \overline{s_j^2}} \right) \\ &= w_{rj'}^2 \frac{\partial G^2(s_{j'}^2)}{\partial \overline{s_{j'}^2}}, \end{aligned}$$

$$\begin{aligned} g_{rj'}^{3,2} &= \sum_j \left( h_{jj'}^{2,2} \overline{w_{rj}^2} \frac{\partial \overline{G^2(s_j^2)}}{\partial s_j^2} + g_{jj'}^{2,2} \overline{w_{rj}^2} \frac{\partial \overline{G^2(s_j^2)}}{\partial \overline{s_j^2}} \right) \\ &= \overline{w_{rj'}^2} \frac{\partial \overline{G^2(s_{j'}^2)}}{\partial \overline{s_{j'}^2}}, \end{aligned}$$

unde am folosit faptul că $h_{jj'}^{2,2} = 0$, $g_{jj'}^{2,2} = \delta_{jj'}$.

Ultimul caz este acela în care $l_1 = 2$, $l_2 = 3$. În acestă situaţie, avem că $h_{jj'}^{2,2} = 0$, $g_{jj'}^{2,2} = \delta_{jj'}$, deci expresia (3.6.6) devine

$$\frac{\partial^2 E}{\partial \overline{w_{j'i'}^2}\partial \overline{w_{kj}^3}} = \overline{x_{i'}^1} b_{kj'}^{3,2} \overline{x_j^1} + \overline{x_{i'}^1} \delta_{jj'} \frac{\partial \overline{G^2(s_j^2)}}{\partial \overline{s_j^2}} \delta_k^3,$$

în care expresia pentru $b_{kj'}^{3,2}$ a fost dedusă mai sus.

### 3.6.3 Produsul hessianei cu un vector

În unele aplicaţii, cantitatea de interes nu este matricea hessiană $\mathbf{H}$ însăşi, ci produsul ei cu un vector $\mathbf{v}$. Calculul hessianei este o operaţie de ordinul lui $N^2$, unde $N$ este numărul de ponderi şi bias-uri ale reţelei, pe când vectorul $\mathbf{Hv}$ are $N$ elemente, şi calculul lui este de ordinul lui $N$. Din acest motiv, este preferabil să găsim o procedură de calcul direct a produsului $\mathbf{Hv}$, fără a trece prin pasul intermediar al calculului hessianei. Prezentarea urmăreşte pe cea din [209], fiind o adaptare a aceleaşi metode din cazul real în cazul complex.

Fiind dată o reţea neuronală cu funcţia de eroare $E$, din dezvoltarea în serie Taylor de ordinul unu a gradientului $\nabla E$ al funcţiei de eroare, avem că

$$\nabla E(\mathbf{w} + \Delta\mathbf{w}) = \nabla E(\mathbf{w}) + \mathbf{H}(\mathbf{w})\Delta\mathbf{w} + O(||\Delta\mathbf{w}||^2),$$

unde $O(||\Delta \mathbf{w}||^2)$ desemnează termeni de ordinul doi care depind de $||\Delta \mathbf{w}||$, iar $\mathbf{H}(\mathbf{w}) := \nabla^2 E(\mathbf{w})$ reprezintă hessiana funcţiei de eroare. Dacă scriem relaţia de mai sus pentru $\Delta \mathbf{w} = r\mathbf{v}$, unde $r$ este un număr real mic, şi $\mathbf{v}$ este un vector, avem că

$$\mathbf{H}(\mathbf{w})(r\mathbf{v}) = r\mathbf{H}(\mathbf{w})\mathbf{v} = \nabla E(\mathbf{w} + \Delta \mathbf{w}) - \nabla E(\mathbf{w}) + O(r^2),$$

sau, echivalent, că

$$\mathbf{H}(\mathbf{w})\mathbf{v} = \frac{\nabla E(\mathbf{w} + \Delta \mathbf{w}) - \nabla E(\mathbf{w})}{r} + O(r).$$

Trecând la limită pentru $r \to 0$, avem că

$$\mathbf{H}(\mathbf{w})\mathbf{v} = \lim_{r \to 0} \frac{\nabla E(\mathbf{w} + \Delta \mathbf{w}) - \nabla E(\mathbf{w})}{r} = \frac{\partial}{\partial r} \nabla E(\mathbf{w} + r\mathbf{v}) \bigg|_{r=0}.$$

Acum, pentru a calcula exact produsul $\mathbf{H}(\mathbf{w})\mathbf{v}$ vom folosi ceea ce poartă denumirea de tehnica $\mathcal{R}$. Aceasta este o transformare care converteşte un algoritm care calculează gradientul unui sistem într-un algoritm care calculează pe $\mathbf{H}(\mathbf{w})\mathbf{v}$ . Cheia acestei transformări este să definim operatorul

$$\mathcal{R}\{f(\mathbf{w})\} = \frac{\partial}{\partial r} f(\mathbf{w} + r\mathbf{v}) \bigg|_{r=0},$$

astfel încât avem

$$\mathbf{H}(\mathbf{w})\mathbf{v} = \mathcal{R}\{\nabla E(\mathbf{w})\}.$$

Acum, putem să luăm toate ecuaţiile unui algoritm care calculează gradientul, de exemplu metoda gradient, şi să aplicăm operatorul $\mathcal{R}$ pentru fiecare ecuaţie. Deoarece $\mathcal{R}$ este un operator diferenţial, el satisface regulile uzuale ale operatorilor diferenţiali, cum ar fi, de exemplu următoarele reguli:

$$\mathcal{R}\{cf(\mathbf{w})\} = c\mathcal{R}\{f(\mathbf{w})\}.$$
$$\mathcal{R}\{f(\mathbf{w}) + g(\mathbf{w})\} = \mathcal{R}\{f(\mathbf{w})\} + \mathcal{R}\{g(\mathbf{w})\},$$
$$\mathcal{R}\{f(\mathbf{w})g(\mathbf{w})\} = \mathcal{R}\{f(\mathbf{w})\}g(\mathbf{w}) + f(\mathbf{w})\mathcal{R}\{g(\mathbf{w})\},$$
$$\mathcal{R}\{f(g(\mathbf{w}))\} = f'(g(\mathbf{w}))\mathcal{R}\{g(\mathbf{w})\},$$
$$\mathcal{R}\left\{\frac{df(\mathbf{w})}{dt}\right\} = \frac{d\mathcal{R}\{f(\mathbf{w})\}}{dt}.$$

Se mai poate observa uşor că avem

$$\mathcal{R}\{\mathbf{w}\} = \mathbf{v}.$$

Folosind această tehnică, metoda gradient poate fi transformată într-o metodă de calcul al lui $\mathcal{R}\{\nabla E(\mathbf{w})\}$, care este tocmai produsul matricii hessiane cu un vector, $\mathbf{H}(\mathbf{w})\mathbf{v}$, pe care dorim să-l calculăm.

Aplicăm deci operatorul $\mathcal{R}$ formulelor pentru propagarea înainte din metoda gradient

$$s_j^l := \sum_k w_{jk}^l x_k^{l-1} + w_{j0}^l,$$

$$y_j^l := G^l(s_j^l),$$

şi obţinem

$$\mathcal{R}\{s_j^l\} = \sum_k \left(\mathcal{R}\{w_{jk}^l\}x_k^{l-1} + w_{jk}^l\mathcal{R}\{x_k^{l-1}\}\right) + \mathcal{R}\{w_{j0}^l\} = \sum_k \left(v_{jk}^l x_k^{l-1} + w_{jk}^l\mathcal{R}\{x_k^{l-1}\}\right) + v_{j0}^l,$$

şi respectiv

$$\mathcal{R}\{y_j^l\} = \mathcal{R}\{s_j^l\}\frac{\partial G^l(s_j^l)}{\partial s_j^l} + \mathcal{R}\{\overline{s_j^l}\}\frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}}.$$

Apoi, din

$$\frac{\partial E}{\partial \overline{w_{jk}^L}} = \frac{\partial E}{\partial \overline{s_j^L}}\overline{x_k^{L-1}},$$

avem

$$\mathcal{R}\left\{\frac{\partial E}{\partial \overline{w_{jk}^L}}\right\} = \mathcal{R}\left\{\frac{\partial E}{\partial \overline{s_j^L}}\right\}\overline{x_k^{L-1}} + \frac{\partial E}{\partial \overline{s_j^L}}\mathcal{R}\left\{\overline{x_k^{L-1}}\right\},$$

şi din

$$\frac{\partial E}{\partial \overline{s_j^L}} = \frac{1}{2}(\overline{y_j^L - t_j})\frac{\partial G^L(s_j^L)}{\partial \overline{s_j^L}} + \frac{1}{2}(y_j^L - t_j)\frac{\partial \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}},$$

că

$$\begin{aligned}\mathcal{R}\left\{\frac{\partial E}{\partial \overline{s_j^L}}\right\} &= \frac{1}{2}\mathcal{R}\left\{\overline{y_j^L}\right\}\frac{\partial G^L(s_j^L)}{\partial \overline{s_j^L}} + \frac{1}{2}(\overline{y_j^L - t_j})\mathcal{R}\left\{\frac{\partial G^L(s_j^L)}{\partial \overline{s_j^L}}\right\} \\ &+ \frac{1}{2}\mathcal{R}\left\{y_j^L\right\}\frac{\partial \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}} + \frac{1}{2}(y_j^L - t_j)\mathcal{R}\left\{\frac{\partial \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}}\right\}.\end{aligned}$$

Putem scrie că

$$\mathcal{R}\left\{\frac{\partial G^L(s_j^L)}{\partial \overline{s_j^L}}\right\} = \mathcal{R}\{s_j^L\}\frac{\partial^2 G^L(s_j^L)}{\partial \overline{s_j^L}\partial s_j^L} + \mathcal{R}\{\overline{s_j^L}\}\frac{\partial^2 G^L(s_j^L)}{\partial \overline{s_j^L}^2}$$

şi

$$\mathcal{R}\left\{\frac{\partial \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}}\right\} = \mathcal{R}\{s_j^L\}\frac{\partial^2 \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}\partial s_j^L} + \mathcal{R}\{\overline{s_j^L}\}\frac{\partial^2 \overline{G^L(s_j^L)}}{\partial \overline{s_j^L}^2}.$$

Trecem acum la un strat arbitrar $l \in \{1, ..., L-1\}$. Avem din

$$\frac{\partial E}{\partial \overline{w_{jk}^l}} = \frac{\partial E}{\partial \overline{s_j^l}}\overline{x_k^{l-1}},$$

că

$$\mathcal{R}\left\{\frac{\partial E}{\partial \overline{w_{jk}^l}}\right\} = \mathcal{R}\left\{\frac{\partial E}{\partial \overline{s_j^l}}\right\}\overline{x_k^{l-1}} + \frac{\partial E}{\partial \overline{s_j^l}}\mathcal{R}\left\{\overline{x_k^{l-1}}\right\},$$

şi din

$$\frac{\partial E}{\partial \overline{s_j^l}} = \sum_r \left( \frac{\partial E}{\partial s_r^{l+1}} \overline{w_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \frac{\partial E}{\partial \overline{s_r^{l+1}}} w_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right),$$

că

$$\begin{aligned}
\mathcal{R}\left\{\frac{\partial E}{\partial \overline{s_j^l}}\right\} &= \sum_r \mathcal{R}\left\{ \frac{\partial E}{\partial s_r^{l+1}} \overline{w_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \frac{\partial E}{\partial \overline{s_r^{l+1}}} w_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right\} \\
&= \sum_r \mathcal{R}\left\{ \frac{\partial E}{\partial s_r^{l+1}} \right\} \overline{w_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} + \frac{\partial E}{\partial s_r^{l+1}} \overline{v_{rj}^{l+1}} \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} \\
&+ \sum_r \frac{\partial E}{\partial s_r^{l+1}} \overline{w_{rj}^{l+1}} \mathcal{R}\left\{ \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} \right\} + \mathcal{R}\left\{ \frac{\partial E}{\partial \overline{s_r^{l+1}}} \right\} w_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \\
&+ \sum_r \frac{\partial E}{\partial \overline{s_r^{l+1}}} v_{rj}^{l+1} \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} + \frac{\partial E}{\partial \overline{s_r^{l+1}}} w_{rj}^{l+1} \mathcal{R}\left\{ \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right\},
\end{aligned}$$

unde, ca mai sus, putem scrie că

$$\mathcal{R}\left\{ \frac{\partial \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}} \right\} = \mathcal{R}\{s_j^l\} \frac{\partial^2 \overline{G^l(s_j^l)}}{\partial \overline{s_j^l} \partial s_j^l} + \mathcal{R}\{\overline{s_j^l}\} \frac{\partial^2 \overline{G^l(s_j^l)}}{\partial \overline{s_j^l}^2}$$

şi

$$\mathcal{R}\left\{ \frac{\partial G^l(s_j^l)}{\partial \overline{s_j^l}} \right\} = \mathcal{R}\{s_j^l\} \frac{\partial^2 G^l(s_j^l)}{\partial \overline{s_j^l} \partial s_j^l} + \mathcal{R}\{\overline{s_j^l}\} \frac{\partial^2 G^l(s_j^l)}{\partial \overline{s_j^l}^2}.$$

## 3.7 Metode quasi-Newton

După cum am evidenţiat în secţiunea anterioară, principalul dezavantaj al metodei Newton este acela că hessiana şi inversa ei sunt dificil de calculat. Din acest motiv, *metodele quasi-Newton* urmăresc înlocuirea hessianei funcţiei de eroare cu o aproximare a ei, care ar fi preferabil să fie pozitiv definită prin construcţie, pentru a se evita problemele apărute în acest sens în cazul metodei Newton. Prezentarea acestor metode pentru cazul complex este făcută ca în articolul nostru [220].

Astfel, fie $E : \mathbb{C}^{2N} \to \mathbb{R}$ funcţia de eroare a unei reţele neuronale cu valori complexe. Metoda Newton ne dă următoarea regulă de actualizare pentru vectorul $\overset{c}{\mathbf{w}} \in \mathbb{C}^{2N}$ al ponderilor şi bias-urilor reţelei:

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k - (\nabla^2 E(\overset{c}{\mathbf{w}}_k))^{-1} \nabla E(\overset{c}{\mathbf{w}}_k).$$

În metodele quasi-Newton se înlocuieşte inversa hessianei $(\nabla^2 E(\overset{c}{\mathbf{w}}_k))^{-1}$ cu o matrice $\overset{c}{\mathbf{H}}_{k+1}$ care se poate calcula mai uşor. Din dezvoltarea în serie Taylor de ordinul unu a gradientului $\nabla E$ al funcţiei de eroare, avem că

$$\nabla E(\overset{c}{\mathbf{w}}_{k+1}) = \nabla E(\overset{c}{\mathbf{w}}_k) + \nabla^2 E(\overset{c}{\mathbf{w}}_k)(\overset{c}{\mathbf{w}}_{k+1} - \overset{c}{\mathbf{w}}_k).$$

Dacă aproximăm hessiana funcţiei de eroare $\nabla^2 E(\overset{c}{\mathbf{w}}_k)$ cu o matrice $\overset{c}{\mathbf{B}}_{k+1}$ , relaţia de mai sus devine

$$\nabla E(\overset{c}{\mathbf{w}}_{k+1}) - \nabla E(\overset{c}{\mathbf{w}}_k) = \overset{c}{\mathbf{B}}_{k+1}(\overset{c}{\mathbf{w}}_{k+1} - \overset{c}{\mathbf{w}}_k). \tag{3.7.1}$$

Făcând notaţiile $\overset{c}{\mathbf{g}}_k := \nabla E(\overset{c}{\mathbf{w}}_k)$ şi respectiv

$$\overset{c}{\mathbf{p}}_k := \overset{c}{\mathbf{w}}_{k+1} - \overset{c}{\mathbf{w}}_k,$$

şi

$$\overset{c}{\mathbf{q}}_k := \overset{c}{\mathbf{g}}_{k+1} - \overset{c}{\mathbf{g}}_k,$$

ecuaţia (3.7.1) devine

$$\overset{c}{\mathbf{q}}_k = \overset{c}{\mathbf{B}}_{k+1}\overset{c}{\mathbf{p}}_k, \tag{3.7.2}$$

relaţie care poartă denumirea de *ecuaţia secantei.* Aceasta poate fi scrisă echivalent şi sub forma

$$\overset{c}{\mathbf{p}}_k = \overset{c}{\mathbf{H}}_{k+1}\overset{c}{\mathbf{q}}_k, \tag{3.7.3}$$

unde evident avem că $\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{B}}_{k+1}^{-1}$.

Cea mai simplă dintre metodele quasi-Newton este *metoda actualizării de rang unu*, în care se porneşte de la o matrice iniţială $\overset{c}{\mathbf{H}}_0$ hermitiană şi pozitiv definită, şi se calculează iterativ matricea $\overset{c}{\mathbf{H}}_k$ folosind recurenţa

$$\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{H}}_k + a_k \overset{c}{\mathbf{r}}_k \overset{c}{\mathbf{r}}_k^H, \tag{3.7.4}$$

unde $a_k \in \mathbb{R}$ şi vectorul $\overset{c}{\mathbf{r}}_k \in \mathbb{C}^{2N}$ sunt alese astfel încât să fie satisfăcută ecuaţia secantei (3.7.2) sau (3.7.3). Combinând acum ecuaţia (3.7.3) cu (3.7.4), rezultă că

$$\overset{c}{\mathbf{p}}_k = \overset{c}{\mathbf{H}}_{k+1}\overset{c}{\mathbf{q}}_k = (\overset{c}{\mathbf{H}}_k + a_k \overset{c}{\mathbf{r}}_k \overset{c}{\mathbf{r}}_k^H)\overset{c}{\mathbf{q}}_k = \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k + a_k \overset{c}{\mathbf{r}}_k \overset{c}{\mathbf{r}}_k^H \overset{c}{\mathbf{q}}_k.$$

De aici, rezultă că

$$(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)^H = a_k^2 \overset{c}{\mathbf{r}}_k |\overset{c}{\mathbf{r}}_k^H \overset{c}{\mathbf{q}}_k|^2 \overset{c}{\mathbf{r}}_k^H,$$

şi respectiv

$$(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)^H\overset{c}{\mathbf{q}}_k = (a_k \overset{c}{\mathbf{r}}_k \overset{c}{\mathbf{r}}_k^H \overset{c}{\mathbf{q}}_k)^H\overset{c}{\mathbf{q}}_k = a_k |\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{r}}_k|^2,$$

(unde $|z|$ reprezintă modulul numărului complex $z$) pe care dacă le împărţim, obţinem că

$$a_k \overset{c}{\mathbf{r}}_k \overset{c}{\mathbf{r}}_k^H = \frac{(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)^H}{(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)^H\overset{c}{\mathbf{q}}_k},$$

şi întorcându-ne în ecuaţia (3.7.4), obţinem iteraţia pentru calculul aproximării inversei hessianei $\overset{c}{\mathbf{H}}_k$ sub forma

$$\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{H}}_k + \frac{(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)^H}{(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k)^H\overset{c}{\mathbf{q}}_k}, \tag{3.7.5}$$

care este cunoscută sub denumirea de *metoda actualizării de rang unu*. Din faptul că matricea iniţială $\overset{c}{\mathbf{H}}_0$ este hermitiană şi pozitiv definită, rezultă imediat că toate matricile $\overset{c}{\mathbf{H}}_k$ au această proprietate.

Dacă dorim să facem direct aproximarea hessianei $\overset{c}{\mathbf{B}}_k$, pornim la fel cu o matrice iniţială hermitiană şi pozitiv definită $\overset{c}{\mathbf{B}}_0$. Apoi, folosind formula Sherman-Morrison, care spune că dacă $\mathbf{A}$ este o matrice pătratică inversabilă de ordin $2N$ cu elemente complexe, şi $\mathbf{u}, \mathbf{v} \in \mathbb{C}^{2N}$ astfel încât $1 + \mathbf{v}^H \mathbf{A}^{-1} \mathbf{u} \neq 0$, atunci are loc

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^H)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^H\mathbf{A}^{-1}}{1 + \mathbf{v}^H\mathbf{A}^{-1}\mathbf{u}}, \tag{3.7.6}$$

şi ecuaţia (3.7.5), obţinem actualizarea pentru $\overset{c}{\mathbf{B}}_k$ sub forma

$$\overset{c}{\mathbf{B}}_{k+1} = \overset{c}{\mathbf{B}}_k + \frac{(\overset{c}{\mathbf{q}}_k - \overset{c}{\mathbf{B}}_k \overset{c}{\mathbf{p}}_k)(\overset{c}{\mathbf{q}}_k - \overset{c}{\mathbf{B}}_k \overset{c}{\mathbf{p}}_k)^H}{(\overset{c}{\mathbf{q}}_k - \overset{c}{\mathbf{B}}_k \overset{c}{\mathbf{p}}_k)^H \overset{c}{\mathbf{p}}_k}.$$

În continuare, pentru a prezenta metodele quasi-Newton cu actualizări de rang doi, folosim ideile din [76]. Pornim de la ecuaţia secantei scrisă sub forma (3.7.3):

$$\overset{c}{\mathbf{p}}_k = \overset{c}{\mathbf{H}}_{k+1} \overset{c}{\mathbf{q}}_k.$$

Vom defini iteraţia pentru calculul lui $\overset{c}{\mathbf{H}}_{k+1}$ sub forma

$$\overset{c}{\mathbf{H}}_{k+1} = \underset{\substack{\overset{c}{\mathbf{H}} = \overset{c}{\mathbf{H}}^H \\ \overset{c}{\mathbf{p}}_k = \overset{c}{\mathbf{H}} \overset{c}{\mathbf{q}}_k}}{\arg\min} ||\overset{c}{\mathbf{H}} - \overset{c}{\mathbf{H}}_k||, \tag{3.7.7}$$

unde $||\overset{c}{\mathbf{H}}||$ este norma Frobenius a matricii $\overset{c}{\mathbf{H}}$, definită prin $||\overset{c}{\mathbf{H}}|| := \sqrt{\mathrm{Tr}(\overset{c}{\mathbf{H}}\overset{c}{\mathbf{H}}^H)}$, $\mathrm{Tr}(\overset{c}{\mathbf{H}})$ fiind urma matricii $\overset{c}{\mathbf{H}}$. Relaţia (3.7.7) ne spune că vrem să găsim acea matrice hermitiană $\overset{c}{\mathbf{H}}_{k+1}$ care este cea mai apropiată în sensul normei Frobenius de matricea $\overset{c}{\mathbf{H}}_k$, şi care satisface ecuaţia secantei (3.7.3). Se poate demonstra uşor că soluţia pentru (3.7.7) este dată de

$$\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{H}}_k + \frac{(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{H}}_k \overset{c}{\mathbf{q}}_k)\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{q}}_k}.$$

Luând descompunerea Cholesky a matricii $\overset{c}{\mathbf{H}}_{k+1}$ sub forma

$$\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{L}}_{k+1} \overset{c}{\mathbf{L}}_{k+1}^H,$$

se poate demonstra că $\overset{c}{\mathbf{H}}_{k+1}$ este hermitiană şi pozitiv definită dacă şi numai dacă există $\overset{c}{\mathbf{v}}_k \in \mathbb{C}^{2N}$ astfel încât $\overset{c}{\mathbf{L}}_{k+1} \overset{c}{\mathbf{v}}_k = \overset{c}{\mathbf{p}}_k$ şi $\overset{c}{\mathbf{v}}_k = \overset{c}{\mathbf{L}}_{k+1}^H \overset{c}{\mathbf{q}}_k$. Dacă vom lua pe $\overset{c}{\mathbf{L}}_{k+1}$ astfel încât

$$\overset{c}{\mathbf{L}}_{k+1} = \underset{\substack{\overset{c}{\mathbf{L}}=\overset{c}{\mathbf{L}}^H \\ \overset{c}{\mathbf{p}}_k=\overset{c}{\mathbf{L}}\overset{c}{\mathbf{v}}_k}}{\arg\min} ||\overset{c}{\mathbf{L}} - \overset{c}{\mathbf{L}}_k||,$$

rezultă ca mai sus că

$$\overset{c}{\mathbf{L}}_{k+1} = \overset{c}{\mathbf{L}}_k + \frac{(\overset{c}{\mathbf{p}}_k - \overset{c}{\mathbf{L}}_k\overset{c}{\mathbf{v}}_k)\overset{c}{\mathbf{v}}_k^H}{\overset{c}{\mathbf{v}}_k^H\overset{c}{\mathbf{v}}_k}. \tag{3.7.8}$$

Avem în continuare că

$$\overset{c}{\mathbf{v}}_k = \overset{c}{\mathbf{L}}_{k+1}^H\overset{c}{\mathbf{q}}_k = \overset{c}{\mathbf{L}}_k^H\overset{c}{\mathbf{q}}_k + \frac{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k - \overset{c}{\mathbf{v}}_k^H\overset{c}{\mathbf{L}}_k^H\overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{v}}_k^H\overset{c}{\mathbf{v}}_k}\mathbf{v}_k,$$

de unde rezultă că există $\alpha_k \in \mathbb{C}$ astfel încât să avem $\overset{c}{\mathbf{v}}_k = \alpha_k\overset{c}{\mathbf{L}}_k^H\overset{c}{\mathbf{q}}_k$. Obţinem mai departe că

$$\alpha_k = 1 + \frac{\alpha_k\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k - |\alpha_k|^2\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k}{|\alpha_k|^2\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k},$$

şi apoi

$$|\alpha_k|^2 = \frac{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k},$$

iar dacă revenim în (3.7.8), rezultă că

$$\overset{c}{\mathbf{L}}_{k+1} = \overset{c}{\mathbf{L}}_k\left(\mathbf{I}_{2N} \pm \sqrt{\frac{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k}}\frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k} - \frac{\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k}\right).$$

ţinând seama că $\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{L}}_{k+1}\overset{c}{\mathbf{L}}_{k+1}^H$, obţinem în final actualizarea pentru calculul matricii $\overset{c}{\mathbf{H}}_k$:

$$\overset{c}{\mathbf{H}}_{k+1} = \overset{c}{\mathbf{H}}_k + \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k} - \frac{\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{H}}_k\overset{c}{\mathbf{q}}_k}.$$

Aceasta este cunoscută sub denumirea de *metoda Davidon-Fletcher-Powell (DFP)*, a se vedea [85]. Folosind formula Sherman-Morrison dată în (3.7.6), obţinem actualizarea pentru aproximarea $\overset{c}{\mathbf{B}}_k$ a hessianei ca fiind

$$\overset{c}{\mathbf{B}}_{k+1} = \left(\mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k}\right)\overset{c}{\mathbf{B}}_k\left(\mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k}\right) + \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{q}}_k}.$$

Acum, pornind cu ecuaţia secantei sub forma (3.7.2)

$$\overset{c}{\mathbf{B}}_{k+1}\overset{c}{\mathbf{p}}_k = \overset{c}{\mathbf{q}}_k,$$

şi definind iteraţia pentru calculul lui $\overset{c}{\mathbf{B}}_{k+1}$ sub forma

$$\overset{c}{\mathbf{B}}_{k+1} = \underset{\substack{\overset{c}{\mathbf{B}}=\overset{c}{\mathbf{B}}^H \\ \overset{c}{\mathbf{q}}_k=\overset{c}{\mathbf{B}}\overset{c}{\mathbf{p}}_k}}{\arg\min} ||\overset{c}{\mathbf{B}} - \overset{c}{\mathbf{B}}_k||,$$

obţinem, analog ca mai sus, că

$$\overset{c}{\mathbf{B}}_{k+1} = \overset{c}{\mathbf{B}}_k + \frac{(\overset{c}{\mathbf{q}}_k - \overset{c}{\mathbf{B}}_k\overset{c}{\mathbf{p}}_k)\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{p}}_k},$$

unde am folosit aceeaşi normă Frobenius pentru matrici.

Folosind acelaşi raţionament ca în cazul metodei DFP, avem în final următoarea actualizare pentru $\overset{c}{\mathbf{B}}_k$:

$$\mathbf{B}_{k+1} = \overset{c}{\mathbf{B}}_k + \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} - \frac{\overset{c}{\mathbf{B}}_k\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{B}}_k}{\overset{c}{\mathbf{p}}_k^H\overset{c}{\mathbf{B}}_k\overset{c}{\mathbf{p}}_k}.$$

Aceasta poartă denumirea de *metoda Broyden-Fletcher-Goldfarb-Shanno (BFGS)*, a se vedea [238]. Folosind din nou formula Sherman-Morrison (3.7.6), obţinem acum actualizarea pentru aproximarea $\overset{c}{\mathbf{H}}_k$ a inversei hessianei ca fiind

$$\overset{c}{\mathbf{H}}_{k+1} = \left(\mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}\right)\overset{c}{\mathbf{H}}_k\left(\mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}\right) + \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}.$$

O metodă simplificatoare care presupune că $\overset{c}{\mathbf{H}}_k = \mathbf{I}_{2N}$, calculează următoarea direcţie de căutare după formula

$$\overset{c}{\mathbf{d}}_{k+1} = -\overset{c}{\mathbf{H}}_{k+1}\overset{c}{\mathbf{g}}_{k+1},$$

unde dacă facem înlocuirea

$$\begin{aligned}
\overset{c}{\mathbf{H}}_{k+1} &= \left(\mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}\right)\left(\mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}\right) + \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} \\
&= \mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} - \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} + \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{p}}_k^H}{\left(\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k\right)^2} + \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} \\
&= \mathbf{I}_{2N} - \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{q}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} - \frac{\overset{c}{\mathbf{q}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k} + \frac{\overset{c}{\mathbf{p}}_k\overset{c}{\mathbf{p}}_k^H}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}\left(1 + \frac{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{q}}_k^H\overset{c}{\mathbf{p}}_k}\right),
\end{aligned}$$

avem că

$$\begin{aligned}\overset{c}{\mathbf{d}}_{k+1} &= -\overset{c}{\mathbf{g}}_{k+1} + \overset{c}{\mathbf{p}}_k \frac{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k} + \overset{c}{\mathbf{q}}_k \frac{\overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k} - \overset{c}{\mathbf{p}}_k \frac{\overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k} \left(1 + \frac{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k}\right) \\ &= -\overset{c}{\mathbf{g}}_{k+1} + \overset{c}{\mathbf{p}}_k \left(\frac{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k} - \frac{\overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k} \left(1 + \frac{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k}\right)\right) + \overset{c}{\mathbf{q}}_k \frac{\overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k},\end{aligned}$$

sau, scriind compact, că

$$\overset{c}{\mathbf{d}}_{k+1} = -\overset{c}{\mathbf{g}}_{k+1} + A_k \overset{c}{\mathbf{p}}_k + B_k \overset{c}{\mathbf{q}}_k,$$

unde

$$A_k = -\left(1 + \frac{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{q}}_k}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k}\right) \frac{\overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k} + \frac{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k},$$

$$B_k = \frac{\overset{c}{\mathbf{p}}_k^H \overset{c}{\mathbf{g}}_{k+1}}{\overset{c}{\mathbf{q}}_k^H \overset{c}{\mathbf{p}}_k}.$$

Metoda descrisă poartă numele de *one step secant* (a se vedea [23]), şi este o metodă aflată ca performanţă între metoda gradienţilor conjugaţi şi metodele quasi-Newton, mai exact metoda BFGS din care a fost derivată.

## 3.8 Metoda Levenberg-Marquardt

Metoda Levenberg-Marquardt a fost pentru prima dată aplicată reţelelor neuronale în [102]. Prezentarea noastră urmăreşte îndeosebi pe cea din [277]. Pornim de la expresia funcţiei de eroare scrisă sub forma

$$E = \frac{1}{2} \sum_{m=1}^{M} \sum_{j=1}^{c} ||e_j^m||^2 = \frac{1}{2} \sum_{m=1}^{M} \sum_{j=1}^{c} e_j^m \overline{e_j^m},$$

unde $e_j^m := y_j^{L,m} - t_j^m$ reprezintă eroarea pentru fiecare dintre cele $m$ tipare de antrenare. Notăm cu $\mathbf{w}$ vectorul tuturor celor $N$ ponderi şi bias-uri ale reţelei, însă acum vom nota fiecare pondere cu $w_i$. Ca mai sus, vom nota $\overset{c}{\mathbf{w}} = (\mathbf{w}^T, \overline{\mathbf{w}}^T)^T$. Putem forma acum matricea Jacobiană de dimensiune $2cM \times 2N$, ale cărei componente sunt $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatele parţiale ale fiecărei erori de forma $e_j^m$ în raport cu ponderile $w_i$,

$1 \leq i \leq N$:

$$\overset{c}{\mathbf{J}} = \begin{pmatrix} \frac{\partial e_1^1}{\partial w_1} & \frac{\partial e_1^1}{\partial \overline{w_1}} & \cdots & \frac{\partial e_1^1}{\partial w_N} & \frac{\partial e_1^1}{\partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial e_1^M}{\partial w_1} & \frac{\partial e_1^M}{\partial \overline{w_1}} & \cdots & \frac{\partial e_1^M}{\partial w_N} & \frac{\partial e_1^M}{\partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial e_c^1}{\partial w_1} & \frac{\partial e_c^1}{\partial \overline{w_1}} & \cdots & \frac{\partial e_c^1}{\partial w_N} & \frac{\partial e_c^1}{\partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial e_c^M}{\partial w_1} & \frac{\partial e_c^M}{\partial \overline{w_1}} & \cdots & \frac{\partial e_c^M}{\partial w_N} & \frac{\partial e_c^M}{\partial \overline{w_N}} \\ \frac{\partial e_1^1}{\partial w_1} & \frac{\partial e_1^1}{\partial \overline{w_1}} & \cdots & \frac{\partial e_1^1}{\partial w_N} & \frac{\partial e_1^1}{\partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial e_1^M}{\partial w_1} & \frac{\partial e_1^M}{\partial \overline{w_1}} & \cdots & \frac{\partial e_1^M}{\partial w_N} & \frac{\partial e_1^M}{\partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial e_c^1}{\partial w_1} & \frac{\partial e_c^1}{\partial \overline{w_1}} & \cdots & \frac{\partial e_c^1}{\partial w_N} & \frac{\partial e_c^1}{\partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial e_c^M}{\partial w_1} & \frac{\partial e_c^M}{\partial \overline{w_1}} & \cdots & \frac{\partial e_c^M}{\partial w_N} & \frac{\partial e_c^M}{\partial \overline{w_N}} \end{pmatrix}.$$

Acum, hessiana funcţiei de eroare este matricea de dimensiune $2N \times 2N$, ale cărei componente sunt $\mathbb{R}$ şi $\overline{\mathbb{R}}$-derivatele parţiale de ordinul doi ale funcţiei de eroare $E$ în raport cu ponderile $w_i$, $1 \leq i \leq N$:

$$\overset{c}{\mathbf{H}} = \begin{pmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \frac{\partial^2 E}{\partial \overline{w_1} \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} & \frac{\partial^2 E}{\partial \overline{w_1} \partial w_N} \\ \frac{\partial^2 E}{\partial w_1 \partial \overline{w_1}} & \frac{\partial^2 E}{\partial \overline{w_1} \partial \overline{w_1}} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial \overline{w_N}} & \frac{\partial^2 E}{\partial \overline{w_1} \partial \overline{w_N}} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial \overline{w_2} \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} & \frac{\partial^2 E}{\partial \overline{w_2} \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial \overline{w_1}} & \frac{\partial^2 E}{\partial \overline{w_2} \partial \overline{w_1}} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial \overline{w_N}} & \frac{\partial^2 E}{\partial \overline{w_2} \partial \overline{w_N}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial \overline{w_N} \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_N \partial w_N} & \frac{\partial^2 E}{\partial \overline{w_N} \partial w_N} \\ \frac{\partial^2 E}{\partial w_N \partial \overline{w_1}} & \frac{\partial^2 E}{\partial \overline{w_N} \partial \overline{w_1}} & \cdots & \frac{\partial^2 E}{\partial w_N \partial \overline{w_N}} & \frac{\partial^2 E}{\partial \overline{w_N} \partial \overline{w_N}} \end{pmatrix}.$$

Vom defini vectorul $2cM \times 1$

$$\overset{c}{\mathbf{e}} = \begin{pmatrix} e_1^1 \\ \vdots \\ e_1^M \\ \vdots \\ e_c^1 \\ \vdots \\ e_c^M \\ \overline{e_1^1} \\ \vdots \\ \overline{e_1^M} \\ \vdots \\ \overline{e_c^1} \\ \vdots \\ \overline{e_c^M} \end{pmatrix},$$

de unde gradientul $\overset{c}{\mathbf{g}}$ de dimensiune $2N \times 1$ va fi

$$\overset{c}{\mathbf{g}} = \overset{c}{\mathbf{J}}^H \overset{c}{\mathbf{e}}, \tag{3.8.1}$$

pentru că

$$\frac{\partial E}{\partial w_i} = \sum_{m=1}^{M} \sum_{j=1}^{c} \left( \frac{\partial e_j^m}{\partial w_i} \overline{e_j^m} + \frac{\partial \overline{e_j^m}}{\partial w_i} e_j^m \right),$$

şi cele analoage.

Avem acum pentru $\mathbb{R}$- şi $\overline{\mathbb{R}}$-derivatele parţiale de ordinul doi ale funcţiei de eroare $E$ în raport cu ponderile $w_i$, $1 \leq i \leq N$:

$$\frac{\partial^2 E}{\partial w_i \partial w_k} = \sum_{m=1}^{M} \sum_{j=1}^{c} \left( \frac{\partial^2 e_j^m}{\partial w_i \partial w_k} \overline{e_j^m} + \frac{\partial^2 \overline{e_j^m}}{\partial w_i \partial w_k} e_j^m + \frac{\partial e_j^m}{\partial w_i} \frac{\partial \overline{e_j^m}}{\partial w_k} + \frac{\partial \overline{e_j^m}}{\partial w_i} \frac{\partial e_j^m}{\partial w_k} \right),$$

şi cele analoage. Algoritmul Levenberg-Marquardt se bazează pe aproximările $\frac{\partial^2 e_j^m}{\partial w_i \partial w_k} \approx 0$ şi respectiv $\frac{\partial^2 \overline{e_j^m}}{\partial w_i \partial w_k} \approx 0$, şi cele analoage, iar expresia pentru un element al hessianei $\overset{c}{\mathbf{H}}$ devine acum

$$\frac{\partial^2 E}{\partial w_i \partial w_k} = \sum_{m=1}^{M} \sum_{j=1}^{c} \left( \frac{\partial e_j^m}{\partial w_i} \frac{\partial \overline{e_j^m}}{\partial w_k} + \frac{\partial \overline{e_j^m}}{\partial w_i} \frac{\partial e_j^m}{\partial w_k} \right),$$

de unde avem aproximarea

$$\overset{c}{\mathbf{H}} \approx \overset{c}{\mathbf{J}}^H \overset{c}{\mathbf{J}}.$$

Folosind această aproximare, formula de actualizare modificată din cadrul metodei Newton

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k - (\overset{c}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2N})^{-1} \overset{c}{\mathbf{g}}_k,$$

unde $\lambda_k$ este o constantă care se alege astfel încât matricea $\overset{c}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2N}$ să fie pozitiv definită, devine

$$\overset{c}{\mathbf{w}}_{k+1} = \overset{c}{\mathbf{w}}_k - (\overset{c}{\mathbf{J}}_k^H \overset{c}{\mathbf{J}}_k + \lambda_k \mathbf{I}_{2N})^{-1} \overset{c}{\mathbf{J}}_k^H \overset{c}{\mathbf{e}}_k,$$

unde am folosit şi relaţia (3.8.1). Aceasta este regula de actualizare pentru *metoda Levenberg-Marquardt* (a se vedea [174]).

## 3.9 Concluzii

Prezentul capitol este dedicat introducerii unor noi algoritmi de învăţare pentru reţelele neuronale cu valori complexe de tip feedforward, algoritmi care au fost extinşi din domeniul real în domeniul complex. Dintre algoritmii prezentaţi doar metoda gradient (a se vedea, spre exemplu [71, 184]), metoda gradient cu moment [143], algoritmul resilient backpropagation [141] şi algoritmul Levenberg-Marquardt [11] au fost extinşi anterior în domeniul complex, restul reprezentând o contribuţie originală a tezei. Astfel,

- Secţiunea 3.1 face o introducere într-un tip de calcul diferenţial devenit standard pentru literatura de specialitate din domeniul reţelelor neuronale cu valori complexe, şi anume **calculul Wirtinger sau calculul** $\mathbb{CR}$. Acesta reprezintă o alternativă mai naturală la calculul cu derivate parţiale reale, folosit în Capitolul 2, Secţiunea 2.1 pentru a deduce algoritmul backpropagation. Acest tip de calcul este folosit în tot capitolul pentru dezvoltarea algoritmilor de învăţare propuşi.

- Secţiunea 3.2 prezintă algoritmul backpropagation sau **metoda gradient** folosind tipul de calcul diferenţial prezentat în secţiunea anterioară. Puţin surprinzător, rezultatele finale sunt aceleaşi cu cele obţinute în cazul folosirii calculului diferenţial cu derivate parţiale reale.

- În Secţiunea 3.3, pe lângă **metoda gradient cu moment**, au fost introduse **metodele quickprop**, **resilient backpropagation**, **delta-bar-delta** şi **Super-SAB**. Acestea fac parte din clasa metodelor gradient îmbunătăţite, şi constituie un prim pas făcut în direcţia găsirii unor algoritmi care să depăşească performanţele metodei gradient clasice. Ultimele patru din cele cinci metode introduse au reguli de actualizare separate pentru părţile reală şi imaginară ale numerelor complexe implicate, deoarece fac presupuneri asupra semnului derivatelor parţiale, iar aceste derivate parţiale trebuie să aibă valori reale pentru a putea fi comparate cu zero. Această secţiune constituie singura excepţie în care nu au fost folosite derivate de tip Wirtinger.

- O metodă ceva mai complicată, **metoda gradienţilor conjugaţi** a fost prezentată în Secţiunea 3.4. După ce am discutat cazul unei funcţii de eroare pătratice, corespunzător metodei gradienţilor conjugaţi liniară, am prezentat metoda gradienţilor conjugaţi neliniară, în cadrul căreia am dedus formule pentru **metoda gradienţilor conjugaţi cu actualizări Hestenes-Stiefel**, **metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere**, **metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves**, **metoda gradienţilor conjugaţi cu actualizări Dai-Yuan**, **variaţiile algoritmilor Hestenes-Stiefel** şi **Polak-Ribiere** şi **metoda gradienţilor conjugaţi cu reporniri Powell-Beale**.

- O extindere interesantă a metodei gradienţilor conjugaţi este **metoda gradienţilor conjugaţi scalaţi**, dedusă pentru reţelele neuronale cu valori complexe de tip feedforward în Secţiunea 3.5. Această metodă reprezintă o îmbunătăţire a

metodei de bază cu elemente care corectează principalele neajunsuri ale acestui algoritm de învăţare.

- Secţiunea 3.6 prezintă **metoda Newton**. Elementul principal al metodei Newton este hessiana funcţiei de eroare a cărei inversă trebuie calculată exact. Din acest motiv, în cadrul acestei secţiuni, am făcut deducerea completă a metodei de calcul a **hessianei** funcţiei de eroare pentru o reţea neuronală cu valori complexe de tip feedforward, precum şi o particularizare a acestei metode pentru o reţea cu un singur strat ascuns. De asemenea, tot în cadrul acestei secţiuni, am prezentat o metodă de calcul a **produsului hessianei cu un vector**, cantitate care este necesară pentru implementarea multor algoritmi de învăţare.

- O variantă mai simplă din punct de vedere computaţional al metodei Newton este **metoda quasi-Newton**, discutată în Secţiunea 3.7. Această metodă înlocuieşte calculul dificil computaţional al hessianei din metoda Newton, cu calculul iterativ al unei aproximări a inversei acestei matrici. Patru algoritmi din această clasă de metode au fost detaliaţi în această secţiune, şi anume **metoda actualizării de rang unu**, **metoda Davidon-Fletcher-Powell**, **metoda Broyden-Fletcher-Goldfarb-Shanno** şi **metoda one step secant**.

- Ultima metodă prezentată este **metoda Levenberg-Marquardt**, în Secţiunea 3.8. În această metodă, hessiana din metoda Newton este înlocuită cu o altă aproximare a ei, şi anume una în care este folosit un Jacobian care este mai uşor de calculat decât hessiana.

# Capitolul 4

# Algoritmi de învăţare pentru reţele neuronale Clifford

Acest capitol este dedicat deducerii complete a algoritmilor de învăţare pentru reţele neuronale Clifford de tip feedforward. Aceşti algoritmi sunt practic o generalizare $2^n$-dimensională a algoritmilor prezentaţi în capitolul anterior pentru reţele neuronale cu valori complexe. Pe lângă argumentele prezentate în acel capitol cu privire la faptul că performanţele algoritmilor sunt puternic dependente de problema de rezolvat, motivul pentru care am decis să extindem aceiaşi algoritmi şi pentru cazul Clifford este acela că, adesea, nu se cunoaşte de la început care algebră Clifford este cea mai potrivită pentru a reprezenta datele dintr-o anumită aplicaţie.

Astfel, am pus la dispoziţia celor interesaţi o paletă largă de algoritmi, care pot fi aplicaţi pentru reţele neuronale cu valori în diferite algebre Clifford, pentru a determina experimental care este cea mai potrivită configuraţie arhitectură-algoritm de învăţare pentru fiecare problemă de interes. De exemplu, pentru cazul $2$-dimensional există $2$ algebre Clifford, pentru cazul $4$-dimensional există $3$ algebre Clifford, iar pentru cazul $8$-dimensional avem $4$ algebre Clifford cu care se poate experimenta.

Secţiunea 4.1 este dedicată prezentării din nou a metodei gradient pentru antrenarea unei reţele neuronale Clifford, după ce aceasta a fost făcută în Capitolul 2, în Secţiunea 2.4. Ea a fost repetată aici, pentru completitudinea şi claritatea expunerii. Pe parcursul acestui capitol, este folosit calculul diferenţial cu derivate parţiale reale, deoarece nu există în cazul Clifford general ceva asemănător calculului Wirtinger din cazul complex. De asemenea, funcţia de activare a fost considerată ca fiind funcţie Clifford divizat, care tratează separat fiecare componentă a unui număr Clifford, spre deosebire de funcţia Clifford complet, care tratează numărul Clifford ca întreg.

Următoarea secţiune, Secţiunea 4.2, introduce metodele gradient îmbunătăţite: metoda gradient cu moment, metoda quickprop, metoda resilient backpropagation, metoda delta-bar-delta şi metoda SuperSAB. Toate apar aici pentru prima oară, şi constituie o generalizare directă a aceloraşi algoritmi din cazul complex, deoarece şi aceştia au fost deduşi folosind derivate parţiale reale.

După această clasă simplă de algoritmi, Secţiunea 4.3 este dedicată metodei gradienţilor conjugaţi. Prima subsecţiune explică metoda gradienţilor conjugaţi liniară, care stă la baza celei de a doua subsecţiuni, unde sunt deduse expresii pentru metoda gradienţilor conjugaţi cu actualizări Hestenes-Stiefel, metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere, metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves, metoda gradienţilor conjugaţi cu actualizări Dai-Yuan, variaţiile algoritmilor Hestenes-Stiefel şi Polak-Ribiere şi metoda gradienţilor conjugaţi cu reporniri Powell-Beale.

În continuare, Secţiunea 4.4 introduce o variaţiune a metodei gradienţilor conju-

gaţi, şi anume metoda gradienţilor conjugaţi scalaţi, care propune două îmbunătăţiri ale metodei mai sus menţionate.

Cea mai importantă metodă de ordinul doi, metoda Newton, este prezentată în Secţiunea 4.5. Un algoritm pentru calcularea explicită a hessianei este propus într-o subsecţiune a acestei secţiuni, cu o particularizare pentru o reţea cu trei straturi în subsecţiunea următoare, deoarece metoda Newton necesită hessiana pentru calculul actualizării unei ponderi. O simplificare a metodei porneşte de la observaţia că nu este necesar atât calculul hessianei, cât calculul produsului hessianei cu un vector. Astfel, ultima subsecţiune prezintă un algoritm pentru efectuarea acestui calcul.

Datorită faptului că metoda Newton este foarte intensivă computaţional, Secţiunea 4.6 introduce o clasă importantă de metode de ordinul doi, care se bazează pe o aproximare a hessianei, şi anume metodele quasi-Newton. Astfel, sunt deduse progresiv metoda metoda actualizării de rang unu, metoda Davidon-Fletcher-Powell, metoda Broyden-Fletcher-Goldfarb-Shanno şi metoda one step secant.

Secţiunea 4.7 prezintă metoda de învăţare Levenberg-Marquardt, care, pentru cazul Clifford, apare aici pentru prima dată, şi care realizează o aproximare a hessianei din metoda Newton, diferită ca idee de cele folosite în metodele quasi-Newton.

În final, Secţiunea 4.8 este dedicată concluziilor acestui capitol.

## 4.1 Metoda gradient

Să presupunem că avem o reţea neuronală cu valori în algebra Clifford nedegenerată $C\ell_{p,q}$, de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile notate cu $\{2,\ldots,L-1\}$ sunt straturi ascunse. Funcţia de eroare $E:(C\ell_{p,q})^N\to\mathbb{R}$ pentru o asemenea reţea este definită prin

$$E(\mathbf{w})=\frac{1}{2}\sum_{i=1}^{c}||y_i^L-t_i||^2, \tag{4.1.1}$$

unde $||y||$ reprezintă norma euclidiană a lui $y$. $\mathbf{y}^L=(y_i^L)_{1\le i\le c}\in(C\ell_{p,q})^c$ sunt *ieşirile* reţelei, $\mathbf{t}=(t_i)_{1\le i\le c}\in(C\ell_{p,q})^c$ sunt *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{w}\in(C\ell_{p,q})^N$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei.

Dacă notăm cu $w_{jk}^l\in C\ell_{p,q}$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l\in\{2,\ldots,L\}$, putem defini *pasul de actualizare* al ponderii $w_{jk}^l$ în epoca $t$ ca fiind

$$\Delta w_{jk}^l(t)=w_{jk}^l(t+1)-w_{jk}^l(t).$$

Cu această notaţie, pentru *metoda gradient*, *regula de actualizare* pentru ponderea $w_{jk}^l$ este

$$\Delta w_{jk}^l(t)=-\varepsilon\sum_{A\in\mathcal{I}}\frac{\partial E}{\partial[w_{jk}^l]_A}(t)e_A,$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi am notat cu $\frac{\partial E}{\partial[w_{jk}^l]_A}(t)$ derivata parţială a funcţiei $E$ în raport cu fiecare componentă $[w_{jk}^l]_A$ a numărului Clifford $w_{jk}^l$, cu $A\in\mathcal{I}$.

Scris sub formă vectorială, pasul de actualizare devine

$$\Delta\mathbf{w}(t)=\mathbf{w}(t+1)-\mathbf{w}(t),$$

unde $\mathbf{w} \in (C\ell_{p,q})^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient se poate scrie acum în formă vectorială, astfel:

$$\Delta\mathbf{w}(t) = -\varepsilon\nabla E(t),$$

unde $\nabla E(t) := \nabla E(\mathbf{w}(t)) \in (C\ell_{p,q})^N$, este gradientul funcţiei $E$, ale cărui componente $\sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A}(\mathbf{w}(t))e_A$ au fost scrise prescurtat mai sus sub forma $\sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A}(t)e_A$. Prin urmare, pentru a minimiza funcţia de eroare $E$, avem nevoie să calculăm gradientul $\nabla E$, adică derivatele parţiale de forma $\frac{\partial E}{\partial [w_{jk}^l]_A}$, pentru $A \in \mathcal{I}$.

Pentru aceasta, facem următoarele notaţii

$$s_j^l := \sum_k w_{jk}^l \otimes_{p,q} x_k^{l-1}, \tag{4.1.2}$$

$$y_j^l := G^l(s_j^l), \tag{4.1.3}$$

unde (4.1.2) înseamnă că înmulţirea de la reţelele neuronale cu valori reale este înlocuită cu înmulţirea Clifford, $G^l$ este *funcţia de activare* a stratului $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1\leq k\leq d} \in (C\ell_{p,q})^d$ sunt intrările reţelei, şi avem că $x_k^l = y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$. Funcţia de activare este considerată a fi definită pe componente. De pildă, pentru numărul Clifford $x = \sum_{I\in\mathcal{I}}[x]_I e_I \in C\ell_{p,q}$, un exemplu de funcţie de activare este funcţia tangentă hiperbolică pe componente definită prin

$$G\left(\sum_{I\in\mathcal{I}}[x]_I e_I\right) = \sum_{I\in\mathcal{I}}(\tanh[x]_I)e_I.$$

Vom calcula mai întâi actualizarea pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta w_{jk}^L(t) = -\varepsilon \sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w_{jk}^L]_A}(t)e_A.$$

Folosind regula înlănţuirii, putem scrie următorul set de relaţii:

$$\frac{\partial E}{\partial [w_{jk}^L]_A} = \sum_{B\in\mathcal{I}} \frac{\partial E}{\partial [s_j^L]_B}\frac{\partial [s_j^L]_B}{\partial [w_{jk}^L]_A}, \tag{4.1.4}$$

pentru orice $A \in \mathcal{I}$.

În continuare, avem, $\forall A, B \in \mathcal{I}$, că

$$\frac{\partial [s_j^L]_B}{\partial [w_{jk}^L]_A} = \frac{\partial [\sum_k w_{jk}^L \otimes_{p,q} x_k^{L-1}]_B}{\partial [w_{jk}^L]_A} = \frac{\partial [w_{jk}^L \otimes_{p,q} x_k^{L-1}]_B}{\partial [w_{jk}^L]_A}. \tag{4.1.5}$$

Folosind faptul că

$$w_{jk}^L \otimes_{p,q} x_k^{L-1} = \sum_{C,D\in\mathcal{I}} [w_{jk}^L]_C [x_k^{L-1}]_D e_C e_D, \tag{4.1.6}$$

deducem că ecuaţia (4.1.5) se poate scrie

$$\begin{aligned}\frac{\partial[w_{jk}^L \otimes_{p,q} x_k^{L-1}]_B}{\partial[w_{jk}^L]_A} &= \frac{\partial[\sum_{C,D\in\mathcal{I}}[w_{jk}^L]_C[x_k^{L-1}]_D e_C e_D]_B}{\partial[w_{jk}^L]_A} \\ &= \frac{\partial(\sum_{\substack{C,D\in\mathcal{I}\\ \kappa_{C,D}e_Ce_D=e_B}}\kappa_{C,D}[w_{jk}^L]_C[x_k^{L-1}]_D)}{\partial[w_{jk}^L]_A} \\ &= \kappa_{A,D}[x_k^{L-1}]_D, \text{ unde } \kappa_{A,D}e_Ae_D = e_B, \kappa_{A,D}\in\{\pm 1\}.\end{aligned}$$

Am obţinut prin urmare

$$\frac{\partial[s_j^L]_B}{\partial[w_{jk}^L]_A} = \kappa_{A,D}[x_k^{L-1}]_D, \text{ unde } \kappa_{A,D}e_Ae_D = e_B,$$

de unde relaţia (4.1.4) se poate scrie sub forma

$$\frac{\partial E}{\partial[w_{jk}^L]_A} = \sum_{\substack{B\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_B}} \frac{\partial E}{\partial[s_j^L]_B}\kappa_{A,D}[x_k^{L-1}]_D. \tag{4.1.7}$$

Mai departe, notând $\delta_j^L := \frac{\partial E}{\partial s_j^L}$, avem din regula înlănţuirii că

$$[\delta_j^L]_B = \frac{\partial E}{\partial[s_j^L]_B} = \sum_{E\in\mathcal{I}}\frac{\partial E}{\partial[y_j^L]_E}\frac{\partial[y_j^L]_E}{\partial[s_j^L]_B},$$

$\forall B\in\mathcal{I}$, sau, ţinând seama de notaţia (4.1.3) şi de expresia pentru funcţia de eroare (4.1.1), că

$$\begin{aligned}[\delta_j^L]_B &= \sum_{E\in\mathcal{I}}([y_j^L]_E - [t_j]_E)\frac{\partial[G^L(s_j^L)]_E}{\partial[s_j^L]_B} \\ &= ([y_j^L]_B - [t_j]_B)\frac{\partial[G^L(s_j^L)]_B}{\partial[s_j^L]_B},\end{aligned}$$

$\forall B\in\mathcal{I}$, deoarece $[G^L(s_j^L)]_E$ depinde de $[s_j^L]_B$ doar pentru $E=B$, ceea ce înseamnă că $\frac{\partial[G^L(s_j^L)]_E}{\partial[s_j^L]_B} = 0, \forall E\neq B$. Dacă notăm cu $\odot$ înmulţirea pe componente a două numere Clifford, relaţia de mai sus ne dă

$$\delta_j^L = (y_j^L - t_j)\odot\frac{\partial G^L(s_j^L)}{\partial s_j^L}, \tag{4.1.8}$$

unde $\frac{\partial G^L(s_j^L)}{\partial s_j^L}$ reprezintă numărul Clifford al derivatei pe componente a funcţiei de activare $G^L$. De exemplu, dacă $x=\sum_{I\in\mathcal{I}}[x]_I e_I$, atunci

$$\frac{\partial G(x)}{\partial x} = \sum_{I\in\mathcal{I}}(\operatorname{sech}^2[x]_I)e_I,$$

cu funcţia $G$ definită ca în exemplul de mai sus. În final, din ecuaţia (4.1.7), obţinem expresia pentru actualizarea dorită:

$$\Delta w_{jk}^{L}(t) = -\varepsilon \delta_j^L \otimes_{p,q} (x_k^{L-1})^*,$$

unde numărul Clifford $\delta_j^L \in C\ell_{p,q}$ este dat de relaţia (4.1.8), şi $x^*$ reprezintă conjugatul Clifford al numărului Clifford $x$, operaţia de conjugare fiind cea definită în Secţiunea 2.4.

Acum, vom calcula actualizarea pentru o pondere arbitrară $w_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, putem scrie că

$$\Delta w_{jk}^{l}(t) = -\varepsilon \sum_{A \in \mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A} e_A,$$

iar apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial [w_{jk}^l]_A} = \sum_{B \in \mathcal{I}} \frac{\partial E}{\partial [s_j^l]_B} \frac{\partial [s_j^l]_B}{\partial [w_{jk}^l]_A}, \forall A \in \mathcal{I}. \tag{4.1.9}$$

Aplicând din nou regula înlănţuirii, obţinem că

$$\frac{\partial E}{\partial [s_j^l]_B} = \sum_r \sum_{C \in \mathcal{I}} \frac{\partial E}{\partial [s_r^{l+1}]_C} \frac{\partial [s_r^{l+1}]_C}{\partial [s_j^l]_B}, \forall B \in \mathcal{I}, \tag{4.1.10}$$

unde suma se ia după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite legături. Apoi avem că

$$\frac{\partial [s_r^{l+1}]_C}{\partial [s_j^l]_B} = \sum_{D \in \mathcal{I}} \frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} \frac{\partial [y_j^l]_D}{\partial [s_j^l]_B}, \forall B, C \in \mathcal{I}.$$

Ca mai sus, avem, $\forall A, B \in \mathcal{I}$, că

$$\frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} = \frac{\partial [\sum_j w_{rj}^{l+1} \otimes_{p,q} y_j^l]_C}{\partial [y_j^{(l)}]_D} = \frac{\partial [w_{rj}^{l+1} \otimes_{p,q} y_j^l]_C}{\partial [y_j^{(l)}]_D}. \tag{4.1.11}$$

Din

$$w_{rj}^{l+1} \otimes_{p,q} y_j^l = \sum_{E,F \in \mathcal{I}} [w_{rj}^{l+1}]_E [y_j^l]_F e_E e_F, \tag{4.1.12}$$

rezultă că ecuaţia (4.1.11) se poate scrie

$$\begin{aligned}
\frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} &= \frac{\partial [\sum_{E,F \in \mathcal{I}} [w_{rj}^{l+1}]_E [y_j^l]_F e_E e_F]_C}{\partial [y_j^l]_D} \\
&= \frac{\partial (\sum_{\substack{E,F \in \mathcal{I} \\ \kappa_{E,F} e_E e_F = e_C}} \kappa_{E,F} [w_{rj}^{l+1}]_E [y_j^l]_F)}{\partial [y_j^l]_D} \\
&= \kappa_{E,D} [w_{rj}^{l+1}]_E, \text{ unde } \kappa_{E,D} e_E e_D = e_C.
\end{aligned}$$

Deci

$$\frac{\partial [s_r^{l+1}]_C}{\partial [y_j^l]_D} = \kappa_{E,D} [w_{rj}^{l+1}]_E, \text{ unde } \kappa_{E,D} e_E e_D = e_C.$$

şi apoi

$$\begin{aligned}\frac{\partial[s_r^{l+1}]_C}{\partial[s_j^l]_B} &= \sum_{\substack{D\in\mathcal{I}\\ \kappa_{E,D}e_Ee_D=e_C}} \kappa_{E,D}[w_{rj}^{l+1}]_E\frac{\partial[G^l(s_j^l)]_D}{\partial[s_j^l]_B}\\ &= \kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B},\ \text{unde } \kappa_{E,B}e_Ee_B=e_C,\end{aligned}$$

$\forall B,C\in\mathcal{I}$, unde din nou am ţinut cont de faptul că $\frac{\partial[G^l(s_j^l)]_D}{\partial[s_j^l]_B}=0,\forall D\neq B$. Revenind înapoi în ecuaţia (4.1.10), avem succesiv

$$\begin{aligned}\frac{\partial E}{\partial[s_j^l]_B} &= \sum_r\sum_{\substack{C\in\mathcal{I}\\ \kappa_{E,B}e_Ee_B=e_C}}\frac{\partial E}{\partial[s_r^{l+1}]_C}\kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B}\\ &= \sum_r\left(\sum_{\substack{C\in\mathcal{I}\\ \kappa_{E,B}e_Ee_B=e_C}}\kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial E}{\partial[s_r^{l+1}]_C}\right)\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B}\\ &= \sum_r[(w_{rj}^{l+1})^*\otimes_{p,q}\delta_r^{l+1}]_B\frac{\partial[G^l(s_j^l)]_B}{\partial[s_j^l]_B},\end{aligned}$$

$\forall B\in\mathcal{I}$, şi am făcut notaţia $\delta_j^l:=\frac{\partial E}{\partial s_j^l}$. Acum, putem scrie relaţia de mai sus sub forma

$$\delta_j^l=\left(\sum_r(w_{rj}^{l+1})^*\otimes_{p,q}\delta_r^{l+1}\right)\odot\frac{\partial G^l(s_j^l)}{\partial s_j^l},\tag{4.1.13}$$

unde prin $\odot$ am notat înmulţirea pe componente a două numere Clifford.

Apoi, ţinând cont că

$$\frac{\partial[s_j^l]_B}{\partial[w_{jk}^l]_A}=\kappa_{A,D}[x_k^{l-1}]_D,\ \text{unde } \kappa_{A,D}e_Ae_D=e_B,$$

relaţia (4.1.9) devine

$$\begin{aligned}\frac{\partial E}{\partial[w_{jk}^l]_A} &= \sum_{\substack{B\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_B}}\frac{\partial E}{\partial[s_j^l]_B}\kappa_{A,D}[x_k^{l-1}]_D\\ &= [\delta_j^l\otimes_{p,q}(x_k^{l-1})^*]_A,\end{aligned}$$

unde $\delta_j^l:=\frac{\partial E}{\partial s_j^l}$, ca mai sus.

În final am obţinut o formulă de calcul pentru actualizarea ponderii $w_{jk}^l$ asemănătoare cu cea din primul caz tratat, şi anume

$$\Delta w_{jk}^l(t)=-\varepsilon\delta_j^l\otimes_{p,q}(x_k^{l-1})^*.$$

În concluzie, avem următoarea formulă pentru actualizarea ponderii $w_{jk}^l$:

$$\Delta w_{jk}^l(t) = -\varepsilon \delta_j^l \otimes_{p,q} (x_k^{l-1})^*, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left(\sum_r (w_{rj}^{l+1})^* \otimes_{p,q} \delta_r^{l+1}\right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l \leq L-1 \\ (y_j^l - t_j) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l = L \end{cases}.$$

## 4.2 Metode gradient îmbunătăţite

Una dintre cele mai simple clase de algoritmi care au o performanţă mai bună decât metoda gradient clasică în domeniul real este aşa-numita clasă a *metodelor gradient îmbunătăţite*. Algoritmii din această clasă cresc performanţa metodei gradient făcând diferite modificări ad hoc algoritmului original. Există o literatură vastă care prezintă astfel de algoritmi, din care am ales unele dintre cele mai cunoscute, robuste şi bine fundamentate teoretic metode de optimizare pentru a le extinde în domeniul Clifford. Acesta este un prim pas făcut în direcţia găsirii unor algoritmi de optimizare a funcţiei de eroare care să aibă performanţe mai bune decât clasica metodă gradient.

### 4.2.1 Metoda gradient cu moment

Cu notaţiile din secţiunea precedentă, regula de actualizare pentru *metoda gradient cu moment* este

$$\Delta w_{jk}^l(t) = -\varepsilon \sum_{A \in \mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A}(t) e_A + \mu \Delta w_{jk}^l(t-1), \tag{4.2.1}$$

unde parametrul $\mu$ reprezintă *momentul*, şi este, la fel ca rata de învăţare $\varepsilon$, un număr real. Ideea din spatele acestei modificări a metodei gradient clasice este de a da o anumită inerţie mişcării în spaţiul ponderilor pentru a netezi posibilele oscilaţii ale unor paşi succesivi din metoda gradient.

### 4.2.2 Metoda quickprop

Să presupunem că avem o funcţie $f$ definită pe $\mathbb{R}^{2^n}$, al cărei minim dorim să-l găsim. Pentru aceasta, trebuie să căutăm un $x$ astfel încât $f'(x) = 0$. O metodă pentru a face acest lucru, în ipoteza că $f$ este convexă, este să-l calculăm pe $x$ iterativ, folosind metoda Newton:

$$x(t+1) = x(t) + \Delta x(t),$$

unde

$$\nabla^2 f(x(t)) \Delta x(t) = -\nabla f(x(t)), \tag{4.2.2}$$

sau echivalent

$$(\Delta x(t))^T \nabla^2 f(x(t)) = -(\nabla f(x(t)))^T, \tag{4.2.3}$$

unde am ţinut cont de faptul că hessiana este o matrice simetrică (adică $\nabla^2 f(x(t)) = (\nabla^2 f(x(t)))^T$) pentru $f$ indefinit derivabilă, cum este funcţia de eroare în cazul nostru.

Dacă nu putem sau nu dorim să calculăm a doua derivată a lui $f$, putem să o înlocuim printr-o aproximare de tip secantă:

$$\nabla^2 f(x(t))(x(t)-x(t-1)) \simeq \nabla f(x(t)) - \nabla f(x(t-1)). \tag{4.2.4}$$

Înmulţind această expresie la stânga cu $(\Delta x(t))^T$ şi ţinând cont de (4.2.3), avem că

$$-(\nabla f(x(t)))^T \Delta x(t-1) = (\Delta x(t))^T (\nabla f(x(t)) - \nabla f(x(t-1))).$$

Înmulţind această relaţie cu $(\nabla f(x(t)) - \nabla f(x(t-1)))^T$ la dreapta, obţinem

$$\frac{(\nabla f(x(t)))^T \Delta x(t-1)(\nabla f(x(t-1) - \nabla f(x(t)))^T}{||\nabla f(x(t)) - \nabla f(x(t-1))||^2} = (\Delta x(t))^T,$$

şi luând transpusa acestei relaţii, avem

$$\Delta x(t) = \frac{(\nabla f(x(t-1) - \nabla f(x(t)))(\Delta x(t-1))^T \nabla f(x(t))}{||\nabla f(x(t)) - \nabla f(x(t-1))||^2},$$

unde $||x||$ reprezintă norma euclidiană din $\mathbb{R}^{2^n}$ a vectorului $x$.

Folosind formula de mai sus pentru actualizarea ponderilor într-o reţea neuronală cu valori Clifford, unde funcţia $f$ este funcţia de eroare $E$, avem pentru o anumită pondere $w^l_{jk}$ (văzută ca un vector din $\mathbb{R}^{2^n}$), definită ca mai sus, următoarea regulă de actualizare:

$$\Delta w^l_{jk}(t) = \frac{\left(\frac{\partial E}{\partial w^l_{jk}}(t-1) - \frac{\partial E}{\partial w^l_{jk}}(t)\right)(\Delta w^l_{jk}(t-1))^T \frac{\partial E}{\partial w^l_{jk}}(t)}{\left\|\frac{\partial E}{\partial w^l_{jk}}(t-1) - \frac{\partial E}{\partial w^l_{jk}}(t)\right\|^2}, \tag{4.2.5}$$

unde derivatele parţiale $\frac{\partial E}{\partial w^l_{jk}}(t-1)$ şi $\frac{\partial E}{\partial w^l_{jk}}(t)$ sunt exprimări sub formă de vectori din $\mathbb{R}^{2^n}$ ale numerelor Clifford $\sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w^l_{jk}]_A}(t-1)e_A$, respectiv $\sum_{A\in\mathcal{I}} \frac{\partial E}{\partial [w^l_{jk}]_A}(t)e_A$.

Această schemă de actualizare este cunoscută sub numele de algoritmul *quickprop*, şi a fost pentru prima dată propus de către Fahlman în [80]. Expresia de actualizare este asemănătoare în cazul Clifford cu cazul real. Pornirea algoritmului necesită doar un singur pas făcut cu metoda gradient, pentru a calcula pasul de actualizare în prima epocă. Euristica din spatele algoritmului este aceea că pasul de actualizare aproximează funcţia de eroare în raport cu fiecare pondere printr-un polinom de gradul doi, pentru care formula (4.2.4) este exactă. Această aproximare presupune că fiecare pondere este independentă de toate celelalte şi că polinomul de gradul doi are un minim, şi nu un maxim. De asemenea, o limită a valorii maxime a pasului de actualizare trebuie impusă pentru a evita probleme în cazul în care numitorul din (4.2.5) devine foarte mic.

### 4.2.3 Metoda RPROP

Pentru a elimina influenţa potenţial negativă a mărimii derivatei parţiale asupra pasului de actualizare, Riedmiller şi Braun [226] au conceput algoritmul *resilient backpropagation*, sau prescurtat *RPROP*. Ideea de bază a algoritmului este de a înlocui derivata parţială din formula de actualizare a ponderilor, şi de a folosi doar semnul ei pentru a calcula o cantitate notată $\Delta^l_{jk}$, care va fi apoi folosită pentru actualizarea ponderii $w^l_{jk}$. Pentru că suntem în planul Clifford, trebuie să tratăm fiecare componentă a unui număr Clifford separat. Astfel, formulele pentru calcularea cantităţilor $\Delta^l_{jk}$ sunt:

$$[\Delta^l_{jk}(t)]_I = \begin{cases} \eta^+[\Delta^l_{jk}(t-1)]_I, & \text{dacă } \frac{\partial E}{\partial [w^l_{jk}]_I}(t-1) \times \frac{\partial E}{\partial [w^l_{jk}]_I}(t) > 0 \\ \eta^-[\Delta^l_{jk}(t-1)]_I, & \text{dacă } \frac{\partial E}{\partial [w^l_{jk}]_I}(t-1) \times \frac{\partial E}{\partial [w^l_{jk}]_I}(t) < 0 \\ [\Delta^l_{jk}(t-1)]_I, & \text{altfel} \end{cases}, \forall I \in \mathcal{I}.$$

Aceste formule depind de parametrii reali $\eta^+$ şi $\eta^-$, care trebuie să satisfacă

$$0 < \eta^- < 1 < \eta^+,$$

şi sunt de obicei folosiţi cu valorile $\eta^- = 0.5$ şi $\eta^+ = 1.2$. Aceste formule spun că de fiecare dată când una dintre derivatele parţiale ale funcţiei de eroare în raport cu $[w^l_{jk}]_I$ îşi schimbă semnul, ceea ce indică faptul că ultima actualizare a fost prea mare şi că algoritmul a sărit peste un minim local, cantităţile $[\Delta^l_{jk}]_I$ sunt micşorate cu un factor de $\eta^-$, $\forall I \in \mathcal{I}$. Dacă acele derivate parţiale au acelaşi semn, aceste cantităţi sunt mărite cu un factor de $\eta^+$ pentru a accelera convergenţa.

Odată ce am calculat valorile $[\Delta^l_{jk}]_I$, putem scrie formulele pentru actualizarea componentelor ponderii $w^l_{jk}$:

$$[\Delta w^l_{jk}(t)]_I = \begin{cases} -[\Delta^l_{jk}(t)]_I, & \text{dacă } \frac{\partial E}{\partial [w^l_{jk}]_I}(t) > 0 \\ [\Delta^l_{jk}(t)]_I, & \text{dacă } \frac{\partial E}{\partial [w^l_{jk}]_I}(t) < 0 \\ 0, & \text{altfel} \end{cases}, \forall I \in \mathcal{I}.$$

Se poate vedea că regula de actualizare este simplă: dacă derivata este pozitivă, ceea ce înseamnă o creştere a erorii, ponderea este micşorată cu $[\Delta^l_{jk}]_I$, iar dacă derivata este negativă, aceste valori sunt adunate la pasul de actualizare a ponderii $[w^l_{jk}]_I$. La începutul algoritmului, trebuie să dăm o valoare pentru cantităţile $[\Delta^l_{jk}(0)]_I$, iar această valoare o luăm ca fiind $0.07$.

### 4.2.4 Metoda delta-bar-delta

O altă cale de a îmbunătăţi performanţa metodei gradient este de a introduce o rată de învăţare diferită pentru fiecare pondere a reţelei, şi de a concepe o procedură pentru actualizarea acestor rate de învăţare pe parcursul antrenării reţelei. Metoda gradient devine în acest caz

$$[\Delta w^l_{jk}(t)]_I = -[\varepsilon^l_{jk}(t)]_I \frac{\partial E}{\partial [w^l_{jk}]_I}(t), \forall I \in \mathcal{I},$$

unde fiecare rată de învăţare $\varepsilon^l_{jk}(t)$ depinde de ponderea $w^l_{jk}$ şi de epoca $t$. Dintru început, facem observaţia că rata de învăţare $\varepsilon$ este un număr real în metoda gradient pentru reţele cu valori Clifford, în vreme ce în acest nou algoritm, rata de învăţare este un număr Clifford. Formulele pentru actualizarea componentelor ratei de învăţare $\varepsilon^l_{jk}(t)$ sunt:

$$[\varepsilon^l_{jk}(t)]_I = \begin{cases} \kappa + [\varepsilon^l_{jk}(t-1)]_I, & \text{dacă } \frac{\partial \bar{E}}{\partial [w^l_{jk}]_I}(t-1) \times \frac{\partial E}{\partial [w^l_{jk}]_I}(t) > 0 \\ \eta^-[\varepsilon^l_{jk}(t-1)]_I, & \text{dacă } \frac{\partial \bar{E}}{\partial [w^l_{jk}]_I}(t-1) \times \frac{\partial E}{\partial [w^l_{jk}]_I}(t) < 0 \\ [\varepsilon^l_{jk}(t-1)]_I, & \text{altfel} \end{cases}, \forall I \in \mathcal{I},$$

unde am făcut notaţiile

$$\frac{\partial \bar{E}}{\partial [w_{jk}^l]_I}(t) = (1-\theta)\frac{\partial E}{\partial [w_{jk}^l]_I}(t) + \theta \frac{\partial \bar{E}}{\partial [w_{jk}^l]_I}(t-1), \forall I \in \mathcal{I}.$$

Parametrul $\eta^-$ trebuie să satisfacă relaţia $0 < \eta^- < 1$, şi este de obicei considerat, ca în algoritmul RPROP, ca fiind $0.5$. Se poate lua de asemenea $\kappa = 10^{-6}$ şi $\theta = 0.7$, însă aceste valori sunt de obicei determinate empiric.

Ideea de bază a acestei metode este următoarea: dacă două derivate parţiale consecutive în raport cu $[w_{jk}^l]_I$ au acelaşi semn, atunci rata de învăţare este mărită cu o constantă de valoare mică $\kappa$ pentru a accelera învăţarea. O schimbare a semnului derivatelor parţiale în raport cu $[w_{jk}^l]_I$ arată că s-a trecut peste minimul local, ceea ce înseamnă că pasul de învăţare a fost prea mare. În consecinţă, rata de învăţare este micşorată prin înmulţire cu un factor pozitiv subunitar $\eta^-$. Pentru a asigura convergenţa, derivata parţială anterioară $\frac{\partial E}{\partial [w_{jk}^l]_I}(t-1)$ a fost înlocuită cu media ponderată exponenţial între derivata parţială curentă şi cea anterioară, având pe $\theta$ ca bază a acestei medii şi epoca în calitate de exponent. Această medie ponderată a fost notată cu $\frac{\partial \bar{E}}{\partial [w_{jk}^l]_I}(t)$, şi a fost definită mai sus.

Algoritmul pe care tocmai l-am descris a fost propus de Jacobs în [133], şi este numit algoritmul *delta-bar-delta*, deoarece, în acel articol, autorul a notat derivata parţială $\frac{\partial E}{\partial w_{jk}^l}(t)$ cu $\delta$ (delta), şi media ponderată $\frac{\partial \bar{E}}{\partial w_{jk}^l}(t)$ cu $\bar{\delta}$ (delta-bar).

### 4.2.5 Metoda SuperSAB

Ultimul algoritm din seria metodelor gradient îmbunătăţite se numeşte *SuperSAB* şi a fost propus de Tollenaere în [250]. Este foarte asemănător cu algoritmul delta-bar-delta, în aceea că pentru fiecare pondere a reţelei este folosită o rată de învăţare diferită. Prin urmare, ca în cazul algoritmului anterior, regula de actualizare este

$$[\Delta w_{jk}^l(t)]_I = -[\varepsilon_{jk}^l(t)]_I \frac{\partial E}{\partial [w_{jk}^l]_I}(t), \forall I \in \mathcal{I},$$

cu excepţia faptului că aici rata de învăţare $\varepsilon_{jk}^l(t)$ corespunzătoare ponderii $w_{jk}^l$ şi epocii $t$ are o formulă de actualizare diferită. De fapt, se poate spune că algoritmul SuperSAB este o combinaţie între algoritmii RPROP şi delta-bar-delta, pentru că formulele pentru actualizarea componentelor lui $\varepsilon_{jk}^l(t)$ sunt:

$$[\varepsilon_{jk}^l(t)]_I = \begin{cases} \eta^+[\varepsilon_{jk}^l(t-1)]_I, & \text{dacă } \frac{\partial E}{\partial [w_{jk}^l]_I}(t-1) \times \frac{\partial E}{\partial [w_{jk}^l]_I}(t) > 0 \\ \eta^-[\varepsilon_{jk}^l(t-1)]_I, & \text{dacă } \frac{\partial E}{\partial [w_{jk}^l]_I}(t-1) \times \frac{\partial E}{\partial [w_{jk}^l]_I}(t) < 0 \\ [\varepsilon_{jk}^l(t-1)]_I, & \text{altfel} \end{cases}, \forall I \in \mathcal{I},$$

unde parametrii $\eta^+$ şi $\eta^-$ trebuie să satisfacă

$$0 < \eta^- < 1 < \eta^+,$$

exact ca în algoritmul resilient backpropagation. Datorită acestei asemănări, se pot lua aceleaşi valori pentru aceşti parametri ca în cazul algoritmului RPROP, şi anume $\eta^- = 0.5$ şi $\eta^+ = 1.2$.

Ideea este destul de uşor de văzut: în loc să adunăm un termen la rata de învăţare dacă derivatele parţiale au acelaşi semn, facem o înmulţire cu un factor pozitiv

supraunitar $\eta^+$ a acesteia, ca în cazul algoritmului RPROP. În acest caz, ultimele două derivate parţiale au fost luate în considerare cu valorile iniţiale, fără a folosi media ponderată, ca pentru algoritmul delta-bar-delta.

Toţi aceşti algoritmi sunt strategii de adaptare locală, în care pentru actualizarea fiecărei ponderi este folosită informaţie care este proprie doar acelei ponderi, fără a avea o cunoaştere globală a întregii stări a reţelei, aşa cum fac strategiile de adaptare globală. Deşi aceste strategii folosesc mai puţină informaţie, şi operaţiile de actualizare se fac mai uşor şi mai repede, pentru multe aplicaţii s-a dovedit că ele au o performanţă mai bună decât strategiile globale.

Una dintre principalele diferenţe între primii doi algoritmi şi următorii trei, este aceea că regulile de actualizare (4.2.1) şi (4.2.5) depind de numărul Clifford ca întreg, şi nu separat de componentele sale ca regulile de actualizare ale celorlalţi trei algoritmi. Motivul principal al acestei diferenţe este că aceşti algoritmi fac presupuneri despre valoarea derivatei parţiale a funcţiei de eroare în raport cu ponderea care este actualizată, presupuneri care implică realizarea de comparaţii. Este cunoscut că, spre deosebire de numerele reale, numerele Clifford nu au o ordonare naturală, şi prin urmare nu există un analog al inegalităţilor de numere reale între numere Clifford. În consecinţă, pentru RPROP, delta-bar-delta şi SuperSAB, trebuie să dăm reguli de actualizare diferite pentru componentele fiecărei ponderi, care depind de presupuneri făcute asupra derivatelor parţiale în raport cu respectivele componente. Cu excepţia acestui fapt, putem observa că regulile de actualizare arată exact la fel în domeniul Clifford ca în domeniul real, şi nu pun niciun fel de probleme de implementare.

## 4.3 Metoda gradienţilor conjugaţi

Metodele gradienţilor conjugaţi sunt un caz particular dintr-o clasă mai largă de *algoritmi de căutare liniară*. Pentru a minimiza funcţia de eroare a unei reţele neuronale, sunt necesari mai mulţi paşi prin spaţiul valorilor ponderilor pentru a găsi acea valoare a ponderilor pentru care minimul funcţiei de eroare este atins. Fiecare pas este determinat de o *direcţie de căutare* şi de un număr real care ne spune cât de mult trebuie să ne deplasăm în acea direcţie. În metoda gradient clasică, direcţia de căutare este aceea a gradientului cu semn schimbat, iar numărul real este rata de învăţare. În cazul general, putem considera o anumită direcţie de căutare, şi apoi putem determina minimul funcţiei de eroare în acea direcţie, obţinând în acest fel acel număr real care ne spune cât de mult trebuie să ne deplasăm în direcţia de căutare. Acesta reprezintă *algoritmul de căutare liniară*, şi se află la baza unei familii de metode care au o performanţă mai bună decât metoda gradient. Deducerea algoritmilor gradienţilor conjugaţi prezentată mai jos, urmăreşte în principal referinţa [46], care la rândul ei urmăreşte referinţa [140], unul dintre primele articole în care se aplică aceşti algoritmi pentru antrenarea reţelelor neuronale, un altul fiind [64].

Să presupunem că avem o reţea neuronală cu valori Clifford care are funcţia de eroare notată cu $E$, un vector al ponderilor $N$-dimensional $\mathbf{w}$, şi notăm $\overset{\mathcal{R}}{\mathbf{w}} = (([\mathbf{w}]_I^T)_{I\in\mathcal{I}})^T \in \mathbb{R}^{2^n N}$. Trebuie să găsim vectorul ponderilor $\overset{\mathcal{R}}{\mathbf{w}}^*$ care minimizează funcţia $E(\overset{\mathcal{R}}{\mathbf{w}})$. Să presupunem că facem paşi iterativi prin spaţiul ponderilor pentru a găsi valoarea lui $\overset{\mathcal{R}}{\mathbf{w}}^*$, sau o foarte bună aproximare a ei. Mai mult, să presupunem că la pasul $k$ în iteraţie, vrem ca direcţia de căutare să fie $\overset{\mathcal{R}}{\mathbf{p}}_k$, unde $\overset{\mathcal{R}}{\mathbf{p}}_k$ este evident un vector $2^n N$-dimensional. În acest caz, următoarea valoare pentru vectorul ponderilor este dată prin

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k + \lambda_k \overset{\mathcal{R}}{\mathbf{p}}_k,$$

unde parametrul $\lambda_k$ este un număr real care ne spune cât de mult în direcţia $\overset{\mathcal{R}}{\mathbf{p}}_k$ vrem să mergem, ceea ce înseamnă că $\lambda_k$ trebuie ales pentru a minimiza funcţia

$$E(\lambda) = E(\overset{\mathcal{R}}{\mathbf{w}}_k + \lambda \overset{\mathcal{R}}{\mathbf{p}}_k).$$

Aceasta este o funcţie reală de o variabilă reală, ceea ce înseamnă că minimul ei este atins când

$$\frac{\partial E(\lambda)}{\partial \lambda} = \frac{\partial E(\overset{\mathcal{R}}{\mathbf{w}}_k + \lambda \overset{\mathcal{R}}{\mathbf{p}}_k)}{\partial \lambda} = 0.$$

Folosind regula înlănţuirii, putem scrie că

$$\begin{aligned} \frac{\partial E(\overset{\mathcal{R}}{\mathbf{w}}_k + \lambda \overset{\mathcal{R}}{\mathbf{p}}_k)}{\partial \lambda} &= \sum_{j=1}^{N} \sum_{I \in \mathcal{I}} \frac{\partial E(\overset{\mathcal{R}}{\mathbf{w}}_k + \lambda \overset{\mathcal{R}}{\mathbf{p}}_k)}{\partial [w_{j,k} + \lambda p_{j,k}]_I} \frac{\partial [w_{j,k} + \lambda p_{j,k}]_I}{\partial \lambda} \\ &= \sum_{j=1}^{N} \sum_{I \in \mathcal{I}} \frac{\partial E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1})}{\partial [w_{j,k+1}]_I} [p_{j,k}]_I \\ &= \langle \nabla E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}), \overset{\mathcal{R}}{\mathbf{p}}_k \rangle, \end{aligned} \tag{4.3.1}$$

unde $\langle \overset{\mathcal{R}}{\mathbf{a}}, \overset{\mathcal{R}}{\mathbf{b}} \rangle$ este *produsul scalar euclidian* din $\mathbb{R}^{2^n N}$, dat prin

$$\langle \overset{\mathcal{R}}{\mathbf{a}}, \overset{\mathcal{R}}{\mathbf{b}} \rangle = \langle (([\mathbf{a}]_I^T)_{I \in \mathcal{I}})^T, (([\mathbf{b}]_I^T)_{I \in \mathcal{I}})^T \rangle = \sum_{I \in \mathcal{I}} \langle [\mathbf{a}]_I, [\mathbf{b}]_I \rangle = \sum_{j=1}^{N} \sum_{I \in \mathcal{I}} [a_j]_I [b_j]_I,$$

pentru orice $\overset{\mathcal{R}}{\mathbf{a}}, \overset{\mathcal{R}}{\mathbf{b}} \in \mathbb{R}^{2^n N}$, şi gradientul $\nabla E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1})$ este definit prin

$$\nabla E(\overset{\mathcal{R}}{\mathbf{w}}) := \left( \frac{\partial E}{\partial \overset{\mathcal{R}}{\mathbf{w}}} (\overset{\mathcal{R}}{\mathbf{w}}) \right)^T.$$

Prin urmare, dacă notăm $\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}) := \nabla E(\overset{\mathcal{R}}{\mathbf{w}})$, atunci (4.3.1) poate fi scrisă sub forma

$$\langle \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}), \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0, \tag{4.3.2}$$

ceea ce arată că vectorii $\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1})$ şi $\overset{\mathcal{R}}{\mathbf{p}}_k$ sunt ortogonali în $\mathbb{R}^{2^n N}$.

Următoarea direcţie de căutare $\overset{\mathcal{R}}{\mathbf{p}}_{k+1}$ este aleasă astfel încât componenta gradientului paralelă cu direcţia de căutare anterioară $\overset{\mathcal{R}}{\mathbf{p}}_k$ rămâne zero. În consecinţă, avem că

$$\langle \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1} + \lambda \overset{\mathcal{R}}{\mathbf{p}}_{k+1}), \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0.$$

Din dezvoltarea în serie Taylor până la gradul întâi, avem că

$$\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1} + \lambda \overset{\mathcal{R}}{\mathbf{p}}_{k+1}) = \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) + \nabla \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1})^T \lambda \overset{\mathcal{R}}{\mathbf{p}}_{k+1},$$

şi apoi, dacă ţinem cont de (4.3.2), obţinem că

$$\lambda \langle \nabla \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1})^T \overset{\mathcal{R}}{\mathbf{p}}_{k+1}, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0,$$

ceea ce este echivalent cu

$$\langle \mathbf{H}^T (\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) \overset{\mathcal{R}}{\mathbf{p}}_{k+1}, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0,$$

sau, mai departe, cu

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_{k+1}, \mathbf{H}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0, \tag{4.3.3}$$

unde am notat prin $\mathbf{H}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1})$ hessiana funcţiei de eroare $E(\overset{\mathcal{R}}{\mathbf{w}})$, deoarece $\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}) = \nabla E(\overset{\mathcal{R}}{\mathbf{w}})$, şi deci $\nabla \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}) = \frac{\partial^2 E}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T \partial \overset{\mathcal{R}}{\mathbf{w}}}(\overset{\mathcal{R}}{\mathbf{w}}) = \frac{\partial^2 E}{\partial \overset{\mathcal{R}}{\mathbf{w}} \partial \overset{\mathcal{R}}{\mathbf{w}}^T}(\overset{\mathcal{R}}{\mathbf{w}})$ este hessiana.

Direcţiile de căutare care satisfac ecuaţia (4.3.3) se numesc *direcţii conjugate*. Metoda direcţiilor conjugate construieşte direcţiile de căutare $\overset{\mathcal{R}}{\mathbf{p}}_k$, astfel încât fiecare nouă direcţie este conjugată cu toate cele anterioare.

### 4.3.1 Metoda gradienţilor conjugaţi liniară

În continuare, vom explica metoda gradienţilor conjugaţi liniară. Pentru aceasta, să considerăm o funcţie de eroare de forma

$$E(\overset{\mathcal{R}}{\mathbf{w}}) = E_0 + \mathbf{b}^T \overset{\mathcal{R}}{\mathbf{w}} + \frac{1}{2} \overset{\mathcal{R}}{\mathbf{w}}^T \mathbf{H} \overset{\mathcal{R}}{\mathbf{w}}, \tag{4.3.4}$$

unde $\mathbf{b}$ şi $\mathbf{H}$ sunt constante, şi matricea $\mathbf{H}$ este presupusă a fi pozitiv definită. Gradientul acestei funcţii este dat de

$$\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}) = \mathbf{b} + \mathbf{H} \overset{\mathcal{R}}{\mathbf{w}}. \tag{4.3.5}$$

Vectorul ponderilor $\overset{\mathcal{R}}{\mathbf{w}}^*$ care minimizează funcţia $E(\overset{\mathcal{R}}{\mathbf{w}})$ satisface ecuaţia

$$\mathbf{b} + \mathbf{H} \overset{\mathcal{R}}{\mathbf{w}}^* = 0. \tag{4.3.6}$$

După cum am văzut mai devreme din ecuaţia (4.3.3), o mulţime de $2^n N$ vectori nenuli $\{\overset{\mathcal{R}}{\mathbf{p}}_1, \overset{\mathcal{R}}{\mathbf{p}}_2, \ldots, \overset{\mathcal{R}}{\mathbf{p}}_{2^n N}\} \subset \mathbb{R}^{2^n N}$ se zice a fi *conjugată faţă de matricea pozitiv definită* $\mathbf{H}$ dacă şi numai dacă

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_i, \mathbf{H} \overset{\mathcal{R}}{\mathbf{p}}_j \rangle = 0, \forall i \neq j. \tag{4.3.7}$$

Este uşor de văzut că, în aceste condiţii, mulţimea de $2^n N$ vectori este de asemenea liniar independentă, ceea ce înseamnă că aceşti vectori formează o bază în $\mathbb{R}^{2^n N}$. Dacă începem din punctul iniţial notat $\overset{\mathcal{R}}{\mathbf{w}}_1$ şi vrem să găsim valoarea lui $\overset{\mathcal{R}}{\mathbf{w}}^*$ care minimizează funcţia de eroare dată în (4.3.4), luând în considerare remarcile anterioare, putem scrie că

$$\overset{\mathcal{R}}{\mathbf{w}}^* - \overset{\mathcal{R}}{\mathbf{w}}_1 = \sum_{i=1}^{2N} \alpha_i \overset{\mathcal{R}}{\mathbf{p}}_i. \tag{4.3.8}$$

Acum, dacă punem

$$\overset{\mathcal{R}}{\mathbf{w}}_k = \overset{\mathcal{R}}{\mathbf{w}}_1 + \sum_{i=1}^{k-1} \alpha_i \overset{\mathcal{R}}{\mathbf{p}}_i, \tag{4.3.9}$$

atunci (4.3.8) poate fi scrisă în formă iterativă, astfel

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{4.3.10}$$

ceea ce înseamnă că valoarea lui $\overset{\mathcal{R}}{\mathbf{w}}^*$ poate fi determinată în cel mult $2^n N$ paşi pentru funcţia de eroare (4.3.4), folosind iteraţia de mai sus. Mai trebuie să determinăm parametrii reali $\alpha_k$ care ne spun cât de mult trebuie să avansăm în fiecare dintre cele $2^n N$ direcţii conjugate $\overset{\mathcal{R}}{\mathbf{p}}_k$.

Pentru aceasta, înmulţim ecuaţia (4.3.8) cu $\mathbf{H}$ la stânga, şi luăm produsul scalar euclidian din $\mathbb{R}^{2^n N}$ cu $\overset{\mathcal{R}}{\mathbf{p}}_k$. Ţinând cont de ecuaţia (4.3.6), obţinem că

$$-\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{b} + \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_1 \rangle = \sum_{i=1}^{N} \alpha_i \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_i \rangle.$$

Dar, pentru că direcţiile $\overset{\mathcal{R}}{\mathbf{p}}_k$ sunt conjugate faţă de matricea $\mathbf{H}$, avem din (4.3.7) că $\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_i \rangle = 0, \forall i \neq k$, şi prin urmare ecuaţia de mai sus dă următoarea expresie pentru calculul lui $\alpha_k$:

$$\alpha_k = -\frac{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{b} + \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_1 \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k \rangle}. \tag{4.3.11}$$

Acum, dacă înmulţim ecuaţia (4.3.9) cu $\mathbf{H}$ la stânga şi luăm acelaşi produs scalar cu $\overset{\mathcal{R}}{\mathbf{p}}_k$, avem că:

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_k \rangle = \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_1 \rangle + \sum_{i=1}^{k-1} \alpha_i \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_i \rangle,$$

sau, ţinând cont că $\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_i \rangle = 0, \forall i \neq k$, avem că

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_k \rangle = \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_1 \rangle,$$

şi astfel relaţia (4.3.11) pentru calculul lui $\alpha_k$ devine:

$$\alpha_k = -\frac{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{b} + \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k \rangle} = -\frac{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k \rangle}, \tag{4.3.12}$$

unde $\overset{\mathcal{R}}{\mathbf{g}}_k := \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_k) = \mathbf{b} + \mathbf{H}\overset{\mathcal{R}}{\mathbf{w}}_k$, după cum rezultă din relaţia (4.3.5).

În final, mai trebuie doar să construim direcţiile mutual conjugate $\overset{\mathcal{R}}{\mathbf{p}}_k$. Pentru aceasta, prima direcţie este iniţializată cu gradientul cu semn schimbat al funcţiei de eroare în punctul iniţial $\overset{\mathcal{R}}{\mathbf{w}}_1$, i.e. $\overset{\mathcal{R}}{\mathbf{p}}_1 = -\overset{\mathcal{R}}{\mathbf{g}}_1$. Avem următoarea regulă de actualizare pentru direcţiile conjugate:

$$\overset{\mathcal{R}}{\mathbf{p}}_{k+1} = -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + \beta_k \overset{\mathcal{R}}{\mathbf{p}}_k. \tag{4.3.13}$$

Luând produsul scalar cu $\mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k$, şi impunând condiţia de conjugare $\langle \overset{\mathcal{R}}{\mathbf{p}}_{k+1}, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0$, obţinem că

$$\beta_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \mathbf{H}\overset{\mathcal{R}}{\mathbf{p}}_k \rangle}. \tag{4.3.14}$$

Se poate arăta uşor prin inducţie că aplicarea repetată a relaţiilor (4.3.13) şi (4.3.14), dă o mulţime de direcţii mutual conjugate faţă de matricea pozitiv definită $\mathbf{H}$.

### 4.3.2 Metoda gradienţilor conjugaţi neliniară

Până acum, am tratat cazul unei funcţii de eroare polinomiale de gradul doi care are o matrice hessiană pozitiv definită $\mathbf{H}$. Dar în aplicaţii practice, funcţia de eroare poate fi departe de a fi polinomială de gradul doi, şi astfel expresiile pentru calculul lui $\alpha_k$ şi $\beta_k$ pe care le-am dedus mai sus, s-ar putea să nu fie la fel de precise ca în cazul polinomial. Mai mult, aceste expresii au nevoie de calculul explicit al matricii hessiane

$\mathbf{H}$ pentru fiecare pas al algoritmului, pentru că hessiana este constantă doar în cazul unei funcţii de eroare polinomiale de gradul doi. Acest calcul este dificil de făcut, şi ar trebui să fie evitat. În cele ce urmează, vom deduce expresii pentru $\alpha_k$ şi $\beta_k$ care nu necesită calculul explicit al matricii hessiane, şi nici măcar nu presupun această matrice ca fiind pozitiv definită.

În primul rând, să luăm în considerare expresia pentru $\alpha_k$ , dată în (4.3.12). Datorită relaţiei iterative (4.3.10), putem să înlocuim calculul explicit al lui $\alpha_k$ cu o *căutare liniară inexactă* care minimizează $E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) = E(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k)$, i.e. o minimizare liniară de-a lungul direcţiei $\overset{\mathcal{R}}{\mathbf{p}}_k$, începând de la punctul $\overset{\mathcal{R}}{\mathbf{w}}_k$. Un exemplu de căutare liniară este metoda secţiunii de aur, care s-a demonstrat că are convergenţă liniară, de exemplu în [167, 47].

Acum, să ne îndreptăm atenţia către $\beta_k$. Din (4.3.5), avem că

$$\overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k = \overset{\mathcal{R}}{\mathbf{H}}_k(\overset{\mathcal{R}}{\mathbf{w}}_{k+1} - \overset{\mathcal{R}}{\mathbf{w}}_k) = \alpha_k \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{4.3.15}$$

unde $\overset{\mathcal{R}}{\mathbf{H}}_k := \mathbf{H}(\overset{\mathcal{R}}{\mathbf{w}}_k)$, şi deci expresia (4.3.14) devine:

$$\beta_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}. \tag{4.3.16}$$

Aceasta este cunoscută ca fiind expresia de actualizare *Hestenes-Stiefel*, a se vedea [112].

Pentru a prelucra mai departe această expresie, trebuie să luăm produsul scalar euclidian din $\mathbb{R}^{2^n N}$ al ecuaţiei (4.3.15) cu $\overset{\mathcal{R}}{\mathbf{p}}_k$ şi să ţinem cont de expresia (4.3.12) pentru $\alpha_k$ obţinând astfel

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle = 0. \tag{4.3.17}$$

Luând produsul scalar dintre (4.3.13) şi $\overset{\mathcal{R}}{\mathbf{g}}_{k+1}$, şi folosind ecuaţia de mai sus, rezultă că

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle = -\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle. \tag{4.3.18}$$

Acum, inserând relaţiile (4.3.17) şi (4.3.18) în expresia (4.3.16), obţinem expresia de actualizare *Polak-Ribiere* (a se vedea [214]):

$$\beta_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{g}}_k, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}. \tag{4.3.19}$$

În continuare, luăm produsul scalar euclidian $\mathbb{R}^{2^n N}$ al ecuaţiei (4.3.15) cu $\overset{\mathcal{R}}{\mathbf{p}}_i$ , $i < k$, pentru a obţine

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_i, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle = \alpha_k \langle \overset{\mathcal{R}}{\mathbf{p}}_i, \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = 0, \tag{4.3.20}$$

unde am luat în considerare faptul că $\overset{\mathcal{R}}{\mathbf{p}}_i$ şi $\overset{\mathcal{R}}{\mathbf{p}}_k$ sunt direcţii conjugate, pentru $i < k$. Ecuaţiile (4.3.20) şi (4.3.17) pot fi folosite pentru a demonstra prin inducţie că avem

$$\langle \overset{\mathcal{R}}{\mathbf{p}}_i, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle = 0, \forall i < k \leq N. \tag{4.3.21}$$

Din expresia pentru $\overset{\mathcal{R}}{\mathbf{p}}_{k+1}$ dată în ecuaţia (4.3.13), deducem că $\overset{\mathcal{R}}{\mathbf{p}}_k$ este dat de o combinaţie liniară a tuturor vectorilor gradient anteriori, şi anume putem scrie că

$$\overset{\mathcal{R}}{\mathbf{p}}_i = -\overset{\mathcal{R}}{\mathbf{g}}_i + \sum_{j=1}^{k-1} \gamma_j \overset{\mathcal{R}}{\mathbf{g}}_j. \tag{4.3.22}$$

Luând produsul scalar al ecuaţiei (4.3.22) cu $\overset{\mathcal{R}}{\mathbf{p}}_i$, $i < k$, şi folosind ecuaţia (4.3.21), obţinem următoarea relaţie:

$$\langle \overset{\mathcal{R}}{\mathbf{g}}_i, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle = \sum_{j=1}^{k-1} \gamma_j \langle \overset{\mathcal{R}}{\mathbf{g}}_j, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle, \forall i < k \leq N.$$

Pornind de la faptul că $\overset{\mathcal{R}}{\mathbf{p}}_1 = -\overset{\mathcal{R}}{\mathbf{g}}_1$ şi de la ecuaţia (4.3.21), obţinem prin inducţie că

$$\langle \overset{\mathcal{R}}{\mathbf{g}}_i, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle = 0, \forall i < k \leq N. \tag{4.3.23}$$

În particular, avem că $\langle \overset{\mathcal{R}}{\mathbf{g}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle = 0$, şi astfel expresia (4.3.19) devine:

$$\beta_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle}{\langle \overset{\mathcal{R}}{\mathbf{g}}_k, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}. \tag{4.3.24}$$

Această expresie este cunoscută ca formula de actualizare *Fletcher-Reeves,* a se vedea [225].

În fine, utilizând (4.3.23), şi anume faptul că $\langle \overset{\mathcal{R}}{\mathbf{g}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle = 0$, în ecuaţia (4.3.16), obţinem următoarea formulă de actualizare:

$$\beta_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle},$$

care este cunoscută sub numele de expresia de actualizare *Dai-Yuan*, a se vedea [73].

Există două variaţii ale formulelor de actualizare Hestenes-Stiefel şi Polak-Ribiere, a căror convergenţă s-a demonstrat în [103, 91] chiar dacă este folosită o căutare liniară inexactă pentru a determina valoarea lui $\alpha_k$. Dacă notăm cu $\beta_k^{HS}$, şi respectiv cu $\beta_k^{PR}$, valorile paşilor de actualizare ale celor doi algoritmi, expresiile pentru aceşti paşi care garantează convergenţa sunt

$$\beta_k^{HS+} = \max\{0, \beta_k^{HS}\},$$

şi respectiv

$$\beta_k^{PR+} = \max\{0, \beta_k^{PR}\}.$$

Dacă funcţia de eroare $E$ este pătratică (polinom de gradul doi), atunci metoda gradienţilor conjugaţi va găsi garantat minimul în cel mult $2^n N$ paşi. Dar, în general, funcţia de eroare poate fi departe de a fi pătratică, şi astfel algoritmul va avea nevoie de mai mult de $2^n N$ paşi pentru a se apropia de minim. De-a lungul acestor paşi, conjugarea direcţiilor de căutare tinde să se deterioreze, şi astfel este o practică comună aceea de a reporni algoritmul cu gradientul cu semn schimbat

$$\overset{\mathcal{R}}{\mathbf{p}}_k = -\overset{\mathcal{R}}{\mathbf{g}}_k,$$

după $2^n N$ paşi.

Un algoritm de repornire mai sofisticat, propus de Powell în [222], urmând o idee a lui Beale din [35], este de a reporni dacă există puţină ortogonalitate rămasă între gradientul curent şi gradientul anterior. Pentru a testa aceasta, verificăm faptul că are loc următoarea inegalitate:

$$|\langle \overset{\mathcal{R}}{\mathbf{g}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle| \geq 0.2 \langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} \rangle.$$

Regula de actualizare (4.3.13) pentru direcţia de căutare $\overset{\mathcal{R}}{\mathbf{p}}_k$ este de asemenea schimbată în

$$\overset{\mathcal{R}}{\mathbf{p}}_{k+1} = -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + \beta_k \overset{\mathcal{R}}{\mathbf{p}}_k + \gamma_k \overset{\mathcal{R}}{\mathbf{p}}_t,$$

unde coeficienţii $\beta_k$ şi $\gamma_k$ sunt calculaţi în felul următor:

$$\beta_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}$$

$$\gamma_k = \frac{\langle \overset{\mathcal{R}}{\mathbf{g}}_{k+1}, \overset{\mathcal{R}}{\mathbf{g}}_{t+1} - \overset{\mathcal{R}}{\mathbf{g}}_t \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_t, \overset{\mathcal{R}}{\mathbf{g}}_{t+1} - \overset{\mathcal{R}}{\mathbf{g}}_t \rangle},$$

iar $\overset{\mathcal{R}}{\mathbf{p}}_t$ este o direcţie descendentă de repornire arbitrar aleasă. Aceste expresii sunt deduse în acelaşi fel cu cele de mai sus. Metoda gradienţilor conjugaţi cu aceste caracteristici este numită metoda *Powell-Beale*.

## 4.4 Metoda gradienţilor conjugaţi scalaţi

După cum am văzut mai sus, în aplicaţiile din lumea reală, matricea hessiană poate fi departe de a fi pozitiv definită. Datorită acestui fapt, Møller a propus în [178], metoda gradienţilor conjugaţi scalaţi care foloseşte metoda *model trust region* cunoscută de la algoritmul Levenberg-Marquardt, combinată cu metoda gradienţilor conjugaţi. Pentru a asigura pozitiv definirea, trebuie să adăugăm la matricea hessiană o constantă pozitivă suficient de mare $\lambda_k$, înmulţită cu matricea identitate. Cu această modificare, formula pentru pasul de actualizare dată în (4.3.12), devine

$$\alpha_k = -\frac{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k \rangle + \lambda_k \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle}. \tag{4.4.1}$$

Notăm numitorul din (4.4.1) cu $\delta_k := \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k \rangle + \lambda_k \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle$. Pentru o hessiană pozitiv definită, avem că $\delta_k > 0$. Dar dacă $\delta_k \leq 0$, atunci va trebui să creştem valoarea lui $\delta_k$ pentru a o face pozitivă. Fie $\overline{\delta_k}$ noua valoare a lui $\delta_k$, şi, corespunzător, fie $\overline{\lambda_k}$ noua valoare a lui $\lambda_k$. Este clar că avem următoarea relaţie

$$\overline{\delta_k} = \delta_k + (\overline{\lambda_k} - \lambda_k) \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle, \tag{4.4.2}$$

şi, pentru ca $\overline{\delta_k} > 0$, trebuie să avem $\overline{\lambda_k} > \lambda_k - \delta_k / \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle$. Møller în [178] alege pentru $\overline{\lambda_k}$ valoarea

$$\overline{\lambda_k} = 2 \left( \lambda_k - \frac{\delta_k}{\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle} \right),$$

şi astfel expresia pentru noua valoare a lui $\delta_k$ dată în (4.4.2), devine

$$\overline{\delta_k} = -\delta_k + \lambda_k \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{p}}_k \rangle = -\langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k \rangle,$$

care este acum pozitivă şi va fi folosită în (4.4.1) pentru a calcula valoarea lui $\alpha_k$.

O altă problemă semnalată mai sus este aproximarea pătratică pentru funcţia de eroare $E$. Algoritmul gradienţilor conjugaţi scalaţi rezolvă această problemă prin considerarea unui parametru de comparaţie definit prin

$$\Delta_k = \frac{E(\overset{\mathcal{R}}{\mathbf{w}}_k) - E(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k)}{E(\overset{\mathcal{R}}{\mathbf{w}}_k) - E_Q(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k)}, \tag{4.4.3}$$

unde $E_Q(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k)$ reprezintă aproximarea pătratică locală a funcţiei de eroare într-o vecinătate a punctului $\overset{\mathcal{R}}{\mathbf{w}}_k$, dată prin

$$E_Q(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k) = E(\overset{\mathcal{R}}{\mathbf{w}}_k) + \alpha_k \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle + \frac{1}{2}\alpha_k^2 \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k \rangle. \tag{4.4.4}$$

Putem vedea cu uşurinţă că $\Delta_k$ măsoară cât de bună este această aproximare pătratică. Introducând relaţia (4.4.4) în relaţia (4.4.3), şi ţinând cont de expresia (4.3.12) pentru $\alpha_k$, avem că

$$\Delta_k = \frac{2(E(\overset{\mathcal{R}}{\mathbf{w}}_k) - E(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k))}{\alpha_k \langle \overset{\mathcal{R}}{\mathbf{p}}_k, \overset{\mathcal{R}}{\mathbf{g}}_k \rangle}.$$

Valoarea lui $\lambda_k$ este apoi actualizată în felul următor:

$$\lambda_{k+1} = \begin{cases} \lambda_k/2, & \text{dacă } \Delta_k > 0.75 \\ 4\lambda_k, & \text{dacă } \Delta_k < 0.25 \\ \lambda_k, & \text{altfel} \end{cases}, \tag{4.4.5}$$

pentru a asigura o mai bună aproximare pătratică.

Prin urmare, există două stadii ale actualizării lui $\lambda_k$: una pentru a ne asigura că $\delta_k > 0$, şi una corespunzătoare validităţii aproximării pătratice. Cele două stadii sunt aplicate succesiv după fiecare actualizare a ponderilor.

## 4.5 Metoda Newton

Să presupunem că avem o reţea neuronală cu valori complexe, a cărei funcţie de eroare este $E : \mathbb{R}^{2^n N} \to \mathbb{R}$. Dacă notăm $\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}) := \nabla E(\overset{\mathcal{R}}{\mathbf{w}})$, gradientul funcţiei $E$, şi cu $\mathbf{H}(\overset{\mathcal{R}}{\mathbf{w}}) := \nabla^2 E(\overset{\mathcal{R}}{\mathbf{w}})$, hessiana funcţiei $E$, atunci, presupunând că $\overset{\mathcal{R}}{\mathbf{w}} = (([\mathbf{w}]_I^T)_{I \in \mathcal{I}})^T \in \mathbb{R}^{2^n N}$ este vectorul $2^n N$-dimensional al componentelor tuturor ponderilor şi bias-urilor reţelei, putem găsi minimul funcţiei de eroare folosind *metoda lui Newton* care presupune calculul iterativ al vectorului $\overset{\mathcal{R}}{\mathbf{w}}^*$ care minimizează funcţia $E$ folosind următoarea regulă de actualizare

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - \mathbf{H}^{-1}(\overset{\mathcal{R}}{\mathbf{w}}_k)\mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_k),$$

sau, echivalent, dacă notăm $\overset{\mathcal{R}}{\mathbf{H}}_k := \mathbf{H}(\overset{\mathcal{R}}{\mathbf{w}}_k)$ şi respectiv $\overset{\mathcal{R}}{\mathbf{g}}_k := \mathbf{g}(\overset{\mathcal{R}}{\mathbf{w}}_k)$,

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k^{-1} \overset{\mathcal{R}}{\mathbf{g}}_k.$$

Această regulă de actualizare face o presupunere esenţială asupra matricii hessiane, şi anume aceea că ea trebuie să fie pozitiv definită. Dacă această condiţie nu este îndeplinită, atunci se foloseşte formula de actualizare modificată

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - (\overset{\mathcal{R}}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2^n N})^{-1} \overset{\mathcal{R}}{\mathbf{g}}_k,$$

unde $\lambda_k$ este o constantă care se alege astfel încât matricea $\overset{\mathcal{R}}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2^n N}$ să fie pozitiv definită.

Deoarece am dat în cadrul metodei gradient un algoritm de calculare a gradientului funcţiei de eroare a unei reţele neuronale cu valori Clifford, pentru a putea implementa metoda Newton, mai trebuie să găsim o modalitate de calcul a hessianei funcţiei de eroare. Din acest motiv, restul acestei secţiuni este dedicat deducerii complete a unei metode de calcul a hessianei funcţiei de eroare a unei reţele neuronale cu valori Clifford.

Chiar dacă este complicată din punct de vedere computaţional, metoda Newton este cea mai exactă dintre metodele de optimizare de ordinul doi folosite pentru găsirea minimului funcţiei de eroare, motiv pentru care am decis s-o includem în studiul nostru despre algoritmii de învăţare pentru reţelele neuronale cu valori Clifford. În plus, hessiana are şi alte aplicaţii în domeniul reţelelor neuronale, cum ar fi spre exemplu diferiţi algoritmi de antrenare a reţelelor, în reantrenarea rapidă a reţelei după o mică modificare în datele de antrenare [44], pentru identificarea celor mai puţin semnificative ponderi ca o bază pentru tehnici de simplificare a reţelelor [158], pentru îmbunătăţirea vitezei algoritmilor de antrenare [36], pentru estimarea bayesiană a parametrilor de regularizare [168], pentru asignarea de probabilităţi diferitelor soluţii ale reţelei, şi multe altele. Prezentarea noastră urmăreşte în principal primul articol în care a fost calculată exact hessiana pentru reţelele neuronale cu valori reale, şi anume [45], care a fost apoi reluat în cartea clasică [46].

### 4.5.1 Calculul hessianei

Să considerăm o reţea neuronală cu valori complexe care are $L$ straturi notate $\{1, \ldots, L\}$, unde $1$ este stratul de intrare, $L$ este stratul de ieşire, iar restul sunt straturi ascunse. Pentru această reţea, avem următoarea funcţie de eroare:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{c} ||y_i^L - t_i||^2,$$

unde $\mathbf{w}$ reprezintă vectorul tuturor ponderilor şi bias-urilor reţelei, $\mathbf{y}^L = (y_i^L)_{1 \leq i \leq c}$ reprezintă ieşirile reţelei, şi $\mathbf{t} = (t_i)_{1 \leq i \leq c}$ reprezintă ieşirile dorite ale reţelei. Am notat cu $E : (C\ell_{p,q})^N \to \mathbb{R}$ funcţia de eroare care depinde de cele $N$ ponderi şi bias-uri ale reţelei.

Vrem să calculăm hessiana funcţiei de eroare $E$. Pentru hessiană, avem nevoie de componentele vectorului gradient. Dacă notăm cu $w_{jk}^l$ ponderea dintre neuronul $j$ din stratul $l$ şi neuronul $k$ din stratul $l-1$, putem scrie componenta vectorului gradient al funcţiei de eroare $E$ în raport cu ponderea $w_{jk}^l$ ca fiind

$$\sum_{A \in \mathcal{I}} \frac{\partial E}{\partial [w_{jk}^l]_A} e_A = \delta_j^l \otimes_{p,q} (x_k^{l-1})^*, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left(\sum_r (w_{rj}^{l+1})^* \otimes_{p,q} \delta_r^{l+1}\right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l \leq L-1 \\ (y_j^l - t_j) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l = L \end{cases}. \tag{4.5.1}$$

şi am notat

$$s_j^l := \sum_k w_{jk}^l \otimes_{p,q} x_k^{l-1}, \tag{4.5.2}$$

$$y_j^l := G^l(s_j^l), \tag{4.5.3}$$

unde $G^l : C\ell_{p,q} \to C\ell_{p,q}$ reprezintă funcţia de activare a stratului $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1 \le k \le d}$ sunt intrările reţelei şi $x_k^l = y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$.

Elementele hessianei sunt derivate parţiale de ordinul doi ale funcţiei de eroare $E$ în raport cu oricare două ponderi sau bias-uri arbitrare ale reţelei. Să notăm cu $w_{mn}^{l_1}$ şi $w_{jk}^{l_2}$ doi astfel de parametri ai reţelei neuronale cu valori Clifford, şi să presupunem de asemenea că $2 \le l_1 \le l_2 \le L$. Avem de calculat $2^{2n}$ derivate parţiale de forma

$$\frac{\partial^2 E}{\partial [w_{mn}^{l_1}]_A \partial [w_{jk}^{l_2}]_B}, \forall A, B \in \mathcal{I}.$$

Începem din nou cu regula înlănţuirii

$$\begin{aligned}
\frac{\partial^2 E}{\partial [w_{mn}^{l_1}]_A \partial [w_{jk}^{l_2}]_B} &= \sum_{C \in \mathcal{I}} \frac{\partial [s_m^{l_1}]_C}{\partial [w_{mn}^{l_1}]_A} \frac{\partial^2 E}{\partial [s_m^{l_1}]_C \partial [w_{jk}^{l_2}]_B} \\
&= \sum_{\substack{C \in \mathcal{I} \\ \kappa_{A,D} e_A e_D = e_C}} \kappa_{A,D} [x_n^{l_1 - 1}]_D \frac{\partial^2 E}{\partial [s_m^{l_1}]_C \partial [w_{jk}^{l_2}]_B}.
\end{aligned}$$

Acum, folosind (4.5.1), putem scrie că

$$\delta_j^{l_2} \otimes_{p,q} (x_k^{l_2 - 1})^* = \sum_{E,F \in \mathcal{I}} [\delta_j^{l_2}]_E [x_k^{l_2 - 1}]_F e_E e_F,$$

şi deci

$$\begin{aligned}
\frac{\partial^2 E}{\partial [s_m^{l_1}]_C \partial [w_{jk}^{l_2}]_B} &= \frac{\partial [\delta_j^{l_2} \otimes_{p,q} (x_k^{l_2 - 1})^*]_B}{\partial [s_m^{l_1}]_C} \\
&= \frac{\partial [\sum_{E,F \in \mathcal{I}} [\delta_j^{l_2}]_E [x_k^{l_2 - 1}]_F \kappa_{E,F} e_E e_F]_B}{\partial [s_m^{l_1}]_C} \\
&= \sum_{\substack{E,F \in \mathcal{I} \\ \kappa_{E,F} e_E e_F = e_B}} \frac{\partial [\delta_j^{l_2}]_E}{\partial [s_m^{l_1}]_C} [x_k^{l_2 - 1}]_F + [\delta_j^{l_2}]_E \frac{\partial [x_k^{l_2 - 1}]_F}{\partial [s_m^{l_1}]_C}
\end{aligned} \tag{4.5.4}$$

şi, din regula înlănţuirii, că

$$\begin{aligned}
\frac{\partial [x_k^{l_2 - 1}]_F}{\partial [s_m^{l_1}]_C} &= \sum_{G \in \mathcal{I}} \frac{\partial [s_k^{l_2 - 1}]_G}{\partial [s_m^{l_1}]_C} \frac{\partial [x_k^{l_2 - 1}]_F}{\partial [s_k^{l_2 - 1}]_G} \\
&= \sum_{G \in \mathcal{I}} \frac{\partial [s_k^{l_2 - 1}]_G}{\partial [s_m^{l_1}]_C} \frac{\partial [G^{l_2 - 1}(s_k^{l_2 - 1})]_F}{\partial [s_k^{l_2 - 1}]_G} \\
&= \frac{\partial [s_k^{l_2 - 1}]_F}{\partial [s_m^{l_1}]_C} \frac{\partial [G^{l_2 - 1}(s_k^{l_2 - 1})]_F}{\partial [s_k^{l_2 - 1}]_F} \\
&= [h_{km}^{l_2 - 1, l_1}]_{FC} \frac{\partial [G^{l_2 - 1}(s_k^{l_2 - 1})]_F}{\partial [s_k^{l_2 - 1}]_F},
\end{aligned}$$

unde am notat $[h_{km}^{l_2-1,l_1}]_{FC} := \frac{\partial [s_k^{l_2-1}]_F}{\partial [s_m^{l_1}]_C}$ . Dacă notăm în continuare $[b_{jm}^{l_2,l_1}]_{EC} := \frac{\partial [\delta_j^{l_2}]_E}{\partial [s_m^{l_1}]_C}$, ecuaţia (4.5.4) devine

$$\frac{\partial^2 E}{\partial [s_m^{l_1}]_C \partial [w_{jk}^{l_2}]_B} = \sum_{\substack{E,F\in\mathcal{I} \\ \kappa_{E,F} e_E e_F = e_B}} \left( [b_{jm}^{l_2,l_1}]_{EC} [x_k^{l_2-1}]_F + [h_{km}^{l_2-1,l_1}]_{FC} \frac{\partial [G^{l_2-1}(s_k^{l_2-1})]_F}{\partial [s_k^{l_2-1}]_F} [\delta_j^{l_2}]_E \right).$$

Acum trebuie să găsim o metodă de a calcula $[h_{km}^{l_2-1,l_1}]_{FC}$ şi $[b_{jm}^{l_2,l_1}]_{EC}$. Începem cu prima. Dacă $l_2 = l_1$, avem că $h_{km}^{l_2-1,l_1} = 0$, deoarece $s_k^{l_2-1}$ nu depinde de $s_m^{l_1}$. Dacă $l_2 - 1 = l_1$, $h_{km}^{l_2-1,l_1} = \delta_{km}$, deoarece $s_k^{l_2-1}$ şi $s_m^{l_1}$ sunt acum pe acelaşi strat, şi $\delta_{km}$ este simbolul delta al lui Kronecker: $\delta_{km} = \mathbf{I}_{2^n}$ dacă $k = m$ şi $\delta_{km} = \mathbf{O}_{2^n}$ dacă $k \neq m$. Pentru $l_2 \geq l_1 + 2$ (în care caz trebuie să avem $L \geq 4$), putem să aplicăm din nou regula înlănţuirii

$$\begin{aligned} \frac{\partial [s_k^{l_2-1}]_F}{\partial [s_m^{l_1}]_C} &= \sum_r \left( \sum_{G\in\mathcal{I}} \frac{\partial [s_r^{l_2-2}]_G}{\partial [s_m^{l_1}]_C} \frac{\partial [s_k^{l_2-1}]_F}{\partial [s_r^{l_2-2}]_G} \right) \\ &= \sum_r \sum_{\substack{G\in\mathcal{I} \\ \kappa_{G,I} e_G e_I = e_F}} \left( [h_{rm}^{l_2-2,l_1}]_{GC} \kappa_{G,I} [w_{kr}^{l_2-1}]_I \frac{\partial [G^{l_2-2}(s_r^{l_2-2})]_G}{\partial [s_r^{l_2-2}]_G} \right), \end{aligned}$$

unde am folosit faptul că

$$\begin{aligned} \frac{\partial [s_k^{l_2-1}]_F}{\partial [s_r^{l_2-2}]_G} &= \sum_{H\in\mathcal{I}} \frac{\partial [s_k^{l_2-1}]_F}{\partial [y_r^{l_2-2}]_H} \frac{\partial [y_r^{l_2-2}]_H}{\partial [s_r^{l_2-2}]_G} \\ &= \kappa_{G,I} [w_{kr}^{l_2-1}]_I \frac{\partial [G^{l_2-2}(s_r^{l_2-2})]_G}{\partial [s_r^{l_2-2}]_G}, \kappa_{G,I} e_G e_I = e_F, \end{aligned}$$

iar suma se ia după toţi neuronii $r$ din stratul $l_2 - 2$ care trimit conexiuni către neuronul $k$ din stratul $l_2 - 1$.

În concluzie, am obţinut următoarea relaţie de recurenţă pentru calculul lui $[h_{km}^{l_2-1,l_1}]_{FC}$:

$$[h_{km}^{l_2-1,l_1}]_{FC} = \sum_r \sum_{\substack{G\in\mathcal{I} \\ \kappa_{G,I} e_G e_I = e_F}} \left( [h_{rm}^{l_2-2,l_1}]_{GC} \kappa_{G,I} [w_{kr}^{l_2-1}]_I \frac{\partial [G^{l_2-2}(s_r^{l_2-2})]_G}{\partial [s_r^{l_2-2}]_G} \right), \quad (4.5.5)$$

Acum, să ne îndreptăm atenţia către $[b_{jm}^{l_2,l_1}]_{EC}$. Tratăm mai întâi cazul în care $l_2 \leq L - 1$. Din formula pentru $\delta_j^l$, dată în ecuaţia (4.5.1), avem că

$$\left( \sum_r (w_{rj}^{l_2+1})^* \otimes_{p,q} \delta_r^{l_2+1} \right) \odot \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}} = \sum_r \sum_{\substack{H,I\in\mathcal{I} \\ \kappa_{H,I} e_H e_I = e_J}} \kappa_{H,I} [w_{rj}^{l_2+1}]_H [\delta_r^{l_2+1}]_I \frac{\partial [G^{l_2}(s_j^{l_2})]_J}{\partial [s_j^{l_2}]_J} e_H e_I,$$

şi deci

$$
\begin{aligned}
\frac{\partial[\delta_j^{l_2}]_E}{\partial[s_m^{l_1}]_C} &= \frac{\left[\sum_r \sum_{\substack{H,I\in\mathcal{I}\\ \kappa_{H,I}e_He_I=e_J}} \kappa_{H,I}[w_{rj}^{l_2+1}]_H[\delta_r^{l_2+1}]_I \frac{\partial[G^{l_2}(s_j^{l_2})]_J}{\partial[s_j^{l_2}]_J} e_He_I\right]_E}{\partial[s_m^{l_1}]_C} \\
&= \frac{\partial\left(\sum_r \sum_{\substack{H,I\in\mathcal{I}\\ \kappa_{H,I}e_He_I=e_E}} \kappa_{H,I}[w_{rj}^{l_2+1}]_H[\delta_r^{l_2+1}]_I \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E}\right)}{\partial[s_m^{l_1}]_C} \\
&= \sum_r \sum_{\substack{H,I\in\mathcal{I}\\ \kappa_{H,I}e_He_I=e_E}} \left(\frac{\partial[w_{rj}^{l_2+1}]_H}{\partial[s_m^{l_1}]_C}[\delta_r^{l_2+1}]_I \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E}\right. \\
&+ \left. [w_{rj}^{l_2+1}]_H \frac{\partial[\delta_r^{l_2+1}]_I}{\partial[s_m^{l_1}]_C} \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E} + [w_{rj}^{l_2+1}]_H[\delta_r^{l_2+1}]_I \frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C\partial[s_j^{l_2}]_E}\right) \\
&= \sum_r \sum_{\substack{H,I\in\mathcal{I}\\ \kappa_{H,I}e_He_I=e_E}} \left([w_{rj}^{l_2+1}]_H[b_{rm}^{l_2+1,l_1}]_{IC} \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E}\right. \\
&+ \left. [w_{rj}^{l_2+1}]_H[\delta_r^{l_2+1}]_I \frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C\partial[s_j^{l_2}]_E}\right), \qquad (4.5.6)
\end{aligned}
$$

unde suma se ia după toţi neuronii $r$ din stratul $l_2+1$ către care neuronul $j$ din stratul $l_2$ trimite conexiuni, şi $\frac{\partial[w_{rj}^{l_2+1}]_H}{\partial[s_m^{l_1}]_C} = 0$. Expresiile pentru derivatele parţiale ale funcţiilor de activare rezultă uşor din aplicarea regulii înlănţuirii:

$$
\begin{aligned}
\frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C\partial[s_j^{l_2}]_E} &= \sum_{H\in\mathcal{I}} \frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_H\partial[s_j^{l_2}]_E} \frac{\partial[s_j^{l_2}]_H}{\partial[s_m^{l_1}]_C} \\
&= \frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E^2}[h_{jm}^{l_2,l_1}]_{EC}
\end{aligned}
$$

În continuare, avem de tratat cazul în care $l_2 = L$. Atunci, din expresia (4.5.1), rezultă că

$$
\begin{aligned}
\frac{\partial[\delta_j^{l_2}]_E}{\partial[s_m^{l_1}]_C} &= \frac{\partial\left[(y_j^{l_2} - t_j) \odot \frac{\partial G^{l_2}(s_j^{l_2})}{\partial s_j^{l_2}}\right]_E}{\partial[s_m^{l_1}]_C} \\
&= \frac{\partial[y_j^{l_2} - t_j]_E}{\partial[s_m^{l_1}]_C} \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E} + [y_j^{l_2} - t_j]_E \frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C\partial[s_j^{l_2}]_E}. \\
&= \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C} \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E} + [y_j^{l_2} - t_j]_E \frac{\partial^2[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C\partial[s_j^{l_2}]_E} \qquad (4.5.7)
\end{aligned}
$$

Expresiile pentru calculul derivatelor parţiale de ordinul doi ale funcţiilor de activare sunt la fel ca cele de mai sus. Prin urmare, trebuie doar să mai dăm expresii pentru

calculul derivatelor parţiale ale funcţiilor de activare folosite în expresiile de mai sus. Acestea sunt deduse prin aplicarea regulii înlănţuirii:

$$\begin{aligned}\frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_m^{l_1}]_C} &= \sum_{H\in\mathcal{I}} \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_H}\frac{\partial[s_j^{l_2}]_H}{\partial[s_m^{l_1}]_C} \\ &= \frac{\partial[G^{l_2}(s_j^{l_2})]_E}{\partial[s_j^{l_2}]_E}[h_{jm}^{l_2,l_1}]_{EC}.\end{aligned}$$

Se poate observa că expresia pentru $h_{km}^{l_2-1,l_1}$ este calculată prin propagare înainte, iar expresia pentru $b_{jm}^{l_2,l_1}$ este calculată prin propagare înapoi. În fine, punând totul împreună, obţinem următoarea formulă pentru o componentă arbitrară a matricii hessiane a funcţiei de eroare $E$:

$$\begin{aligned}\frac{\partial^2 E}{\partial[w_{mn}^{l_1}]_A\partial[w_{jk}^{l_2}]_B} &= \sum_{\substack{C,E,F\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_C\\ \kappa_{E,F}e_Ee_F=e_B}} \Big(\kappa_{A,D}[x_n^{l_1-1}]_D[b_{jm}^{l_2,l_1}]_{EC}[x_k^{l_2-1}]_F \\ &+ \kappa_{A,D}[x_n^{l_1-1}]_D[h_{km}^{l_2-1,l_1}]_{FC}\frac{\partial[G^{l_2-1}(s_k^{l_2-1})]_F}{\partial[s_k^{l_2-1}]_F}[\delta_j^{l_2}]_E\Big). \quad (4.5.8)\end{aligned}$$

### 4.5.2 Calculul hessianei pentru o reţea neuronală cu trei straturi

În această subsecţiune, vom considera cazul special al unei reţele cu un singur strat ascuns, pentru care $L = 3$. Vom nota prin $i, i'$ neuronii din stratul de intrare, prin $j, j'$ neuronii din stratul ascuns, şi prin $k, k'$ neuronii stratul de ieşire. Cu aceste convenţii, ecuaţiile (4.5.2) şi (4.5.3) devin

$$s_j^2 := \sum_i w_{ji}^2 \otimes_{p,q} x_i^1,$$

$$x_j^2 := G^2(s_j^2),$$

$$s_k^3 := \sum_j w_{kj}^3 \otimes_{p,q} x_j^2,$$

$$x_k^3 := G^3(s_k^3).$$

Expresia (4.5.8) pentru derivata parţială de ordinul doi a funcţiei de eroare, în cazul $l_1 = l_2 = 3$, este

$$\frac{\partial^2 E}{\partial[w_{k'j'}^3]_A\partial[w_{kj}^3]_B} = \sum_{\substack{C,E,F\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_C\\ \kappa_{E,F}e_Ee_F=e_B}} \kappa_{A,D}[x_{j'}^2]_D\frac{\partial[\delta_k^3]_E}{\partial[s_{k'}^3]_C}[x_j^2]_F,$$

deoarece $h_{jk'}^{2,3} = 0$. Acum, mai trebuie doar să calculăm $\frac{\partial[\delta_k^3]_E}{\partial[s_{k'}^3]_C}$ folosind ecuaţia (4.5.7):

$$\begin{aligned}\frac{\partial[\delta_k^3]_E}{\partial[s_{k'}^3]_C} &= \frac{\partial[G^3(s_k^3)]_E}{\partial[s_{k'}^3]_C}\frac{\partial[G^3(s_k^3)]_E}{\partial[s_k^3]_E} + [y_k^3 - t_k]_E \frac{\partial^2[G^3(s_k^3)]_E}{\partial[s_{k'}^3]_C \partial[s_k^3]_E} \\ &= \frac{\partial[G^3(s_k^3)]_E}{\partial[s_k^3]_E}[h_{kk'}^{3,3}]_{EC}\frac{\partial[G^3(s_k^3)]_E}{\partial[s_k^3]_E} + [y_k^3 - t_k]_E \frac{\partial^2[G^3(s_k^3)]_E}{\partial[s_k^3]_E^2}[h_{kk'}^{3,3}]_{EC} \\ &= [\delta_{k'k}]_{EC}\left[\left(\frac{\partial[G^3(s_k^3)]_E}{\partial[s_k^3]_E}\right)^2 + [y_k^3 - t_k]_E \frac{\partial^2[G^3(s_k^3)]_E}{\partial[s_k^3]_E^2}\right],\end{aligned}$$

unde am folosit expresia (4.5.1) pentru $\delta_k^3$, şi $\delta_{k'k}$ este simbolul delta al lui Kronecker definit mai sus. Expresia finală pentru derivata parţială de ordinul doi este

$$\begin{aligned}\frac{\partial^2 E}{\partial[w_{k'j'}^3]_A \partial[w_{kj}^3]_B} &= \sum_{\substack{C,E,F\in\mathcal{I} \\ \kappa_{A,D}e_A e_D = e_C \\ \kappa_{E,F}e_E e_F = e_B}} \kappa_{A,D}[x_{j'}^2]_D [x_j^2]_F [\delta_{k'k}]_{EC}\left[\left(\frac{\partial[G^3(s_k^3)]_E}{\partial[s_k^3]_E}\right)^2\right. \\ &+ \left.[y_k^3 - t_k]_E \frac{\partial^2[G^3(s_k^3)]_E}{\partial[s_k^3]_E^2}\right].\end{aligned}$$

Acum, considerăm cazul în care $l_1 = l_2 = 2$. Avem de asemenea $h_{ij'}^{1,2} = g_{ij'}^{1,2} = 0$, deci expresia (4.5.8) devine

$$\frac{\partial^2 E}{\partial[w_{j'i'}^2]_A \partial[w_{ji}^2]_B} = \sum_{\substack{C,E,F\in\mathcal{I} \\ \kappa_{A,D}e_A e_D = e_C \\ \kappa_{E,F}e_E e_F = e_B}} \kappa_{A,D}[x_{i'}^1]_D \frac{\partial[\delta_j^2]_E}{\partial[s_{j'}^2]_C}[x_i^1]_F.$$

Trebuie să calculăm $\frac{\partial[\delta_j^2]_E}{\partial[s_{j'}^2]_C}$, şi pentru aceasta vom folosi formula (4.5.6):

$$\frac{\partial[\delta_j^2]_E}{\partial[s_{j'}^2]_C} = \sum_r \sum_{\substack{H,I\in\mathcal{I} \\ \kappa_{H,I}e_H e_I = e_E}} \left([w_{rj}^3]_H [b_{rj'}^{3,2}]_{IC}\frac{\partial[G^2(s_j^2)]_E}{\partial[s_j^2]_E} + [w_{rj}^3]_H [\delta_r^3]_I \frac{\partial^2[G^2(s_j^2)]_E}{\partial[s_{j'}^2]_C \partial[s_j^2]_E}\right).$$

Relaţia de mai sus arată că avem de asemenea nevoie de o expresie pentru $[b_{rj'}^{3,2}]_{IC}$. Folosind din nou ecuaţia (4.5.7), avem că

$$\frac{\partial[\delta_r^3]_I}{\partial[s_{j'}^2]_C} = \frac{\partial[G^3(s_r^3)]_I}{\partial[s_{j'}^2]_C}\frac{\partial[G^3(s_r^3)]_I}{\partial[s_r^3]_I} + [y_r^3 - t_r]_I \frac{\partial^2[G^3(s_r^3)]_I}{\partial[s_{j'}^2]_C \partial[s_r^3]_I},$$

unde avem următoarele expresii:

$$\frac{\partial[G^3(s_r^3)]_I}{\partial[s_{j'}^2]_C} = \frac{\partial[G^3(s_r^3)]_I}{\partial[s_r^3]_I}[h_{rj'}^{3,2}]_{IC}$$

şi

$$\frac{\partial^2[G^3(s_r^3)]_I}{\partial[s_{j'}^2]_C \partial[s_r^3]_I} = \frac{\partial^2[G^3(s_r^3)]_I}{\partial[s_r^3]_I^2}[h_{rj'}^{3,2}]_{IC}.$$

În fine, expresia pentru $[h^{3,2}_{rj'}]_{IC}$ poate fi dedusă folosind ecuaţia (4.5.5):

$$\begin{aligned}[h^{3,2}_{rj'}]_{IC} &= \sum_j \sum_{\substack{G\in\mathcal{I}\\ \kappa_{G,J}e_Ge_J=e_I}} \left([h^{2,2}_{jj'}]_{GC}\kappa_{G,J}[w^2_{rj}]_J \frac{\partial[G^2(s^2_j)]_G}{\partial[s^2_j]_G}\right)\\ &= \sum_j \sum_{\substack{G\in\mathcal{I}\\ \kappa_{G,J}e_Ge_J=e_I}} \left([\delta_{jj'}]_{GC}\kappa_{G,J}[w^2_{rj}]_J \frac{\partial[G^2(s^2_j)]_G}{\partial[s^2_j]_G}\right),\end{aligned}$$

unde am folosit faptul că $h^{2,2}_{jj'} = \delta_{jj'}$.

Ultimul caz este acela în care $l_1 = 2$, $l_2 = 3$. În această situaţie, avem că $h^{2,2}_{jj'} = \delta_{jj'}$, deci expresia (4.5.8) devine

$$\begin{aligned}\frac{\partial^2 E}{\partial[w^2_{j'i'}]_A\partial[w^3_{kj}]_B} &= \sum_{\substack{C,E,F\in\mathcal{I}\\ \kappa_{A,D}e_Ae_D=e_C\\ \kappa_{E,F}e_Ee_F=e_B}} \Big(\kappa_{A,D}[x^1_{i'}]_D[b^{3,2}_{kj'}]_{EC}[x^1_j]_F\\ &+ \kappa_{A,D}[x^1_{i'}]_D[\delta_{jj'}]_{FC}\frac{\partial[G^2(s^2_j)]_F}{\partial[s^2_j]_F}[\delta^3_k]_E\Big),\end{aligned}$$

în care expresia pentru $[b^{3,2}_{kj'}]_{EC}$ a fost dedusă mai sus.

### 4.5.3 Produsul hessianei cu un vector

În unele aplicaţii, cantitatea de interes nu este matricea hessiană $\mathbf{H}$ însăşi, ci produsul ei cu un vector $\mathbf{v}$. Calculul hessianei este o operaţie de ordinul lui $N^2$, unde $N$ este numărul de ponderi şi bias-uri ale reţelei, pe când vectorul $\mathbf{Hv}$ are $N$ elemente, şi calculul lui este de ordinul lui $N$. Din acest motiv, este preferabil să găsim o procedură de calcul direct a produsului $\mathbf{Hv}$, fără a trece prin pasul intermediar al calculului hessianei. Prezentarea urmăreşte pe cea din [209], fiind o adaptare a aceleaşi metode din cazul real în cazul Clifford.

Fiind dată o reţea neuronală cu funcţia de eroare $E$, din dezvoltarea în serie Taylor de ordinul unu a gradientului $\nabla E$ al funcţiei de eroare, avem că

$$\nabla E(\mathbf{w}+\Delta\mathbf{w}) = \nabla E(\mathbf{w}) + \mathbf{H}(\mathbf{w})\Delta\mathbf{w} + O(||\Delta\mathbf{w}||^2),$$

unde $O(||\Delta\mathbf{w}||^2)$ desemnează termeni de ordinul doi care depind de $||\Delta\mathbf{w}||$, iar $\mathbf{H}(\mathbf{w}) := \nabla^2 E(\mathbf{w})$ reprezintă hessiana funcţiei de eroare. Dacă scriem relaţia de mai sus pentru $\Delta\mathbf{w} = r\mathbf{v}$, unde $r$ este un număr real mic, şi $\mathbf{v}$ este un vector, avem că

$$\mathbf{H}(\mathbf{w})(r\mathbf{v}) = r\mathbf{H}(\mathbf{w})\mathbf{v} = \nabla E(\mathbf{w}+\Delta\mathbf{w}) - \nabla E(\mathbf{w}) + O(r^2),$$

sau, echivalent, că

$$\mathbf{H}(\mathbf{w})\mathbf{v} = \frac{\nabla E(\mathbf{w}+\Delta\mathbf{w}) - \nabla E(\mathbf{w})}{r} + O(r).$$

Trecând la limită pentru $r\to 0$, avem că

$$\mathbf{H}(\mathbf{w})\mathbf{v} = \lim_{r\to 0}\frac{\nabla E(\mathbf{w}+\Delta\mathbf{w}) - \nabla E(\mathbf{w})}{r} = \frac{\partial}{\partial r}\nabla E(\mathbf{w}+r\mathbf{v})\Big|_{r=0}.$$

Acum, pentru a calcula exact produsul $\mathbf{H}(\mathbf{w})\mathbf{v}$, vom folosi ceea ce poartă denumirea de tehnica $\mathcal{R}$. Aceasta este o transformare care converteşte un algoritm care calculează gradientul unui sistem într-un algoritm care calculează pe $\mathbf{H}(\mathbf{w})\mathbf{v}$ . Cheia acestei transformări este să definim operatorul

$$\mathcal{R}\{f(\mathbf{w})\} = \frac{\partial}{\partial r} f(\mathbf{w} + r\mathbf{v})\bigg|_{r=0},$$

astfel încât avem

$$\mathbf{H}(\mathbf{w})\mathbf{v} = \mathcal{R}\{\nabla E(\mathbf{w})\}.$$

Acum, putem să luăm toate ecuaţiile unui algoritm care calculează gradientul, de exemplu metoda gradient, şi să aplicăm operatorul $\mathcal{R}$ pentru fiecare ecuaţie. Deoarece $\mathcal{R}$ este un operator diferenţial, el satisface regulile uzuale ale operatorilor diferenţiali, cum ar fi, spre exemplu, următoarele:

$$\begin{aligned}
\mathcal{R}\{cf(\mathbf{w})\} &= c\mathcal{R}\{f(\mathbf{w})\}.\\
\mathcal{R}\{f(\mathbf{w}) + g(\mathbf{w})\} &= \mathcal{R}\{f(\mathbf{w})\} + \mathcal{R}\{g(\mathbf{w})\},\\
\mathcal{R}\{f(\mathbf{w})g(\mathbf{w})\} &= \mathcal{R}\{f(\mathbf{w})\}g(\mathbf{w}) + f(\mathbf{w})\mathcal{R}\{g(\mathbf{w})\},\\
\mathcal{R}\{f(g(\mathbf{w}))\} &= f'(g(\mathbf{w}))\mathcal{R}\{g(\mathbf{w})\},\\
\mathcal{R}\left\{\frac{df(\mathbf{w})}{dt}\right\} &= \frac{d\mathcal{R}\{f(\mathbf{w})\}}{dt}.
\end{aligned}$$

Se mai poate observa uşor că avem

$$\mathcal{R}\{\mathbf{w}\} = \mathbf{v}.$$

Folosind această tehnică, metoda gradient poate fi transformată într-o metodă de calcul al lui $\mathcal{R}\{\nabla E(\mathbf{w})\}$, care este tocmai produsul matricii hessiane cu un vector, $\mathbf{H}(\mathbf{w})\mathbf{v}$, pe care dorim să-l calculăm.

Aplicăm acum operatorul $\mathcal{R}$ pentru metoda gradient. Din

$$s_j^l = \sum_k w_{jk}^l \otimes_{p,q} x_k^{l-1},$$

avem

$$s_j^l = \sum_k w_{jk}^l \otimes_{p,q} x_k^{l-1} = \sum_{C,D\in\mathcal{I}} [w_{jk}^l]_C [x_k^{l-1}]_D e_C e_D,$$

şi

$$[s_j^l]_B = \sum_k \sum_{\substack{C,D\in\mathcal{I}\\ \kappa_{C,D} e_C e_D = e_B}} \kappa_{C,D} [w_{jk}^l]_C [x_k^{l-1}]_D,$$

de unde

$$\begin{aligned}
\mathcal{R}\{[s_j^l]_B\} &= \sum_k \sum_{\substack{C,D\in\mathcal{I}\\ \kappa_{C,D} e_C e_D = e_B}} \left(\kappa_{C,D}\mathcal{R}\{[w_{jk}^l]_C\}[x_k^{l-1}]_D + \kappa_{C,D}[w_{jk}^l]_C\mathcal{R}\{[x_k^{l-1}]_D\}\right)\\
&= \sum_k \sum_{\substack{C,D\in\mathcal{I}\\ \kappa_{C,D} e_C e_D = e_B}} \left(\kappa_{C,D}[v_{jk}^l]_C[x_k^{l-1}]_D + \kappa_{C,D}[w_{jk}^l]_C\mathcal{R}\{[x_k^{l-1}]_D\}\right).
\end{aligned}$$

Din

$$y_j^l := G^l(s_j^l),$$

avem

$$[y_j^l]_B = [G^l(s_j^l)]_B,$$

şi

$$\mathcal{R}\{[y_j^l]_B\} = \mathcal{R}\{[s_j^l]_B\}\frac{\partial [G^l(s_j^l)]_B}{\partial [s_j^l]_B},$$

deoarece $\frac{\partial [G^l(s_j^l)]_B}{\partial [s_j^l]_C} = 0$, pentru $C \neq B$. Acum, aplicând operatorul $\mathcal{R}$ în relaţia

$$\frac{\partial E}{\partial [w_{jk}^L]_A} = \sum_{\substack{B\in\mathcal{I} \\ \kappa_{A,D}e_Ae_D=e_B}} \frac{\partial E}{\partial [s_j^L]_B}\kappa_{A,D}[x_k^{L-1}]_D,$$

obţinem

$$\mathcal{R}\left\{\frac{\partial E}{\partial [w_{jk}^L]_A}\right\} = \sum_{\substack{B\in\mathcal{I} \\ \kappa_{A,D}e_Ae_D=e_B}} \left(\mathcal{R}\left\{\frac{\partial E}{\partial [s_j^L]_B}\right\}\kappa_{A,D}[x_k^{L-1}]_D + \frac{\partial E}{\partial [s_j^L]_B}\kappa_{A,D}\mathcal{R}\left\{[x_k^{L-1}]_D\right\}\right),$$

şi în relaţia

$$\frac{\partial E}{\partial [s_j^L]_B} = ([y_j^L]_B - [t_j]_B)\frac{\partial [G^L(s_j^L)]_B}{\partial [s_j^L]_B},$$

obţinem

$$\mathcal{R}\left\{\frac{\partial E}{\partial [s_j^L]_B}\right\} = \mathcal{R}\left\{[y_j^L]_B\right\}\frac{\partial [G^L(s_j^L)]_B}{\partial [s_j^L]_B} + ([y_j^L]_B - [t_j]_B)\frac{\partial^2 [G^L(s_j^L)]_B}{\partial [s_j^L]_B^2},$$

unde am folosit că $\frac{\partial^2 [G^L(s_j^L)]_B}{\partial [s_j^L]_B \partial [s_j^L]_C} = 0$, pentru $C \neq B$.

Trecând la un strat ascuns $l$, avem din relaţia

$$\frac{\partial E}{\partial [w_{jk}^l]_A} = \sum_{\substack{B\in\mathcal{I} \\ \kappa_{A,D}e_Ae_D=e_B}} \frac{\partial E}{\partial [s_j^l]_B}\kappa_{A,D}[x_k^{l-1}]_D,$$

că

$$\mathcal{R}\left\{\frac{\partial E}{\partial [w_{jk}^l]_A}\right\} = \sum_{\substack{B\in\mathcal{I} \\ \kappa_{A,D}e_Ae_D=e_B}} \left(\mathcal{R}\left\{\frac{\partial E}{\partial [s_j^l]_B}\right\}\kappa_{A,D}[x_k^{l-1}]_D + \frac{\partial E}{\partial [s_j^l]_B}\kappa_{A,D}\mathcal{R}\left\{[x_k^{l-1}]_D\right\}\right),$$

şi apoi din relaţia

$$\frac{\partial E}{\partial [s_j^l]_B} = \sum_r \left(\sum_{\substack{C\in\mathcal{I} \\ \kappa_{E,B}e_Ee_B=e_C}} \kappa_{E,B}[w_{rj}^{l+1}]_E \frac{\partial E}{\partial [s_r^{l+1}]_C}\right)\frac{\partial [G^l(s_j^l)]_B}{\partial [s_j^l]_B},$$

că

$$\begin{aligned}\mathcal{R}\left\{\frac{\partial E}{\partial [s_j^l]_B}\right\} &= \sum_r \mathcal{R}\left\{\sum_{\substack{C\in\mathcal{I}\\ \kappa_{E,B}e_Ee_E e_B=e_C}} \kappa_{E,B}[w_{rj}^{l+1}]_E \frac{\partial E}{\partial [s_r^{l+1}]_C}\frac{\partial [G^l(s_j^l)]_B}{\partial [s_j^l]_B}\right\}\\ &= \sum_r \sum_{\substack{C\in\mathcal{I}\\ \kappa_{E,B}e_Ee_E e_B=e_C}} \left(\mathcal{R}\left\{\frac{\partial E}{\partial [s_r^{l+1}]_C}\right\}\kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial [G^l(s_j^l)]_B}{\partial [s_j^l]_B}\right.\\ &+ \left.\frac{\partial E}{\partial [s_r^{l+1}]_C}\kappa_{E,B}[v_{rj}^{l+1}]_E\frac{\partial [G^l(s_j^l)]_B}{\partial [s_j^l]_B} + \frac{\partial E}{\partial [s_r^{l+1}]_C}\kappa_{E,B}[w_{rj}^{l+1}]_E\frac{\partial^2 [G^l(s_j^l)]_B}{\partial [s_j^l]_B^2}\right),\end{aligned}$$

unde din nou am folosit faptul că $\frac{\partial^2 [G^l(s_j^l)]_B}{\partial [s_j^l]_B \partial [s_j^l]_C} = 0$, pentru $C \neq B$.

## 4.6 Metode quasi-Newton

După cum am evidenţiat în secţiunea anterioară, principalul dezavantaj al metodei Newton este acela că hessiana şi inversa ei sunt dificil de calculat. Din acest motiv, *metodele quasi-Newton* urmăresc înlocuirea hessianei funcţiei de eroare cu o aproximare a ei, care ar fi preferabil să fie pozitiv definită prin construcţie, pentru a se evita problemele apărute în acest sens în cazul metodei Newton. Astfel, fie $E : \mathbb{R}^{2^n N} \to \mathbb{R}$ funcţia de eroare a unei reţele neuronale Clifford. Metoda Newton ne dă următoarea regulă de actualizare pentru vectorul $\overset{\mathcal{R}}{\mathbf{w}} \in \mathbb{R}^{2^n N}$ al componentelor ponderilor şi bias-urilor reţelei:

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - (\nabla^2 E(\overset{\mathcal{R}}{\mathbf{w}}_k))^{-1}\nabla E(\overset{\mathcal{R}}{\mathbf{w}}_k).$$

În metodele quasi-Newton se înlocuieşte inversa hessianei $(\nabla^2 E(\overset{\mathcal{R}}{\mathbf{w}}_k))^{-1}$ cu o matrice $\overset{\mathcal{R}}{\mathbf{H}}_{k+1}$ care se poate calcula mai uşor. Din dezvoltarea în serie Taylor de ordinul unu a gradientului $\nabla E$ al funcţiei de eroare, avem că

$$\nabla E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) = \nabla E(\overset{\mathcal{R}}{\mathbf{w}}_k) + \nabla^2 E(\overset{\mathcal{R}}{\mathbf{w}}_k)(\overset{\mathcal{R}}{\mathbf{w}}_{k+1} - \overset{\mathcal{R}}{\mathbf{w}}_k).$$

Dacă aproximăm hessiana funcţiei de eroare $\nabla^2 E(\overset{\mathcal{R}}{\mathbf{w}}_k)$ cu o matrice $\overset{\mathcal{R}}{\mathbf{B}}_{k+1}$ , relaţia de mai sus devine

$$\nabla E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) - \nabla E(\overset{\mathcal{R}}{\mathbf{w}}_k) = \overset{\mathcal{R}}{\mathbf{B}}_{k+1}(\overset{\mathcal{R}}{\mathbf{w}}_{k+1} - \overset{\mathcal{R}}{\mathbf{w}}_k). \tag{4.6.1}$$

Făcând notaţiile $\overset{\mathcal{R}}{\mathbf{g}}_k := \nabla E(\overset{\mathcal{R}}{\mathbf{w}}_k)$ şi respectiv

$$\overset{\mathcal{R}}{\mathbf{p}}_k := \overset{\mathcal{R}}{\mathbf{w}}_{k+1} - \overset{\mathcal{R}}{\mathbf{w}}_k,$$

şi

$$\overset{\mathcal{R}}{\mathbf{q}}_k := \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k,$$

ecuaţia (4.6.1) devine

$$\overset{\mathcal{R}}{\mathbf{q}}_k = \overset{\mathcal{R}}{\mathbf{B}}_{k+1}\overset{\mathcal{R}}{\mathbf{p}}_k, \tag{4.6.2}$$

relaţie care poartă denumirea de *ecuaţia secantei.* Aceasta poate fi scrisă echivalent şi sub forma

$$\overset{\mathcal{R}}{\mathbf{p}}_k = \overset{\mathcal{R}}{\mathbf{H}}_{k+1}\overset{\mathcal{R}}{\mathbf{q}}_k, \tag{4.6.3}$$

unde evident avem că $\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{B}}_{k+1}^{-1}$.

Cea mai simplă dintre metodele quasi-Newton este *metoda actualizării de rang unu*, în care se porneşte de la o matrice iniţială $\overset{\mathcal{R}}{\mathbf{H}}_0$ simetrică şi pozitiv definită, şi se calculează iterativ matricea $\overset{\mathcal{R}}{\mathbf{H}}_k$ folosind recurenţa

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{H}}_k + a_k\overset{\mathcal{R}}{\mathbf{r}}_k\overset{\mathcal{R}}{\mathbf{r}}_k^T, \tag{4.6.4}$$

unde $a_k \in \mathbb{R}$ şi vectorul $\overset{\mathcal{R}}{\mathbf{r}}_k \in \mathbb{R}^{2^n N}$ sunt alese astfel încât să fie satisfăcută ecuaţia secantei (4.6.2) sau (4.6.3), şi $\mathbf{r}^T$ reprezintă transpusa lui $\mathbf{r}$. Combinând acum ecuaţia (4.6.3) cu (4.6.4), rezultă că

$$\overset{\mathcal{R}}{\mathbf{p}}_k = \overset{\mathcal{R}}{\mathbf{H}}_{k+1}\overset{\mathcal{R}}{\mathbf{q}}_k = (\overset{\mathcal{R}}{\mathbf{H}}_k + a_k\overset{\mathcal{R}}{\mathbf{r}}_k\overset{\mathcal{R}}{\mathbf{r}}_k^T)\overset{\mathcal{R}}{\mathbf{q}}_k = \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k + a_k\overset{\mathcal{R}}{\mathbf{r}}_k\overset{\mathcal{R}}{\mathbf{r}}_k^T\overset{\mathcal{R}}{\mathbf{q}}_k.$$

De aici, rezultă că

$$(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)^T = a_k^2\overset{\mathcal{R}}{\mathbf{r}}_k(\overset{\mathcal{R}}{\mathbf{r}}_k^T\overset{\mathcal{R}}{\mathbf{q}}_k)^2\overset{\mathcal{R}}{\mathbf{r}}_k^T,$$

şi respectiv

$$(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)^T\mathbf{q}_k = (a_k\overset{\mathcal{R}}{\mathbf{r}}_k\overset{\mathcal{R}}{\mathbf{r}}_k^T\overset{\mathcal{R}}{\mathbf{q}}_k)^T\overset{\mathcal{R}}{\mathbf{q}}_k = a_k(\overset{\mathcal{R}}{\mathbf{q}}_k^T\overset{\mathcal{R}}{\mathbf{r}}_k)^2,$$

pe care dacă le împărţim, obţinem că

$$a_k\overset{\mathcal{R}}{\mathbf{r}}_k\overset{\mathcal{R}}{\mathbf{r}}_k^T = \frac{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)^T}{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)^T\overset{\mathcal{R}}{\mathbf{q}}_k},$$

şi întorcându-ne în ecuaţia (4.6.4), obţinem iteraţia pentru calculul aproximării inversei hessianei $\overset{\mathcal{R}}{\mathbf{H}}_k$ sub forma

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{H}}_k + \frac{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)^T}{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k\overset{\mathcal{R}}{\mathbf{q}}_k)^T\overset{\mathcal{R}}{\mathbf{q}}_k}, \tag{4.6.5}$$

care este cunoscută sub denumirea de *metoda actualizării de rang unu*. Din faptul că matricea iniţială $\overset{\mathcal{R}}{\mathbf{H}}_0$ este simetrică şi pozitiv definită, rezultă imediat că toate matricile $\overset{\mathcal{R}}{\mathbf{H}}_k$ au această proprietate.

Dacă dorim să facem direct aproximarea hessianei $\overset{\mathcal{R}}{\mathbf{B}}_k$, pornim la fel cu o matrice iniţială simetrică şi pozitiv definită $\overset{\mathcal{R}}{\mathbf{B}}_0$. Apoi, folosind formula Sherman-Morrison, care spune că dacă $\mathbf{A}$ este o matrice pătratică inversabilă de ordin $2^n N$ cu elemente reale, şi $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{2^n N}$ astfel încât $1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u} \neq 0$, atunci are loc

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}, \tag{4.6.6}$$

şi ecuaţia (4.6.5), obţinem actualizarea pentru $\overset{\mathcal{R}}{\mathbf{B}}_k$ sub forma

$$\overset{\mathcal{R}}{\mathbf{B}}_{k+1} = \overset{\mathcal{R}}{\mathbf{B}}_k + \frac{(\overset{\mathcal{R}}{\mathbf{q}}_k - \overset{\mathcal{R}}{\mathbf{B}}_k \overset{\mathcal{R}}{\mathbf{p}}_k)(\overset{\mathcal{R}}{\mathbf{q}}_k - \overset{\mathcal{R}}{\mathbf{B}}_k \overset{\mathcal{R}}{\mathbf{p}}_k)^T}{(\overset{\mathcal{R}}{\mathbf{q}}_k - \overset{\mathcal{R}}{\mathbf{B}}_k \overset{\mathcal{R}}{\mathbf{p}}_k)^T \overset{\mathcal{R}}{\mathbf{p}}_k}.$$

În continuare, pentru a prezenta metodele quasi-Newton cu actualizări de rang doi, folosim ideile din [76]. Pornim de la ecuaţia secantei scrisă sub forma (4.6.3):

$$\overset{\mathcal{R}}{\mathbf{p}}_k = \overset{\mathcal{R}}{\mathbf{H}}_{k+1} \overset{\mathcal{R}}{\mathbf{q}}_k.$$

Vom defini iteraţia pentru calculul lui $\overset{\mathcal{R}}{\mathbf{H}}_{k+1}$ sub forma

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \underset{\substack{\overset{\mathcal{R}}{\mathbf{H}} = \overset{\mathcal{R}}{\mathbf{H}}^T \\ \overset{\mathcal{R}}{\mathbf{p}}_k = \overset{\mathcal{R}}{\mathbf{H}} \overset{\mathcal{R}}{\mathbf{q}}_k}}{\arg\min} ||\overset{\mathcal{R}}{\mathbf{H}} - \overset{\mathcal{R}}{\mathbf{H}}_k||, \tag{4.6.7}$$

unde $||\overset{\mathcal{R}}{\mathbf{H}}||$ este norma Frobenius a matricii $\overset{\mathcal{R}}{\mathbf{H}}$, definită prin $||\overset{\mathcal{R}}{\mathbf{H}}|| := \sqrt{\mathrm{Tr}(\overset{\mathcal{R}}{\mathbf{H}} \overset{\mathcal{R}}{\mathbf{H}}^T)}$, $\mathrm{Tr}(\overset{\mathcal{R}}{\mathbf{H}})$ fiind urma matricii $\overset{\mathcal{R}}{\mathbf{H}}$. Relaţia (4.6.7) ne spune că vrem să găsim acea matrice simetrică $\overset{\mathcal{R}}{\mathbf{H}}_{k+1}$ care este cea mai apropiată în sensul normei Frobenius de matricea $\overset{\mathcal{R}}{\mathbf{H}}_k$ şi care satisface ecuaţia secantei (4.6.3). Se poate demonstra uşor că soluţia pentru (4.6.7) este dată de

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{H}}_k + \frac{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k) \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}.$$

Luând descompunerea Cholesky a matricii $\overset{\mathcal{R}}{\mathbf{H}}_{k+1}$ sub forma

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{L}}_{k+1} \overset{\mathcal{R}}{\mathbf{L}}_{k+1}^T,$$

se poate demonstra că $\overset{\mathcal{R}}{\mathbf{H}}_{k+1}$ este simetrică şi pozitiv definită dacă şi numai dacă există $\overset{\mathcal{R}}{\mathbf{v}}_k \in \mathbb{R}^{2^n N}$ astfel încât $\overset{\mathcal{R}}{\mathbf{L}}_{k+1} \overset{\mathcal{R}}{\mathbf{v}}_k = \overset{\mathcal{R}}{\mathbf{p}}_k$ şi $\overset{\mathcal{R}}{\mathbf{v}}_k = \overset{\mathcal{R}}{\mathbf{L}}_{k+1}^T \overset{\mathcal{R}}{\mathbf{q}}_k$. Dacă vom lua pe $\overset{\mathcal{R}}{\mathbf{L}}_{k+1}$ astfel încât

$$\overset{\mathcal{R}}{\mathbf{L}}_{k+1} = \underset{\substack{\overset{\mathcal{R}}{\mathbf{L}} = \overset{\mathcal{R}}{\mathbf{L}}^T \\ \overset{\mathcal{R}}{\mathbf{p}}_k = \overset{\mathcal{R}}{\mathbf{L}} \overset{\mathcal{R}}{\mathbf{v}}_k}}{\arg\min} ||\overset{\mathcal{R}}{\mathbf{L}} - \overset{\mathcal{R}}{\mathbf{L}}_k||,$$

rezultă ca mai sus că

$$\overset{\mathcal{R}}{\mathbf{L}}_{k+1} = \overset{\mathcal{R}}{\mathbf{L}}_k + \frac{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{L}}_k \overset{\mathcal{R}}{\mathbf{v}}_k) \overset{\mathcal{R}}{\mathbf{v}}_k^T}{\overset{\mathcal{R}}{\mathbf{v}}_k^T \overset{\mathcal{R}}{\mathbf{v}}_k}. \tag{4.6.8}$$

Avem în continuare că

$$\overset{\mathcal{R}}{\mathbf{v}}_k = \overset{\mathcal{R}}{\mathbf{L}}_{k+1}^T \overset{\mathcal{R}}{\mathbf{q}}_k = \overset{\mathcal{R}}{\mathbf{L}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k - \overset{\mathcal{R}}{\mathbf{v}}_k^T \overset{\mathcal{R}}{\mathbf{L}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{v}}_k^T \overset{\mathcal{R}}{\mathbf{v}}_k} \overset{\mathcal{R}}{\mathbf{v}}_k,$$

de unde rezultă că există $\alpha_k \in \mathbb{R}$ astfel încât să avem $\overset{\mathcal{R}}{\mathbf{v}}_k = \alpha_k \overset{\mathcal{R}}{\mathbf{L}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k$. Obţinem mai departe că

$$\alpha_k = 1 + \frac{\alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k - \alpha_k^2 \overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k}{\alpha_k^2 \overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k},$$

şi apoi

$$\alpha_k^2 = \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k},$$

iar dacă revenim în (4.6.8), rezultă că

$$\overset{\mathcal{R}}{\mathbf{L}}_{k+1} = \overset{\mathcal{R}}{\mathbf{L}}_k \left( \mathbf{I}_{2^n N} \pm \sqrt{\frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k}} \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k} - \frac{\overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k} \right).$$

ţinând seama că $\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{L}}_{k+1} \overset{\mathcal{R}}{\mathbf{L}}_{k+1}^T$, obţinem în final actualizarea pentru calculul matricii $\overset{\mathcal{R}}{\mathbf{H}}_k$:

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{H}}_k + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k} - \frac{\overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k}.$$

Aceasta este cunoscută sub denumirea de *metoda Davidon-Fletcher-Powell (DFP)*, a se vedea [85]. Folosind formula Sherman-Morrison dată în (4.6.6), obţinem actualizarea pentru aproximarea $\overset{\mathcal{R}}{\mathbf{B}}_k$ a hessianei ca fiind

$$\overset{\mathcal{R}}{\mathbf{B}}_{k+1} = \left( \mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k} \right) \overset{\mathcal{R}}{\mathbf{B}}_k \left( \mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k} \right) + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}.$$

Acum, pornind cu ecuaţia secantei sub forma (4.6.2)

$$\overset{\mathcal{R}}{\mathbf{B}}_{k+1} \overset{\mathcal{R}}{\mathbf{p}}_k = \overset{\mathcal{R}}{\mathbf{q}}_k,$$

şi definind iteraţia pentru calculul lui $\overset{\mathcal{R}}{\mathbf{B}}_{k+1}$ sub forma

$$\overset{\mathcal{R}}{\mathbf{B}}_{k+1} = \underset{\substack{\overset{\mathcal{R}}{\mathbf{B}} = \overset{\mathcal{R}}{\mathbf{B}}^T \\ \overset{\mathcal{R}}{\mathbf{q}}_k = \overset{\mathcal{R}}{\mathbf{B}} \overset{\mathcal{R}}{\mathbf{p}}_k}}{\arg\min} ||\overset{\mathcal{R}}{\mathbf{B}} - \overset{\mathcal{R}}{\mathbf{B}}_k||,$$

obţinem, analog ca mai sus, că

$$\overset{\mathcal{R}}{\mathbf{B}}_{k+1} = \overset{\mathcal{R}}{\mathbf{B}}_k + \frac{(\overset{\mathcal{R}}{\mathbf{q}}_k - \overset{\mathcal{R}}{\mathbf{B}}_k \overset{\mathcal{R}}{\mathbf{p}}_k) \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k},$$

unde am folosit aceeaşi normă Frobenius pentru matrici.

Folosind acelaşi raţionament ca în cazul metodei DFP, avem în final următoarea actualizare pentru $\overset{\mathcal{R}}{\mathbf{B}}_k$:

$$\overset{\mathcal{R}}{\mathbf{B}}_{k+1} = \overset{\mathcal{R}}{\mathbf{B}}_k + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} - \frac{\overset{\mathcal{R}}{\mathbf{B}}_k \overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{B}}_k}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{B}}_k \overset{\mathcal{R}}{\mathbf{p}}_k}.$$

Aceasta poartă denumirea de *metoda Broyden-Fletcher-Goldfarb-Shanno (BFGS)*, a se vedea [238]. Folosind din nou formula Sherman-Morrison (4.6.6), obţinem acum actualizarea pentru aproximarea $\overset{\mathcal{R}}{\mathbf{H}}_k$ a inversei hessianei ca fiind

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \left(\mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right) \overset{\mathcal{R}}{\mathbf{H}}_k \left(\mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right) + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}.$$

O metodă simplificatoare care presupune că $\overset{\mathcal{R}}{\mathbf{H}}_k = \mathbf{I}_{2^n N}$, calculează următoarea direcţie de căutare după formula

$$\overset{\mathcal{R}}{\mathbf{d}}_{k+1} = -\overset{\mathcal{R}}{\mathbf{H}}_{k+1} \overset{\mathcal{R}}{\mathbf{g}}_{k+1},$$

unde, dacă facem înlocuirea

$$\begin{aligned}
\overset{\mathcal{R}}{\mathbf{H}}_{k+1} &= \left(\mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right)\left(\mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right) + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} \\
&= \mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} - \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\left(\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k\right)^2} + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} \\
&= \mathbf{I}_{2^n N} - \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{q}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} - \frac{\overset{\mathcal{R}}{\mathbf{q}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} + \frac{\overset{\mathcal{R}}{\mathbf{p}}_k \overset{\mathcal{R}}{\mathbf{p}}_k^T}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\left(1 + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right),
\end{aligned}$$

avem

$$\begin{aligned}
\overset{\mathcal{R}}{\mathbf{d}}_{k+1} &= -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + \overset{\mathcal{R}}{\mathbf{p}}_k \frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} + \overset{\mathcal{R}}{\mathbf{q}}_k \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} - \overset{\mathcal{R}}{\mathbf{p}}_k \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\left(1 + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right) \\
&= -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + \overset{\mathcal{R}}{\mathbf{p}}_k \left(\frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} - \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\left(1 + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right)\right) + \overset{\mathcal{R}}{\mathbf{q}}_k \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k},
\end{aligned}$$

sau, scriind compact avem că

$$\overset{\mathcal{R}}{\mathbf{d}}_{k+1} = -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + A_k \overset{\mathcal{R}}{\mathbf{p}}_k + B_k \overset{\mathcal{R}}{\mathbf{q}}_k,$$

unde

$$A_k = -\left(1 + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{q}}_k}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}\right) \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k} + \frac{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k},$$

$$B_k = \frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{R}}{\mathbf{q}}_k^T \overset{\mathcal{R}}{\mathbf{p}}_k}.$$

Metoda descrisă poartă numele de *one step secant* (a se vedea [23]), şi este o metodă aflată între metoda gradienţilor conjugaţi şi metodele quasi-Newton, mai exact metoda BFGS din care a fost derivată.

## 4.7 Metoda Levenberg-Marquardt

Metoda Levenberg-Marquardt a fost pentru prima dată aplicată reţelelor neuronale în [102]. Prezentarea noastră urmăreşte îndeosebi pe cea din [277]. Pornim de la expresia funcţiei de eroare scrisă sub forma

$$E = \frac{1}{2}\sum_{m=1}^{M}\sum_{j=1}^{c}||e_j^m||^2 = \frac{1}{2}\sum_{m=1}^{M}\sum_{j=1}^{c}\sum_{I\in\mathcal{I}}[e_j^m]_I^2,$$

unde $e_j^m := y_j^{L,m} - t_j^m$ reprezintă eroarea pentru fiecare dintre cele $m$ tipare de antrenare. Notăm cu $\mathbf{w}$ vectorul tuturor celor $N$ ponderi şi bias-uri ale reţelei, însă acum vom nota fiecare pondere cu $w_i$. Ca mai sus, vom nota $\overset{\mathcal{R}}{\mathbf{w}} = (([\mathbf{w}]_I^T)_{I\in\mathcal{I}})^T$. Putem forma acum matricea Jacobiană de dimensiune $2^n cM \times 2^n N$, ale cărei componente sunt derivatele parţiale ale fiecărei erori de forma $e_j^m$ în raport cu ponderile $w_i$, $1 \le i \le N$:

$$\overset{\mathcal{R}}{\mathbf{J}} = \begin{pmatrix}
\frac{\partial [e_1^1]_0}{\partial [w_1]_0} & \cdots & \frac{\partial [e_1^1]_0}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_1^1]_0}{\partial [w_N]_0} & \cdots & \frac{\partial [e_1^1]_0}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_1^M]_0}{\partial [w_1]_0} & \cdots & \frac{\partial [e_1^M]_0}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_1^M]_0}{\partial [w_N]_0} & \cdots & \frac{\partial [e_1^M]_0}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_c^1]_0}{\partial [w_1]_0} & \cdots & \frac{\partial [e_c^1]_0}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_c^1]_0}{\partial [w_N]_0} & \cdots & \frac{\partial [e_c^1]_0}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_c^M]_0}{\partial [w_1]_0} & \cdots & \frac{\partial [e_c^M]_0}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_c^M]_0}{\partial [w_N]_0} & \cdots & \frac{\partial [e_c^M]_0}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_1^1]_{2^n-1}}{\partial [w_1]_0} & \cdots & \frac{\partial [e_1^1]_{2^n-1}}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_1^1]_{2^n-1}}{\partial [w_N]_0} & \cdots & \frac{\partial [e_1^1]_{2^n-1}}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_1^M]_{2^n-1}}{\partial [w_1]_0} & \cdots & \frac{\partial [e_1^M]_{2^n-1}}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_1^M]_{2^n-1}}{\partial [w_N]_0} & \cdots & \frac{\partial [e_1^M]_{2^n-1}}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_c^1]_{2^n-1}}{\partial [w_1]_0} & \cdots & \frac{\partial [e_c^1]_{2^n-1}}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_c^1]_{2^n-1}}{\partial [w_N]_0} & \cdots & \frac{\partial [e_c^1]_{2^n-1}}{\partial [w_N]_{2^n-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial [e_c^M]_{2^n-1}}{\partial [w_1]_0} & \cdots & \frac{\partial [e_c^M]_{2^n-1}}{\partial [w_1]_{2^n-1}} & \cdots & \frac{\partial [e_c^M]_{2^n-1}}{\partial [w_N]_0} & \cdots & \frac{\partial [e_c^M]_{2^n-1}}{\partial [w_N]_{2^n-1}}
\end{pmatrix}.$$

Acum, hessiana funcţiei de eroare este matricea de dimensiune $2^n N \times 2^n N$, ale cărei componente sunt derivatele parţiale de ordinul doi ale funcţiei de eroare $E$ în raport

cu ponderile $w_i$, $1 \leq i \leq N$:

$$\overset{\mathcal{R}}{\mathbf{H}} = \begin{pmatrix} \frac{\partial^2 E}{\partial [w_1]_0 \partial [w_1]_0} & \cdots & \frac{\partial^2 E}{\partial [w_1]_0 \partial [w_1]_{2^n-1}} & \cdots & \frac{\partial^2 E}{\partial [w_1]_0 \partial [w_N]_0} & \cdots & \frac{\partial^2 E}{\partial [w_1]_0 \partial [w_N]_{2^n-1}} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial [w_1]_{2^n-1} \partial [w_1]_0} & \cdots & \frac{\partial^2 E}{\partial [w_1]_{2^n-1} \partial [w_1]_{2^n-1}} & \cdots & \frac{\partial^2 E}{\partial [w_1]_{2^n-1} \partial [w_N]_0} & \cdots & \frac{\partial^2 E}{\partial [w_1]_{2^n-1} \partial [w_N]_{2^n-1}} \\ \frac{\partial^2 E}{\partial [w_2]_0 \partial [w_1]_0} & \cdots & \frac{\partial^2 E}{\partial [w_2]_0 \partial [w_1]_{2^n-1}} & \cdots & \frac{\partial^2 E}{\partial [w_2]_0 \partial [w_N]_0} & \cdots & \frac{\partial^2 E}{\partial [w_2]_0 \partial [w_N]_{2^n-1}} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial [w_2]_{2^n-1} \partial [w_1]_0} & \cdots & \frac{\partial^2 E}{\partial [w_2]_{2^n-1} \partial [w_1]_{2^n-1}} & \cdots & \frac{\partial^2 E}{\partial [w_2]_{2^n-1} \partial [w_N]_0} & \cdots & \frac{\partial^2 E}{\partial [w_2]_{2^n-1} \partial [w_N]_{2^n-1}} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial [w_N]_0 \partial [w_1]_0} & \cdots & \frac{\partial^2 E}{\partial [w_N]_0 \partial [w_1]_{2^n-1}} & \cdots & \frac{\partial^2 E}{\partial [w_N]_0 \partial [w_N]_0} & \cdots & \frac{\partial^2 E}{\partial [w_N]_0 \partial [w_N]_0} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial [w_N]_{2^n-1} \partial [w_1]_0} & \cdots & \frac{\partial^2 E}{\partial [w_N]_{2^n-1} \partial [w_1]_{2^n-1}} & \cdots & \frac{\partial^2 E}{\partial [w_N]_0 \partial [w_N]_0} & \cdots & \frac{\partial^2 E}{\partial [w_N]_{2^n-1} \partial [w_N]_{2^n-1}} \end{pmatrix}.$$

Vom defini vectorul $2^n cM \times 1$

$$\overset{\mathcal{R}}{\mathbf{e}} = \begin{pmatrix} [e_1^1]_0 \\ \vdots \\ [e_1^M]_0 \\ \vdots \\ [e_c^1]_0 \\ \vdots \\ [e_c^M]_0 \\ \vdots \\ [e_1^1]_{2^n-1} \\ \vdots \\ [e_1^M]_{2^n-1} \\ \vdots \\ [e_c^1]_{2^n-1} \\ \vdots \\ [e_c^M]_{2^n-1} \end{pmatrix},$$

de unde gradientul $\overset{\mathcal{R}}{\mathbf{g}}$ de dimensiune $2^n N \times 1$ va fi

$$\overset{\mathcal{R}}{\mathbf{g}} = \overset{\mathcal{R}}{\mathbf{J}}^T \overset{\mathcal{R}}{\mathbf{e}}, \tag{4.7.1}$$

pentru că

$$\frac{\partial E}{\partial [w_i]_A} = \sum_{m=1}^{M} \sum_{j=1}^{c} \sum_{I \in \mathcal{I}} \frac{\partial [e_j^m]_I}{\partial [w_i]_A} [e_j^m]_I, \forall A \in \mathcal{I}.$$

Avem acum pentru derivatele parţiale de ordinul doi ale funcţiei de eroare $E$ în raport cu ponderile $w_i$, $1 \leq i \leq N$:

$$\frac{\partial^2 E}{\partial [w_i]_A \partial [w_k]_B} = \sum_{m=1}^{M} \sum_{j=1}^{c} \sum_{I \in \mathcal{I}} \left( \frac{\partial^2 [e_j^m]_I}{\partial [w_i]_A \partial [w_k]_B} [e_j^m]_I + \frac{\partial [e_j^m]_I}{\partial [w_i]_A} \frac{\partial [e_j^m]_I}{\partial [w_k]_B} \right), \forall A, B \in \mathcal{I}.$$

Algoritmul Levenberg-Marquardt se bazează pe aproximarea

$$\frac{\partial^2 [e_j^m]_I}{\partial [w_i]_A \partial [w_k]_B} \approx 0,$$

iar expresia pentru un element al hessianei $\overset{\mathcal{R}}{\mathbf{H}}$ devine acum

$$\frac{\partial^2 E}{\partial [w_i]_A \partial [w_k]_B} = \sum_{m=1}^{M} \sum_{j=1}^{c} \sum_{I \in \mathcal{I}} \frac{\partial [e_j^m]_I}{\partial [w_i]_A} \frac{\partial [e_j^m]_I}{\partial [w_k]_B},$$

de unde avem aproximarea

$$\overset{\mathcal{R}}{\mathbf{H}} \approx \overset{\mathcal{R}}{\mathbf{J}}^T \overset{\mathcal{R}}{\mathbf{J}}.$$

Folosind această aproximare, formula de actualizare modificată din cadrul metodei Newton

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - (\overset{\mathcal{R}}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2N})^{-1} \overset{\mathcal{R}}{\mathbf{g}}_k,$$

unde $\lambda_k$ este o constantă care se alege astfel încât matricea $\overset{\mathcal{R}}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{2^n N}$ să fie pozitiv definită, devine

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - (\overset{\mathcal{R}}{\mathbf{J}}_k^T \overset{\mathcal{R}}{\mathbf{J}}_k + \lambda_k \mathbf{I}_{2N})^{-1} \overset{\mathcal{R}}{\mathbf{J}}_k^T \overset{\mathcal{R}}{\mathbf{e}}_k,$$

unde am folosit şi relaţia (4.7.1). Aceasta este regula de actualizare pentru *metoda Levenberg-Marquardt* (a se vedea [174]).

## 4.8 Concluzii

Acest capitol este dedicat prezentării unor noi algoritmi de învăţare pentru reţele neuronale Clifford de tip feedforward, algoritmi care au fost generalizaţi din domeniul real. Doar metoda gradient (a se vedea, spre exemplu [211]) a fost extinsă anterior în domeniul Clifford, restul reprezentând o contribuţie originală a tezei. Toţi aceşti algoritmi pot fi particularizaţi uşor pentru reţele neuronale cu valori complexe, cu valori hiperbolice, sau cu valori cuaternionice. Pentru fiecare dintre aceste tipuri de reţele, algoritmii nu au fost încă deduşi în literatura de specialitate, chiar dacă în domeniul reţelelor neuronale cu valori cuaternionice cercetarea s-a dezvoltat foarte mult în ultimii doi-trei ani. Astfel,

- Secţiunea 4.1 a prezentat algoritmul backpropagation sau **metoda gradient** pentru reţelele Clifford folosind calculul diferenţial cu derivate parţiale reale, singurul posibil în acest context.
- În Secţiunea 4.2, au fost introduse **metoda gradient cu moment**, **metodele quickprop**, **resilient backpropagation**, **delta-bar-delta** şi **SuperSAB**. Acestea fac parte din clasa metodelor gradient îmbunătăţite, şi constituie un prim pas făcut în direcţia găsirii unor algoritmi care să depăşească performanţele metodei gradient clasice.
- O metodă ceva mai complicată, **metoda gradienţilor conjugaţi** a fost prezentată în Secţiunea 4.3. După ce am discutat cazul unei funcţii de eroare pătratice, corespunzător metodei gradienţilor conjugaţi liniară, am prezentat metoda gradienţilor conjugaţi neliniară, în cadrul căreia am dedus formule pentru **metoda**

**gradienţilor conjugaţi cu actualizări Hestenes-Stiefel**, **metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere**, **metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves**, **metoda gradienţilor conjugaţi cu actualizări Dai-Yuan**, **variaţiile algoritmilor Hestenes-Stiefel** şi **Polak-Ribiere** şi **metoda gradienţilor conjugaţi cu reporniri Powell-Beale**.

- O extindere interesantă a metodei gradienţilor conjugaţi este **metoda gradienţilor conjugaţi scalaţi**, dedusă pentru reţelele neuronale cu valori Clifford de tip feedforward în Secţiunea 4.4. Această metodă reprezintă o îmbunătăţire a metodei de bază cu elemente care corectează principalele neajunsuri ale acestui algoritm de învăţare.

- Secţiunea 4.5 prezintă **metoda Newton**. Elementul principal al metodei Newton este hessiana funcţiei de eroare a cărei inversă trebuie calculată exact. Din acest motiv, în cadrul acestei secţiuni am făcut deducerea completă a metodei de calcul a **hessianei** funcţiei de eroare pentru o reţea neuronală cu valori Clifford de tip feedforward, precum şi o particularizare a acestei metode pentru o reţea cu un singur strat ascuns. De asemenea, tot în cadrul acestei secţiuni, am prezentat o metodă de calcul a **produsului hessianei cu un vector**, cantitate care este necesară pentru implementarea multor algoritmi de învăţare.

- O variantă mai simplă din punct de vedere computaţional al metodei Newton este **metoda quasi-Newton**, discutată în Secţiunea 4.6. Această metodă înlocuieşte calculul dificil computaţional al hessianei din metoda Newton, cu calculul iterativ al unei aproximări a inversei acestei matrici. Patru algoritmi din această clasă de metode au fost detaliaţi în această secţiune, şi anume **metoda actualizării de rang unu**, **metoda Davidon-Fletcher-Powell**, **metoda Broyden-Fletcher-Goldfarb-Shanno** şi **metoda one step secant**.

- Ultima metodă prezentată este **metoda Levenberg-Marquardt**, în Secţiunea 4.7. În această metodă, hessiana din metoda Newton este înlocuită cu o altă aproximare a ei, şi anume una în care este folosit un Jacobian care este mai uşor de calculat decât hessiana.

# Capitolul 5

# Generalizări ale reţelelor neuronale Clifford

Prezentul capitol are în vedere diferite generalizări, altele decât reţelele Clifford, ale reţelelor neuronale cu valori reale, precum şi generalizări ale reţelelor Clifford. El se înscrie tot în cadrul demersului de a obţine reţele neuronale care să realizeze o reprezentare diferită a datelor faţă de reţelele clasice fiind, practic, o altă abordare făcută în direcţia reţelelor neuronale definite pe algebre multidimensionale. Apartenenţa la o algebră este singura condiţie pe care trebuie să o satisfacă intrările, ieşirile şi ponderile unei reţele, deoarece pentru definirea unui model de reţea neuronală, trebuie să avem operaţiile de adunare şi înmulţire, care să dea rezultate în mulţimea de intrare.

Astfel, Secţiunea 5.1 are în vedere reţele neuronale cu valori vectori tridimensionali, care au fost prezentate anterior în literatură. Ele sunt definite pe algebra vectorilor de dimensiune $3$, ceea ce le face mai eficiente în reprezentarea datelor tridimensionale decât, spre exemplu, reţelele Clifford definite pe algebre de dimensiune $4$. După o scurtă introducere a celor mai importante contribuţii din domeniu, este prezentat algoritmul backpropagation pentru antrenarea unei reţele de tip feedforward cu valori vectori tridimensionali.

O abordare oarecum diferită, este cea din Secţiunea 5.2, unde sunt introduse reţele neuronale cu valori vectori $n$-dimensionali. În literatură există doar două contribuţii care vizează un singur neuron $n$-dimensional, în care ponderea este o matrice ortogonală. Abordarea noastră este mai generală, deoarece sunt definite reţele neuronale de tip feedforward cu valori vectori $n$-dimensionali, în care intrările şi ieşirile sunt vectori $n$-dimensionali, dar în care ponderile sunt matrici pătratice, fără proprietăţi speciale. Pe viitor, intenţionăm să definim reţele neuronale care să generalizeze neuronul $n$-dimensional existent în literatură, adică să aibă ponderi matrici ortogonale.

Secţiunea 5.3 este dedicată introducerii, în premieră după cunoştinţele noastre, a reţelelor neuronale cu valori matrici pătratice. Ele constituie o generalizare directă a reţelelor Clifford, deoarece orice algebră Clifford se poate scrie ca o subalgebră a unei algebre de matrici pătratice de dimensiune $2^n$. Motivul pentru care am introdus aceste reţele este, în primul rând, gradul lor de generalitate şi dimensiunea algebrei de $n^2$, cu mult mai mică decât dimensiunea unei algebre Clifford, care este $2^n$. Apoi, calculul matricial este extrem de folosit în multe ramuri ale ingineriei, şi nu numai, fapt care dă speranţe că pe viitor aceste reţele vor putea fi utilizate pentru a rezolva mai eficient probleme care apar în acest domeniu. Imaginile sunt reprezentate ca matrici de pixeli, ceea ce ar face aceste reţele foarte potrivite pentru probleme ce ţin de procesarea imaginilor, practic o imagine fiind o singură intrare pentru aceste reţele, şi nu multe intrări reale, cum se întâmplă în prezent. Este introdusă o reţea

feedforward cu valori matriciale, pentru care este dedus algoritmul backpropagation, ca în secţiunile precedente ale acestui capitol.

Următoarea secţiune, Secţiunea 5.4, defineşte, tot în premieră după cunoştinţele noastre, reţelele neuronale cu valori matrici antisimetrice. Deoarece algebra vectorilor de dimensiune $3$ cu adunarea şi produsul vectorial este izomorfă cu algebra matricilor antisimetrice de dimensiune $3$ cu adunarea şi înmulţirea matricilor, aceste reţele constituie o generalizare directă a reţelelor neuronale cu valori vectori tridimensionali, introduse în Secţiunea 5.1. Motivaţia introducerii acestor reţele este tot una de generalizare a acestor reţele anterior definite în literatură, însă şi una practică, vizând posibilitatea manipulării unor obiecte geometrice folosind aceste reţele. Deoarece algebra matricilor antisimetrice face parte din clasa mai generală a algebrelor Lie, a căror definiţie vine din geometrie, având multe aplicaţii în fizică şi inginerie (a se vedea, spre exemplu, [130, 74, 229]), iar mai recent în computer vision (a se vedea, spre exemplu, [269], şi referinţele citate în acest articol), am considerat oportună definirea unor reţele neuronale pe această algebră, ele putând avea pe viitor aplicaţii în aceste domenii. Ca mai sus, metoda gradient de antrenare a unei astfel de reţele de tip feedforward este prezentată în detaliu.

În Secţiunea 5.5 este introdus un nou algoritm, numit algoritmul de încorporare, pentru medierea matricilor ortogonale. Problema medierii matricilor ortogonale, care sunt de fapt reprezentări ale rotaţiilor, este una importantă în mai multe domenii ale ingineriei (pentru o descriere completă a acestei probleme, a soluţiilor existente şi a aplicaţiilor, a se vedea, spre exemplu, [109]). Astfel, este dezvoltat un algoritm care presupune calcule doar în coordonate carteziene, fără a folosi coordonate locale pe grupul matricilor ortogonale. După o introducere în problematica de interes, a doua subsecţiune prezintă algoritmul în cazul general, pe o varietate riemanniană. Apoi, subsecţiunea a treia, particularizează algoritmul general pentru matricile ortogonale de ordinul $3$, arătând cum se aplică acesta pentru cele mai cunoscute patru funcţii de cost din literatură. Ultima subsecţiune vizează matricile ortogonale de ordinul $n$, cu particularizarea algoritmului de încorporare pentru aceleaşi patru funcţii de cost.

Secţiunea 5.6 prezintă concluziile capitolului de faţă.

## 5.1 Reţele neuronale cu valori vectori tridimensionali

O abordare diferită a generalizării în algebre multidimensionale a reţelelor neuronale cu valori reale, care nu are nicio legătură directă cu diferitele algebre Clifford discutate mai sus, este aceea a *reţelelor neuronale cu valori vectoriale*, definite în [185, 186, 188, 199]. Aceste reţele procesează vectorii tridimensionali în două feluri: unul bazat pe produsul vectorial, în care ponderile sunt tot vectori tridimensional, şi unul în care ponderile sunt matrici ortogonale (i.e. matrici care satisfac $AA^T = A^TA = I$). Domeniul acestor reţele neuronale este încă la început, rezultatele experimentale pe transformări geometrice fiind promiţătoare pentru aplicaţii viitoare în procesarea imaginilor.

Să considerăm spaţiul vectorial de dimensiune $3$ al vectorilor cu valori reale, $\mathbb{R}^3$, pe care îl vom nota în continuare $\mathcal{V}_3(\mathbb{R}) := \mathbb{R}^3$.

În cele ce urmează vom introduce reţele neuronale cu valori din $\mathcal{V}_3(\mathbb{R})$, bazate pe produsul vectorial. Să presupunem că avem o reţea neuronală cu valori vectoriale de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile $\{2, \ldots, L-1\}$ sunt straturi ascunse.

Funcţia de eroare $E : \mathcal{V}_3(\mathbb{R})^N \to \mathbb{R}$ pentru o asemenea reţea este definită prin

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{c} ||y_i^L - t_i||^2, \tag{5.1.1}$$

unde $||y||$ reprezintă norma în $\mathcal{V}_n(\mathbb{R})$ a lui $y$. $\mathbf{y}^L = (y_i^L)_{1\leq i\leq c} \in \mathcal{V}_3(\mathbb{R})^c$ reprezintă *ieşirile* reţelei, $\mathbf{t} = (t_i)_{1\leq i\leq c} \in \mathcal{V}_3(\mathbb{R})^c$ reprezintă *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{w} \in \mathcal{V}_3(\mathbb{R})^N$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei, care sunt vectori din $\mathcal{V}_3(\mathbb{R})$.

Notând cu $w_{jk}^l \in \mathcal{V}_3(\mathbb{R})$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l \in \{2, \ldots, L\}$, putem defini *pasul de actualizare* al ponderii $w_{jk}^l$ în epoca $t$ ca fiind

$$\Delta w_{jk}^l(t) = w_{jk}^l(t+1) - w_{jk}^l(t).$$

În cazul *metodei gradient*, *regula de actualizare* pentru ponderea $w_{jk}^l$ se poate scrie

$$\Delta w_{jk}^l(t) = -\varepsilon \left( \frac{\partial E}{\partial [w_{jk}^l]_1}(t), \frac{\partial E}{\partial [w_{jk}^l]_2}(t), \frac{\partial E}{\partial [w_{jk}^l]_3}(t) \right)^T,$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi $\frac{\partial E}{\partial [w_{jk}^l]_i}(t)$ este derivata parţială a funcţiei de eroare $E$ în raport cu fiecare componentă $[w_{jk}^l]_i$ a vectorului $w_{jk}^l$, unde $1 \leq i \leq 3$.

Scris sub formă vectorială, ca vectori de vectori, pasul de actualizare devine

$$\Delta \mathbf{w}(t) = \mathbf{w}(t+1) - \mathbf{w}(t),$$

unde $\mathbf{w} \in \mathcal{V}_3(\mathbb{R})^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient ia acum forma:

$$\Delta \mathbf{w}(t) = -\varepsilon \nabla E(t),$$

unde $\nabla E(t) := \nabla E(\mathbf{w}(t)) \in \mathcal{V}_3(\mathbb{R})^N$ este gradientul funcţiei $E$, ale cărui componente

$$\left( \frac{\partial E}{\partial [w_{jk}^l]_1}(\mathbf{w}(t)), \frac{\partial E}{\partial [w_{jk}^l]_2}(\mathbf{w}(t)), \frac{\partial E}{\partial [w_{jk}^l]_3}(\mathbf{w}(t)) \right)^T,$$

au fost scrise prescurtat mai sus sub forma

$$\left( \frac{\partial E}{\partial [w_{jk}^l]_1}(t), \frac{\partial E}{\partial [w_{jk}^l]_2}(t), \frac{\partial E}{\partial [w_{jk}^l]_3}(t) \right)^T.$$

Prin urmare, pentru metoda gradient, avem nevoie să calculăm derivatele parţiale de forma $\frac{\partial E}{\partial [w_{jk}^l]_i}(t)$, cu $1 \leq i \leq 3$.

Facem mai întâi următoarele notaţii

$$s_j^l = \sum_k w_{jk}^l \times x_k^{l-1}, \tag{5.1.2}$$

$$y_j^l = G^l(s_j^l), \tag{5.1.3}$$

unde (5.1.2) ne arată faptul că înmulţirea de la reţelele neuronale cu valori reale este înlocuită de produsul vectorial dintre doi vectori, $G^l$ este *funcţia de activare* a stratului $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1 \leq k \leq d} \in \mathcal{V}_3(\mathbb{R})^d$ sunt intrările reţelei, şi avem că $x_k^l = y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$.

Funcţia de activare este considerată a fi definită pe componente. Un exemplu de funcţie de activare este funcţia tangentă hiperbolică pe componente definită, pentru orice vector $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \in \mathcal{V}_3(\mathbb{R})$, prin

$$G\left(\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}\right) = \begin{pmatrix} \tanh a_1 \\ \tanh a_2 \\ \tanh a_3 \end{pmatrix}.$$

Vom începe cu actualizarea ponderilor dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta w_{jk}^L(t) = -\varepsilon \left( \frac{\partial E}{\partial [w_{jk}^L]_1}(t), \frac{\partial E}{\partial [w_{jk}^L]_2}(t), \frac{\partial E}{\partial [w_{jk}^L]_3}(t) \right)^T.$$

Regula înlănţuirii ne permite să scriem următorul set de relaţii:

$$\frac{\partial E}{\partial [w_{jk}^L]_i} = \sum_{c=1}^{3} \frac{\partial E}{\partial [s_j^L]_c} \frac{\partial [s_j^L]_c}{\partial [w_{jk}^L]_i}, \tag{5.1.4}$$

pentru orice $1 \leq i \leq 3$.

Ne vom ocupa mai întâi de $\frac{\partial [s_j^L]_c}{\partial [w_{jk}^L]_i}$. Pentru calculul acestei derivate, avem nevoie de o formulă explicită pentru $[s_j^L]_c$, care se poate deduce din (5.1.2):

$$[s_j^L]_c = \sum_k [w_{jk}^L]_{c+1} [x_k^{L-1}]_{c+2} - [w_{jk}^L]_{c+2} [x_k^{L-1}]_{c+1}, \forall 1 \leq c \leq 3.$$

Acum, se vede uşor că

$$\frac{\partial [s_j^L]_c}{\partial [w_{jk}^L]_i} = \begin{cases} [x_k^{L-1}]_{c+2}, & \text{dacă } c+1 = i \\ -[x_k^{L-1}]_{c+1}, & \text{dacă } c+2 = i \\ 0, & \text{dacă } c = i \end{cases},$$

de unde relaţia (5.1.4) se poate scrie sub forma

$$\frac{\partial E}{\partial [w_{jk}^L]_i} = \frac{\partial E}{\partial [s_j^L]_{i-1}} [x_k^{L-1}]_{i+1} - \frac{\partial E}{\partial [s_j^L]_{i-2}} [x_k^{L-1}]_{i-1}, \forall 1 \leq i \leq 3,$$

sau, ţinând seama de definiţia produsului vectorial,

$$\frac{\partial E}{\partial [w_{jk}^L]_i} = \left[ \frac{\partial E}{\partial s_j^L} \times x_k^{L-1} \right]_i, \forall 1 \leq i \leq 3. \tag{5.1.5}$$

Mai departe, cu notaţia $\delta_j^L := \frac{\partial E}{\partial s_j^L}$, avem din regula înlănţuirii că

$$[\delta_j^L]_c = \frac{\partial E}{\partial [s_j^L]_c} = \sum_{d=1}^{3} \frac{\partial E}{\partial [y_j^L]_d} \frac{\partial [y_j^L]_d}{\partial [s_j^L]_c},$$

$\forall 1 \leq c \leq 3$, sau, ţinând seama de (5.1.3) şi de expresia pentru funcţia de eroare dată în (5.1.1), că

$$\begin{aligned}[\delta_j^L]_c &= \sum_{d=1}^{3}([y_j^L]_d - [t_j]_d)\frac{\partial [G^L(s_j^L)]_d}{\partial [s_j^L]_c} \\ &= ([y_j^L]_c - [t_j]_c)\frac{\partial [G^L(s_j^L)]_c}{\partial [s_j^L]_c},\end{aligned}$$

$\forall 1 \leq c \leq 3$, deoarece $[G^L(s_j^L)]_d$ depinde de $[s_j^L]_c$ doar pentru $d = c$, ceea ce înseamnă că $\frac{\partial [G^L(s_j^L)]_d}{\partial [s_j^L]_c} = 0, \forall d \neq c$. Dacă notăm cu $\odot$ înmulţirea pe componente a doi vectori, relaţia de mai sus se scrie compactat

$$\delta_j^L = (y_j^L - t_j) \odot \frac{\partial G^L(s_j^L)}{\partial s_j^L}, \tag{5.1.6}$$

unde $\frac{\partial G^L(s_j^L)}{\partial s_j^L}$ reprezintă vectorul derivatei pe componente a funcţiei de activare $G^L$. De exemplu, dacă $S = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \in \mathcal{V}_3(\mathbb{R})$, atunci

$$\frac{\partial G(S)}{\partial S} = \begin{pmatrix} \operatorname{sech}^2 a_1 \\ \operatorname{sech}^2 a_2 \\ \operatorname{sech}^2 a_3 \end{pmatrix},$$

cu funcţia $G$ definită ca în exemplul de mai sus. În final, din ecuaţia (5.1.5), obţinem expresia pentru actualizarea dorită:

$$\Delta w_{jk}^L(t) = -\varepsilon \delta_j^L \times x_k^{L-1},$$

unde vectorul $\delta_j^L \in \mathcal{V}_3(\mathbb{R})$ este dat de relaţia (5.1.6).

Acum, trecem la actualizarea pentru o pondere arbitrară $w_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, aceasta se scrie

$$\Delta w_{jk}^l(t) = -\varepsilon \left( \frac{\partial E}{\partial [w_{jk}^l]_1}(t), \frac{\partial E}{\partial [w_{jk}^l]_2}(t), \frac{\partial E}{\partial [w_{jk}^l]_3}(t) \right)^T,$$

iar apoi, din regula înlănţuirii, rezultă

$$\frac{\partial E}{\partial [w_{jk}^l]_i} = \sum_{c=1}^{3} \frac{\partial E}{\partial [s_j^l]_c} \frac{\partial [s_j^l]_c}{\partial [w_{jk}^l]_i}, \forall 1 \leq i \leq 3. \tag{5.1.7}$$

Aplicând din nou această regulă, obţinem că

$$\frac{\partial E}{\partial [s_j^l]_c} = \sum_r \sum_{d=1}^{3} \frac{\partial E}{\partial [s_r^{l+1}]_d} \frac{\partial [s_r^{l+1}]_d}{\partial [s_j^l]_c}, \forall 1 \leq c \leq 3, \tag{5.1.8}$$

unde suma se ia după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite legături. Mai apoi

$$\frac{\partial [s_r^{l+1}]_d}{\partial [s_j^l]_c} = \sum_{e=1}^{3} \frac{\partial [s_r^{l+1}]_d}{\partial [y_j^l]_e} \frac{\partial [y_j^l]_e}{\partial [s_j^l]_c},$$

$\forall 1 \leq c, d \leq 3$. Din nou din (5.1.2), avem

$$\frac{\partial [s_r^{l+1}]_d}{\partial [y_j^l]_e} = \begin{cases} [w_{rj}^{l+1}]_{d+1}, & \text{dacă } d+2 = e \\ -[w_{rj}^{l+1}]_{d+2}, & \text{dacă } d+1 = e \\ 0, & \text{dacă } d = e \end{cases},$$

şi apoi

$$\frac{\partial [s_r^{l+1}]_d}{\partial [s_j^l]_c} = [w_{rj}^{l+1}]_{d+1} \frac{\partial [G^l(s_j^l)]_{d+2}}{\partial [s_j^l]_c} - [w_{rj}^{l+1}]_{d+2} \frac{\partial [G^l(s_j^l)]_{d+1}}{\partial [s_j^l]_c},$$

$\forall 1 \leq c, d \leq 3$, şi revenind înapoi în ecuaţia (5.1.8), obţinem succesiv

$$\begin{aligned} \frac{\partial E}{\partial [s_j^l]_c} &= \sum_r \sum_{d=1}^{3} \frac{\partial E}{\partial [s_r^{l+1}]_d} \left( [w_{rj}^{l+1}]_{d+1} \frac{\partial [G^l(s_j^l)]_{d+2}}{\partial [s_j^l]_c} - [w_{rj}^{l+1}]_{d+2} \frac{\partial [G^l(s_j^l)]_{d+1}}{\partial [s_j^l]_c} \right) \\ &= \sum_r \sum_{d=1}^{3} [w_{rj}^{l+1}]_{d+1} \frac{\partial E}{\partial [s_r^{l+1}]_d} \frac{\partial [G^l(s_j^l)]_{d+2}}{\partial [s_j^l]_c} - [w_{rj}^{l+1}]_{d+2} \frac{\partial E}{\partial [s_r^{l+1}]_d} \frac{\partial [G^l(s_j^l)]_{d+1}}{\partial [s_j^l]_c} \\ &= \sum_r \left( [w_{rj}^{l+1}]_{c-1} \frac{\partial E}{\partial [s_r^{l+1}]_{c-2}} - [w_{rj}^{l+1}]_{c+1} \frac{\partial E}{\partial [s_r^{l+1}]_{c-1}} \right) \frac{\partial [G^l(s_j^l)]_c}{\partial [s_j^l]_c} \\ &= \sum_r [w_{rj}^{l+1} \times \delta_r^{l+1}]_c \frac{\partial [G^l(s_j^l)]_c}{\partial [s_j^l]_c}, \end{aligned}$$

$\forall 1 \leq c \leq 3$, unde am ţinut cont de faptul că $\frac{\partial [G^l(s_j^l)]_e}{\partial [s_j^l]_c} = 0, \forall e \neq c$ şi am făcut notaţia $\delta_j^l := \frac{\partial E}{\partial s_j^l}$. Acum, relaţia de mai sus se poate scrie sub forma

$$\delta_j^l = \left( \sum_r w_{rj}^{l+1} \times \delta_r^{l+1} \right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, \tag{5.1.9}$$

unde $\odot$ este notaţia pentru înmulţirea pe componente a doi vectori.

Apoi, ţinând cont că

$$\frac{\partial [s_j^l]_c}{\partial [w_{jk}^l]_i} = \begin{cases} [x_k^{l-1}]_{c+2}, & \text{dacă } c+1 = i \\ -[x_k^{l-1}]_{c+1}, & \text{dacă } c+2 = i \\ 0, & \text{dacă } c = i \end{cases},$$

relaţia (5.1.7) devine

$$\frac{\partial E}{\partial [w_{jk}^l]_i} = \frac{\partial E}{\partial [s_j^l]_{i-1}} [x_k^{l-1}]_{i+1} - \frac{\partial E}{\partial [s_j^l]_{i-2}} [x_k^{l-1}]_{i-1}, \forall 1 \leq i \leq 3,$$

sau, echivalent,

$$\frac{\partial E}{\partial [w_{jk}^l]_i} = \left[ \delta_j^l \times x_k^{l-1} \right]_i, \forall 1 \leq i \leq 3,$$

unde $\delta_j^l := \frac{\partial E}{\partial s_j^l}$, ca mai sus.

În final am obţinut o formulă de calcul pentru actualizarea ponderii $w_{jk}^l$ asemănătoare cu cea din primul caz tratat, şi anume

$$\Delta w_{jk}^l(t) = -\varepsilon \delta_j^l \times x_k^{l-1}.$$

Avem, prin urmare, următoarea formulă pentru actualizarea ponderii $w_{jk}^l$ în cazul metodei gradient:

$$\Delta w_{jk}^l(t) = -\varepsilon \delta_j^l \times x_k^{l-1}, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left(\sum_r w_{rj}^{l+1} \times \delta_r^{l+1}\right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l \leq L-1 \\ (y_j^l - t_j) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l = L \end{cases}.$$

## 5.2 Reţele neuronale cu valori vectori $n$-dimensionali

Dintre cele două tipuri de reţele cu valori vectoriale discutate în secţiunea anterioară, cel de-al doilea poate fi generalizat la vectori $n$-dimensionali, rezultând astfel *reţele neuronale $n$-dimensionale* (pentru definiţia neuronului $n$-dimensional, a se vedea [200, 202]). Aceste reţele au intrări şi ieşiri vectori $n$-dimensionali, dar ponderile şi bias-urile sunt matrici ortogonale. În cele două referinţe citate, neuronul $n$-dimensional a fost folosit pentru rezolvarea cu succes a problemei parităţii pe $n$ biţi. Teoria reţelelor neuronale $n$-dimensionale este încă la început, singurele articole din domeniu fiind cele specificate mai sus. Există însă motive de speranţă că aceste reţele vor putea fi utilizate cu succes în rezolvarea unor probleme $n$-dimensionale la care reţelele clasice au eşuat sau au avut performanţe slabe.

Considerăm spaţiul vectorial de dimensiune $n$ al vectorilor cu valori reale, $\mathbb{R}^n$, pe care îl vom nota în continuare $\mathcal{V}_n(\mathbb{R}) := \mathbb{R}^n$.

În cele ce urmează vom introduce reţele neuronale cu valori din $\mathcal{V}_n(\mathbb{R})$, dar ale căror ponderi sunt matrici pătratice oarecare, şi nu matrici pătratice ortogonale, ca în cazul neuronului $n$-dimensional. Să presupunem că avem o reţea neuronală cu valori vectoriale de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile intermediare $\{2, \ldots, L-1\}$ sunt straturi ascunse. Funcţia de eroare $E : \mathcal{M}_n(\mathbb{R})^N \to \mathbb{R}$ a acestei reţele este definită prin

$$E(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{c} ||y_i^L - t_i||^2, \tag{5.2.1}$$

unde $||y||$ reprezintă norma în $\mathcal{V}_n(\mathbb{R})$ a lui $y$. $\mathbf{y}^L = (y_i^L)_{1 \leq i \leq c} \in \mathcal{V}_n(\mathbb{R})^c$ reprezintă *ieşirile* reţelei, $\mathbf{t} = (t_i)_{1 \leq i \leq c} \in \mathcal{V}_n(\mathbb{R})^c$ reprezintă *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{W} \in \mathcal{M}_n(\mathbb{R})^N$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei, care sunt matrici pătratice de ordinul $n$.

Notând cu $W_{jk}^l \in \mathcal{M}_n(\mathbb{R})$ ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l \in \{2, \ldots, L\}$, se poate defini *pasul de actualizare* al ponderii $W_{jk}^l$ în epoca $t$ ca fiind

$$\Delta W_{jk}^l(t) = W_{jk}^l(t+1) - W_{jk}^l(t).$$

*Regula de actualizare* pentru ponderea $W_{jk}^l$, în cadrul *metodei gradient*, este

$$\Delta W_{jk}^l(t) = -\varepsilon \left( \frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t) \right)_{1 \leq a,b \leq n},$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi $\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t)$ reprezintă derivata parţială a funcţiei $E$ în raport cu fiecare componentă $[W_{jk}^l]_{ab}$ a matricii $W_{jk}^l$, cu $1 \leq a, b \leq n$.

Scris sub formă vectorială, pasul de actualizare devine

$$\Delta \mathbf{W}(t) = \mathbf{W}(t+1) - \mathbf{W}(t),$$

unde $\mathbf{W} \in \mathcal{M}_n(\mathbb{R})^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, care a fost definit mai sus. Regula de actualizare se poate scrie, în formă vectorială, pentru metoda gradient, astfel:

$$\Delta \mathbf{W}(t) = -\varepsilon \nabla E(t),$$

unde $\nabla E(t) := \nabla E(\mathbf{W}(t)) \in \mathcal{M}_n(\mathbb{R})^N$, reprezintă gradientul funcţiei $E$, ale cărui componente

$$\left( \frac{\partial E}{\partial [W_{jk}^l]_{ab}}(\mathbf{W}(t)) \right)_{1 \leq a,b \leq n}$$

au fost prescurtate mai sus în forma

$$\left( \frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t) \right)_{1 \leq a,b \leq n}.$$

Prin urmare, a minimiza funcţia de eroare $E$ înseamnă a calcula gradientul $\nabla E$, adică derivatele parţiale de forma $\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t)$, cu $1 \leq a, b \leq n$.

Se impun următoarele notaţii

$$s_j^l = \sum_k W_{jk}^l x_k^{l-1}, \tag{5.2.2}$$

$$y_j^l = G^l(s_j^l), \tag{5.2.3}$$

unde (5.2.2) ne arată că înmulţirea de la reţelele neuronale cu valori reale este înlocuită de înmulţirea între o matrice şi un vector, $G^l$ este *funcţia de activare* a stratului $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1 \leq k \leq d} \in \mathcal{V}_n(\mathbb{R})^d$ sunt intrările reţelei, şi $x_k^l = y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$.

Funcţia de activare este considerată a fi definită pe componente. De pildă, pentru vectorul $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \in \mathcal{V}_3(\mathbb{R})$, un exemplu de funcţie de activare este funcţia tangentă hiperbolică pe componente definită prin

$$G\left( \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \right) = \begin{pmatrix} \tanh a_1 \\ \tanh a_2 \\ \tanh a_3 \end{pmatrix}.$$

Mai întâi, vom calcula actualizarea pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta W_{jk}^L(t) = -\varepsilon \left( \frac{\partial E}{\partial [W_{jk}^L]_{ab}} \right)_{1 \leq a,b \leq n}.$$

Folosind regula înlănţuirii, putem scrie setul de relaţii:

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{c=1}^{n} \frac{\partial E}{\partial [s_j^L]_c} \frac{\partial [s_j^L]_c}{\partial [W_{jk}^L]_{ab}}, \tag{5.2.4}$$

pentru orice $1 \leq a, b \leq n$.

Ne vom ocupa mai întâi de $\frac{\partial [s_j^L]_c}{\partial [W_{jk}^L]_{ab}}$. Pentru calculul acestei derivate, avem nevoie de o formulă explicită pentru $[s_j^L]_c$, care se calculează uşor din (5.2.2):

$$[s_j^L]_c = \sum_k \sum_{m=1}^{n} [W_{jk}^L]_{cm} [x_k^{L-1}]_m, \forall 1 \leq c \leq n.$$

Acum, se vede imediat că

$$\frac{\partial [s_j^L]_c}{\partial [W_{jk}^L]_{ab}} = \begin{cases} [x_k^{L-1}]_b, & \text{dacă } c = a \\ 0, & \text{dacă } c \neq a \end{cases},$$

de unde relaţia (5.2.4) se poate scrie sub forma

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \frac{\partial E}{\partial [s_j^L]_a} [x_k^{L-1}]_b,$$

sau, echivalent

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \frac{\partial E}{\partial [s_j^L]_a} [(x_k^{L-1})^T]_b, \forall 1 \leq a, b \leq n. \tag{5.2.5}$$

Mai departe, notând $\delta_j^L := \frac{\partial E}{\partial s_j^L}$, avem din regula înlănţuirii că

$$[\delta_j^L]_c = \frac{\partial E}{\partial [s_j^L]_c} = \sum_{d=1}^{n} \frac{\partial E}{\partial [y_j^L]_d} \frac{\partial [y_j^L]_d}{\partial [s_j^L]_c},$$

$\forall 1 \leq c \leq n$, sau, ţinând seama de notaţia (5.2.3) şi de expresia pentru funcţia de eroare (5.2.1), că

$$\begin{aligned} [\delta_j^L]_c &= \sum_{d=1}^{n} ([y_j^L]_d - [t_j]_d) \frac{\partial [G^L(s_j^L)]_d}{\partial [s_j^L]_c} \\ &= ([y_j^L]_c - [t_j]_c) \frac{\partial [G^L(s_j^L)]_c}{\partial [s_j^L]_c}, \end{aligned}$$

$\forall 1 \leq c \leq n$, deoarece $[G^L(s_j^L)]_d$ depinde de $[s_j^L]_c$ doar pentru $d = c$, ceea ce înseamnă că $\frac{\partial [G^L(s_j^L)]_d}{\partial [s_j^L]_c} = 0, \forall d \neq c$. Notând cu $\odot$ înmulţirea pe componente a doi vectori, relaţia de mai sus ne dă

$$\delta_j^L = (y_j^L - t_j) \odot \frac{\partial G^L(s_j^L)}{\partial s_j^L}, \tag{5.2.6}$$

unde $\frac{\partial G^L(s_j^L)}{\partial s_j^L}$ reprezintă vectorul derivatei pe componente a funcţiei de activare $G^L$. De exemplu, dacă $S = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \in \mathcal{V}_3(\mathbb{R})$, atunci

$$\frac{\partial G(S)}{\partial S} = \begin{pmatrix} \operatorname{sech}^2 a_1 \\ \operatorname{sech}^2 a_2 \\ \operatorname{sech}^2 a_3 \end{pmatrix},$$

cu funcţia $G$ definită ca în exemplul de mai sus. În final, din ecuaţia (5.2.5), obţinem expresia pentru actualizarea dorită:

$$\Delta W_{jk}^L(t) = -\varepsilon \delta_j^L (x_k^{L-1})^T,$$

unde vectorul $\delta_j^L \in \mathcal{V}_n(\mathbb{R})$ este dat de relaţia (5.2.6).

Acum, trecem la actualizarea pentru o pondere arbitrară $W_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, putem scrie că

$$\Delta W_{jk}^l(t) = -\varepsilon \left( \frac{\partial E}{\partial [W_{jk}^l]_{ab}} \right)_{1 \le a,b \le n},$$

iar apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial [W_{jk}^l]_{ab}} = \sum_{c=1}^{n} \frac{\partial E}{\partial [s_j^l]_c} \frac{\partial [s_j^l]_c}{\partial [W_{jk}^l]_{ab}}, \forall 1 \le a, b \le n. \tag{5.2.7}$$

Tot cu regula înlănţuirii, obţinem că

$$\frac{\partial E}{\partial [s_j^l]_c} = \sum_{r} \sum_{d=1}^{n} \frac{\partial E}{\partial [s_r^{l+1}]_d} \frac{\partial [s_r^{l+1}]_d}{\partial [s_j^l]_c}, \forall 1 \le c \le n, \tag{5.2.8}$$

unde suma se ia după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite legături. Mai apoi

$$\frac{\partial [s_r^{l+1}]_d}{\partial [s_j^l]_c} = \sum_{e=1}^{n} \frac{\partial [s_r^{l+1}]_d}{\partial [y_j^l]_e} \frac{\partial [y_j^l]_e}{\partial [s_j^l]_c},$$

$\forall 1 \le c, d \le n$. Din nou din (5.2.2), avem

$$\frac{\partial [s_r^{l+1}]_d}{\partial [y_j^l]_e} = [W_{rj}^{l+1}]_{de},$$

şi apoi

$$\frac{\partial [s_r^{l+1}]_d}{\partial [s_j^l]_c} = \sum_{e=1}^{n} [W_{rj}^{l+1}]_{de} \frac{\partial [G^l(s_j^l)]_e}{\partial [s_j^l]_c},$$

$\forall 1 \le c, d \le n$, şi revenind înapoi în (5.2.8), rezultă succesiv

$$\begin{aligned}
\frac{\partial E}{\partial [s_j^l]_c} &= \sum_r \sum_{d=1}^n \frac{\partial E}{\partial [s_r^{l+1}]_d} \sum_{e=1}^n [W_{rj}^{l+1}]_{de} \frac{\partial [G^l(s_j^l)]_e}{\partial [s_j^l]_c} \\
&= \sum_r \sum_{1 \le d,e \le n} [W_{rj}^{l+1}]_{de} \frac{\partial E}{\partial [s_r^{l+1}]_d} \frac{\partial [G^l(s_j^l)]_e}{\partial [s_j^l]_c} \\
&= \sum_r \left( \sum_{d=1}^n [(W_{rj}^{l+1})^T]_{cd} \frac{\partial E}{\partial [s_r^{l+1}]_d} \right) \frac{\partial [G^l(s_j^l)]_c}{\partial [s_j^l]_c} \\
&= \sum_r [(W_{rj}^{l+1})^T \delta_r^{l+1}]_c \frac{\partial [G^l(s_j^l)]_c}{\partial [s_j^l]_c},
\end{aligned}$$

$\forall 1 \le c \le n$, unde din nou am ţinut cont de faptul că $\frac{\partial [G^l(s_j^l)]_e}{\partial [s_j^l]_c} = 0, \forall e \ne c$ şi am făcut notaţia $\delta_j^l := \frac{\partial E}{\partial s_j^l}$. Acum, putem scrie relaţia de mai sus sub forma

$$\delta_j^l = \left( \sum_r (W_{rj}^{l+1})^T \delta_r^{l+1} \right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, \tag{5.2.9}$$

unde prin $\odot$ am notat înmulţirea pe componente a doi vectori.

Ţinând acum cont că

$$\frac{\partial [s_j^l]_c}{\partial [W_{jk}^l]_{ab}} = \begin{cases} [x_k^{l-1}]_b, & \text{dacă } c = a \\ 0, & \text{dacă } c \ne a \end{cases},$$

relaţia (5.2.7) devine

$$\begin{aligned}
\frac{\partial E}{\partial [W_{jk}^l]_{ab}} &= \frac{\partial E}{\partial [s_j^l]_a} [x_k^{l-1}]_b \\
&= \frac{\partial E}{\partial [s_j^l]_a} [(x_k^{l-1})^T]_b \\
&= [\delta_j^l (X_k^{l-1})^T]_{ab}, \forall 1 \le a, b \le n,
\end{aligned}$$

cu notaţia $\delta_j^l := \frac{\partial E}{\partial s_j^l}$, ca mai sus.

În final, am obţinut o formulă de calcul pentru actualizarea ponderii $W_{jk}^l$ asemănătoare cu cea din primul caz tratat, şi anume

$$\Delta W_{jk}^l(t) = -\varepsilon \delta_j^l (x_k^{l-1})^T.$$

Prin urmare, avem următoarea formulă pentru actualizarea ponderii $W_{jk}^l$ la metoda gradient:

$$\Delta W_{jk}^l(t) = -\varepsilon \delta_j^l (x_k^{l-1})^T, \forall l \in \{2, \ldots, L\},$$

unde

$$\delta_j^l = \begin{cases} \left(\sum_r (W_{rj}^{l+1})^T \delta_r^{l+1}\right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l \le L-1 \\ (y_j^l - t_j) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l = L \end{cases}.$$

## 5.3 Reţele neuronale cu valori matrici pătratice

Faptul că numerele complexe, hiperbolice, cuaternionice şi Clifford pot fi scrise sub formă matricială (spre exemplu, un număr complex $a+ib$, $i=\sqrt{-1}$, poate fi scris sub forma $\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$), a condus la ideea naturală de a defini reţele neuronale multidimensionale ale căror intrări, ieşiri, ponderi şi bias-uri să fie matrici. *Reţele neuronale cu valori matriciale* sunt deci o generalizare a reţelelor neuronale studiate anterior, deoarece fiecare dintre algebrele pe care au fost definite aceste reţele neuronale, poate fi văzută ca o subalgebră a algebrei matricilor pătratice, cu operaţiile naturale de adunare şi înmulţire a matricilor. Datorită gradului lor de generalitate, aceste reţele neuronale sunt menite sa aibă multe aplicaţii în viitor la rezolvarea acelor probleme la care reţelele neuronale cu valori reale au eşuat sau au avut performanţe slabe.

Considerăm algebra matricilor pătratice de dimensiune $n$, $\mathcal{M}_n(\mathbb{R})$, cu operaţiile naturale de adunare şi înmulţire a matricilor.

În cele ce urmează vom introduce reţele neuronale cu valori în această algebră, ca în articolul nostru [219]. Să presupunem că avem o reţea neuronală cu valori matriciale de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile numerotate $\{2,\ldots,L-1\}$ sunt straturi ascunse. Funcţia de eroare $E:\mathcal{M}_n(\mathbb{R})^N\to\mathbb{R}$ a acestei reţele este dată prin

$$E(\mathbf{W})=\frac{1}{2}\sum_{i=1}^{c}||Y_i^L-T_i||^2, \tag{5.3.1}$$

unde $||Y||$ este norma Frobenius a matricii $Y$ definită prin $||Y||:=\sqrt{\mathrm{Tr}(YY^T)}$, şi $\mathrm{Tr}(Y)$ este urma matricii $Y$. $\mathbf{Y}^L=(Y_i^L)_{1\leq i\leq c}\in\mathcal{M}_n(\mathbb{R})^c$ reprezintă *ieşirile* reţelei, $\mathbf{T}=(T_i)_{1\leq i\leq c}\in\mathcal{M}_n(\mathbb{R})^c$ reprezintă *ieşirile dorite* (*target*-urile) ale reţelei, iar $\mathbf{W}\in\mathcal{M}_n(\mathbb{R})^N$ este vectorul tuturor celor $N$ *ponderi* şi *bias*-uri ale reţelei.

Dacă $W_{jk}^l\in\mathcal{M}_n(\mathbb{R})$ reprezintă ponderea care leagă neuronul $j$ din stratul $l$ de neuronul $k$ din stratul $l-1$, pentru orice $l\in\{2,\ldots,L\}$, atunci putem defini *pasul de actualizare* al ponderii $W_{jk}^l$ în epoca $t$ ca fiind

$$\Delta W_{jk}^l(t)=W_{jk}^l(t+1)-W_{jk}^l(t).$$

Cu această notaţie, pentru *metoda gradient*, *regula de actualizare* pentru ponderea $W_{jk}^l$ este

$$\Delta W_{jk}^l(t)=-\varepsilon\left(\frac{\partial E}{\partial[W_{jk}^l]_{ab}}(t)\right)_{1\leq a,b\leq n},$$

unde $\varepsilon$ este un număr real care reprezintă *rata de învăţare*, şi am notat cu $\frac{\partial E}{\partial[W_{jk}^l]_{ab}}(t)$ derivata parţială a funcţiei $E$ în funcţie de fiecare componentă $[W_{jk}^l]_{ab}$ a matricii $W_{jk}^l$, cu $1\leq a,b\leq n$.

Sub formă vectorială, pasul de actualizare devine

$$\Delta\mathbf{W}(t)=\mathbf{W}(t+1)-\mathbf{W}(t),$$

unde $\mathbf{W}\in\mathcal{M}_n(\mathbb{R})^N$ este vectorul celor $N$ ponderi şi bias-uri ale reţelei, definit mai sus. Regula de actualizare pentru metoda gradient se poate scrie acum

$$\Delta\mathbf{W}(t)=-\varepsilon\nabla E(t),$$

unde $\nabla E(t) := \nabla E(\mathbf{W}(t)) \in \mathcal{M}_n(\mathbb{R})^N$, este gradientul funcţiei $E$, ale cărui componente

$$\left(\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(\mathbf{W}(t))\right)_{1\leq a,b\leq n},$$

au fost scrise prescurtat mai sus sub forma

$$\left(\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t)\right)_{1\leq a,b\leq n}.$$

Prin urmare, pentru metoda gradient, avem nevoie să calculăm pe $\nabla E$, adică derivatele parţiale de forma $\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t)$, cu $1 \leq a, b \leq n$.

Începem prin a face următoarele notaţii

$$S_j^l = \sum_k W_{jk}^l X_k^{l-1}, \tag{5.3.2}$$

$$Y_j^l = G^l(S_j^l), \tag{5.3.3}$$

unde (5.3.2) denotă faptul că înmulţirea de la reţelele neuronale cu valori reale este înlocuită de înmulţirea între matrici, $G^l$ este *funcţia de activare* a stratului $l \in \{2, \ldots, L\}$, $\mathbf{X}^1 = (X_k^1)_{1\leq k\leq d} \in \mathcal{M}_n(\mathbb{R})^d$ sunt intrările reţelei, şi avem că $X_k^l = Y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$.

Funcţia de activare este considerată a fi definită pe componente. Un exemplu de funcţie de activare este funcţia tangentă hiperbolică pe componente definită, pentru matricea $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in \mathcal{M}_3(\mathbb{R})$, prin

$$G\left(\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\right) = \begin{pmatrix} \tanh a_{11} & \tanh a_{12} & \tanh a_{13} \\ \tanh a_{21} & \tanh a_{22} & \tanh a_{23} \\ \tanh a_{31} & \tanh a_{32} & \tanh a_{33} \end{pmatrix}.$$

Vom începe cu actualizarea pentru ponderile dintre penultimul strat ascuns $L-1$ şi stratul de ieşire $L$, care se poate scrie

$$\Delta W_{jk}^L(t) = -\varepsilon \left(\frac{\partial E}{\partial [W_{jk}^L]_{ab}}\right)_{1\leq a,b\leq n}.$$

Regula înlănţuirii, ne permite să scriem următorul set de relaţii:

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{1\leq c,d\leq n} \frac{\partial E}{\partial [S_j^L]_{cd}} \frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}}, \tag{5.3.4}$$

pentru orice $1 \leq a, b \leq n$.

Vom începe cu derivata parţială $\frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}}$. Pentru calculul acestei derivate, este nevoie de o formulă explicită pentru $[S_j^L]_{cd}$, care se poate deduce din (5.3.2):

$$[S_j^L]_{cd} = \sum_k \sum_{m=1}^n [W_{jk}^L]_{cm} [X_k^{L-1}]_{md}, \forall 1 \leq c, d \leq n.$$

Acum, se vede uşor că

$$\frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}} = \begin{cases} [X_k^{L-1}]_{bd}, & \text{dacă } c = a \\ 0, & \text{dacă } c \neq a \end{cases},$$

de unde relaţia (5.3.4) se poate scrie astfel:

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^L]_{ad}} [X_k^{L-1}]_{bd},$$

sau, echivalent

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{1 \leq d \leq n} \frac{\partial E}{\partial [S_j^L]_{ad}} [(X_k^{L-1})^T]_{db}. \tag{5.3.5}$$

Mai departe, cu notaţia $\Delta_j^L := \frac{\partial E}{\partial S_j^L}$, avem din regula înlănţuirii că

$$[\Delta_j^L]_{cd} = \frac{\partial E}{\partial [S_j^L]_{cd}} = \sum_{1 \leq e,f \leq n} \frac{\partial E}{\partial [Y_j^L]_{ef}} \frac{\partial [Y_j^L]_{ef}}{\partial [S_j^L]_{cd}},$$

$\forall 1 \leq c, d \leq n$, sau, ţinând seama de (5.3.3) şi de expresia pentru funcţia de eroare (5.3.1), că

$$\begin{aligned} [\Delta_j^L]_{cd} &= \sum_{1 \leq e,f \leq n} ([Y_j^L]_{ef} - [T_j]_{ef}) \frac{\partial [G^L(S_j^L)]_{ef}}{\partial [S_j^L]_{cd}} \\ &= ([Y_j^L]_{cd} - [T_j]_{cd}) \frac{\partial [G^L(S_j^L)]_{cd}}{\partial [S_j^L]_{cd}}, \end{aligned}$$

$\forall 1 \leq c, d \leq n$, deoarece $[G^L(S_j^L)]_{ef}$ depinde de $[S_j^L]_{cd}$ doar pentru $e = c$ şi $f = d$, ceea ce înseamnă că $\frac{\partial [G^L(S_j^L)]_{ef}}{\partial [S_j^L]_{cd}} = 0, \forall (e,f) \neq (c,d)$. Dacă notăm cu $\odot$ înmulţirea pe componente a două matrici, relaţia de mai sus ne dă

$$\Delta_j^L = (Y_j^L - T_j) \odot \frac{\partial G^L(S_j^L)}{\partial S_j^L}, \tag{5.3.6}$$

unde $\frac{\partial G^L(S_j^L)}{\partial S_j^L}$ reprezintă matricea derivatei pe componente a funcţiei de activare $G^L$. De exemplu, dacă $S = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in \mathcal{M}_3(\mathbb{R})$, atunci

$$\frac{\partial G(S)}{\partial S} = \begin{pmatrix} \operatorname{sech}^2 a_{11} & \operatorname{sech}^2 a_{12} & \operatorname{sech}^2 a_{13} \\ \operatorname{sech}^2 a_{21} & \operatorname{sech}^2 a_{22} & \operatorname{sech}^2 a_{23} \\ \operatorname{sech}^2 a_{31} & \operatorname{sech}^2 a_{32} & \operatorname{sech}^2 a_{33} \end{pmatrix},$$

cu funcţia $G$ definită ca în exemplul de mai sus. În fine, din ecuaţia (5.3.5), obţinem actualizarea dorită:

$$\Delta W_{jk}^L(t) = -\varepsilon \Delta_j^L (X_k^{L-1})^T,$$

unde matricea $\Delta_j^L \in \mathcal{M}_n(\mathbb{R})$ este dată de relaţia (5.3.6).

Acum, vom calcula actualizarea pentru o pondere arbitrară $W_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, această pondere este

$$\Delta W_{jk}^l(t) = -\varepsilon \left( \frac{\partial E}{\partial [W_{jk}^l]_{ab}} \right)_{1 \leq a,b \leq n},$$

iar apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial [W_{jk}^l]_{ab}} = \sum_{1 \leq c,d \leq n} \frac{\partial E}{\partial [S_j^l]_{cd}} \frac{\partial [S_j^l]_{cd}}{\partial [W_{jk}^l]_{ab}}, \forall 1 \leq a, b \leq n. \tag{5.3.7}$$

Aplicând din nou regula înlănţuirii, obţinem că

$$\frac{\partial E}{\partial [S_j^l]_{cd}} = \sum_r \sum_{1 \leq e,f \leq n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}}, \forall 1 \leq c, d \leq n, \tag{5.3.8}$$

unde suma se ia după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite legături. Apoi avem

$$\frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}} = \sum_{1 \leq g,h \leq n} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [Y_j^l]_{gh}} \frac{\partial [Y_j^l]_{gh}}{\partial [S_j^l]_{cd}},$$

$\forall 1 \leq c, d \leq n$, $\forall 1 \leq e, f \leq n$. Din nou din (5.3.2), obţinem

$$\frac{\partial [S_r^{l+1}]_{ef}}{\partial [Y_j^l]_{gh}} = \begin{cases} [W_{rj}^{l+1}]_{eg}, & \text{dacă } f = h \\ 0, & \text{dacă } f \neq h \end{cases},$$

şi apoi

$$\frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}} = \sum_{g=1}^n [W_{rj}^{l+1}]_{eg} \frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}},$$

$\forall 1 \leq c, d \leq n$, $\forall 1 \leq e, f \leq n$, şi revenind înapoi în ecuaţia (5.3.8), avem succesiv

$$\begin{aligned}
\frac{\partial E}{\partial [S_j^l]_{cd}} &= \sum_r \sum_{1 \leq e,f \leq n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \sum_{g=1}^n [W_{rj}^{l+1}]_{eg} \frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}} \\
&= \sum_r \sum_{1 \leq e,f,g \leq n} [W_{rj}^{l+1}]_{eg} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}} \\
&= \sum_r \left( \sum_{e=1}^n [(W_{rj}^{l+1})^T]_{ce} \frac{\partial E}{\partial [S_r^{l+1}]_{ed}} \right) \frac{\partial [G^l(S_j^l)]_{cd}}{\partial [S_j^l]_{cd}} \\
&= \sum_r [(W_{rj}^{l+1})^T \Delta_r^{l+1}]_{cd} \frac{\partial [G^l(S_j^l)]_{cd}}{\partial [S_j^l]_{cd}},
\end{aligned}$$

$\forall 1 \leq c, d \leq n$, unde din nou am ţinut cont de faptul că $\frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}} = 0, \forall (g,f) \neq (c,d)$. Acum, notând $\Delta_j^l := \frac{\partial E}{\partial S_j^l}$, putem scrie relaţia de mai sus sub forma

$$\Delta_j^l = \left( \sum_r (W_{rj}^{l+1})^T \Delta_r^{l+1} \right) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, \tag{5.3.9}$$

unde $\odot$ este înmulţirea pe componente a două matrici.

Ţinând cont că

$$\frac{\partial [S_j^l]_{cd}}{\partial [W_{jk}^l]_{ab}} = \begin{cases} [X_k^{l-1}]_{bd}, & \text{dacă } c = a \\ 0, & \text{dacă } c \neq a \end{cases},$$

relaţia (5.3.7) devine

$$\begin{aligned} \frac{\partial E}{\partial [W_{jk}^l]_{ab}} &= \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^l]_{ad}} [X_k^{l-1}]_{bd} \\ &= \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^l]_{ad}} [(X_k^{l-1})^T]_{db} \\ &= [\Delta_j^l (X_k^{l-1})^T]_{ab}, \forall 1 \leq a, b \leq n, \end{aligned}$$

unde am făcut notaţia $\Delta_j^l := \frac{\partial E}{\partial S_j^l}$, analog ca mai sus.

În final, a rezultat o formulă de calcul pentru actualizarea ponderii $W_{jk}^l$ asemănătoare cu cea din primul caz tratat, şi anume

$$\Delta W_{jk}^l(t) = -\varepsilon \Delta_j^l (X_k^{l-1})^T.$$

Pentru metoda gradient, avem, în concluzie, următoarea formulă de actualizare a ponderii $W_{jk}^l$:

$$\Delta W_{jk}^l(t) = -\varepsilon \Delta_j^l (X_k^{l-1})^T, \forall l \in \{2, \ldots, L\},$$

unde

$$\Delta_j^l = \begin{cases} \left( \sum_r (W_{rj}^{l+1})^T \Delta_r^{l+1} \right) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, & l \leq L-1 \\ (Y_j^l - T_j) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, & l = L \end{cases}.$$

## 5.4 Reţele neuronale cu valori matrici antisimetrice

O generalizare diferită a reţelelor neuronale cu valori reale în domenii multidimensionale sunt *reţelele neuronale* care au intrări, ieşiri, ponderi şi bias-uri dintr-o *algebră Lie*. Pentru că algebrele Lie pot avea orice dimensiune $n$, ele reprezintă de asemenea o alternativă la reţelele neuronale $N$-dimensionale discutate mai sus. Ţinând cont de faptul că definiţia lor vine din geometrie, şi că au numeroase aplicaţii în fizică şi inginerie (a se vedea [130, 74, 229]), şi de asemenea de faptul că au fost folosite cu succes în domeniul computer vision în ultimii ani (pentru o trecere în revistă, a se vedea [269], şi referinţele de acolo), am considerat o idee promiţătoare aceea de a defini reţele neuronale cu valori în algebre Lie.

Reţelele neuronale cu valori în algebre Lie ar putea avea aplicaţii interesante în computer vision şi toate celelalte domenii care au legătură cu transformările geometrice ale obiectelor tridimensionale, dar ar putea de asemenea să aibă performanţe mai bune pe anumite probleme $n$-dimensionale decât soluţiile disponibile în acest moment.

O *algebră Lie* este un spaţiu vectorial $\mathfrak{g}$ peste un corp $F$ împreună cu o operaţie $[\cdot,\cdot]:\mathfrak{g}\times\mathfrak{g}\to\mathfrak{g}$ numită *paranteză Lie*, care satisface următoarele axiome:

- Este biliniară: $[ax+by,z]=a[x,z]+b[y,z]$, $[x,ay+bz]=a[x,y]+b[x,z]$, $\forall a,b\in F$, $\forall x,y\in\mathfrak{g}$.
- Este antisimetrică: $[x,x]=0$, ceea ce implică $[x,y]=-[y,x]$, $\forall x,y\in\mathfrak{g}$.
- Satisface identitatea lui Jacobi: $[x,[y,z]]+[y,[z,x]]+[z,[x,y]]=0$, $\forall x,y,z\in\mathfrak{g}$.

Considerăm spaţiul vectorial $\mathfrak{so}(n)$ al matricilor pătratice antisimetrice de ordin $n$. Este uşor de verificat că operaţia dată prin

$$[A,B]:=AB-BA,\forall A,B\in\mathfrak{so}(n),$$

satisface axiomele de mai sus, ceea ce înseamnă că $\mathfrak{so}(n)$ este o algebră Lie, operaţia definită mai sus fiind paranteza sa Lie. Pentru uşurinţa expunerii, am decis să lucrăm doar cu această algebră Lie, dar generalizarea la o algebră Lie arbitrară este imediată.

În cele ce urmează vom introduce reţele neuronale cu valori în această algebră, ca în articolul nostru [218]. Să presupunem că avem o reţea neuronală cu valori din $\mathfrak{so}(n)$ de tip feedforward, total conectată, care are $L$ straturi, unde stratul $1$ este stratul de intrare, stratul $L$ este stratul de ieşire, iar straturile $\{2,\ldots,L-1\}$ sunt straturi ascunse. Funcţia de eroare $E:\mathfrak{so}(n)^N\to\mathbb{R}$ este dată prin următoarea expresie:

$$E(\mathbf{W})=\frac{1}{2}\sum_{i=1}^{c}||Y_i^L-T_i||^2, \tag{5.4.1}$$

unde $||Y||$ este norma Frobenius a matricii $Y$, definită prin $||Y||:=\sqrt{\mathrm{Tr}(YY^T)}$, şi $\mathrm{Tr}(Y)$ este urma matricii $Y$. $\mathbf{Y}^L=(Y_i^L)_{1\leq i\leq c}\in\mathfrak{so}(n)^c$ reprezintă *ieşirile* reţelei, $\mathbf{T}=(T_i)_{1\leq i\leq c}\in\mathfrak{so}(n)^c$ reprezintă *ieşirile dorite* ale reţelei, iar $\mathbf{W}\in\mathfrak{so}(n)^N$ este vectorul tuturor celor $N$ ponderi şi bias-uri ale reţelei, care este un vector ale cărui componente sunt matrici din $\mathfrak{so}(n)$.

Fie $W_{jk}^l\in\mathfrak{so}(n)$ ponderea care leagă neuronul $j$ din stratul $l$ cu neuronul $k$ din stratul $l-1$, pentru orice $l\in\{2,\ldots,L\}$. Putem defini *pasul de actualizare* al ponderii $W_{jk}^l$ în epoca $t$ ca fiind

$$\Delta W_{jk}^l(t)=W_{jk}^l(t+1)-W_{jk}^l(t).$$

Cu această notaţie, metoda gradient are următoarea *regulă de actualizare* pentru ponderea $W_{jk}^l\in\mathfrak{so}(n)$:

$$\Delta W_{jk}^l(t)=-\varepsilon\left(\frac{\partial E}{\partial[W_{jk}^l]_{ab}}(t)\right)_{1\leq a,b\leq n},$$

unde $\varepsilon$ este un număr real reprezentând *rata de învăţare*, şi am notat cu $\frac{\partial E}{\partial[W_{jk}^l]_{ab}}(t)$ derivata parţială a funcţiei de eroare $E$ în raport cu fiecare element $[W_{jk}^l]_{ab}$ al matricii $W_{jk}^l\in\mathfrak{so}(n)$, unde $1\leq a,b\leq n$. Astfel, pentru a minimiza funcţia $E$, trebuie să calculăm derivatele parţiale $\frac{\partial E}{\partial[W_{jk}^l]_{ab}}(t)$, cu $1\leq a,b\leq n$.

Acum, facem următoarele notaţii

$$S_j^l = \sum_k [W_{jk}^l, X_k^{l-1}], \tag{5.4.2}$$

$$Y_j^l = G^l(S_j^l), \tag{5.4.3}$$

unde (5.4.2) înseamnă că înmulţirea din cazul reţelelor cu valori reale este înlocuită de paranteza Lie, $G^l$ reprezintă funcţia de activare a stratului $l \in \{2, \ldots, L\}$, $\mathbf{X}^1 = (X_k^1)_{1 \leq k \leq d} \in \mathfrak{so}(n)^d$ sunt intrările reţelei, şi avem că $X_k^l = Y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$.

Funcţia de activare este considerată a fi definită pe componente. De exemplu, pentru matricea $\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix} \in \mathfrak{so}(3)$, un exemplu de funcţie de activare este funcţia tangentă hiperbolică pe componente definită prin

$$G\left(\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & \tanh x & \tanh y \\ -\tanh x & 0 & \tanh z \\ -\tanh y & -\tanh z & 0 \end{pmatrix}.$$

Vom calcula mai întâi actualizarea pentru ponderile dintre stratul $L-1$ şi stratul de ieşire $L$, i.e.

$$\Delta W_{jk}^L(t) = -\varepsilon \left(\frac{\partial E}{\partial [W_{jk}^L]_{ab}}\right)_{1 \leq a,b \leq n}.$$

Folosind din nou regula înlănţuirii, putem scrie următorul set de relaţii $\forall 1 \leq a, b \leq n$:

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{1 \leq c,d \leq n} \frac{\partial E}{\partial [S_j^L]_{cd}} \frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}}. \tag{5.4.4}$$

Ne vom ocupa mai întâi de $\frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}}$. Pentru a calcula această derivată, avem nevoie de o formulă explicită pentru $[S_j^L]_{cd}$, care poate fi uşor dedusă din (5.4.2):

$$[S_j^L]_{cd} = \sum_k \sum_{m=1}^n [W_{jk}^L]_{cm} [X_k^{L-1}]_{md} - [X_k^{L-1}]_{cm} [W_{jk}^L]_{md},$$

$\forall 1 \leq c, d \leq n$. Acum, se poate vedea uşor că

$$\frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}} = \begin{cases} [X_k^{L-1}]_{bd}, & \text{dacă } c = a,\ d \neq b \\ -[X_k^{L-1}]_{ca}, & \text{dacă } c \neq a,\ d = b \\ 0, & \text{altfel} \end{cases},$$

şi relaţia (5.4.4) poate fi scrisă sub forma

$$\begin{aligned} \frac{\partial E}{\partial [W_{jk}^L]_{ab}} &= \sum_{d=1}^n \frac{\partial E}{\partial [S_j^L]_{ad}} [X_k^{L-1}]_{bd} \\ &- \sum_{c=1}^n [X_k^{L-1}]_{ca} \frac{\partial E}{\partial [S_j^L]_{cb}}, \end{aligned}$$

sau, echivalent

$$\begin{aligned}\frac{\partial E}{\partial [W_{jk}^L]_{ab}} &= \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^L]_{ad}} [(X_k^{L-1})^T]_{db} \\ &- \sum_{c=1}^{n} [(X_k^{L-1})^T]_{ac} \frac{\partial E}{\partial [S_j^L]_{cb}}, \forall 1 \leq a, b \leq n. \end{aligned} \tag{5.4.5}$$

Mai departe, notând $\Delta_j^L := \frac{\partial E}{\partial S_j^L}$, avem din regula înlănţuirii că

$$[\Delta_j^L]_{cd} = \frac{\partial E}{\partial [S_j^L]_{cd}} = \sum_{1 \leq e,f \leq n} \frac{\partial E}{\partial [Y_j^L]_{ef}} \frac{\partial [Y_j^L]_{ef}}{\partial [S_j^L]_{cd}},$$

$\forall 1 \leq c, d \leq n$. Ţinând cont de notaţia (5.4.3), şi de expresia funcţiei de eroare dată în (5.4.1), avem că

$$\begin{aligned}[\Delta_j^L]_{cd} &= \sum_{1 \leq e,f \leq n} ([Y_j^L]_{ef} - [T_j]_{ef}) \frac{\partial [G^L(S_j^L)]_{ef}}{\partial [S_j^L]_{cd}} \\ &= ([Y_j^L]_{cd} - [T_j]_{cd}) \frac{\partial [G^L(S_j^L)]_{cd}}{\partial [S_j^L]_{cd}},\end{aligned}$$

$\forall 1 \leq c, d \leq n$, deoarece $[G^L(S_j^L)]_{ef}$ depinde de $[S_j^L]_{cd}$ doar pentru $e = c$ şi $f = d$, ceea ce înseamnă că $\frac{\partial [G^L(S_j^L)]_{ef}}{\partial [S_j^L]_{cd}} = 0, \forall (e, f) \neq (c, d)$. Dacă notăm cu $\odot$ înmulţirea pe componente a două matrici, relaţia de mai sus ne dă următoarea expresie pentru $\Delta_j^L$:

$$\Delta_j^L = (Y_j^L - T_j) \odot \frac{\partial G^L(S_j^L)}{\partial S_j^L}, \tag{5.4.6}$$

unde $\frac{\partial G^L(S_j^L)}{\partial S_j^L}$ reprezintă matricea derivatelor pe componente ale funcţiei de activare $G^L$. De exemplu, dacă $S = \begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix} \in \mathfrak{so}(3)$, atunci

$$\frac{\partial G(S)}{\partial S} = \begin{pmatrix} 0 & \operatorname{sech}^2 x & \operatorname{sech}^2 y \\ -\operatorname{sech}^2 x & 0 & \operatorname{sech}^2 z \\ -\operatorname{sech}^2 y & -\operatorname{sech}^2 z & 0 \end{pmatrix},$$

cu funcţia $G$ definită ca în exemplul de mai sus.

În fine, obţinem expresia pentru actualizarea dorită sub forma

$$\Delta W_{jk}^L(t) = -\varepsilon [\Delta_j^L, (X_k^{L-1})^T],$$

unde matricea $\Delta_j^L \in \mathfrak{so}(n)$ este dată de relaţia (5.4.6).

Acum, vom calcula actualizarea pentru o pondere arbitrară $W_{jk}^l$, unde $l \in \{2, \ldots, L-1\}$. În primul rând, putem scrie că

$$\Delta W_{jk}^l(t) = -\varepsilon \left( \frac{\partial E}{\partial [W_{jk}^l]_{ab}} \right)_{1 \leq a,b \leq n},$$

şi apoi, din regula înlănţuirii, avem că

$$\frac{\partial E}{\partial [W_{jk}^l]_{ab}} = \sum_{1\leq c,d\leq n} \frac{\partial E}{\partial [S_j^l]_{cd}} \frac{\partial [S_j^l]_{cd}}{\partial [W_{jk}^l]_{ab}}. \tag{5.4.7}$$

$\forall 1 \leq a, b \leq n$. Aplicând regula înlănţuirii din nou, obţinem că

$$\frac{\partial E}{\partial [S_j^l]_{cd}} = \sum_r \sum_{1\leq e,f\leq n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}}, \tag{5.4.8}$$

$\forall 1 \leq c, d \leq n$, unde suma este luată după toţi neuronii $r$ din stratul $l+1$ către care neuronul $j$ din stratul $l$ trimite conexiuni. Avem mai departe că

$$\frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}} = \sum_{1\leq g,h\leq n} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [Y_j^l]_{gh}} \frac{\partial [Y_j^l]_{gh}}{\partial [S_j^l]_{cd}},$$

$\forall 1 \leq c, d \leq n$, $\forall 1 \leq e, f \leq n$. Din nou din (5.4.2), putem să calculăm

$$\frac{\partial [S_r^{l+1}]_{ef}}{\partial [Y_j^l]_{gh}} = \begin{cases} [W_{rj}^{l+1}]_{eg}, & \text{dacă } f = h,\ e \neq g \\ -[W_{rj}^{l+1}]_{hf}, & \text{dacă } f \neq h,\ e = g \\ 0, & \text{altfel} \end{cases},$$

şi atunci

$$\begin{aligned} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}} &= \sum_{g=1}^n [W_{rj}^{l+1}]_{eg} \frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}} \\ &- \sum_{h=1}^n [W_{rj}^{l+1}]_{hf} \frac{\partial [G^l(S_j^l)]_{eh}}{\partial [S_j^l]_{cd}}, \end{aligned}$$

$\forall 1 \leq c, d \leq n$, $\forall 1 \leq e, f \leq n$. Acum, revenind la ecuaţia (5.4.8), şi punând totul împreună, avem că

$$\begin{aligned}
\frac{\partial E}{\partial [S_j^l]_{cd}} &= \sum_r \sum_{1\le e,f\le n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \sum_{g=1}^{n} [W_{rj}^{l+1}]_{eg} \frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}} \\
&- \sum_r \sum_{1\le e,f\le n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \sum_{h=1}^{n} [W_{rj}^{l+1}]_{hf} \frac{\partial [G^l(S_j^l)]_{eh}}{\partial [S_j^l]_{cd}} \\
&= \sum_r \sum_{1\le e,f,g\le n} [W_{rj}^{l+1}]_{eg} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \frac{\partial [G^l(S_j^l)]_{gf}}{\partial [S_j^l]_{cd}} \\
&- \sum_r \sum_{1\le e,f,h\le n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} [W_{rj}^{l+1}]_{hf} \frac{\partial [G^l(S_j^l)]_{eh}}{\partial [S_j^l]_{cd}} \\
&= \sum_r \left( \sum_{e=1}^{n} [(W_{rj}^{l+1})^T]_{ce} \frac{\partial E}{\partial [S_r^{l+1}]_{ed}} \right) \frac{\partial [G^l(S_j^l)]_{cd}}{\partial [S_j^l]_{cd}} \\
&- \sum_r \left( \sum_{f=1}^{n} \frac{\partial E}{\partial [S_r^{l+1}]_{cf}} [(W_{rj}^{l+1})^T]_{fd} \right) \frac{\partial [G^l(S_j^l)]_{cd}}{\partial [S_j^l]_{cd}} \\
&= \sum_r [(W_{rj}^{l+1})^T, \Delta_r^{l+1}]_{cd} \frac{\partial [G^l(S_j^l)]_{cd}}{\partial [S_j^l]_{cd}},
\end{aligned}$$

$\forall 1 \le c, d \le n$, unde din nou am ţinut cont de faptul că $\frac{\partial [G^l(S_j^l)]_{ef}}{\partial [S_j^l]_{cd}} = 0, \forall (e,f) \neq (c,d)$. Acum, notând $\Delta_j^l := \frac{\partial E}{\partial S_j^l}$, putem scrie relaţia de mai sus sub forma

$$\Delta_j^l = \left( \sum_r [(W_{rj}^{l+1})^T, \Delta_r^{l+1}] \right) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, \tag{5.4.9}$$

unde prin $\odot$ am notat înmulţirea pe componente a două matrici.

În final, ţinând cont de faptul că

$$\frac{\partial [S_j^l]_{cd}}{\partial [W_{jk}^l]_{ab}} = \begin{cases} [X_k^{l-1}]_{bd}, & \text{dacă } c = a,\ d \neq b \\ -[X_k^{l-1}]_{ca}, & \text{dacă } c \neq a,\ d = b \\ 0 & \text{altfel} \end{cases},$$

relaţia (5.4.7) devine

$$\begin{aligned}\frac{\partial E}{\partial [W_{jk}^L]_{ab}} &= \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^l]_{ad}} [X_k^{l-1}]_{bd} \\ &- \sum_{c=1}^{n} [X_k^{l-1}]_{ca} \frac{\partial E}{\partial [S_j^l]_{cb}} \\ &= \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^l]_{ad}} [(X_k^{l-1})^T]_{db} \\ &- \sum_{c=1}^{n} [(X_k^{l-1})^T]_{ac} \frac{\partial E}{\partial [S_j^l]_{cb}}. \\ &= [\Delta_j^l, (X_k^{l-1})^T]_{ab},\end{aligned}$$

$\forall 1 \leq a, b \leq n$. Relaţia de mai sus poate fi scrisă în formă matricială în modul următor:

$$\Delta W_{jk}^l(t) = -\varepsilon[\Delta_j^l, (X_k^{l-1})^T],$$

care este asemănătoare cu formula pe care am obţinut-o pentru stratul de ieşire $L$.

În concluzie, avem următoarea formulă pentru actualizarea ponderii $W_{jk}^l$:

$$\Delta W_{jk}^l(t) = -\varepsilon[\Delta_j^l, (X_k^{l-1})^T], \forall l \in \{2, \ldots, L\},$$

unde

$$\Delta_j^l = \begin{cases} \left(\sum_r [(W_{rj}^{l+1})^T, \Delta_r^{l+1}]\right) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, & l \leq L-1 \\ (Y_j^l - T_j) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, & l = L \end{cases}.$$

## 5.5 Medierea matricilor ortogonale

### 5.5.1 Introducere

Această secţiune este structurată după cum urmează. După prezentarea contextului, facem o prezentare detaliată a *algoritmului de încorporare*, sau *embedding algorithm*, cum a fost numit în articolul nostru [41]. Apoi prezentăm aplicarea acestui algoritm asupra grupului special ortogonal $SO(3)$, prin scufundare în $\mathbb{R}^4$, ca în articolul [41]. După aceea, aplicăm algoritmul grupului special ortogonal $SO(n)$, acestă aplicaţie apărând aici pentru prima dată.

O problemă des întâlnită în aplicaţii este de a găsi media unui set de puncte aparţinând unei varietăţi diferenţiabile $N$. Mai precis, fie $\{y_1, y_2, \ldots, y_m\} \subset N$ o mulţime finită de puncte şi $G_N : N \to \mathbb{R}$ o funcţie cost. Atunci media este definită ca fiind

$$\arg\min_{y \in N} G(y).$$

Un interes deosebit îl prezintă funcţiile cost de tip cele mai mici pătrate

$$G_N(y) = \sum_{i=1}^{m} d^2(y, y_i),$$

unde $d$ este o funcţie distanţă pe $N$. Media punctelor $\{y_1, y_2, \ldots, y_m\}$ pe varietatea $N$ este mulţimea definită prin

$$\arg\min_{y \in N} \sum_{i=1}^{m} d^2(y, y_i).$$

Când funcţia $G_N$ este diferenţiabilă, aceasta este echivalent cu rezolvarea ecuaţiei $dG_N(y) = 0$ şi testarea pentru care soluţii se obţine valoarea minimă. Scrierea acestei ecuaţii implică o cunoaştere a unui sistem local de coordonate pe varietatea $N$, o cerinţă care poate fi dificil de satisfăcut în multe situaţii practice. O altă problemă care poate apărea în această abordare a problemei este aceea că punctele $\{y_1, y_2, \ldots, y_m\}$ pot să nu aparţină toate domeniului unui singur sistem local de coordonate. O modalitate de a scăpa de aceste probleme este de a rezolva problema pe o varietate mai simplă $S$.

Fie $\mathbf{P} : S \to N$ o submersie surjectivă. Funcţia cost liftată $G_S : S \to \mathbb{R}$ este definită prin $G_S = G_N \circ \mathbf{P}$. Egalitatea de mulţimi $\mathbf{P}(\{x \in S | dG_S(x) = 0\}) = \{y \in N | dG_S(y) = 0\}$ arată că este suficient să rezolvăm ecuaţia $dG_S(x) = 0$ pe varietatea mai simplă $S$ şi să proiectăm aceste soluţii pe varietatea $N$.

O modalitate de a rezolva această problemă este de a înzestra $S$ cu o metrică riemanniană $\tau$, a calcula punctele critice ale câmpului de vectori $\nabla_\tau G_S$ şi a găsi care sunt minimele locale sau globale. Observăm că dacă $x_0 \in S$ este un punct critic pentru $\nabla_\tau G_S$, atunci este un punct critic şi pentru $\nabla_{\tau'} G_S$, unde $\tau'$ este orice altă metrică riemanniană pe $S$. Deşi varietatea $S$ s-ar putea să aibă o structură geometrică mai simplă decât varietatea iniţială $N$, totuşi avem nevoie de un sistem de coordonate locale pentru a putea calcula punctele critice ale funcţiei $G_S$, a se vedea [1], [2], [79], [83], [177], [227]. Pentru a evita utilizarea coordonatelor locale, vom scufunda $S$ într-un spaţiu mai mare $M$. O teoremă a lui Whitney ne garantează că această scufundare se poate întotdeauna face, cu $M$ spaţiu euclidian.

În cele ce urmează, vom prezenta o soluţie pentru găsirea punctelor critice ale funcţiei cost liftate $G_S$ prin construirea unui câmp de vectori pe spaţiul ambient $M$ (de obicei un spaţiu euclidian) $\mathbf{v_0} \in \mathscr{X}(M)$, care este tangent subvarietăţii $S$ şi construirea unei metrici riemanniene $\tau$ pe $S$ astfel încât $\mathbf{v_0}_{|S} = \nabla_\tau G_S$. Prin urmare, punctele critice ale câmpului de vectori $\mathbf{v_0}$ (care este de obicei scris în coordonate euclidiene) care aparţin lui $S$ sunt puncte critice ale funcţiei de cost liftate $G_S$ iar proiecţiile lor prin submersia surjectivă $\mathbf{P}$ dau punctele critice ale funcţiei de cost iniţiale $G_N$. Această construcţie va fi ilustrată în detaliu pentru problema de mediere pe grupul Lie $SO(3)$ asociat cu patru funcţii de cost diferite. Printre aceste funcţii vom studia funcţiile de cost de tip $L^p$. Vom analiza şi compara problemele de mediere pentru cazurile $L^2$ şi $L^4$. Cazul $L^2$ a fost studiat în literatură. Vom arăta că în cazul $L^4$ anumite aspecte importante diferă faţă de cazul $L^2$.

În literatură, problema medierii pe grupuri Lie este adesea rezolvată folosind aplicaţia exponenţială ca o unealtă de a introduce coordonate locale şi a lifta astfel problema pe spaţiul tangent, a se vedea [2], [22], [83], [84], [100], [213]. Această metodă poate fi generalizată în contextul varietăţilor riemanniene deoarece şi acolo există o aplicaţie exponenţială. Aplicaţia exponenţială poate fi mai departe înlocuită cu noţiunea de retractă dezvoltată în [2], [3], [82]. Problemele de mediere vin din aplicaţii din viaţa reală şi au fost studiate în [1], [2], [111], [227], [243], [262], folosind noţiunile de derivată covariantă şi geometria geodezicelor pe diferite varietăţi riemanniene. În acestă teză vom propune un algoritm diferit pentru rezolvarea problemei medierii pe varietăţi diferenţiabile generale.

Alte funcţii de cost interesante sunt cele care provin din problema Fermat-Torricelli. Această problemă de mediere a fost studiată pe diferite spaţii, a se vedea [77], [78],

[108], [207], şi ar fi de interes aplicarea tehnicilor dezvoltate în această teză unor astfel de probleme.

### 5.5.2 Medierea pe o varietate riemanniană

Vom rezolva problema de mediere prezentată în Introducere prin scufundarea varietăţii $S$ ca o subvarietate a varietăţii riemanniene $(M, g)$. Mai mult, vom presupune în continuare că $S$ este preimaginea unei valori regulate pentru o funcţie netedă $\mathbf{F} := (F_1, \ldots, F_k) : M \to \mathbb{R}^k$, adică $S = \mathbf{F}^{-1}(c)$, pentru $c$ o valoare regulată a lui $\mathbf{F}$.

După cum deja am menţionat, abordarea constă în găsirea unei metrici riemanniene $\tau$ pe subvarietatea $S$ şi un câmp de vectori $\mathbf{v_0} \in \mathscr{X}(M)$ astfel încât

$$\mathbf{v_0}_{|S} = \nabla_\tau G_S. \tag{5.5.1}$$

Prin urmare, problema iniţială este echivalentă cu găsirea punctelor critice ale câmpului de vectori $\mathbf{v_0}$, care este definit pe spaţiul ambient $M$, şi alegerea acelora care aparţin lui $S$.

În cele ce urmează, vom prezenta construcţia câmpului de vectori de control standard $\mathbf{v_0}$. Fie $(M, g)$ o varietate riemannianã $n$ dimensională şi $F_1, \ldots, F_k, G : M \to \mathbb{R}$ $k+1$ funcţii netede. Câmpul de vectori de control standard este definit prin

$$\mathbf{v_0} = \sum_{i=1}^{k} (-1)^{i+k+1} \det \Sigma^{(F_1,\ldots,F_k,G)}_{(F_1,\ldots,\hat{F_i},\ldots,F_k,G)} \nabla F_i + \det \Sigma^{(F_1,\ldots,F_k)}_{(F_1,\ldots,F_k)} \nabla G, \tag{5.5.2}$$

unde $\hat{\cdot}$ reprezintă un termen lipsă, iar $\Sigma^{(f_1,\ldots,f_r)}_{(g_1,\ldots,g_s)}$ reprezintă matricea Gram de dimensiune $r \times s$:

$$\Sigma^{(f_1,\ldots,f_r)}_{(g_1,\ldots,g_s)} = \begin{pmatrix} < \nabla g_1, \nabla f_1 > & \ldots & < \nabla g_s, \nabla f_1 > \\ \ldots & \ldots & \ldots \\ < \nabla g_1, \nabla f_r > & \ldots & < \nabla g_s, \nabla f_r > \end{pmatrix}, \tag{5.5.3}$$

generată de funcţiile netede $f_1, ..., f_r, g_1, ..., g_s : M \to \mathbb{R}$.

Câmpul de vectori $\mathbf{v_0}$ conservă foile regulate ale funcţiei $\mathbf{F} := (F_1, \ldots, F_k) : M \to \mathbb{R}^k$ şi disipă funcţia $G$, mai precis derivarea funcţiei $G$ de-a lungul câmpului de vectori $\mathbf{v_0}$ este dat de derivata Lie $\mathcal{L}_{\mathbf{v_0}} G = \det \Sigma^{(F_1,\ldots,F_k,G)}_{(F_1,\ldots,F_k,G)} \geq 0$.

Vom descrie în continuare geometria câmpului de vectori de control standard. Fie $\Omega^1(M)$ spaţiul vectorial real al 1-formelor diferenţiale pe varietatea $M$. Fie $\mathbf{T} : \Omega^1(M) \times \Omega^1(M) \to \mathbb{R}$ 2-tensorul simetric contravariant degenerat dat de

$$\mathbf{T} := \sum_{i,j=1}^{k} (-1)^{i+j+1} \det \Sigma^{(F_1,\ldots,\hat{F_j},\ldots,F_k)}_{(F_1,\ldots,\hat{F_i},\ldots,F_k)} \nabla F_i \otimes \nabla F_j + \det \Sigma^{(F_1,\ldots,F_k)}_{(F_1,\ldots,F_k)} g^{-1}. \tag{5.5.4}$$

unde $g^{-1}$ este 2-tensorul cometric $g^{-1}(x) = g^{pq}(x) \frac{\partial}{\partial x^p} \otimes \frac{\partial}{\partial x^q}$ construit din tensorul metric $g$ şi 2-tensorul contravariant $\nabla F_i \otimes \nabla F_j : \Omega^1(M) \times \Omega^1(M) \to \mathbb{R}$ este definit de formula

$$\nabla F_i \otimes \nabla F_j \ (\alpha, \beta) := \alpha(\nabla F_i)\beta(\nabla F_j).$$

În geometria riemannianã, câmpul de vectori gradient al unei funcţii netede $G$ poate fi definit prin formula $\nabla G = \mathbf{i}_{dG} g^{-1}$, unde prin $\mathbf{i}$ am notat produsul interior definit prin $\mathbf{i}_{dG} g^{-1}(\alpha) := g^{-1}(dG, \alpha), \ \forall \alpha \in \Omega^1(M)$. Următorul rezultat arată că $\mathbf{v_0}$ arată ca un câmp de vectori gradient al funcţiei $G$, în funcţie de 2-tensorul simetric contravariant degenerat $\mathbf{T}$.

**Teorema 2.** *[40] Pe varietatea riemannianã* $(M, g)$ *câmpul de vectori de control standard* $\mathbf{v_0}$ *este dat de următoarea formulă:*

$$\mathbf{v_0} = \mathbf{i}_{dG}(\mathbf{T}).$$

Din teorema de mai sus, a calcula punctele critice ale lui $\mathbf{v_0}$ (care este scris în coordonate locale pe $M$) care aparţin de asemenea lui $S$ este echivalent cu a rezolva următorul sistem de $n+k$ ecuaţii pe $M$

$$\mathbf{v_0}(x) = \begin{bmatrix} \mathbf{T}^{11}(x) & \dots & \mathbf{T}^{1n}(x) \\ \vdots & \dots & \vdots \\ \mathbf{T}^{n1}(x) & \dots & \mathbf{T}^{nn}(x) \end{bmatrix} \begin{bmatrix} \dfrac{\partial G}{\partial x^1}(x) \\ \vdots \\ \dfrac{\partial G}{\partial x^n}(x) \end{bmatrix}, \tag{5.5.5}$$

unde $\mathbf{T} = \mathbf{T}^{pq} \dfrac{\partial}{\partial x^p} \otimes \dfrac{\partial}{\partial x^q}$ şi $dG = \dfrac{\partial G}{\partial x^i} dx^i$. În general, un sistem de $n+k$ ecuaţii cu $n$ necunoscute poate să nu aibă soluţii. Dar sistemul de mai sus care are $n$ necunoscute, şi anume coordonatele lui $x$ în $M$, are exact $n$ ecuaţii funcţional independente datorită faptului că rangul tensorului $\mathbf{T}$ este $n-k$, pentru fiecare punct regulat din mulţimea deschisă în $M$ a punctelor regulate ale lui $\mathbf{F}$ (rangul tensorului $\mathbf{T}$ este $m-k$ în fiecare punct $x \in M$ unde $\nabla F_1(x), ..., \nabla F_k(x)$ sunt liniar independenţi, a se vedea Secţiunea 4 din [40]) .

Câmpul de vectori de control standard $\mathbf{v_0}$ este tangent la fiecare foaie regulată şi în continuare îi vom prezenta geometria când este restricţionat la o foaie regulată. Fie $i_c : L_c \to M$ incluziunea canonică a foii regulate $L_c$ în varietatea $M$. 2-tensorul simetric contravariant $\mathbf{T}$ este nedegenerat atunci când este restricţionat la o foaie regulată $L_c$ şi prin urmare restricţia poate fi inversată, generând, după cum se poate vedea în [40], metrica riemannianã $\tau_c : \mathscr{X}(L_c) \times \mathscr{X}(L_c) \to \mathcal{C}^\infty(L_c)$ definită prin

$$\tau_c(X^c, Y^c) := \mathbf{T}^{-1}(i_{c_*} X^c, i_{c_*} Y^c),$$

unde $(i_c)_* : \mathscr{X}(L_c) \to \mathscr{X}(M)$ este aplicaţia push-forward.

**Teorema 3.** *[40] Pe o foaie regulată* $L_c$ *avem următoarele caracterizări*

*(i)* $\tau_c = \dfrac{1}{\det \Sigma^{(F_1,\dots,F_k)}_{(F_1,\dots,F_k)}} i_c^* g$

*(ii)* $\mathbf{v_0}_{|L_c} = i_{c_*} \nabla_{\tau_c}(G \circ i_c)$.

Teorema de mai sus arată că tensorul $\mathbf{T}$ induce o metrică riemannianã pe fiecare foaie regulată $L_c$ care este prima formă fundamentală a subvarietăţii $L_c \subset (M, g)$ înmulţită cu o funcţie pozitivă. Câmpul de vectori $\mathbf{v_0}$ este tangent la subvarietatea $L_c$ şi restricţia este egală cu câmpul de vectori gradient al funcţiei restricţionate $G_{|L_c}$ în funcţie de metrica $\tau_c$. Cum $S$ este o astfel de foaie regulată, metrica riemannianã pe $S$ este $\tau = \tau_{c_0}$ şi teorema de mai sus arată că egalitatea (5.5.1) are loc, unde $G_S = G_{|S}$.

În concluzie, pentru a rezolva problema noastră de mediere pe varietatea $N$ care are asociată o funcţie de cost $G_N$, folosind câmpul de vectori de control standard, aplicăm următorul **algoritm de încorporare** (embedding algorithm):

**(i)** Alegem un cadrul geometric constituit de varietatea $S$ şi o submersie surjectivă $\mathbf{P} : S \to N$ şi construim funcţia de cost liftată $G_S = G_N \circ \mathbf{P}$.

**(ii)** Găsim un spaţiu ambient o varietate riemannianã $(M, g)$ şi o funcţie diferenţiabilă $\mathbf{F} : M \to \mathbb{R}^k$ astfel încât $S = \mathbf{F}^{-1}(c)$, unde $c$ este o valoare regulată a lui $\mathbf{F}$. Construim 2-tensorul simetric contravariant $\mathbf{T}$.

**(iii)** Găsim o funcţie $G : M \to \mathbb{R}$ astfel încât $G_{|S} = G_S$ şi construim câmpul de vectori de control standard $\mathbf{v_0}$ cu datele iniţiale $\mathbf{F}, G$.

**(iv)** Rezolvăm sistemul $\mathbf{v_0}_{|S}(x) = 0$ (mai precis sistemul (5.5.5)), care este echivalent cu găsirea punctelor critice ale funcţiei de cost liftate $G_S$. Proiectăm prin $\mathbf{P}$ aceste soluţii, găsind astfel punctele critice ale funcţiei de cost originale $G_N$.

**(v)** Găsim acele puncte critice care sunt minime locale sau globale pentru funcţia de cost $G_N$.

În cazul în care $N$ poate fi scrisă ca preimaginea unei valori regulate a unei funcţii netede $\mathbf{F}$, atunci pasul **(i)** nu mai este necesar ($S = N$ şi $\mathbf{P}$ este identitatea).

În cazul în care $\mathbf{P}_*\mathbf{v_0}_{|S}$ este un câmp de vectori pe varietatea $N$, atunci avem incluziunea de mulţimi $\mathbf{P}(\{x \in S | \mathbf{v_0}_{|S}(x) = 0\}) \subset \{y \in N | \mathbf{P}_*\mathbf{v_0}_{|S}(y) = 0\}$. Dacă $\mathbf{P}$ este de asemenea un difeomorfism local obţinem egalitate în incluziunea de mulţimi de mai sus şi prin urmare punctele critice ale funcţiei de cost $G_N$ sunt caracterizate prin ecuaţia $\mathbf{P}_*\mathbf{v_0}_{|S}(y) = 0$. În acest caz particular, pasul **(iv)** poate fi înlocuit prin

**(iv')** Presupunem că $\mathbf{P}_*\mathbf{v_0}_{|S} \in \mathscr{X}(N)$ iar $\mathbf{P}$ este un difeomorfism local, atunci rezolvăm ecuaţia $\mathbf{P}_*\mathbf{v_0}_{|S}(y) = 0$ (soluţiile sunt puncte critice pentru funcţia de cost $G_N$).

Pentru rezolvarea punctului **(v)** putem studia derivatele de ordinul doi ale funcţiei de cost $G_N$ (a se vedea [79], [111], [177]) sau să studiem proprietăţile de convexitate ale funcţiilor de cost ca în [108].

### 5.5.3 Medierea pe $SO(3)$

O problemă frecvent întâlnită în practică este de a găsi o rotaţie care este media unei mulţimi finite de rotaţii în spaţiul tridimensional. Grupul special al rotaţiilor $SO(3)$ nu este un spaţiu euclidian, dar este o varietate diferenţiabilă. Prin urmare, o problemă de mediere pe un astfel de spaţiu trebuie considerată pe tărâmul geometriei diferenţiale. Vom discuta scufundarea lui $SO(3)$ în spaţiul euclidian $\mathbb{R}^4$.

Grupul ortogonal special $SO(3)$ poate fi identificat cu ajutorul unei funcţii de acoperire dublă cu sfera $S^3$ din $\mathbb{R}^4$, a se vedea [10], [240]. Varietatea noastră $S$ va fi $S^3$ şi spaţiul ambient $(M, g)$ va fi spaţiul euclidian $\mathbb{R}^4$. Vom lua diferite funcţii distanţă pe $SO(3)$ care vor genera diferite funcţii de cost şi vom rezolva problemele de mediere asociate, după cum am arătat în subsecţiunea anterioară.

Cuaternionii unitate $\mathbf{q} = (q^1, q^2, q^3, q^4) \in S^3 \subset \mathbb{R}^4$ şi $-\mathbf{q} \in S^3 \subset \mathbb{R}^4$ corespund următoarei rotaţii din $SO(3)$:

$$\mathbf{R^q} = \begin{pmatrix} (q^0)^2 + (q^1)^2 + (q^2)^2 + (q^3)^2 & 2(q^1q^2 - q^0q^3) & 2(q^1q^3 + q^0q^2) \\ 2(q^1q^2 + q^0q^3) & (q^0)^2 - (q^1)^2 + (q^2)^2 - (q^3)^2 & 2(q^2q^3 - q^0q^1) \\ 2(q^1q^3 - q^0q^2) & 2(q^2q^3 + q^0q^1) & (q^0)^2 - (q^1)^2 - (q^2)^2 + (q^3)^2 \end{pmatrix}.$$

Aceasta generează proiecţia de acoperire cu două foi $\mathbf{P} : S^3 \to SO(3)$, $\mathbf{P}(\mathbf{q}) = \mathbf{R^q}$. Proiecţia de acoperire este un difeomorfism local, şi prin urmare, în loc să lucrăm cu funcţia distanţă (şi funcţia cost) pe $SO(3)$, vom lucra cu funcţii de cost pe sfera unitate $S^3 \subset \mathbb{R}^4$, după cum am explicat în subsecţiunea anterioară. În acest caz, avem că $S^3 = F^{-1}(1)$, unde $F : \mathbb{R}^4 \to \mathbb{R}$, $F(\mathbf{q}) = (q^0)^2 + (q^1)^2 + (q^2)^2 + (q^3)^2$ şi tensorul $\mathbf{T}$ este dat de formula $\mathbf{T} = -\nabla F \otimes \nabla F + ||\nabla F||^2 g^{-1}$, unde $g$ este metrica euclidiană pe $\mathbb{R}^4$. Scriind explicit, matricea asociată 2-tensorului simetric contravariant $\mathbf{T}$ este dată de:

$$\mathbf{T}(\mathbf{q}) = 4\begin{pmatrix} (q^1)^2+(q^2)^2+(q^3)^2 & -q^0q^1 & -q^0q^2 & -q^0q^3 \\ -q^1q^0 & (q^0)^2+(q^2)^2+(q^3)^2 & -q^1q^2 & -q^1q^3 \\ -q^2q^0 & -q^2q^1 & (q^0)^2+(q^1)^2+(q^3)^2 & -q^2q^3 \\ -q^3q^0 & -q^3q^1 & -q^3q^2 & (q^0)^2+(q^1)^2+(q^2)^2 \end{pmatrix}.$$

Dacă $\omega(\mathbf{q}) = \omega_0(\mathbf{q})dq^0+\omega_1(\mathbf{q})dq^1+\omega_2(\mathbf{q})dq^2+\omega_3(\mathbf{q})dq^3$ este o 1-formă pe $\mathbb{R}^4$, atunci

$$\mathbf{i}_\omega\mathbf{T}(\mathbf{q}) = 4(\langle\mathbf{q},\mathbf{q}\rangle\overline{\omega}(\mathbf{q}) - \langle\mathbf{q},\overline{\omega}(\mathbf{q})\rangle\mathbf{q}), \tag{5.5.6}$$

unde am făcut notaţia $\overline{\omega}(\mathbf{q}) := (\omega_0(\mathbf{q}),\omega_1(\mathbf{q}),\omega_2(\mathbf{q}),\omega_3(\mathbf{q}))$.

Vom exemplifica construcţia folosind diverse funcţii distanţă folosite în literatură pentru măsurarea distanţelor dintre transformările euclidiene. O listă extinsă este prezentată în [129], unde este de asemenea studiată echivalenţa şi funcţional dependenţa lor.

**I.** Pe grupul Lie $SO(3)$ vom considera funcţia distanţă $d_1 : SO(3)\times SO(3)\to\mathbb{R}_+$, $d_1(\mathbf{R}_1,\mathbf{R}_2) = ||\mathbf{R}_1-\mathbf{R}_2||_F$, unde $||\cdot||_F$ este norma Frobenius. Funcţia de cost asociată este $G_{1_{SO(3)}} : SO(3)\to\mathbb{R}$,

$$G_{1_{SO(3)}}(\mathbf{R}) = \sum_{i=1}^m ||\mathbf{R}-\mathbf{R}_i||_F^2,$$

unde $\{\mathbf{R}_1,...,\mathbf{R}_m\}$ sunt matricile de rotaţie cărora dorim să le facem media, a se vedea [177], [239].

Conform pasului **(i)** al algoritmului de încorporare, vom transforma problema găsirii punctelor critice ale funcţiei de cost $G_{1_{SO(3)}}$ în problema găsirii punctelor critice pentru funcţia de cost liftată $G_{1_{S^3}} : S^3\to\mathbb{R}$, $G_{1_{S^3}} := G_{1_{SO(3)}}\circ\mathbf{P}$.

Pentru a calcula funcţia de cost liftată $G_{1_{S^3}}$, folosind surjectivitatea proiecţiei de acoperire $\mathbf{P}$, pentru fiecare matrice $\mathbf{R}_i$ vom alege un cuaternion corespunzător $\mathbf{q}_i$. Avem următorul calcul:

$$\begin{aligned} G_{1_{S^3}}(\mathbf{q}) &= \sum_{i=1}^m ||\mathbf{R}^{\mathbf{q}}-\mathbf{R}^{\mathbf{q_i}}||_F^2 = \sum_{i=1}^m \mathrm{tr}((\mathbf{R}^{\mathbf{q}}-\mathbf{R}^{\mathbf{q_i}})(\mathbf{R}^{\mathbf{q}}-\mathbf{R}^{\mathbf{q_i}})^T) \\ &= 2\sum_{i=1}^m(3-\mathrm{tr}(\mathbf{R}^{\mathbf{q}T}\mathbf{R}^{\mathbf{q_i}})) \\ &= 8\sum_{i=1}^m(1-\langle\mathbf{q},\mathbf{q}_i\rangle^2), \end{aligned}$$

unde am folosit egalitatea

$$\langle\mathbf{q},\mathbf{q}_i\rangle^2 = \frac{1}{4}(\mathrm{tr}(\mathbf{R}^{\mathbf{q}T}\mathbf{R}^{\mathbf{q_i}})+1). \tag{5.5.7}$$

Pentru implementarea pasului **(iii)**, trebuie să construim funcţia prelungită $G_1 : \mathbb{R}^4\to\mathbb{R}$,

$$G_1(\mathbf{q}) = \sum_{i=1}^m 8(1-\langle\mathbf{q},\mathbf{q}_i\rangle^2).$$

Să observăm că $G_1$ este o funcţie pară şi prin urmare nu depinde de alegerea cuaternionilor $\mathbf{q}_i$ sau $-\mathbf{q}_i$ care reprezintă aceeaşi rotaţie $\mathbf{R}_i$. Această problemă mai

subtilă a fost de asemenea tratată într-o manieră diferită în [173]. Calculând 1-forma diferenţială $dG_1$ obţinem:

$$\begin{aligned}\overline{dG_1}(\mathbf{q}) &= \left(\frac{\partial G_1}{\partial q^0}(\mathbf{q}),\ldots,\frac{\partial G_1}{\partial q^3}(\mathbf{q})\right) = \left(-16\sum_{i=1}^{m} q_i^0\langle \mathbf{q},\mathbf{q}_i\rangle,\ldots,-16\sum_{i=1}^{m} q_i^3\langle \mathbf{q},\mathbf{q}_i\rangle\right)\\ &= -16\sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle\mathbf{q}_i.\end{aligned}$$

Folosind (5.5.6), unde 1-forma $\omega$ este $dG_1$, sistemul de ecuaţii (5.5.5) devine:

$$\begin{cases}\displaystyle\sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle(\langle \mathbf{q},\mathbf{q}_i\rangle\mathbf{q}_i - \langle \mathbf{q},\mathbf{q}_i\rangle\mathbf{q}) = \mathbf{0}\\ \langle \mathbf{q},\mathbf{q}\rangle = 1\end{cases}, \tag{5.5.8}$$

şi acesta reprezintă ecuaţia $\mathbf{v_0}_{|S^3}(\mathbf{q}) = \mathbf{0}$. Sistemul de mai sus are patru ecuaţii de gradul trei în cele patru necunoscute $\mathbf{q} = (q^0, q^1, q^2, q^3)$ corespunzând câmpului de vectori de control standard $\mathbf{v_0} = \mathbf{0}$, plus constrângerea care descrie pe $S^3$.

Cele patru ecuaţii corespunzând lui $\mathbf{v_0} = \mathbf{0}$ nu sunt funcţional independente deoarece, după cum am arătat în subsecţiunea anterioară, tensorul $\mathbf{T}$ este degenerat. Totuşi, deoarece $\mathbf{v_0}(\mathbf{q}) \in T_{\mathbf{q}}S^3$, pentru orice $\mathbf{q} \in S^3$, sistemul (5.5.8) poate fi descris prin doar trei ecuaţii plus constrângerea.

Pentru a scrie $\mathbf{v_0}_{|S^3}(\mathbf{q}) = \mathbf{0}$ ca un sistem de trei ecuaţii independente, vom împinge înainte câmpul de vectori $\mathbf{v_0}_{|S^3}$ prin proiecţia de acoperire $\mathbf{P}$, care ne va plasa în ipoteza pasului **(iv')** al algoritmului de încorporare. Din definiţia operatorului push-forward, avem că

$$\mathbf{P}_*\mathbf{v_0}_{|S^3}(\mathbf{R^q}) = T_{\mathbf{q}}\mathbf{P}(\mathbf{v_0}_{|S^3}(\mathbf{q})) \in T_{\mathbf{R^q}}SO(3).$$

Cum $\mathbf{P}$ nu este o aplicaţie injectivă, pentru a obţine un câmp de vectori pe spaţiul destinaţie $SO(3)$ egalitatea $T_{\mathbf{q}}\mathbf{P}(\mathbf{v_0}_{|S^3}(\mathbf{q})) = T_{-\mathbf{q}}\mathbf{P}(\mathbf{v_0}_{|S^3}(-\mathbf{q}))$ trebuie să fie satisfăcută. Vom arăta în cele ce urmează că ea este într-adevăr satisfăcută.

În continuare, trebuie să calculăm aplicaţia tangentă a proiecţiei de acoperire $\mathbf{P} : S^3 \to SO(3)$. Această aplicaţie poate fi scrisă ca restricţia aplicaţiei $\widetilde{\mathbf{P}} : \mathbb{R}^4 \to \mathbb{R}^9$ definită prin $\widetilde{\mathbf{P}}(q^0,q^1,q^2,q^3) = ((q^0)^2+(q^1)^2-(q^2)^2-(q^3)^2, 2(q^1q^2-q^0q^3), 2(q^1q^3+q^0q^2), 2(q^1q^2+q^0q^3), (q^0)^2-(q^1)^2+(q^2)^2-(q^3)^2, 2(q^2q^3-q^0q^1), 2(q^1q^3-q^0q^2), 2(q^2q^3+q^0q^1), (q^0)^2-(q^1)^2-(q^2)^2+(q^3)^2)$, unde o matrice de rotaţie $\mathbf{R^q}$ a fost identificată cu un punct din $\mathbb{R}^9$.

Matricea corespunzătoare aplicaţiei liniare $T_{\mathbf{q}}\widetilde{\mathbf{P}} : \mathbb{R}^4 \to \mathbb{R}^9$ este dată prin

$$\mathrm{Jac}_{\mathbf{q}}\widetilde{\mathbf{P}} = 2\begin{bmatrix} q^0 & q^1 & -q^2 & -q^3\\ -q^3 & q^2 & q^1 & -q^0\\ q^2 & q^3 & q^0 & q^1\\ q^3 & q^2 & q^1 & q^0\\ q^0 & -q^1 & q^2 & -q^3\\ -q^1 & -q^0 & q^3 & q^2\\ -q^2 & q^3 & -q^0 & q^1\\ q^1 & q^0 & q^3 & q^2\\ q^0 & -q^1 & -q^2 & q^3 \end{bmatrix}.$$

Printr-un calcul direct, în care identificăm un vector din $\mathbb{R}^9$ cu un vector tangent $\mathbf{R^q}\Delta \in T_{\mathbf{R^q}}SO(3) = \{\mathbf{R^q}\Delta \,|\, \Delta \text{ este o matrice } 3\times 3 \text{ antisimetrică}\}$, pentru $\mathbf{q} \in S^3$ avem că

$$\begin{aligned} T_{\mathbf{q}}\mathbf{P}\cdot \mathbf{v_0}_{|S^3}(\mathbf{q}) &= T_{\mathbf{q}}\widetilde{\mathbf{P}}\cdot \mathbf{v_0}_{|S^3}(\mathbf{q}) \\ &= \sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle T_{\mathbf{q}}\widetilde{\mathbf{P}}\cdot(\langle \mathbf{q},\mathbf{q}\rangle \mathbf{q}_i - \langle \mathbf{q},\mathbf{q}_i\rangle \mathbf{q}) \\ &= \sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle \mathrm{Jac}_{\mathbf{q}}\widetilde{\mathbf{P}}\cdot(\langle \mathbf{q},\mathbf{q}\rangle \mathbf{q}_i - \langle \mathbf{q},\mathbf{q}_i\rangle \mathbf{q}) \\ &\cong \sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle \mathbf{R^q}\Delta_i(\mathbf{q}) \\ &= \mathbf{R^q}\sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle \Delta_i(\mathbf{q}), \end{aligned}$$

unde

$$\Delta_i(\mathbf{q}) = \begin{pmatrix} 0 & -q^0q_i^3 + q^1q_i^2 - q^2q_i^1 + q^3q_i^0 & q^0q_i^2 + q^1q_i^3 - q^2q_i^0 - q^3q_i^1 \\ q^0q_i^3 - q^1q_i^2 + q^2q_i^1 - q^3q_i^0 & 0 & -q^0q_i^1 + q^1q_i^0 + q^2q_i^3 - q^3q_i^2 \\ -q^0q_i^2 - q^1q_i^3 + q^2q_i^0 + q^3q_i^1 & q^0q_i^1 - q^1q_i^0 - q^2q_i^3 + q^3q_i^2 & 0 \end{pmatrix}.$$

Identificând vectorul $\mathrm{Jac}_{\mathbf{q}}\widetilde{\mathbf{P}}\cdot(\langle \mathbf{q},\mathbf{q}\rangle \mathbf{q}_i - \langle \mathbf{q},\mathbf{q}_i\rangle \mathbf{q}) \in \mathbb{R}^9$ cu o matrice $3\times 3$, matricea antisimetrică este dată de formula

$$\Delta_i(\mathbf{q}) = \mathbf{R^q}^T \mathrm{Jac}_{\mathbf{q}}\widetilde{\mathbf{P}}\cdot(\langle \mathbf{q},\mathbf{q}\rangle \mathbf{q}_i - \langle \mathbf{q},\mathbf{q}_i\rangle \mathbf{q}).$$

Observând că $\Delta_i(-\mathbf{q}) = -\Delta_i(\mathbf{q})$ avem că $T_{\mathbf{q}}\mathbf{P}(\mathbf{v_0}_{|S^3}(\mathbf{q})) = T_{-\mathbf{q}}\mathbf{P}(\mathbf{v_0}_{|S^3}(-\mathbf{q}))$ şi prin urmare, $\mathbf{P}_*\mathbf{v_0}_{|S^3}$ este un câmp de vectori pe varietatea $SO(3)$.

Ca urmare obţinem că punctele critice ale funcţiei de cost $G_{1_{S^3}}$ sunt soluţiile următorului sistem (care este echivalent cu (5.5.8)):

$$\begin{cases} \displaystyle\sum_{i=1}^{m}\langle \mathbf{q},\mathbf{q}_i\rangle \Delta_i(\mathbf{q}) = \mathbf{0} \\ \langle \mathbf{q},\mathbf{q}\rangle = 1 \end{cases}. \tag{5.5.9}$$

Avantajul sistemului de mai sus comparativ cu sistemul (5.5.8) este că avem doar trei ecuaţii de gradul doi în locul a patru ecuaţii de gradul trei plus o constrângere.

Dacă transformăm sistemul de mai sus în rotaţii, prin calcul direct avem că:

$$\langle \mathbf{q},\mathbf{q}_i\rangle \Delta_i(\mathbf{q}) = \frac{1}{4}(\mathbf{R^{q_i}}^T\mathbf{R^q} - \mathbf{R^q}^T\mathbf{R^{q_i}}), \tag{5.5.10}$$

şi sistemul de mai sus devine:

$$\sum_{i=1}^{m}(\mathbf{R^{q_i}}^T\mathbf{R^q} - \mathbf{R^q}^T\mathbf{R^{q_i}}) = \mathbf{0},$$

care este echivalent cu

$$\overline{\mathbf{R}}^T\mathbf{R^q} - \mathbf{R^q}^T\overline{\mathbf{R}} = \mathbf{0}, \tag{5.5.11}$$

unde $\overline{\mathbf{R}} = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}\mathbf{R^{q_i}} = \dfrac{1}{m}\sum_{i=1}^{m}\mathbf{R}_i$. Soluţiile ecuaţiei de mai sus dau punctele critice ale funcţiei de cost $G_{1_{SO(3)}}$. Găsirea acestor soluţii corespunde pasului **(iv')** din algoritmul

de încorporare. De asemenea, această ecuaţie este aceeaşi cu caracterizarea pentru media aritmetică obţinută în [177]. Prin urmare căutarea punctelor critice ale funcţiei de cost $G_{1_{SO(3)}}$ poate fi făcută în două moduri. Mai precis, putem rezolva (5.5.11) sau putem transforma problema în cuaternioni şi să rezolvăm sistemul (5.5.9) transformând apoi soluţiile în rotaţii.

**II.** În continuare, vom lua în considerare distanţa geodezică pe grupul Lie $SO(3)$ care este definită prin unghiul dintre două rotaţii, după cum se arată în [10], [22], [101], [177], [213], [239]. Pentru două rotaţii $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$,

$$d_2(\mathbf{R}_1, \mathbf{R}_2) = ||\mathrm{Log}(\mathbf{R}_1^T \mathbf{R}_2)||_F = \sqrt{2}|\theta|,$$

unde $\theta \in (-\pi, \pi)$ este unghiul dintre rotaţiile $\mathbf{R}_1$ şi $\mathbf{R}_2$. Funcţia de cost asociată este $G_{2_{SO(3)}} : SO(3) \to \mathbb{R}$,

$$G_{2_{SO(3)}}(\mathbf{R}) = \sum_{i=1}^{m} ||\mathrm{Log}(\mathbf{R}_i^T \mathbf{R})||_F^2,$$

unde $\{\mathbf{R}_1, ..., \mathbf{R}_m\}$ sunt matricile de rotaţie cărora dorim să le facem media.

Ca mai înainte, vom calcula funcţia de cost liftată $G_{2_{S^3}}$, ca în pasul **(i)** al algoritmului de încorporare. Mai precis, avem că

$$\begin{aligned} G_{2_{S^3}}(\mathbf{q}) &= \sum_{i=1}^{m} ||\mathrm{Log}(\mathbf{R}_i^T \mathbf{R})||_F^2 = 2\sum_{i=1}^{m} \theta_i^2 \\ &= 2\sum_{i=1}^{m} \arccos^2\left(\frac{\mathrm{tr}(\mathbf{R}^{\mathbf{q}_i T}\mathbf{R}^{\mathbf{q}}) - 1}{2}\right) \\ &= 2\sum_{i=1}^{m} \arccos^2(2\langle \mathbf{q}_i, \mathbf{q}\rangle^2 - 1) \\ &= 2\sum_{i=1}^{m} \arccos^2(|\langle \mathbf{q}_i, \mathbf{q}\rangle|). \end{aligned}$$

Funcţia de cost liftată $G_{2_{S^3}}$ nu depinde de alegerea cuaternionilor $\mathbf{q}_i$ sau $-\mathbf{q}_i$ care reprezintă aceeaşi rotaţie $\mathbf{R}_i$. Pentru pasul **(iii)** al algoritmului de încorporare, funcţia prelungită este $G_2 : \mathbb{R}^4 \backslash \{\mathbf{0}\} \to \mathbb{R}$,

$$G_2(\mathbf{q}) = 2\sum_{i=1}^{m} \arccos^2\left(\frac{|\langle \mathbf{q}, \mathbf{q}_i\rangle|}{||\mathbf{q}|| \cdot ||\mathbf{q}_i||}\right).$$

Coeficienţii 1-formei diferenţiale $dG_2$ sunt daţi de:

$$\begin{aligned} \overline{dG_2}(\mathbf{q}) &= \left(\frac{\partial G_2}{\partial q^0}(\mathbf{q}), ..., \frac{\partial G_2}{\partial q^3}(\mathbf{q})\right) \\ &= \left(..., -4\sum_{i=1}^{m} \frac{\mathrm{sgn}(\langle \mathbf{q}, \mathbf{q}_i\rangle)||\mathbf{q}||^2 q_i^j - |\langle \mathbf{q}, \mathbf{q}_i\rangle| q^j}{||\mathbf{q}||^2\sqrt{||\mathbf{q}||^2||\mathbf{q}_i||^2 - \langle \mathbf{q}, \mathbf{q}_i\rangle^2}} \arccos\left(\frac{|\langle \mathbf{q}, \mathbf{q}_i\rangle|}{||\mathbf{q}|| \cdot ||\mathbf{q}_i||}\right), ...\right) \\ &= -4\sum_{i=1}^{m} \arccos\left(\frac{|\langle \mathbf{q}, \mathbf{q}_i\rangle|}{||\mathbf{q}|| \cdot ||\mathbf{q}_i||}\right) \frac{\mathrm{sgn}(\langle \mathbf{q}, \mathbf{q}_i\rangle)}{||\mathbf{q}||^2\sqrt{||\mathbf{q}||^2||\mathbf{q}_i||^2 - \langle \mathbf{q}, \mathbf{q}_i\rangle^2}}(\langle \mathbf{q}, \mathbf{q}\rangle \mathbf{q}_i - \langle \mathbf{q}, \mathbf{q}_i\rangle \mathbf{q}). \end{aligned}$$

Pentru a rezolva problema diferenţiabilităţii, vom elimina din domeniul de definiţie al funcţiei $G_2$ hiperplanele $\Pi_i = \{\mathbf{q} \in \mathbb{R}^4 \,|\, \langle \mathbf{q}, \mathbf{q}_i \rangle = 0\}$ şi dreptele $d_i$ care trec prin punctele $\mathbf{0}$ şi $\mathbf{q}_i$.

În acest caz, ecuaţia $\mathbf{v_0}_{|S^3}(\mathbf{q}) = \mathbf{0}$ este echivalentă cu

$$\begin{cases} \sum_{i=1}^{m} \frac{\operatorname{sgn}(\langle \mathbf{q}, \mathbf{q}_i \rangle) \arccos(|\langle \mathbf{q}, \mathbf{q}_i \rangle|)}{\sqrt{1 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2}} (\langle \mathbf{q}, \mathbf{q} \rangle \mathbf{q}_i - \langle \mathbf{q}, \mathbf{q}_i \rangle \mathbf{q}) = \mathbf{0} \\ \mathbf{q} \in S^3 \setminus \bigcup_{i=1}^{m} \Pi_i \\ \mathbf{q} \neq \pm \mathbf{q}_i \end{cases}.$$

Folosind aceleaşi argumente ca mai sus, ecuaţia $\mathbf{v_0}_{|S^3}(\mathbf{q}) = \mathbf{0}$ este echivalentă cu $\mathbf{P}_* \mathbf{v_0}_{|S^3} = \mathbf{0}$ care are următoarea formă

$$\begin{cases} \sum_{i=1}^{m} \frac{\operatorname{sgn}(\langle \mathbf{q}, \mathbf{q}_i \rangle) \arccos(|\langle \mathbf{q}, \mathbf{q}_i \rangle|)}{\sqrt{1 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2}} \Delta_i(\mathbf{q}) = \mathbf{0} \\ \mathbf{q} \in S^3 \setminus \bigcup_{i=1}^{m} \Pi_i \\ \mathbf{q} \neq \pm \mathbf{q}_i \end{cases}. \tag{5.5.12}$$

Expresia de mai sus arată de asemenea că $\mathbf{P}_* \mathbf{v_0}_{|S^3}$ este un câmp de vectori pe varietatea $SO(3)$.

Pentru a transforma ecuaţia (5.5.12) în rotaţii avem nevoie de următorul calcul:

$$\begin{aligned} \frac{\operatorname{sgn}(\langle \mathbf{q}, \mathbf{q}_i \rangle) \arccos(|\langle \mathbf{q}, \mathbf{q}_i \rangle|)}{\sqrt{1 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2}} &= \frac{\arccos(|\langle \mathbf{q}, \mathbf{q}_i \rangle|) \langle \mathbf{q}, \mathbf{q}_i \rangle}{|\langle \mathbf{q}, \mathbf{q}_i \rangle| \sqrt{1 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2}} = \frac{\frac{|\theta_i|}{2}}{|\cos(\frac{\theta_i}{2})| \sqrt{1 - \cos^2(\frac{\theta_i}{2})}} \langle \mathbf{q}, \mathbf{q}_i \rangle \\ &= \frac{|\theta_i|}{|\sin(\theta_i)|} \langle \mathbf{q}, \mathbf{q}_i \rangle = \frac{\theta_i}{\sin(\theta_i)} \langle \mathbf{q}, \mathbf{q}_i \rangle, \end{aligned}$$

unde am folosit proprietatea că $\mathbf{q} \in S^3 \setminus \bigcup_{i=1}^{m} \Pi_i$ care implică $\langle \mathbf{q}, \mathbf{q}_i \rangle \neq 0$. Folosind (5.5.10) obţinem sistemul echivalent în rotaţii, care corespunde pasului **(iv')** al algoritmului de încorporare:

$$\sum_{i=1}^{m} (\mathbf{R}^{\mathbf{q}_i T} \mathbf{R}^{\mathbf{q}} - \mathbf{R}^{\mathbf{q} T} \mathbf{R}^{\mathbf{q}_i}) \frac{\theta_i}{\sin \theta_i} = \mathbf{0} \Leftrightarrow \sum_{i=1}^{m} \operatorname{Log}(\mathbf{R}^{\mathbf{q}_i T} \mathbf{R}^{\mathbf{q}}) = \mathbf{0}. \tag{5.5.13}$$

Ecuaţia de mai sus a fost obţinută în [177] ca fiind o caracterizare a mediei geometrice. Acest lucru era de aşteptat, din moment ce facem medierea pentru aceeaşi funcţie de cost prin două metode diferite, care în mod natural duc la acelaşi rezultat. Să observăm de asemenea că funcţia de cost $G_{2_{SO(3)}}$ nu este definită pentru unghiurile $\theta_i = \pm\pi$ sau echivalent $\langle \mathbf{q}, \mathbf{q}_i \rangle = 0$, situaţie în care funcţia prelungită $G_2$ nu este diferenţiabilă.

**III.** O altă distanţă folosită pe grupul Lie $SO(3)$ este dată prin $d_3(\mathbf{R}_1, \mathbf{R}_2) = 1 - \frac{1}{2}\sqrt{\operatorname{tr}(\mathbf{R}_1^T \mathbf{R}_2) + 1}$. Această distanţă apare în [129], unde dependenţa funcţională dintre $d_3$ şi $d_2$ este de asemenea dată. Observăm că această funcţie distanţă pe

$SO(3)$ poate fi obţinută din pseudodistanţa $d_3 : S^3 \times S^3 \to \mathbb{R}$, $d_3(\mathbf{q}_1, \mathbf{q}_2) = 1 - |\langle \mathbf{q}_1, \mathbf{q}_2 \rangle|$. Funcţia de cost $G_{3_{SO(3)}}(\mathbf{R}) = \sum_{i=1}^{m} d_3^2(\mathbf{R}, \mathbf{R}_i)$ este liftată în funcţia de cost

$$G_{3_{S^3}}(\mathbf{q}) = \sum_{i=1}^{m} (1 - |\langle \mathbf{q}, \mathbf{q}_i \rangle|)^2.$$

Funcţia prelungită este $G_3 : \mathbb{R}^4 \to \mathbb{R}$,

$$G_3(\mathbf{q}) = \sum_{i=1}^{m} (1 - |\langle \mathbf{q}, \mathbf{q}_i \rangle|)^2.$$

Coeficienţii $1$-formei diferenţiale $dG_3$ sunt daţi prin:

$$\overline{dG_3}(\mathbf{q}) = -2 \sum_{i=1}^{m} (1 - |\langle \mathbf{q}, \mathbf{q}_i \rangle|) \mathrm{sgn}(\langle \mathbf{q}, \mathbf{q}_i \rangle) \mathbf{q}_i,$$

unde, pentru a avea diferenţiabilitate, ne restricţionăm la mulţimea deschisă $\mathbf{q} \in S^3 \setminus \bigcup_{i=1}^{m} \Pi_i$ (mulţimea $\Pi_i$ este hiperplanul determinat de $\mathbf{q}_i$ ca mai sus).

Sistemul de ecuaţii (5.5.5) devine

$$\begin{cases} \sum_{i=1}^{m} (1 - |\langle \mathbf{q}, \mathbf{q}_i \rangle|) \mathrm{sgn}(\langle \mathbf{q}, \mathbf{q}_i \rangle)(\langle \mathbf{q}, \mathbf{q} \rangle \mathbf{q}_i - \langle \mathbf{q}, \mathbf{q}_i \rangle \mathbf{q}) = \mathbf{0} \\ \mathbf{q} \in S^3 \setminus \bigcup_{i=1}^{m} \Pi_i \end{cases}, \tag{5.5.14}$$

iar acesta reprezintă ecuaţia $\mathbf{v_0} = \mathbf{0}$ restricţionată la mulţimea deschisă $S^3 \setminus \bigcup_{i=1}^{m} \Pi_i$ pe sfera $S^3$.

Ca în cazurile anterioare, după aplicarea operatorului de proiecţie $\mathbf{P}_*$, generăm câmpul de vectori $\mathbf{P}_* \mathbf{v_0}_{|S^3}$ şi, prin urmare, obţinem sistemul de ecuaţii echivalent:

$$\begin{cases} \sum_{i=1}^{m} (1 - |\langle \mathbf{q}, \mathbf{q}_i \rangle|) \mathrm{sgn}(\langle \mathbf{q}, \mathbf{q}_i \rangle) \Delta_i(\mathbf{q}) = \mathbf{0} \\ \mathbf{q} \in S^3 \setminus \bigcup_{i=1}^{m} \Pi_i \end{cases}. \tag{5.5.15}$$

Transformând sistemul de mai sus în rotaţii, după calcule algebrice folosind (5.5.10) şi (5.5.7), obţinem ecuaţia corespunzătoare pasului **(iv')** al algoritmului de încorporare

$$\sum_{i=1}^{m} \left( \frac{2}{\sqrt{\mathrm{tr}(\mathbf{R}^{\mathbf{q}^T} \mathbf{R}^{\mathbf{q}_i}) + 1}} - 1 \right) (\mathbf{R}^{\mathbf{q}_i T} \mathbf{R}^{\mathbf{q}} - \mathbf{R}^{\mathbf{q}^T} \mathbf{R}^{\mathbf{q}_i}) = \mathbf{0}.$$

**IV.** În continuare, vom aplica tehnicile descrise mai sus pentru o funcţie de cost care este de tip $L^p$ cu $p \geq 1$, a se vedea [108]. Pornim cu distanţa $d_1 : SO(3) \times$

$SO(3) \to \mathbb{R}_+$, $d_1(\mathbf{R}_1, \mathbf{R}_2) = ||\mathbf{R}_1 - \mathbf{R}_2||_F$. Funcţia de cost asociată de tip medie $L^p$ este $G_{(p)_{SO(3)}} : SO(3) \to \mathbb{R}$,

$$G_{(p)_{SO(3)}}(\mathbf{R}) = \sum_{i=1}^{m} ||\mathbf{R} - \mathbf{R}_i||_F^p.$$

Funcţia de cost liftată $G_{(p)_{S^3}} : S^3 \to \mathbb{R}$ este

$$G_{(p)_{S^3}}(\mathbf{q}) = 8^{\frac{p}{2}} \sum_{i=1}^{m} (1 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2)^{\frac{p}{2}}.$$

Alegem funcţia prelungită $G_{(p)} : \mathbb{R}^4 \to \mathbb{R}$,

$$G_{(p)}(\mathbf{q}) = 8^{\frac{p}{2}} \sum_{i=1}^{m} (||\mathbf{q}||^2 \cdot ||\mathbf{q}_i||^2 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2)^{\frac{p}{2}}.$$

Prin calcul direct obţinem:

$$\overline{dG_{(p)}}(\mathbf{q}) = -p8^{\frac{p}{2}} \sum_{i=1}^{m} (||\mathbf{q}||^2 \cdot ||\mathbf{q}_i||^2 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2)^{\frac{p}{2}-1} (\langle \mathbf{q}, \mathbf{q}_i \rangle \mathbf{q}_i - ||\mathbf{q}_i||^2 \mathbf{q}).$$

Pentru a avea diferenţiabilitate pentru funcţia $G_{(p)}$, pentru $p \in [1, 2)$ trebuie să eliminăm din domeniul de definiţie dreptele $d_i$ care trec prin punctele $\mathbf{0}$ şi $\mathbf{q}_i$. Ca în cazul funcţiei $G_1$, sistemul (5.5.5) corespunzând funcţiei $G_{(p)}$ devine:

$$\begin{cases} \sum_{i=1}^{m} \langle \mathbf{q}, \mathbf{q}_i \rangle (||\mathbf{q}||^2 \cdot ||\mathbf{q}_i||^2 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2)^{\frac{p}{2}-1} (\langle \mathbf{q}, \mathbf{q} \rangle \mathbf{q}_i - \langle \mathbf{q}, \mathbf{q}_i \rangle \mathbf{q}) = \mathbf{0} \\ \langle \mathbf{q}, \mathbf{q} \rangle = 1 \\ \mathbf{q} \neq \pm \mathbf{q}_i, \text{ pentru cazul când } p \in [1, 2) \end{cases}. \tag{5.5.16}$$

Folosind aceleaşi argumente ca mai sus, în cazul funcţiei $G_{(p)}$, ecuaţia $\mathbf{v}_{\mathbf{0}|S^3}(\mathbf{q}) = \mathbf{0}$ este echivalentă cu $\mathbf{P}_* \mathbf{v}_{\mathbf{0}|S^3} = \mathbf{0}$, care are următoarea formă

$$\begin{cases} \sum_{i=1}^{m} (1 - \langle \mathbf{q}, \mathbf{q}_i \rangle^2)^{\frac{p}{2}-1} \langle \mathbf{q}, \mathbf{q}_i \rangle \Delta_i(\mathbf{q}) = \mathbf{0} \\ \langle \mathbf{q}, \mathbf{q} \rangle = 1 \\ \mathbf{q} \neq \pm \mathbf{q}_i, \text{ pentru cazul când } p \in [1, 2) \end{cases}. \tag{5.5.17}$$

Folosind din nou (5.5.7) şi (5.5.10) obţinem, conform pasului **(iv')** al algoritmului de încorporare, ecuaţiile în rotaţii care dau punctele critice ale funcţiei de cost $G_{(p)_{SO(3)}}$,

$$\sum_{i=1}^{m} (3 - \operatorname{tr}(\mathbf{R}^{\mathbf{q}^T} \mathbf{R}^{\mathbf{q}_i}))^{\frac{p}{2}-1} (\mathbf{R}^{\mathbf{q}_i{}^T} \mathbf{R}^{\mathbf{q}} - \mathbf{R}^{\mathbf{q}^T} \mathbf{R}^{\mathbf{q}_i}) = \mathbf{0},$$

cu condiţia suplimentară $\mathbf{R}^{\mathbf{q}} \neq \mathbf{R}^{\mathbf{q}_i}$ pentru cazul când $p \in [1, 2)$.

### 5.5.4 Medierea pe $SO(n)$

Acum vom lucra pe grupul ortogonal al matricilor de ordinul $n$ pe $\mathbb{R}$, notat $SO(n)$. Grupul ortogonal $O(n)$ este un subgrup al grupului liniar general $GL(n)$ definit prin

$$O(n) = \{\mathbf{R} \in GL(n) | \mathbf{R}^T\mathbf{R} = \mathbf{I}_n\}.$$

Ca spaţiu topologic, are două componente conexe, iar componenta conţinând identitatea este chiar grupul ortogonal special $SO(n)$. Ca în cazul secţiunii anterioare, se poate demonstra că $O(n)$ poate fi privit ca preimaginea valorii regulate $\mathbf{I}_n$ pentru funcţia $\mathbf{F} : GL(n) \to S(n)$, unde $S(n)$ este spaţiul vectorial al matricilor simetrice $n \times n$ şi $\mathbf{F}(\mathbf{A}) = \mathbf{A}^T\mathbf{A}$. Avem astfel cadrul pentru prima subsecţiune, în care varietatea $N$ este $O(n)$, iar spaţiul ambient $(M, g)$ este spaţiul euclidian $\mathbb{R}^{n^2}$. Deci acest caz general corespunde scufundării în $\mathbb{R}^9$ a lui $SO(3)$. Suntem de asemenea în cazul în care $S = N$ şi $\mathbf{P}$ este identitatea.

Un punct $\mathbf{x} = (x^1, \ldots, x^{n^2}) \in \mathbb{R}^{n^2}$ corespunde rotaţiei din $O(n)$:

$$\mathbf{R} = \begin{pmatrix} x^1 & x^2 & \ldots & x^n \\ x^{n+1} & x^{n+2} & \ldots & x^{2n} \\ \ldots & \ldots & \ldots & \ldots \\ x^{n^2-n+1} & x^{n^2-n+2} & \ldots & x^{n^2} \end{pmatrix}.$$

Condiţia $\mathbf{R}^T\mathbf{R} = \mathbf{I}_n$ devine

$$\begin{pmatrix} (x^1)^2 + \ldots + (x^n)^2 & x^1x^{n+1} + \ldots + x^nx^{2n} & \ldots & x^1x^{n^2-n+1} + \ldots + x^nx^{n^2} \\ x^1x^{n+1} + \ldots + x^nx^{2n} & (x^{n+1})^2 + \ldots + (x^{2n})^2 & \ldots & x^{n+1}x^{n^2-n+1} + \ldots + x^{2n}x^{n^2} \\ \ldots & \ldots & \ldots & \ldots \\ x^1x^{n^2-n+1} + \ldots + x^nx^{n^2} & x^{n+1}x^{n^2-n+1} + \ldots + x^{2n}x^{n^2} & \ldots & (x^{n^2-n+1})^2 + \ldots + (x^{n^2})^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{pmatrix}$$

Definim funcţia $\mathbf{F} : \mathbb{R}^{n^2} \to \mathbb{R}^{\frac{n(n+1)}{2}}$, $\mathbf{F} = (F_1, F_2, \ldots, F_{\frac{n(n+1)}{2}})$, unde

$$\begin{aligned} F_1(x^1, \ldots, x^{n^2}) &= (x^1)^2 + \ldots + (x^n)^2 \\ F_2(x^1, \ldots, x^{n^2}) &= x^1x^{n+1} + \ldots + x^nx^{2n} \\ \ldots \quad &\ldots \quad \ldots \\ F_n(x^1, \ldots, x^{n^2}) &= x^1x^{n^2-n+1} + \ldots + x^nx^{n^2} \\ F_{n+1}(x^1, \ldots, x^{n^2}) &= (x^{n+1})^2 + \ldots + (x^{2n})^2 \\ \ldots \quad &\ldots \quad \ldots \\ F_{2n-1}(x^1, \ldots, x^{n^2}) &= x^{n+1}x^{n^2-n+1} + \ldots + x^{2n}x^{n^2} \\ \ldots \quad &\ldots \quad \ldots \\ F_{\frac{n(n+1)}{2}}(x^1, \ldots, x^{n^2}) &= (x^{n^2-n+1})^2 + \ldots + (x^{n^2})^2. \end{aligned}$$

Prin urmare $O(n)$ poate fi identificat cu foaia regulată

$$O(n) = \mathbf{F}^{-1}(1, 0, \ldots, 0, 0, 1, \ldots, 0, \ldots, 0, \ldots 0, 1).$$

Pentru o simplificare a calculelor, introducem funcţiile vectoriale $\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_n : \mathbb{R}^{n^2} \to \mathbb{R}^n$, $\overline{u}_i(x^1, \ldots, x^{n^2}) = (x^{n(i-1)+1}, x^{n(i-1)+2}, \ldots, x^{n(i-1)+n})$, $\forall i = \overline{1, n}$. Pentru a calcula

câmpul de vectori standard de control $\mathbf{v_0}$, trebuie să determinăm matricea asociată 2-tensorului simetric contravariant $\mathbf{T}$. Pentru aceasta, trebuie să determinăm matricea Gram $\Sigma^{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}$, şi anume

$$\Sigma^{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,F_{\frac{n(n+1)}{2}})} = \begin{pmatrix} \langle \nabla F_1, \nabla F_1 \rangle & \ldots & \langle \nabla F_{\frac{n(n+1)}{2}}, \nabla F_1 \rangle \\ \vdots & \ldots & \vdots \\ \langle \nabla F_1, \nabla F_{\frac{n(n+1)}{2}} \rangle & \ldots & \langle \nabla F_{\frac{n(n+1)}{2}}, \nabla F_{\frac{n(n+1)}{2}} \rangle \end{pmatrix}.$$

Utilizând notaţiile de mai sus, rezultă imediat că

$$\begin{aligned} F_1(x^1,\ldots,x^{n^2}) &= \langle \overline{u}_1, \overline{u}_1 \rangle \\ F_2(x^1,\ldots,x^{n^2}) &= \langle \overline{u}_1, \overline{u}_2 \rangle \\ \ldots \quad \ldots &\quad \ldots \\ F_n(x^1,\ldots,x^{n^2}) &= \langle \overline{u}_1, \overline{u}_n \rangle \\ F_{n+1}(x^1,\ldots,x^{n^2}) &= \langle \overline{u}_2, \overline{u}_2 \rangle \\ \ldots \quad \ldots &\quad \ldots \\ F_{2n-1}(x^1,\ldots,x^{n^2}) &= \langle \overline{u}_2, \overline{u}_n \rangle \\ \ldots \quad \ldots &\quad \ldots \\ F_{\frac{n(n+1)}{2}}(x^1,\ldots,x^{n^2}) &= \langle \overline{u}_n, \overline{u}_n \rangle. \end{aligned}$$

şi prin urmare

$$\begin{aligned} \nabla F_1(x^1,\ldots,x^{n^2}) &= (2\overline{u_1}, \overline{0}, \ldots, \overline{0}) \\ \nabla F_2(x^1,\ldots,x^{n^2}) &= (\overline{u_2}, \overline{u_1}, \ldots, \overline{0}) \\ \ldots \quad \ldots &\quad \ldots \\ \nabla F_n(x^1,\ldots,x^{n^2}) &= (\overline{u_n}, \overline{0}, \ldots, \overline{u_1}) \\ \nabla F_{n+1}(x^1,\ldots,x^{n^2}) &= (\overline{0}, 2\overline{u_2}, \ldots, \overline{0}) \\ \ldots \quad \ldots &\quad \ldots \\ \nabla F_{2n-1}(x^1,\ldots,x^{n^2}) &= (\overline{0}, \overline{u_n}, \ldots, \overline{u_2}) \\ \ldots \quad \ldots &\quad \ldots \\ \nabla F_{\frac{n(n+1)}{2}}(x^1,\ldots,x^{n^2}) &= (\overline{0}, \overline{0}, \ldots, 2\overline{u_n}). \end{aligned}$$

Matricea Gram devine astfel

$$\Sigma^{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,F_{\frac{n(n+1)}{2}})} = \begin{pmatrix} 4F_1 & \ldots & 0 \\ \vdots & \ldots & \vdots \\ 0 & \ldots & 4F_{\frac{n(n+1)}{2}} \end{pmatrix}.$$

Trebuie să calculăm câmpul de vectori $\mathbf{v_0}$ doar restricţionat la $O(n)$ şi prin urmare trebuie să calculăm matricea asociată lui $\mathbf{T}$ doar pentru elemente din $O(n)$, pentru care au loc relaţiile $F_1(x^1,\ldots,x^{n^2}) = F_{n+1}(x^1,\ldots,x^{n^2}) = \ldots = F_{\frac{n(n+1)}{2}}(x^1,\ldots,x^{n^2}) = 1$ şi $F_j(x^1,\ldots,x^{n^2}) = 0$, pentru $j \in \left\{1,\ldots,\frac{n(n+1)}{2}\right\} \setminus K$, unde am notat

$$K := \left\{ (k-1)\left(n+1-\frac{k}{2}\right)+1 \,\middle|\, k = \overline{1,n} \right\}.$$

Matricea Gram calculată în puncte din $O(n)$ devine, pentru $\forall \mathbf{R} \in O(n)$, următoarea:

$$\left(\Sigma^{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}(\mathbf{R})\right)_{1\leq i,j\leq \frac{n(n+1)}{2}} = \begin{cases} 4, & i=j\in K \\ 2, & i=j\in \left\{1,\ldots,\frac{n(n+1)}{2}\right\}\setminus K \\ 0, & i\neq j \end{cases}.$$

Calcule elementare arată acum că pentru orice $\mathbf{R}\in O(n)$, avem că

$$\det \Sigma^{(F_1,\ldots,\hat{F}_j,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,\hat{F}_i,\ldots,F_{\frac{n(n+1)}{2}})}(\mathbf{R}) = 0$$

pentru $\forall i\neq j$, $i,j=\overline{1,\frac{n(n+1)}{2}}$, $\det \Sigma^{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,F_{\frac{n(n+1)}{2}})}(\mathbf{R}) = 2^{n^2}$ şi

$$\begin{aligned}\det \Sigma^{(\hat{F}_1,\ldots,F_{\frac{n(n+1)}{2}})}_{(\hat{F}_1,\ldots,F_{\frac{n(n+1)}{2}})}(\mathbf{R}) &= \det \Sigma^{(F_1,\ldots,\hat{F}_{n+1}\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,\hat{F}_{n+1},\ldots,F_{\frac{n(n+1)}{2}})}(\mathbf{R}) = \ldots \\ &= \det \Sigma^{(F_1,\ldots,\hat{F}_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,\hat{F}_{\frac{n(n+1)}{2}})}(\mathbf{R}) = 2^{n^2-2},\end{aligned}$$

$$\det \Sigma^{(F_1,\ldots,\hat{F}_j,\ldots,F_{\frac{n(n+1)}{2}})}_{(F_1,\ldots,\hat{F}_j,\ldots,F_{\frac{n(n+1)}{2}})}(\mathbf{R}) = 2^{n^2-1}, \forall j\in\left\{1,\ldots,\frac{n(n+1)}{2}\right\}\setminus K.$$

Luând în considerare faptul că pe $\mathbb{R}^{n^2}$ avem metrica euclidiană, avem că matricea asociată 2-tensorului contravariant $g^{-1}$ este matricea identitate $\mathbf{I}_{n^2}$. Mai mult, vom introduce următoarea notaţie

$$k(\mathbf{R}) = \overline{u}_1(\mathbf{R})\otimes\overline{u}_1(\mathbf{R}) + \overline{u}_2(\mathbf{R})\otimes\overline{u}_2(\mathbf{R}) + \ldots + \overline{u}_n(\mathbf{R})\otimes\overline{u}_n(\mathbf{R}) - 2\mathbf{I}_n, \forall \mathbf{R}\in O(n).$$

Folosind această notaţie, matricea asociată 2-tensorului contravariant $\mathbf{T}$ calculată în puncte din $O(n)$ devine

$$\mathbf{T}(\mathbf{R}) = -2^{n^2-1}\begin{pmatrix} k(\mathbf{R})+\overline{u_1}(\mathbf{R})\otimes\overline{u_1}(\mathbf{R}) & \overline{u_1}(\mathbf{R})\otimes\overline{u_2}(\mathbf{R}) & \ldots & \overline{u_1}(\mathbf{R})\otimes\overline{u_n}(\mathbf{R}) \\ \overline{u_1}(\mathbf{R})\otimes\overline{u_2}(\mathbf{R}) & k(\mathbf{R})+\overline{u_2}(\mathbf{R})\otimes\overline{u_2}(\mathbf{R}) & \ldots & \overline{u_2}(\mathbf{R})\otimes\overline{u_n}(\mathbf{R}) \\ \ldots & \ldots & \ldots & \ldots \\ \overline{u_1}(\mathbf{R})\otimes\overline{u_n}(\mathbf{R}) & \overline{u_2}(\mathbf{R})\otimes\overline{u_n}(\mathbf{R}) & \ldots & k(\mathbf{R})+\overline{u_n}(\mathbf{R})\otimes\overline{u_n}(\mathbf{R}) \end{pmatrix}.$$

Dacă $\omega(\mathbf{R}) = \omega_1(\mathbf{R})dx^1 + \ldots + \omega_{n^2}(\mathbf{R})dx^{n^2}$ este o 1-formă pe $\mathbb{R}^{n^2}$, atunci

$$\mathbf{i}_\omega\mathbf{T}(\mathbf{R}) = -2^{n^2-1}(\mathbf{R}\overline{\omega}(\mathbf{R})^T\mathbf{R} - \overline{\omega}(\mathbf{R})), \tag{5.5.18}$$

unde am făcut notaţia $\overline{\omega}(\mathbf{R}) = \left(\omega_{n(k-1)+l}(\mathbf{R})\right)_{1\leq k,l\leq n}$. Se observă că aceste calcule sunt similare celor din cazul $O(3)$, reprezentând o generalizare a lor.

**I.** Din nou vom lucra pe $SO(n)$, deoarece acest grup prezintă interes din punctul de vedere al aplicaţiilor practice. De asemenea, vom folosi aceleaşi funcţii de cost, pentru motivele explicate mai sus, în cazul $SO(3)$. Pe grupul Lie $SO(n)$ vom considera funcţia distanţă $d_1 : SO(n)\times SO(n)\to\mathbb{R}_+$, $d_1(\mathbf{R}_1,\mathbf{R}_2) = ||\mathbf{R}_1-\mathbf{R}_2||_F$, unde $||\cdot||_F$ este norma Frobenius. Funcţia de cost asociată este $G_{1_{SO(n)}} : SO(n)\to\mathbb{R}$,

$$G_{1_{SO(n)}}(\mathbf{R}) = \sum_{i=1}^{m}||\mathbf{R}-\mathbf{R}_i||_F^2,$$

unde $\{\mathbf{R}_1, ..., \mathbf{R}_m\}$ sunt matricile de rotaţie cărora dorim să le facem media, notate $\mathbf{R}_i = (x_i^{n(k-1)+l})_{1\leq k,l\leq n}$. Din $\mathbf{R} = (x^{n(k-1)+l})_{1\leq k,l\leq n}$, putem scrie

$$G_{1_{SO(n)}}(\mathbf{R}) = \sum_{i=1}^{m} ||\mathbf{R} - \mathbf{R}_i||_F^2 = \sum_{i=1}^{m} \mathrm{tr}[(\mathbf{R} - \mathbf{R}_i)^T(\mathbf{R} - \mathbf{R}_i)] = \sum_{i=1}^{m} \langle \mathbf{x} - \mathbf{x}_i, \mathbf{x} - \mathbf{x}_i \rangle.$$

Funcţia prelungită este $G_1 : \mathbb{R}^{n^2} \to \mathbb{R}$,

$$G_1(\mathbf{x}) = \sum_{i=1}^{m} \langle \mathbf{x} - \mathbf{x}_i, \mathbf{x} - \mathbf{x}_i \rangle,$$

aceeaşi ca în cazul $SO(3)$. Calculând $1$-forma diferenţială $dG_1$ obţinem:

$$\overline{dG_1}(\mathbf{R}) = \left( \frac{\partial G_1}{\partial x^{n(k-1)+l}} \right)_{1\leq k,l\leq n} = 2\sum_{i=1}^{m} (\mathbf{R} - \mathbf{R}_i) = 2m(\mathbf{R} - \overline{\mathbf{R}}),$$

unde am notat $\overline{\mathbf{R}} = \dfrac{\sum_{i=1}^{m} \mathbf{R}_i}{m}$. Folosind (5.5.18), unde $1$-forma $\omega$ este $dG_1$, sistemul de ecuaţii (5.5.5) devine:

$$\left\{ \begin{array}{l} \mathbf{R}(\mathbf{R} - \overline{\mathbf{R}})^T\mathbf{R} - (\mathbf{R} - \overline{\mathbf{R}}) = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} \mathbf{R}\overline{\mathbf{R}}^T\mathbf{R} - \overline{\mathbf{R}} = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array} \right. \tag{5.5.19}$$

$$\Leftrightarrow \left\{ \begin{array}{l} \overline{\mathbf{R}}^T\mathbf{R} - \mathbf{R}^T\overline{\mathbf{R}} = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array} \right. .$$

Regăsim formula pentru media aritmetică din [177], care de asemenea este făcută pentru cazul general $SO(n)$. Articolul [177] foloseşte însă pentru a obţine această formulă derivata covariantă, adică o metodă mai laborioasă decât algoritmul de încorporare prezentat mai sus.

**II.** În continuare, vom lua în considerare distanţa geodezică pe grupul Lie $SO(n)$: pentru două rotaţii $\mathbf{R}_1, \mathbf{R}_2 \in SO(n)$, $d_2(\mathbf{R}_1, \mathbf{R}_2) = ||\mathrm{Log}(\mathbf{R}_1^T\mathbf{R}_2)||_F = \sqrt{2}|\theta|$, unde $\theta \in (-\pi, \pi)$ este unghiul dintre rotaţiile $\mathbf{R}_1$ şi $\mathbf{R}_2$. Funcţia de cost asociată este $G_{2_{SO(n)}} : SO(n) \to \mathbb{R}$,

$$G_{2_{SO(n)}}(\mathbf{R}) = \sum_{i=1}^{m} ||\mathrm{Log}(\mathbf{R}_i^T\mathbf{R})||_F^2,$$

unde $\{\mathbf{R}_1, ..., \mathbf{R}_m\}$ sunt matricile de rotaţie cărora dorim să le facem media, pe care le notăm $\mathbf{R}_i = (x_i^{n(k-1)+l})_{1\leq k,l\leq n}$. Din $\mathbf{R} = (x^{n(k-1)+l})_{1\leq k,l\leq n}$, putem scrie

$$\begin{aligned} G_{2_{SO(n)}}(\mathbf{R}) &= \sum_{i=1}^{m} ||\mathrm{Log}(\mathbf{R}_i^T\mathbf{R})||_F^2 = 2\sum_{i=1}^{m} \theta_i^2 \\ &= 2\sum_{i=1}^{m} \arccos^2\left( \frac{\mathrm{tr}(\mathbf{R}_i^T\mathbf{R}) - 1}{2} \right) \\ &= 2\sum_{i=1}^{m} \arccos^2\left( \frac{\langle \mathbf{x}_i, \mathbf{x} \rangle - 1}{2} \right). \end{aligned}$$

Funcţia prelungită este $G_2 : \mathbb{R}^{n^2} \to \mathbb{R}$,

$$G_2(\mathbf{x}) = 2\sum_{i=1}^{m} \arccos^2\left(\frac{\langle \mathbf{x}_i, \mathbf{x}\rangle - 1}{2}\right).$$

Coeficienţii $1$-formei diferenţiale $dG_2$ sunt daţi de:

$$\overline{dG_2}(\mathbf{R}) = \left(\frac{\partial G_2}{\partial x^{n(k-1)+l}}\right)_{1\le k,l\le n} = -4\sum_{i=1}^{m} \mathbf{R}_i \frac{\theta_i}{\sin\theta_i}.$$

Sistemul de ecuaţii (5.5.5) devine în acest caz:

$$\left\{\begin{array}{l} \sum\limits_{i=1}^{m} \mathbf{R}\left(\mathbf{R}_i \frac{\theta_i}{\sin\theta_i}\right)^T \mathbf{R} - \mathbf{R}_i \frac{\theta_i}{\sin\theta_i} = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} \sum\limits_{i=1}^{m} (\mathbf{R}\mathbf{R}_i^T\mathbf{R} - \mathbf{R}_i)\frac{\theta_i}{\sin\theta_i} = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array}\right. \tag{5.5.20}$$

$$\Leftrightarrow \left\{\begin{array}{l} \sum\limits_{i=1}^{m} \mathrm{Log}(\mathbf{R}_i^T\mathbf{R}) = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array}\right. .$$

Şi aici regăsim caracterizarea mediei geometrice din [177], făcută pentru cazul general $SO(n)$, unde din nou este folosită o metodă mult mai greoaie pentru calculul acestei medii.

**III.** O altă distanţă folosită pe grupul Lie $SO(n)$ este dată prin $d_3(\mathbf{R}_1, \mathbf{R}_2) = 1 - \frac{1}{2}\sqrt{\mathrm{tr}(\mathbf{R}_1^T\mathbf{R}_2) + 1}$. Această distanţă apare în [129] unde dependenţa funcţională dintre $d_3$ şi $d_2$ este de asemenea dată. Funcţia de cost $G_{3_{SO(n)}}$ este:

$$G_{3_{SO(n)}}(\mathbf{R}) = \sum_{i=1}^{m}\left(1 - \frac{1}{2}\sqrt{\mathrm{tr}(\mathbf{R}_i^T\mathbf{R}) + 1}\right)^2 = \sum_{i=1}^{m}\left(1 - \frac{1}{2}\sqrt{\langle \mathbf{x}_i, \mathbf{x}\rangle + 1}\right)^2,$$

unde $\{\mathbf{R}_1, ..., \mathbf{R}_m\}$ sunt matricile de rotaţie cărora dorim să le facem media şi am considerat $\mathbf{R}_i = (x_i^{n(k-1)+l})_{1\le k,l\le n}$ şi $\mathbf{R} = (x^{n(k-1)+l})_{1\le k,l\le n}$.

Funcţia prelungită este $G_3 : \mathbb{R}^{n^2} \to \mathbb{R}$,

$$G_3(\mathbf{x}) = \sum_{i=1}^{m}\left(1 - \frac{1}{2}\sqrt{\langle \mathbf{x}_i, \mathbf{x}\rangle + 1}\right)^2.$$

Coeficienţii $1$-formei diferenţiale $dG_3$ sunt daţi prin:

$$\overline{dG_3}(\mathbf{R}) = \left(\frac{\partial G_3}{\partial x^{n(k-1)+l}}\right)_{1\le k,l\le n} = 2\sum_{i=1}^{m} \mathbf{R}_i\left(\frac{2}{\sqrt{\mathrm{tr}(\mathbf{R}_i^T\mathbf{R}) + 1}} - 1\right).$$

Transformând sistemul de mai sus în rotaţii, după calcule algebrice folosind (5.5.10) şi (5.5.7), obţinem

$$\left\{\begin{array}{l} \sum\limits_{i=1}^{m} \mathbf{R}\left[\mathbf{R}_i\left(\frac{2}{\sqrt{\mathrm{tr}(\mathbf{R}_i^T\mathbf{R}) + 1}} - 1\right)\right]^T \mathbf{R} - \mathbf{R}_i\left(\frac{2}{\sqrt{\mathrm{tr}(\mathbf{R}_i^T\mathbf{R}) + 1}} - 1\right) = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array}\right. \tag{5.5.21}$$

$$\Leftrightarrow \left\{ \begin{array}{l} \sum\limits_{i=1}^{m} \left( \dfrac{2}{\sqrt{\mathrm{tr}(\mathbf{R}_i^T\mathbf{R})+1}} - 1 \right) (\mathbf{R}_i^T\mathbf{R} - \mathbf{R}^T\mathbf{R}_i) = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array} \right. .$$

**IV.** În continuare, vom aplica tehnicile descrise mai sus pentru o funcţie de cost care este de tip $L^p$, cu $p \geq 1$. Pornim cu distanţa $d_1 : SO(n) \times SO(n) \to \mathbb{R}_+$, $d_1(\mathbf{R}_1, \mathbf{R}_2) = ||\mathbf{R}_1 - \mathbf{R}_2||_F$. Dacă $\{\mathbf{R}_1, ..., \mathbf{R}_m\}$ sunt matricile de rotaţie cărora dorim să le facem media, funcţia de cost de tip medie $L^p$ asociată este $G_{(p)_{SO(n)}} : SO(n) \to \mathbb{R}$,

$$G_{(p)_{SO(n)}}(\mathbf{R}) = \sum_{i=1}^{m} ||\mathbf{R} - \mathbf{R}_i||_F^p.$$

Aceasta se poate scrie, ţinând cont că $\mathbf{R}_i = (x_i^{n(k-1)+l})_{1\leq k,l\leq n}$ şi $\mathbf{R} = (x^{n(k-1)+l})_{1\leq k,l\leq n}$ astfel:

$$G_{4_{SO(n)}}(\mathbf{R}) = \sum_{i=1}^{m} ||\mathbf{R} - \mathbf{R}_i||_F^p = \sum_{i=1}^{m} (\mathrm{tr}[(\mathbf{R} - \mathbf{R}_i)^T(\mathbf{R} - \mathbf{R}_i)])^{\frac{p}{2}} = \sum_{i=1}^{m} \langle \mathbf{x} - \mathbf{x}_i, \mathbf{x} - \mathbf{x}_i \rangle^{\frac{p}{2}}.$$

Funcţia prelungită este $G_{(p)} : \mathbb{R}^{n^2} \to \mathbb{R}$,

$$G_{(p)}(\mathbf{x}) = \sum_{i=1}^{m} \langle \mathbf{x} - \mathbf{x}_i, \mathbf{x} - \mathbf{x}_i \rangle^{\frac{p}{2}}.$$

Coeficienţii 1-formei diferenţiale $dG_4$ sunt daţi prin:

$$\overline{dG_{(p)}}(\mathbf{R}) = \left( \frac{\partial G_{(p)}}{\partial x^{n(k-1)+l}} \right)_{1\leq k,l\leq n} = p2^{\frac{p}{2}-1} \sum_{i=1}^{m} (\mathbf{R} - \mathbf{R}_i)(n - \mathrm{tr}(\mathbf{R}_i^T\mathbf{R}))^{\frac{p}{2}-1}.$$

În final sistemul devine

$$\left\{ \begin{array}{l} \sum\limits_{i=1}^{m} \mathbf{R}[(\mathbf{R} - \mathbf{R}_i)(n - \mathrm{tr}(\mathbf{R}_i^T\mathbf{R}))^{\frac{p}{2}-1}]^T\mathbf{R} - (\mathbf{R} - \mathbf{R}_i)(n - \mathrm{tr}(\mathbf{R}_i^T\mathbf{R}))^{\frac{p}{2}-1} = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array} \right.$$

$$\Leftrightarrow \left\{ \begin{array}{l} \sum\limits_{i=1}^{m} (n - \mathrm{tr}(\mathbf{R}_i^T\mathbf{R}))^{\frac{p}{2}-1}(\mathbf{R}_i^T\mathbf{R} - \mathbf{R}^T\mathbf{R}_i) = \mathbf{0} \\ \mathbf{R}^T\mathbf{R} = \mathbf{I}_n \end{array} \right. .$$

## 5.6 Concluzii

Prezentul capitol a prezentat mai multe generalizări posibile ale reţelelor neuronale Clifford. Astfel,

- Secţiunea 5.1 a prezentat **reţelele neuronale cu valori vectori tridimensionali**, mai exact, bazate pe produsul vectorial din spaţiul tridimensional. Aceste reţele nu sunt o generalizare directă a reţelelor Clifford, ci mai degrabă reprezintă o încercare paralelă de definire a unor reţele neuronale multidimensionale care să poată procesa direct intrări tridimensionale. A fost prezentată definiţia acestui tip de reţele, precum şi algoritmul backpropagation de învăţare pentru reţelele de tip feedforward.

- Secţiunea 5.2 a introdus **reţelele neuronale cu valori vectori $n$-dimensionali**, diferite de cele cu valori vectori tridimensionali prin aceea că în aceste reţele, ponderile sunt matrici, pe când în cele definite în secţiunea anterioară, ponderile sunt vectori tridimensionali. Acest tip de reţele sunt o contribuţie originală a prezentei teze, ele fiind definite doar pentru cazul $n = 3$ în literatură, iar pentru cazul general, fiind definit doar neuronul, fără ca acesta să fie inclus într-o reţea neuronală. În această secţiune, am definit reţelele de tip feedforward şi am dedus algoritmul backpropagation pentru aceste reţele.

- Secţiunea 5.3 a definit, în premieră după cunoştinţele noastre, **reţelele neuronale cu valori matrici pătratice**. Acest tip de reţele constituie o generalizare directă a reţelelor neuronale Clifford, deoarece orice algebră Clifford poate fi scrisă ca o subalgebră a algebrei matricilor pătratice. Un avantaj deosebit al acestor reţele este acela că dimensiunea lor este $n^2$, spre deosebire de dimensiunea reţelelor Clifford, care este $2^n$. În acestă secţiune, am prezentat reţelele neuronale cu valori matrici pătratice de tip feedforward, şi am dedus algoritmul de învăţare backpropagation pentru acest tip de reţele.

- În Secţiunea 5.4 am introdus, tot în premieră, **reţelele neuronale cu valori matrici antisimetrice**, sau mai general reţelele neuronale cu valori în algebre Lie. Datorită aplicaţiilor pe care le au algebrele Lie în domeniul computer vision, am considerat că o generalizare interesantă a reţelelor cu valori reale ar fi cea a reţelelor cu valori în algebre Lie, în particular cu valori matrici antisimetrice. Acestea constituie, de asemenea, o generalizare de dimensiune $\frac{n(n-1)}{2}$ a reţelelor cu valori vectori tridimensionali (de dimensiune $3$), care au fost prezentate în prima secţiune a acestui capitol.

- În fine, Secţiunea 5.5 a prezentat o problemă legată de reţelele neuronale cu valori matrici, şi anume **problema medierii matricilor ortogonale**. După o scurtă introducere, am prezentat un nou algoritm pentru **medierea pe o varietate riemannianã**, care, spre deosebire de algoritmii existenţi în literatură, permite calcule în coordonate carteziene (pe $\mathbb{R}^n$), care pot fi uşor automatizate cu ajutorul calculatorului. Deoarece o problemă de interes în aplicaţiile inginereşti o constituie medierea pe grupul matricilor ortogonale, am prezentat particularizarea algoritmului general pentru medierea pe $SO(3)$ prin scufundarea în $\mathbb{R}^4$, pentru patru funcţii de cost prezente în literatură. În finalul acestei secţiuni, am realizat, de asemenea, particularizarea algoritmului general pe $SO(n)$ (grupul matricilor ortogonale de ordinul $n$), prin scufundare în $\mathbb{R}^{n^2}$, pentru aceleaşi patru funcţii de cost.

# Capitolul 6

# Rezultate experimentale

Prezentul capitol este dedicat experimentelor cu algoritmii introduşi de-a lungul tezei. Implementarea acestor algoritmi a fost făcută folosind MATLAB, care reprezintă standardul în ceea ce priveşte cercetarea din inteligenţa artificială, şi mai ales din învăţarea automată.

Secţiunea 6.1 este dedicată reţelelor neuronale cu valori complexe. Prima subsecţiune descrie caracteristicile de implementare şi execuţie a experimentelor cu acest tip de reţele. Apoi, sunt prezentate $12$ seturi de experimente, fiecare set făcându-se pe câte un benchmark foarte cunoscut din domeniu, atât sintetic, cât şi din lumea reală. Astfel, aplicaţiile sintetice includ problema XOR şi problema XOR extinsă, din domeniul recunoaşterii de tipare şi două funcţii complexe complet şi trei funcţii complexe divizat, din domeniul aproximării de funcţii. Aplicaţiile din lumea reală includ egalizarea cu canal liniar şi neliniar, predicţia seriilor de timp liniare şi neliniare şi predicţia direcţiei şi vitezei vântului, din domeniul predicţiei de semnale.

Secţiunea 6.2 prezintă rezultatele experimentale a trei funcţii sintetice, din domeniul aproximării de funcţii, folosind reţele neuronale cu valori matrici pătratice. Singurul algoritm folosit este metoda gradient.

Următoarea secţiune, Secţiunea 6.3 este dedicată reţelelor neuronale cu valori matrici antisimetrice. Există 5 subsecţiuni, câte una pentru fiecare dintre cele două funcţii sintetice, şi cele trei transformări geometrice, toate făcând parte din domeniul aproximării de funcţii.

În Secţiunea 6.4 sunt tratate două funcţii de cost din cele patru pentru care a fost aplicat algoritmul de încorporare, pentru medierea unor matrici ortogonale de dimensiune $3$. Practic, fiecare dintre cele două funcţii de cost dă o altă medie, iar alegerea uneia sau a alteia depinde exclusiv de aplicaţia care necesită medierea rotaţiilor. În unele cazuri este posibil ca o medie să fie mai aproape de ceea ce se consideră intuitiv a fi media matricilor ortogonale, iar în alte cazuri este posibil să fie cealaltă. După cum contextul este cel care impune alegerea mediei aritmetice sau geometrice ca fiind cea mai potrivită în cazul numerelor reale, aşa şi în acest caz, doar contextul poate arăta care medie este cea mai bună. Algoritmul nostru însă are meritul de a da posibilitatea alegerii oricărei medii, fără niciun fel de restricţii asupra funcţiei de cost, ceea ce nu se întâmplă în cazul algoritmilor existenţi deja în literatură, unde calculele sunt practic imposibil de realizat pentru anumite expresii ale funcţiilor de cost.

În fine, Secţiunea 6.5 prezintă pe scurt concluziile capitolului dedicat experimentelor.

## 6.1 Reţele neuronale cu valori complexe

### 6.1.1 Implementarea

Implementarea algoritmilor prezentaţi în Capitolul 3 s-a realizat folosind MATLAB R2013b. Practic, însăşi ideea dezvoltării acestor algoritmi a venit din MATLAB, şi anume din Neural Network Toolbox, care, chiar şi în cea mai recentă versiune de MATLAB, şi anume R2015a, cuprinde majoritatea acestor algoritmi. Metodele de antrenare a reţelelor feedforward din Neural Network Toolbox sunt date în Tabelul 6.1. Abrevierea lor din tabel este exact numele funcţiei din MATLAB care trebuie dat la crearea unei reţele de tip feedforward folosind funcţia `feedforwardnet(hiddenSizes,trainFcn)`, al cărei prim parametru este un vector linie al dimensiunilor straturilor ascunse, şi al doilea parametru este funcţia de antrenare, care poate fi una dintre cele date în tabel.

| Abreviere | Denumire |
|---|---|
| TRAINGD | Metoda gradient |
| TRAINGDA | Metoda gradient adaptivă |
| TRAINGDM | Metoda gradient cu moment |
| TRAINGDX | Metoda gradient adaptivă cu moment |
| TRAINRP | Metoda resilient backpropagation |
| TRAINCGP | Metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere |
| TRAINCGF | Metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves |
| TRAINCGB | Metoda gradienţilor conjugaţi cu reporniri Powell-Beale |
| TRAINSCG | Metoda gradienţilor conjugaţi scalaţi |
| TRAINBFG | Metoda Broyden-Fletcher-Goldfarb-Shanno |
| TRAINOSS | Metoda one step secant |
| TRAINLM | Metoda Levenberg-Marquardt |
| TRAINBR | Metoda regularizării bayesiene |

Tabela 6.1: Algoritmii din Neural Network Toolbox din MATLAB

Pornind de la această schemă, am creat un cadru asemănător celui din Neural Network Toolbox, pentru reţele neuronale feedforward cu valori complexe. Astfel, am implementat câte o funcţie pentru fiecare dintre algoritmii descrişi în Capitolul 3, a căror denumire apare în prima coloană a Tabelului 6.2, a doua coloană prezentând numele pe larg al fiecărui algoritm. Fiecare astfel de funcţie are ca parametri aceeaşi parametri care pot fi setaţi şi în `trainParam` pentru `feedforwardnet`, în Neural Network Toolbox. Aceste funcţii se pot folosi în acelaşi fel cu cele standard din MATLAB, dând celor interesaţi posibilitatea experimentării cu aceşti algoritmi de antrenare, fără a fi nevoie de o perioadă de familiarizare prea îndelungată cu cadrul propus.

### 6.1.2 Problema XOR

*Problema XOR* este un benchmark clasic pentru reţelele neuronale, pentru că este cunoscut faptul că un neuron cu valori reale nu poate învăţa această problemă. Cu toate acestea, o reţea neuronală multistrat cu valori reale poate învăţa cu succes problema XOR. Pe de altă parte, s-a arătat în [194] că problema XOR complexă, care este doar o codificare în domeniul complex a problemei XOR reale, dată în Tabelul

| Abreviere | Denumire |
|---|---|
| GD | Metoda gradient |
| GDM | Metoda gradient cu moment |
| QCP | Metoda quickprop |
| RPR | Metoda resilient backpropagation |
| DBD | Metoda delta-bar-delta |
| SAB | Metoda SuperSAB |
| CGHS | Metoda gradienţilor conjugaţi cu actualizări Hestenes-Stiefel |
| CGPR | Metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere |
| CGFR | Metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves |
| CGDY | Metoda gradienţilor conjugaţi cu actualizări Dai-Yuan |
| CGHS+ | Metoda gradienţilor conjugaţi cu actualizări Hestenes-Stiefel pozitive |
| CGPR+ | Metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere pozitive |
| CGPB | Metoda gradienţilor conjugaţi cu reporniri Powell-Beale |
| SCG | Metoda gradienţilor conjugaţi scalaţi |
| SR1 | Metoda actualizării de grad unu |
| DFP | Metoda Davidon-Fletcher-Powell |
| BFG | Metoda Broyden-Fletcher-Goldfarb-Shanno |
| OSS | Metoda one step secant |

Tabela 6.2: Algoritmii implementaţi pentru reţele neuronale cu valori complexe

6.3, poate fi învăţată de un singur neuron cu valori complexe. Aceasta demonstrează puterea acestui tip de neuron şi constituie un bun argument pentru studierea reţelelor neuronale cu valori complexe.

| $z_I$ | $z_O$ |
|---|---|
| $-1-i$ | $1$ |
| $-1+i$ | $0$ |
| $1-i$ | $1+i$ |
| $1+i$ | $i$ |

Tabela 6.3: Problema XOR

Problema XOR complexă a fost folosită ca benchmark pentru mai multe tipuri de reţele neuronale cu valori complexe, spre exemplu de [143, 189, 196, 194, 70, 257, 12, 236, 237, 279, 278], şi ca o consecinţă am decis să începem experimentele noastre, cu algoritmii de învăţare propuşi, pe acest exemplu simplu.

În primul set de experimente, am antrenat o reţea neuronală cu valori complexe cu un strat ascuns constând din $2$ neuroni timp de $100$ de epoci. Funcţia de activare a stratului ascuns a fost funcţia complexă divizat tangentă hiperbolică, dată prin

$$G^2(x+iy) = \tanh x + i \tanh y = \frac{e^x - e^{-x}}{e^x + e^{-x}} + i\frac{e^y - e^{-y}}{e^y + e^{-y}},$$

şi a stratului de ieşire funcţia identitate $G^3(z) = z$. Ponderile au fost iniţializate cu valori aflate în interiorul cercului cu centrul în origine de rază $0.5$.

Antrenarea a fost făcută utilizând metoda gradient (abreviată GD), metoda gradient cu moment (GDM), algoritmul quickprop (QCP), algoritmul resilient backpropagation (RPR), algoritmul delta-bar-delta (DBD) şi algoritmul SuperSAB (SAB), din seria metodelor gradient îmbunătăţite. Rata de învăţare pentru metoda gradient a fost aleasă $0.1$, iar momentul pentru metoda gradient cu moment a fost $0.9$. Din metodele gradienţilor conjugaţi, am făcut experimente cu metoda gradienţilor conjugaţi cu actualizări Hestenes-Stiefel (CGHS), metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere (CGPR), metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves (CGFR), metoda gradienţilor conjugaţi cu actualizări Dai-Yuan (CGDY), variaţiile algoritmilor Hestenes-Stiefel (CGHS+) şi Polak-Ribiere (CGPR+) şi metoda gradienţilor conjugaţi cu reporniri Powell-Beale (CGPB) şi cu metoda gradienţilor conjugaţi scalaţi (SCG). Din clasa metodelor quasi-Newton, am folosit pentru antrenare metoda actualizării de rang unu (SR1), metoda Davidon-Fletcher-Powell (DFP), metoda Broyden-Fletcher-Goldfarb-Shanno (BFG) şi metoda one step secant (OSS).

Antrenarea a fost repetată de $50$ de ori, şi eroarea medie pătratică (EMP) este dată în Tabelul 6.4, specificând pentru fiecare algoritm media şi deviaţia standard a EMP de-a lungul celor $50$ de rulări. Tabelul a fost împărţit în funcţie de cele cinci clase de algoritmi implementaţi, cu evidenţierea algoritmului care a avut cea mai bună performanţă din fiecare clasă. În Figura 6.1 am prezentat, într-o formă sintetică, cantitatea $-10\log_{10}(EMP)$, adică logaritmul zecimal al erorii medii pătratice, înmulţit cu $-10$, astfel că cel mai bun algoritm are cea mai mare valoare pentru această cantitate.

| Algoritm | EMP Antrenare |
|---|---|
| **GD** | $\mathbf{2.58e-4 \pm 2.01e-4}$ |
| GDM | $3.28e-4 \pm 1.80e-4$ |
| QCP | $2.03e-5 \pm 2.34e-5$ |
| **RPR** | $\mathbf{1.04e-5 \pm 8.31e-6}$ |
| DBD | $7.98e-1 \pm 1.52e-1$ |
| SAB | $1.58e-4 \pm 1.06e-4$ |
| CGHS | $2.73e-8 \pm 3.22e-8$ |
| CGPR | $4.69e-7 \pm 6.16e-7$ |
| CGFR | $2.10e-8 \pm 2.83e-8$ |
| CGDY | $2.83e-8 \pm 3.00e-8$ |
| CGHS+ | $2.73e-8 \pm 3.01e-8$ |
| CGPR+ | $9.04e-7 \pm 1.20e-6$ |
| **CGPB** | $\mathbf{9.04e-10 \pm 1.35e-9}$ |
| **SCG** | $\mathbf{1.39e-10 \pm 2.56e-10}$ |
| SR1 | $1.06e-8 \pm 1.90e-8$ |
| DFP | $3.56e-9 \pm 5.63e-9$ |
| **BFG** | $\mathbf{8.29e-11 \pm 1.70e-10}$ |
| OSS | $3.54e-8 \pm 3.85e-8$ |

Tabela 6.4: Rezultate experimentale pentru problema XOR

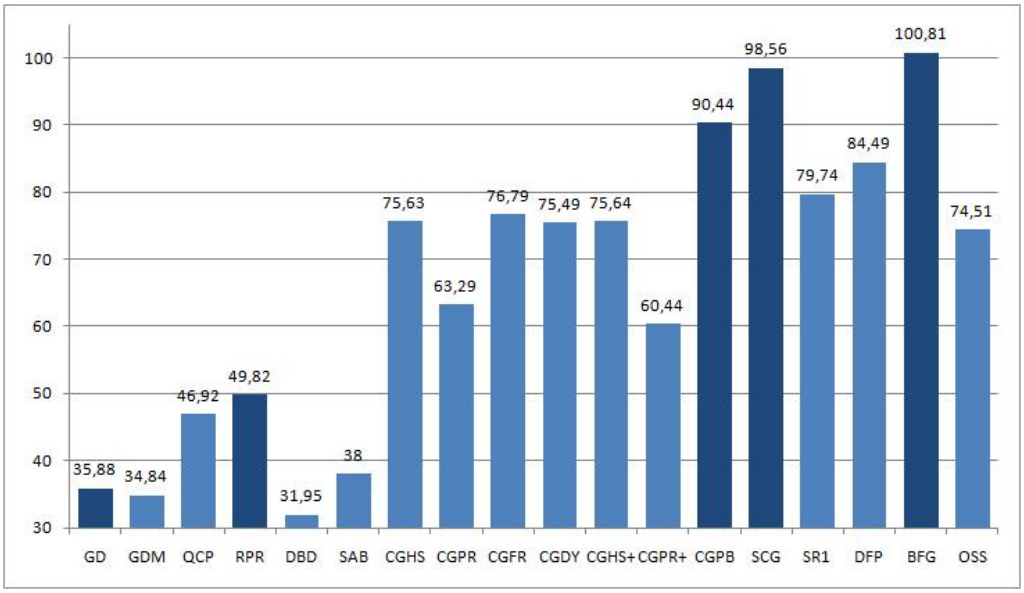


Figura 6.1: Rezultate experimentale pentru problema XOR

### 6.1.3 Problema XOR extinsă

Pentru al doilea set de experimente, am decis să considerăm o versiune augmentată a problemei XOR complexe, pe care am numit-o *problema XOR extinsă*. A fost folosită ca benchmark pentru testarea reţelelor neuronale cu valori complexe, de exemplu în [87, 88, 61, 57]. Este puţin mai complicată decât problema XOR complexă considerată mai sus, în aceea că are două intrări şi $16$ *tipare de antrenare*, după cum se poate vedea în Tabelul 6.5.

La fel ca în experimentul de mai sus, am repetat antrenarea folosind aceiaşi $18$ algoritmi din cele cinci clase, descrişi în subsecţiunea anterioară, de $50$ de ori, şi media şi deviaţia standard a erorii medii pătratice (EMP) este dată în Tabelul 6.6. Toate reţelele au avut $5$ neuroni ascunşi pe un singur strat, aceleaşi funcţii de activare ca mai sus, şi au fost antrenate timp de $500$ de epoci. Rata de învăţare pentru metoda gradient a fost tot $0.1$, iar momentul pentru metoda gradient cu moment a fost $0.9$. Iniţializarea ponderilor s-a făcut cu valori aleatoare din interiorul cercului cu centrul în origine de rază $0.5$. În fiecare clasă marcată în tabel, s-a evidenţiat algoritmul cu cea mai bună performanţă. Figura 6.2 prezintă sintetic cantitatea $-10\log_{10}(EMP)$, astfel încât cei mai buni algoritmi au cea mai mare valoare pentru această cantitate.

### 6.1.4 Funcţia complexă complet I

Prima funcţie complexă complet sintetică pe care vom testa algoritmii propuşi este funcţia pătratică de două variabile

$$f_1(z_1, z_2) = \frac{1}{6}\left(z_1^2 + z_2^2\right).$$

Cea mai importantă proprietate a funcţiilor complexe complet este aceea că tratează numărul complex ca un întreg, şi nu părţile reală şi imaginară separat. Această funcţie

| $z_{I1}$ | $z_{I2}$ | $z_O$ |
|---|---|---|
| $0$ | $0$ | $1$ |
| $0$ | $-i$ | $i$ |
| $-i$ | $0$ | $0$ |
| $-i$ | $-i$ | $1+i$ |
| $-i$ | $1$ | $i$ |
| $1$ | $1$ | $1+i$ |
| $1-i$ | $-i$ | $i$ |
| $1-i$ | $1-i$ | $1$ |
| $0$ | $1$ | $i$ |
| $0$ | $1-i$ | $0$ |
| $-i$ | $1-i$ | $0$ |
| $1$ | $0$ | $0$ |
| $1$ | $-i$ | $i$ |
| $1$ | $1-i$ | $0$ |
| $1-i$ | $0$ | $0$ |
| $1-i$ | $1$ | $i$ |

Tabela 6.5: Problema XOR extinsă

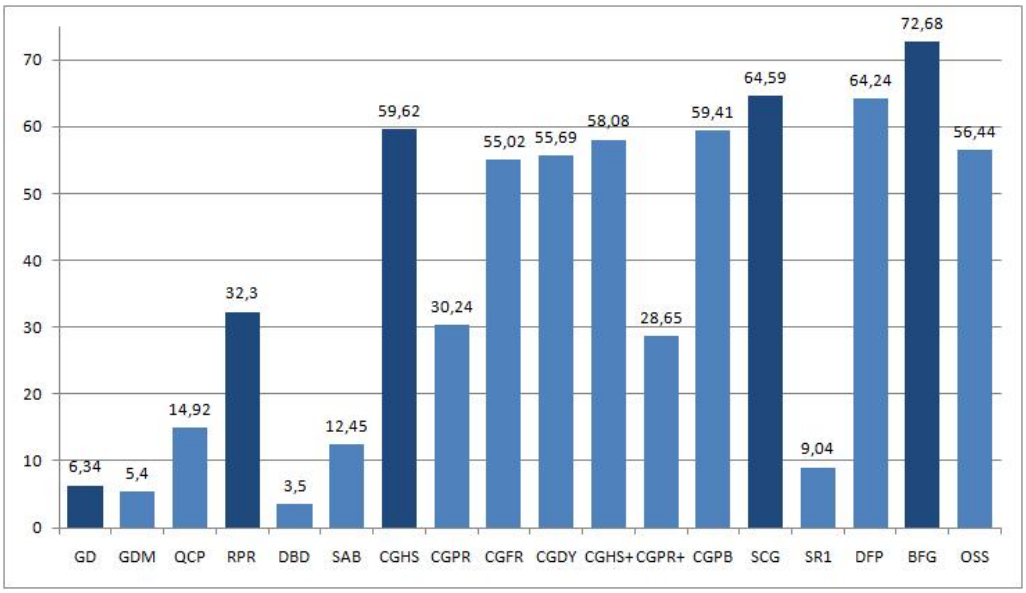


Figura 6.2: Rezultate experimentale pentru problema XOR extinsă

a fost folosită pentru a testa eficienţa diferitelor arhitecturi de reţele neuronale cu valori complexe şi a diferiţilor algoritmi de antrenare a acestora, de exemplu în [237, 231, 230, 232, 244].

Pentru a antrena reţelele, am generat $3000$ de tipare de antrenare, astfel încât fiecare tipar are intrările $z_1, z_2$ în interiorul discului centrat în origine cu raza $2.5$. Pentru testare, am generat $1000$ de tipare de testare având aceleaşi caracteristici. Toate reţelele au avut un singur strat ascuns format din $15$ neuroni şi au fost antrenate $5000$ de epoci. Funcţia de activare a stratului ascuns a fost funcţia complexă complet

| Algoritm | EMP Antrenare |
|---|---|
| **GD** | $\mathbf{2.33e-1 \pm 9.05e-2}$ |
| GDM | $2.88e-1 \pm 8.17e-2$ |
| QCP | $3.22e-2 \pm 1.69e-2$ |
| **RPR** | $\mathbf{5.89e-4 \pm 4.47e-4}$ |
| DBD | $4.47e-1 \pm 5.09e-3$ |
| SAB | $5.69e-2 \pm 3.50e-2$ |
| **CGHS** | $\mathbf{1.09e-6 \pm 8.02e-7}$ |
| CGPR | $9.47e-4 \pm 1.01e-3$ |
| CGFR | $3.15e-6 \pm 2.47e-6$ |
| CGDY | $2.70e-6 \pm 1.91e-6$ |
| CGHS+ | $1.55e-6 \pm 1.24e-6$ |
| CGPR+ | $1.36e-3 \pm 1.44e-3$ |
| CGPB | $1.14e-6 \pm 8.74e-7$ |
| **SCG** | $\mathbf{3.47e-7 \pm 3.27e-7}$ |
| SR1 | $1.25e-1 \pm 9.70e-2$ |
| DFP | $3.77e-7 \pm 4.79e-7$ |
| **BFG** | $\mathbf{5.39e-8 \pm 7.69e-8}$ |
| OSS | $2.27e-6 \pm 1.45e-6$ |

Tabela 6.6: Rezultate experimentale pentru problema XOR extinsă

tangentă hiperbolică, dată prin

$$G^2(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

şi a stratului de ieşire a fost tot funcţia identitate $G^3(z) = z$.

Media şi deviaţia standard a EMP luate pentru $50$ de rulări ale fiecărui algoritm sunt date în Tabelul 6.7, iar dinamica învăţării apare în Figura 6.4, separat pentru fiecare dintre cele trei clase de algoritmi studiaţi. Pentru a observa felul în care EMP evoluează de-a lungul procesului de învăţare a reţelelor, am reprezentat $10\log_{10}(EMP)$ pe axa verticală a figurii, cu cea mai mică valoare fiind cea mai bună. Ca o consecinţă, valorile pe acea axă sunt negative. Axa orizontală reprezintă epocile de antrenare. În tabel sunt date de asemenea valorile EMP pentru alţi algoritmi folosiţi pentru a învăţa această funcţie, împreună cu referinţele în care aceşti algoritmi şi reţele au apărut pentru prima dată. Ca mai sus, Figura 6.3 prezintă valoarea lui $-10\log_{10}(EMP)$ pentru algoritmii propuşi.

### 6.1.5 Funcţia complexă complet II

Un exemplu mai complicat, care presupune patru variabile de intrare şi inversul uneia dintre variabile, este dat de următoarea funcţie:

$$f_2(z_1, z_2, z_3, z_4) = \frac{1}{10}\left(z_3 + 10 z_1 z_4 + \frac{z_2^2}{z_1}\right),$$

care a fost folosită ca benchmark în [237, 232, 13, 233, 234]. Am generat $3000$ de tipare aleatoare de antrenare şi $1000$ de tipare de testare, toate având intrările în interiorul discului centrat în origine şi cu raza $1$. Variabila $z_1$ a fost aleasă astfel încât raza ei să fie mai mare decât $0.1$, deoarece inversa ei apare în expresia funcţiei, şi altfel ar fi putut rezulta valori foarte mari ale funcţiei în comparaţie cu celelalte

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{5.23e-5 \pm 9.29e-6}$ | $\mathbf{5.84e-5 \pm 1.16e-5}$ |
| GDM | $5.05e-5 \pm 9.23e-6$ | $5.65e-5 \pm 1.10e-5$ |
| QCP | $8.28e-6 \pm 2.06e-6$ | $9.51e-6 \pm 2.65e-6$ |
| **RPR** | $\mathbf{2.09e-6 \pm 4.02e-7}$ | $\mathbf{2.45e-6 \pm 5.22e-7}$ |
| DBD | $5.50e-6 \pm 1.26e-6$ | $6.28e-6 \pm 1.60e-6$ |
| SAB | $1.55e-5 \pm 5.65e-6$ | $1.74e-5 \pm 6.58e-6$ |
| **CGHS** | $\mathbf{1.21e-7 \pm 2.96e-8}$ | $\mathbf{1.38e-7 \pm 3.70e-8}$ |
| CGPR | $1.61e-5 \pm 5.14e-6$ | $1.80e-5 \pm 6.19e-6$ |
| CGFR | $1.51e-7 \pm 3.95e-8$ | $1.78e-7 \pm 4.98e-8$ |
| CGDY | $1.45e-7 \pm 4.02e-8$ | $1.69e-7 \pm 4.66e-8$ |
| CGHS**+** | $1.38e-7 \pm 2.01e-8$ | $1.63e-7 \pm 2.74e-8$ |
| CGPR+ | $1.73e-5 \pm 4.18e-6$ | $1.89e-5 \pm 4.75e-6$ |
| CGPB | $1.78e-7 \pm 4.98e-8$ | $2.21e-7 \pm 5.78e-8$ |
| **SCG** | $\mathbf{7.22e-9 \pm 2.75e-9}$ | $\mathbf{8.77e-9 \pm 3.34e-9}$ |
| SR1 | $1.51e-5 \pm 4.58e-6$ | $1.53e-5 \pm 6.03e-6$ |
| DFP | $2.56e-9 \pm 1.15e-9$ | $2.94e-9 \pm 1.38e-9$ |
| **BFG** | $\mathbf{7.78e-12 \pm 4.29e-12}$ | $\mathbf{9.43e-12 \pm 5.33e-12}$ |
| OSS | $2.70e-7 \pm 7.36e-8$ | $3.23e-7 \pm 9.82e-8$ |
| CRBF [231, 230] | $3.50e-1$ | $3.88e-1$ |
| C-ELM (RBF) [231, 230] | $4.74e-1$ | $4.95e-1$ |
| FC-RBF [231, 230] | $3.61e-6$ | $9.00e-6$ |
| **FC-RBF with KMC** [231] | $\mathbf{2.01e-6}$ | $\mathbf{1.87e-6}$ |
| Mc-FCRBF [232] | $2.50e-5$ | $2.56e-6$ |
| CSRAN [244] | $9.00e-6$ | $9.00e-6$ |
| CMRAN [231, 230] | $4.60e-3$ | $4.90e-3$ |

Tabela 6.7: Rezultate experimentale pentru funcţia complexă complet I

variabile. Singurul strat ascuns a avut $25$ de neuroni, aceleaşi funcţii de activare ca în experimentele din subsecţiunea anterioară, iar reţelele au fost antrenate pentru $5000$ de epoci. Tabelul 6.8 arată rezultatele antrenării, utilizând algoritmii descrişi, pe parcursul a $50$ de rulări ale fiecărui algoritm, iar Figura 6.5 ilustrează, pe o scară logaritmică identică cu cele din experimentele anterioare, aceleaşi rezultate.

### 6.1.6 Funcţia complexă divizat I

Testăm acum algoritmii propuşi pe funcţii complexe divizat, i.e. funcţii care tratează separat părţile reală şi imaginară ale argumentului complex. Prima astfel de funcţie, care a fost folosită în [21, 128, 54] pentru a testa performanţa reţelelor neuronale cu valori complexe, este

$$f_3(x+iy) = e^{iy}(1-x^2-y^2).$$

Setul de antrenare a fost compus din $3000$ de tipare de antrenare, iar setul de testare din $1000$ de tipare de testare generate aleator din interiorul discului unitate. Reţelele neuronale au avut $15$ neuroni pe un singur strat ascuns. Ca în experimentele din Subsecţiunea 6.1.2, funcţiile de activare au fost tangenta hiperbolică complexă divizat şi respectiv funcţia identitate. Ponderile au fost iniţializate aleator cu valori din interiorul discului unitate.

Rezultatele mediate de-a lungul a $50$ de rulări ale fiecărui algoritm antrenat timp de $5000$ de epoci sunt prezentate în Tabelul 6.9 şi, pe o scară logaritmică, în Figura

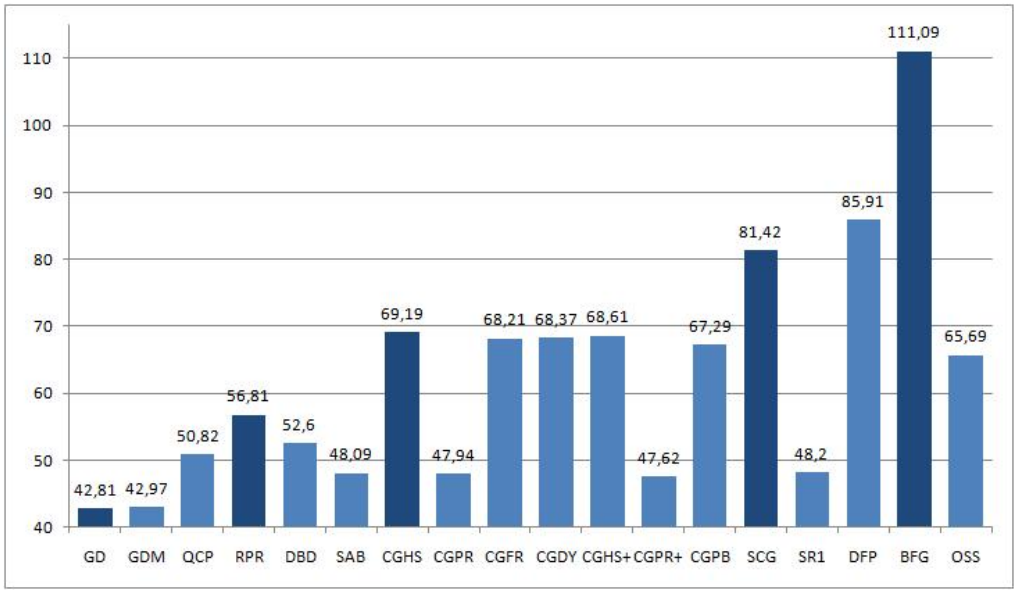


Figura 6.3: Rezultate experimentale pentru funcţia complexă complet I

6.6. Dinamica învăţării este înfăţişată în Figura 6.7.

### 6.1.7 Funcţia complexă divizat II

A doua funcţie complexă divizat, care a fost folosită în unul dintre primele articole din domeniul reţelelor neuronale cu valori complexe, şi anume [20], dar de asemenea în [21, 128, 54], este

$$f_4(x+iy) = x^2 + y^2 + 2ixy.$$

Seturile de antrenare şi de testare, precum şi arhitectura reţelelor sunt la fel ca în setul precedent de experimente. Tabelul 6.10 prezintă rezultatele de antrenare şi testare cu valoarea EMP mediată de-a lungul a $50$ de rulări ale fiecărui algoritm, iar Figura 6.8 ilustrează aceleaşi rezultate, folosind o scară logaritmică.

### 6.1.8 Funcţia complexă divizat III

Ultima funcţie de test din categoria complexă divizat, de asemenea folosită în [21, 128, 54], care conţine funcţii trigonometrice şi hiperbolice, este

$$f_5(x+iy) = \sin x \cosh y + i \cos x \sinh y.$$

Acest set de experimente a avut aceleaşi arhitecturi de reţele şi seturile de antrenare şi testare generate în acelaşi fel cu anterioarele două. Media şi deviaţia standard a erorii medii pătratice calculate pentru $50$ de rulări ale fiecăruia dintre cei $18$ algoritmi sunt prezentate în Tabelul 6.11. Figura 6.9 prezintă grafic valoarea lui $-10\log_{10}(EMP)$ pentru fiecare dintre cei $18$ algoritmi implementaţi.

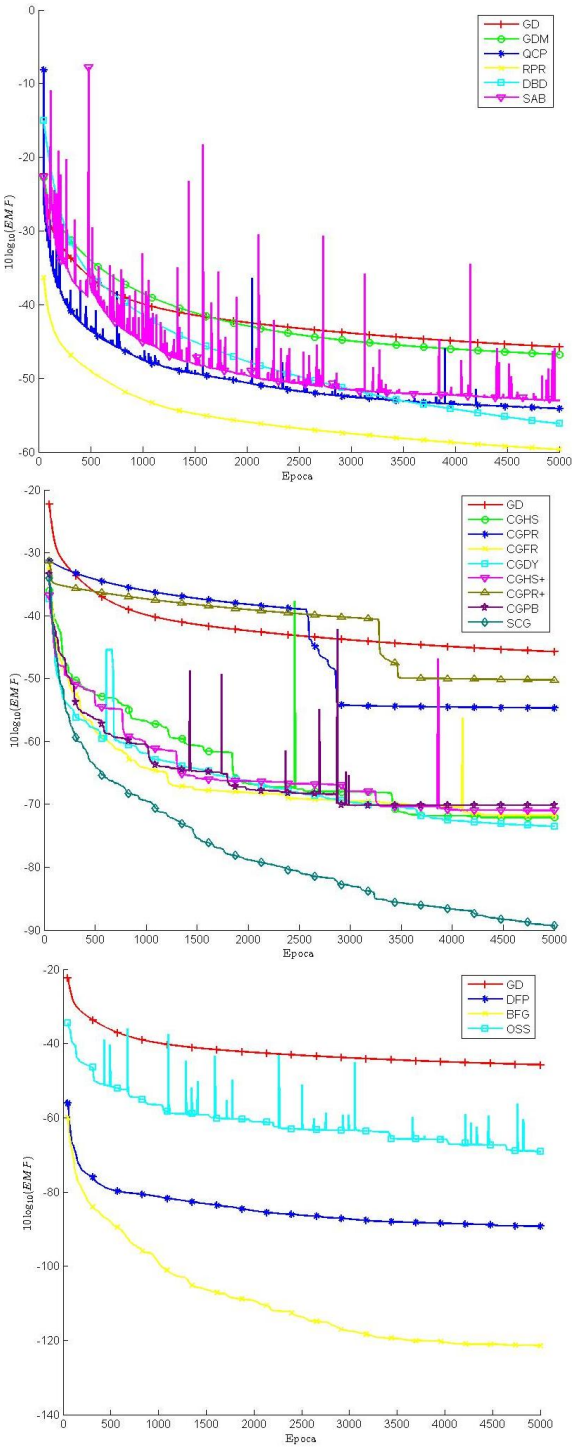


Figura 6.4: Dinamica învăţării pentru funcţia complexă complet I

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{5.32e-4 \pm 4.35e-5}$ | $\mathbf{6.26e-4 \pm 1.40e-4}$ |
| GDM | $5.42e-4 \pm 5.05e-5$ | $6.69e-4 \pm 2.17e-4$ |
| QCP | $1.46e-4 \pm 4.70e-6$ | $1.76e-4 \pm 9.61e-6$ |
| **RPR** | $\mathbf{1.42e-4 \pm 2.93e-6}$ | $\mathbf{1.67e-4 \pm 4.85e-6}$ |
| DBD | $1.54e-4 \pm 2.14e-6$ | $1.71e-4 \pm 3.60e-6$ |
| SAB | $1.67e-4 \pm 7.81e-6$ | $1.86e-4 \pm 1.37e-5$ |
| CGHS | $1.49e-4 \pm 2.35e-6$ | $1.66e-4 \pm 5.24e-6$ |
| CGPR | $1.70e-4 \pm 5.16e-6$ | $1.87e-4 \pm 8.86e-6$ |
| CGFR | $1.48e-4 \pm 1.71e-6$ | $1.64e-4 \pm 3.83e-6$ |
| **CGDY** | $\mathbf{1.41e-4 \pm 2.69e-6}$ | $\mathbf{1.61e-4 \pm 2.56e-6}$ |
| CGHS+ | $1.49e-4 \pm 2.56e-6$ | $1.64e-4 \pm 3.19e-6$ |
| CGPR+ | $1.72e-4 \pm 4.44e-6$ | $1.91e-4 \pm 1.14e-5$ |
| CGPB | $1.49e-4 \pm 2.82e-6$ | $1.65e-4 \pm 4.01e-6$ |
| **SCG** | $\mathbf{1.42e-4 \pm 3.01e-6}$ | $\mathbf{1.61e-4 \pm 3.42e-6}$ |
| SR1 | $1.45e-5 \pm 1.16e-6$ | $1.56e-5 \pm 1.15e-6$ |
| DFP | $6.03e-6 \pm 1.40e-6$ | $6.59e-6 \pm 1.52e-6$ |
| **BFG** | $\mathbf{5.12e-6 \pm 1.33e-6}$ | $\mathbf{5.63e-6 \pm 1.48e-6}$ |
| OSS | $1.37e-4 \pm 3.52e-6$ | $1.61e-4 \pm 3.42e-6$ |
| CRBF [231, 230] | $2.15e-2$ | $3.30e-2$ |
| FCRN [234] | $9.00e-4$ | $3.60e-3$ |
| C-ELM (RBF) [231, 230] | $3.67e-2$ | $5.28e-2$ |
| FC-RBF [231, 230] | $3.84e-4$ | $2.28e-3$ |
| FC-RBF with KMC [231] | $1.29e-4$ | $8.26e-3$ |
| **Mc-FCRBF [232]** | $\mathbf{8.10e-7}$ | $\mathbf{8.10e-7}$ |
| CSRAN [244] | $6.40e-5$ | $4.00e-4$ |
| CMRAN [231, 230] | $6.60e-4$ | $2.50e-1$ |

Tabela 6.8: Rezultate experimentale pentru funcţia complexă complet II

### 6.1.9 Egalizarea cu canal liniar

Egalizarea modulării amplitudinii în cuadratură (QAM) a semnalelor transmise prin canale liniare sau neliniare este o aplicaţie importantă a reţelelor neuronale cu valori complexe. Datorită acestui fapt, am decis să aplicăm algoritmii propuşi pe nişte benchmark-uri cunoscute în literatura din domeniul egalizării canalelor de comunicaţie.

Primul asemenea benchmark, propus în [69], şi folosit, de exemplu, în [139, 165, 281] este o problemă care vizează egalizarea unui canal liniar 4-QAM. Funcţia de activare a canalului este dată de

$$\begin{aligned} H(z) &= (0.7409 - 0.7406i)(1 - (0.2 - 0.1i)z^{-1}) \\ &\quad \times (1 - (0.6 - i0.3)z^{-1}). \end{aligned}$$

Constelaţia pentru egalizatorul 4-QAM este dată de simbolurile din mulţimea $\{\pm 1 \pm i\}$. Pentru antrenarea reţelelor, $10000$ de simboluri uniform generate din această mulţime au fost trecute prin canalul liniar dat mai sus şi zgomot alb gaussian a fost adăugat, care a avut un *raport semnal zgomot* (SNR, signal-to-noise ratio) de $15$. Mulţimea de testare a avut $5000$ de tipare generate la un SNR de $10$. Ordinul canalului este $3$ şi întârzierea a fost aleasă ca fiind $1$. În consecinţă, reţelele au avut $3$ intrări, $15$ neuroni ascunşi pe un singur strat ascuns şi un neuron pe stratul de ieşire, şi funcţia

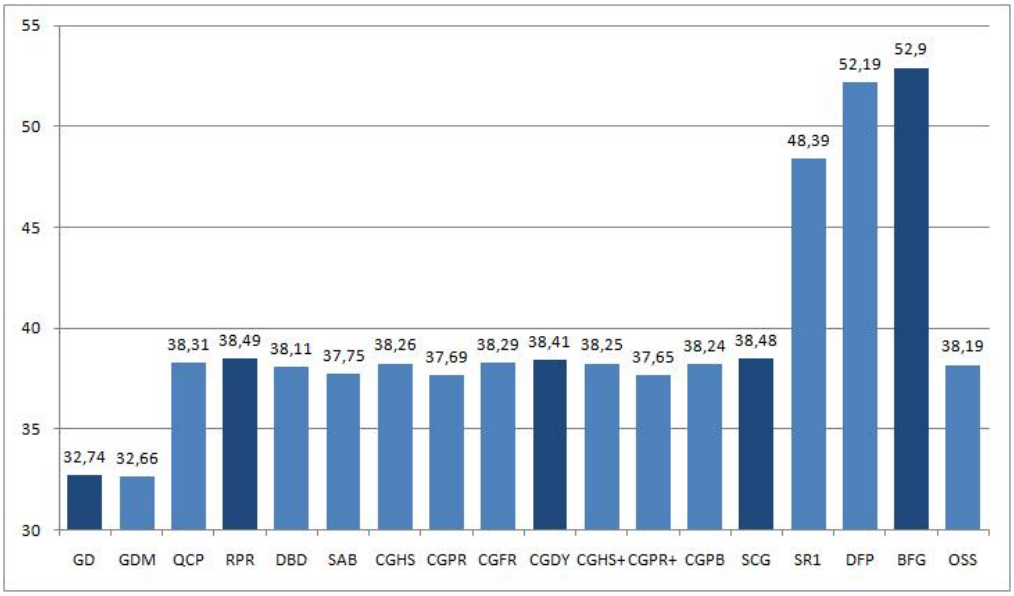


Figura 6.5: Rezultate experimentale pentru funcţia complexă complet II

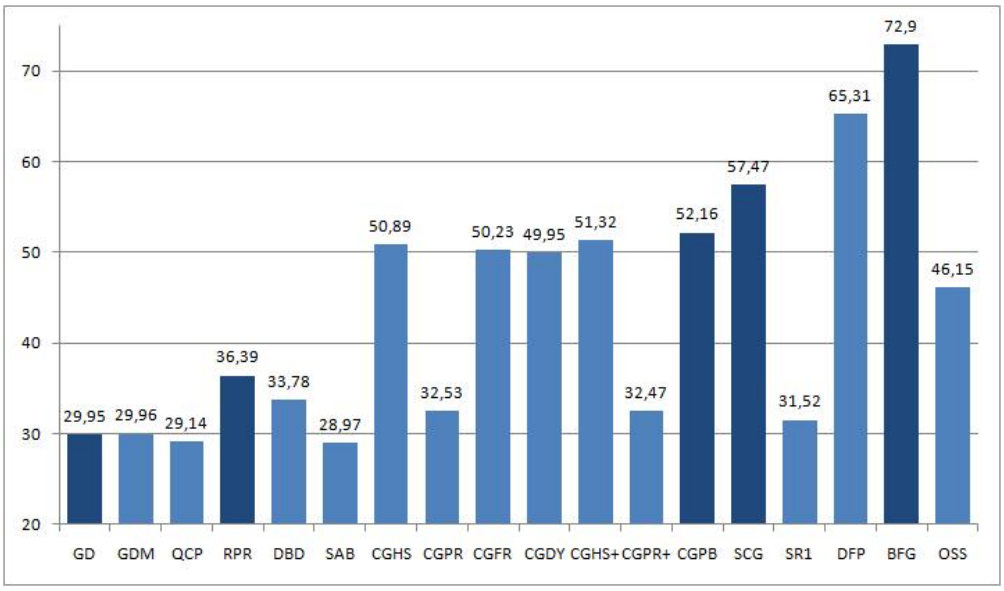


Figura 6.6: Rezultate experimentale pentru funcţia complexă divizat I

de activare tangentă hiperbolică complexă divizat pe stratul ascuns şi identitatea pe stratul de ieşire.

Media şi deviaţia standard a erorii medii pătratice (EMP) la antrenare şi testare calculate pentru $50$ de rulări ale fiecărui algoritm dintre cei $18$ aparţinând celor cinci clase

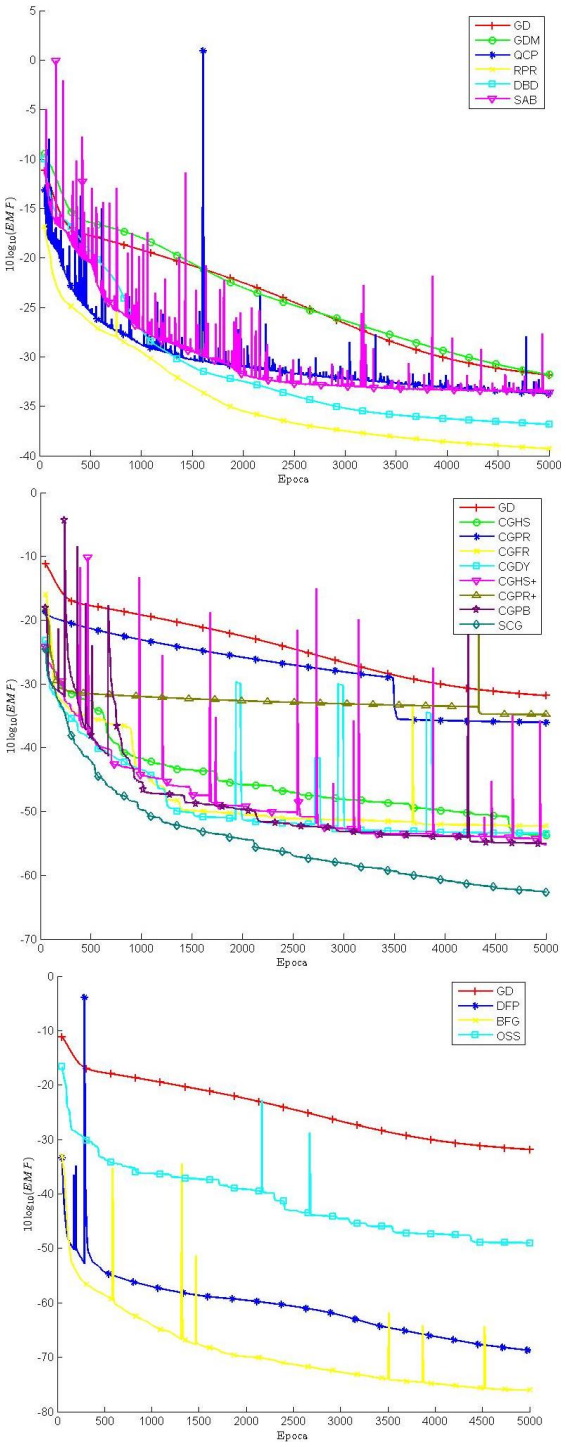


Figura 6.7: Dinamica învăţării pentru funcţia complexă divizat I

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{1.01e-3 \pm 1.55e-4}$ | $\mathbf{1.12e-3 \pm 1.60e-4}$ |
| GDM | $1.01e-3 \pm 1.63e-4$ | $1.11e-3 \pm 1.67e-4$ |
| QCP | $1.22e-3 \pm 5.67e-4$ | $1.35e-3 \pm 6.17e-4$ |
| **RPR** | $\mathbf{2.29e-4 \pm 5.08e-5}$ | $\mathbf{2.62e-4 \pm 5.85e-5}$ |
| DBD | $4.19e-4 \pm 1.01e-4$ | $4.70e-4 \pm 1.09e-4$ |
| SAB | $1.26e-3 \pm 6.36e-4$ | $1.39e-3 \pm 6.89e-4$ |
| CGHS | $8.15e-6 \pm 1.51e-6$ | $8.76e-6 \pm 1.52e-6$ |
| CGPR | $5.58e-4 \pm 9.62e-5$ | $6.29e-4 \pm 1.09e-4$ |
| CGFR | $9.49e-6 \pm 1.75e-6$ | $1.04e-5 \pm 1.81e-6$ |
| CGDY | $1.01e-5 \pm 2.26e-6$ | $1.09e-5 \pm 2.38e-6$ |
| CGHS+ | $7.38e-6 \pm 1.93e-6$ | $8.01e-6 \pm 2.05e-6$ |
| CGPR+ | $5.67e-4 \pm 1.08e-4$ | $6.39e-4 \pm 1.18e-4$ |
| **CGPB** | $\mathbf{6.08e-6 \pm 1.59e-6}$ | $\mathbf{6.60e-6 \pm 1.70e-6}$ |
| **SCG** | $\mathbf{1.79e-6 \pm 3.64e-7}$ | $\mathbf{2.05e-6 \pm 4.21e-7}$ |
| SR1 | $7.05e-4 \pm 1.76e-4$ | $7.62e-4 \pm 1.99e-4$ |
| DFP | $2.95e-7 \pm 9.15e-8$ | $3.40e-7 \pm 1.05e-7$ |
| **BFG** | $\mathbf{5.13e-8 \pm 1.30e-8}$ | $\mathbf{6.03e-8 \pm 1.61e-8}$ |
| OSS | $2.43e-5 \pm 4.41e-6$ | $2.69e-5 \pm 4.87e-6$ |

Tabela 6.9: Rezultate experimentale pentru funcţia complexă divizat I

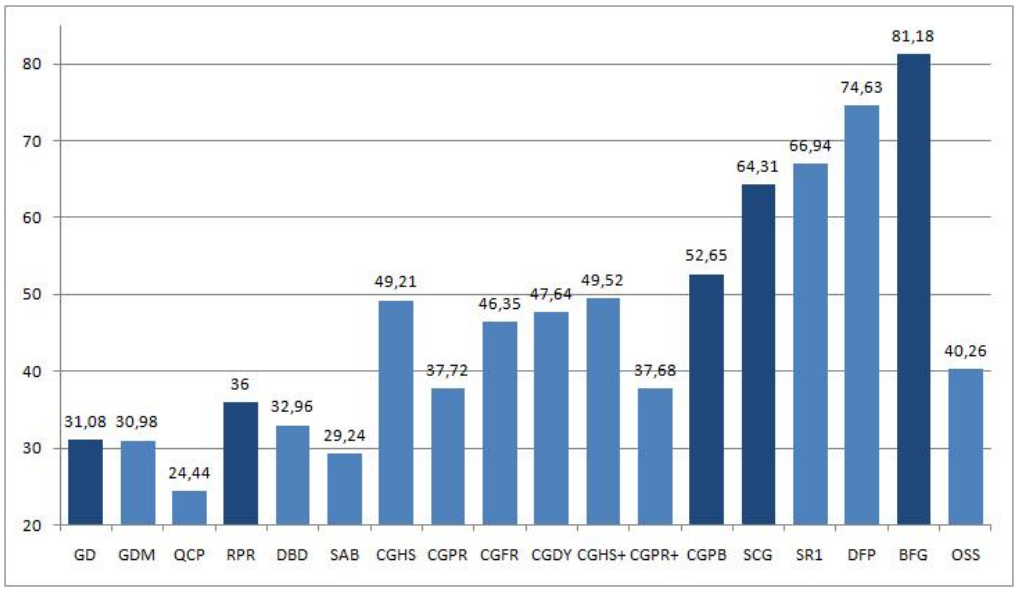


Figura 6.8: Rezultate experimentale pentru funcţia complexă divizat II

sunt date în Tabelul 6.12, iar dinamica învăţării este înfăţişată, separat pentru fiecare clasă de algoritmi, în Figura 6.11. Figura 6.10 prezintă rezultatele experimentale folosind o scară logaritmică.

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{7.80e-4 \pm 1.19e-4}$ | $\mathbf{8.63e-4 \pm 1.32e-4}$ |
| GDM | $7.98e-4 \pm 1.58e-4$ | $8.81e-4 \pm 1.81e-4$ |
| QCP | $3.57e-3 \pm 4.02e-3$ | $3.74e-3 \pm 4.08e-3$ |
| **RPR** | $\mathbf{2.51e-4 \pm 5.91e-5}$ | $\mathbf{2.92e-4 \pm 6.91e-5}$ |
| DBD | $5.06e-4 \pm 1.76e-4$ | $5.67e-4 \pm 2.00e-4$ |
| SAB | $1.18e-3 \pm 4.27e-4$ | $1.33e-3 \pm 4.57e-4$ |
| CGHS | $1.19e-5 \pm 4.72e-6$ | $1.37e-5 \pm 5.47e-6$ |
| CGPR | $1.69e-4 \pm 3.42e-5$ | $1.90e-4 \pm 3.92e-5$ |
| CGFR | $2.31e-5 \pm 8.98e-6$ | $2.62e-5 \pm 1.03e-5$ |
| CGDY | $1.72e-5 \pm 8.53e-6$ | $1.95e-5 \pm 9.67e-6$ |
| CGHS+ | $1.11e-5 \pm 4.81e-6$ | $1.28e-5 \pm 5.72e-6$ |
| CGPR+ | $1.71e-4 \pm 3.38e-5$ | $1.92e-4 \pm 3.81e-5$ |
| **CGPB** | $\mathbf{5.38e-6 \pm 2.83e-6}$ | $\mathbf{6.21e-6 \pm 3.33e-6}$ |
| **SCG** | $\mathbf{9.42e-5 \pm 2.62e-5}$ | $\mathbf{1.06e-4 \pm 2.98e-5}$ |
| SR1 | $2.02e-7 \pm 7.05e-8$ | $2.44e-7 \pm 8.64e-8$ |
| DFP | $3.44e-8 \pm 1.04e-8$ | $4.25e-8 \pm 1.28e-8$ |
| **BFG** | $\mathbf{7.62e-9 \pm 3.24e-9}$ | $\mathbf{9.86e-9 \pm 4.31e-9}$ |
| OSS | $9.42e-5 \pm 2.62e-5$ | $1.06e-4 \pm 2.98e-5$ |

Tabela 6.10: Rezultate experimentale pentru funcţia complexă divizat II

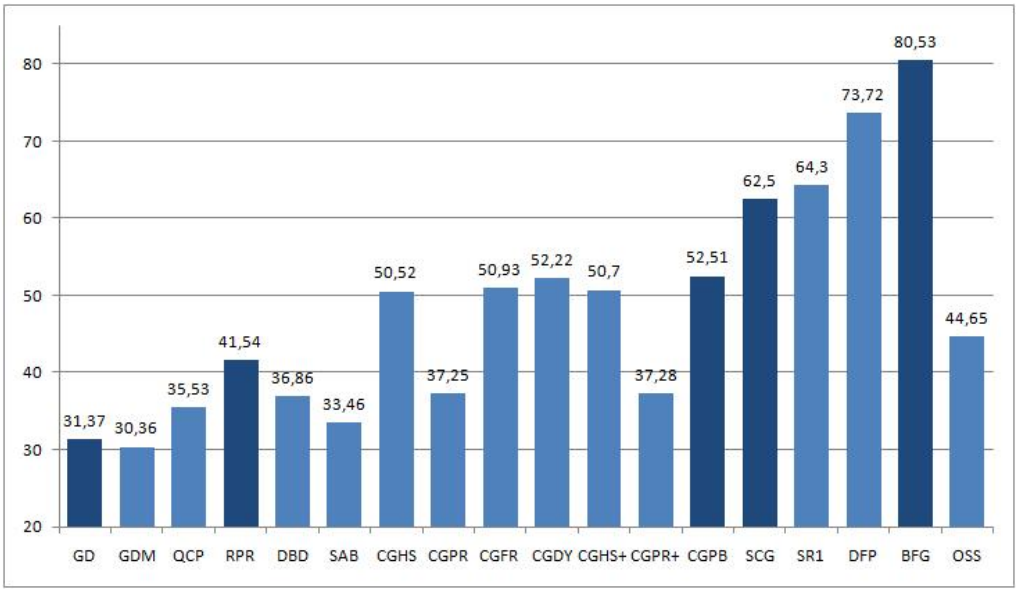


Figura 6.9: Rezultate experimentale pentru funcţia complexă divizat III

### 6.1.10 Egalizarea cu canal neliniar

Unul dintre cele mai cunoscute benchmark-uri ale reţelelor neuronale cu valori complexe este problema egalizării canalului neliniar 4-QAM propusă în [62], şi folosită ulterior în [237, 231, 230, 244, 13, 233, 273, 258, 235]. Modelul pentru canalul neliniar este

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{7.29e-4 \pm 1.71e-4}$ | $\mathbf{7.72e-4 \pm 1.78e-4}$ |
| GDM | $9.20e-4 \pm 2.15e-4$ | $9.67e-4 \pm 2.20e-4$ |
| QCP | $2.79e-4 \pm 6.69e-5$ | $3.07e-4 \pm 7.11e-5$ |
| **RPR** | $\mathbf{7.01e-5 \pm 1.81e-5}$ | $\mathbf{7.74e-5 \pm 2.04e-5}$ |
| DBD | $2.06e-4 \pm 4.41e-5$ | $2.26e-4 \pm 4.78e-5$ |
| SAB | $4.48e-4 \pm 2.14e-4$ | $5.01e-4 \pm 2.82e-4$ |
| CGHS | $8.83e-6 \pm 2.57e-6$ | $9.82e-6 \pm 2.83e-6$ |
| CGPR | $1.88e-4 \pm 4.14e-5$ | $2.04e-4 \pm 4.42e-5$ |
| CGFR | $8.02e-6 \pm 1.85e-6$ | $8.83e-6 \pm 2.15e-6$ |
| CGDY | $5.97e-6 \pm 1.28e-6$ | $6.52e-6 \pm 1.43e-6$ |
| CGHS+ | $8.48e-6 \pm 2.57e-6$ | $9.42e-6 \pm 2.92e-6$ |
| CGPR+ | $1.87e-4 \pm 4.63e-5$ | $2.03e-4 \pm 5.00e-5$ |
| **CGPB** | $\mathbf{5.57e-6 \pm 2.30e-6}$ | $\mathbf{6.13e-6 \pm 2.53e-6}$ |
| **SCG** | $\mathbf{5.62e-7 \pm 1.12e-7}$ | $\mathbf{3.73e-5 \pm 1.12e-5}$ |
| SR1 | $3.71e-7 \pm 1.14e-7$ | $4.13e-7 \pm 1.25e-7$ |
| DFP | $4.24e-8 \pm 1.28e-8$ | $4.67e-8 \pm 1.41e-8$ |
| **BFG** | $\mathbf{8.86e-9 \pm 2.26e-9}$ | $\mathbf{1.01e-8 \pm 2.65e-9}$ |
| OSS | $3.43e-5 \pm 1.04e-5$ | $3.73e-5 \pm 1.12e-5$ |

Tabela 6.11: Rezultate experimentale pentru funcţia complexă divizat III

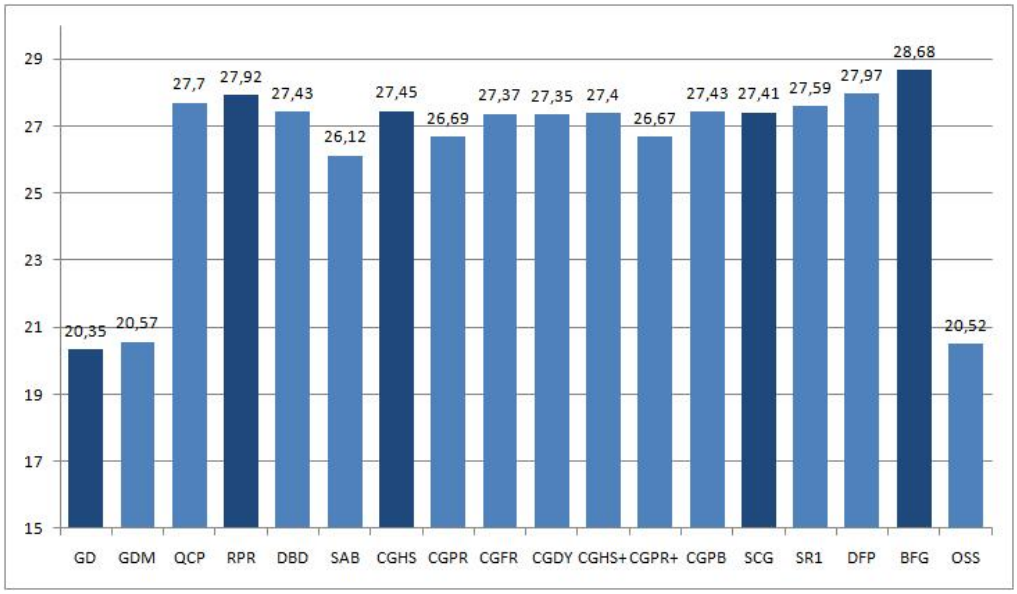


Figura 6.10: Rezultate experimentale pentru egalizarea cu canal liniar

$$
\begin{aligned}
o(k) &= (0.34\text{-}0.27i)s(k) + (0.87 + 0.43i)s(k-1) \\
&\quad +(0.34\text{-}0.21i)s(k-2) \\
z(k) &= o(k) + 0.1o(k)^2 + 0.05o(k)^3 + n(k),
\end{aligned}
$$

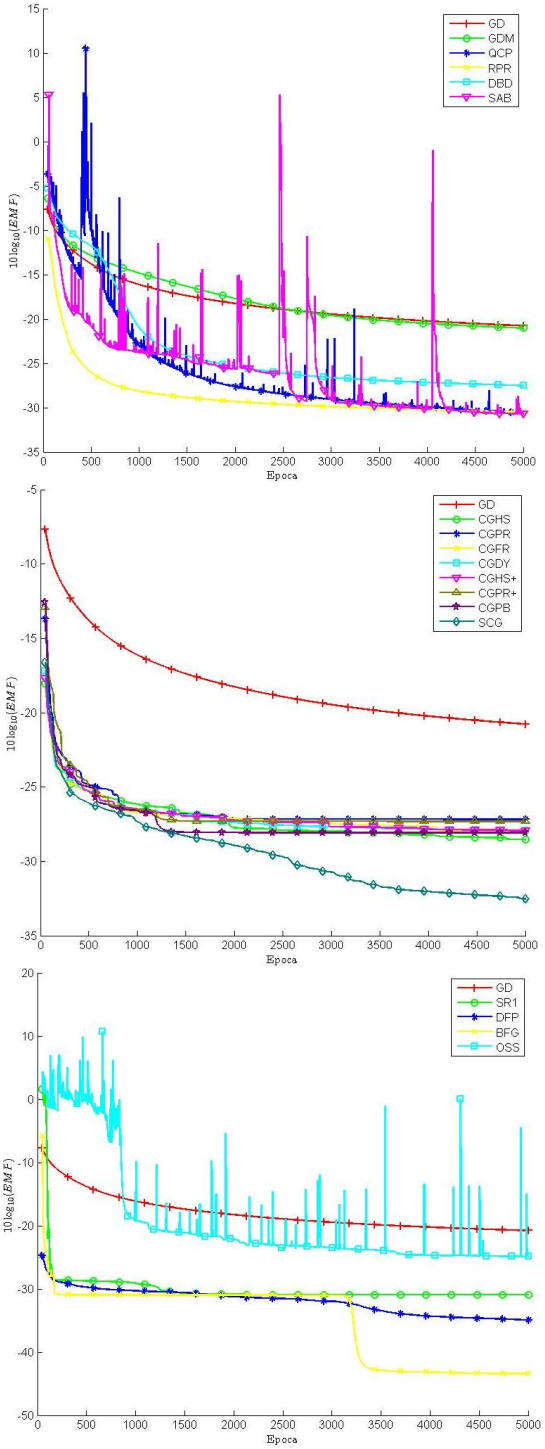


Figura 6.11: Dinamica învăţării pentru egalizarea cu canal liniar

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{9.22e-3 \pm 3.61e-4}$ | $\mathbf{8.43e-3 \pm 3.51e-4}$ |
| GDM | $8.76e-3 \pm 3.21e-4$ | $8.02e-3 \pm 3.00e-4$ |
| QCP | $1.70e-3 \pm 2.29e-4$ | $3.19e-3 \pm 4.12e-4$ |
| **RPR** | $\mathbf{1.61e-3 \pm 3.36e-5}$ | $\mathbf{2.82e-3 \pm 9.59e-5}$ |
| DBD | $1.81e-3 \pm 1.34e-5$ | $2.66e-3 \pm 1.59e-5$ |
| SAB | $2.44e-3 \pm 6.42e-4$ | $3.82e-3 \pm 9.99e-4$ |
| **CGHS** | $\mathbf{1.80e-3 \pm 5.24e-5}$ | $\mathbf{2.70e-3 \pm 4.49e-5}$ |
| CGPR | $2.14e-3 \pm 7.83e-5$ | $2.79e-3 \pm 3.00e-5$ |
| CGFR | $1.83e-3 \pm 4.00e-5$ | $2.71e-3 \pm 3.47e-5$ |
| CGDY | $1.84e-3 \pm 4.96e-5$ | $2.70e-3 \pm 4.75e-5$ |
| CGHS+ | $1.82e-3 \pm 4.57e-5$ | $2.71e-3 \pm 3.60e-5$ |
| CGPR+ | $2.14e-3 \pm 7.39e-5$ | $2.80e-3 \pm 3.34e-5$ |
| CGPB | $1.52e-3 \pm 8.02e-5$ | $2.70e-3 \pm 4.15e-5$ |
| **SCG** | $\mathbf{1.81e-3 \pm 1.24e-4}$ | $\mathbf{2.69e-3 \pm 4.50e-5}$ |
| SR1 | $1.74e-3 \pm 9.67e-5$ | $2.64e-3 \pm 5.57e-5$ |
| DFP | $1.59e-3 \pm 1.52e-4$ | $2.64e-3 \pm 8.69e-5$ |
| **BFG** | $\mathbf{1.36e-3 \pm 3.23e-4}$ | $\mathbf{2.98e-3 \pm 4.76e-4}$ |
| OSS | $8.88e-3 \pm 3.27e-3$ | $1.43e-2 \pm 4.15e-3$ |

Tabela 6.12: Rezultate experimentale pentru egalizarea cu canal liniar

unde $s(k)$, $s(k-1)$, şi $s(k-2)$ sunt intrări, $z(k)$ este ieşirea neliniară, şi $n(k)$ este un zgomot alb gaussian adăugat. Pentru experimentele noastre, am generat $10000$ de tipare de antrenare uniform distribuite din constelaţia 4-QAM $\{\pm 1 \pm i\}$ având un SNR de $20$, şi $5000$ de tipare de testare având un SNR de $16$. După cum se poate observa cu uşurinţă, ordinul canalului este $3$, iar întârzierea a fost aleasă să fie $1$. Reţelele au avut aceleaşi arhitecturi ca cele din setul precedent de experimente.

Rezultatele obţinute după rularea de $50$ de ori a fiecărui algoritm, sunt date în aceeaşi formă ca mai sus, în Tabelul 6.13, şi respectiv în Figura 6.12. Tabelul mai prezintă şi rezultatele altor arhitecturi de reţele neuronale folosite pentru a învăţa aceeaşi problemă. Dinamica învăţării este prezentată grafic în Figura 6.13, diferenţiat pe fiecare clasă de algoritmi.

### 6.1.11 Predicţia seriilor de timp liniare

O altă aplicaţie importantă a reţelelor neuronale cu valori complexe este la predicţia semnalelor liniare şi neliniare. Un benchmark foarte cunoscut, propus în [170], şi folosit în [105, 93, 95, 98, 96, 266], este predicţia zgomotului alb cu valori complexe de medie zero şi varianţă unitară $n(k)$, trecut prin filtrul autoregresiv stabil dat prin

$$\begin{aligned} y(k) &= 1.79y(k-1) - 1.85y(k-2) \\ &\quad +1.27y(k-3) - 0.41y(k-4) + n(k). \end{aligned}$$

Zgomotul a fost generat astfel încât varianţa semnalului ca întreg să fie $1$, ţinând cont de faptul că $\sigma^2 = (\sigma^R)^2 + (\sigma^I)^2$. Numărul de intrări ale filtrului a fost ales să fie $4$. Ca o consecinţă, reţelele au avut $4$ intrări, $4$ neuroni pe un singur strat ascuns şi o ieşire. Funcţia de activare pentru stratul ascuns a fost funcţia tangentă hiperbolică complexă complet, iar pentru stratul de ieşire tot funcţia identitate. Antrenarea a fost făcută folosind $5000$ de tipare de antrenare şi a fost repetată de $50$ de ori.

| Algoritm | EMP Antrenare | EMP Testare |
|---|---|---|
| **GD** | $\mathbf{1.47e-1 \pm 7.59e-2}$ | $\mathbf{1.61e-1 \pm 8.24e-2}$ |
| GDM | $1.25e-1 \pm 7.16e-2$ | $1.35e-1 \pm 7.80e-2$ |
| QCP | $2.05e-2 \pm 1.63e-2$ | $2.51e-2 \pm 1.89e-2$ |
| **RPR** | $\mathbf{1.23e-2 \pm 1.23e-2}$ | $\mathbf{1.58e-2 \pm 1.47e-2}$ |
| DBD | $1.89e-2 \pm 2.13e-2$ | $2.31e-2 \pm 2.45e-2$ |
| SAB | $1.75e-1 \pm 5.49e-2$ | $1.91e-1 \pm 6.11e-2$ |
| CGHS | $7.07e-4 \pm 1.28e-3$ | $1.63e-3 \pm 1.53e-3$ |
| CGPR | $3.83e-4 \pm 1.23e-4$ | $1.25e-3 \pm 2.65e-4$ |
| CGFR | $1.85e-4 \pm 1.20e-4$ | $8.86e-4 \pm 2.67e-4$ |
| CGDY | $1.11e-4 \pm 1.16e-4$ | $7.91e-4 \pm 3.95e-4$ |
| CGHS+ | $1.74e-4 \pm 1.08e-4$ | $8.89e-4 \pm 2.98e-4$ |
| CGPR+ | $3.75e-4 \pm 9.28e-5$ | $1.23e-3 \pm 2.16e-4$ |
| **CGPB** | $\mathbf{8.23e-5 \pm 8.99e-5}$ | $\mathbf{6.59e-4 \pm 2.98e-4}$ |
| **SCG** | $\mathbf{2.71e-4 \pm 8.19e-5}$ | $\mathbf{1.13e-3 \pm 2.22e-4}$ |
| SR1 | $3.48e-5 \pm 1.63e-5$ | $8.28e-4 \pm 1.58e-4$ |
| DFP | $6.93e-7 \pm 7.08e-7$ | $6.21e-4 \pm 1.82e-4$ |
| **BFG** | $\mathbf{5.70e-7 \pm 8.07e-7}$ | $\mathbf{8.14e-4 \pm 3.62e-4}$ |
| OSS | $3.11e-4 \pm 9.22e-5$ | $1.13e-3 \pm 2.22e-4$ |
| CRBF [231, 230] | $3.17e-1$ | $3.56e-1$ |
| C-ELM (RBF) [231, 230] | $3.27e-1$ | $3.33e-1$ |
| FC-RBF [231, 230] | $1.60e-1$ | $1.68e-1$ |
| FCRN [234] | $9.00e-2$ | $1.22e-1$ |
| **Mc-FCRBF** [232] | $\mathbf{2.04e-2}$ | $\mathbf{2.77e-2}$ |
| CSRAN [244] | $4.00e-2$ | $1.44e-1$ |
| CMRAN [231, 230] | $1.60e-1$ | $1.84e-1$ |

Tabela 6.13: Rezultate experimentale pentru egalizarea cu canal neliniar

Rezultatele după aceste rulări ale fiecărui algoritm sunt date în Tabelul 6.14. În tabel am prezentat o măsură a performanţei de predicţie numită *câştigul în predicţie*, definit prin

$$R_p = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2},$$

unde $\sigma_x^2$ reprezintă varianţa semnalului de intrare şi $\sigma_e^2$ reprezintă varianţa erorii de predicţie, fiind exprimat în dB. Evident, datorită modului în care este definit, un câştig în predicţie mai mare relevă o performanţă mai bună. În acelaşi tabel sunt citate şi rezultatele altor algoritmi clasici din literatura de specialitate folosiţi pe aceeaşi problemă. Câştigul în predicţie pentru algoritmii implementaţi este ilustrat grafic în Figura 6.14.

### 6.1.12 Predicţia seriilor de timp neliniare

Folosim acum un benchmark pentru predicţia cu valori complexe neliniară, propus în [180], şi apoi utilizat în [105, 93, 95, 98]. În experimentele noastre problema de predicţie implică trecerea ieşirii filtrului autoregresiv dat de

$$\begin{aligned} y(k) &= 1.79y(k-1) - 1.85y(k-2) \\ &\quad +1.27y(k-3) - 0.41y(k-4) + n(k), \end{aligned}$$

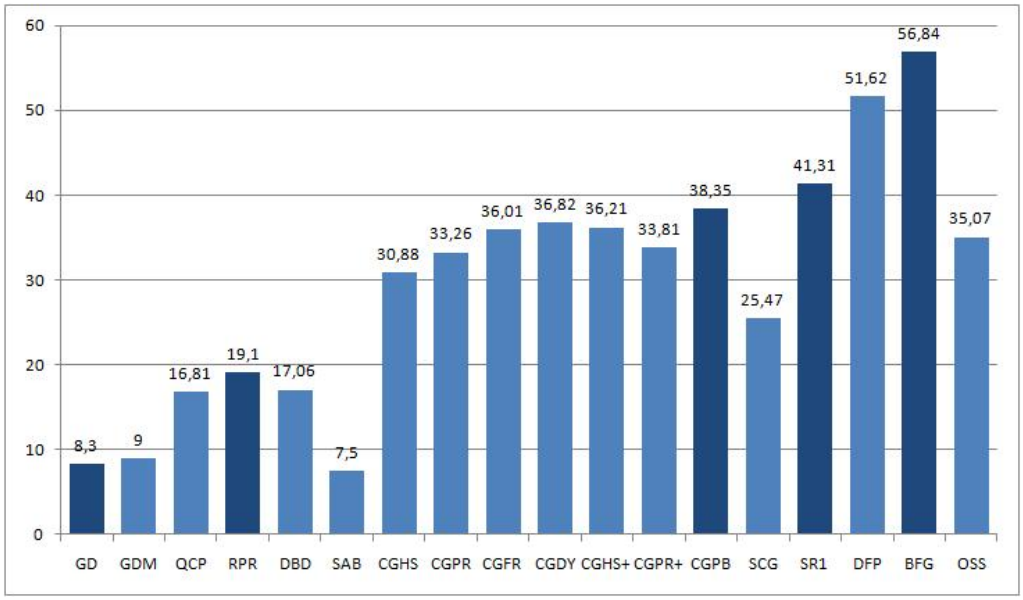


Figura 6.12: Rezultate experimentale pentru egalizarea cu canal neliniar

prin neliniaritatea dată de

$$z(k) = \frac{z(k-1)}{1+z^2(k-1)} + y^3(k).$$

Zgomotul cu valori complexe de la intrare $n(k)$ a fost generat ca în setul anterior de experimente, cu medie zero şi varianţă unitară. Numărul de intrări ale filtrului a fost de asemenea $4$, şi deci reţelele au avut aceleaşi arhitecturi şi funcţii de activare ca mai înainte.

Rezultatele, prezentate în Tabelul 6.15 şi în Figura 6.15, calculate după $50$ de rulări ale fiecărui algoritm, sunt foarte asemănătoare cu cele din experimentul precedent. Dinamica învăţării în acest caz este dată în aceeaşi formă ca mai înainte în Figura 6.16. În tabel este prezentat câştigul în predicţie definit, ca în setul precedent de experimente, prin

$$R_p = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2},$$

unde $\sigma_x^2$ reprezintă varianţa semnalului de intrare şi $\sigma_e^2$ reprezintă varianţa erorii de predicţie, fiind exprimat în dB. Tabelul prezintă, de asemenea, câştigul în predicţie pentru alţi algoritmi folosiţi pentru această problemă.

### 6.1.13 Predicţia direcţiei şi vitezei vântului

În fine, ultima aplicaţie din lumea reală pe care vom testa algoritmii propuşi este problema predicţiei vitezei şi direcţiei vântului, care a fost discutată în [93, 99, 94, 95, 263, 98, 97, 172, 169, 150, 267, 228].

Datele de antrenare au fost generate utilizând un anemometru ultrasonic, care a fost folosit pentru a măsura viteza vântului pe direcţia nord-sud ($V_N$) şi respectiv pe

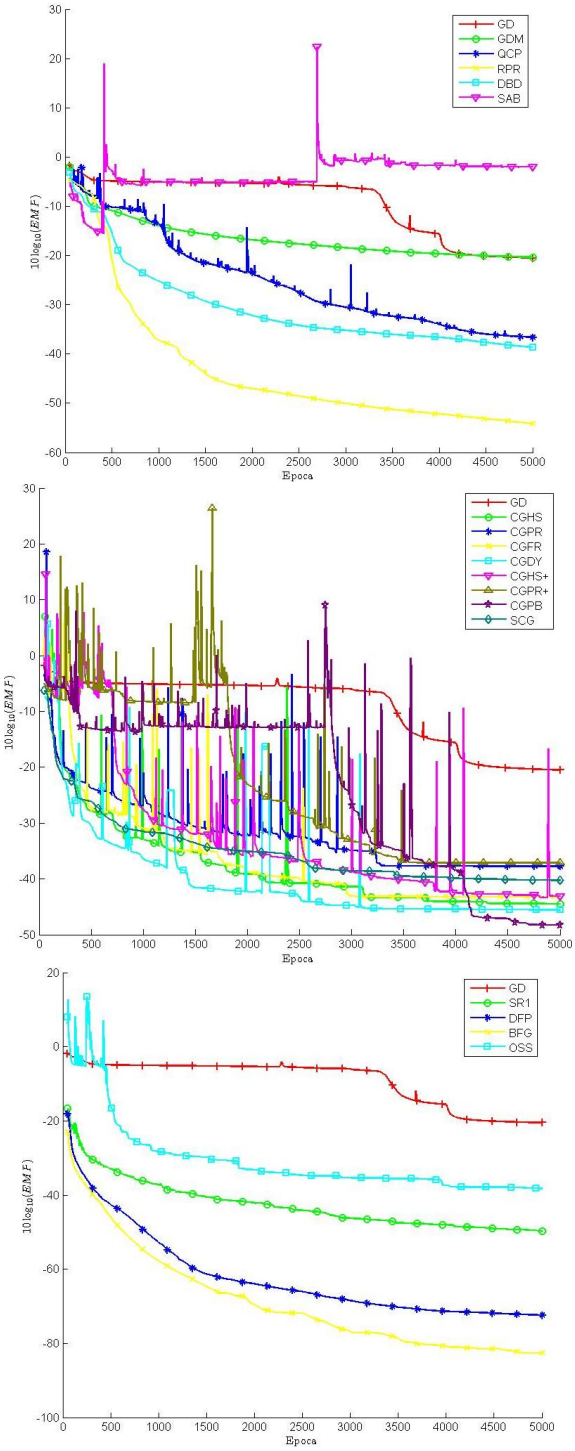


Figura 6.13: Dinamica învăţării pentru egalizarea cu canal neliniar

| Algoritm | Câştigul în predicţie |
|---|---|
| **GD** | $\mathbf{3.57 \pm 4.12e-1}$ |
| GDM | $3.62 \pm 4.01e-1$ |
| QCP | $8.29 \pm 1.96e-3$ |
| **RPR** | $\mathbf{8.29 \pm 1.69e-3}$ |
| DBD | $8.27 \pm 7.53e-3$ |
| SAB | $8.28 \pm 3.76e-3$ |
| **CGHS** | $\mathbf{8.29 \pm 8.66e-4}$ |
| CGPR | $6.69 \pm 6.18e-1$ |
| CGFR | $8.29 \pm 9.68e-4$ |
| CGDY | $8.29 \pm 1.00e-3$ |
| CGHS+ | $8.29 \pm 1.09e-3$ |
| CGPR+ | $6.43 \pm 5.52e-1$ |
| CGPB | $8.29 \pm 1.18e-3$ |
| **SCG** | $\mathbf{8.30 \pm 7.84e-4}$ |
| SR1 | $8.30 \pm 2.09e-3$ |
| DFP | $8.30 \pm 1.44e-3$ |
| **BFG** | $\mathbf{8.32 \pm 5.34e-3}$ |
| OSS | $8.29 \pm 7.54e-4$ |
| CLMS [264] | 2.99 |
| CNGD [98] | 3.10 |
| CRTRL [92] | 3.54 |
| **ACRTRL** [96] | **4.10** |

Tabela 6.14: Rezultate experimentale pentru predicţia seriilor de timp liniare

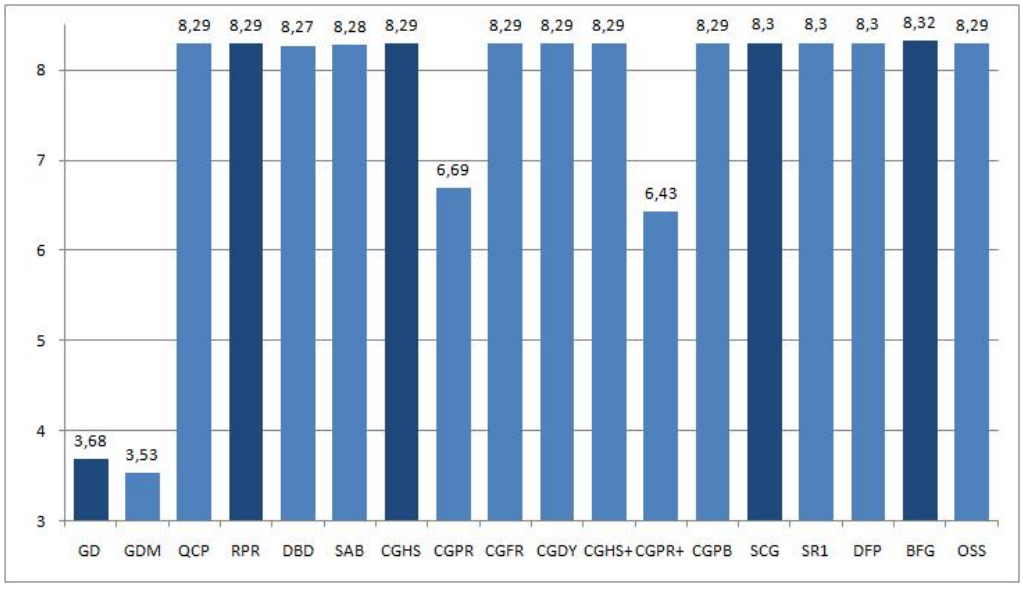


Figura 6.14: Rezultate experimentale pentru predicţia seriilor de timp liniare

direcţia est-vest ($V_E$), care au fost folosite pentru crea semnalul cu valori complexe

| Algoritm | Câştigul în predicţie |
|---|---|
| **GD** | $\mathbf{3.95 \pm 4.67e-1}$ |
| GDM | $3.98 \pm 4.71e-1$ |
| QCP | $8.35 \pm 1.25e-3$ |
| **RPR** | $\mathbf{8.35 \pm 6.45e-4}$ |
| DBD | $8.34 \pm 5.28e-3$ |
| SAB | $8.35 \pm 2.18e-3$ |
| CGHS | $8.35 \pm 8.34e-4$ |
| CGPR | $8.31 \pm 2.59e-2$ |
| CGFR | $8.34 \pm 7.49e-4$ |
| CGDY | $8.35 \pm 1.09e-3$ |
| CGHS**+** | $8.35 \pm 6.61e-4$ |
| CGPR+ | $6.45 \pm 3.90e-1$ |
| **CGPB** | $\mathbf{8.35 \pm 4.11e-4}$ |
| **SCG** | $\mathbf{8.35 \pm 3.51e-4}$ |
| SR1 | $8.36 \pm 2.52e-3$ |
| DFP | $8.35 \pm 8.87e-4$ |
| **BFG** | $\mathbf{8.38 \pm 5.84e-3}$ |
| OSS | $8.35 \pm 7.32e-4$ |
| CLMS [264] | 1.87 |
| CNGD [98] | 2.50 |
| **CRTRL** [92] | **3.76** |

Tabela 6.15: Rezultate experimentale pentru predicţia seriilor de timp neliniare

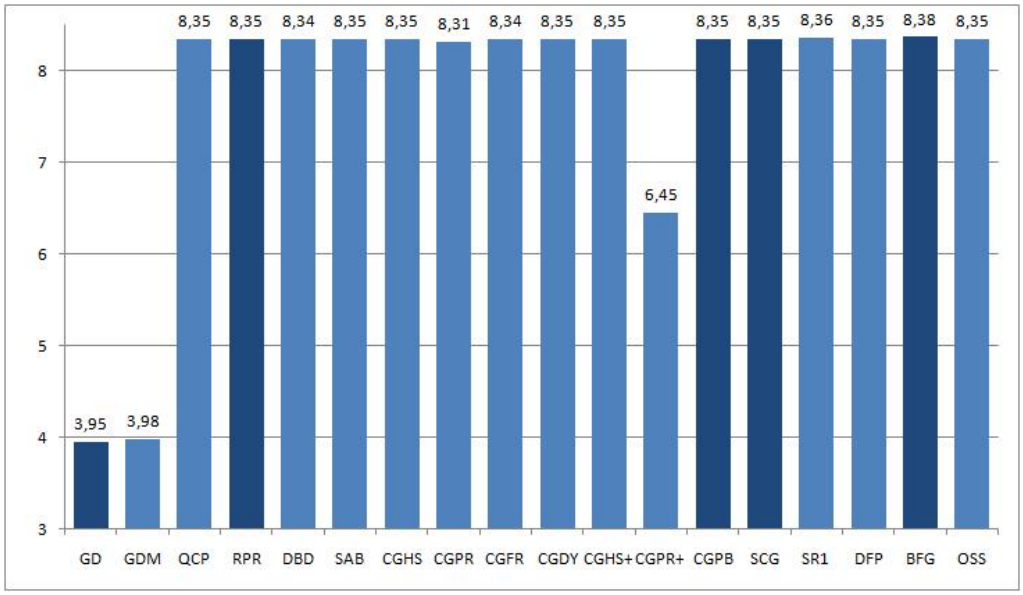


Figura 6.15: Rezultate experimentale pentru predicţia seriilor de timp neliniare

$V$, astfel:

$$V = V_E + iV_N.$$

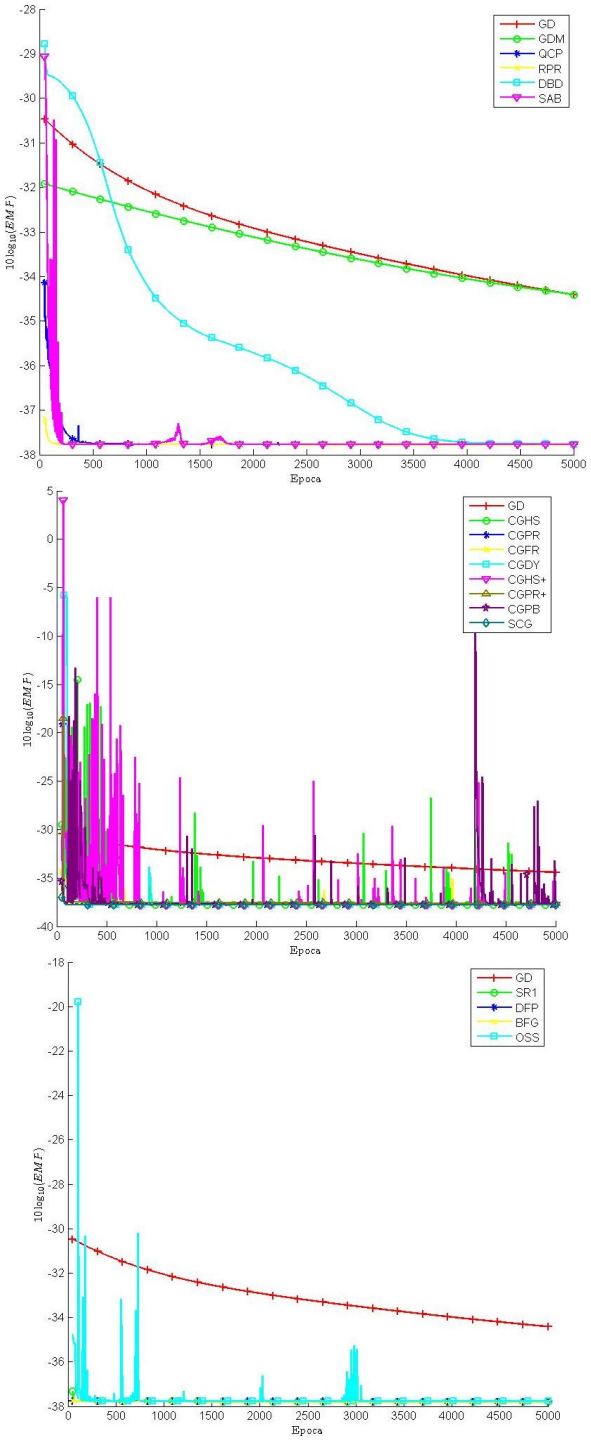


Figura 6.16: Dinamica învăţării pentru predicţia seriilor de timp neliniare

Acesta va fi semnalul care urmează să fie prezis folosind reţele neuronale cu valori complexe. Setul de date conţine $5000$ de tipare[1] care sunt folosite pentru a antrena o reţea de tip feedforward utilizând algoritmii propuşi. Numărul de intrări a fost ales să fie $10$, deci predicţia se bazează pe ultimele $10$ valori înregistrate, şi astfel reţelele au $10$ intrări, $15$ neuroni pe un singur strat ascuns şi o ieşire.

După $50$ de rulări, pe parcursul a $5000$ de epoci, ale fiecărui algoritm, rezultatele sunt înfăţişate în Tabelul 6.16 şi în Figura 6.17. În tabel şi în figură este prezentat acelaşi câştig în predicţie $R_p$ definit mai sus. În tabel mai apare, de asemenea, şi câştigul în predicţie al altor algoritmi folosiţi pentru aceeaşi aplicaţie.

| Algoritm | Câştigul în predicţie |
|---|---|
| **GD** | $\mathbf{7.61 \pm 1.05e-2}$ |
| GDM | $7.69 \pm 9.67e-3$ |
| QCP | $6.97 \pm 4.72e-1$ |
| RPR | $7.63 \pm 2.17e-2$ |
| **DBD** | $\mathbf{7.75 \pm 1.50e-2}$ |
| SAB | $7.68 \pm 3.31e-2$ |
| CGHS | $8.12 \pm 1.64e-2$ |
| CGPR | $7.96 \pm 1.70e-2$ |
| CGFR | $8.13 \pm 1.26e-2$ |
| **CGDY** | $\mathbf{8.15 \pm 1.80e-2}$ |
| CGHS**+** | $8.12 \pm 1.83e-2$ |
| CGPR+ | $7.97 \pm 1.58e-2$ |
| CGPB | $8.13 \pm 1.92e-2$ |
| **SCG** | $\mathbf{8.19 \pm 1.18e-2}$ |
| SR1 | $8.17 \pm 3.20e-2$ |
| **DFP** | $\mathbf{8.23 \pm 4.30e-2}$ |
| BFG | $8.17 \pm 1.07e-1$ |
| OSS | $8.00 \pm 1.29e-2$ |
| CLMS [264] | $5.14$ |
| ACLMS [172] | $5.35$ |
| D-CMLS [267] | $5.66$ |
| **D-ACMLS** [267] | $\mathbf{5.93}$ |

Tabela 6.16: Rezultate experimentale pentru predicţia direcţiei şi vitezei vântului

## 6.2 Reţele neuronale cu valori matrici pătratice

### 6.2.1 Funcţia sintetică I

În experimentele noastre, am considerat matrici pătratice de ordinul $3$. Prima funcţie pe care vom testa algoritmul propus este media aritmetică a două matrici

$$f_1(A, B) = \frac{1}{2}(A + B).$$

Pentru antrenarea reţelei neuronale cu valori matriciale, am generat $3000$ de tipare de antrenare cu elemente aleatoare între $0$ şi $1$. Setul de testare a conţinut $1000$ de tipare generate în acelaşi fel. Reţeaua a avut $15$ neuroni pe un singur strat ascuns. Funcţia de activare pentru stratul ascuns a fost funcţia tangentă hiperbolică pe

[1] Disponibile la adresa http://www.commsp.ee.ic.ac.uk/∼mandic/wind-dataset.zip.

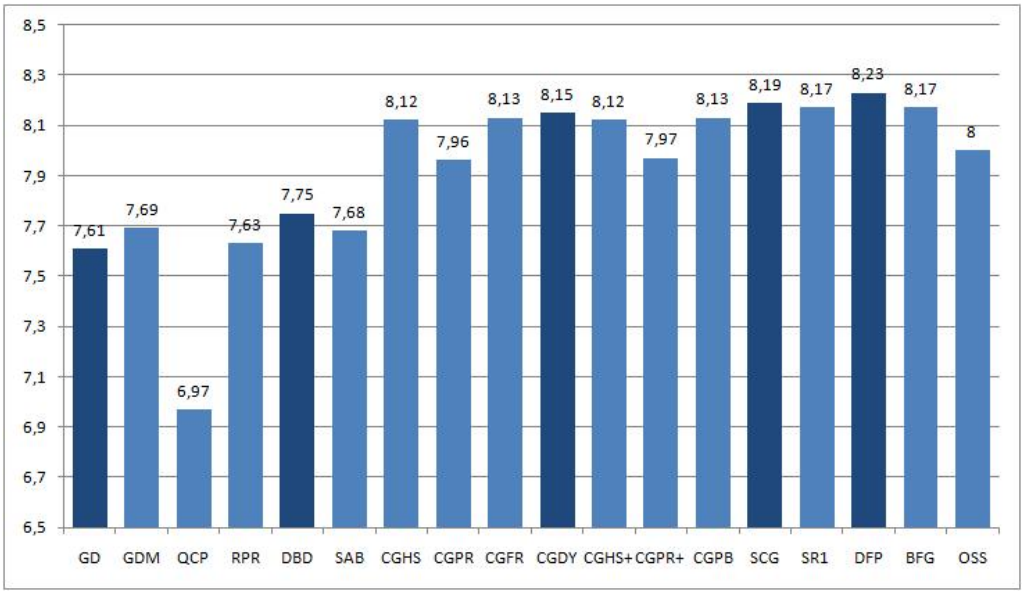


Figura 6.17: Rezultate experimentale pentru predicţia direcţiei şi vitezei vântului

componente dată prin

$$G^2\left(\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\right) = \begin{pmatrix} \tanh a_{11} & \tanh a_{12} & \tanh a_{13} \\ \tanh a_{21} & \tanh a_{22} & \tanh a_{23} \\ \tanh a_{31} & \tanh a_{32} & \tanh a_{33} \end{pmatrix},$$

iar pentru stratul de ieşire, funcţia de activare a fost funcţia identică: $G^3(S) = S$.

Experimentul a arătat că reţeaua neuronală converge, şi că eroarea medie pătratică (EMP) atât pentru setul de antrenare, cât şi pentru setul de testare a fost aproximativ la fel, şi egală cu $0.009030$. Antrenarea a fost făcută pe parcursul a $1000$ de epoci. Deşi rezultatul nu este spectaculos, trebuie să ţinem cont de faptul că fiecare matrice este formată din $9$ numere reale, spre deosebire de două sau patru numere reale din cazul numerelor complexe, hiperbolice, şi respectiv cuaternionilor.

### 6.2.2 Funcţia sintetică II

O funcţie mai complicată este funcţia pătratică de două variabile

$$f_2(A, B) = A^2 + B^2.$$

Acest exemplu a fost inspirat de funcţiile sintetice pe care au fost testate reţelele neuronale cu valori complexe respectiv cuaternionice. Matricile pătratice de ordinul $3$ care compun seturile de antrenare şi de testare au fost generate aleator cu elemente între $0$ şi $1$, $3000$ pentru setul de antrenare şi $1000$ pentru setul de testare. Funcţiile de activare au fost aceleaşi cu cele din experimentul anterior. Arhitectura reţelei a avut $15$ neuroni pe un singur strat ascuns, şi a fost antrenată $1000$ de epoci.

Ca în experimentul anterior, EMP pentru seturile de antrenare şi de testare a avut valori similare, egale în acest caz cu $0.009897$. Performanţa este uşor mai slabă decât cea obţinută anterior, dar, în acest caz, funcţia a fost mai complicată. Aceste rezultate

dau motive de speranţă că în viitor aceste reţele pot fi optimizate pentru a avea performanţe mai bune pe probleme de aproximare a funcţiilor cu valori matriciale.

### 6.2.3 Funcţia sintetică III

Ultimul experiment vizează inversul unui singur argument matricial:

$$f_3(A) = A^{-1}.$$

Acest experiment a fost făcut avându-se în vedere faptul că inversa unei matrici este o operaţie complicată, care este însă necesară în multe aplicaţii. Am vrut să vedem dacă inversa unei matrici poate fi învăţată de o reţea neuronală, în particular de o reţea neuronală cu valori matriciale.

Rezultatele nu au fost la fel de bune ca cele din experientele anterioare, dar au dat motive de optimism. Reţeaua a fost antrenată folosind $6000$ de matrici pătratice de ordinul $3$, generate în acelaşi fel ca în experimentele de mai sus. O arhitectură de reţea cu un singur strat ascuns compus din $15$ neuroni a fost antrenată timp de $1000$ de epoci. Eroarea medie pătratică (EMP) pentru seturile de antrenare şi testare a fost de $0.021971$, mai mare decât în experimentele anterioare, dar promiţătoare ţinând cont de complexitatea problemei care a trebuit învăţată, şi de numărul relativ mic de tipare de antrenare.

## 6.3 Reţele neuronale cu valori matrici antisimetrice

### 6.3.1 Funcţia sintetică I

În experimentele noastre, lucrăm cu algebra Lie $\mathfrak{so}(3)$, care este formată din matricile antisimetrice de ordinul $3$. Prima funcţie pe care vom testa algoritmul propus este simpla medie aritmetică a două matrici

$$f_1(A, B) = \frac{1}{2}(A + B).$$

Pentru antrenarea reţelei neuronale cu valori matrici antisimetrice, am generat $3000$ de tipare de antrenare cu elemente aleatoare între $0$ şi $1$. Setul de testare a conţinut $1000$ de tipare generate în acelaşi fel. Reţeaua a avut $15$ neuroni pe un singur strat ascuns. Funcţia de activare pentru stratul ascuns a fost funcţia tangentă hiperbolică pe componente dată prin

$$G^2\left(\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & \tanh x & \tanh y \\ -\tanh x & 0 & \tanh z \\ -\tanh y & -\tanh z & 0 \end{pmatrix},$$

iar pentru stratul de ieşire, funcţia de activare a fost identitatea: $G^3(S) = S$.

Experimentul a arătat că reţeaua neuronală converge, iar eroarea medie pătratică (EMP) atât pentru setul de antrenare cât şi pentru setul de testare a fost aproximativ aceeaşi, şi egală cu $0.000792$. Antrenarea a fost făcută timp de $1000$ de epoci. Deşi rezultatul nu este spectaculos, trebuie să ţinem cont de faptul că fiecare matrice este formată din $3$ numere reale.

### 6.3.2 Funcţia sintetică II

A doua funcţie de test este reprezentată prin paranteza Lie dintre două argumente matriciale:

$$f_2(A, B) = [A, B].$$

Matricile antisimetrice de ordin $3$ care compun seturile de antrenare şi de testare au fost generate aleator cu elemente între $0$ şi $1$, $3000$ pentru setul de antrenare, şi $1000$ pentru setul de testare. Funcţiile de activare au fost aceleaşi ca mai sus. Arhitectura reţelei a avut $15$ neuroni pe un singur strat ascuns, şi a fost antrenată de-a lungul a $1000$ de epoci.

Ca în experimentul de mai sus, EMP-urile de antrenare şi testare au avut valori similare, egale în acest caz cu $0.01504$. Performanţa este mai slabă decât cea obţinută în experimentul anterior, dar, în acest caz, funcţia a fost mai complicată. Aceste rezultate dau motive de speranţă că în viitor aceste reţele pot fi optimizate pentru a avea performanţe mai bune în aproximarea de funcţii cu valori în algebre Lie.

### 6.3.3 Translaţia

Următoarele trei experimente sunt făcute cu transformări geometrice. Pentru o matrice arbitrară $\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix} \in \mathfrak{so}(3)$, definim următoarea funcţie

$$f_3\left(\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & x+1 & y+2 \\ -x-1 & 0 & z+3 \\ -y-2 & -z-3 & 0 \end{pmatrix},$$

care corespunde unei translaţii a vectorului $(x, y, z)^T$ cu vectorul $(1, 2, 3)^T$. Acest tip de funcţie, şi următoarele două, sunt similare cu funcţiile complexe divizat din domeniul reţelelor neuronale cu valori complexe, spre deosebire de funcţiile complexe complet, care sunt similare cu cele folosite în primele două experimente.

Arhitectura reţelei folosite pentru a învăţa această translaţie a fost identică cu cea de mai sus, cu $3000$ de tipare de antrenare şi $1000$ de tipare de testare, generate în aceeaşi manieră. După cum era de aşteptat, dată fiind simplitatea funcţiei, rezultatele au fost bune. Reţeaua a obţinut o eroare medie pătratică (EMP) de $0.001116$ pe seturile de antrenare şi de testare.

### 6.3.4 Scalarea

A doua transformare geometrică pe care vom testa reţeaua noastră este scalarea. Ca mai sus, pentru o matrice arbitrară $\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix} \in \mathfrak{so}(3)$, definim funcţia

$$f_4\left(\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & x & 2y \\ -x & 0 & 3z \\ -2y & -3z & 0 \end{pmatrix},$$

care corespunde unei scalări a vectorului $(x, y, z)^T$ cu un factor de $1$, $2$, şi respectiv $3$ pe fiecare dintre cele trei axe.

Rezultatele nu au fost la fel de bune ca în experimentele anterioare, dar au dat motive de optimism. Reţeaua a fost antrenată folosind $3000$ de matrici pătratice antisimetrice de ordinul $3$, generate în acelaşi fel ca cele din experimentele anterioare. O arhitectură de reţea cu un singur strat ascuns compus din $15$ neuroni a fost antrenată de-a lungul a $1000$ de epoci. EMP-ul pentru seturile de antrenare şi de testare a fost de $0.1258$, mai mare decât în experimentele de mai sus, dar promiţătoare ţinând cont de complexitatea problemei de învăţat şi de numărul relativ mic de tipare de antrenare.

### 6.3.5 Rotaţia

În fine, considerăm rotaţia. Pentru orice $\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix} \in \mathfrak{so}(3)$, definim funcţia

$$f_5\left(\begin{pmatrix} 0 & x & y \\ -x & 0 & z \\ -y & -z & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & \frac{\sqrt{3}}{4}x - \frac{1}{4}y + \frac{\sqrt{3}}{2}z & \frac{3}{4}x - \frac{\sqrt{3}}{4}y - \frac{1}{2}z \\ -\frac{\sqrt{3}}{4}x + \frac{1}{4}y - \frac{\sqrt{3}}{2}z & 0 & \frac{1}{2}x + \frac{\sqrt{3}}{2}y \\ -\frac{3}{4}x + \frac{\sqrt{3}}{4}y + \frac{1}{2}z & -\frac{1}{2}x - \frac{\sqrt{3}}{2}y & 0 \end{pmatrix},$$

care corespunde unei rotaţii a vectorului $(x, y, z)^T$ cu un unghi de $\frac{\pi}{2}$ după axa $x$, de $\frac{\pi}{3}$ după axa $y$, şi de $\frac{\pi}{6}$ după axa $z$.

Reţeaua a avut aceleaşi caracteristici ca mai sus, şi a obţinut cele mai bune rezultate: o EMP pentru seturile de antrenare şi de testare de $0.000238$, care recomandă acest tip de reţea pentru a învăţa transformări geometrice, şi în special rotaţii.

## 6.4 Medierea matricilor ortogonale

Pentru funcţia de cost de tip $L^p$, vom analiza schimbările care apar în problema de mediere pentru cazul a trei rotaţii în jurul axei $x$ când $p = 2$, respectiv $p = 4$.

Astfel, considerăm următoarele rotaţii în jurul axei $x$:

$$\mathbf{R}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \mathbf{R}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \mathbf{R}_3(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix},$$

unde $\alpha \in [-\pi, \pi]$, care au unghiurile de rotaţie $\theta_1 = \pi$, $\theta_2 = \frac{\pi}{2}$, şi $\theta_3(\alpha) = \alpha$. Cuaternionii asociaţi acestor rotaţii sunt, de exemplu:

$$\mathbf{q}_1 = (0, 1, 0, 0), \mathbf{q}_2 = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0, 0), \mathbf{q}_3 = (\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}, 0, 0),$$

unde $\alpha \in [-\pi, \pi]$, ca mai sus.

În continuare, vom găsi punctele critice pentru funcţiile de cost $G_{(p)_{SO(3)}}$ în cazul $p = 2$, respectiv $p = 4$ când $\alpha$ variază între $-\pi$ şi $\pi$. Conform algoritmului de încorporare, este suficient să studiem punctele critice pentru funcţia corespunzătoare liftată $G_{(2)_{S^3}}$, respectiv $G_{(4)_{S^3}}$. Mai mult, aceasta este echivalent cu rezolvarea sistemului de ecuaţii (5.5.17) pentru $p = 2$ şi $p = 4$.

Începem cu cazul $p = 2$. Rezolvând sistemul obţinem cinci familii de puncte critice:

$\text{Set}_{black} = \{(0, 0, \pm\sqrt{1 - t^2}, t) \,|\, t \in [-1, 1]\}$;

$\text{Set}_{green} = \{(\sqrt{1 - x_{2,min}^2(\alpha)}, x_{2,min}(\alpha), 0, 0) \,|\, \alpha \in [-\pi, \pi]\}$;

$\text{Set}_{pink} = \{(-\sqrt{1 - x_{2,min}^2(\alpha)}, x_{2,min}(\alpha), 0, 0) \,|\, \alpha \in [-\pi, \pi]\}$;

$\text{Set}_{red} = \{(\sqrt{1 - x_{2,max}^2(\alpha)}, x_{2,max}(\alpha), 0, 0) \,|\, \alpha \in [-\pi, \pi]\}$;

$\text{Set}_{blue} = \{(-\sqrt{1 - x_{2,max}^2(\alpha)}, x_{2,max}(\alpha), 0, 0) \,|\, \alpha \in [-\pi, \pi]\}$,

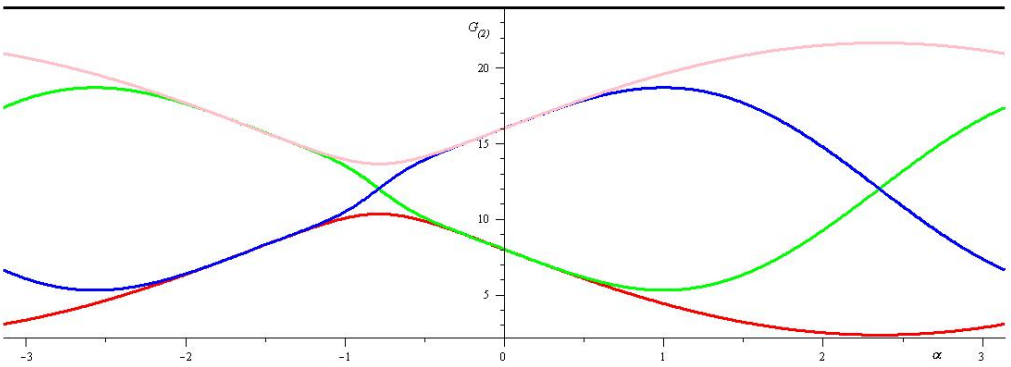


Figura 6.18: Valorile lui $G_{(2)_{SO(3)}}$ pe mulţimile de puncte critice

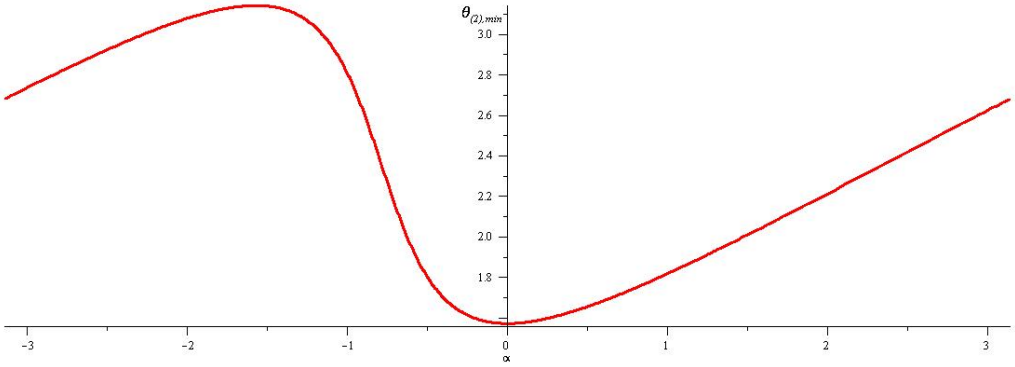


Figura 6.19: Unghiul de rotaţie $\theta_{(2),min}$ pentru rotaţia la care minimul este atins

unde $x_{2,min}(\alpha)$ şi $x_{2,max}(\alpha)$ sunt cea mai mică, respectiv cea mai mare soluţie reală pozitivă a polinomului

$$\begin{aligned} Q_{2,\alpha}(Z) &= \left(128\sin^4\frac{\alpha}{2} - 32\sin^2\frac{\alpha}{2} + 4\right)Z^4 - \left(128\sin^4\frac{\alpha}{2} - 32\sin^2\frac{\alpha}{2} + 4\right)Z^2 \\ &- 16\sin^6\frac{\alpha}{2} + 16\sin^5\frac{\alpha}{2}\cos\frac{\alpha}{2} + 28\sin^4\frac{\alpha}{2} - 8\sin^2\frac{\alpha}{2} + 1. \end{aligned}$$

Datorită simetriei polinomului $Q_{2,\alpha}$, acesta are doar două rădăcini reale pozitive.

În Figura 6.18 reprezentăm valorile funcţiei de cost $G_{(2)_{SO(3)}}$ pe mulţimile de puncte critice descoperite mai sus cu parametrul $\alpha \in [-\pi, \pi]$. Observăm că maximul absolut este atins pe mulţimea $\text{Set}_{black}$ care reprezintă rotaţiile de unghi $\pi$ în jurul axelor care sunt perpendiculare pe axa $x$ iar acest maxim absolut nu depinde de parametrul $\alpha$. Pentru o valoare fixată a parametrului $\alpha \in [-\pi, \pi]$, obţinem un minim absolut pe mulţimea $\text{Set}_{red}$ care reprezintă rotaţii în jurul axei $x$. Notăm unghiul de rotaţie de-a lungul axei $x$ unde valoarea minimă a funcţiei cost $G_{(2)_{SO(3)}}$ este obţinută prin $\theta_{(2),min}$ şi dependenţa de parametrul $\alpha$ este reprezentată în Figura 6.19 de mai jos.

În continuare, vom analiza comportamentul unghiului de rotaţie $\theta_{(2),min}$ pentru

diferite configuraţii ale rotaţiilor considerate pentru mediere. Pentru $\alpha = -\pi$ (rotaţiile $\mathbf{R}_1$ şi $\mathbf{R}_3$ coincid) avem că valoarea minimă a funcţiei cost $G_{(2)_{SO(3)}}$ este atinsă pentru o rotaţie de-a lungul axei $x$ de unghi $\theta_{(2),min}(-\pi) = 2.677945044$. Când $\alpha \in [-\pi, -\frac{\pi}{2}]$ avem că $\theta_{(2),min}$ este o funcţie monoton crescătoare cu $\theta_{(2),min}(-\frac{\pi}{2}) = \pi$ (rotaţia minimă coincide cu rotaţia $\mathbf{R}_1$). Pentru $\alpha \in [-\frac{\pi}{2}, 0]$, funcţia $\theta_{(2),min}$ este monoton descrescătoare şi $\theta_{(2),min}(0) = \frac{\pi}{2}$ (rotaţia minimă coincide cu rotaţia $\mathbf{R}_2$). Dacă $\alpha \in [0, \pi]$, funcţia $\theta_{(2),min}$ este din nou monoton crescătoare cu $\theta_{(2),min}(\pi) = 2.677945044$. În consecinţă, unghiul $\theta_{(2),min}$ acoperă al doilea cadran, i.e. $\theta_{(2),min}([-\pi, \pi]) = [\frac{\pi}{2}, \pi]$.

Vom analiza acum cazul $p = 4$. Sistemul are o familie largă de puncte critice. Printre ele, obţinem ca mai înainte, mulţimea $\text{Set}_{black}$ unde maximul absolut pentru funcţia de cost $G_{(4)_{SO(3)}}$ este atins. În cele ce urmează, vom studia comportamentul funcţiei de cost $G_{(4)_{SO(3)}}$ doar pe mulţimile de puncte critice care reprezintă rotaţii de-a lungul axei $x$. Mulţimea de puncte critice care reprezintă rotaţii de-a lungul axei $x$ sunt construite, ca mai sus, dintre soluţiile polinomului

$$Q_{4,\alpha}(Z) = a_8 Z^8 + a_6 Z^6 + a_4 Z^4 + a_2 Z^2 + a_0,$$

unde

$$a_8 = 16;$$

$$a_6 = -128 \sin^4 \tfrac{\alpha}{2} + 96 \sin^2 \tfrac{\alpha}{2} - 128 \sin^3 \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} + 64 \sin \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} - 32;$$

$$a_4 = 192 \sin^4 \tfrac{\alpha}{2} - 128 \sin^2 \tfrac{\alpha}{2} + 192 \sin^3 \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} - 80 \sin \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} + 24;$$

$$a_2 = 32 \sin^6 \tfrac{\alpha}{2} - 112 \sin^4 \tfrac{\alpha}{2} + 48 \sin^2 \tfrac{\alpha}{2} - 80 \sin^3 \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} + 24 \sin \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} - 8;$$

$$a_0 = -16 \sin^8 \tfrac{\alpha}{2} + 16 \sin^6 \tfrac{\alpha}{2} + 8 \sin^3 \tfrac{\alpha}{2} \cos \tfrac{\alpha}{2} + 1.$$

Dând valori parametrului $\alpha$, găsim rădăcinile reale ale polinomului $Q_{4,\alpha}$. Pentru $\alpha \in [-\pi, -1.02) \cup (-0.55, \pi]$ avem două rădăcini pozitive reale $x_{4,min}(\alpha)$ şi $x_{4,max}(\alpha)$. Ca mai înainte, avem cele patru mulţimi de puncte critice $\text{Set}_{green}$, $\text{Set}_{pink}$, $\text{Set}_{red}$, şi $\text{Set}_{blue}$. Pentru $\alpha \in [-1.02, -0.55]$ găsim patru rădăcini pozitive reale $x_{4,min}(\alpha)$, $x_{4,*}(\alpha)$, $x_{4,**}(\alpha)$, $x_{4,max}(\alpha)$. Ca mulţimi de puncte critice, pentru $\alpha$ în acest interval, găsim pe $\text{Set}_{green}$, $\text{Set}_{pink}$, $\text{Set}_{red}$, $\text{Set}_{blue}$, şi încă patru mulţimi de puncte critice corespunzând lui $x_{4,*}(\alpha)$, respectiv $x_{4,**}(\alpha)$. Şi anume, avem mulţimile de puncte critice suplimentare corespunzând unor rotaţii de-a lungul axei $x$,

$$\text{Set}_{yellow} = \{(\sqrt{1 - x_{4,*}^2(\alpha)}, x_{4,*}(\alpha), 0, 0) \,|\, \alpha \in [-1.02, -0.55]\};$$

$$\text{Set}_{violet} = \{(-\sqrt{1 - x_{4,*}^2(\alpha)}, x_{4,*}(\alpha), 0, 0) \,|\, \alpha \in [-1.02, -0.55]\};$$

$$\text{Set}_{maroon} = \{(\sqrt{1 - x_{4,**}^2(\alpha)}, x_{4,**}(\alpha), 0, 0) \,|\, \alpha \in [-1.02, -0.55]\};$$

$$\text{Set}_{gold} = \{(-\sqrt{1 - x_{4,**}^2(\alpha)}, x_{4,**}(\alpha), 0, 0) \,|\, \alpha \in [-1.02, -0.55]\}.$$

Funcţia $\alpha \to x_{4,min}(\alpha)$ este continuă pe tot intervalul $[-\pi, \pi]$, şi în consecinţă rezultă că $G_{(4)_{S^3}}(\text{Set}_{green})$ şi $G_{(4)_{S^3}}(\text{Set}_{pink})$ sunt curbe continue. Discontinuitatea funcţiei $\alpha \to x_{4,max}(\alpha)$ implică discontinuităţi ale curbelor $G_{(4)_{S^3}}(\text{Set}_{red})$ şi $G_{(4)_{S^3}}(\text{Set}_{blue})$.

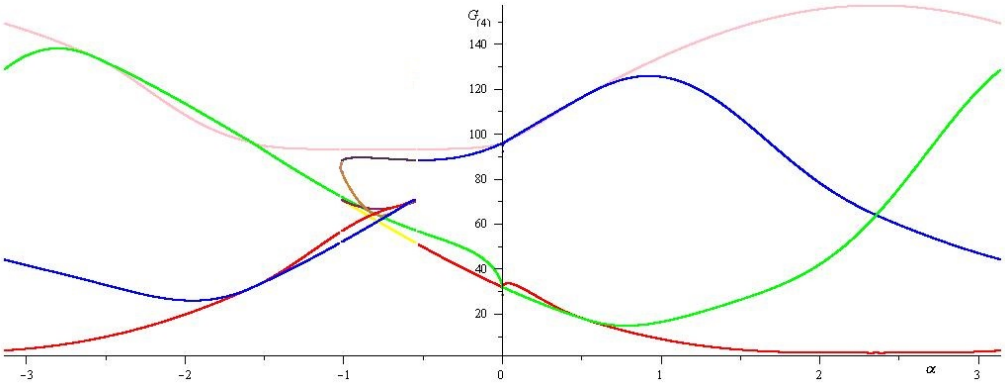


Figura 6.20: Valorile lui $G_{(4)_{SO(3)}}$ pe mulţimile de puncte critice

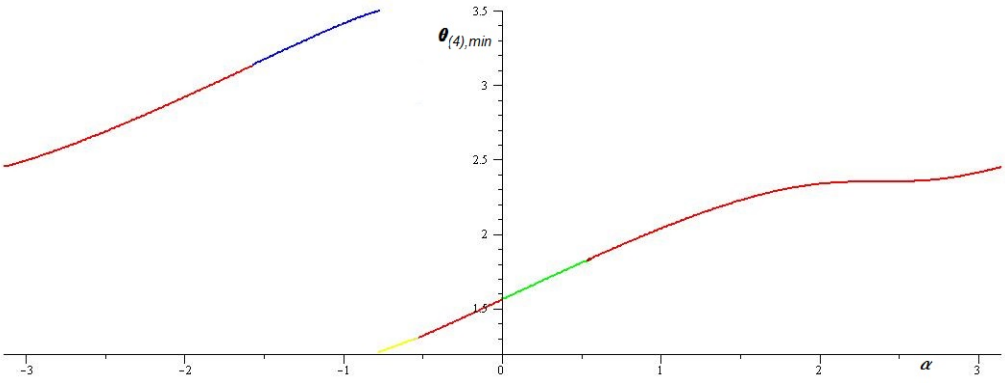


Figura 6.21: Unghiul de rotaţie $\theta_{(4),min}$ pentru rotaţia la care minimul este atins

În Figura 6.20 am reprezentat grafic valorile funcţiei de cost $G_{(4)_{SO(3)}}$ pe mulţimile critice corespunzătoare rotaţiilor de-a lungul axei $x$ când parametrul $\alpha \in [-\pi, \pi]$. Spre deosebire de cazul $p = 2$, pentru $p = 4$ avem o schimbare a mulţimilor de puncte critice unde minimul este atins. Notăm unghiul de rotaţie de-a lungul axei $x$ unde minimul funcţiei $G_{(4)_{SO(3)}}$ este atins prin $\theta_{(4),min}$ şi dependenţa lui de parametrul $\alpha$ este reprezentată în Figura 6.21 de mai jos.

În ceea ce priveşte comportamentul unghiului $\theta_{(4),min}$, există mai multe diferenţe majore faţă de comportamentul unghiului $\theta_{(2),min}$:

(i) Unghiul $\theta_{(4),min}$ schimbă culorile, i.e. valoarea minimă a funcţiei cost $G_{(4)_{SO(3)}}$ este obţinută pentru rotaţii corespunzătoare unor cuaternioni venind din diferite mulţimi de puncte critice având culorile aşa cum apar ele în Figura 6.21.

(ii) $\theta_{(4),min}$ este o funcţie monoton crescătoare şi continuă pe fiecare interval $[-\pi, -\frac{\pi}{4})$, respectiv $(-\frac{\pi}{4}, \pi]$.

(iii) Pentru $\alpha = -\frac{\pi}{4}$, spre deosebire de cazul $p = 2$, unghiul $\theta_{(4),min}$ are două valori, a se vedea Figura 6.22, şi prin urmare, $\alpha \to \theta_{(4),min}(\alpha)$ are două ramuri. Aceasta se

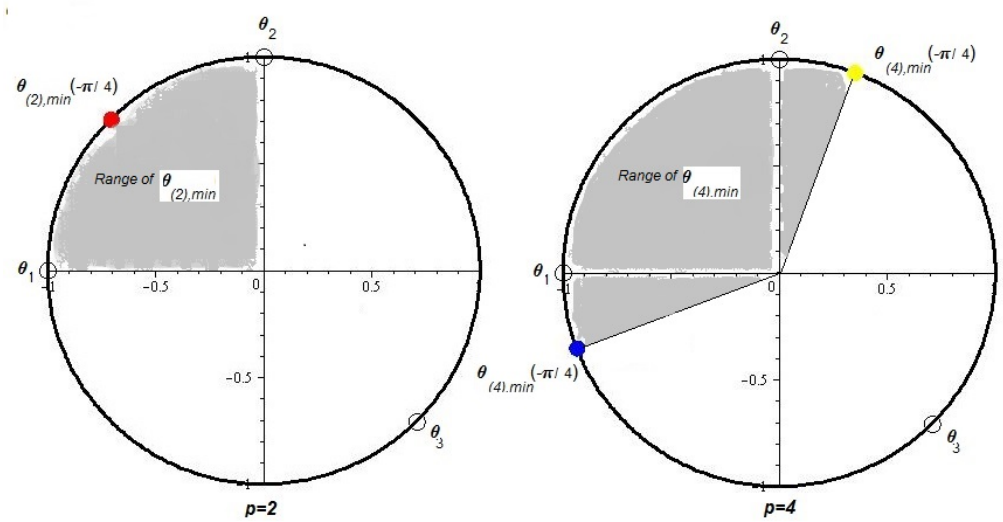


Figura 6.22: Non-unicitatea rotaţiei minimale pentru $\alpha = -\frac{\pi}{4}$ şi $p = 4$

datorează non-unicităţii punctului critic unde valoarea minimă a funcţiei de cost $G_{(4)_{SO(3)}}$ este atinsă. Mai precis, rotaţiile critice unde valoarea minimă este atinsă corespund cuaternionilor $(0.82, 0.56, 0, 0) \in \text{Set}_{yellow}$, respectiv $(-0.17, 0.98, 0, 0) \in \text{Set}_{blue}$.

(iv) Pentru cazul $p = 4$, unghiul $\theta_{(4),min}$ acoperă al doilea cadran şi părţi din primul şi al treilea cadran, a se vedea Figura 6.21 şi Figura 6.22.

## 6.5 Concluzii

Prezentul capitol a fost dedicat rezultatelor experimentale ale implementării algoritmilor prezentaţi în Capitolele 3 şi 5. Astfel,

- Secţiunea 6.1 a prezentat rezultatele experimentale ale antrenării unor **reţele neuronale cu valori complexe** de tip feedforward folosind **optsprezece** algoritmi prezentaţi în Capitolul 3, dintre care **cincisprezece** au fost implementaţi aici pentru prima dată. Aceşti algoritmi au fost folosiţi pentru învăţarea problemei XOR, a problemei XOR extinse, a două funcţii complexe complet şi a trei funcţii complexe divizat, aceste probleme de aproximare a unor funcţii sintetice fiind cele mai cunoscute benchmark-uri din literatura de specialitate în domeniu. Dintre aplicaţiile din lumea reală, am ales egalizarea semnalelor cu valori complexe liniare şi neliniare, predicţia seriilor de timp liniare şi neliniare şi predicţia direcţiei şi vitezei vântului, acestea constituind de asemenea cele mai folosite benchmark-uri pentru testarea reţelelor cu valori complexe, din categoria aplicaţiilor din lumea reală.

- Secţiunea 6.2 a prezentat rezultatele aplicării metodei backpropagation deduse în Secţiunea 5.3 pentru **reţele neuronale cu valori matrici pătratice**. Aplicaţiile luate în considerare au fost trei probleme de aproximare a unor funcţii sintetice,

ale căror expresii au fost inspirate din funcţiile folosite pentru testarea reţelelor cu valori complexe.

- Secţiunea 6.3 a descris rezultatele obţinute în urma aplicării metodei backpropagation prezentate în Secţiunea 5.4 pentru **reţele neuronale cu valori matrici antisimetrice**. Datorită legăturii strânse a acestui tip de matrici cu geometria, pe lângă două probleme de aproximare a unor funcţii sintetice, similare celor din secţiunea anterioară, am testat aceste reţele pentru transformări geometrice: translaţia, scalarea şi rotaţia.
- În fine, Secţiunea 6.4 a dat rezultatele aplicării algoritmului propus în Secţiunea 5.5 pentru **medierea matricilor ortogonale** de ordinul trei, folosind o funcţie de cost de tip $L^p$, care constituie o noutate în literatură şi pentru care nu se poate aplica cu succes niciun alt algoritm dintre cei deja existenţi.

# Capitolul 7

# Concluzii

## 7.1 Sumarul tezei

În acestă teză, am extins cei mai cunoscuţi algoritmi de învăţare din cadrul reţelelor neuronale de tip feedforward cu valori reale în domeniul **reţelelor neuronale de tip feedforward cu valori complexe**. Motivul pentru care am ales această temă este acela că am constatat, studiind literatura de specialitate, faptul că, cu excepţia metodei gradient, a metodei gradient cu moment, a algoritmului resilient backpropagation şi a algoritmului Levenberg-Marquardt, principalii algoritmi de antrenare a reţelelor cu valori reale nu au fost extinşi în domeniul complex.

Ţinând cont de această constatare, am prezentat deducerea completă a **metodelor gradient îmbunătăţite** (respectiv algoritmii quickprop, resilient backpropagation, delta-bar-delta şi SuperSAB), a **metodelor gradienţilor conjugaţi** (cu actualizări Hestenes-Stiefel, Polak-Ribiere, Fletcher-Reeves şi Dai-Yuan, şi respectiv cu reporniri Powell-Beale, precum şi două variaţiuni ale actualizărilor Hestenes-Stiefel şi Polak-Ribiere), a **metodei gradienţilor conjugaţi scalaţi**, a **metodei Newton** (cu deducerea completă a algoritmului de calcul al **hessianei** funcţiei de eroare, care are şi alte aplicaţii înafara acestui algoritm, şi a **produsului hessianei cu un vector**), a **metodelor quasi-Newton** (respectiv metoda actualizării de rang unu şi algoritmii Davidon-Fletcher-Powell, Broyden-Fletcher-Goldfarb-Shanno şi one step secant) şi a **metodei Levenberg-Marquardt**. Deducerea s-a făcut respectând toate particularităţile de calcul impuse de trecerea de la domeniul real la domeniul complex, avându-se în vedere obţinerea unui grad maxim de generalitate, astfel încât algoritmii să poată fi utilizaţi într-un context cât mai larg. A fost folosit un tip de calcul special creat pentru derivarea în planul complex, şi anume calculul Wirtinger, sau calculul $\mathbb{CR}$.

**Optsprezece** algoritmi, dintre care **cincisprezece** apar aici pentru prima dată, au fost implementaţi în reţele neuronale cu valori complexe, care au fost testate pe mai multe benchmark-uri specifice domeniului şi prezente extensiv în literatura de specialitate. Astfel, am prezentat rezultatele experimentale pentru aplicaţii sintetice, respectiv problema XOR, problema XOR extinsă, două probleme de aproximare a unor funcţii complexe complet şi trei probleme de aproximare a unor funcţii complexe divizat, şi pentru aplicaţii din lumea reală, respectiv egalizarea canalelor liniare şi neliniare, predicţia seriilor de timp liniare şi neliniare şi predicţia direcţiei şi vitezei vântului.

Rezultatele rulării de câte $50$ de ori a fiecărui algoritm au fost prezentate în tabele şi, pe o scară logaritmică, în figuri. Dinamica învăţării a fost arătată în figuri. Se poate constata că din clasa metodelor gradient îmbunătăţite, metoda resilient backpropagation şi delta-bar-delta au avut, în general, cele mai bune performanţe. Din clasa

metodelor gradienţilor conjugaţi, metodele cu actualizări Hestenes-Stiefel, cu actualizări Dai-Yuan, şi respectiv cu reporniri Powell-Beale, au dat cele mai bune rezultate pe majoritatea problemelor. Dintre metodele quasi-Newton, metoda BFGS, a avut, în general, cea mai bună performanţă.

Se poate constata de asemenea că toate clasele de algoritmi implementaţi au performanţe mai bune decât clasica metodă gradient. În general, metodele gradient îmbunătăţite sunt cu un ordin de mărime mai bune în termeni de eroare medie pătratică decât metoda gradient, apoi urmează metodele gradienţilor conjugaţi, iar în final, cu cel puţin un ordin de mărime mai bune decât acestea din urmă, sunt metodele quasi-Newton. Se mai poate constata însă în cazul anumitor probleme, tendinţa de overfitting în cazul metodelor quasi-Newton, ceea ce înseamnă că alegerea unuia sau altuia dintre algoritmi este foarte dependentă de problema de rezolvat, şi este dificil de estimat apriori care dintre metode va da rezultatele cele mai bune.

Ca o concluzie generală a prezentului studiu, putem afirma că este de dorit şi binevenită extinderea tuturor algoritmilor de optimizare a funcţiei de eroare pentru reţelele neuronale cu valori reale în domeniul complex, pentru că, în cazul anumitor probleme, ierarhiile dintre algoritmi în cazul real se inversează în cazul complex. Aşa se întâmplă, de exemplu, cu metoda gradienţilor conjugaţi cu actualizări Polak-Ribiere care în cazul real dă rezultate mai bune decât, spre exemplu, metoda gradienţilor conjugaţi cu actualizări Fletcher-Reeves, iar în cazul complex, această ultimă metodă poate fi şi cu un ordin de mărime în termeni de eroare medie pătratică mai bun decât algoritmul cu actualizări Polak-Ribiere.

Acest fapt ne încurajează ca pe viitor să considerăm extinderea şi a altor algoritmi de învăţare mai puţin cunoscuţi din domeniul real în domeniul complex, deoarece, cu progresul aplicaţiilor reţelelor neuronale cu valori complexe în lumea reală, este necesară o cât mai bună performanţă a acestui tip de reţele. Datorită diferenţelor fundamentale care există între domeniul real şi complex, tot ceea ce se cunoaşte în termeni de performanţă în domeniul real, este posibil să nu se mai aplice în domeniul complex. În plus, aceşti algoritmi pot fi combinaţi cu tehnici specifice domeniului complex, pentru a le îmbunătăţi şi mai mult performanţele.

Ca o generalizare a reţelelor neuronale cu valori complexe, am dedus aceiaşi algoritmi pentru **reţelele neuronale Clifford**. Datorită interesului din ultimii ani pentru acest tip de reţele, am considerat oportun să prezentăm aceleaşi metode de învăţare şi pentru acest tip de reţele, ţinând cont de faptul că doar metoda gradient a mai fost dedusă în altă parte în literatură, restul algoritmilor apărând aici pentru prima dată. Calculul diferenţial folosit este cel cu derivate parţiale reale, care se poate particulariza şi pentru reţelele neuronale cu valori complexe, dând formule diferite pentru algoritmi, care însă sunt echivalente cu cele deduse folosind calculul Wirtinger, sau calculul $\mathbb{CR}$.

Astfel, am dedus formule pentru **metodele gradient îmbunătăţite** (respectiv algoritmii quickprop, resilient backpropagation, delta-bar-delta şi SuperSAB), **metodele gradienţilor conjugaţi** (cu actualizări Hestenes-Stiefel, Polak-Ribiere, Fletcher-Reeves şi Dai-Yuan, şi respectiv cu reporniri Powell-Beale, şi cele două variaţiuni ale actualizărilor Hestenes-Stiefel şi Polak-Ribiere), **metoda gradienţilor conjugaţi scalaţi**, **metoda Newton** (cu deducerea completă a metodei de calcul al **hessianei** funcţiei de eroare şi a **produsului hessianei cu un vector**), **metodele quasi-Newton** (respectiv metoda actualizării de rang unu şi algoritmii Davidon-Fletcher-Powell, Broyden-Fletcher-Goldfarb-Shanno şi one step secant) şi pentru **algoritmul Levenberg-Marquardt**.

Motivul pentru care am decis să extindem aceiaşi algoritmi şi pentru cazul Clifford este acela că, adesea, nu se cunoaşte de la început care algebră Clifford este cea mai potrivită pentru a reprezenta datele dintr-o anumită aplicaţie. Astfel, am pus

la dispoziţia celor interesaţi o paletă largă de algoritmi, care pot fi aplicaţi pentru reţele neuronale cu valori în diferite algebre Clifford, pentru a determina experimental care este cea mai potrivită configuraţie arhitectură-algoritm de învăţare pentru fiecare problemă de interes. De exemplu, pentru cazul $2$-dimensional există $2$ algebre Clifford, pentru cazul $4$-dimensional există $3$ algebre Clifford, iar pentru cazul $8$-dimensional avem $4$ algebre Clifford cu care se poate experimenta.

O particularizare interesantă, cu multe aplicaţii în ultimul timp în literatura de specialitate, este aceea a **reţelelor neuronale cu valori cuaternionice**. Prin urmare, algoritmii mai sus propuşi se pot particulariza uşor pentru algebra Clifford a cuaternionilor, de dimensiune $4$. Pe viitor, implementări ale acestor algoritmi pentru reţelele neuronale cu valori cuaternionice promit să dea rezultate asemănătoare cu cele obţinute de aceiaşi algoritmi în cazul complex. Literatura de specialitate din acest domeniu abia în ultimii trei ani a început să ia naştere, iar viitorul va putea da suficienţi termeni de comparaţie pentru algoritmii propuşi.

În această teză, am introdus de asemenea un algoritm ce permite calculul **mediei pe varietăţi diferenţiabile** fără a utiliza coordonate locale şi derivate covariante pe varietatea respectivă. Acest algoritm, numit **algoritm de încorporare** (embedding algorithm) presupune liftarea funcţiei de cost iniţiale la o varietate diferenţiabilă care poate fi scufundată într-o varietate a riemanniană (spaţiu euclidian) şi construirea unui câmp de vectori definit pe spaţiul ambient a cărui restricţie la varietatea scufundată este câmpul de vectori gradient vector al funcţiei de cost liftate.

Astfel, pot fi calculate medii date de orice funcţie de cost, indiferent de cât de complicată este expresia ei, pentru că algoritmul presupune doar calcule în coordonate carteziene. Au fost prezentate rezultate experimentale folosind funcţii de tip $L^p$, ceea ce constituie o noutate în literatură, şi în particular a fost făcută comparaţia între cazul $L^2$ şi cazul $L^4$, evidenţiind principalele deosebiri în ceea ce priveşte medierea folosind aceste două tipuri de funcţii de cost.

O altă contribuţie a prezentei teze este introducerea, în premieră după cunoştinţele noastre, a unor generalizări ale reţelelor neuronale Clifford, şi anume a **reţelelor neuronale cu valori matriciale**. În aceste reţele neuronale intrările, ieşirile, ponderile şi bias-urile sunt matrici. Am considerat două cazuri, şi anume **reţelele neuronale cu valori matrici pătratice**, care sunt definite pe algebra matricilor cu operaţiile naturale de adunare şi înmulţire, şi respectiv **reţelele neuronale cu valori matrici antisimetrice**, adică reţele neuronale pe algebra asociată grupului de rotaţii.

În ambele cazuri, este introdusă metoda gradient de antrenare a reţelelor neuronale de tip feedforward. Ideea acestor reţele vine de la faptul că algoritmul propus pentru medierea matricilor ortogonale poate fi folosit pentru a genera seturi de date de antrenare pentru reţele neuronale, care pot astfel învăţa medii de orice fel, permiţându-ne să găsim media unor rotaţii fără a mai trece măcar prin calculele presupuse de algoritmul propus. Având valorile din grup, ele pot fi trecute în algebra asociată, date reţelei neuronale, iar rezultatele trecute înapoi în grup.

Rezultatele experimentale, pe trei probleme de aproximare a unor funcţii sintetice în cazul reţelelor neuronale cu valori matrici pătratice, respectiv pe două probleme de aproximare a unor funcţii sintetice şi pe transformări geometrice (translaţia, scalarea şi rotaţia) în cazul reţelelor neuronale cu valori matrici antisimetrice sunt promiţătoare pentru viitorul acestor tipuri de reţele, care pot fi folosite pentru a rezolva probleme $n$-dimensionale pe care alţi algoritmi cu valori reale şi/sau Clifford nu au putut să le înveţe, sau au avut performanţe slabe.

## 7.2 Articole publicate sau comunicate

Rezultatele obţinute în teză au fost valorificate în următoarele articole, dintre care 7 au fost publicate, iar unul este comunicat:

1. P. Birtea, D. Comănescu, and C.-A. Popa. Averaging on manifolds by embedding algorithm. *Journal of Mathematical Imaging and Vision*, 49(2):454 - 466, June 2014. (ISI Impact factor 2,33) [41]

2. C.-A. Popa. Enhanced Gradient Descent Algorithms for Complex-Valued Neural Networks. *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. pages 272 - 279. IEEE, September 2014. (IEEE, ISI Proceedings) [215]

3. C.-A. Popa. Exact Hessian Matrix Calculation for Complex-Valued Neural Networks. *International Workshop on Soft Computing Applications (SOFA)*. July 2014. (SpringerLink, ISI Proceedings) [216]

4. C.-A. Popa. Scaled Conjugate Gradient Learning for Complex-Valued Neural Networks. *International Conference on Soft Computing (MENDEL)*. June 2015. (SpringerLink, ISI Proceedings) [221]

5. C.-A. Popa. Matrix-Valued Neural Networks. *International Conference on Soft Computing (MENDEL)*. June 2015. (SpringerLink, ISI Proceedings) [219]

6. C.-A. Popa. Quasi-Newton Learning Methods for Complex-Valued Neural Networks. *International Joint Conference on Neural Networks (IJCNN)*. July 2015 (IEEE, ISI Proceedings) [220]

7. C.-A. Popa. Lie Algebra-Valued Neural Networks. *International Joint Conference on Neural Networks (IJCNN)*. July 2015 (IEEE, ISI Proceedings) [218]

8. C.-A. Popa. Conjugate Gradient Algorithms for Complex-Valued Neural Networks. *International Conference on Neural Information Processing (ICONIP)*. November 2015 (SpringerLink, ISI Proceedings) – comunicat [217]

Primul articol, publicat în jurnalul *Journal of Mathematical Imaging and Vision*, prezintă algoritmul de încorporare şi este scris împreună cu doi matematicieni, P. Birtea şi D. Comănescu, autorii apărând în ordine alfabetică, după cutuma existentă în cadrul comunităţii matematice. Având un factor mare de impact, jurnalul este unul dintre cele mai importante din domeniul inteligenţei artificiale, şi mai ales din domeniul imaging-ului, în care se aplică în particular algoritmul de încorporare pentru medierea matricilor ortogonale.

Pentru restul articolelor, autorul acestei teze este unic autor. Astfel, articolele 2 şi 3 au apărut la conferinţele *SYNASC 2014* şi *SOFA 2014*, ambele ţinute la Timişoara, şi valorifică doi algoritmi pentru reţele neuronale cu valori complexe. Cele două conferinţe sunt specifice domeniului inteligenţei artificiale, iar a doua chiar soft computing-ului, din care face parte şi domeniul reţelelor neuronale.

Articolele 4 şi 5 au apărut la conferinţa *MENDEL 2015*, ţinută la Brno, Cehia, de asemenea o conferinţă specifică domeniului soft computing, cu oarecare prestigiu internaţional. De remarcat faptul că articolul 5 este cel care introduce, în premieră pentru literatura de specialitate, reţelele neuronale cu valori matrici pătratice, pe când articolul 4 prezintă un alt algoritm pentru reţelele neuronale cu valori complexe.

Însă poate cele mai importante articole, care dau cea mai mare legitimitate tezei, sunt articolele 6 şi 7, prezentate la conferinţa *IJCNN 2015*, ţinută la Killarney,

Irlanda. Aceasta este cea mai importantă conferinţă în domeniul reţelelor neuronale, fiind sponsorizată de către International Neural Networks Society, şi, în acest an, şi de către European Neural Networks Society, pentru că ICANN, conferinţa acestei organizaţii, nu s-a ţinut în 2015, tocmai pentru a face loc conferinţei IJCNN, care are loc în Europa. După cum am menţionat în introducere, IJCNN a avut în ultimii ani o secţiune specială dedicată reţelelor neuronale cu valori complexe, şi are în Program Committee unii dintre cei mai importanţi cercetători în acest domeniu. Câtă vreme articolul 6 prezintă un algoritm pentru antrenarea reţelelor cu valori complexe, articolul 7 introduce, tot în premieră pentru literatura de specialitate, reţelele neuronale cu valori matrici antisimetrice.

Articolul 8 este dedicat unui alt algoritm pentru reţele neuronale cu valori complexe, fiind comunicat la *ICONIP 2015*.

În consecinţă, se poate spune că cercetarea făcută în această teză a fost validată de experţi din domeniul imaging-ului, al inteligenţei artificiale, al soft computing-ului, şi mai ales din domeniul reţelelor neuronale cu valori complexe, fapt care îi oferă recunoaştere academică şi vizibilitate internaţională.

## 7.3 Continuări posibile

O primă continuare posibilă, care este deja în lucru, este de a crea un **cadru** asemănător celui creat pentru reţelele neuronale cu valori complexe, şi **pentru reţelele neuronale cu valori cuaternionice**. Aceasta nu presupune un grad sporit de dificultate, fiind o simplă extindere de la $2$ la $4$ dimensiuni a cadrului deja existent, care a fost creat după modelul celui din Neural Networks Toolbox, din MATLAB. Diferenţele sunt date de faptul că algebra cuaternionilor nu este comutativă, deci se pot concepe mai multe modele de reţele cu valori cuaternionice, toate la fel de valide din punct de vedere matematic.

Apoi, se poate extinde şi mai mult acest **cadru**, prin crearea unuia şi mai general, care să lucreze cu algoritmi de învăţare pentru **reţele Clifford de orice dimensiune**. Practic, s-ar dori crearea unui suport care să poată fi folosit în mod flexibil pentru oricare dintre algebrele Clifford de dimensiune $2^n$, $n \geq 1$, folosind, probabil, programarea orientată pe obiecte care există şi în MATLAB. Un astfel de cadru ar putea fi util tuturor celor care au nevoie să rezolve probleme în care nu este cunoscută sau determinată apriori forma în care trebuie manipulate datele, putând fi alese diferite algebre, cu care se poate experimenta pentru a se găsi varianta optimă.

**Toţi algoritmii deduşi** pentru reţele de tip feedforward cu valori complexe sau Clifford **pot fi folosiţi şi pentru alte tipuri de reţele**, de exemplu pentru reţele de tip funcţii radiale (RBF) sau pentru reţele recurente. După cunoştinţa noastră, în literatură această problemă nu este încă tratată suficient, chiar şi pentru reţelele cu valori reale, nemaivorbind de cele cu valori complexe sau Clifford. Prin urmare, şi în această direcţie există nevoie de experimentare, pentru a se găsi algoritmi din ce în ce mai eficienţi, care în tandem cu cei folosiţi în mod obişnuit pentru aceste reţele, să dea rezultate cât mai bune.

Ca o primă aplicaţie din viaţa reală a reţelelor cu valori matrici antisimetrice, în care poate fi folosit algoritmul de încorporare propus, este la **folosirea reţelelor cu valori matrici antisimetrice pentru a învăţa medii**. Problema medierii matricilor este una foarte importantă şi de interes în aplicaţii practice, după cum am arătat mai sus. O reţea neuronală care să poată fi antrenată pentru a învăţa medii ar scurta cu mult timpul de calcul pe care îl presupune orice algoritm analitic existent în prezent.

O direcţie foarte importantă poate să fie **dezvoltarea în continuare a teoriei reţelelor neuronale cu valori matrici pătratice** şi **cu valori matrici antisime-**

**trice**. În prezenta lucrare am dedus doar algoritmul backpropagation pentru aceste reţele, însă, după modelul teoriei reţelelor cu valori cuaternionice care este în plină dezvoltare, vom încerca să extindem cele mai importante arhitecturi şi algoritmi de învăţare din cazurile real, complex şi cuaternionic, pentru reţelele Clifford în primul rând, iar mai apoi pentru reţelele cu valori matrici pătratice şi cu valori matrici antisimetrice. Datorită faptului că imaginile se pot prezenta ca matrici de pixeli, o posibilă direcţie interesantă ar fi aceea a reţelelor recurente şi a memoriilor asociative cu valori matriciale.

Din cele enumerate mai sus putem conchide că prezenta teză deschide mai multe direcţii de cercetare pe viitor, care pot da rezultate şi pot avea aplicaţii interesante, dacă vor fi explorate, prezentul demers reprezentând doar un prim pas făcut în direcţia unei **teorii generale a reţelelor neuronale cu valori în algebre multidimensionale**. S-ar putea, după cum s-a şi dovedit în anumite aplicaţii, ca nu doar mărirea numărului de straturi şi de neuroni pe fiecare strat să conducă la performanţe superioare, ci şi mărirea dimensiunii spaţiului căruia îi aparţin intrările, ieşirile, ponderile şi bias-urile reţelei, deoarece anumite date se organizează mai natural în alte algebre decât algebra numerelor reale.

# Bibliografie


[1] P.-A. Absil, R. Mahony, and R. Sepulchre. Riemannian geometry of grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematica*, 80(2):199 – 220, January 2004. doi:10.1023/B:ACAP.0000013855.14971.91.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton and Oxford, 2008. URL: http://press.princeton.edu/chapters/absil/.

[3] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135 – 158, January 2012. doi:10.1137/100802529.

[4] I. Aizenberg. *Complex-Valued Neural Networks with Multi-Valued Neurons*, volume 353 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-20353-4.

[5] I. Aizenberg and C. Butakoff. Image processing using cellular neural networks based on multi-valued and universal binary neurons. *Journal of VLSI signal processing systems for signal, image and video technology*, 32(1 - 2):169 – 188, August 2002. doi:10.1023/A:1016331805575.

[6] I. Aizenberg, E. Myasnikova, M. Samsonova, and J. Reinitz. Temporal classification of drosophila segmentation gene expression patterns by the multi-valued neural recognition method. *Mathematical Biosciences*, 176(1):145 – 159, March 2002. doi:10.1016/S0025-5564(01)00104-3.

[7] I. Aizenberg, D. Paliy, J.M. Zurada, and J.T. Astola. Blur identification by multilayer neural network based on multivalued neurons. *IEEE Transactions on Neural Networks*, 19(5):883 – 898, May 2008. doi:10.1109/TNN.2007.914158.

[8] I.N. Aizenberg, N.N. Aizenberg, and J.P. Vandewalle. *Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[9] N.N. Aizenberg, Y.L. Ivaskiv, and D.A. Pospelov. A certain generalization of threshould functions. *Dokrady Akademii Nauk SSSR*, 196:1287 – 1290, 1971.

[10] S.L. Altmann. *Rotations, Quaternions, and Double Groups*. Clarendon Press, Oxford, 1986. doi:10.1002/qua.560320310.

[11] M.F. Amin, M. Amin, A.Y.H. Al-Nuaimi, and K. Murase. Wirtinger calculus based gradient descent and levenberg-marquardt learning algorithms in complex-valued neural networks. In B.-L. Lu, L. Zhang, and J. Kwok, editors, *Neural Information Processing*, volume 7062 of *Lecture Notes in Computer Science*, pages 550 – 559. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-24955-6_66.

[12] M.F. Amin, M.M. Islam, and K. Murase. Single-layered complex-valued neural networks and their ensembles for real-valued classification problems. In *IEEE International Joint Conference on Neural Networks, IJCNN (IEEE World Congress on Computational Intelligence)*, pages 2500 – 2506. IEEE, June 2008.

doi:10.1109/IJCNN.2008.4634147.

[13] M.F. Amin, R. Savitha, M.I. Amin, and K. Murase. Complex-valued functional link network design by orthogonal least squares method for function approximation problems. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1489 – 1496. IEEE, July 2011. doi:10.1109/IJCNN.2011.6033400.

[14] J. Anemüller, T.J. Sejnowski, and S. Makeig. Complex independent component analysis of frequency-domain electroencephalographic data. *Neural Networks*, 16(9):1311 – 1323, November 2003. doi:10.1016/j.neunet.2003.08.003.

[15] H. Aoki and Y. Kosugi. Characteristics of the complex-valued associative memory model having penalty term. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 83(7):62 – 70, July 2000. doi:10.1002/(SICI)1520-6440(200007)83:7<62::AID-ECJC7>3.0.CO;2-L.

[16] P. Arena, S. Baglio, L. Fortuna, and M.G. Xibilia. Chaotic time series prediction via quaternionic multilayer perceptrons. In *International Conference on Systems, Man and Cybernetics*, volume 2, pages 1790 – 1794. IEEE, 1995. doi:10.1109/ICSMC.1995.538035.

[17] P. Arena, L. Fortuna, G. Muscato, and M.G. Xibilia. Multilayer perceptrons to approximate quaternion valued functions. *Neural Networks*, 10(2):335 – 342, March 1997. doi:10.1016/S0893-6080(96)00048-2.

[18] P. Arena, L. Fortuna, G. Muscato, and M.G. Xibilia. *Neural Networks in Multidimensional Domains Fundamentals and New Trends in Modelling and Control*, volume 234 of *Lecture Notes in Control and Information Sciences*. Springer London, 1998. doi:10.1007/BFb0047683.

[19] P. Arena, L. Fortuna, L. Occhipinti, and M.G. Xibilia. Neural networks for quaternion-valued function approximation. In *International Symposium on Circuits and Systems (ISCAS)*, volume 6, pages 307 – 310. IEEE, 1994. doi:10.1109/ISCAS.1994.409587.

[20] P. Arena, L. Fortuna, R. Re, and M.G. Xibilia. On the capability of neural networks with complex neurons in complex valued functions approximation. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2168 – 2171. IEEE, May 1993.

[21] P. Arena, L. Fortuna, R. Re, and M.G. Xibilia. Multilayer perceptrons to approximate complex valued functions. *International Journal of Neural Systems*, 6(4):435 – 446, December 1995. doi:10.1142/S0129065795000299.

[22] V. Arsigny, X. Pennec, and N. Ayache. Bi-invariant means in lie groups. application to left-invariant polyaffine transformations. Technical Report 5885, INRIA, 2006. URL: http://hal.inria.fr/inria-00071383/PS/RR-5885.ps.

[23] R. Battiti. First and second-order methods for learning between steepest descent and newton's method. *Neural Computation*, 4(2):141 – 166, March 1992. doi:10.1162/neco.1992.4.2.141.

[24] E. Bayro-Corrochano. Geometric neural computing. *IEEE Transactions on Neural Networks*, 12(5):968 – 986, September 2001. doi:10.1109/72.950128.

[25] E. Bayro-Corrochano, N. Arana, and R. Vallejo. Design of kernels for support multivector machines involving the clifford geometric product and the conformal geometric neuron. In *International Joint Conference on Neural Networks (IJCNN)*, volume 4, pages 2893 – 2898. IEEE, July 2003. doi:10.1109/IJCNN.2003.1224030.

[26] E. Bayro-Corrochano and N. Arana-Daniel. Theory and applications of clifford support vector machines. *Journal of Mathematical Imaging and Vision*, 28(1):29 – 46, May 2007. doi:10.1007/s10851-007-0008-7.

[27] E. Bayro-Corrochano, S. Buchholz, and G. Sommer. A new self-organizing ne-

ural network using geometric algebra. In *International Conference on Pattern Recognition*, 1996.

[28] E. Bayro-Corrochano, S. Buchholz, and G. Sommer. Selforganizing clifford neural network. In *International Conference on Neural Networks (ICNN)*, volume 1, pages 120 – 125. IEEE, June 1996. doi:10.1109/ICNN.1996.548877.

[29] E. Bayro-Corrochano and G. Sommer. *Algebraic Frames for the Perception-Action Cycle*, chapter Geometric Neural Networks, pages 379 – 394. Number 1315 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997. doi:10.1007/BFb0017879.

[30] E. Bayro-Corrochano and R. Vallejo. Geometric neural networks and support multi-vector machines. In *International Joint Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint*, volume 6, pages 389 – 394. IEEE, July 2000. doi:10.1109/IJCNN.2000.859426.

[31] E. Bayro-Corrochano and R. Vallejo. Svms using geometric algebra for 3d computer vision. In *International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 872 – 877. IEEE, July 2001. doi:10.1109/IJCNN.2001.939474.

[32] E. Bayro-Corrochano and R. Vallejo. Geometric neurocomputing for pattern recognition and pose estimation. In *International Conference on Pattern Recognition*, 2002.

[33] E. Bayro-Corrochano, J.R. Vallejo-Gutierrez, and N. Arana-Daniel. Recurrent clifford support machines. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3613 – 3618. IEEE, June 2008. doi:10.1109/IJCNN.2008.4634315.

[34] E.J. Bayro-Corrochano and N. Arana-Daniel. Clifford support vector machines for classification, regression, and recurrence. *IEEE Transactions on Neural Networks*, 21(11):1731 – 1746, November 2010. doi:10.1109/TNN.2010.2060352.

[35] E.M.L. Beale. A derivation of conjugate gradients. In F. A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 39–43. Academic Press, London, 1972.

[36] S. Becker and Y. LeCun. Improving the convergence of back-propagation learning with second-order methods. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. of the 1988 Connectionist Models Summer School*, pages 29 – 37, San Mateo, 1989. Morgan Kaufman.

[37] N. Benvenuto, M. Marchesi, F. Piazza, and A. Uncini. A comparison between real and complex valued neural networks in communication applications. In *International Conference on Neural Networks (ICANN)*, pages 1177 – 1180. Elsevier Science Publishers, June 1991. URL: http://www.uncini.com/research_activity/pdf/030_icann91.pdf.

[38] N. Benvenuto, M. Marchesi, F. Piazza, and A. Uncini. Complex neural networks for equalizing nonlinear digital radio links. In *Workshop CEE COST 229*, March 1991. URL: http://www.uncini.com/research_activity/pdf/033_cee_cost91.pdf.

[39] N. Benvenuto and F. Piazza. On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4):967 – 969, April 1992. doi:10.1109/78.127967.

[40] P. Birtea and D. Comanescu. Geometric dissipation for dynamical systems. *Communications in Mathematical Physics*, 316(2):375 – 394, December 2012. doi:10.1007/s00220-012-1589-6.

[41] P. Birtea, D. Comanescu, and C.-A. Popa. Averaging on manifolds by embedding algorithm. *Journal of Mathematical Imaging and Vision*, 49(2):454 – 466, June 2014. doi:10.1007/s10851-013-0478-8.

[42] D.L. Birx and S.J. Pipenberg. Chaotic oscillators and complex mapping feed forward networks (cmffns) for signal detection in noisy environments. In *Inter-*

*national Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 881 – 888. IEEE, June 1992. doi:10.1109/IJCNN.1992.226876.

[43] D.L. Birx and S.J. Pipenberg. A complex mapping network for phase sensitive classification. *IEEE Transactions on Neural Networks*, 4(1):127 – 135, January 1993. doi:10.1109/72.182703.

[44] C.M. Bishop. A fast procedure for retraining the multilayer perceptron. *International Journal of Neural Systems*, 2(3):229 – 236, 1991. doi:10.1142/S0129065791000212.

[45] C.M. Bishop. Exact calculation of the hessian matrix for the multi-layer perceptron. *Neural Computation*, 4(4):494 – 501, July 1992. doi:10.1162/neco.1992.4.4.494.

[46] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[47] R.P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[48] S. Buchholz. *A Theory of Neural Computation with Clifford Algebras*. PhD thesis, University of Kiel, 2005. URL: http://www.informatik.uni-kiel.de/inf/Sommer/doc/Dissertationen/Sven_Buchholz/diss.pdf.

[49] S. Buchholz, E.M.S. Hitzer, and K. Tachibana. Coordinate independent update formulas for versor clifford neurons. In *International Conference on Soft Computing and Intelligent Systems (SCIS) International Symposium on Advanced Intelligent Systems (ISIS)*, pages 814 – 819, September 2008.

[50] S. Buchholz and N. Le Bihan. Polarized signal classification by complex and quaternionic multi-layer perceptrons. *International Journal of Neural Systems*, 18(2):75 – 85, April 2008. doi:10.1142/S0129065708001403.

[51] S. Buchholz and G. Sommer. *Algebraic Frames for the Perception-Action Cycle*, chapter Learning Geometric Transformations with Clifford Neurons, pages 144 – 153. Number 1888 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000. doi:10.1007/10722492_9.

[52] S. Buchholz and G. Sommer. A hyperbolic multilayer perceptron. In *International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 129 – 133. IEEE, July 2000. doi:10.1109/IJCNN.2000.857886.

[53] S. Buchholz and G. Sommer. Quaternionic spinor mlp. In *European Symposium on Artificial Neural Networks*, pages 377 – 382, April 2000. URL: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2000-28.pdf.

[54] S. Buchholz and G. Sommer. On clifford neurons and clifford multi-layer perceptrons. *Neural Networks*, 21(7):925 – 935, 2008. doi:10.1016/j.neunet.2008.03.004.

[55] S. Buchholz, K. Tachibana, and E.M.S. Hitzer. *Artificial Neural Networks – ICANN 2007*, chapter Optimal Learning Rates for Clifford Neurons, pages 864 – 873. Number 4668 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-74690-4_88.

[56] S. Buchholz, K. Tachibana, and E.M.S. Hitzer. Introduction to theory, construction and application of clifford neural networks. In *International Conference on Neural Information Processing (ICONIP)*, pages 1 – 14, November 2007.

[57] K. Burse, A. Pandey, and A. Somkuwar. Convergence analysis of complex valued multiplicative neural network for various activation functions. In *International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 279 – 282. IEEE, October 2011. doi:10.1109/CICN.2011.57.

[58] T. Burwick. Oscillatory networks: Pattern recognition without a superposition catastrophe. *Neural Computation*, 18(2):356 – 380, February 2006. doi:10.

1162/089976606775093864.

[59] T. Burwick. Oscillatory neural networks with self-organized segmentation of overlapping patterns. *Neural Computation*, 19(8):2093 – 2123, August 2007. doi:10.1162/neco.2007.19.8.2093.

[60] D. Casasent and S. Natarajan. A classifier neural net with complex-valued weights and square-law nonlinearities. *Neural Networks*, 8(6):989 – 998, 1995. doi:10.1016/0893-6080(95)00008-N.

[61] M. Ceylan. Combined complex-valued artificial neural network (ccvann). In *Proceedings of the World Congress on Engineering*, volume 2, pages 955 – 959, July 2011. URL: http://www.iaeng.org/publication/WCE2011/WCE2011_pp955-959.pdf.

[62] I. Cha and S.A. Kassam. Channel equalization using adaptive complex radial basis function networks. *IEEE Journal on Selected Areas in Communications*, 13(1):122 – 131, January 1995. doi:10.1109/49.363139.

[63] S.V. Chakravarthy and J. Ghosh. A complex-valued associative memory for storing patterns as oscillatory states. *Biological Cybernetics*, 75(3):229 – 238, September 1996. doi:10.1007/s004220050290.

[64] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEE Proceedings G Circuits, Devices and Systems*, 139(3):301 – 310, June 1992.

[65] B. Che Ujang, C.C. Took, and D.P. Mandic. Split quaternion nonlinear adaptive filtering. *Neural Networks*, 23(3):426 – 434, April 2010. doi:10.1016/j.neunet.2009.10.006.

[66] B. Che Ujang, C.C. Took, and D.P. Mandic. Quaternion-valued nonlinear adaptive filtering. *IEEE Transactions on Neural Networks*, 22(8):1193 – 1206, August 2011. doi:10.1109/TNN.2011.2157358.

[67] B. Che Ujang, C.C. Took, and D.P. Mandic. On quaternion analyticity: Enabling quaternion-valued nonlinear adaptive filtering. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2117 – 2120. IEEE, March 2012. doi:10.1109/ICASSP.2012.6288329.

[68] S. Chen, L. Hanzo, and S. Tan. Symmetric complex-valued rbf receiver for multiple-antenna-aided wireless systems. *IEEE Transactions on Neural Networks*, 19(9):1659 – 1665, September 2008. doi:10.1109/TNN.2008.2000582.

[69] S. Chen, S. McLaughlin, and B. Mulgrew. Complex-valued radial basis function network, part ii: Application to digital communications channel equalisation. *Signal Processing*, 36(2):175 – 188, 1994. doi:10.1016/0165-1684(94)90206-2.

[70] X. Chen, Z. Tang, C. Variappan, S. Li, and T. Okada. A modified error backpropagation algorithm for complex-value neural networks. *International Journal of Neural Systems*, 15(6):435 – 443, December 2005. doi:10.1142/S0129065705000426.

[71] T.L. Clarke. Generalization of neural networks to the complex plane. In *International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 435 – 440. IEEE, June 1990. doi:10.1109/IJCNN.1990.137751.

[72] Y. Cui, K. Takahashi, and M. Hashimoto. Remarks on robot controller application of clifford multi-layer neural networks. In *International Workshop on Advanced Motion Control (AMC)*, pages 410 – 415. IEEE, 2014. doi:10.1109/AMC.2014.6823317.

[73] Y.H. Dai and Y. Yuan. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on Optimization*, 10(1):177 – 182, May 1999. doi:10.1137/S1052623497318992.

[74] E.A. de Kerf, G.G.A. Bauerle, and A.P.E. ten Kroode. *Lie Algebras: Finite and*

*Infinite Dimensional Lie Algebras and Applications in Physics*. North Holland, 1997.

[75] Y. Deng and Z. Yang. Comments on "complex-bilinear recurrent neural network for equalization of a digital satellite channel". *IEEE Transactions on Neural Networks*, 17(1):268 – 268, January 2006. doi:10.1109/TNN.2005.860863.

[76] J.E. Dennis and R.B. Schnabel. A new derivation of symmetric positive definite secant updates. Technical Report CU-CS-185-80, University of Colorado at Boulder, Department of Computer Science, August 1980.

[77] S.S. Dragomir and D. Comanescu. On the torricellian point in inner product spaces. *Demonstratio Mathematica*, XLI(3):639 – 650, 2008. URL: http://www.mini.pw.edu.pl/~demmath/archive/dm41_3/15.pdf.

[78] S.S. Dragomir, D. Comanescu, and E. Kikianty. Torricellian points in normed linear spaces. *Journal of Inequalities and Applications*, 2013(258):1 – 15, May 2013. doi:10.1186/1029-242X-2013-258.

[79] A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303 – 353, 1998. doi:10.1137/S0895479895290954.

[80] S.E. Fahlman. An empirical study of learning speed in backpropagation networks. Technical Report 1800, Carnegie Mellon University, January 1988. URL: http://repository.cmu.edu/compsci/1800.

[81] S. Fiori. Nonlinear complex-valued extensions of hebbian learning: An essay. *Neural Computation*, 17(4):779 – 838, April 2005. doi:10.1162/0899766053429381.

[82] S. Fiori. Lie-group-type neural system learning by manifold retractions. *Neural Networks*, 21(10):1524 – 1529, December 2008. doi:10.1016/j.neunet.2008.09.009.

[83] S. Fiori. Solving minimal-distance problems over the manifold of real-symplectic matrices. *SIAM Journal on Matrix Analysis and Applications*, 32(3):938 – 968, September 2011. doi:10.1137/100817115.

[84] S. Fiori and T. Tanaka. An algorithm to compute averages on matrix lie groups. *IEEE Transactions on Signal Processing*, 57(12):4734 – 4743, December 2009. doi:10.1109/TSP.2009.2027754.

[85] R. Fletcher and M.J.D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163 – 168, August 1963. doi:10.1093/comjnl/6.2.163.

[86] Q. Gan, P. Saratchandran, N. Sundararajan, and K.R. Subramanian. A complex valued radial basis function network for equalization of fast time varying channels. *IEEE Transactions on Neural Networks*, 10(4):958 – 960, July 1999. doi:10.1109/72.774271.

[87] A.S. Gangal, P.K. Kalra, and D.S. Chauhan. Performance evaluation of complex valued neural networks using various error functions. *International Journal of Electrical, Robotics, Electronics and Communications Engineering*, 1(5):728 – 733, 2007. URL: http://waset.org/publications/11670.

[88] A.S. Gangal, P.K. Kalra, and D.S. Chauhan. Inversion of complex valued neural networks using complex back-propagation algorithm. *International Journal of Mathematics and Computers in Simulation*, 3(1):1 – 8, 2009.

[89] G.M. Georgiou. The multivalued and continuous perceptrons. In *World Congress on Neural Networks*, volume 4, pages 679 – 683. Lawrence Erlbaum Associates, July 1993.

[90] G.M. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*,

39(5):330 – 334, May 1992. doi:10.1109/82.142037.

[91] J.C. Gilbert and J. Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, 2(1):21 – 42, 1992. doi:10.1137/0802003.

[92] S.L. Goh and D.P. Mandic. A class of low complexity and fast converging algorithms for complex-valued neural networks. In *IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 13 – 22. IEEE, September 2004. doi:10.1109/MLSP.2004.1422955.

[93] S.L. Goh and D.P. Mandic. A complex-valued rtrl algorithm for recurrent neural networks. *Neural Computation*, 16(12):2699 – 2713, December 2004. doi:10.1162/0899766042321779.

[94] S.L. Goh and D.P. Mandic. A class of gradient-adaptive step size algorithms for complex-valued nonlinear neural adaptive filters. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 253 – 256. IEEE, March 2005. doi:10.1109/ICASSP.2005.1416288.

[95] S.L. Goh and D.P. Mandic. Nonlinear adaptive prediction of complex-valued signals by complex-valued prnn. *IEEE Transactions on Signal Processing*, 53(5):1827 – 1836, May 2005. doi:10.1109/TSP.2005.845462.

[96] S.L. Goh and D.P. Mandic. An augmented crtrl for complex-valued recurrent neural networks. *Neural Networks*, 20(10):1061 – 1066, December 2007. doi:10.1016/j.neunet.2007.09.015.

[97] S.L. Goh and D.P. Mandic. An augmented extended kalman filter algorithm for complex-valued recurrent neural networks. *Neural Computation*, 19(4):1039 – 1055, April 2007. doi:10.1162/neco.2007.19.4.1039.

[98] S.L. Goh and D.P. Mandic. Stochastic gradient-adaptive complex-valued nonlinear neural adaptive filters with a gradient-adaptive step size. *IEEE Transactions on Neural Networks*, 18(5):1511 – 1516, September 2007. doi:10.1109/TNN.2007.895828.

[99] S.L. Goh, D.H. Popovic, and D.P. Mandic. Complex-valued estimation of wind profile and wind power. In *IEEE Mediterranean Electrotechnical Conference (MELECON)*, volume 3, pages 1037 – 1040. IEEE, May 2004. doi:10.1109/MELCON.2004.1348231.

[100] V.M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 684 – 691, June 2004. doi:10.1109/CVPR.2004.1315098.

[101] C. Gramkow. On averaging rotations. *Journal of Mathematical Imaging and Vision*, 15(1-2):7 – 16, July 2001. doi:10.1023/A:1011217513455.

[102] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989 – 993, November 1994. doi:10.1109/72.329697.

[103] W.W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2(1):35 – 58, 2006. URL: http://people.cs.vt.edu/~asandu/Public/Qual2011/Optim/Hager_2006_CG-survey.pdf.

[104] A.I. Hanna and D.P. Mandic. A complex-valued nonlinear neural adaptive filter with a gradient adaptive amplitude of the activation function. *Neural Networks*, 16(2):155 – 159, March 2003. doi:10.1016/S0893-6080(02)00236-8.

[105] A.I. Hanna and D.P. Mandic. A data-reusing nonlinear gradient descent algorithm for a class of complex-valued neural adaptive filters. *Neural Processing Letters*, 17(1):85 – 91, 2003. doi:10.1023/A:1022915613633.

[106] A.I. Hanna and D.P. Mandic. A fully adaptive normalized nonlinear gradient descent algorithm for complex-valued nonlinear adaptive filters. *IEEE Tran-*

*sactions on Signal Processing*, 51(10):2540 – 2549, October 2003. doi:10.1109/TSP.2003.816878.

[107] T. Hara and A. Hirose. Plastic mine detecting radar system using complex-valued self-organizing map that deals with multiple-frequency interferometric images. *Neural Networks*, 17(8 - 9):1201 – 1210, October - November 2004. doi:10.1016/j.neunet.2004.07.012.

[108] R. Hartley, J. Trumpf, and Y. Dai. Rotation averaging and weak convexity. In *International Symposium on Mathematical Theory of Networks and Systems*, pages 2435 – 2442, 2010.

[109] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International Journal of Computer Vision*, 103(3):267 – 305, July 2013. doi:10.1007/s11263-012-0601-0.

[110] R. Hata and K. Murase. Quaternion neuro-fuzzy for real-valued classification problems. In *Joint International Conference on Soft Computing and Intelligent Systems (SCIS), and Advanced Intelligent Systems (ISIS), 15th International Symposium on*, pages 655 – 660. IEEE, December 2014. doi:10.1109/SCIS-ISIS.2014.7044665.

[111] U. Helmke and M.A. Shayman. Critical points of matrix least squares distance functions. *Linear Algebra and its Applications*, 215:1 – 19, January 1995. doi:10.1016/0024-3795(93)00070-G.

[112] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409 – 436, December 1952.

[113] A. Hirose. Continuous complex-valued back-propagation learning. *Electronics Letters*, 28(20):1854 – 1855, September 1992. doi:10.1049/el:19921186.

[114] A. Hirose. Dynamics of fully complex-valued neural networks. *Electronics Letters*, 28(16):1492 – 1494, July 1992. doi:10.1049/el:19920948.

[115] A. Hirose. Applications of complex-valued neural networks to coherent optical computing using phase-sensitive detection scheme. *Information Sciences - Applications*, 2(2):103 – 117, September 1994. doi:10.1016/1069-0115(94)90014-0.

[116] A. Hirose. Fractal variation of attractors in complex-valued neural networks. *Neural Processing Letters*, 1(1):6 – 8, 1994. doi:10.1007/BF02312393.

[117] A. Hirose, editor. *Complex-Valued Neural Networks: Theories and Applications*, volume 5 of *Series on Innovative Intelligence*. World Scientific, 2003. doi:10.1142/5345.

[118] A. Hirose. *Complex-Valued Neural Networks*, volume 400 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-27632-3.

[119] A. Hirose. *Complex-Valued Neural Networks: Advances and Applications*. John Wiley & Sons, Inc., 2013. doi:10.1002/9781118590072.

[120] A. Hirose and R. Eckmiller. Behavior control of coherent-type neural networks by carrier-frequency modulation. *IEEE Transactions on Neural Networks*, 7(4):1032 – 1034, July 1996. doi:10.1109/72.508945.

[121] A. Hirose and R. Eckmiller. Coherent optical neural networks and the generalization characteristics. *Optical Review*, 3(6):A418–A422, Nov./Dec. 1996. doi:10.1007/BF02935948.

[122] A. Hirose and R. Eckmiller. Coherent optical neural networks that have optical-frequency-controlled behavior and generalization ability in the frequency domain. *Applied Optics*, 35(5):836 – 843, 1996. doi:10.1364/AO.35.000836.

[123] A. Hirose and R. Eckmiller. Proposal of frequency-domain multiplexing in optical

neural networks. *Neurocomputing*, 10(2):197 – 204, March 1996. doi:10.1016/0925-2312(95)00129-8.

[124] A. Hirose and H. Onishi. Observation of positive lyapunov exponent induced by gain increase of neuron nonlinearity in complex-valued associative memories. *Electronics Letters*, 32(8):745 – 746, April 1996. doi:10.1049/el:19960472.

[125] A. Hirose and H. Onishi. Proposal of relative-minimization learning for behavior stabilization of complex-valued recurrent neural networks. *Neurocomputing*, 24(1 - 3):163 – 171, February 1999. doi:10.1016/S0925-2312(98)00093-9.

[126] E. Hitzer. Non-constant bounded holomorphic functions of hyperbolic numbers - candidates for hyperbolic activation functions, 2013. URL: http://arxiv.org/abs/1306.1653.

[127] G. Howells, D.L. Bisset, and M.C. Fairhurst. Clifford networks novel neural architectures for multidimensional data. In *International Conference on Image Processing and Its Applications*, volume 2, pages 670 – 675. IEEE, July 1997. doi:10.1049/cp:19970979.

[128] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew. Incremental extreme learning machine with fully complex hidden nodes. *Neurocomputing*, 71(4 - 6):576 – 583, January 2008. doi:10.1016/j.neucom.2007.07.025.

[129] D.Q. Huynh. Metrics for 3d rotations comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155 – 164, October 2009. doi:10.1007/s10851-009-0161-2.

[130] F. Iachello. *Lie Algebras and Applications*, volume 891 of *Lecture Notes in Physics*. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-44494-8.

[131] T. Isokawa, T. Kusakabe, N. Matsui, and F. Peper. Quaternion neural network and its application. In V. Palade, R.J. Howlett, and Jai, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 2774 of *Lecture Notes in Computer Science*, pages 318 – 324, 2003. doi:10.1007/978-3-540-45226-3_44.

[132] T. Isokawa, H. Nishimura, N. Kamiura, and N. Matsui. Associative memory in quaternionic hopfield neural network. *International Journal of Neural Systems*, 18(2):135 – 145, April 2008. doi:10.1142/S0129065708001440.

[133] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295 – 307, 1988. doi:10.1016/0893-6080(88)90003-2.

[134] C. Jahanchahi and D.P. Mandic. A class of quaternion kalman filters. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3):533 – 544, March 2014. doi:10.1109/TNNLS.2013.2277540.

[135] C. Jahanchahi, C.C. Took, and D.P. Mandic. On hr calculus, quaternion valued stochastic gradient, and adaptive three dimensional wind forecasting. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 5. IEEE, July 2010. doi:10.1109/IJCNN.2010.5596629.

[136] C. Jahanchahi, C.C. Took, and D.P. Mandic. The widely linear quaternion recursive least squares filter. In *International Workshop on Cognitive Information Processing (CIP)*, pages 87 – 92. IEEE, June 2010. doi:10.1109/CIP.2010.5604211.

[137] S. Jankowski, A. Lozowski, and J.M. Zurada. Complex-valued multistate neural associative memory. *IEEE Transactions on Neural Networks*, 7(6):1491 – 1496, November 1996. doi:10.1109/72.548176.

[138] G.L. Jenkins and M.E. Dacey. A unit quaternion based som for anatomical joint constraint modelling. In *International Conference on Computer Modelling and Simulation (UKSim)*, pages 89 – 95. IEEE, March 2014. doi:10.1109/UKSim.2014.21.

[139] D. Jianping, N. Sundararajan, and P. Saratchandran. Communication channel equalization using complex-valued minimal radial basis function neural networks. *IEEE Transactions on Neural Networks*, 13(3):687 – 696, May 2002. doi:10.1109/TNN.2002.1000133.

[140] E.M. Johansson, F.U. Dowla, and D.M. Goodman. Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems*, 2(4):291 – 301, 1991. doi:10.1142/S0129065791000261.

[141] A. Kantsila, M. Lehtokangas, and J. Saarinen. Complex rprop-algorithm for neural network equalization of gsm data bursts. *Neurocomputing*, 61:339 – 360, October 2004. doi:10.1016/j.neucom.2003.11.007.

[142] S. Kawata and A. Hirose. Frequency-multiplexed logic circuit based on a coherent optical neural network. *Applied Optics*, 44(19):4053 – 4059, 2005. doi:10.1364/AO.44.004053.

[143] M.S. Kim and C.C. Guest. Modification of backpropagation networks for complex-valued signal processing in frequency domain. In *International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 27 – 31. IEEE, June 1990. doi:10.1109/IJCNN.1990.137820.

[144] T. Kim and T. Adali. Fully complex backpropagation for constant envelope signal processing. In *IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing*, volume 1, pages 231 – 240. IEEE, December 2000. doi:10.1109/NNSP.2000.889414.

[145] T. Kim and T. Adali. Approximation by fully complex mlp using elementary transcendental activation functions. In *IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing*, pages 203 – 212. IEEE, September 2001. doi:10.1109/NNSP.2001.943125.

[146] T. Kim and T. Adali. Fully complex multi-layer perceptron network for nonlinear signal processing. *Journal of VLSI signal processing systems for signal, image and video technology*, 32(1 - 2):29 – 43, August 2002. doi:10.1023/A:1016359216961.

[147] T. Kim and T. Adali. Universal approximation of fully complex feed-forward neural networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 973 – 976. IEEE, May 2002. doi:10.1109/ICASSP.2002.5743956.

[148] T. Kim and T. Adali. Approximation by fully complex multilayer perceptrons. *Neural Computation*, 15(7):1641 – 1666, July 2003. doi:10.1162/089976603321891846.

[149] M. Kinouchi and M. Hagiwara. Memorization of melodies by complex-valued recurrent network. In *IEEE International Conference on Neural Networks*, volume 2, pages 1324 – 1328. IEEE, June 1996. URL: 10.1109/ICNN.1996.549090.

[150] T. Kitajima and T. Yasuno. Output prediction of wind power generation system using complex-valued neural network. In *Proceedings of SICE Annual Conference*, pages 3610 – 3613. IEEE, August 2010.

[151] M. Kobayashi. Exceptional reducibility of complex-valued neural networks. *IEEE Transactions on Neural Networks*, 21(7):1060 – 1072, July 2010. doi:10.1109/TNN.2010.2048040.

[152] M. Kobayashi. Hyperbolic hopfield neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2):335 – 341, February 2013. doi:10.1109/TNNLS.2012.2230450.

[153] K. Kreutz-Delgado. The complex gradient operator and the cr-calculus. Technical Report UCSD-ECE275CG-S2009v1.0, University of California, San Diego,

June 2009.

[154] Y. Kuroe. Models of clifford recurrent neural networks and their dynamics. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1035 – 1041. IEEE, 2011. doi:10.1109/IJCNN.2011.6033336.

[155] Y. Kuroe, N. Hashimoto, and T. Mori. Qualitative analysis of a self-correlation type complex-valued associative memories. *Nonlinear Analysis: Theory, Methods & Applications*, 47(9):5795 – 5806, August 2001. doi:10.1016/S0362-546X(01)00823-9.

[156] Y. Kuroe, S. Tanigawa, and H. Iima. Models of hopfield-type clifford neural networks and their energy functions - hyperbolic and dual valued networks -. In *International Conference on Neural Information Processing*, number 7062 in Lecture Notes in Computer Science, pages 560 – 569, 2011. doi:10.1007/978-3-642-24955-6_67.

[157] H. Kusamichi, T. Isokawa, N. Matsui, Y. Ogawa, and K. Maeda. A new scheme for color night vision by quaternion neural network. In *International Conference on Autonomous Robots and Agents*, pages 101 – 106, December 2004. URL: http://www-ist.massey.ac.nz/conferences/icara2004/files/Papers/Paper17_ICARA2004_101_106.pdf.

[158] Y. LeCun, J.S. Denker, S. Solla, R.E. Howard, and L.D. Jackel. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems (NIPS 1989)*, volume 2, Denver, CO, 1990. Morgan Kaufman.

[159] D.L. Lee. Improving the capacity of complex-valued neural networks with a modified gradient descent learning rule. *IEEE Transactions on Neural Networks*, 12(2):439 – 443, March 2001. doi:10.1109/72.914540.

[160] D.L. Lee. Relaxation of the stability condition of the complex-valued neural networks. *IEEE Transactions on Neural Networks*, 12(5):1260 – 1262, September 2001. doi:10.1109/72.950156.

[161] D.L. Lee. Improvements of complex-valued hopfield associative memory by using generalized projection rules. *IEEE Transactions on Neural Networks*, 17(5):1341 – 1347, September 2006. doi:10.1109/TNN.2006.878786.

[162] D.L. Lee and W.J. Wang. A multivalued bidirectional associative memory operating on a complex domain. *Neural Networks*, 11(9):1623 – 1635, December 1998. doi:10.1016/S0893-6080(98)00078-1.

[163] H. Leung and S. Haykin. The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 39(9):2101 – 2104, September 1991. doi:10.1109/78.134446.

[164] H. Li and T. Adali. A class of complex ica algorithms based on the kurtosis cost function. *IEEE Transactions on Neural Networks*, 19(3):408 – 420, March 2008. doi:10.1109/TNN.2007.908636.

[165] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan. Fully complex extreme learning machine. *Neurocomputing*, 68:306 – 314, October 2005. doi:10.1016/j.neucom.2005.03.002.

[166] X.L. Li and T. Adali. Complex independent component analysis by entropy bound minimization. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(7):1417 – 1430, July 2010. doi:10.1109/TCSI.2010.2046207.

[167] D.G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*, volume 116 of *International Series in Operations Research & Management Science*. Springer, 2008. doi:10.1007/978-0-387-74503-9.

[168] D.J.C. Mackay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448 – 472, 1992. doi:10.1162/neco.1992.4.3.448.

[169] D.P. Mandic. Complex valued recurrent neural networks for noncircular complex

signals. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1987 – 1992. IEEE, June 2009. doi:10.1109/IJCNN.2009.5178960.

[170] D.P. Mandic and J. Chambers. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA, 2001. doi:10.1002/047084535X.

[171] D.P. Mandic and S.L. Goh. *Complex Valued Nonlinear Adaptive Filters Noncircularity, Widely Linear and Neural Models*. John Wiley & Sons, Inc., 2009. doi:10.1002/9780470742624.

[172] D.P. Mandic, S.L. Javidi, S .and Goh, A. Kuh, and K. Aihara. Complex-valued prediction of wind profile using augmented complex statistics. *Renewable Energy*, 34(1):196 – 201, January 2009. doi:doi:10.1016/j.renene.2008.03.022.

[173] F.L. Markley, Y. Cheng, J.L. Crassidis, and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193 – 1197, July 2007. doi:10.2514/1.28949.

[174] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431 – 441, June 1963. doi:10.1137/0111030.

[175] S. Masuyama and A. Hirose. Walled ltsa array for rapid, high spatial resolution, and phase-sensitive imaging to visualize plastic landmines. *IEEE Transactions on Geoscience and Remote Sensing*, 48(5):2536 – 2543, August 2007. doi:10.1109/TGRS.2007.897418.

[176] S. Masuyama, K. Yasuda, and A. Hirose. Multiple-mode selection of walled-ltsa array elements for high-resolution imaging to visualize antipersonnel plastic landmines. *IEEE Geoscience and Remote Sensing Letters*, 5(4):745 – 749, October 2008. doi:10.1109/LGRS.2008.2004509.

[177] M. Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1 – 16, July 2002. doi:10.1137/S0895479801383877.

[178] M.F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533, 1993. doi:10.1016/S0893-6080(05)80056-5.

[179] M.K. Muezzinoglu, C. Guzelis, and J.M. Zurada. A new design method for the complex-valued multistate hopfield associative memory. *IEEE Transactions on Neural Networks*, 14(4):891 – 899, July 2003. doi:10.1109/TNN.2003.813844.

[180] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4 – 27, March 1990. doi:10.1109/72.80202.

[181] I. Nemoto and M. Kubono. Complex associative memory. *Neural Networks*, 9(2):253 – 261, March 1996.

[182] I. Nemoto and K. Saito. A complex-valued version of nagumo-sato model of a single neuron and its behavior. *Neural Networks*, 15(7):833 – 853, September 2002. doi:10.1016/S0893-6080(02)00066-7.

[183] I. Nishikawa and Y. Kuroe. Dynamics of complex-valued neural networks and its relation to a phase oscillator system. In N.R. Pal, N. Kasabov, R.K. Mudi, S. Pal, and S.K. Parui, editors, *Neural Information Processing (ICONIP)*, volume 3316 of *Lecture Notes in Computer Science*, pages 122 – 129, November 2004. doi:10.1007/978-3-540-30499-9_18.

[184] T. Nitta. A back-propagation algorithm for complex numbered neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 1649 – 1652. IEEE, October 1993. doi:10.1109/IJCNN.1993.716968.

[185] T. Nitta. A back-propagation algorithm for neural networks based on 3d vector product. In *International Joint Conference on Neural Netwoks (IJCNN)*, vo-

lume 1, pages 589 – 592. IEEE, 1993. doi:10.1109/IJCNN.1993.713984.

[186] T. Nitta. An extension of the back-propagation algorithm to three dimensions by vector product. In *International Conference on Tools with Artificial Intelligence (TAI)*, pages 460 – 461. IEEE, 1993. doi:10.1109/TAI.1993.634002.

[187] T. Nitta. An analysis on decision boundaries in the complex back-propagation network. In *IEEE International Conference on Neural Networks and IEEE World Congress on Computational Intelligence*, volume 2, pages 934 – 939. IEEE, July 1994. doi:10.1109/ICNN.1994.374306.

[188] T. Nitta. Generalization ability of the three-dimensional back-propagation network. In *International Conference on Neural Networks*, volume 5, pages 2895 – 2900. IEEE, 1994. doi:10.1109/ICNN.1994.374691.

[189] T. Nitta. Structure of learning in the complex numbered back-propagation network. In *IEEE World Congress on Computational Intelligence and IEEE International Conference on Neural Networks*, volume 1, pages 269 – 274. IEEE, July 1994. doi:10.1109/ICNN.1994.374173.

[190] T. Nitta. A quaternary version of the back-propagation algorithm. In *International Conference on Neural Networks*, number 5, pages 2753 – 2756. IEEE, 1995. doi:10.1109/ICNN.1995.488166.

[191] T. Nitta. An extension of the back-propagation algorithm to complex numbers. *Neural Networks*, 10(8):1391 – 1415, November 1997. doi:10.1016/S0893-6080(97)00036-1.

[192] T. Nitta. A sufficient condition for decision boundaries in neural networks with complex numbered weights to intersect orthogonally. In *International Conference on Neural Information Processing (ICONIP)*, volume 1, pages 95 – 100. IEEE, November 1999. doi:10.1109/ICONIP.1999.843968.

[193] T. Nitta. An analysis of the fundamental structure of complex-valued neurons. *Neural Processing Letters*, 12(3):239 – 246, December 2000. doi:10.1023/A:1026582217675.

[194] T. Nitta. The computational power of complex-valued neuron. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003*, volume 2714 of *Lecture Notes in Computer Science*, pages 993 – 1000. Springer Berlin Heidelberg, 2003. doi:10.1007/3-540-44989-2_118.

[195] T. Nitta. On the inherent property of the decision boundary in complex-valued neural networks. *Neurocomputing*, 50:291 – 303, January 2003. doi:10.1016/S0925-2312(02)00568-4.

[196] T. Nitta. Solving the xor problem and the detection of symmetry using a single complex-valued neuron. *Neural Networks*, 16(8):1101 – 1105, October 2003. doi:10.1016/S0893-6080(03)00168-0.

[197] T. Nitta. Orthogonality of decision boundaries in complex-valued neural networks. *Neural Computation*, 16(1):73 – 97, January 2004. doi:10.1162/08997660460734001.

[198] T. Nitta. A solution to the 4-bit parity problem with a single quaternary neuron. *Neural Information Processing - Letters and Review*, 5(2):33 – 39, November 2004.

[199] T. Nitta. Three-dimensional vector valued neural network and its generalization ability. *Neural Information Processing - Letters and Reviews*, 10(10):237 – 242, 2006.

[200] T. Nitta. N-dimensional vector neuron. In *IJCAI Workshop on Complex-Valued Neural Networks and Neuro-Computing: Novel Methods, Applications and Implementations*, pages 2 – 7, 2007.

[201] T. Nitta, editor. *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*. Information Science Publishing, Hershey, New York, 2009. doi:10.4018/978-1-60566-214-5.

[202] T. Nitta. *Complex-Valued Neural Networks: Advances and Applications*, chapter N-Dimensional Vector Neuron and its Application to the N-Bit Parity Problem, pages 59 – 74. John Wiley & Sons, Inc., 2013. doi:10.1002/9781118590072.ch3.

[203] T. Nitta and S. Buchholz. On the decision boundaries of hyperbolic neurons. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2974 – 2980. IEEE, 2008. doi:10.1109/IJCNN.2008.4634216.

[204] A.J. Noest. Associative memory in sparse phasor neural networks. *Europhysics Letters*, 6(6):469 – 474, 1988. doi:10.1209/0295-5075/6/5/016.

[205] A.J. Noest. Discrete-state phasor neural networks. *Physics Review A*, 38(4):2196 – 2199, August 1988. doi:10.1103/PhysRevA.38.2196.

[206] M. Novey and T. Adali. Complex ica by negentropy maximization. *IEEE Transactions on Neural Networks*, 19(4):596 – 609, April 2008. doi:10.1109/TNN.2007.911747.

[207] P.L. Papini and J. Puerto. Averaging the k largest distances among n: k-centra in banach spaces. *Journal of Mathematical Analysis and Applications*, 291(2):477 – 487, March 2004. doi:10.1016/j.jmaa.2003.11.011.

[208] D.-C. Park and T.-K.J. Jeong. Complex-bilinear recurrent neural network for equalization of a digital satellite channel. *IEEE Transactions on Neural Networks*, 13(3):711 – 725, May 2002. doi:10.1109/TNN.2002.1000135.

[209] B.A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147 – 160, January 1994. doi:10.1162/neco.1994.6.1.147.

[210] J.K. Pearson. *Clifford Networks*. PhD thesis, University of Kent, 1995. URL: http://user.it.uu.se/~justin/Research/thesis.ps.

[211] J.K. Pearson and D.L. Bisset. Back propagation in a clifford algebra. In *International Conference on Artificial Neural Networks*, volume 2, pages 413 – 416, 1992.

[212] J.K. Pearson and D.L. Bisset. Neural networks in the clifford domain. In *International Conference on Neural Networks*, volume 3, pages 1465 – 1469. IEEE, 1994. doi:10.1109/ICNN.1994.374502.

[213] X. Pennec. Computing the mean of geometric features application to the mean rotation. Rapport des Recherche 3371, Institut National de Recherche en Informatique et en Automatique, 1998. URL: https://hal.archives-ouvertes.fr/file/index/docid/73318/filename/RR-3371.pdf.

[214] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 3(16):35 – 43, 1969.

[215] C.-A. Popa. Enhanced gradient descent algorithms for complex-valued neural networks. In *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 272 – 279. IEEE, September 2014. doi:10.1109/SYNASC.2014.44.

[216] C.-A. Popa. Exact hessian matrix calculation for complex-valued neural networks. In *International Workshop on Soft Computing Applications (SOFA)*. Springer, July 2014.

[217] C.-A. Popa. Conjugate gradient algorithms for complex-valued neural networks. communicated, 2015.

[218] C.-A. Popa. Lie algebra-valued neural networks. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015.

[219] C.-A. Popa. Matrix-valued neural networks. In *International Conference on Soft*

*Computing (MENDEL)*. Springer, June 2015.

[220] C.-A. Popa. Quasi-newton learning methods for complex-valued neural networks. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015.

[221] C.-A. Popa. Scaled conjugate gradient learning for complex-valued neural networks. In *International Conference on Soft Computing (MENDEL)*. Springer, June 2015.

[222] M.J.D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241 – 254, 1977. doi:10.1007/BF01593790.

[223] A.F.R. Rahman, G. Howells, and M.C. Fairhurst. A multiexpert framework for character recognition: A novel application of clifford networks. *IEEE Transactions on Neural Networks*, 12(1):101 – 112, January 2001. doi:10.1109/72.896799.

[224] S.S.P. Rattan and W.W. Hsieh. Complex-valued neural networks for nonlinear complex principal component analysis. *Neural Networks*, 18(1):61 – 69, January 2005. doi:10.1016/j.neunet.2004.08.002.

[225] C.M. Reeves and R. Fletcher. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149 – 154, 1964. doi:10.1093/comjnl/7.2.149.

[226] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE International Conference on Neural Networks*, volume 1, pages 586 – 591. IEEE, March 1993. doi:10.1109/ICNN.1993.298623.

[227] C. Samir, P.-A. Absil, A. Srivastava, and E. Klassen. A gradient-descent method for curve fitting on riemannian manifolds. *Foundations of Computational Mathematics*, 12(1):49 – 73, February 2012. doi:10.1007/s10208-011-9091-7.

[228] E. Sathish, M. Sivachitra, R. Savitha, and S. Vijayachitra. Wind profile prediction using a meta-cognitive fully complex-valued neural network. In *International Conference on Advanced Computing (ICoAC)*, pages 1 – 6. IEEE, December 2012. doi:10.1109/ICoAC.2012.6416850.

[229] D.H. Sattinger and O.L. Weaver. *Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics*, volume 61 of *Applied Mathematical Sciences*. Springer-Verlag New York, 1986. doi:10.1007/978-1-4757-1910-9.

[230] R. Savitha, S. Suresh, and N. Sundararajan. Complex-valued function approximation using a fully complex-valued rbf (fc-rbf) learning algorithm. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2819 – 2825. IEEE, June 2009. doi:10.1109/IJCNN.2009.5178624.

[231] R. Savitha, S. Suresh, and N. Sundararajan. A fully complex-valued radial basis function network and its learning algorithm. *International Journal of Neural Systems*, 19(4):253 – 267, August 2009. doi:10.1142/S0129065709002026.

[232] R. Savitha, S. Suresh, and N. Sundararajan. A self-regulated learning in fully complex-valued radial basis function networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 8. IEEE, July 2010. doi:10.1109/IJCNN.2010.5596781.

[233] R. Savitha, S. Suresh, and N. Sundararajan. A fast learning complex-valued neural classifier for real-valued classification problems. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2243 – 2249. IEEE, August 2011. doi:10.1109/IJCNN.2011.6033508.

[234] R. Savitha, S. Suresh, and N. Sundararajan. A meta-cognitive learning algorithm for a fully complex-valued relaxation network. *Neural Networks*, 32:209 – 218, August 2012. doi:10.1016/j.neunet.2012.02.015.

[235] R. Savitha, S. Suresh, and N. Sundararajan. Projection-based fast learning

fully complex-valued relaxation neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 24(4):529 – 541, April 2013. doi:10.1109/TNNLS.2012.2235460.

[236] R. Savitha, S. Suresh, N. Sundararajan, and P. Saratchandran. Complex-valued function approximation using an improved bp learning algorithm for feed-forward networks. In *IEEE International Joint Conference on Neural Networks, IJCNN (IEEE World Congress on Computational Intelligence)*, pages 2251 – 2258. IEEE, June 2008. doi:10.1109/IJCNN.2008.4634109.

[237] R. Savitha, S. Suresh, N. Sundararajan, and P. Saratchandran. A new learning algorithm with logarithmic performance index for complex-valued neural networks. *Neurocomputing*, 72(16 - 18):3771 – 3781, October 2009. doi:10.1016/j.neucom.2009.06.004.

[238] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647 – 656, July 1970. doi:10.1090/S0025-5718-1970-0274029-X.

[239] I. Sharf, A. Wolf, and M.B. Rubin. Arithmetic and geometric solutions for average rigid-body rotation. *Mechanism and Machine Theory*, 45(9):1239 – 1251, September 2010. doi:10.1016/j.mechmachtheory.2010.05.002.

[240] M.D. Shuster. A survey of attitude representations. *The Journal of the Astronautical Sciences*, 41(4):439 – 517, October - December 1993. URL: http://malcolmdshuster.com/Pub_1993h_J_Repsurv_scan.pdf.

[241] G. Sommer, editor. *Geometric Computing with Clifford Algebras Theoretical Foundations and Applications in Computer Vision and Robotics*. Springer Berlin Heidelberg, 2001. doi:10.1007/978-3-662-04621-0.

[242] J. Song and Y. Yam. Complex recurrent neural network for computing the inverse and pseudo-inverse of the complex matrix. *Applied Mathematics and Computation*, 93(2 - 3):195 – 205, July 1998. doi:10.1016/S0096-3003(97)10064-9.

[243] R. Subbarao and P. Meer. Nonlinear mean shift over riemannian manifolds. *International Journal of Computer Vision*, 84(1):1 – 20, August 2009. doi:10.1007/s11263-008-0195-8.

[244] S. Suresh, R. Savitha, and N. Sundararajan. A sequential learning algorithm for complex-valued self-regulating resource allocation network-csran. *IEEE Transactions on Neural Networks*, 22(7):1061 – 1072, July 2011. doi:10.1109/TNN.2011.2144618.

[245] S. Suresh, N. Sundararajan, and R. Savitha. *Supervised Learning with Complex-Valued Neural Networks*, volume 421 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-29491-4.

[246] K. Tachibana and E.M.S. Hitzer. Tutorial note on geometric algebra neural networks. In *AGACSE 3*, pages 1 – 20, August 2008. URL: http://erkenntnis.icu.ac.jp/gcj/publications/AGACSE2008NNtut/AGACSE2008NNtut.pdf.

[247] H. Takahashi. Covariance phasor neural network as a mean field model. In *International Conference on Neural Information Processing (ICONIP)*, volume 3, pages 1089 – 1093. IEEE, November 2002. doi:10.1109/ICONIP.2002.1202790.

[248] M. Takeda and T. Kishigami. Complex neural fields with a hopfield-like energy function and an analogy to optical fields generated in phase-conjugate resonators. *Journal of Optical Society of America A*, 9(12):2182 – 2191, December 1992. doi:10.1364/JOSAA.9.002182.

[249] G. Tanaka and K. Aihara. Complex-valued multistate associative memory with nonlinear multilevel functions for gray-level image reconstruction. *IEEE Transactions on Neural Networks*, 20(9):1463 – 1473, 2009. doi:10.1109/TNN.2009.

2025500.

[250] T. Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3(5):561 – 573, 1990. doi:10.1016/0893-6080(90)90006-7.

[251] C.C. Took and D.P. Mandic. The quaternion lms algorithm for adaptive filtering of hypercomplex processes. *IEEE Transactions on Signal Processing*, 57(4):1316 – 1327, April 2009. doi:10.1109/TSP.2008.2010600.

[252] C.C. Took and D.P. Mandic. Quaternion-valued stochastic gradient-based adaptive iir filtering. *IEEE Transactions on Signal Processing*, 58(7):3895 – 3901, July 2010. doi:10.1109/TSP.2010.2047719.

[253] C.C. Took and D.P. Mandic. A quaternion widely linear adaptive filter. *IEEE Transactions on Signal Processing*, 58(8):4427 – 4431, August 2010. doi:10.1109/TSP.2010.2048323.

[254] C.C. Took, D.P. Mandic, and K. Aihara. Quaternion-valued short term forecasting of wind profile. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 6. IEEE, July 2010. doi:10.1109/IJCNN.2010.5596690.

[255] C.C. Took, D.P. Mandic, and J. Benesty. Study of the quaternion lms and four-channel lms algorithms. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3109 – 3112. IEEE, April 2009. doi:10.1109/ICASSP.2009.4960282.

[256] C.C. Took, G. Strbac, K. Aihara, and D.P. Mandic. Quaternion-valued short-term joint forecasting of three-dimensional wind and atmospheric parameters. *Renewable Energy*, 36(6):1754 – 1760, June 2011. doi:10.1016/j.renene.2010.12.013.

[257] B.K. Tripathi and P.K. Kalra. Learning machine using heterogeneous neurons with high dimensional parameters. In *IEEE International Conference on System of Systems Engineering*, pages 1 – 6. IEEE, April 2007. doi:10.1109/SYSOSE.2007.4304324.

[258] B.K. Tripathi and P.K. Kalra. The novel aggregation function-based neuron models in complex domain. *Soft Computing*, 14(10):1069 – 1081, 2010. doi:10.1007/s00500-009-0502-5.

[259] A. Uncini and F. Piazza. Blind signal processing by complex domain adaptive spline neural networks. *IEEE Transactions on Neural Networks*, 14(2):399 – 412, March 2003. doi:10.1109/TNN.2003.809411.

[260] M.E. Valle. A novel continuous-valued quaternionic hopfield neural network. In *Brazilian Conference on Intelligent Systems (BRACIS)*, pages 97 – 102. IEEE, October 2014. doi:10.1109/BRACIS.2014.28.

[261] J.R. Vallejo and E. Bayro-Corrochano. Clifford hopfield neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3609 – 3612. IEEE, June 2008. doi:10.1109/IJCNN.2008.4634314.

[262] B. Vandereycken, P.-A. Absil, and S. Vandewalle. A riemannian geometry with complete geodesics for the set of positive semidefinite matrices of fixed rank. *IMA Journal of Numerical Analysis*, 33(2):481 – 514, April 2013. doi:10.1093/imanum/drs006.

[263] P. Vayanos, S.L. Goh, and D.P. Mandic. Online detection of the nature of complex-valued signals. In *IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 173 – 178. IEEE, September 2006. doi:10.1109/MLSP.2006.275543.

[264] B. Widrow, J. McCool, and M. Ball. The complex lms algorithm. *Proceedings of the IEEE*, 63(4):719 – 720, April 1975. doi:10.1109/PROC.1975.9807.

[265] Y. Xia, C. Jahanchahi, and D.P. Mandic. Quaternion-valued echo state networks.

*IEEE Transactions on Neural Networks and Learning Systems*, 26(4):663 – 673, April 2015. doi:10.1109/TNNLS.2014.2320715.

[266] Y. Xia, B. Jelfs, M.M. Van Hulle, J.C. Principe, and D.P. Mandic. An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals. *IEEE Transactions on Neural Networks*, 22(1):74 – 83, January 2011. doi:10.1109/TNN.2010.2085444.

[267] Y. Xia and D.P. Mandic. An adaptive diffusion augmented clms algorithm for distributed filtering of noncircular complex signals. *IEEE Signal Processing Letters*, 2011.

[268] D. Xu, H. Zhang, and L. Liu. Convergence analysis of three classes of split-complex gradient algorithms for complex-valued recurrent neural networks. *Neural Computation*, 22(10):2655 – 2677, October 2010. doi:10.1162/NECO_a_00021.

[269] Q. Xu and D. Ma. Applications of lie groups and lie algebra to computer vision a brief survey. In *International Conference on Systems and Informatics (ICSAI)*, pages 2024 – 2029. IEEE, May 2012. doi:10.1109/ICSAI.2012.6223449.

[270] R. Yamaki and A. Hirose. Singular unit restoration in interferograms based on complex-valued markov random field model for phase unwrapping. *IEEE Geoscience and Remote Sensing Letters*, 6(1):18 – 22, January 2009. doi:10.1109/LGRS.2008.2005588.

[271] C.-C. Yang and N.K. Bose. Landmine detection and classification with complex-valued hybrid neural network using scattering parameters dataset. *IEEE Transactions on Neural Networks*, 16(3):743 – 753, May 2005. doi:10.1109/TNN.2005.844906.

[272] S.-S. Yang, C.-L. Ho, and S. Siu. Sensitivity analysis of the split-complex valued multilayer perceptron due to the errors of the i.i.d. inputs and weights. *IEEE Transactions on Neural Networks*, 18(5):1280 – 1293, September 2007. doi:10.1109/TNN.2007.894038.

[273] S.-S. Yang, S. Siu, and C.-L. Ho. Analysis of the initial values in split-complex backpropagation algorithm. *IEEE Transactions on Neural Networks*, 19(9):1564 – 1573, September 2008. doi:10.1109/TNN.2008.2000805.

[274] W.-H. Yang, K.-K. Chan, and P.-R. Chang. Complex-valued neural network for direction of arrival estimation. *Electronics Letters*, 30(7):574 – 575, March 1994. doi:10.1049/el:19940400.

[275] Q. Yi and B.W. York. Some results for chaotic times series prediction using clifford neural networks. In *International Conference on Signal Processing (ICSP)*, volume 1, pages 61 – 64. IEEE, September 2004. doi:10.1109/ICOSP.2004.1452581.

[276] C. You and D. Hong. Nonlinear blind equalization schemes using complex-valued multilayer feedforward neural networks. *IEEE Transactions on Neural Networks*, 9(6):1442 – 1455, November 1998. doi:10.1109/72.728394.

[277] H. Yu and B.M. Wilamowski. *Intelligent Systems*, volume 5 of *Industrial Electronics Handbook*, chapter Levenberg-Marquardt Training, pages 12–1 – 12–15. CRC Press, 2nd edition, February 2011.

[278] H. Zhang and W. Wu. Convergence of split-complex backpropagation algorithm with momentum. *Neural Network World*, 21(1):75 – 90, 2011. doi:10.14311/nnw.2011.21.006.

[279] H. Zhang, D. Xu, and Wang Z. Convergence of an online split-complex gradient algorithm for complex-valued neural networks. *Discrete Dynamics in Nature and Society*, 2010:1 – 27, January 2010. doi:10.1155/2010/829692.

[280] Y. Zhang and Y. Ma. Cgha for principal component extraction in the complex

domain. *IEEE Transactions on Neural Networks*, 8(5):1031 – 1036, September 1997. doi:10.1109/72.623205.

[281] H.Q. Zhao, X.P. Zeng, Z.Y. He, W.D. Jin, and T.R. Li. Complex-valued pipelined decision feedback recurrent neural network for non-linear channel equalisation. *IET Communications*, 6(9):1082 – 1096, June 2012. doi:10.1049/iet-com.2011.0415.