Contribuţii la antrenarea reţelelor neuronale. Învăţarea pe baza corecţiei erorii cu exemple negative

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul "Ştiinţa calculatoarelor"
de către

Cernăzanu-Glăvan Cosmin

Conducător ştiinţific: Prof. Univ. Dr. Ing. Ştefan Holban

octombrie 2009

BUPT

CUPRINS

BUPT

BUPT

BUPT

BUPT

LISTA FIGURILOR

BUPT

BUPT

BUPT

BUPT

LISTA TABELELOR

BUPT

BUPT

1. INTRODUCERE

Singurul obstacol împotriva lumii este o cunoaştere aprofundată a acesteia.(John Locke)

Supravieţuirea noastră ca indivizi şi a societăţii ca întreg, este cea mai importantă nevoie a omenirii şi de cele mai multe ori este strâns legată de cunoaşterea mediului înconjurător. Pentru satisfacerea acesteia omul a trebuit să creeze diverse unelte capabile să-l ajute de-a lungul acestui proces ce durează de zeci de mii de ani.

Ca ultimă unealtă, calculatorul electronic a fost creat pentru a procesa informaţia la o viteză şi într-un mod care depăşeşte de miliarde de ori capacitatea creierului omenesc. Totuşi, această imensă putere de calcul s-a dovedit a fi neputincioasă în rezolvarea unor probleme simple dar pentru care nu există un algoritm de calcul predefinit.

Un algoritm de calcul este, pentru un calculator, similar unei hărţi pentru un om aflat prima oară într-un oraş străin. Pentru a putea ajunge într-un loc anume, omul are nevoie în permanenţă de a şti în ce loc se află şi care este direcţia în care să meargă. Din această cauză, pentru a putea servi cu succes, harta trebuie în prealabil făcută de o altă persoană care trebuie să aibă toate informaţiile despre acea regiune deoarece orice inexactitate poate conduce spre un drum greşit.

Comparaţia unui algoritm de calcul cu o hartă pentru un om străin de un loc se încheie aici, omul dispunând de multe alte resurse pentru a putea afla un drum bun chiar şi cu o hartă greşită. Astfel, un om poate afla informaţii întrebând un alt om, încercând să se orienteze după alte lucruri (clădiri, forme de relief, puncte cardinale, etc.), încercând aplicarea unei strategii de căutare dezvoltată într-o situaţie similară, etc.

Se conturează un anumit model de comportament care evidenţiază clar capacitatea omului, aflat într-o situaţie nouă, de a lua decizii bazându-se pe experienţa şi pe informaţiile acumulate anterior. Acest mod de lucru este caracteristic creierului şi din această cauză pentru a rezolva acest gen de probleme s-a încercat dezvoltarea unor structuri care încearcă copierea lui. Plecând de la modelele neurale biologice, oamenii au încercat să folosească proprietăţile şi modul de construcţie al acestora pentru a crea un model artificial asemănător.

Problema majoră întâlnită în procesul de construcţie a fost şi rămâne uimitoarea complexitate a creierului uman. „Creierul uman este lucrul cel mai minunat şi cel mai misterios din tot Universul”, spune Hanry F. Osborn, un renumit antropolog. Dispunând de 10^{11} neuroni (celule nervoase) conectaţi între ei prin 10^{14} sinapse [1] este greu de imaginat un dispozitiv artificial capabil de a imita această complexitate.

Chiar dacă viteza de lucru pentru un neuron biologic este foarte mică în comparaţie cu viteza de care este capabil un calculator, datorită numărului mare de neuroni şi de conexiuni, creierul uman este net superior faţă de cel mai dezvoltat supercalculator actual.

Mai mult, încă de la naştere, un creier are o structură care îi permite acumularea de informaţii şi dezvoltarea unor noi reguli de aplicare a acestora, ce conduce în timp la apariţia *experienţei*. Aceasta ne va însoţi pe toată durata vieţii acumulându-se şi crescând în complexitate pentru fiecare acţiune de-a noastră.

BUPT

În acest fel s-a dorit implementarea reţelelor neuronale artificiale: capabile de acumulări de cunoştinţe în vederea utilizării acestora într-o situaţie complet similară.

1.1. Motivaţie

Plecând de la cele scrise mai sus, se poate spune că domeniul reţelelor neuronale artificiale este un domeniu în continuă dezvoltare cu limite care sunt departe de a fi atinse. Dacă începutul acestuia a fost dat de o maşină care să poată converti o imagine în informaţie şi să poată ulterior să recunoască acea imagine, sfârşitul acestuia ţine de domeniul filosofiei şi al eticii. Spun acestea deoarece de-a lungul dezvoltării acestui domeniu s-au ridicat o serie de întrebări legate de posibile repercusiuni care pot apărea.

Vor reuşi reţelele neuronale să construiască o *conştiinţă* în interiorul unei maşini?

Avem nevoie de o astfel de maşină, care să poată discerne informaţiile şi chiar să poate emite raţionamente cu noţiuni complexe (exemplu fiind conceptele de *bine* şi *rău*).

Unde va fi trasată graniţa de control asupra acestei maşini şi cine va deţine în viitor puterea, omul sau maşina?

Deocamdată omul este cea mai inteligentă fiinţă de pe Pământ şi acest lucru nu se va schimba curând. În sprijinul acestei afirmaţii stă argumentul evoluţiei creierului, care de milioane de ani a parcurs rigorile de supravieţuire date de condiţiile ostile de viaţă. Chiar dacă reţelele neuronale artificiale vor avansa într-atât de mult încât se va putea face o comparaţie cu creierul uman, ce ne împiedică pe noi ca indivizi să nu evoluăm mai mult decât suntem azi capabili?

De fapt, însăşi dezvoltarea acestui domeniu este un act de evoluţie semnificativă a omului, acesta ajungând astfel să conceapă unelte care pot modela procese, până acum considerate, strict biologice.

Un alt aspect care trebuie luat în considerare este diferenţa dintre cele două inteligenţe, cea umană şi cea artificială. Un exemplu în acest sens poate fi făcut dacă raportăm inteligenţa umană la inteligenţa unui delfin (al doilea organism viu inteligent de pe această planetă după om). Delfinii utilizează inteligenţa preponderent în explorarea mediului marin, deoarece, deşi acesta este mai simplu decât cel terestru, necesită o coordonare mult mai complicată şi mai rapidă a răspunsurilor animalelor faţă de factorii de mediu[2].

Aşa cum inteligenţa delfinilor este influenţată strict de mediul în care acesta trăieşte se poate afirma că inteligenţa maşinii va fi influenţată de mediul în care aceasta va fi antrenată. Este aproape imposibil de imaginat forma pe care o va lua inteligenţa unei reţele neuronale, dar cu siguranţă suntem implicaţi în procesul de formare a acesteia şi putem controla, cel puţin la început, toate informaţiile care îi sunt accesibile.

Dacă considerăm celebrul calculator Deep Blue [3], creat special pentru jocul de şah, putem spune că această *inteligenţă* a fost formată numai cu scopul de cunoaştere a unui anumit joc. Cred că este oarecum ironic faptul că cea mai mare realizare a noastră în domeniul *inteligenţei* este un calculator care ştie să joace un singur joc, dar dacă privim la progresele care s-au făcut în ultimii 50 de ani ne dăm seama că suntem pe drumul cel bun în acest domeniu chiar dacă limita pare din ce în ce mai departe.

BUPT

Început ca un model ce simula un neuron artificial, de către Warren McCulloch şi Walter Pitts în 1943 [4], domeniul reţelelor neuronale este unul în continuă dezvoltare, capabil de a stimula şi a înfierbânta minţile multor oameni de acum înainte. Deşi ridică numeroase întrebări la care nu putem găsi încă răspuns, el oferă şi numeroase soluţii la problemele omenirii. O reţea neuronală artificială, capabilă de a discerne în situaţii noi, este o realizare extraordinară din punct de vedere tehnologic şi informaţional. Deja o serie de roboţi lucrează cu succes pentru noi în medii inaccesibile omului, dând dovadă de o precizie şi o rapiditate imposibil de atins de către om.

În încercarea de a micşora tot mai mult distanţa dintre o reţea neuronală artificială şi una biologică am acordat o foarte mare atenţie exemplelor negative, deoarece pentru o reţea biologică acestea sunt la fel de importante ca şi cele pozitive.

De ce să nu facem un pas mai departe în acest domeniu şi să-i oferim reţelei neuronale şi capacitatea de a învăţa şi din exemple negative, lucru ce poate să-i confere acesteia o personalitate şi o evoluţie în viitor?

1.2. Reţele neuronale artificiale(RNA)

Reţelele neuronale artificiale sunt modele matematice care încearcă emularea unei reţele neuronale biologice(creierul uman). Pentru a arăta acest lucru vom porni de la cea mai simplă entitate prezentă în ambele reţele: neuronul.

Neuronul biologic este elementul fundamental al sistemului nervos şi se prezintă schematic în Figura 1.

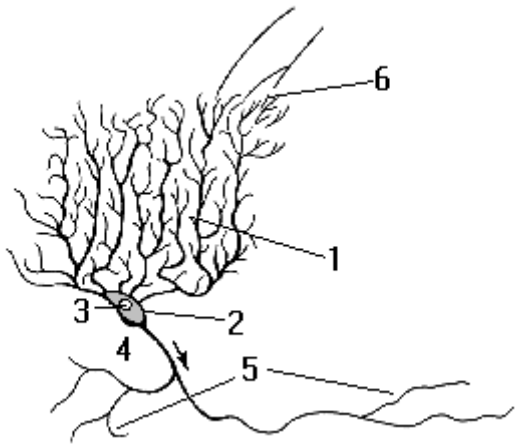


Figura 1. Reprezentarea schematică a neuronului biologic. . 1 – Arborele dendritic; 2 – Soma(corpul celular); 3 – Nucleul celulei neuronale; 4 – Axonul; 5 – Arborele axonic; 6 – conexiuni sinaptice

Neuronul biologic este o celulă capabilă de captarea, procesarea şi trimiterea semnalelor electromagnetice de-a lungul conexiunilor sinaptice. Semnalele electrice primite de la alţi neuroni prin intermediul conexiunilor sinaptice sunt *însumate* la nivelul corpului celular prin diverse procese fizico chimice. Atunci când potenţialul recepţionat depăşeşte o anumită valoare de prag se eliberează un potenţial(numit potenţial de acţiune) de-a lungul arborelui axonic.[5]

BUPT

S-a observat că dacă o pereche de neuroni este stimulată succesiv creşte permeabilitatea sinapsei dintre cei doi neuroni, asigurând astfel o trecere mai facilă a semnalului electric. O altă observaţie este aceea că potenţialul de acţiune pentru un neuron nu depinde de durata sau intensitatea stimulilor având în toate situaţiile amplitudinea constantă. De asemenea sinapsele neuronilor sunt de două feluri excitatoare sau inhibitoare generând astfel o sarcină pozitivă sau negativă.

De cealaltă parte, neuronul artificial se prezintă ca în Figura 2.

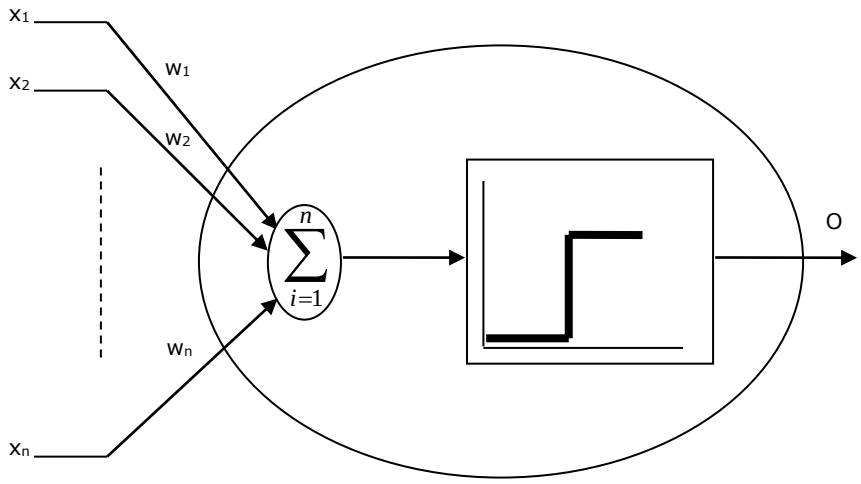


Figura 2. Modelul neuronului artificial. Vectorul X – intrările reţelei, W – ponderile reţelei, O – ieşirea reţelei

Dacă facem o comparaţie cu neuronul biologic observăm că rolul dendritelor este preluat de intrările neuronului, permeabilitatea sinapsei este dată de ponderile reţelei iar valoarea de prag şi eliberarea potenţialului de acţiune a fost înlocuită printr-o funcţie de activare.

Funcţionalitatea acestuia este destul de simplă: fiecare intrare are asociată o pondere; se însumează toate produsele dintre intrare şi ponderea asociată obţinându-se o valoare. Această valoare este aplicată unei funţii de activare, care are un rol de barieră, lăsând să treacă doar valorile ce depăşesc un anumit prag.

Modelul artificial pleacă de la o descriere simplistă a neuronului biologic. Deşi neuronul biologic prezintă mai multe funcţii, existând chiar în forme şi cu funcţii diferite, procesul prin care acesta stochează şi prelucrează informaţie este asemănător cu cel artificial. Daca la acest nivel există o mare asemănare între cele două tipuri de neuroni, de ce reţelele neuronale artificiale nu sunt la fel de performante ca reţelele neuronale biologice?

Puterea creierului stă însă în capacitatea neuronilor de a se interconecta în reţele neuronale. Într-o astfel de reţea un neuron poate să fie conectat cu încă 10.000 de alţi neuroni rezultând astfel o structură extrem de complexă (Figura 3)

Puterea de calcul ce rezidă în această interconectivitate este ilustrată de exemplul următor. Viteza de propagare a unei informaţii printr-un neuron biologic este de 10^{-3} secunde în timp ce viteza de propagare a informaţiei printr-un neuron artificial este de 10^{-10} secunde, ceea ce spune foarte mult despre posibilităţile unui

BUPT

calculator. Cu toate acestea recunoaşterea vizuală a unei persoane apropiate se face în mai puţin de 10^{-1} secunde. Din această cauză putem nota faptul că informaţie nu poate parcurge mai mult de 100 de neuroni. Dată fiind complexitatea operaţiei (calculatoarele nu sunt capabile de o asemenea procesare decât după o amplă antrenare în prealabil şi nu obţin succes în toate cazurile) deducem că într-o reţea neuronală biologică operează cu succes o paralelizare masiva a informaţiei. [6]

Acest aspect important al activităţii reţelei neuronale biologice este şi cel mai utilizat în proiectarea reţelelor neuronale artificiale. Fiecare nou tip de reţea neuronală artificială încearcă să obţină cât mai multe avantaje din această paralelizare.

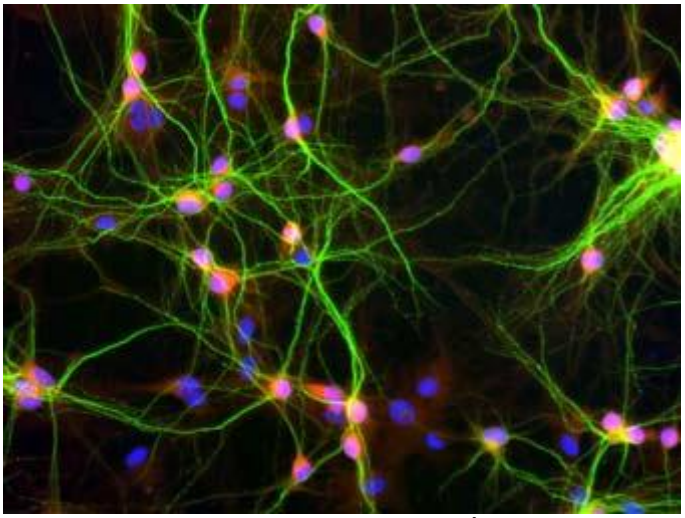


Figura 3. Interconexiunea neuronilor biologici într-o reţea neuronală

Un alt aspect pe care reţelele artificiale l-au copiat de la cele naturale este acela al creşterii permeabilităţii dintr-o pereche de neuroni stimulată succesiv. Plecând iniţial de la prezumţiile enumerate de către Ramon y Cajal[7], Daniel Hebb defineşte în 1949 un concept fundamental despre modul în care omul învaţă[8]. El afirmă că dacă doi neuroni eliberează în acelaşi timp un potenţial de acţiune, conectivitatea dintre ei va creşte.

Acest concept a fost preluat de Rosenblatt abia în anul 1958, când a fost dezvoltat primul algoritm de antrenare a unei reţele neuronale artificiale.[9] Aceasta este cea mai simplă formă de reţea neuronală şi a fost numită perceptron.

În 1962, Bernard Widrow şi Marcian Hoff definesc algoritmul de învăţare bazat pe minimizarea funcţiei de eroare[25]. În acest fel eroarea rezultată la ieşirile reţelei este „distribuită" ponderilor în funcţie de valoarea acestora.

În 1972, Kohonen, descrie un nou tip de reţea neuronală.[10] Particularitatea acestei reţele este dată de faptul că ieşirile mai multor neuroni sunt active pentru aceeaşi intrare. Aceasta este văzută ca fiind o mapare între două spaţii diferite, unul mai dens decât altul.

Urmează o perioadă în care atenţia asupra domeniului reţelelor neuronale creşte foarte mult. Conceptele acestora sunt preluate de scriitori importanţi şi de producători de film.[11][12] Apar astfel o serie de idei negative despre roboţi care domină umanitatea precum şi de calculatoare dotate cu inteligenţă ce influenţează negativ dezvoltarea rasei umane. Toate acestea duc la apariţia unor critici vizavi de

BUPT

inteligenţa artificială şi reţelele neuronale şi la o diminuare a fondurilor pentru aceste domenii.

Interesul reapare în 1982, când John Hopfield de la Caltech, încearcă o nouă abordare a acestui domeniu, de această dată încercând să copieze mecanismele de gândire ale creierului şi nu structura sa internă.[13]

Tot în acest an Japonia anunţă începutul unei noi dezvoltări în industria calculatoarelor, şi anume apariţia generaţiei a 5-a de calculatoare.[14] Impulsionată de faptul că ar putea rămâne în urmă în această competiţie, Statele Unite încep să refinanţeze proiectele legate de inteligenţă artificială şi reţele neuronale.

În 1986 se dezvoltă algoritmul de antrenare pentru o reţea de tip perceptron cu mai multe straturi de către Rumelhart, Hinton şi Williams.[15] Numit, „Algoritm cu propagare înapoi a erorii" acesta se dovedeşte a fi prea lent pentru a putea antrena o reţea neuronală complexă.

În prezent, reţelele neuronale sunt utilizate în foarte multe aplicaţii având un caracter „domestic", dar antrenarea acestora este un proces de lungă durată. Performanţa acestora este însă pusă în evidenţă atunci când avem şi un hardware specializat care să implementeze electronic principiile de funcţionare ale sistemului nervos.

Tocmai de aceea se pune un mare accent pe dezvoltarea neuro-chip-urilor şi utilizarea acestora în procesul de creare şi antrenare a reţelelor neuronale.

1.3. Obiective

Îmi propun să fac încă un pas în dezvoltarea reţelelor neuronale, prin studierea rolului pe care exemplele negative îl au în procesul de antrenare, aducând astfel RNA mai aproape de un comportament uman.

Plecând de la premisa că în dezvoltarea unui model comportamental experienţele negative au un rol cel puţin egal cu cele pozitive(cine nu a auzit de expresia „Şi din greşeli ai ce învăţa") se încearcă o analiză a acestor exemple negative mergând până la modificarea topologiei reţelei neuronale pentru a accepta şi accentua aceste exemple. Toate informaţiile rezultate vor fi apoi utilizate pentru definirea unui nou algoritm de învăţare : învăţarea pe baza corecţiei erorii cu ajutorul exemplelor negative.

Am definit astfel ca direcţie principală de cercetare, rolul exemplelor negative în antrenarea unei reţele neuronale de tip multilayer perceptron. Doresc să analizez modul în care acestea influenţează procesul de antrenare şi dacă este posibil, să ofer o îmbunătăţire a performanţelor reţelelor.

Astfel, formulez o serie de obiective ştiinţifice ce se doresc a fi rezolvate de către această lucrare:

- **Găsirea procentului de exemple negative** care să poată fi utilizat cu succes la antrenarea reţelelor neuronale, şi care să aducă un maxim de performanţă la utilizarea în producţie a reţelei. **Crearea unui model matematic general** pentru procesul de antrenare cu exemple negative.
- **Modificarea arhitecturii RNA** astfel încât să permită o mai bună evidenţiere a exemplelor negative. În acest caz se intenţionează **modificarea straturilor de ieşire** ale reţelei neuronale pentru a

BUPT

permite existenţa unuia sau mai multor neuroni care să fie responsabili cu detectarea exemplelor negative. Aceşti neuroni trebuie să fie activi atunci când la intrarea reţelei este prezent un exemplu negativ.

- Cum **influenţează funcţiile de activarea modificarea procesului de antrenare** cu exemple negative? Pentru aceasta sunt cercetate o serie de funcţii de antrenare la care se vor modifica diverşi parametri. Rezultatele obţinute vor fi apoi comparate şi se va încerca o identificare clară a elementelor ce îmbunătăţesc performanţa antrenării reţelelor.
- **Metode de accelerare a antrenării unei RNA** utilizând tehnici de data mining, şi asta deoarece foarte multe date care sunt folosite în etapa de antrenare prezintă foarte multe recurenţe. O mai bună alegere a datelor de intrare **simplifică procesul de antrenare şi aduc un plus de valoare** prin faptul că numai caracteristicile cu adevărat importante sunt luate în considerare.

Fiecare dintre aceste metode va fi iniţial descrisă teoretic şi se va încerca verificarea corectitudinii ei printr-o serie de experimente.

1.4. Organizarea tezei

Teza este alcătuită din 7 capitole şi are următoarea structură:

După acest prim capitol introductiv, urmează **Capitolul 2**, care conţine aspecte teoretice legate de principalele arhitecturi de reţele neuronale. Este făcută o trecere în revistă asupra celor mai importante reţele neuronale utilizate în prezent şi sunt enunţate proprietăţile şi domeniile în care acestea excelează. Pentru rezolvarea unei probleme, o reţea neuronală trebuie să *înveţe,* proces care apare în urma antrenării cu date specifice fiecărei probleme.

În **Capitolul 3** sunt prezentate câteva din paradigmele de învăţare utilizate pentru antrenarea unei reţele neuronale. Sunt prezentaţi în detaliu „Algoritmul de învăţare pe baza corecţiei erorii" şi „Algoritmul de învăţare Hebbiană" deoarece sunt utilizaţi preponderent în cursul acestei lucrări. Tot aici va apare şi prima descriere pentru noua tehnică de antrenare dezvoltată în această teză. Ca ultim subcapitol sunt enunţate principalele funcţii de activare şi o împărţire a acestora pe categorii după felul ieşirii şi în funcţie de domeniul de activitate în care sunt folosite.

Prezentarea noilor metode de utilizare a exemplelor negative în procesul de antrenare este realizată în **Capitolul 4**. Capitolul începe cu o descriere a stadiului actual al domeniului antrenării reţelelor neuronale cu exemple negative şi modalităţile alese de diverşi autori pentru soluţionarea probleme apărute. În continuare sunt prezentate trei noi metode de folosire a exemplelor negative în procesul de antrenare, care vor conduce în final la formarea unei tehnici de antrenare a RNA utilizând exemple negative.

Punctul de plecare pentru metodele nou analizate este legat de problemele antrenării unei reţele neuronale având la dispoziţie un set de antrenare restrâns. S-a arătat astfel că în astfel de cazuri, cele mai multe dintre ele fiind speciale, raţiunile care au condus la micimea setului de antrenare ţin de condiţiile de aplicare, securitate sau resurse financiare.

O primă metodă este aceea în care procentul de exemple negative este folosit pe baza unui model matematic pentru a obţine performanţe maxime la utilizarea reţelei. Am arătat astfel că antrenarea folosind un set mixt de exemple

BUPT

conferă robusteţe reţelei împiedicând transformarea acesteia într-o reţea super confidentă şi mult mai predispusă la erori.

A doua metodă accentuează şi mai mult asupra folosirii exemplelor negative în procesul de antrenare şi propune o modificare asupra configuraţiei reţelei pentru a reuşi o mai bună recunoaştere a tiparelor negative.

Modificarea presupune adăugarea unui neuron suplimentar pe ultimul strat. Acesta este numit *neuron block* şi este direct responsabil de detectarea exemplelor negative; în acest caz realizându-se o îmbunătăţire majoră a modului de utilizare a reţelei neuronale.

Metoda această este propusă pentru a deveni un nou algoritm de învăţare. Ea se bazează pe informaţiile relevate de prima metodă şi are ca extensie ultima tehnică prezentată.

Ultima tehnică este cea care încearcă folosirea mai multor neuroni **block** pe stratul de ieşire fiecare dintre aceştia având funcţii de activare diferite. Se încearcă în acest fel o rezolvare a problemelor care apar la antrenarea unei reţele cu un set neomogen de exemple.

Felul în care a fost implementată fiecare metodă şi rezultatele obţinute de aceasta se găsesc în **Capitolul 5**. Acest capitol este dedicat în întregime experimentelor efectuate, şi trebuie spus că sunt foarte multe. Scopul fiecărui experiment a fost acela de a obţine o reuşită în validarea uneia dintre cele trei noi metode şi a performanţelor obţinute de aceasta. Şi el, la rândul lui, este împărţit în mai multe subcapitole, câte unul dedicat fiecărei metode propuse. Trebuie menţionat şi primul subcapitol care se ocupă de prezentarea programelor şi a seturilor de date folosite în cadrul experimentelor.

De asemenea acest capitol este responsabil şi de apariţia capitolului următor, **Capitolul 6**. Deoarece timpul dedicat experimentelor a fost de mai multe luni (timp în care un calculator cu procesor Quad Core a funcţionat permanent antrenând simultan câte 5 reţele) a apărut ideea reducerii acestui timp prin folosirea de caracteristici ale datelor de intrare la intrarea reţelei.

Este astfel prezentată o metodă de accelerare a procesului de antrenare, prin folosirea tehnicilor de data mining, responsabilă de selecţia caracteristicilor datelor de intrare şi de folosirea acestor caracteristici pentru antrenare.

Performanţele obţinute recomandă cu succes folosirea acestei metode, în special pentru probleme cu un număr foarte mare de intrări şi un set larg de date de intrare.

Toate concluziile rezultate în urma acestei teze şi posibilităţile viitoare de dezvoltare a tehnicii prezentate şi a altora noi, sunt prezentate în **Capitolul 7**. Acesta rezumă şi accentuează principalele idei care s-au desprins în urma acestei teze.

Teza se încheie cu o bibliografie aferentă, care a făcut posibil acest studiu, urmată de câteva anexe.

BUPT

2. ARHITECTURI ALE REŢELELOR NEURONALE ARTIFICIALE

2.1. Reţele neuronale de tip perceptron

Sunt reţelele neuronale cele mai cunoscute datorită capacităţii acestora de generalizare adică de a opera cu date diferite de cele prezentate în etapa de antrenament. Sunt reţele în care propagarea datelor este făcută de la intrare la ieşire (*feed forward*), caracteristică care este aplicată prin faptul că un neuron poate avea ca intrări numai ieşirile neuronilor de pe straturile precedente. Vom începe analiza noastră plecând de la o reţea neuronală simplă cu un singur strat, *perceptronul simplu*, şi vom continua analiza până la reţelele neuronale multistrat cu propagare înainte (multi-layer feed forward networks).

2.1.1. Perceptronul cu un singur strat

Este o reţea simplă care a reuşit să stârnească interesul datorită capacităţii de recunoaştere a unor tipare simple. Prezentăm în Figura 4 o astfel de reţea având *N* intrări şi o singură ieşire.

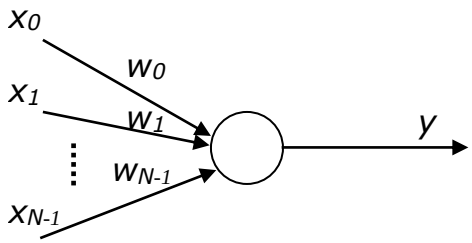


Figura 4. Perceptronul cu un singur strat

Ieşirea reţelei este calculată după următoarea formulă:

$$y = f_h\left(\sum_{i=0}^{N-1} w_i x_i - \theta\right) \tag{2.1}$$

Termenul θ poartă denumirea de *bias* şi poate fi orice funcţie sau constantă. Vom vedea mai târziu importanţa acestui termen. Funcţia de activare f_h poate fi orice funcţie liniară (în acest caz avem o reţea liniară) sau neliniară. Pentru perceptronul simplu considerăm că f_h este funcţia treaptă.

BUPT

$$f_h(s) = \begin{cases} 1 \text{ daca } s \geq 0 \\ -1 \text{ daca } s < 0 \end{cases} \quad (2.2)$$

Această reţea clasifică într-una din cele două clase disponibile (A sau B) un set de intrări. Astfel dacă ieşirea *y* este 1 atunci avem clasa A, în timp ce pentru ieşirea -1 avem clasa B. Altfel spus reţeaua reuşeşte divizarea spaţiului intrărilor în două regiuni separate de un hiperplan.

Putem observa acest lucru considerând că avem 2 intrări. În acest caz reţeaua separă spaţiul intrărilor în două după o linie ce are următoarea ecuaţie:

$$w_1 x_1 + w_2 x_2 + \theta = 0 \quad (2.3)$$

Transformând ecuaţia (2.3) obţinem reprezentarea geometrică a liniei ce separă cele două regiuni :

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{\theta}{w_2} \quad (2.4)$$

Observăm cum toate intrările care sunt deasupra liniei aparţin clasei A în timp ce intrările de sub linie aparţin clasei B (Figura 5).

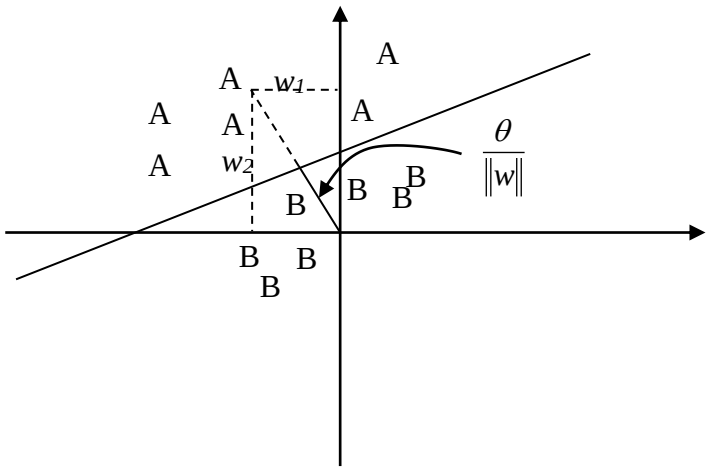


Figura 5. Reprezentarea geometrică a liniar separabilităţii spaţiului de intrare în două regiuni

Trebuie spus că ponderile determină panta liniei în timp ce termenul *bias* determină ofsetul (la ce distanţă de origine este linia). Găsirea termenului *bias* şi a ponderilor reţelei este posibilă în urma aplicării algoritmului de învăţare asupra reţelei.

Ca şi limitare a acestor reţele putem aminti problematica XOR (sau-exclusiv) sau orice tip de problema al cărui strat de intrare nu poate fi separabil în două regiuni.

BUPT

2.1.2. Perceptronul cu mai multe straturi

Reţelele de tip perceptron cu mai multe straturi (multi-layer perceptron) sunt reţele de tip propagare înainte cu unul sau mai multe straturi situate între nodurile de intrare şi nodurile de ieşire. Aceste straturi (straturi ascunse) conţin neuroni care nu sunt direct conectaţi cu intrările sau ieşirile reţelei. Toate limitările impuse de perceptronul cu un singur strat sunt depăşite de acest tip de reţea.

Performanţele unei reţele de tipul perceptron cu mai multe straturi constau în folosirea unor funcţii de activare neliniare pentru nodurile interne. Dacă am folosi numai funcţii de activare liniară putem aproxima o reţea neuronală de tip perceptron cu mai multe straturi cu o reţea de tip perceptron cu un singur strat.

Ca şi arhitectură, perceptronul cu mai multe straturi se prezintă ca în Figura 6.

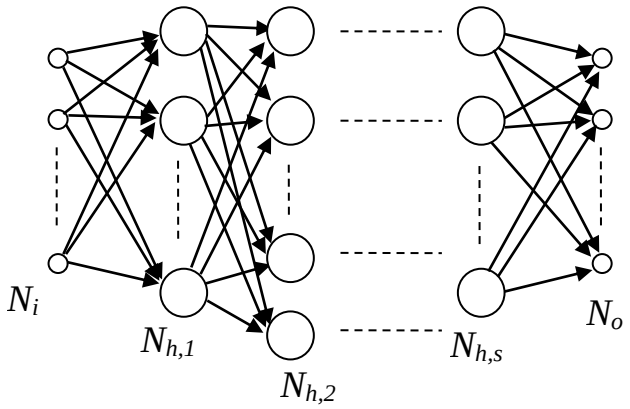


Figura 6. Perceptronul multistrat cu s straturi ascunse

Primul strat N_i este stratul de intrare. Neuronii de pe acest strat sunt lipsiţi de funcţia de activare, adică lasă să treacă orice valoare primită la intrare. Următoarele *s* straturi ($N_{h,1}$, .., $N_{h,s}$) se numesc straturi ascunse. Fiecare strat ascuns ($N_{h,l)}$ primeşte valori la intrare numai de la stratul anterior lui ($N_{h,l-1}$), ieşirile lui constituindu-se ca intrări pentru stratul următor ($N_{h,l+1}$). Nu sunt permise conexiuni în interiorul stratului. Ieşirile ultimului strat ascuns $N_{h,s}$ sunt intrări pentru stratul de ieşire N_o. Trebuie spus că sunt permise şi conexiuni de genul *skip layer* (astfel neuronii pot primi ca şi intrări, ieşirile altor neuroni care nu se află pe un strat imediat precedent lor). Conexiunile directe între stratul de intrare şi cel de ieşire sunt în special foarte folositoare.

Numărul de neuroni de pe straturile ascunse poate fi diferit, de la strat la strat. Fiecare dintre aceşti neuroni are o funcţie de activare F_i care acţionează asupra intrărilor şi *bias*-ului.

Minsky şi Papert[16] au arătat în 1969 că o reţea neuronală de tip perceptron multistrat cu 1 strat ascuns, rezolvă multe dintre problemele rămase nerezolvate de perceptronul cu un singur strat, dar nu au putut prezenta o soluţie la modul de ajustare a ponderilor în această reţea. Soluţia la această problemă a venit în 1986, când Rumelhart, Hinton şi Williams [17] au determinat modul de ajustare al ponderilor propagând eroarea de la stratul de ieşire spre stratul de intrare. Această tehnică s-a şi numit *back-propagation*. Deşi tehnica *back-propagation* poate

BUPT

fi aplicată la reţele de tip perceptron cu un număr oarecare de straturi ascunse, s-a demonstrat ([18], [19], [20], [21]), că pentru reţele cu intrări binare, un singur strat ascuns este suficient pentru aproximarea oricărei funcţii ce are un număr finit de discontinuităţi (*Teorema aproximării universale*). În cele mai multe aplicaţii este utilizată funcţia sigmoid ca şi funcţie de activare a neuronilor.

Tehnica *back-propagation* implică 2 faze:

- Prima fază – avem o intrare *x* pe care o aplicăm la intrarea reţelei. Valorile se propagă înainte trecând din strat în strat şi ajungând până la stratul de ieşire. Comparăm valorile prezentate în stratul de ieşire al reţelei cu valorile dorite rezultând o valoare de eroare.
- Faza a doua – valorile de eroare sunt trecute invers prin reţea, de la ieşire la intrare, având ca scop modificarea ponderilor neuronilor de pe fiecare strat al reţelei.

Aşa cum am spus şi mai sus un singur strat ascuns e suficient pentru rezolvarea majorităţii problemelor. Putem folosi însă şi două sau trei straturi ascunse pentru probleme dificile. Dacă numărul neuronilor de pe straturile de intrare şi ieşire este uşor determinabil (ţine exclusiv de natura aplicaţiei), numărul neuronilor de pe stratul ascuns este dificil de aproximat.

Neuronii de pe straturile ascunse au rol în detectarea trăsăturilor comune pentru tiparele de antrenament. De aceea un număr prea mic de neuroni va influenţa negativ capacitatea de generalizare a reţelei conducând la o eroare medie pătratică mare. Un număr mare de neuroni conduce la un volum de calcul mare şi măreşte exponenţial timpul de antrenament.

Ca şi concluzie putem nota faptul că în prezent alegerea numărului de neuroni de pe stratul ascuns este făcută experimental.

2.2. Reţele neuronale bazate pe funcţii radiale

Sunt la fel de cunoscute ca şi reţelele de tip perceptron multistrat datorită faptului că au multe lucruri în comun cu acestea. Diferenţa majoră între aceste două reţele este modul de calculare al valorii de intrare aplicată funcţiei de activare pentru un neuron.

Astfel, reţelele de tip perceptron calculează intrarea funcţiei de activare ca fiind suma produselor dintre fiecare intrare a neuronului şi ponderea asociată acestei intrări plus o valoare de *bias* (vezi 2.1).

Reţelele neuronale bazate pe funcţii radiale (RBF – Radial basis function) au, de obicei, un singur strat ascuns pentru care funcţia de activare a unui neuron este calculată ca fiind egală cu distanţa euclidiană dintre vectorul intrărilor şi vectorul ponderilor.

RBF nu au ceva asemănător cu termenul *bias* existent în reţelele de tip perceptron. Există însă câteva tipuri de RBF care au un termen numit grosime (*width*) asociat fiecărui neuron sau întregului strat ascuns, dar în loc să-l scădem aşa cum apare el în reţelele de tip perceptron, înmulţim distanţa euclidiană cu acest termen (2.5).

$$y = f_h\left(w * \|t - x\|\right) \tag{2.5}$$

f_h este ceea ce numim o funcţie radială.

BUPT

Caracteristica principală a acestor tipuri de funcţii este aceea că valoarea lor creşte(descreşte) monoton cu creşterea distanţei faţă de un punct central. Elementele de bază ale unei funcţii radiale sunt: punctul central, domeniul de valori şi forma precisă a graficului[22].

Exemplu: funcţia Gaussiană (2.6) al cărei grafic îl putem observa în Figura 7

$$f_h(x) = exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{2.6}$$

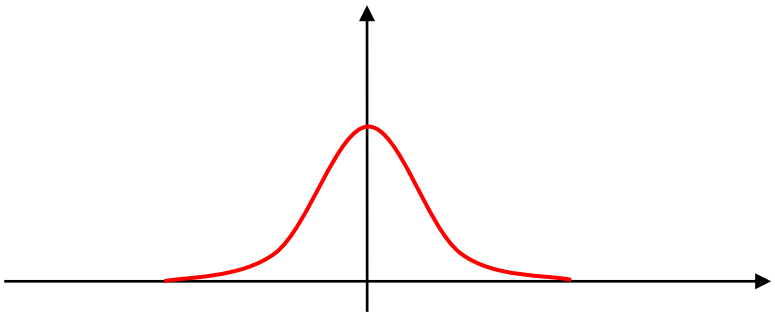


Figura 7. Graficul funcţiei Gaussiene

Se poate observa din Figura 7 cum creşterea distanţei faţă de un punct central (în acest caz fiind vorba de 0), descreşte valoarea funcţiei. Deoarece RBF bazate pe funcţia Gaussiană sunt cele mai răspândite, trebuie să spunem despre ele că se împart în două tipuri[Shorten96].

Primul tip utilizează funcţia de activare de tip exponenţial, ceea ce face ca activarea unui neuron să fie un „cucui" Gaussian(Figura 7) ce depinde de intrări. Acest tip de RBF poartă numele de „RBF ordinare".

La al doilea tip de RBF funcţia de activare utilizează suma tuturor funcţiilor de activare de pe stratul ascuns, ceea ce face ca toate funcţiile de activare să fie normalizate la o sumă (2.7). Reţelele acestea se numesc RBF normalizate.

$$f_h(x) = \frac{f_{h,k}\left(\left\|x - c^k\right\|\right)}{\sum_{l=1}^{k} f_{h,l}\left(\left\|x - c^l\right\|\right)} \tag{2.7}$$

Prezentăm în Figura 8 modelul clasic al unei reţele de tip RBF.

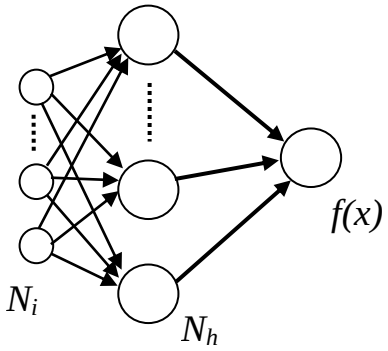


Figura 8. RBF clasică

Aceasta este compusă din *n* neuroni de intrare (aflaţi în componenţa stratului N_i), *m* neuroni din stratul ascuns (care au ca şi funcţii de activare, funcţii radiale $f_h(x)$), ieşirile celor *m* neuroni fiind combinate liniar în neuronul de ieşire ($f(x)$ – funcţie liniară).

Dacă avem mai mult de un strat ascuns sau dacă funcţiile radiale îşi schimbă forma sau punctul central, putem spune că avem o RBF neliniară.

Învăţarea unei RBF este văzută ca o aproximare a unei suprafeţe dintr-un spaţiu multidimensional. Funcţia de transfer radială are un caracter de localitate, ceea ce face ca neuronii să producă ieşiri semnificative doar pentru regiuni relativ mici din spaţiul de intrare.

O bună comportare a reţelei se obţine atunci când datele de intrare acoperă câmpurile receptive ale neuronilor din stratul ascuns, ceea ce presupune un număr mare de vectori de antrenament.

2.3. Memorii asociative

O memorie adresabilă prin conţinut (content-adressable memory) este un tip de memorie ce permite regăsirea unor date bazându-se pe similarităţile existente între tiparele stocate în memorie şi un tipar prezentat la intrare. Această memorie (adresabilă prin conţinutul ei) este diferită de memoria "convenţională"(adresabilă prin adresă) care pe baza unei intrări(adresă) prezintă la ieşire o anumită dată. Memoria adresabilă prin conţinut primeşte ca intrare un tipar incomplet (contaminat) şi reuşeşte regăsirea acelui tipar.

Exemplu: Presupunem că avem o memorie în care am stocat numele câtorva cursuri. Prezentăm ca intrare un nume de curs greşit "Inteligenţ Artficală". Memoria ne va returna "Inteligenţa Artificială" , care este numele corect al cursului.(Figura 9)

BUPT

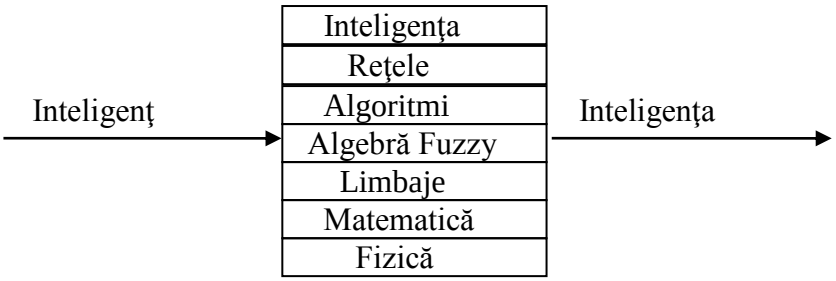


Figura 9. Modul în care acţionează o memorie adresabilă prin conţinut

O memorie asociativă este o structură adresabilă prin conţinut care reuşeşte asocierea a doua tipare X şi Y, astfel încât având unul din tipare putem să-l regăsim pe celălalt tipar. Memoriile asociative se împart în două clase: autoasociative şi heteroasociative. O memorie autoasociativă este utilizată în regăsirea unui tipar care a fost stocat anterior în memorie şi care este foarte apropiat de tiparul curent (X=Y in acest caz). La memoriile heteroasociative tiparul regăsit este în general diferit de tiparul de la intrare, diferenţă care poate fi prin tip, format sau conţinut.

Prezentam in continuare câteva reţele neuronale care sunt utilizate ca şi memorii asociative.

2.3.1. Asocierea liniară

Este cel mai simplu model de memorie asociativă. Fiind cel mai simplu este şi primul pe care-l vom studia. Arhitectura reţelei neuronale care implementează acest model este prezentată în Figura 10.

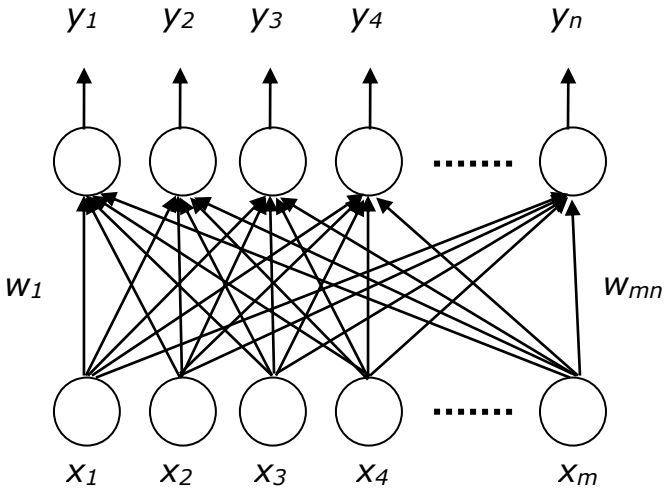


Figura 10. Arhitectura unei reţele de tip asociere liniară

Este o reţea de tip *feed forward* la care ieşirea este produsă într-un singur ciclu de calcul. Se poate observa din figură că toate cele *m* intrări sunt conectate cu

BUPT

toate cele *n* ieşiri. Memorarea celor *p* tipare este făcută cu ajutorul matricei ponderilor sinaptice $W = [w_{ij}]_{m \times n}$, unde w_{ij} reprezintă ponderea legăturii dintre nodul de intrare *i* şi nodul de ieşire *j*.

Procesul de construcţie al memoriei asociative se numeşte *codare* şi el este făcut după formula:

$$W = a \sum_{k=1}^{p} W_k \tag{2.8}$$

in care:

- a este o constantă de normalitate. Este setată de obicei cu valoarea 1/p pentru prevenirea apariţiilor unor valori prea mari pentru ponderile w_{ij} .
- W_k numită şi matrice de corelare, este matricea ponderilor sinaptice obţinută la aplicarea unui tipar *k*. Fiecare element al acestei matrice este de forma $(w_{ij})_k$ şi este calculat astfel :

$$\left(w_{ij}\right)_k = \left(x_i\right)_k \left(y_j\right)_k , \; i = 1 \ldots m, \; j = 1 \ldots n \tag{2.9}$$

unde $(x_i)_k$ este elementul *i* al tiparului x_k aplicat la intrare, iar $(y_j)_k$ este elementul *j* al tiparului y_k dorit la ieşire.

Matricea ponderilor sinaptice permite astfel stocarea/regăsirea a *p* perechi de tipare.

Regăsirea informaţiei este făcută printr-un proces numit *decodare*. Astfel dacă reţelei îi oferim la intrare un tipar *X*, la ieşire obţinem un tipar *Y*, calculat după următoarea formulă:

$$y_j = \begin{cases} +1 & dacă \quad \sum_{i=1}^{m} x_i w_{ij} \geq \varphi_j \\ -1 & altfel \end{cases} \tag{2.10}$$

unde : x_i este elementul *i* al vectorului *X*, y_j este elementul *j* al vectorului *Y*, φ_j este valoarea de prag pentru neuronul de ieşire *j*.

Chiar dacă tiparul de intrare conţine erori sau este incomplet, reţeaua va returna tiparul care este cel mai apropiat de acesta. Această reţea este robustă şi tolerantă la erori. Performanţa acestei memorii este măsurată prin *capacitatea de memorare* şi prin *adresabilitatea conţinutului. Capacitatea de memorare* se referă la numărul maxim de perechi care poate fi stocat şi regăsit corect, în timp ce *adresabilitatea conţinutului* este abilitatea reţelei de a regăsi tiparul corect.

2.3.2. Reţelele Hopfield

Modelul Hopfield a fost propus de John Hopfield de la Institutul de Tehnologie din California la începutul anilor 1980.

BUPT

Diferenţa dintre modelul de *asociere liniară* şi modelul *Hopfield* constă în faptul că acesta din urmă intră în categoria reţelelor neuronale de tip *feedback*. Ieşirile neuronilor sunt intrări pentru ceilalţi neuroni. Sistemul efectuează mai multe cicluri de calcul până când sistemul devine stabil. Prezentăm în Figura 11 o reţea de tip Hopfield cu 5 neuroni.

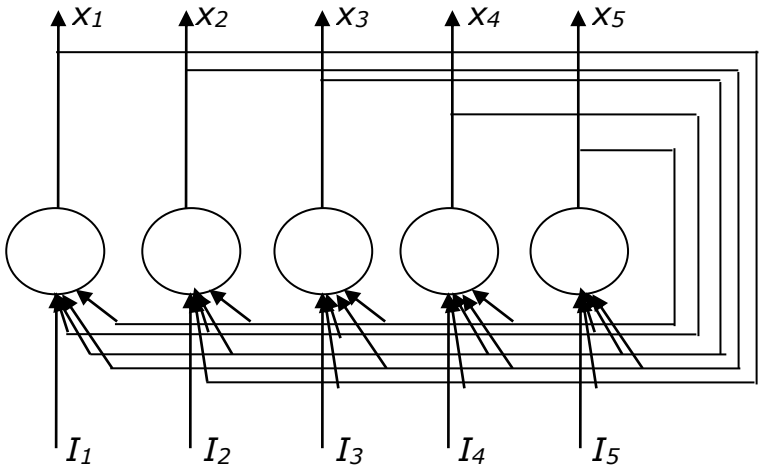


Figura 11. Arhitectura unei reţele de tip Hopfield

Spre deosebire de modelul *asociere liniară* care constă din 2 straturi de procesare (intrare şi ieşire), modelul *Hopfield* constă dintr-un singur strat în care fiecare neuron este conectat cu restul neuronilor.

Matricea ponderilor sinaptice pentru modelul Hopfield este pătratică şi simetrică, adică $w_{ij} = w_{ji}$, cu $i,j = 1,2,\ldots,m$. Fiecare ciclu de calcul modifică intrarea unui nod după următoarea formulă:

$$input_j = \sum_{i=1}^{m} x_i w_{ij} + I_j \qquad (2.11)$$

în care:

- $input_j$ - intrarea nodului j
- x_j - ieşirea nodului j
- w_{ij} - element din matricea ponderilor sinaptice ($w_{ii} = 0$)
- I_j - intrare suplimentară

Introducerea unui tipar în memoria asociativă este realizată ca şi la asocierea liniară prin calcularea elementelor matricei W.

$$W_k = X_k^T Y_k \qquad (2.12)$$

unde $X_K=Y_K$.

BUPT

$$W = a\sum_{k=1}^{p} W_k \tag{2.13}$$

Ultima formulă (2.13) este aplicată pentru memorarea a *p* tipare în memoria asociativă. Hopfield (1982) a demonstrat că reţeaua care îi poartă numele, poate stoca maxim 0.15m (m - numărul de noduri care compun reţeaua). [13]

Faza de memorare fiind terminată, reţeaua este gata pentru faza de decodare. Tiparul de intrare este introdus în reţea cu ajutorul intrărilor suplimentare I_j. La ieşirea reţelei va apărea astfel un tipar *X'* care datorită buclei de reacţie (*feedback)* va fi trimis către intrările reţelei care va genera un nou tipar de ieşire *X''*. Procesul va continua până când tiparul de la ieşire nu se va mai modifica, astfel spus reţeaua stabilizează tiparul astfel încât asupra lui nu se mai produc nici un fel de modificări.

Dacă tiparul introdus este un tipar incomplet, reţeaua va stabiliza în final unul dintre tiparele memorate care seamănă cel mai bine cu tiparul de la intrare. Aceasta este una din trăsăturile reţelei şi poartă numele de *regăsirea tiparelor* (pattern completion), fiind foarte utilă în aplicaţii de procesare a imaginilor.

Pentru a calcula ieşirea nodurilor există 3 metode: sincronă(metoda paralelă), asincronă(metoda secvenţială) sau hibridă(o combinaţie între cele două).

Utilizarea metodei sincrone presupune modificarea ieşirilor reţelei în acelaşi timp, ieşirile fiind văzute ca un grup care trebuie să fie complet înainte ca acesta să alimenteze intrările reţelei. În metoda asincronă ieşirea nodurilor este calculată într-o oarecare ordine(secvenţial sau aleator) şi este trimisă nodurilor separat; cum terminăm de calculat o ieşire o şi trimitem să alimenteze intrările reţelei. Metoda hibridă împleteşte cele două metode în felul următor: se creează diverse subgrupuri care sunt calculate asincron, însă toate nodurile care aparţin unui subgrup sunt calculate sincron. Alegerea unei anumite metode are efect asupra convergenţei reţelei.

Reţelele neuronale recurente au asociată o funcţie de energie. Aceasta este folosită pentru demonstrarea stabilităţii acestui tip de reţele.

Reţele Hopfield discrete

Ieşirea unei astfel de reţele este calculată după următoarea formulă:

$$x_i(t+1) = \begin{cases} +1 & \text{dacă } input_i > \theta_i \\ x_i(t) & \text{dacă } input_i = \theta_i \\ -1 & \text{dacă } input_i < \theta_i \end{cases} \tag{2.14}$$

unde i=1,2,...,m iar t reprezintă momente discrete de timp.
Funcţia de energie pentru modelul Hopfield discret este:

$$E = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} x_i w_{ij} x_j - \sum_{i=1}^{m} x_i I_i + \sum_{i=1}^{m} x_i \theta_i \tag{2.15}$$

BUPT

Minimul local al acestei funcţii corespunde stării de stabilitate pentru un tipar memorat. Hopfield a demonstrat că energia unei astfel de reţea poate să scadă sau să rămână constantă, cu alte cuvinte e posibil ca reţeaua să conveargă către un minim energetic local.

Energia unei reţele Hopfield discrete e limitată inferior la:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{m}\left|w_{ij}\right| - \sum_{i=1}^{m}\left|I_i\right| + \sum_{i=1}^{m}\left|\theta_i\right| \tag{2.16}$$

pentru toate X_K,k=1,2,...,p. Deoarece energia este limitată inferior, reţeaua va converge eventual către un minim local care corespunde unui tipar memorat.

Reţele Hopfield continue

Reţeaua Hopfield continuă este generalizarea reţelei Hopfield discrete. Unităţile acestei reţele modelează neuronul biologic prin aceea că au asociate câte o capacitate C_i şi o rezistenţă r_i. Această capacitate şi rezistenţă corespunde membranei celulare a neuronului. Ecuaţia de stare a reţelei devine astfel:

$$C_j\frac{dinput_j}{dt} = \sum_{i=1}^{m} x_i w_{ij} - \frac{input_j}{R_j} + I_j \tag{2.17}$$

unde:

$$\frac{1}{R_j} = \frac{1}{\rho_j} + \sum_{i=1}^{m} w_{ij} \tag{2.18}$$

Funcţia de energie corespunzătoare acestui model este următoarea:

$$E = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} x_i w_{ij} x_j - \sum_{i=1}^{m} x_i I_i + \sum_{i=1}^{m}\left(\frac{1}{R_i}\right)\int_0^{x_i} f^{-1}(x)\,dx \tag{2.19}$$

unde f este ieşirea unui nod din reţea.

Atunci când $w_{ij} = w_{ji}$ obţinem $\frac{dE}{dt} \neq 0$. Energia acestui model poate să scadă sau să rămână constantă ca şi la modelul discret.

2.3.3. Memorii asociative bidirecţionale (BAM)

În 1988 Bart Kosko [23] propune o extindere a Modelulu Hopfield prin încorporarea unui strat adiţional care să uşureze implementarea unor modele de memorie autoasociative şi heteroasociative.

Pentru aceasta, Kosko a plecat de la modelul de asociere liniară la care a făcut conexiunile dintre straturi să fie bidirecţionale şi w_{ij}=w_{ji} pentru orice i=1,2,...,m şi j=1,2,...,n. (Figura 12)

BUPT

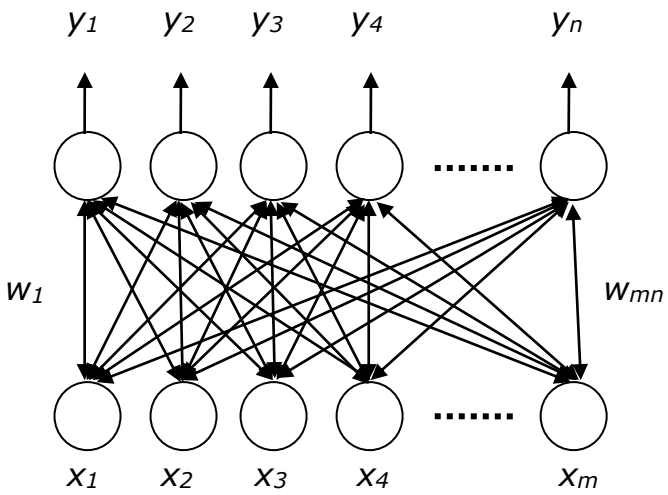


Figura 12. Arhitectura unei reţele de tip Memorie Asociativă Bidirecţională

În funcţie de direcţia de propagare, ambele straturi pot să fie de intrare sau de ieşire. Astfel, dacă propagarea semnalului se face de la stratul X la stratul Y, neuronii din stratul X sunt văzuţi ca neuroni de intrare în timp ce neuronii din stratul Y sunt văzuţi ca neuroni de ieşire. Dacă propagarea semnalului se face de la stratul Y la stratul X, atunci neuronii din stratul Y sunt văzuţi ca neuroni de intrare, în timp ce neuronii de pe stratul X sunt neuroni de ieşire.

Ca şi la asocierea liniară şi la modelul Hopfield, introducerea unei perechi de tipare asociative în memoria asociativă bidirecţională este realizată prin calcularea elementelor matricei *W*.

$$W_k = X_k^T Y_k \tag{2.20}$$

şi

$$W = a\sum_{k=1}^{p} W_k \tag{2.21}$$

Ultima formulă (2.21) este aplicată pentru memorarea a *p* tipare în memoria asociativă bidirecţională.

În procesul de decodare putem introduce tiparul dorit la stratul X sau la stratul Y. Tiparul este propagat către celălalt strat, care va genera un nou tipar ce va fi intrare în stratul iniţial. Procesul continuă până când informaţia din ambele straturi nu se mai modifică. În acest caz una dintre perechile stocate în memoria asociativă este prezentă în straturile reţelei. La fel ca şi la memoria Hopfield, ieşirile celor două straturi pot fi calculate sincron, asincron sau printr-o metodă hibridă.

BUPT

Memorii asociative bidirecţionale discrete

Într-o memorie asociativă bidirecţională discretă stratul X este considerat strat de intrare; în acest caz un tipar de intrare este propagat de la stratul X la stratul Y. Unităţile din stratul Y îşi vor calcula intrarea după următoarea formulă:

$$inputy_j = \sum_{i=1}^{m} x_i w_{ij} \tag{2.22}$$

şi vor avea ca ieşire:

$$y_i(t+1) = \begin{cases} +1 & dacă\ inputy_i > 0 \\ y_i(t) & dacă\ inputy_i = 0 \quad pentru \quad j = 1,2,\text{K}\ ,n \\ -1 & dacă\ inputy_i < 0 \end{cases} \tag{2.23}$$

Stratul Y va produce un tipar care va fi propagat în spate tiparului X; unităţile din stratul X îşi vor calcula astfel intrările:

$$inputx_j = \sum_{i=1}^{n} y_i w_{ij} \tag{2.24}$$

şi ieşirile stratului X:

$$x_i(t+1) = \begin{cases} +1 & dacă\ inputx_i > 0 \\ x_i(t) & dacă\ inputx_i = 0 \\ -1 & dacă\ inputx_i < 0 \end{cases} \tag{2.25}$$

Funcţia de energie ce caracterizează memoriile BAM este calculată:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} x_i w_{ij} y_j \tag{2.26}$$

Kosko a arătat în 1988 că această energie descreşte sau rămâne constantă după fiecare iteraţie. Totodată, reţeaua converge către un minim local ce corespunde cu o pereche de tipare memorate. Energia unei BAM este limitată inferior de valoarea:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} \left| w_{ij} \right| \tag{2.27}$$

BUPT

Memorii asociative bidirecţionale continue

Unităţile din stratul X vor avea o intrare suplimentară I_i iar unităţile din stratul Y vor avea şi ele o intrare suplimentară J_j, cu i=1,2,...,m iar j=1,2,...,n. Aceste intrări suplimentare vor modifica intrările în cele două straturi astfel:

$$input x_i = \sum_{j=1}^{n} y_j w_{ij} + I_i \tag{2.28}$$

şi

$$input y_j = \sum_{i=1}^{m} x_i w_{ij} + J_j \tag{2.29}$$

Energia unei astfel de reţea este:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} f(x_i) f(y_j) w_{ij} - \sum_{i=1}^{m} f(x_i) I_i - \sum_{j=1}^{n} f(y_j) J_j + \sum_{i=1}^{m} \int_{o}^{x_i} f'(x_i) x_i dx_i + \sum_{j=1}^{n} \int_{0}^{y_j} f'(y_j) y_j dy_j \tag{2.30}$$

2.4. Reţele neuronale cu autoorganizare

Există numeroase situaţii când nu există nici o informaţie referitoare la intrările şi ieşirile unei reţele neuronale. Reţeaua însăşi trebuie să descopere tipare, trăsături, regularităţi şi corelaţii existente în datele de intrare precum şi modul de codare sau reprezentare internă al acestora, pentru a-l furniza ieşirilor.[24]

Ca şi exemple de aplicaţie pentru acest tip de reţele, putem defini următoarele:

- Avem un set de intrări pe care dorim să-l grupăm în mai multe mulţimi. Pentru aceasta, reţeaua trebuie să găsească aceste mulţimi, adică numărul şi tipul de elemente specifice fiecărei mulţimi. Ieşirea reţelei trebuie să indice mulţimea de care aparţine o intrare dată.
- Reducerea dimensiunii: reţeaua trebuie să găsească o funcţie care să permită reducerea dimensiunii datelor de intrare fără a pierde posibile variaţii ale acestora. Astfel, datele de intrare sunt grupate în submulţimi, care au o dimensiune mai mică decât cele originale.
- Cuantizare vectorială: trebuie găsită o discretizare optimă între datele de intrare (considerate un spaţiu continuu) şi datele de ieşire (reprezentarea discretă).

BUPT

Există multe reţele neuronale care se înscriu în această categorie. Ca şi trăsătură generale putem enunţa faptul că învăţarea este efectuată fără vreo supervizare exterioară, algoritmii folosiţi bazându-se de obicei, pe o formă de competiţie între neuroni.

2.4.1. Reţele neuronale de tip clustere

Aceste reţele au o arhitectură relativ simplă. Astfel, dacă la intrare avem date de dimensiune N, şi dorim împărţirea acestor date în K clase, avem următoarea arhitectură (Figura 13):

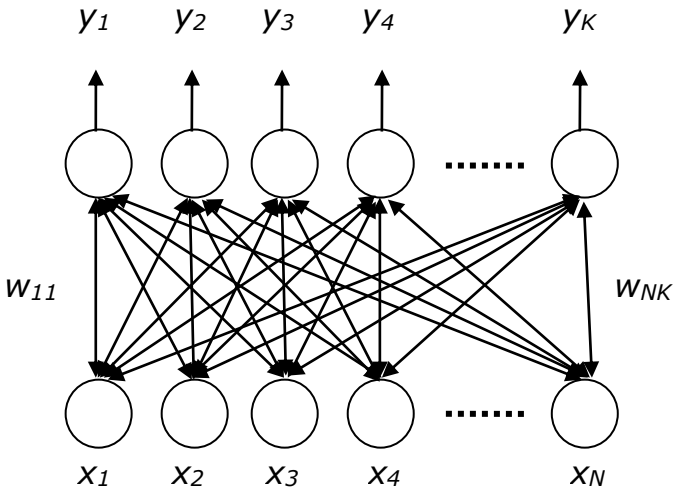


Figura 13. Arhitectura unei reţele de tip clustere

Toate nodurile de ieşire sunt conectate cu toate nodurile de intrare, ponderile conexiunilor putând fi organizate într-o matrice cu K linii şi N coloane. Dacă antrenăm corect reţeaua toate intrările care vor aparţine de un cluster vor activa un singur neuron de ieşire. Fiecare cluster va avea ca şi prototip linia neuronului activat.

Ca şi funcţionare a reţelei putem spune: dacă la intrare este prezent un vector de N elemente, la ieşire reţeaua va activa unul din cele K noduri. Nodul de ieşire activ va fi determinat într-un proces de competiţie care presupune calcularea pentru fiecare din vectorii prototip a unei masuri de similaritate cu intrarea curentă. Prototipul cu măsura de similaritate cea mai bună este cel care activează ieşirea reţelei.

Prezentăm un algoritm de tip WTA (winner take all) în care măsura de similaritate este distanţa euclidiană dintre 2 vectori.

- Se iniţializează prototipurile – completarea unei matrice W cu K linii şi N coloane cu valori aleatoare.
- Se ajustează iterativ prototipurile – pentru un vector de intrare dat se calculează distanţa euclidiană dintre vectorul intrare şi fiecare prototip.

BUPT

- Se alege distanţa euclidiană cea mai mică (măsura de similaritate cea mai bună) şi se ajustează vectorul prototip corespunzător acestei distanţe după formula:

$$W^k = W^k + \eta(t)(X^l - W^k) \tag{2.31}$$

Se poate observa din această formulă că vectorul care determină ieşirea învingătoare este mărit pentru ca detecţia acestuia să devină mai simplă.

2.4.2. Reţele neuronale de tip cuantizare vectorială

Aceste reţele împart mulţimea vectorilor de intrare în clase, în scopul reprezentării vectorului prin clasa asociată acestuia. Diferenţa dintre reţelele neuronale de tip cuantizare vectorială şi cele de tip clustere este aceea, că în timp ce reţelele de tip clustere sunt interesate în găsirea similarităţilor dintre mai multe date (pentru a reuşi formarea unui cluster), reţelele neuronale de tip cuantizare vectorială încearcă împărţirea întregului spaţiu de intrări.

Putem spune că prin cuantizare se asigură împărţirea unui domeniu al datelor de intrare în mai multe regiuni, fiecare regiune având ca reprezentant un vector prototip (codebook vector). În felul acesta fiecare vector dintr-o regiune poate fi asociat cu un scalar, asigurându-se astfel o compresie a datelor de intrare.

Ca şi potenţial pericol putem observa o pierdere a anumitor caracteristici ale datelor de intrare. Cuantizarea trebuie să ţină cont de acesta, asigurând acolo unde este cazul o partiţionare mai fină (rezultând astfel mai multe prototipuri în acea regiune).

Cuantizarea este folosită în aplicaţii în care se doreşte o compresie a datelor; de exemplu telecomunicaţiile sau stocarea de date. Descriem în continuare două metode de utilizare a reţelelor neuronale de tip cuantizare vectorială.

Counter propagation

Aceste reţele pot aproxima cel mai bine funcţiile reale de tipul:

$$f : R^n \rightarrow R^m \tag{2.32}$$

În Figura 14 este prezentată o reţea care combină un strat de tip cuantizare vectorială cu unul de tip reţea cu propagare înainte cu rol de aproximare funcţională.

Fiecare neuron *o* din stratul de ieşire are asociat o valoare a funcţiei f de forma $[w_{1o}, w_{2o}, \ldots, w_{ho}]^T$. Această valoare este reprezentativă pentru valorile $f(x)$, cu *x* situându-se în „intervalul acoperit" de către ieşirea *o*. Modalitatea de implementare a aproximării funcţiei este următoarea: este necesară o tabelă de căutare (*look-up table)* în care unei intrări *x* îi este asociată o valoare *k* din tabelă, astfel încât pentru orice altă valoare *o* din tabelă, să avem următoarea inegalitate:

$$\forall \quad o \neq k, \quad \|x - w_k\| \leq \|x - w_o\| \tag{2.33}$$

BUPT

ceea ce se traduce prin: fiecare intrare are asociată ieşirea corespunzătoare ei (care îi este cea mai *„apropiată”*), iar valoarea acestei ieşiri care este de forma $[w_{1k}, w_{2k}, \ldots, w_{hk}]^T$ este o aproximare a valorii funcţiei $f(x)$.

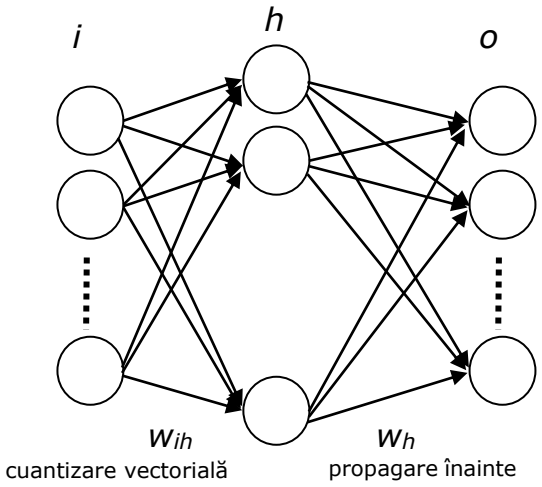


Figura 14. Arhitectura unei reţele de tip cuantizare vectorială

Cuantizarea vectorială poate fi făcută atât înainte cât şi în acelaşi timp cu învăţarea aproximării funcţiei. Folosind ca şi exemplu reţeaua din *Figura 6* putem superviza învăţarea ei urmând paşii:

- Alegem seturile (x , $f(x)$) cu care vom efectua antrenarea; observăm astfel că dorim cuantizarea funcţiei $f(x)$.
- Efectuăm cuantizarea vectorială (nesupervizat) astfel: pentru fiecare vector w_k , calculăm distanţa dintre el şi o intrare x şi alegând cea mai mică distanţă, determinăm câştigătorul k. Modificăm apoi ponderile astfel:

$$w_k(t+1) = w_k(t) + \eta\left(x(t) - w_k(t)\right) \tag{2.34}$$

- Efectuăm aproximarea funcţională (supervizat) astfel:

$$w_{ko}(t+1) = w_{ko}(t) + \eta\left(f(x) - w_{ko}(t)\right) \tag{2.35}$$

- Se observă că avem formula regulii delta în care

$$y_0 = \sum_h y_h w_{ho} = w_{ko} \tag{2.36}$$

- neuronul k este câştigător şi ieşirea dorită este dată de **d**=$f(x)$.

BUPT

Fiecare valoare din tabela de căutare reprezintă o *„medie"* a tuturor intrărilor din subspaţiul definit de respectiva valoare din tabelă.

Trebuie spus că această reţea nu reprezintă cu acurateţe orice tip de funcţie. Astfel dacă avem o combinaţie de funcţii sin şi cos o reţea de tipul multistrat cu un algoritm de tip back-propagation este indicat a fi folosită. Totuşi dacă intrările sunt un subspaţiu din R^n şi ne aşteptăm ca funcţia f să fie discontinuă în multe puncte, atunci combinaţia oferită (o reţea de cuantizare şi o reţea de aproximare) este foarte indicat de folosit. Oricare dintre cele 2 reţele poate să fie înlocuită cu una mai performantă; reţeaua de aproximare poate să fie de tipul multistrat, iar reţeaua de cuantizare poate să fie de tip Kohonen.

Învăţarea prin cuantizare vectorială (LVQ)

Este o tehnică de învăţare supervizată prin care se încearcă o delimitare mai fină a spaţiului de intrări furnizând în final mai multe clase (fiecare clasă conţine un spaţiu al intrărilor). Deşi în literatura de specialitate apar mai mulţi algoritmi LVQ (Learning Vector Quantization) toţi au la bază următorul algoritm:

- Fiecărui neuron de ieşire *o* îi este asociată o clasă y_o
- Fiecare set de antrenament este compus din două valori: vectorul de intrare x^p şi clasa corectă y_o^p
- Calculând distanţele dintre vectorul de intrare x^p şi vectorii pondere w_o putem determina atât cel mai bun vector k_1 cât şi următorul vector performant k_2. Astfel:

$$\left\|x^p - w_{k_1}\right\| < \left\|x^p - w_{k_2}\right\| < \left\|x^p - w_o\right\| \quad \forall o \neq k_1, k_2 \tag{2.37}$$

- Comparând clasele rezultate $y_{k_1}^p$ şi $y_{k_2}^p$ cu răspunsul d^p putem avea diverse criterii de modificare selectivă a ponderilor cu ajutorul ecuaţiei:

$$w_k(t+1) = w_k(t) + \eta\left(x(t) - w_k(t)\right) \tag{2.38}$$

Astfel, algoritmului LVQ2 creat de Kohonen foloseşte următoarea strategie de modificare a ponderilor: dacă $y_{k_1}^p \neq d^p$ şi $y_{k_2}^p = d^p$ şi există ε *a.i.* $\left\|x^p - w_{k_2}\right\| - \left\|x^p - w_{k_1}\right\| < \varepsilon$ atunci

$$w_{k_1}(t+1) = w_{k_1}(t) - \eta\left(x - w_{k_1}(t)\right) \tag{2.39}$$

şi

BUPT

$$w_{k2}(t+1) = w_{k_2}(t) + \eta\left(x - w_{k_2}(t)\right) \qquad (2.40)$$

Diferenţa dintre algoritmii de tip LVQ constă în diferite implementări ale algoritmului de bază şi se referă la definirea claselor corecte, câţi vectori următori folosim, ce regulă de modificare a ponderilor folosim, etc.

2.4.3. Reţele Kohonen

Sunt reţele de tip SOM ce permit proiectarea vectorilor de intrare N dimensionali într-un spaţiu discret, de obicei 1 sau 2 dimensional, şi care asigură conservarea relaţiilor de vecinătate existente în datele de intrare.

Modul de funcţionare al acestei arhitecturi este bazat pe următoarea regulă: dacă avem 2 tipare de intrare alăturate şi ieşirile corespunzătoare intrărilor, trebuie să fie apropiate; altfel spus ieşirile trebuiesc ordonate topologic.

Prezentăm arhitectura unei reţele Kohonen (Figura 15)

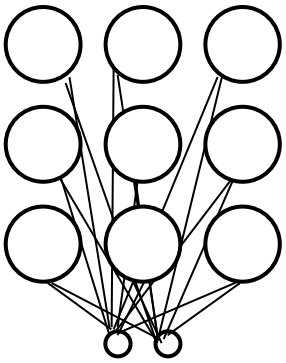


Figura 15. Arhitectura unei reţele Kohonen cu nivel funcţional bidimensional, cu două noduri de intrare şi 9 noduri de ieşire

Putem spune că o reţea neuronală de tip Kohonen este compusă numai dintr-un strat de intrare şi unul de ieşire, nu este prezent nici un strat ascuns. De asemenea toate intrările sunt total conectate la toate ieşirile (avem o reţea complet conectată).

Antrenarea unei reţele de tip Kohonen presupune:

- Alegerea unui tipar de intrare şi prezentarea acestuia la stratul de intrare al reţelei
- Determinarea unui nod învingător
- Ajustarea ponderilor pentru toate nodurile aflate în vecinătatea nodului învingător după formula:

$$w_o(t+1) = w_o(t) + \eta g(o,k)\left(x(t) - w_o(t)\right) \;\forall o \in S \qquad (2.41)$$

unde: *g(o,k)* este o funcţie descrescătoare ce exprimă distanţa dintre *o(nodul din vecinătate)* şi *k(nodul câştigător).* Trebuie menţionat că atât dimensiunea vecinătăţii cât şi rata de învăţare descresc în timp.

BUPT

Pentru o mai bună antrenare a reţelei intrările trebuiesc normalizate şi uniform distribuite de-a lungul intervalului datelor de intrare.

BUPT

3. ALGORITMI DE ÎNVĂŢARE

3.1. Noţiuni generale

În capitolul precedent am încercat să parcurgem câteva dintre cele mai cunoscute arhitecturi ale reţelelor neuronale. Fiecare dintre aceste reţele au proprietăţi care le fac să fie excelente în anumite domenii. Pentru rezolvarea problemelor din domeniile respective, reţelele neuronale trebuie să *înveţe,* proces care apare în urma antrenării cu date specifice domeniului. Reţeaua neuronală învaţă pe baza unui proces iterativ de ajustare al ponderilor sinaptice şi eventual al nivelului de activare. Dacă procesul de învăţare decurge bine, atunci reţeaua neuronală acumulează tot mai multe informaţii, la fiecare iteraţie.

Pentru clarificarea noţiunii de *„proces de învăţare"* propunem următoarea definiţie:

Învăţarea este un proces prin care parametrii reţelei neuronale sunt adaptaţi permanent prin intermediul unor stimuli proveniţi de la mediul înconjurător căruia îi aparţine reţeaua neuronală. Tipul de învăţare este determinat de forma de modificare a parametrilor reţelei neuronale.

Acest lucru este important deoarece noţiunea de învăţare este specifică multor domenii. Definiţia de mai sus este folosită numai în contextul calculului neuronal şi conţine următoarea secvenţă de evenimente:

- Reţeaua neuronală primeşte stimuli de la mediul înconjurător. Acest lucru este necesar deoarece reţeaua neuronală trebuie să dobândească cunoştinţe din domeniul în care ea activează.
- Primind aceşti stimuli din mediul exterior, ea îşi modifică parametrii interni (ponderi, rata de învăţare, structura nodurilor) pentru a face faţă cât mai bine solicitărilor din acel mediu.
- În urma modificărilor parametrilor, reţeaua răspunde mediului într-un mod diferit de cel precedent.

Se observă din această definiţie, cum reţelele neuronale încearcă o adaptare la mediul în care ele sunt antrenate. Adaptarea apare în urma procesului de învăţare şi presupune modificări interne ale reţelei neuronale.

Să încercăm să dăm o formulare matematică acestui proces descris mai sus. Cel mai simplu mod de adaptare presupune modificarea ponderilor dintre neuroni. Astfel:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) \tag{3.1}$$

- wji(t + 1) şi wji(t) reprezintă ponderea noua şi ponderea veche pentru legătura wji care uneşte axonul neuronului i de o dendrită a neuronului j.
- $\Delta w_{ji}(t)$ reprezintă valoarea aplicată ponderii wji(t), la momentul t, pentru obţinerea valorii wji(t + 1) la momentul t + 1

Ecuaţia prezentată încearcă să ilustreze modul prin care o reţea neuronală *învaţă*. Aşa cum a fost definită noţiunea de *învăţare* putem regăsi cele 3 evenimente

BUPT

în ecuaţia (3.1). Astfel : reţeaua neuronală primeşte stimuli de la mediul înconjurător în vederea ajustării wji(t); primind aceşti stimuli din mediul exterior, ea îşi modifică parametrii interni rezultând o nouă pondere wji(t + 1) care defineşte schimbarea din reţeaua neuronală. În urma modificărilor parametrilor, reţeaua răspunde mediului într-un mod diferit de cel precedent (diferenţa fiind dată de valoarea $\Delta w_{ji}(t)$).

Am definit noţiunea de *învăţare*, trebuie în continuare să definim modul cum această noţiune este implementată în cadrul reţelelor neuronale:

Vom numi algoritm de învăţare, un set de reguli predefinite care soluţionează problema „învăţării”.

Trebuie spus că, pentru o anumită reţea neuronală nu există un unic algoritm de învăţare. Modalitatea de a calcula modificare ponderii $\Delta w_{ji}(t)$ este diferită în funcţie de tipul algoritmului folosit. S-a observat însă că anumite reţele neuronale răspund mai repede şi mai bine în urma antrenării cu un anumit algoritm, de aceea majoritatea reţelelor neuronale includ unul sau doi algoritmi care sunt indicaţi a fi folosiţi pentru acea reţea.

Un alt factor important relativ la procesul de învăţare este modalitatea de interacţiune existentă între reţeaua neuronală şi mediul înconjurător. Pornind de la aceasta definim:

Vom numi paradigmă de învăţare, un model al mediului înconjurător în care are loc procesul de învăţare al reţelei neuronale.

Prezentăm în Figura 16 o modalitate de clasificare a algoritmilor şi a paradigmelor de învăţare

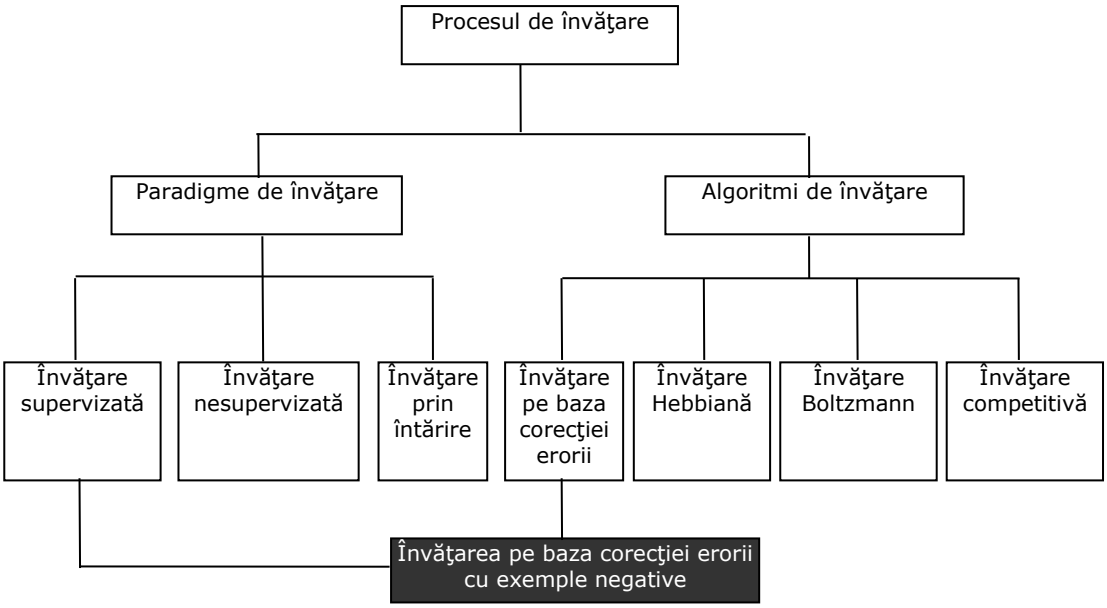


Figura 16. Taxonomia fundamentală a procesului de învăţare.

BUPT

Trebuie menţionat că existenţa unui număr foarte mare de clasificări la ora actuală, face ca cea utilizată de noi să cuprindă numai principalele direcţii ale algoritmilor şi paradigmelor de învăţare.

3.2. Paradigme de învăţare

3.2.1. Învăţare supervizată

Aşa cum reiese din nume, învăţarea supervizată este efectuată cu ajutorul unei *„supervizor”*(profesor). Acesta dispune de un set de date de antrenare de forma (*x(n), z(n)*) în care *x(n)* este un vector intrare, iar *z(n)* este un vector ieşire corespunzător intrării.

Metodele ce sunt caracterizate de paradigma *„învăţării supervizate”* au următoarea diagramă de funcţionare (Figura 17).

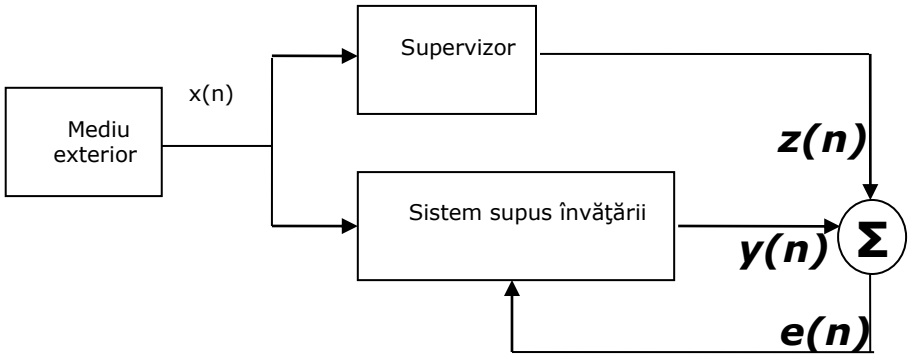


Figura 17. Sistem cu învăţare supervizată

Se observă cum vectorul *x(n)* este aplicat sistemului şi acesta are ca răspuns ieşirea *y(n)*. Diferenţa dintre răspunsul actual al sistemului *y(n)* şi răspunsul dorit *z(n)* este vectorul eroare *e(n).* Pe baza acestui vector sistemul îşi va modifica structura internă pentru a încerca minimizarea acestuia.

Putem aborda acest tip de învăţare atunci când avem definit setul de date de antrenament, adică atunci când avem cunoştinţe apriorice despre mediu exterior.

Tehnica de învăţare „Învăţarea pe baza corecţiei erorii cu exemple negative” propusă în această teză face parte din această categorie. Pentru acest tip de învăţare vectorul *x(n)* poate lua valori pozitive şi valori negative.

3.2.2. Învăţare nesupervizată

Este caracterizată de absenţa unui supervizor. În acest caz sistemul trebuie să descopere singur legăturile dintre datele furnizate la intrare. În absenţa supervizorului, nu cunoaştem ce răspuns trebuie să ofere sistemul atunci când i se prezintă o anumită intrare. Singur, trebuie să descopere corelaţii între datele de intrare şi să încerce ca acestea să fie evidenţiate în ieşirile lui.

BUPT

Acest lucru este posibil numai dacă în datele de intrare este prezentă o anumită redundanţă. Fără aceasta, sistemului îi va fi imposibil să detecteze trăsăturile comune dintre mai mulţi vectori de intrare.

Diagrama unui sistem supus unui algoritm de învăţare nesupervizat este prezentă în Figura 18.

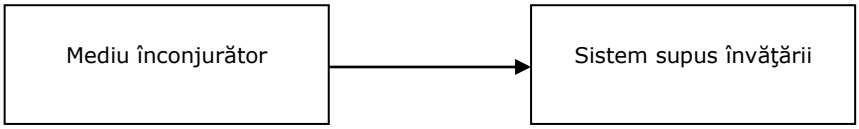


Figura 18. Sistem cu învăţare nesupervizată

Se poate observa din figură cum mediul înconjurător acţionează direct asupra sistemului, acesta neputând primi informaţii şi din altă parte.

3.2.3. Învăţare prin întărire

Învăţarea prin întărire are la bază studii efectuate asupra psihologiei animalelor. S-a observat că animalele sunt învăţate mai uşor să efectueze anumite acţiuni dacă acestea sunt recompensate în cazul în care efectuează acţiunea, respectiv pedepsite dacă nu efectuează acţiunea. Pe baza acestei idei, sistemul supus învăţării încearcă anumite acţiuni asupra mediului înconjurător. Fiecare dintre aceste acţiuni are ca rezultat(în urma unei evaluări) un indice de performanţă numit *semnal de întărire*.

Sistemul va fi astfel încurajat să producă numai acţiunile care au efecte pozitive şi va fi descurajat să producă acţiuni care au ca efect stări negative. Învăţarea prin întărire urmăreşte maximizarea *recompensei* primite. *Recompensa* poate fi de două feluri: *recompensă imediată*(obţinută prin trecerea sistemului într-o nouă stare) şi *recompensă subsecventă*(obţinută în stările următoare ale sistemului).

În cadrul RL se pot observa două faze: *explorare*(este o fază în care se achiziţionează date) şi *exploatare*(datele acumulate sunt prelucrate în scopul maximizării *recompensei*). Există o problemă legată de aceste faze, în sensul că pentru o valoare mare a recompensei este de preferat alegerea unei acţiuni cunoscute care are ca şi rezultat o recompensă mare (*exploatare)*, în timp ce noi acţiuni obţinute în faza de *explorare* pot avea valori mult mai mari ale recompensei.

Diagrama unui sistem supus învăţării prin întărire este prezentată în Figura 19.

BUPT

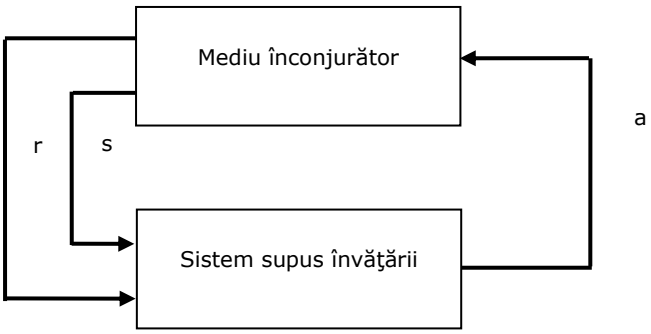


Figura 19. Sistem cu învăţare prin întărire

Mărimile care descriu modelul RL sunt următoarele: intrarea *s*(specifică starea mediului la un moment de timp), ieşirea *a*(acţiunea pe care o întreprinde sistemul asupra mediului), intrarea *r*(recompensa – valoare primită de sistem în urma acţiunii *a*), politica *Π*(comportamentul sistemului la un moment de timp), funcţia valoare *V(s)*(evaluarea stării *s* pe termen lung).

3.3. Algoritmi de învăţare

3.3.1. Învăţare pe baza corecţiei erorii

Introducem următoarele noţiuni:

- $x(n) = (x_1, x_2,\ldots, x_i)$ un vector de intrare aplicat unei reţele neuronale
- $y(n) = (y_1, y_2,\ldots, y_i)$ vectorul de ieşire obţinut de reţeaua neuronală în urma aplicării lui $x(n)$ la intrare
- $z(n) = (z_1, z_2,\ldots, z_i)$ vectorul de ieşire dorit, când la intrare este prezent vectorul $x(n)$

În timpul procesului de învăţare vectorii $z(n)$ şi $y(n)$ nu sunt identici. Diferenţa dintre ei este notată cu e(n) şi reprezintă vectorul eroare obţinut pentru intrarea x(n).

Aşa cum am definit procesul de învăţare putem regăsi stimulul mediului înconjurător în vectorul x(n), răspunsul reţelei neuronale în vectorul y(n) , iar adaptarea reţelei neuronale va fi dată de analiza vectorului e(n).

Învăţarea acestui algoritm este dată de corecţia vectorului eroare e(n). Pe baza acestui vector putem defini o funcţie eroare (sau de cost) care trebuie minimizată.

Cea mai des utilizată funcţie de eroare este funcţia abatere medie pătratică MSE (Mean Square Error), definită astfel:

$$e(n) = z(n) - y(n) \tag{3.2}$$

Aşa cum am definit procesul de învăţare putem regăsi stimulul mediului înconjurător în vectorul *x(n)*, răspunsul reţelei neuronale în vectorul *y(n)* , iar adaptarea reţelei neuronale va fi dată de analiza vectorului *e(n)*.

BUPT

Învăţarea acestui algoritm este dată de corecţia vectorului eroare *e(n)*. Pe baza acestui vector putem defini o funcţie eroare (sau de cost) care trebuie minimizată.

Cea mai des utilizată funcţie de eroare este *funcţia abatere medie pătratică MSE (Mean Square Error)*, definită astfel:

$$MSE = E\left[\frac{1}{2}\sum_{k=1}^{o} e_k(n)^2\right] \tag{3.3}$$

unde am notat cu E operatorul de medie statistică, iar $e_k(n)$ este eroarea obţinută la nivelul fiecărui neuron din stratul de ieşire. Astfel, la momentul *n* putem spune că eroarea dată de reţeaua neuronală în analiza unui stimul exterior este calculată cu formula (3.3).

Dacă vectorul de antrenament *x(n)* este al *n*-lea vector din mulţimea vectorilor de antrenament putem defini *funcţia eroare generală* care ţine cont de diferenţa dintre vectorii de ieşire *y(n)* şi vectorii ţintă *z(n)* relativ la toată mulţimea de antrenament (am notat cu *P* numărul total de vectori de antrenament). Putem scrie:

$$MSE_{general} = E\left[\frac{1}{2}\sum_{m=1}^{P}\sum_{k=1}^{o}\left(e_k(m)\right)^2\right] \tag{3.4}$$

Procesul de minimizare a funcţiei eroare MSE în raport cu parametrii reţelei neuronale (deci şi ăn funcţie de procesul de învăţare) este o optimizare cunoscută numită *metoda gradientului descendent*.

Metoda de optimizare necesită informaţii despre natura statistică a procesului. De cele mai multe ori aceste informaţii nu sunt cunoscute; în acest caz, considerăm o soluţie aproximativă a problemei de optimizare, şi anume, vom lua în considerare suma pătratelor erorilor instantanee dintre vectorul de ieşire *y(n)* şi vectorul ţintă *z(n)*:

$$MSE = \frac{1}{2}\sum_{k=1}^{o} e_k(n)^2 \tag{3.5}$$

Procesul de învăţare va consta în minimizarea funcţiei eroare (3.5), în raport cu ponderile sinaptice w_{ji}, pe baza metodei gradientului conjugat. O metodă de ajustare a ponderilor este oferită de Widrow şi Hoff (regula delta)[25]

$$\Delta w_{ji} = -\eta\frac{\delta MSE}{\delta w_{ji}} = \eta e_j(n)x_i(n) \tag{3.6}$$

Constanta reala η reprezintă *rata de învăţare*. Din formula (3.6) se poate observa că ajustarea ponderilor sinaptice în procesul de învăţare este proporţională cu produsul dintre stimulul (semnalul) de intrare şi semnalul de eroare.

BUPT

Dacă am reprezenta graficul funcţiei eroare în raport cu ponderile sinaptice w_{ji} ce caracterizează reţeaua neuronală, am obţine o hiper-suprafaţă, numită suprafaţa eroare. Putem întâlni două cazuri distincte în studiul suprafeţei eroare, şi anume:

- dacă reţeaua neuronală este constituită doar din neuroni ce au ca funcţie de activare funcţia liniară $f : R \to R, f(x) = ax + b$ (neuroni liniari), suprafaţa eroare are un *punct de minim unic*.
- dacă reţeaua neuronală este constituită din neuroni ce au ca funcţie de activare, funcţii neliniare (neuroni neliniari) (funcţia treaptă(Heaviside) $f : R \to \{0,1\}, f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$, funcţia rampă $f : R \to [-1,1], f(x) = \begin{cases} 1, x \geq 1 \\ x \in (-1,1) \\ -1, x < -1 \end{cases}$, etc.), atunci suprafaţa eroare are un punct de *minim global* şi numeroase alte *minime locale*.

Putem spune că minimizarea erorii pentru un neuron liniar este mai uşor de făcut decât pentru un neuron neliniar. În oricare situaţie, procesul de învăţare sau de minimizare a funcţiei eroare consta din pornirea dintr-un punct arbitrar al suprafeţei eroare (ce se obţine din valorile de iniţializare ale ponderilor reţelei neuronale), şi din deplasarea pas cu pas către punctul de minim global.

Ca şi dezavantaj putem evidenţia situaţia când deplasarea către punctul de minim global este blocată într-un punct de minim local.

3.3.2. Învăţare Hebbiană

În lucrarea sa „The Organization of Behavior”[8], Hebb emite una din cele mai faimoase ipoteze din neuropsihologie:

"Când un axon al celulei nervoase A este suficient de aproape de faza de excitare a unei celule nervoase B, şi în mod repetat sau persistent ia parte la activarea sa, un anumit proces de creştere sau de modificare metabolică are loc într-una sau în ambele celule nervoase, astfel încât eficienţa celulei nervoase A este mărită din punct de vedere al contribuţiei la activarea celulei B".

Acest lucru a fost implementat la nivelul algoritmilor de învăţare astfel: dacă 2 neuroni sunt activi simultan, atunci ponderea legăturii dintre aceştia este mărită; dacă 2 neuroni sunt activi asincron, atunci ponderea legăturii dintre aceştia este micşorată sau legătura este distrusă.

Formularea matematică a acestui algoritm este

$$\Delta w_{ji} = f\left(y_j(n), x_i(n)\right) \tag{3.7}$$

BUPT

unde f(yj(n),xi(n)) este o funcţie de două variabile, prima variabilă reprezentând activitatea pre-sinaptică $y_j(n)$, iar a doua variabilă reprezentând activitatea post-sinaptică $x_i(n)$.

Putem scrie ecuaţia (3.7) şi având forma:

$$\Delta w_{ji} = \eta y_j(n) x_i(n) \tag{3.8}$$

în care am notat cu η rata de învăţare. Creşterea exponenţială a lui Δw_{ji} însă principalul dezavantaj în acest caz, conducând la apariţia unui termen de uitare. Ecuaţia (3.8) devine:

$$\Delta w_{ji} = \eta y_j(n) x_i(n) - a y_j(n) w_{ji}(n) \tag{3.9}$$

3.3.3. Învăţare Boltzmann

Este inspirată dintr-o metodă ce are la bază o metodă probabilistică derivată din teoria termodinamicii şi din teoria informaţională .

Reţeaua neuronală este reprezentată printr-o structură recurentă, în care neuronii sunt reprezentaţi într-o manieră binară. Un neuron activ are o stare cu valoarea (+1) în timp ce un neuron inactiv are o stare cu valoare (-1). Întreaga reţea neuronală este caracterizată de o funcţie de energie *E* calculată astfel:

$$E = -\frac{1}{2}\sum_i \sum_{j, i \neq j} w_{ji} s_j s_i \tag{3.10}$$

unde s_i reprezintă starea neuronului i, s_j reprezintă starea neuronului j, iar w_{ij} reprezintă ponderea sinaptică dintre neuronii i şi j.

Maşina Boltzmann are următorul algoritm de operare:

- se alege un anumit neuron din structura internă a reţelei s_i. Se citeşte starea curentă a acestuia şi se modifică: $s_i \rightarrow -s_i$
- modificarea stării unui neuron este făcută la o anumită „temperatură" T şi cu probabilitatea

$$W(s_j \rightarrow -s_j) = \frac{1}{1 + e^{\frac{\Delta E_j}{T}}} \tag{3.11}$$

în care ΔE_j reprezintă modificarea energetică a reţelei neuronale în urma modificării stării neuronului s_i

- aplicarea acestei reguli în mod treptat conduce reţeaua neuronală câte un punct de *echilibru termic*.

BUPT

Pe lângă neuronii ce urmează comportamentul algoritmului descris (*neuroni vizibili)* există şi neuroni care au un mod de operare liber (*neuroni invizibili*).

Modurile de operare ale reţelei sunt şi ele în număr de 2:

- *condiţii impuse* – neuronii vizibili *copiază* o stare specifică mediului înconjurător;

- *condiţii libere* - toţi neuronii, vizibili şi invizibili sunt lăsaţi liber.

Dacă notăm cu:

- c_{ji}^{+} *corelaţia condiţională* dintre stările neuronilor *i* şi *j*, aflaţi în condiţii impuse (adică modul în care neuronul *i* influenţează pe neuronul *j* în anumite condiţii impuse).
- c_{ji}^{-} corelaţia necondiţională dintre stările neuronilor *i* şi *j*, aflaţi în condiţii libere (adică modul în care neuronul *i* influenţează pe neuronul *j* fără nici un fel de condiţii impuse).

în care c_{ji}^{+} şi c_{ji}^{-} sunt calculate ca medii ale tuturor stărilor posibile ale reţelei neuronale, când aceasta a atins punctul de *echilibru termic.* Formula matematică pentru aceste corelaţii este:

$$c_{ji}^{+} = \sum_{a}\sum_{\beta} P_{a\beta}^{+} S_{j|a\beta} S_{i|a\beta} \tag{3.12}$$

$$c_{ji}^{-} = \sum_{a}\sum_{\beta} P_{a\beta}^{-} S_{j|a\beta} S_{i|a\beta} \tag{3.13}$$

unde am notat:

- $S_{i|a\beta}$ – starea neuronului i, dacă neuronii vizibili se găsesc în starea A şi neuronii invizibili se găsesc în starea B;
- $P_{a\beta}^{+}$ – probabilitatea condiţională ca neuronii vizibili să se găsească în starea A iar toţi neuronii invizibili în starea B, dacă reţeaua neuronală este în modul de operare impus;
- $P_{a\beta}^{-}$ – probabilitatea condiţională ca neuronii vizibili să se găsească în starea A, iar toţi neuronii invizibili în starea B, dacă reţeaua neuronală este în modul de operare liber;

În urma acestor consideraţii regula de învăţare Boltzmann este definită matematic în felul următor:

$$\Delta w_{ji} = \eta\left(c_{ji}^{+} - c_{ji}^{-}\right), i \neq j \tag{3.14}$$

Parametrul η reprezintă rata de învăţare a sistemului supus unui algoritm de învăţare de tip Boltzmann.

BUPT

3.3.4. Învăţare competitivă

Se bazează pe competiţia dintre neuronii stratului de ieşire. În acest algoritm este activat un singur neuron, spre deosebire de alţi algoritmi la care puteau fi activi mai mulţi neuroni.

Scopul acestui algoritm este *specializarea* fiecărui neuron în recunoaşterea anumitor trăsături din datele de intrare.

Formularea matematică a acestui algoritm este dată de relaţia:

$$\Delta w_{ji} = \begin{cases} \eta(x_i - w_{ji}), daca\ \ j\ este\ neuronul\ \ \ câştigător \\ 0, daca\ neuronul\ j\ nu\ este\ învins \end{cases} \quad (3.15)$$

Se poate observa cum ponderea w_j a neuronului j care a câştigat competiţia se apropie de tiparul vectorului intrare x.

Acest algoritm trebuie să aibă definit un mecanism ce permite competiţia dintre neuroni şi alegerea unui neuron ca fiind câştigător al competiţiei. Acest lucru trebuie să permită ca la un moment de timp să avem activat un singur neuron.

3.3.5. Învăţarea pe baza corecţiei erorii cu exemple negative

Face parte din ramura paradigmelor de învăţare supervizate şi are la bază tehnica de învăţare pe baza corecţiei erorii. Este un nou tip de învăţare şi a fost dezvoltată pornind de la următoarele informaţii:

- un studiu asupra rolului exemplelor negative în antrenarea reţelelor neuronale de tip multilayer perceptron. În urma acestui studiu a fost propus şi un model general de antrenare cu exemple negative
- o nouă tipologie de reţea neuronală cu un neuron adiţional pe stratul de ieşire responsabil cu detectarea exemplelor negative

Toate acestea au condus la formarea unei noi tehnici de antrenare care foloseşte un procent exact de exemple negative în setul de antrenare şi utilizează un neuron suplimentar pe stratul de ieşire pentru a semnala tiparele negative de la intrare.

Exemplele negative au fost utilizate până în prezent ca mijloc de control asupra confidenţei reţelei şi indirect pentru îmbunătăţirea performanţelor reţelei. Prin noua tehnică ele devin parte intrinsecă a procesului de antrenare conducând la obţinerea unor performanţe superioare celor atinse în prezent.

3.4. Funcţii de activare

Funcţiile de activare sunt folosite pentru calculul ieşirii unui neuron. Valorile de ieşire tradiţionale pentru un neuron sunt 0 (reprezintă faptul că acel neuron este inactiv) şi 1 (reprezintă faptul că acel neuron este activ). Valoarea obţinută la ieşire

BUPT

este calculată pe baza intrărilor neuronului şi a ponderilor asociate fiecărei intrări (3.16) .

$$y = f_h\left(\sum_{i=0}^{N-1} w_i x_i\right) \tag{3.16}$$

În această formulă f_h este funcţia de activare şi ea este aleasă în funcţie de topologia reţelei şi de datele cu care lucrează reţeaua neuronală.

Pentru straturile ascunse ale reţelei funcţiile de activare trebuie să fie din categoria funcţiilor neliniare. Astfel ele introduc neliniaritatea în neuronii ascunşi, fără de care aceştia nu ar fi decât simpli perceptroni. Motivul este acela că o funcţie compusă din mai multe funcţii liniare este tot o funcţie liniară, ceea ce transformă reţeaua neuronală într-una cu 2 straturi, unul de intrare şi unul de ieşire. Puterea neuronilor ascunşi rezidă din neliniaritatea de care aceştia dau dovadă, fapt ce conduce la obţinerea unei reţele neuronale cu straturi ascunse.

Dacă folosim tehnica de antrenare de tip backpropagation, trebuie să avem grijă ca funcţia de activare să fie diferenţiabilă, deoarece avem nevoie de derivata ei în etapa de modificare a ponderilor. Dacă funcţia de activare are valori de ieşire pe un interval finit obţinem un plus de performanţă pentru această tehnică.

Pentru neuronii ascunşi este de preferat să folosim funcţia sigmoidă în defavoarea funcţiei de prag. Acest aspect este util la antrenarea reţelei neuronale, deoarece pentru funcţia prag ieşirea poate lua doar câteva valori. Astfel, se creează cazuri în care, deşi modificăm intrarea reţelei neuronale, ieşirea reţelei nu se modifică. În acest caz este dificil să spunem dacă modificarea produsă este sau nu benefică antrenării.

Prezentăm în Tabelul 1 definiţia principalelor funcţii de activare şi intervalul în care acestea pot lua valori. Se poate observa că majoritatea funcţiilor sunt diferenţiabile şi au la ieşire valori cuprinse într-un interval finit.

Funcţia	Definiţie	Intervalul în care poate lua valori
Identitate	$f(x) = x$	$(-\infty, +\infty)$
Logistică	$f(x) = \frac{1}{1 - e^{-x}}$	$(0, +1)$
Hiperbolică	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	$(-1, +1)$
Exponenţială	$f(x) = e^{-x}$	$(0, +\infty)$
Softmax	$f(x) = \frac{e^{x}}{\sum_{i} e^{x_i}}$	$(0, +1)$

BUPT

Sumă unitate	$f(x) = \frac{x}{\sum_i x_i}$	$(0, +1)$
Radical	$f(x) = \sqrt{x}$	$(0, +\infty)$
Sinus	$f(x) = sin(x)$	$[0, +1]$
Rampă	$f(x) = \begin{cases} -1 & , x \le -1 \\ x & , -1 < x < +1 \\ +1 & , x \ge +1 \end{cases}$	$[-1, +1]$
Treaptă	$f(x) = \begin{cases} 0 & , x < 0 \\ +1 & , x \ge 0 \end{cases}$	$[0, +1]$
Tangentă hiperbolică	$f(x) = tanh(x)$	$[-1, +1]$

Tabelul 1. Definiţia principalelor funcţii de activare

Există si o împărţire a funcţiilor de activare pe categorii, în funcţie de valoarea pe care reţeaua neuronală o prezintă la ieşire. Astfel avem următoarele categorii:

- Dacă ieşirile reţelei sunt binare se utilizează funcţiile logistică şi softmax. Problemele care se încadrează în această categorie sunt acelea de recunoaştere de tipare, de clasificare, etc. Funcţia softmax este predominant folosită la stratul de ieşire pentru un sistem de clustere. Aceasta converteşte o valoare brută într-o probabilitate posterioară ce ne oferă o măsură a certitudinii.
- Pentru ieşiri ale reţelei care se doresc a fi continue şi limitate de un interval avem funcţia de activare logistică, sinus şi tangentă hiperbolică. În această categorie se regăsesc funcţii de aproximare, metode de interpolare şi metode statistice.
- Pentru ieşiri continue şi nelimitate avem funcţia exponenţială, radical sau identitate.

BUPT

4. ÎMBUNĂTĂŢIREA PERFORMANŢELOR REŢELELOR NEURONALE – ANTRENAREA CU EXEMPLE NEGATIVE

4.1. Problematica

Reţelele neuronale îşi dovedesc în principal utilitatea în rezolvarea unor probleme dificile, cum sunt cele de estimare, identificare şi predicţie, sau de optimizare complexă. Domeniul de aplicabilitate al acestora este foarte vast, ele însoţindu-ne în viaţa de zi cu zi, fie prin încorporarea lor în aparate electrocasnice (telefon celular, maşină de spălat, televizor, cuptoare cu microunde, etc.) sau interacţionând cu ele prin intermediul diverselor aspecte ale vieţii curente (recunoaştere forme, recunoaştere vorbire, diagnostic automat, etc.).[26]

Realizarea unei astfel de reţele este făcută în mai mulţi paşi. Un prim pas este acela prin care se selectează un anumit tip de reţea neuronală(feed forward, recursiva, Kohonen,etc.) şi se defineşte o configuraţie pentru acea reţea (numărul de straturi, numărul de neuroni, funcţie de activare, etc.). Pasul următor, odată ce s-a ales reţeaua pentru o anumită aplicaţie, este procesul de antrenare al reţelei. Ultimul pas este acela în care reţeaua neuronală este testată pe un set de date cât mai general şi în funcţie de rezultatul obţinut, fie procesul este reluat de la pasul 1, fie reţeaua poate fi utilizată cu succes.

Antrenarea unei reţele neuronale este posibilă datorită interacţiunii multiple dintre om şi obiectul ce are incorporată reţeaua neuronală. Aşa cum ştim[27], pentru antrenarea unei reţele neuronale trebuie parcurse 2 faze: cea de explorare şi cea de exploatare. În faza de explorare se achiziţionează datele ce sunt folosite în faza de exploatare.

Există o problemă legată de aceste faze, în sensul că pentru o valoare mare a ratei de recunoaştere este de preferat alegerea unei acţiuni cunoscute care are ca şi rezultat o recompensă mare (*exploatare)*, în timp ce noi acţiuni obţinute în faza de *explorare* pot avea valori mult mai mari ale ratei de recunoaştere. Această problemă este rezolvată prin introducerea în setul de antrenare a unor date corespunzătoare unor acţiuni cât mai variate. Diversitatea datelor din etapa de *explorare*, asigură o antrenare robustă a reţelei neuronale, aceasta fiind pregătită pentru etapa finală de testare şi pentru etapa de utilizare.

O altă problemă care apare în timpul antrenării reţelelor neuronale artificiale este legată de mărimea setului de antrenare achiziţionat în etapa de *explorare* şi utilizat în faza de *exploatare*. Reţelele neuronale sunt metode de învăţare statistică şi, după cum se observă şi în numeroase studii [28][29], rata de recunoaştere a reţelei creşte odată cu creşterea setului de antrenare. Din această cauză, pentru obţinerea unor rate de recunoaştere mari setul de antrenare trebuie să fie foarte mare (Figura 20).

BUPT

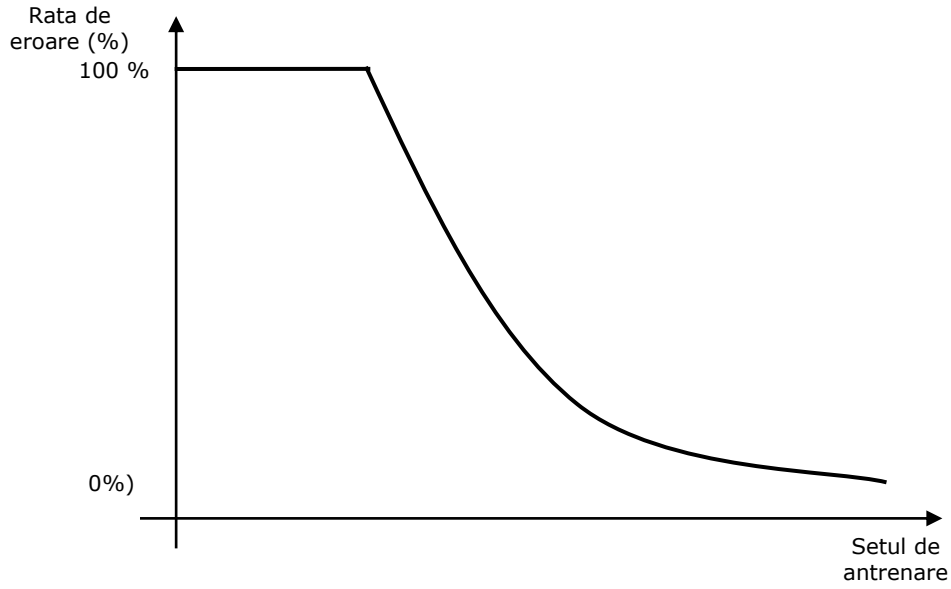


Figura 20. Evoluţia valorii ratei de eroare pentru o reţea neuronală în funcţie de mărimea setului de antrenare folosit

Din figură se poate observa că reţeaua începe să înveţe pentru o valoare minimă a numărului de tipare din setul de antrenare. De asemenea se poate observa că rata de eroare şi setul de antrenare sunt într-o relaţie liniară pe un anumit interval, în final relaţia dintre ei devenind exponenţială şi apoi asimptotică. Acest lucru face ca mărimea setului de antrenare să aibă valori critice pentru anumite intervale.

Indiferent de perioada de timp de care discutăm, au fost şi există situaţii [30][31][32][33] în care obţinerea datelor pentru faza de explorare este un proces foarte dificil dacă nu imposibil de realizat. Aceste situaţii au cauze multiple putând enumera:

- distanţa foarte mare – pentru prelevare de roci lunare, antrenarea trebuie făcută cu date care pot fi furnizate numai de telescoape sau alte sisteme de achiziţionare a datelor de la distanţe mari [33]
- inaccesibilitate – pentru prelevare de roci din interiorul unui vulcan, antrenarea trebuie făcută cu date culese de sisteme care să reziste la anumite temperaturi şi presiuni foarte ridicate [30]
- domenii de frontieră – detectarea unor noi viruşi; antrenarea unei reţele neuronale trebuie făcută cu date despre care nu ştim dacă sunt adevărate sau false. Fiind un domeniu nou, putem doar presupune referitor la validitatea acestor date [32]
- confidenţialitatea datelor – dacă dorim să antrenăm o reţea neuronală care să poată detecta cu succes anumite semnături „inamice” pe un radar, obţinerea unor date de antrenare pentru reţeaua neuronală este o operaţie riscantă şi foarte costisitoare. [31]

BUPT

Cu toate acestea, în fiecare dintre aceste exemple s-a antrenat o reţea neuronală care a fost folosită ulterior în faza de exploatare. Setul de date folosit la antrenarea reţelelor neuronale a trebuit extins în toate cazurile, deoarece iniţial era prea mic pentru acest proces. Extinderea lui a fost posibilă astfel:

- datelor iniţiale li s-au adăugat alte date generate pe calculator, acestea din urmă fiind obţinute în urma unor operaţii de procesare a primelor (procesarea este făcută cu diverşi algoritmi). Acest lucru conduce la obţinerea unui set mărit de date de antrenare, lucru dorit de noi, însă ca şi problemă putem spune că toate aceste date au strămoşi comuni. Acest dezavantaj face ca reţeaua neuronală să recunoască cu succes numai anumite trăsături, acelea care sunt conţinute de setul iniţial de date. Metoda poartă numele de *generare de exemple virtuale.*[34][35]
- la antrenarea reţelei au fost folosite exemple negative generate pe calculator prin diverse distorsionări severe ale datelor iniţiale. Această metodă reuşeşte să limiteze efectele metodei *generare de exemple virtuale*, prin aceea că specifică zonele în care reţeaua neuronală trebuie să genereze o valoare scăzută a funcţiei de recunoaştere. Metoda se numeşte *incorporare de exemple virtuale negative.*[36][31]

Astfel, putem concluziona faptul că obţinerea unui set iniţial de date pentru antrenarea reţelei este un proces dificil şi în unele cazuri imposibil. Prin tehnici de generare de exemple pozitive şi negative s-a reuşit îmbunătăţirea acestei probleme, dar nu rezolvarea ei în totalitate. Generarea unor noi exemple din exemplele aflate la dispoziţie nu face decât să amestece anumite caracteristici deja existente în setul de antrenare. Acest proces nu este capabil de a adăuga noi caracteristici setului de antrenare.

Un alt aspect negativ în procesul de antrenare al reţelei este acurateţea datelor de antrenare. În cele mai multe aplicaţii de clasificare cu ajutorul reţelelor neuronale dorim să reuşim clasificarea tuturor datelor de intrare într-un număr fix de clase. Există situaţii reale în care datele de intrare prezente în faza de utilizare a reţelei nu reuşesc să fie clasificate în clasele deja învăţate de către reţea. Aceste exemple se numesc exemple *confuze* sau *străine* şi astfel dintr-o problemă de clasificare obţinem o problemă de recunoaştere.

În problemele de recunoaştere avem mult mai multe exemple de clase de intrare decât suma tuturor claselor învăţate de către reţea, deoarece apar şi exemple care nu aparţin nici unei clase deja învăţate. Astfel în problemele de recunoaştere, pe lângă acurateţea clasificării (care rămâne în sine un scop), se doreşte şi recunoaşterea exemplelor străine şi refuzarea lor. Acest lucru poate dăuna totuşi unei importante caracteristici a reţelei neuronale, şi anume capacităţii de generalizare a acesteia. Găsirea unei proporţii echilibrate între exemplele de intrare ce pot fi clasificate cu succes de către reţeaua neuronală şi exemplele *confuze* este în sine o problemă care nu are încă o rezolvare totală. Există studii care sugerează valori empirice ale acestor valori, specifice fiecărui tip de problemă ce se doreşte a fi rezolvată cu reţelele neuronale.

O posibilitate a rezolvării conflictului dintre capacitatea de generalizare a reţelei şi capacitatea de refuzare a exemplelor confuze este valoarea de prag[37], dar deşi aceasta ne conferă o oarecare lejeritate în decizie, forma funcţiei de decizie este determinată numai de setul de antrenare şi de topologia folosită în construirea reţelei neuronale.

BUPT

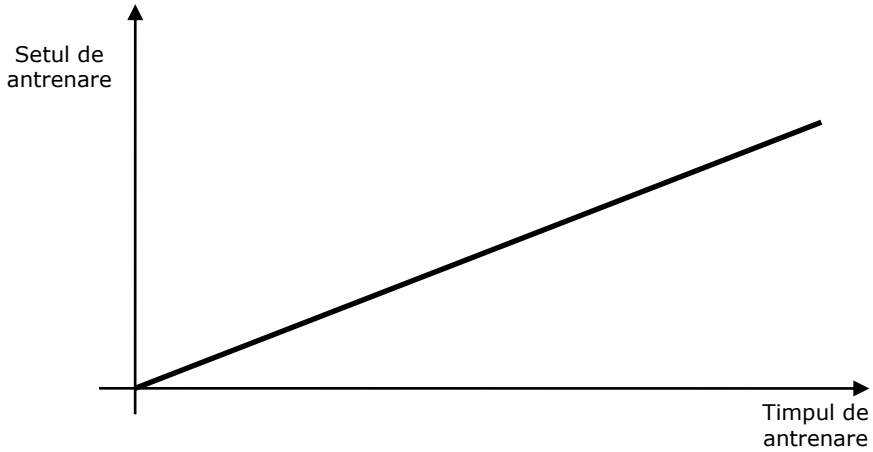


Figura 21. Graficul timpului de antrenare al unei reţele neuronale în funcţie de mărimea setului de antrenare folosit

Mărimea setului de antrenare presupune şi o mărire a timpului de antrenare, aspect care poate crea anumite probleme (o mărire a setului de antrenare cu 80% poate aproape dubla timpul de antrenare). (Figura 21)

Putem găsi oare o soluţie pentru toate aceste probleme? Răspunsul este „Da" şi va fi prezentat în capitolul următor.

4.2. Rezolvarea propusă

Toate problemele antrenării unei reţele neuronale prezentate în capitolul 4.1 au ca punct de pornire micimea setului de antrenare. Am prezentat câteva dintre soluţiile curente pentru mărirea lui şi anume *generarea de exemple negative* şi *generarea de exemple pozitive*. Ambele soluţii se bazează exclusiv pe extinderea setului *curent* de exemple, extindere care este efectuată în cele 2 direcţii, noi exemple pozitive sau noi exemple negative.

Antrenarea unei reţele neuronale pentru aplicaţii de clasificare are ca singur scop folosirea acesteia în faza de testare pentru clasificarea intrărilor viitoare. O bună parte dintre aceste intrări nu reuşesc să fie clasificate sau sunt clasificate incorect.

Ce-ar fi dacă reţeaua neuronală ar reuşi să clasifice şi intrările care nu aparţin nici unei clase de la ieşire? Cu alte cuvinte daca reţeaua neuronală nu este 100% sigură cărei clase de la ieşire îi aparţine intrarea curentă, în unele cazuri ea poate fi sigură 100% că aceasta nu aparţine nici unei clase.

Acest lucru ar putea fi implementat numai dacă am acorda importanţă egală exemplelor negative ca şi celor pozitive. Fiecare exemplu folosit ca intrare de reţeaua neuronală în faza de antrenare, trebuie să aibă asociată o clasă la ieşire. Această clasă poate fi una din clasele dorite sau o clasă specială în care avem toate exemplele care nu aparţin nici unei clase dorite.

Ideea care stă la baza implementării acestei ipoteze a venit din domeniul învăţării automate. Acest domeniu este responsabil cu învăţarea de către calculator a anumitor concepte, care implică mecanisme adaptabile.[38][39][40] Învăţarea

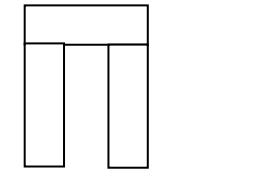
BUPT

automată nu face uz de un set foarte mare de cunoştinţe, atât din cauza costurilor mari presupuse de acumularea unor baze de informaţii mari cât şi din cauza complexităţii memorării şi prelucrării unui volum mare de informaţii. Acest lucru este ideal situaţiei noastre, în care avem la dispoziţie un set mic de exemple în setul de antrenare.

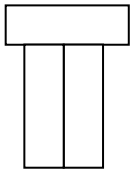
Ca şi reţelele neuronale, învăţarea trebuie să ducă la formularea de suficiente „reguli" atât cât să permită rezolvarea unor probleme dintr-un spaţiu mai larg decât cel pe baza căruia s-a făcut învăţarea, sistemul fiind astfel capabil de rezolvarea unor noi probleme.[41][42]

Mecanismele adaptabile folosite în domeniul învăţării automate sunt capabile de învăţare din experienţă, învăţare din exemple sau învăţare prin analogie.[43][44] Fiecare dintre aceste mecanisme presupune în primul rând identificarea şi implementarea unei modalităţi cât mai eficiente de a reprezenta informaţii, în sensul facilitării căutării, reorganizării şi modificării lor.

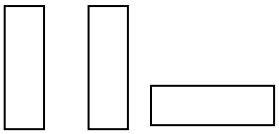
Bazându-ne pe un exemplu din [45] prezentăm în continuare modalitatea de învăţare a unei noţiuni cu ajutorul mecanismelor propuse de domeniul învăţării automate. Se poate observa grafic (Figura 22) că pentru învăţarea noţiunii de „arcadă" au fost propuse 4 exemple: 2 negative şi 2 pozitive.



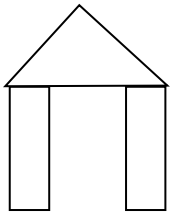
a) Este o arcadă



b) Nu este o arcadă



c) Nu este o arcadă



d) Este o arcadă

Figura 22. Definiţia noţiunii de arcadă

Învăţarea din exemple poartă numele de *inductive learning* (învăţare inductivă).[46][47][48] Calculatorul este *ajutat* să înveţe conceptul de arcadă cu ajutorul exemplelor negative şi exemplelor pozitive. Exemplele sunt procesate rând pe rând, fiecare dintre acestea modificând conceptul curent al noţiunii ce trebuie învăţată. Definiţia numită şi *ipoteză curentă* (current hypothesis) trebuie să susţină la final toate exemplele care au fost procesate de către calculator. Putem spune că o definiţie a unui obiect poate conţine atât caracteristici ce aparţin acelui obiect cât şi caracteristici care nu aparţin obiectului. Cele două tipuri de caracteristici sunt *extrase* şi asimilate de mecanismul de învăţare în funcţie de natura exemplului prezent la intrare.

BUPT

Astfel, dacă exemplul procesat este un exemplu pozitiv, caracteristicile găsite în acel exemplu sunt caracteristici care trebuie să aparţină obiectului învăţat. Daca exemplul procesat este unul negativ, extragem caracteristicile noi (neexplorate până atunci), iar aceste caracteristici nu trebuie să aparţină obiectului învăţat.

Acest mecanism face necesară recunoaşterea ambelor tipuri de caracteristici şi utilizarea lor în procesul de învăţare. Se poate afirma că importanţa exemplelor pozitive este egală cu a celor negative şi că ambele tipuri de exemple sunt utilizate în procesul de învăţare. Mai mult, putem spune că procesul de învăţare a unui concept este greşit dacă nu utilizează şi exemplele negative, deoarece acestea au rolul de a *trasa* graniţele noţiunilor care definesc un concept.

Un astfel de mecanism, prin care putem învăţa atât din exemplele pozitive cât şi din exemplele negative, implementat la nivelul reţelelor neuronale ar putea conduce la rezolvarea tuturor problemelor întâlnite în capitolul anterior (4.2). Am putea antrena reţeaua în mod egal atât cu exemple pozitive, cât şi cu exemple negative, ambele furnizând informaţii care să ajute în procesul de antrenare.

Acest fapt pare a fi în contradicţie cu definiţia oarecum simplistă dată de Alecsander unei reţele neuronală [49]:

O reţea neuronală este un procesor cu distribuţie paralelă, care are o mare capacitate de a stoca cunoştinţe dobândite din experimente, în scopul utilizării active ale acestora.

În acest context, noţiunea de antrenare cu exemple negative în scopul acumulării de cunoştinţe capabile de a fi utilizate în procesul de testare este oarecum forţată. De ce este nevoie de exemple negative, dacă reţeaua neuronală este făcută pentru a acumula cunoştinţe viabile, adică cele din exemple pozitive? Se poate afirma că stocarea caracteristicilor ce nu aparţin obiectului este o risipă a ponderilor unei reţele neuronale precum şi a neuronilor acesteia.

Cu toate acestea, situaţia de la care am plecat, cea în care avem un set de antrenare incapabil de a furniza toate caracteristicile ce stau la definirea unui obiect, face viabilă idea de antrenare a unei reţele neuronale cu exemple pozitive şi negative.

Astfel, o reţea neuronală capabilă să înveţe atât caracteristicile care *construiesc* obiectul, cât şi caracteristicile care sunt străine acestuia, ar reuşi în final să recunoască obiectele care au caracteristicile necesare şi nu au caracteristici nedorite. În plus reţeaua ar recunoaşte clar obiectele care nu aparţin nici unei clase de la ieşire, situându-le pe acestea într-o nouă clasă. Această împărţire scade eroarea reţelei făcând posibilă o demarcaţie clară între obiectele care sunt recunoscute de către reţeaua neuronală şi obiectele care nu sunt recunoscute de aceasta.

4.3. Îmbunătăţirea performanţelor reţelelor neuronale prin folosirea exemplelor negative

Antrenarea reţelelor neuronale cu exemple negative nu este o noutate în domeniu. Mulţi autori au folosit exemple negative pentru îmbunătăţirea performanţelor reţelei, în special atunci când tiparele de la intrare diferă foarte mult de cele cu care a fost antrenată. Un alt caz este acela în care tiparele de la intrare nu aparţin nici unei clase de la ieşire, în acest caz reţeaua furnizând date de ieşire greşite.[50]

BUPT

Cu toate acestea importanţa utilizării exemplelor negative este cumva minimizată ea nefiind o practică folosită în mod curent în antrenarea reţelelor neuronale. Această situaţie este datorată creşterii erorii reţelei neuronale în faza de antrenare din cauza utilizării unor tipare de intrare mult diferite de cele pozitive. O îmbunătăţire semnificativă apare numai atunci când reţeaua este testată cu date diferite de cele din etapa de antrenare. Aceste date conţin multe tipare pe care reţeaua le poate clasifica deoarece fie sunt zgomote, fie aparţin unor clase ce nu trebuie recunoscute de reţea. Altfel spus, atunci când reţeaua neuronală este confruntată cu date din *lumea reală* rata erorii creşte foarte mult deoarece aceasta este *învăţată* să recunoască aproape orice tipar prezent la intrare ca făcând parte dintr-o clasă de ieşire.

Vom prezenta în continuare trei noi metode de antrenare care pun accentul pe utilizarea exemplelor negative la fel de mult ca utilizarea celor pozitive. Fiecare dintre aceste metode oferă o alternativă viabilă în toate cazurile în care setul de antrenare este incomplet sau are prea puţine date de antrenare.

4.3.1. Antrenarea reţelelor neuronale cu exemple negative

Antrenarea *clasică* a reţelelor neuronale presupune folosirea unui număr de exemple în setul de antrenare şi introducerea aleatoare a acestora la intrarea reţelei până când eroarea scade sub un anumit prag.(Figura 23)

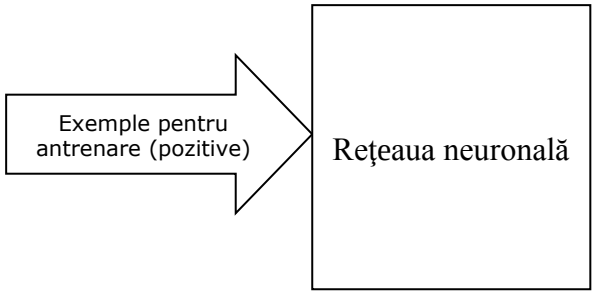


Figura 23. Antrenarea clasică a unei reţele neuronale

După faza de antrenare, la intrarea reţelei poate să fie prezent şi un exemplu care nu a fost folosit în etapa de antrenare. Dacă acesta este asemănător cu exemplele folosite, reţeaua va recunoaşte cu succes acest exemplu.

Dacă, la intrarea reţelei va fi introdus un exemplu total diferit de cele din setul de antrenare, reţeaua îl clasifică şi pe acesta ca aparţinând uneia dintre clasele de exemple cu care a fost antrenată. În acest caz spunem că reţeaua este *super încrezătoare* (over confident). Aceste erori apar frecvent în problemele de clasificare care dispun de un set insuficient de antrenare.[51]

O simplă rezolvare a acestor probleme presupune introducerea în setul de antrenare şi a unor exemple *negative* – exemple care nu trebuie recunoscute de către reţea ca făcând parte din categoriile cunoscute. De exemplu, dacă antrenăm o reţea să recunoască cifrele, în setul de antrenare este recomandat să introducem şi exemple de alte caractere (litere, semne de punctuaţie, alte semne).(Figura 24)

BUPT

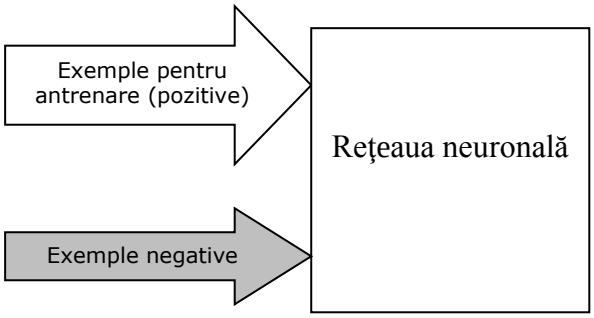


Figura 24. Antrenarea unei reţele neuronale cu exemple pozitive şi negative

În acest caz reţeaua este antrenată să recunoască exemplele din anumite categorii, dar şi exemplele care nu fac parte din categoriile dorite.

Au existat cazuri în care utilizarea exemplelor negative a depăşit cu mult numărul de exemple pozitive, fapt care a condus la diminuarea importanţei exemplelor negative faţă de exemplele pozitive printr-o anumită valoare [52]. Astfel, a fost introdusă o penalitate de exemplu negativ, prin care fiecare exemplu negativ putea influenţa rezultatul doar într-o măsură mai mică (funcţie de numărul total de exemple negative.

Din păcate, nicăieri nu este specificat care este procentul exemplelor negative din totalul exemplelor care trebuie avut în vedere la antrenarea unei reţele neuronale.(Figura 25) Toate studiile făcute până acum sunt specifice unui anumit tip de problemă, iar procentul de exemple negative utilizat la diverse aplicaţii variază de la 5% la 80%. [31][52][53]

Acest lucru este foarte important deoarece variaţia acestui procent modifică semnificativ procentul de recunoaştere al reţelei neuronale. Pentru un set de antrenare cu multe exemple negative se obţin rate foarte mici pentru recunoaşterea tiparelor dorite şi rate foarte mari de recunoaştere a tiparelor nedorite.

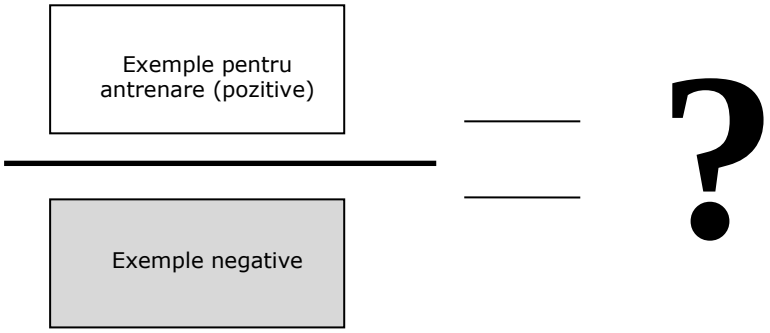


Figura 25. Care este procentul de exemple negative care trebuie folosit pentru o bună antrenare a reţelelor neuronale?

BUPT

Un număr mic de exemple negative în setul de antrenare nu reuşeşte să modifice ponderile reţelei astfel încât aceasta să recunoască cu succes tiparele nedorite.

Pentru rezolvarea problemei vom porni prin a detalia problema învăţării din exemple ce stă la baza antrenării supervizate a unei reţele neuronale artificiale.

Învăţarea din exemple aplicată unei reţele neuronale poate fi văzută ca o problemă de modelare a unei funcţii *f* cu o clasă ipotetică *H* şi un set de antrenare de tipul $D=\{ (x_i,y_i) , i = 1,..m\}$ în care $x_i \in R^n$ este exemplul n-dimensional folosit la antrenare, iar y_i este valoarea de ieşire a reţelei.

Eroarea obţinută în acest proces de modelare are forma următoare:

$$J = Er_{aproximare}(n) + Er_{estimata}\left(\frac{VC(H_n)}{m}\right) \tag{4.1}$$

, unde $VC(H_n)$ reprezintă o măsură a complexităţii clasei ipotetice. Pentru minimizarea acestei funcţii trebuie minimizate cele 2 componente: eroarea aproximată şi eroarea estimată.

Problema constă în faptul că eroarea de aproximare se micşorează cu creşterea complexităţii modelului, iar eroarea de estimare se micşorează odată cu scăderea complexităţii modelului.

Din această cauză pentru aceste tipuri de probleme trebuie găsit un compromis al complexităţii pentru a obţine o valoare minimă a erorii.

Pentru problema învăţării din exemple, modelul clasei ipotetice arată în felul următor:[54]

$$H[f] = \sum_{i=1}^{m}\left(f(x_i) - y_i\right)^2 + \lambda\|Pf\|^2 \tag{4.2}$$

, unde λ este un parametru pozitiv de regularizare, P este operatorul diferenţial iar $||Pf||^2$ este o funcţie de cost ce constrânge rapid spaţiul soluţiilor posibile pentru orice fel de informaţii prioritare.

Conform teoremei de regularizare soluţia acestei probleme are următoarea formă:

$$f(x) = \sum_{i=1}^{m} a_i K(x, x_i) + b \tag{4.3}$$

, unde a_i reprezintă coeficienţii funcţiilor kernel iar b reprezintă coeficienţii funcţiei bias.

La crearea unui număr de exemple negative de tipul $D'=\{ (x_i',y_i') , i = 1,..m\}$ formula (4.2) devine de forma [55]:

$$H[f] = \sum_{i=1}^{m}\left(f(x_i) - y_i\right)^2 - \sum_{i=1}^{m}\left(f\left(x_i'\right) - y_i'^0\right)^2 + \lambda\|Pf\|^2 \tag{4.4}$$

BUPT

, în care al doilea termen corespunde exemplelor negative, iar soluţia finală devine:

$$f(x) = \sum_{i=1}^{m} a_i K(x, x_i) + \sum_{i=1}^{m} a_i' K\left(x, x_i'\right) + b \qquad (4.5)$$

, unde a_i' reprezintă coeficienţii funcţiilor negative.

Rezolvarea problemei propuse presupune găsirea tuturor coeficienţilor formulei (4.5). În acest fel funcţia prin care este aproximată reţeaua neuronală este clar definită.

Plecând de la aceste considerente teoretice vom încerca experimental să găsim o formulă de calcul pentru numărul de exemple negative ce trebuie să alcătuiască setul de antrenare.

Pentru fiecare problemă o să avem nevoie de un număr diferit de exemple negative, dar se va încerca gruparea acestor numere şi găsirea unei formule care să interpoleze aceste valori.

Utilizarea exemplelor negative în antrenarea reţelelor neuronale duce la o îmbunătăţire a ratei de recunoaştere ale acestora. Totodată exemplele negative sunt mai uşor de construit (virtual vorbind toate celelalte exemple care nu sunt pozitive sunt negative) şi există într-un număr foarte mare.

Cu toate acestea pentru o îmbunătăţire a performanţei am dori ca reţeaua să poată recunoaşte şi exemplele negative în maniera în care sunt recunoscute cele pozitive. Descrierea metodelor care au făcut posibil obţinerea acestui lucru sunt detaliate în capitolele următoare.

4.3.2. Utilizarea neuronului block în procesul de antrenare al reţelei

În lumea reală întâlnim foarte des astfel de situaţii când nu dispunem de un set de antrenare adecvat. În special în domeniile militare [56] sau de frontieră ale ştiinţei avem situaţii pentru care o reţea neuronală nu a fost antrenată.

În capitolul 4.3.1 am arătat importanţa exemplelor negative în procesul de antrenare al unei reţele neuronale. Am observat astfel că pentru a evita o super confidenţă a reţelei, un număr de exemple negative trebuie introdus în setul de antrenare. Acestea reuşesc să asigure reţelei neuronale limitări în ceea ce priveşte recunoaşterea tiparelor.

La reţeaua neuronală, neuronii de pe ultimul strat sunt responsabili de recunoaşterea anumitor tipare dorite de utilizator. Fiecare dintre aceştia devine activ atunci când recunoaşte o caracteristică sau mai multe din exemplul de la intrare. Devenind activ, semnalează prezenţa unui anumit tipar la intrare.

Dacă neuronul de ieşire nu este activ, înseamnă ca el nu recunoaşte anumite caracteristici (care îi sunt alocate) în exemplul de la intrare. Atunci când nici un neuron de pe stratul de ieşire nu este activ, spunem că nici o caracteristică ce ne poate interesa nu este prezentă în exemplul de la intrare.

Atunci când toţi neuronii de pe stratul de ieşire sunt inactivi, tiparul de la intrare nu este recunoscut de reţeaua neuronală. Acest fapt nu înseamnă că tiparul de la intrare nu face parte din tiparele care trebuie recunoscute de reţeaua neuronală, ci doar faptul că reţeaua neuronală nu a reuşit să identifice nici o caracteristică care intră în componenţa tiparelor cunoscute.

BUPT

Putem da ca exemplu o reţea neuronală montată pe un echipament capabil de a colecta şi clasifica patru tipuri de roci vulcanice. [57] Putem afirma de la început că un astfel de echipament este scump şi deplasările efectuate de acesta în teren sunt costisitoare, deci se încearcă reducerea acestora la minim.

Pentru a putea colecta şi clasifica roci vulcanice reţeaua neuronală a fost antrenată cu o serie de roci existente ca şi exemple pozitive şi o mulţime de roci non vulcanice ca exemple negative. Cu toate acestea echipamentul poate întâlni pe teren o mulţime de alte roci, care nu au făcut parte din procesul de antrenare. Mare parte din aceste roci cauzează false recunoaşteri din partea reţelei neuronale. Astfel o mulţime de roci vulcanice nu sunt recunoscute ca fiind vulcanice (ieşirile reţelei sunt 0) iar o parte din cele care nu sunt vulcanice sunt recunoscute ca fiind vulcanice (una din ieşirile reţelei este 1).

Dorim sa limităm şi mai mult aceste greşeli, iar acest lucru este realizat prin adăugarea unui neuron pe ultimul strat.

Am observat că antrenarea cu exemple negative pregăteşte reţeaua neuronală pentru memorarea unor caracteristici nedorite. Din păcate nici un neuron de pe stratul de ieşire nu este direct responsabil de recunoaşterea unor tipare nedorite. Acest lucru este făcut atunci când toţi neuronii sunt inactivi, situaţie care poate genera confuzii.

Pentru a preîntâmpina astfel de situaţii avem nevoie de un neuron suplimentar pe ultimul strat. Am propus ca acest neuron să poartă numele de neuron **block** şi principala lui responsabilitate este semnalizarea tiparelor nedorite. Atunci când neuronul este activ putem spune că tiparul aflat la intrarea reţelei neuronale nu este un tipar pe care reţeaua neuronală ar trebui să-l recunoască. Altfel spus, neuronul **block** este responsabil cu recunoaşterea caracteristicilor care nu definesc clasa de obiecte ce trebuie recunoscută.

Vom arăta în continuare ce modificări trebuie făcute la nivelul uneia dintre cele mai folosite reţele neuronale: perceptronul multistrat.

Ca şi arhitectură, perceptronul cu mai multe straturi se prezintă ca în Figura 26.

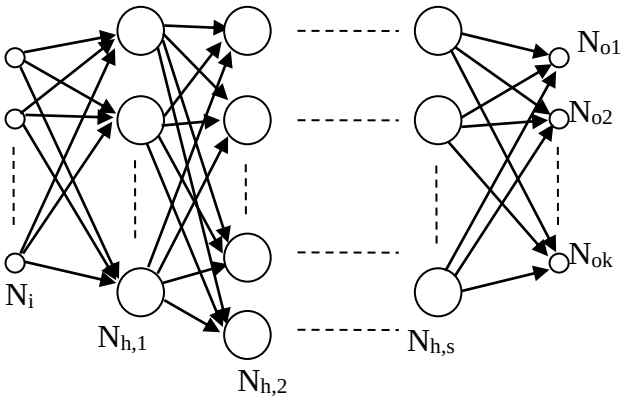


Figura 26. Arhitectura perceptronului cu mai multe straturi ascunse

Primul strat N_i este stratul de intrare. Neuronii de pe acest strat sunt lipsiţi de funcţia de activare, adică lasă să treacă orice valoare primită la intrare.

BUPT

Următoarele *s* straturi ($N_{h,1}$, .., $N_{h,s}$) se numesc straturi ascunse. Fiecare strat ascuns ($N_{h,l)}$ primeşte valori la intrare numai de la stratul anterior lui ($N_{h,l-1}$), ieşirile lui constituindu-se ca intrări pentru stratul următor ($N_{h,l+1}$). Nu sunt permise conexiuni în interiorul stratului. Ieşirile ultimului strat ascuns $N_{h,s}$ sunt intrări pentru stratul de ieşire N_o. Trebuie spus că sunt permise şi conexiuni de genul *skip layer* (astfel neuronii pot primi ca şi intrări, ieşirile altor neuroni care nu se află pe un strat imediat precedent lor). Conexiunile directe între stratul de intrare şi cel de ieşire sunt în special foarte folositoare.

Numărul de neuroni de pe straturile ascunse poate fi diferit de la strat la strat. Fiecare dintre aceşti neuroni are o funcţie de activare F_i care acţionează asupra intrărilor şi *bias*-ului.

Modificarea perceptronului multistrat trebuie făcută astfel încât acesta să poată reţine cu succes caracteristicile nedorite. Pentru aceasta trebuie să adăugăm un neuron **block** pe ultimul strat. Perceptronul multistrat cu 1 neuron **block** are următoarea arhitectură modificată. (

Figura 27)

Exemplu: Daca antrenăm o reţea neuronală pentru recunoaşterea caracterelor a, o, c, u, e putem lua ca exemple negative caracterele d, t, b, g, p, iar caracteristica ce nu aparţine primei clasa ar fi "coada" care intra in componenţa caracterelor din clasa a 2-a. Această caracteristică ar trebui memorată de reţeaua neuronală şi semnalată de neuronul **block** (prin activarea acestuia) de fiecare dată când un caracter ce prezintă această caracteristică este prezent la intrare.

Trebuie precizat că antrenarea unei astfel de reţele nu presupune nici o modificare faţă de antrenarea clasică. Nu este nevoie să antrenăm în mod suplimentar neuronul **block** pentru ca acesta să poată reţine cu succes caracteristicile nedorite. Acest lucru se va întâmpla în etapa de antrenare şi nu este transparent utilizatorului.

Adăugarea unui neuron **block** pe stratul de ieşire şi utilizarea acestuia ca *neuron semnalizator* pentru tipare nedorite, face ca reţeaua neuronală să reuşească să clasifice atât tiparele dorite (unul sau mai mulţi neuroni *normali* este activ) cât şi pe cele nedorite (neuronul **block** este activ).

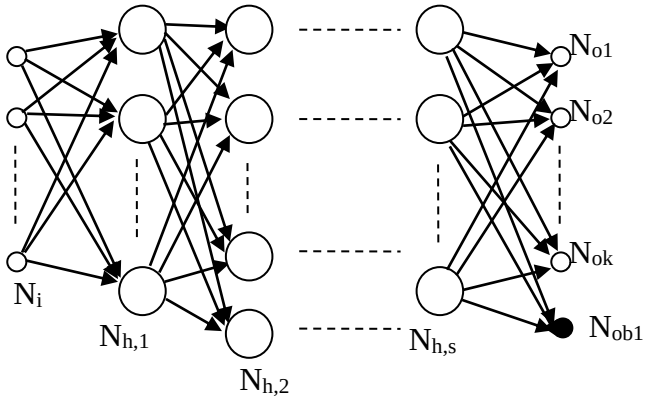


Figura 27. Arhitectura perceptronului cu mai multe straturi ascunse şi 1 neuron block

BUPT

Revenim la exemplul de la care am plecat, şi anume cel despre echipamentul care trebuie să recunoască roci vulcanice, şi considerăm totalitatea rocilor care pot fi intrări pentru reţeaua neuronală ca fiind formată din două mulţimi alcătuite din roci vulcanice şi roci non vulcanice. Cele două mulţimi sunt prezentate în Figura 28 şi după cum se observă, există o linie de demarcaţie clară între ele.

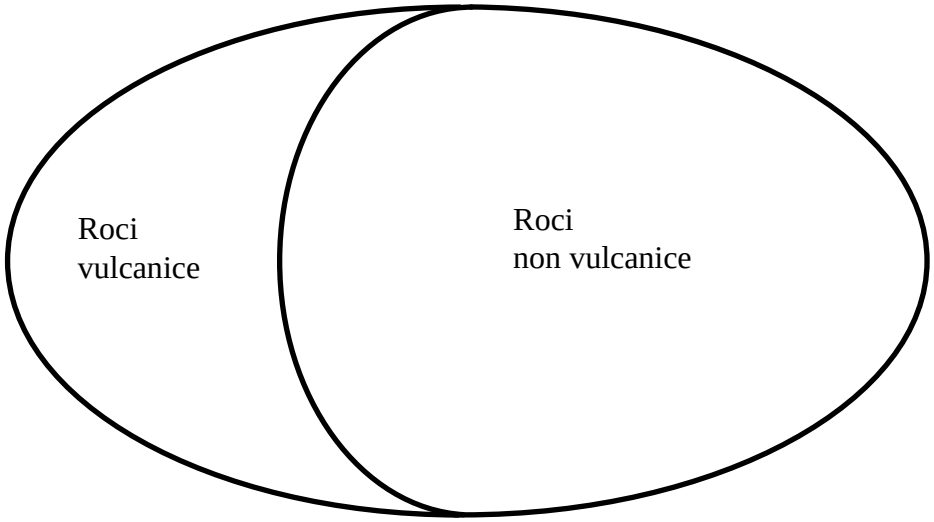


Figura 28. Totalitatea exemplelor pentru reţeaua antrenată în recunoaşterea rocilor vulcanice

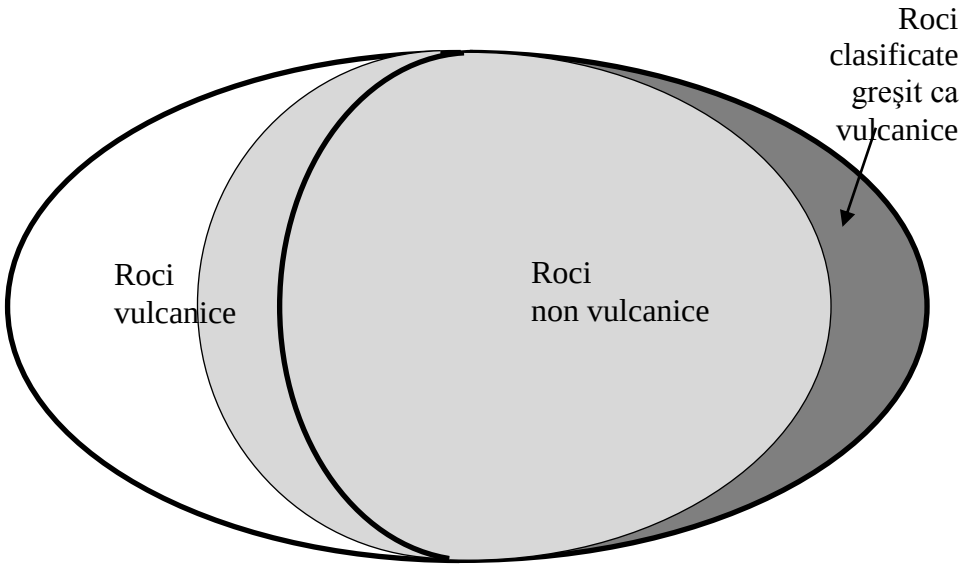


Figura 29. Clasificarea rocilor făcută de o reţea neuronale antrenată cu exemple negative şi pozitive

BUPT

Considerăm primul caz în care avem o reţea neuronală fără neuron **block** şi o antrenăm folosind exemple pozitive şi negative.(Figura 29)

În faza de testare reţeaua neuronală clasifică corect numai o parte din rocile vulcanice (zona albă din desen). O parte din rocile non vulcanice este incorect clasificată ca făcând parte din categoria rocilor vulcanice. Această parte poate varia între 100% şi 0%(cel mai probabil între 10% şi 30%) din mulţimea rocilor non vulcanice. (zona gri închis din dreapta)

Părţile complementare celor două mulţimi, respectiv rocile vulcanice şi non vulcanice nerecunoscute de către reţea sunt cele la care neuronii de la ieşire sunt inactivi. Această zonă este desenată cu gri deschis şi ocupă o mare parte in desen. Aceasta este o zonă de incertitudine, deoarece nu putem şti cu siguranţă în ce categorie intră rocile atunci când ieşirile reţelei sunt inactive. O rocă plasată într-o astfel de zonă poate fi o rocă vulcanică nerecunoscută de reţea sau o rocă non vulcanică.

Pentru mărirea gradului de claritate al clasificării antrenăm o reţea neuronală cu un neuron **block** cu aceleaşi exemple negative şi pozitive. În acest caz distribuţia ieşirilor reţelei este următoarea. (Figura 30)

Se poate observa cum zona albă (certitudine) este mult mai mare decât cea de la reţeaua fără neuron **block**. După introducerea neuronului **block** se poate observa că o mare parte a rocilor care nu sunt vulcanice sunt catalogate corect de către reţea.

Scăpăm astfel de o parte din situaţiile în care toţi neuronii de pe ultimul strat erau inactivi şi care genera o confuzie legată de exemplul de la intrare (nu ştim sigur dacă exemplul de la intrare nu a fost recunoscut de către reţea sau nu face parte din exemplele dorite de noi).

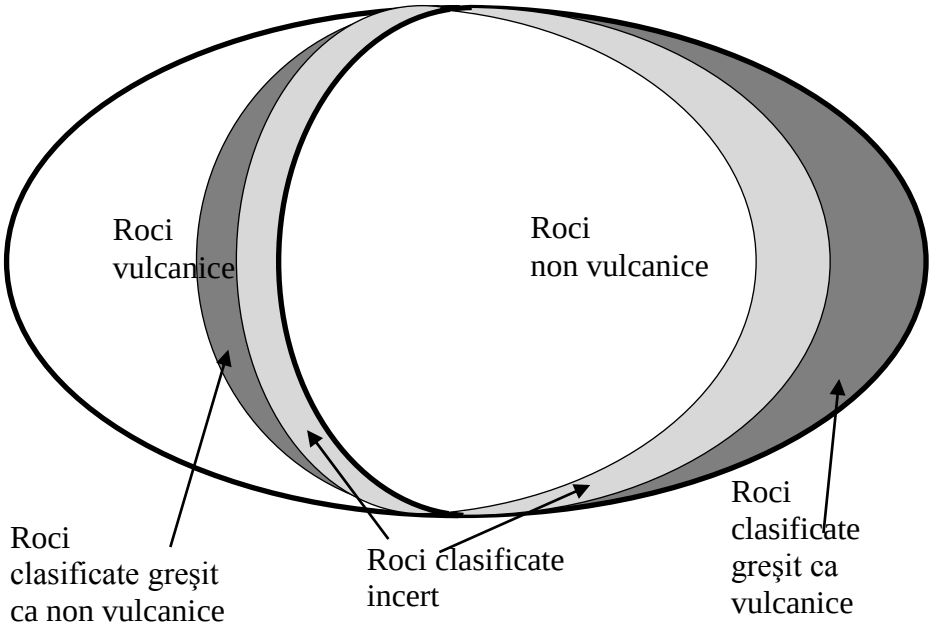


Figura 30. Clasificarea rocilor făcută de o reţea neuronală cu un neuron block antrenată cu exemple negative şi pozitive

BUPT

Avem în continuare zone gri închis (zone în care reţeaua neuronală greşeşte), precum şi zone gri deschis (zone de incertitudine), caz în care toţi neuronii de la ieşire sunt inactivi, inclusiv neuronul **block**.

Folosirea neuronului **block** pe ultimul strat îmbunătăţeşte considerabil procentul de răspunsuri certe oferite de reţea precum şi ratele de recunoaştere ale reţelei neuronale. Care sunt noile rate de recunoaştere şi cu cât s-a îmbunătăţit procentul de răspunsuri certe, sunt răspunsuri date în capitolul 5.

4.3.3. Utilizarea mai multor neuroni block având funcţii de activare diferite

În capitolul anterior am definit neuronul **block** şi am explicat rolul lui în antrenarea unei reţele neuronale. Am arătat că acesta face parte din stratul de ieşire al reţelei şi este răspunzător de semnalarea exemplelor negative aflate la intrarea reţelei neuronale. Modalitatea în care el a fost integrat în stratul de ieşire şi ce fel de rezultate produce au deschis calea către acest capitol.

În acest capitol vom face un pas înainte şi vom încerca să introducem mai mulţi neuroni **block** pe stratul de ieşire. Diferenţa dintre ei va consta tocmai în funcţia de activare specifică fiecăruia.

Rolul funcţiei de activare este foarte important pentru un neuron, deoarece numai aşa putem asigura nonlinearitatea reţelei neuronale. [58] Funcţia de activare este o caracteristică a reţelei şi se alege în funcţie de problema ce trebuie rezolvată.

Din această cauză spunem că fiecare funcţie de activare este capabilă de recunoaşterea numai unor anumite caracteristici din setul de intrare. Folosirea mai multor neuroni cu funcţii de activare diferite este modalitatea prin care se încearcă reţinerea cât mai multor caracteristici şi utilizarea acestora în procesul de recunoaştere.

Raţiunea folosirii mai multor neuroni **block** pe stratul de ieşire derivă din faptul că stratul de antrenare este compus din exemple pozitive şi negative şi pentru fiecare din aceste exemple unul sau mai mulţi neuroni de pe stratul de ieşire devine activ. Exemplele pozitive sunt mult mai omogene decât cele negative şi astfel neuronii de pe stratul de ieşire au aceeaşi funcţie de activare. Omogenitatea provine din faptul că toate exemplele pozitive reprezintă o clasă de obiecte ce trebuie recunoscută de reţeaua neuronală.

Atunci când un exemplu negativ este introdus la intrare, unul din neuronii **block** de pe ultimul strat devine activ şi semnalează astfel un tipar nedorit. Deoarece tiparele nedorite fac parte dintr-o mulţime foarte mare aceasta nu este omogenă. Astfel, reţeaua neuronală poate avea la intrare valori dintre cele mai ciudate (tipare nerecunoscute), iar ea trebuie să reuşească să semnaleze acest fapt prin intermediul neuronilor **block**.

Având funcţii de activare diferite, aceştia sunt capabili în interceptarea unei varietăţi foarte mari de exemple negative.

Modificările aduse reţelei pentru a suporta mai mulţi neuroni **block** pe ultimul strat sunt mici şi similare cu cele pentru 1 neuron block. Astfel, dacă considerăm perceptronul multistrat din Figura 26, pentru a avea mai mulţi neuroni block, arhitectura acestuia devine ca cea din Figura 31.

Fiecare dintre straturile ascunse au în componenţa lor acelaşi număr de neuroni ca şi reţeaua neuronală fără nici un neuron block. Funcţiile de activare sunt stabilite la nivelul fiecărui strat şi sunt dependente de problema ce necesită rezolvare.

BUPT

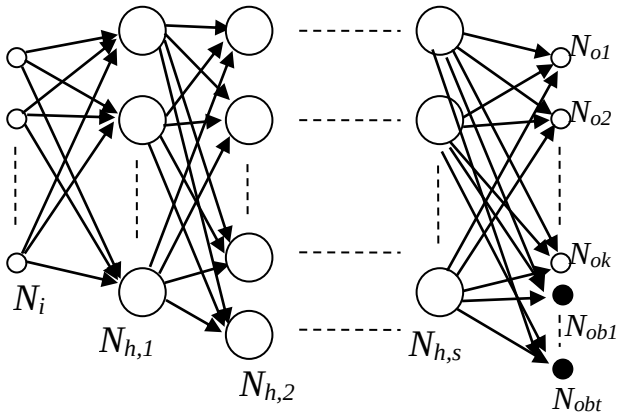


Figura 31. Arhitectura perceptronului cu k neuroni normali şi t neuroni block pe ultimul strat

Pentru neuronii block de pe ultimul strat avem mai multe funcţii de activare. Trebuie reamintit că dacă folosim tehnica de antrenare de tip back propagation, trebuie să avem grijă ca funcţia de activare să fie diferenţiabilă, deoarece avem nevoie de derivata ei în etapa de modificare a ponderilor.

O vedere mai detaliată asupra modului în care funcţionează neuronii block se poate observa în Figura 32.

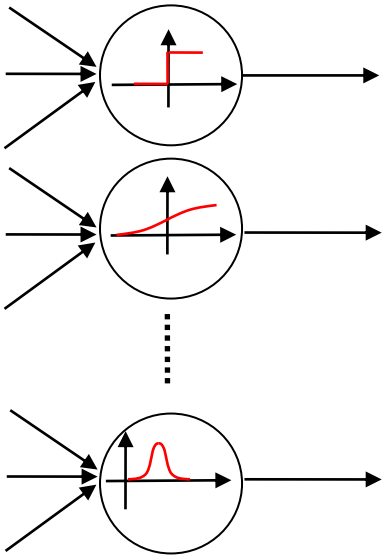


Figura 32. Neuronii block cu funcţii de activare diferită

Se observă că fiecare dintre neuroni are o altă funcţie de activare. Cu cât sunt mai mulţi neuroni cu atât numărul de funcţii de activare diferită creşte.

BUPT

De altfel, considerăm că un număr crescut de neuroni block este capabil de a intercepta mai multe tipare negative decât un număr mai mic.

Experimentele se vor concentra pe determinarea numărului de neuroni block optimi pentru fiecare problemă şi pe găsirea acelor funcţii de activare care satisfac cerinţele cât mai multor probleme.

Se va încerca şi găsirea unui mod de interpretare al rezultatului, în ideea că la un moment de timp dat, la ieşirea reţelei putem avea o combinaţie de valori active şi inactive pentru neuronii block.

Acest aspect interesant va fi studiat în vederea obţinerii unei cât mai bune recunoaşteri a tiparelor negative de către reţeaua neuronală.

4.4. Concluzii

Acest capitol s-a ocupat preponderent de aspectele teoretice ale antrenării unei reţele neuronale utilizând şi exemple negative în setul de antrenare. Au fost enunţate avantajele folosirii unui set mixt de antrenare precum şi posibilele modificări de topologie ale reţelelor neuronale care să permită o mai bună şi rapidă funcţionare a reţelei neuronale.

Punctul de plecare pentru toate ipotezele prezentate este legat de problemele antrenării unei reţele neuronale având la dispoziţie un set de antrenare restrâns. S-a arătat că în astfel de cazuri, cele mai multe dintre ele fiind speciale, raţiunile care au condus la micimea setului de antrenare ţin de condiţiile de aplicare, securitate sau resurse financiare.

Pentru a preîntâmpina aceste probleme am sugerat ca soluţie folosirea exemplelor negative în procesul de antrenare. Această practică nu este una nouă însă nu a fost exploatată suficient până acum, fiind oarecum minimizată şi nefiind folosită în mod curent.

Pentru folosirea cu succes a exemplelor negative în etapa de antrenare au fost prezentate trei noi metode de antrenare ce oferă fiecare o alternativă viabilă pentru cazurile în care setul de antrenare este incomplet sau are prea puţine date.

O primă metodă este cea în care exemplele negative se găsesc în setul de antrenare într-o proporţie ce va fi calculată în capitolul de experimente. Această metodă nu presupune modificări ale arhitecturii reţelei neuronale şi este cea mai comodă în folosire şi cu cel mai mult efect.

O reţea neuronală antrenată cu exemple negative şi pozitive este mult mai sigură pe ea în procesul de testare cu exemple pentru care nu a fost antrenată. Totodată această metodă face ca reţeaua să nu se transforme într-o reţea super confidentă fapt ce ar produce o recunoaştere masivă faţă de orice exemplu care ar fi prezent la intrare.

A doua metodă este aceea în care se foloseşte un neuron suplimentar pe stratul de ieşire pe care l-am numit neuron **block**. Neuronul **block** este responsabil direct cu depistarea exemplelor negative prezente la intrarea reţelei. Acest lucru reprezintă o îmbunătăţire majoră în ceea ce priveşte modalitatea de recunoaştere a reţelei neuronale, deoarece dacă reţeaua nu este 100% sigură cărei clase de la ieşire îi aparţine intrarea curentă, în unele cazuri ea poate fi sigură 100% că aceasta nu aparţine nici unei clase.

Ideea a venit din domeniul învăţării automate şi poate fi implementată numai dacă am acorda importanţă egală exemplelor negative ca şi celor pozitive. Modificările adusei unei reţele neuronale normale constau în adăugarea unui neuron suplimentar pe stratul de ieşire.

BUPT

Performanţa obţinută de reţea în acest caz este mult mai bună decât în cazul precedent, doar că o astfel de implementare presupune şi o modificare a arhitecturii reţelei.

Modul în care această tehnică utilizează exemplele negative în procesul de antrenare şi faptul că s-au adus şi modificări reţelei pentru a putea evidenţia clar aceste exemple, au făcut posibil definirea unei noi tehnici de învăţare şi anume: învăţarea pe baza corecţiei erorii cu exemple negative.

Ultima tehnică se doreşte a fi o extensie pentru noua metodă de antrenare prezentată. Ea se ocupă de problemele la care spaţiul exemplelor negative este mult mai mare decât acela al exemplelor pozitive. Numărul acestora este foarte mare, iar omogenitatea lor este scăzută.

Din această cauză s-a încercat adăugarea mai multor neuroni block pe stratul de ieşire fiecare dintre ei având o funcţie de activare diferită. Funcţiile de activare sunt cele care introduc nonlinearitatea într-o reţea neuronală şi sunt dependente de tipul de problemă folosit.

Din această cauză, ele sunt recomandate pentru a fi folosite într-un număr cât mai mare (un număr mare de neuroni block) deoarece ajută la depistarea cât mai multor exemple negative prezentate la intrare.

BUPT

5. EXPERIMENTE

Au fost prezentate, până în acest moment, conceptele teoretice care au stat la baza acestei teze, precum şi ideile noi menite să îmbunătăţească performanţele antrenării unei reţele neuronale. În acest capitol vor fi prezentate experimentele efectuate pentru infirmarea sau confirmarea tuturor ipotezelor.

Pentru aceasta vor fi prezentate iniţial programele de antrenare şi seturile de date folosite în cadrul experimentelor. Vor urma detaliile experimentelor , iar în final concluziile care au rezultat din acestea.

5.1 Programe de antrenare şi seturi de date folosite pentru experimente

5.1.1 Programe de antrenare a reţelelor neuronale

Experimentele următoare au fost realizate cu ajutorul a două programe ce au ca scop principal antrenarea de reţele neuronale. Fiecare dintre cele două programe a fost ales având drept scop o antrenare cât mai precisă şi mai rapidă pentru orice reţea neuronală. De asemenea datorită cantităţilor mari de date utilizate în setul de antrenare, ambele programe trebuiau să fie capabile de a gestiona seturi de date de zeci de mii de exemple (fiecare experiment a avut în medie ~ 30.000 de exemple de antrenare, care s-au tradus prin ~ 1Gb capacitate fişiere folosite).

NeuroShell 2.0

Primul program este NeuroShell versiunea 2.0 [59] , un program specializat în vederea creării şi antrenării reţelelor neuronale. Din punct de vedere al performanţelor, NeuroShell este un lider de necontestat în acest domeniu deoarece este primul program care a reuşit să se impună pe piaţă ca fiind numărul 1 de 10 ani. Cu ajutorul acestui program se pot antrena o mulţime de configuraţii de reţele neuronale într-un timp rezonabil şi cu multiple posibilităţi de procesare a datelor obţinute.

Interfaţa principală a programului este prezentată în Figura 33, şi după cum se observă, există o mare preocupare pentru permiterea diverselor operaţii de prelucrare a datelor înainte şi după antrenare, specificarea unor parametri de antrenare, diverse modalităţi de vizualizare a rezultatelor, etc.

De asemenea NeuroShell poate alege dintr-un număr foarte mare de reţele neuronale: multilayer perceptron, reţele recurente, reţele Kohonen, reţele generale de regresie.

Antrenarea unei reţele neuronale cu ajutorul programului NeuroShell este intuitivă şi cuprinde următorii paşi:

- crearea unei noi probleme(entitate ce cuprinde toate datele şi operaţiile efectuate asupra unei reţele neuronale)
- importarea datelor de intrare – acest pas poate fi făcut manual (Data Entry) sau printr-un import de fişier cu un anumit format

BUPT

prin care se poate specifica numărul de coloane de pe un rând şi delimitatorul dintre acestea (File Import)

- definirea intrărilor şi a ieşirilor – se specifică care dintre coloanele de pe un rând sunt de intrare şi care sunt de ieşire; totodată se calculează maximul şi minimul datelor ce pot apare la fiecare dintre intrări/ieşiri (Define Input/Output)
- se creează setul de antrenare şi testare (Test Set Extract); aici se poate specifica dacă se doreşte şi crearea unui set destinat testării finale (set de producţie)

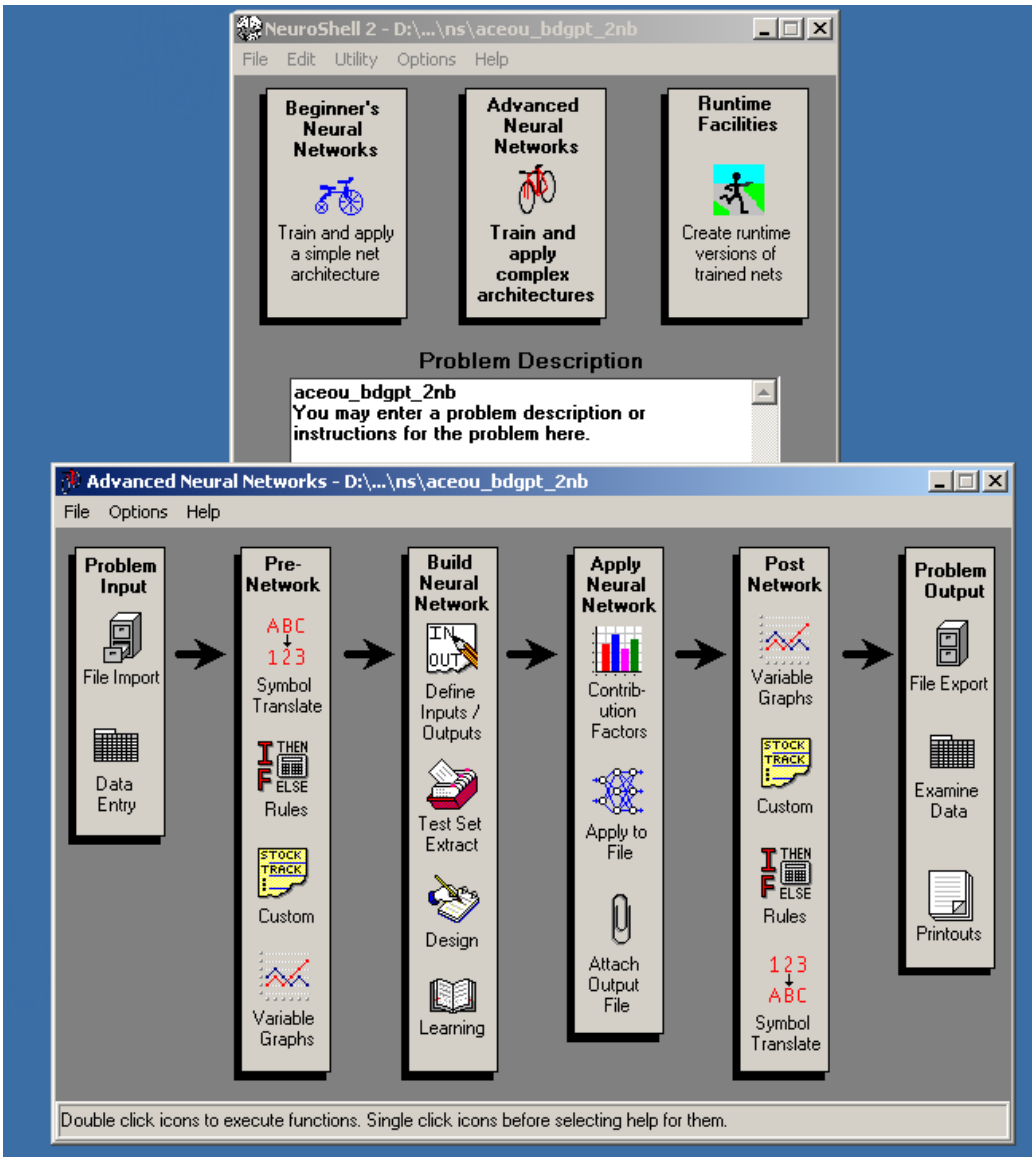


Figura 33. Interfaţa principală a programului NeuroShell

- se specifică tipul reţelei şi parametrii acesteia (număr de straturi, funcţii de activare, ponderi, etc.) (Design)
- se începe antrenarea reţelei pentru un număr de epoci sau până la îndeplinirea unor condiţii (Learning)
- se testează reţeaua antrenată cu setul de producţie sau pe un nou set de date introdus de către utilizator (Apply to File)

Unul dintre dezavantajele programului este imposibilitatea de a alege funcţii de activare diferite pentru neuronii de pe acelaşi strat. Deoarece unele dintre experimente au avut nevoie de această facilitate a fost nevoie de implementarea unui program separat ce permite alocarea unei funcţii de activare independente fiecărui neuron.

Neural Network

Al doilea program este făcut în C++ şi permite antrenarea unei reţele neuronale ce conţine pe stratul de ieşire neuroni cu funcţii de activare diferite. Interfaţa principală a programului este prezentată în Figura 34.

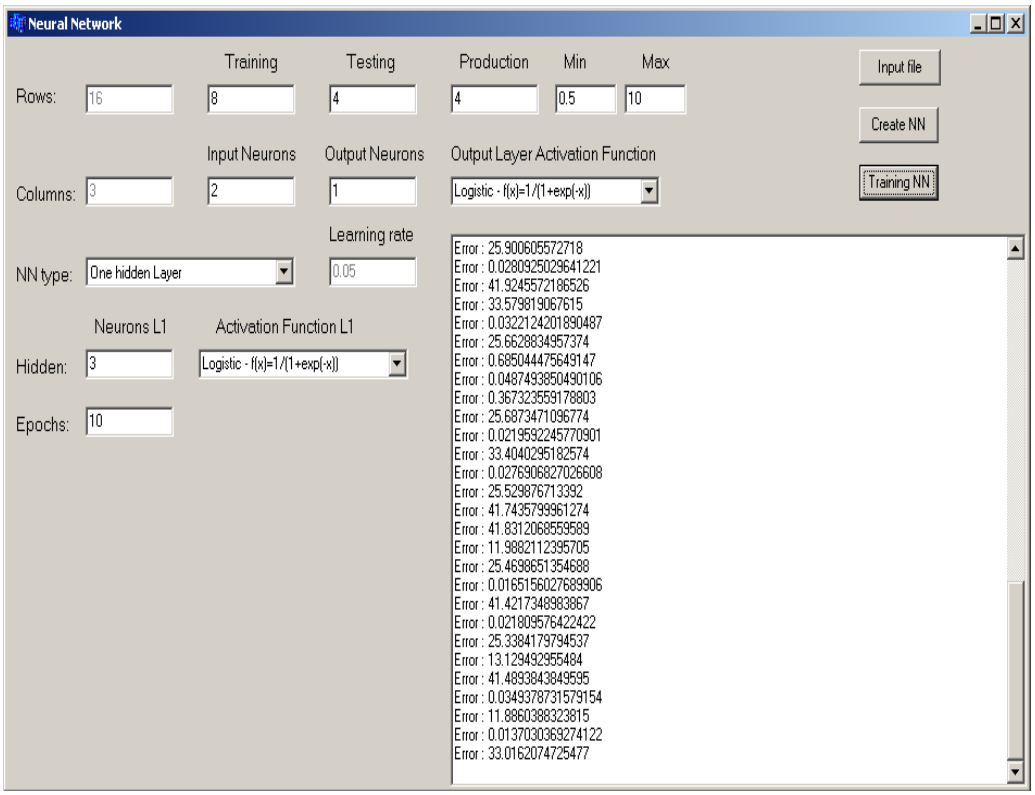


Figura 34. Interfaţa principală a programului de antrenare a reţelelor neuronale cu funcţii de activare independente

Se poate observa că programul permite antrenarea mai multor tipuri de reţele neuronale şi fiecare dintre acestea poate să fie configurată. Astfel, utilizatorul

BUPT

poate alege numărul straturilor ascunse, numărul de neuroni de pe fiecare strat, precum şi funcţiile de activare caracteristicile unui neuron.

Modul de utilizare este simplist şi nu există foarte multe operaţii de prelucrare a datelor, aşa cum întâlnim la NeuroShell. Datele de intrare pot fi citite numai dintr-un fişier care trebuie să respecte un anume format.

Programul a fost construit în scopul validării unor concepte noi şi nu are decât aplicabilitate ştiinţifică. Performanţele lui sunt bune, deşi nu este scris în limbaj asamblare şi nu conţine instrucţiuni pentru calcule vectoriale.

5.1.2 Seturi de date folosite în cadrul experimentelor

Seturile de date folosite pentru experimente sunt dintre cele mai variate ca dimensiune şi conţinut. Fiecare dintre acestea face parte dintr-o bază de date internaţională şi este special construit pentru testarea şi evaluarea unei reţele neuronale.

Am folosit mai multe seturi de date de antrenare pentru a obţine o validare clară a experimentelor. În acest fel s-a dorit să se arate că ideile propuse pentru îmbunătăţirea antrenării reţelelor neuronale sunt general valabile pentru orice domeniu de aplicabilitate. Totodată folosirea mai multor baze de date a fost necesară şi pentru validarea rezultatelor şi încercarea obţinerii unei acuităţi deosebite în evaluarea rezultatelor.

NIST 19

Un prim set de date de antrenare este NIST 19 – o bază internaţională ce conţine mai mult de 800.000 de caractere scrise de diverse persoane. [60] Am ales această bază de date deoarece recunoaşterea caracterelor cu ajutorul reţelelor neuronale este un domeniu de cercetare foarte activ şi care se bucură de multă atenţie în prezent. De asemenea, antrenarea unei reţele neuronale în vederea recunoaşterii caracterelor necesită o reţea bine antrenată şi cu o structură internă foarte mare.

Baza de date NIST 19 conţine mostre de text scrise de 4170 de persoane. Fiecare dintre aceste persoane a completat câte un formular (Figura 35) în care se pot observa mai multe numere, diverse caractere şi un text. Fiecare formular a fost scanat şi literele şi cifrele din interior au fost grupate pe categorii pentru a facilita astfel munca utilizatorului.

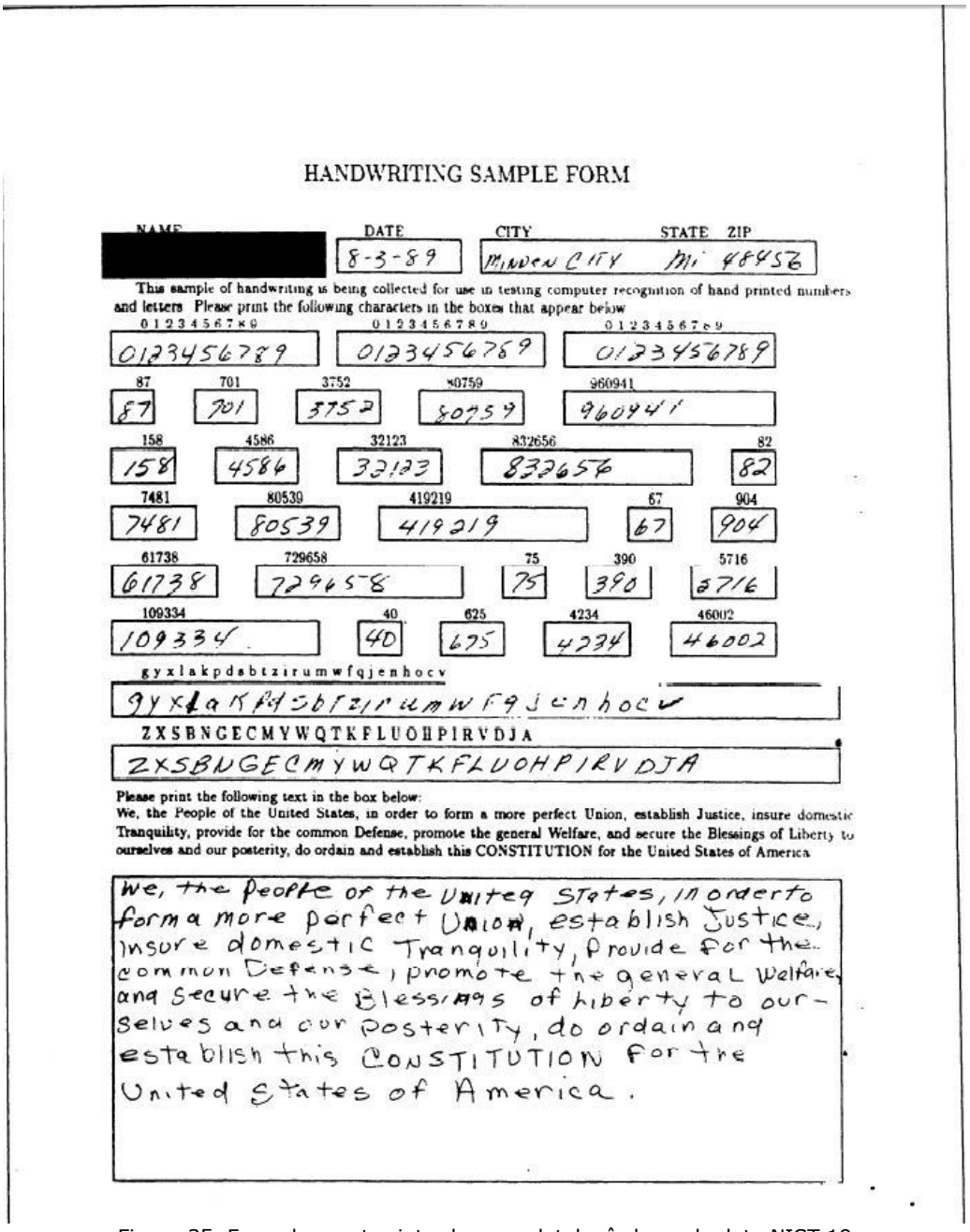
Toate aceste cifre şi litere au fost segmentate realizându-se astfel mai mult de 814255 de imagini (vezi Anexa 1). Caracterele sunt memorate în imagini de 128x128 pixeli şi se folosesc codurile ASCII pentru denumirea lor.

Experimentele din cadrul acestui capitol au avut nevoie numai de imaginile ce conţin litere şi care sunt şi cele mai multe din această bază de date.

Pentru a face posibilă o antrenare de reţea neuronală folosind aceste imagini s-a efectuat o operaţie de normalizare la dimensiunea de 32x32 (dacă erau folosite imaginile originale era nevoie de 16384 de neuroni de intrare). Chiar şi în acest caz este nevoie de 1024 de neuroni de intrare pentru a putea introduce datele în reţeaua neuronală.

După operaţia de normalizare a fost efectuată o operaţie de translaţie necesară deplasării fiecărui caracter în centrul de greutate. Acest lucru este absolut necesar pentru a putea antrena o reţea neuronală în recunoaşterea de caractere, în caz contrar, se puteau obţine rezultate eronate sau reţeaua neuronală nu era capabilă să reuşească învăţarea caracterelor.

BUPT



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789 0123456789 0123456789

0123456789 0123456789 0123456789

87 701 3752 80759 960941

87 701 3752 80759 960941

158 4586 32123 832656 82

158 4586 32123 832656 82

7481 80539 419219 67 904

7481 80539 419219 67 904

61738 729658 75 390 5716

61738 729658 75 390 5716

109334 40 625 4234 46002

109334 40 625 4234 46002

g y x l a k p d s b t z i r u m w f q j e n h o c v

g y x l a k p d s b t z i r u m w f q j e n h o c v

Z X S B N G E C M Y W Q T K F L U O H P I R V D J A

Z X S B N G E C M Y W Q T K F L U O H P I R V D J A

Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America

We, the people of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Figura 35. Formular pentru introducerea datelor în baza de date NIST 19

O altă operaţie geometrică efectuată asupra imaginilor a fost una de scalare. Fiecare caracter a fost mărit/micşorat astfel încât conturul lui să fie conţinut într-un pătrat de 32x32 pixeli. A fost nevoie de aşa ceva deoarece caracterele aveau diverse dimensiuni, datorită diverşilor autori ai acestor caractere.

O parte din imaginile obţinute în urma acestor transformări au servit ca intrare pentru reţeaua neuronală în faza de antrenare. Restul au fost folosite pentru a testa reţelele antrenate şi a vedea care sunt rezultatele obţinute în urma antrenării.

Ocean Data

Al doilea set de date folosit este de la un observator de coastă folosit într-un amplu program guvernamental de monitorizare şi măsurare a diverşi parametri din ocean.[61] Datele sunt culese odată la 20 de minute şi sunt procesate pentru a oferi valori statistice pe pagina de internet a Observatorului Martha's Vineyard.

Datele procesate sunt din 2001 până în 2008, obţinând astfel mai mult de 200.000 de măsurători asupra unor parametrii oceanografici. Aceste date au fost folosite pentru a determina gradul de risc privind apariţia unor valuri foarte mari într-o anumită perioadă a anului.

Parametrii luaţi în considerare la antrenarea reţelelor neuronale sunt următorii (pentru mai multe detalii despre formatul datelor a se vedea Anexa 2):

- data la care s-a făcut respectiva măsurătoare; data este exprimată în nr. de zile trecute de la începerea anului calendaristic
- periodicitatea apariţiei valurilor (secunde)
- direcţia de înaintare a valurilor, observată la o adâncime de 4.7 m deasupra fundului oceanului (grade)
- temperatura la 2.11 m deasupra fundului oceanului (grade Celsius)
- viteza de deplasare a valului aproape de fundul oceanului (la 3.7 m deasupra fundului oceanului) (cm/s)
- direcţia de deplasare a valului măsurată aproape de fundul oceanului
- viteza de deplasare a valului aproape de suprafaţă (la 10.7 m deasupra fundului oceanului) (cm/s)
- direcţia de deplasare a valului măsurată aproape de suprafaţa oceanului
- presiunea
- temperatura media a apei la 12.7 m adâncime
- salinitatea apei la 12.7 m adâncime
- începutul valului (între 0.05 şi 0.15 Hz)
- creasta valului (între 0.15 şi 0.25 Hz)
- perioada de repetiţie pentru începutul valului
- perioada de repetiţie pentru creasta valului
- direcţia de înaintare pentru începutul valului
- direcţia de înaintare pentru creasta valului

Neural Network Benchmarks

Este un set de date folosit pentru compararea performanţelor diverselor tipologii de reţele neuronale, dezvoltat de Universitatea Carnegie Mellon.[62] Cele mai cunoscute baze de date din acest set sunt: Sonar, Mushroom, Diabetes şi Cancer.

Acest set are o lungă istorie şi tradiţie şi există numeroase studii despre reţelele neuronale care s-au bazat pe comparaţii făcute pe acest set.[63][64][65]

BUPT

Din această cauză am găsit necesar introducerea acestui set ca şi comparaţie între diverse reţele neuronale.

Setul de date este bine construit şi are date din mai multe domenii. Fiecare dintre bazele de date prezente sunt omogene într-un anumit procent. Acest lucru este necesar deoarece presupune o antrenare diferită pentru fiecare dintre seturile de date existente.

O listare completă a bazelor de date din acest set şi câteva cuvinte despre fiecare este făcută în Anexa 3.

5.2 Experimente pentru determinarea procentului de exemple negative optim ce trebuie utilizat la antrenarea reţelelor neuronale

În capitolul 4.3.1 am explicat pe larg motivele care au condus la necesitatea aflării procentului de exemple negative care trebuie să existe într-un set de antrenare pentru o reţea neuronală. Pentru a afla cu exactitate procentul de exemple negative au fost efectuate mai multe experimente cu două mari baze de cunoştinţe. Prima dintre bazele de date este NIST 19 (baza de date ce conţine caractere), iar a doua este cu datele de la observatorul marin. Pentru fiecare experiment au fost folosite cantităţi mari de date pentru antrenare, în vederea obţinerii unui număr cât mai exact.

5.2.1 Experimente pe NIST 19

Au fost făcute 2 experimente pe această bază de date cu scopul de a afla procentul de exemple negative. Fiecare dintre aceste două experimente a încercat să găsească cât mai exact valoarea procentului dorit, singura diferenţă constând în valoarea pasului de discretizare. Astfel, experimentul 1 a lucrat cu un set de antrenare la care s-au adăugat la fiecare pas procente mari de exemple negative, ajungându-se în situaţia finală de a avea un set de antrenare care conţine mai multe exemple negative decât exemple pozitive.

Experimentul 2 a preluat datele de la Experimentul 1 şi a încercat aplicarea unei discretizări mai fine. Astfel, deşi au fost efectuate acelaşi număr de antrenări, întreg intervalul de aplicare pentru al doilea experiment a fost de numai 2 paşi de discretizare de la Experimentul 1 (pasul de discretizare pentru experimentul 2 este de 10 ori mai mic decât cel folosit la exemplul 1) .

Experimentul 1

Acest prim experiment pleacă de la un set de antrenare format numai din exemple pozitive şi adaugă incremental câte un procent de 5%(din setul iniţial) de exemple negative. Numărul exemplelor pozitive şi negative pentru fiecare antrenare este prezent în Tabelul 2. Se poate observa că s-au efectuat 23 de antrenări, scopul acestora fiind acela de a parcurge un interval cât mai mare pentru exemplele negative.

Pasul de incrementare pentru exemplele negative este de 5% (1000 de exemple) din numărul iniţial de exemple pozitive (20.000 exemple) cu care s-a pornit acest experiment.

BUPT

În aceste condiţii avem un set de antrenare care a pornit de la o valoare de 20.000 de exemple şi a ajuns la final cu 42.000 de exemple, marea majoritate fiind exemple negative.

Antrenare	Ex. pozitive (număr)	Ex. pozitive (% din total)	Ex. negative (număr)	Ex. negative (% din Ex. pozitive)
1	20.000	100	0	0
2	20.000	95.24	1000	5
3	20.000	90.91	2000	10
4	20.000	86.95	3000	15
5	20.000	83.33	4000	20
6	20.000	80.00	5000	25
7	20.000	76.92	6000	30
8	20.000	74.07	7000	35
9	20.000	71.42	8000	40
10	20.000	68.96	9000	45
11	20.000	66.66	10000	50
12	20.000	64.51	11000	55
13	20.000	62.50	12000	60
14	20.000	60.60	13000	65
15	20.000	58.82	14000	70
16	20.000	57.14	15000	75
17	20.000	55.55	16000	80
18	20.000	54.05	17000	85
19	20.000	52.63	18000	90
20	20.000	51.28	19000	95
21	20.000	50.00	20000	100
22	20.000	48.78	21000	105
23	20.000	47.61	22000	110

Tabelul 2. Procentul de exemple pozitive şi exemple negative utilizat pt. Experimentul 1

Experimentul a constat în antrenarea unei reţele neuronale în recunoaşterea caracterelor „a”, „e”, „n” şi „r”. Pentru experiment am folosit o reţea neuronală de tip multilayer perceptron cu 1 strat ascuns. Configuraţia acestei reţele este următoarea

- stratul de intrare conţine 1024 neuroni – am folosit ca intrări desene de 32x32 pixeli cu literele respective
- stratul ascuns are 700 de neuroni
- stratul de ieşire are 4 neuroni – fiecare dintre cei 4 neuroni este responsabil cu recunoaşterea unui caracter

Funcţiile de activare corespunzătore fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

Antrenarea a pornit cu un set de 4 caractere (a,e,n, şi r), fiecare având câte 5000 de instanţe. Treptat s-au adăugat la fiecare nouă antrenare câte 1000 de noi

BUPT

caractere(preponderent literele b,d,h,l şi t), acestea din urmă având rol de exemple negative. Pentru fiecare antrenare, testarea s-a făcut pe un set de 26.000 de caractere, astfel, pentru fiecare literă a alfabetului am avut câte 1000 de instanţe.

Rezultatele obţinute în cadrul acestui experiment sunt prezentate în Tabelul 3.

Antrenare	Exemple negative %	Litera „a” %	Litera „e” %	Litera „n” %	Litera „r” %	Întreg alfabetul %	Set producţie %
1	0	91.00	94.40	96.30	96.10	31.95	78.94
2	4.76	90.80	92.60	95.20	95.80	54.23	82.06
3	9.09	91.30	91.90	95.30	95.20	58.90	82.96
4	13.05	90.30	92.80	94.80	95.50	62.31	83.33
5	16.67	88.40	91.20	94.30	95.20	63.19	82.68
6	20.00	88.70	90.80	94.20	95.00	66.19	82.76
7	23.08	88.60	89.50	93.70	95.00	67.34	82.54
8	25.93	88.60	90.80	93.10	94.90	69.00	82.81
9	28.58	89.20	90.20	93.00	94.70	68.42	82.79
10	31.04	88.10	88.90	92.60	94.50	72.03	82.28
11	33.33	89.20	88.70	93.30	94.00	70.12	82.50
12	35.49	87.30	88.80	92.90	94.70	70.53	82.23
13	37.50	88.70	88.50	92.40	93.70	71.17	82.20
14	39.40	88.20	87.90	92.40	93.40	72.58	81.95
15	41.18	87.80	88.60	92.20	93.60	72.53	82.02
16	42.86	88.10	89.20	92.20	93.70	73.34	82.24
17	44.44	89.20	89.30	91.20	94.40	71.07	82.38
18	45.95	88.00	87.30	90.30	93.90	74.07	81.53
19	47.37	87.60	87.70	91.00	92.90	74.43	81.54
20	48.72	86.90	88.30	91.40	92.90	75.21	81.60
21	50.00	88.10	88.70	91.40	93.10	75.44	81.98
22	51.22	86.70	86.80	90.50	92.60	75.20	81.04
23	52.39	86.80	88.10	90.50	92.60	75.56	81.32

Tabelul 3. Rezultatele obţinute pe baza de date NIST 19 în cadrul experimentului 1

Se poate observa o mică scădere a ratei de învăţare pentru caracterele „a”, „e”, „n” şi „r” odată cu introducerea exemplelor negative în setul de antrenare. Acest lucru este oarecum firesc şi natural, deoarece reţeaua neuronală nefiind antrenată cu exemple negative căuta să includă orice caracter prezent la intrare într-una din aceste clase. Din această cauză rata de succes este foarte mare, deoarece foarte puţine caractere sunt „lăsate pe dinafară”.

Această abordare este oarecum similară cu ideea de a antrena o reţea neuronală să recunoască un singur caracter. Seturile de antrenare şi testare ar trebui să cuprindă numai acel caracter, lucru ce ar determina o recunoaştere aproape de 100% a acelui caracter de către reţea.

În acest caz problemele apar atunci când în setul de testare sunt caractere diferite de cele cu care a fost antrenată reţeaua, şi nu mă refer la feluri diferite de scriere a unui caracter ci la clase diferite de caractere. După cum se observă din tabel, reţeaua neuronală care a fost antrenată numai cu caracterele „a”, „e”, „n” şi „r” întâmpină mari dificultăţi când la intrare apar şi alte clase de caractere. Astfel, deşi are rate de recunoaştere între 91% şi 96.3% pentru caracterele „a”, „e”, „n” şi

BUPT

„r” pe setul total de litere ale alfabetului nu obţine decât 31.95%. Putem concluziona că multe dintre caracterele străine reţelei au fost „recunoscute” ca făcând parte din clasele cunoscute.

Situaţia se schimbă foarte mult atunci când în setul de antrenare sunt introduse şi exemple negative, clase de caractere care trebuie recunoscute de către reţea ca nefăcând parte din clasele „a”, „e”, „n” şi „r”.

Astfel, pentru un set de testare în care avem incluse toate literele alfabetului, procentul de recunoaştere creşte odată cu creşterea numărului de exemple negative cunoscute.(Figura 37) Pe de altă parte, asistăm la o scădere a procentelor de recunoaştere a caracterelor principale(„a”, „e”, „n” şi „r”), scădere situată între 4 şi 6 procente.(Figura 36)

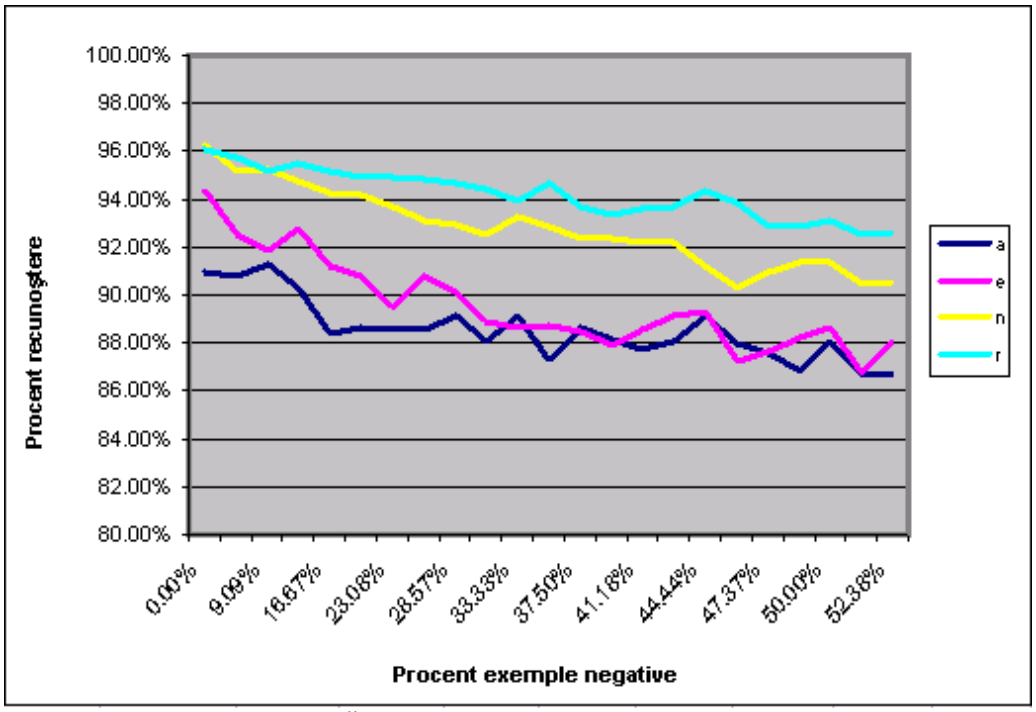


Figura 36. Reprezentarea grafică a procentelor de recunoaştere pentru caracterele „a”, „e”, „n” şi „r”, de către reţeaua neuronală de la experimentul 1

Ne aflăm astfel într-o situaţie în care mărirea unei variabile conduce la micşorarea altei variabile. Din această cauză este necesară, construirea unei formule care să reflecte cât mai bine o situaţie reală.

Pentru a afla procentul de exemple negative optim, am conceput un set de producţie în care numărul de exemple negative deţine o fracţie egală cu fiecare clasă ce trebuie recunoscută de către reţeaua neuronală.

Pentru exemplificare, dacă considerăm exemplul 1, avem 4 clase ce trebuie recunoscute de către reţea, deci procentul de exemple negative trebuie să fie egal cu procentul de exemple pozitive pentru fiecare din cele 4 clase. În acest caz fiecare din cele 4 clase, precum şi totalul exemplelor negative deţin câte 1/5 din setul de antrenare.

BUPT

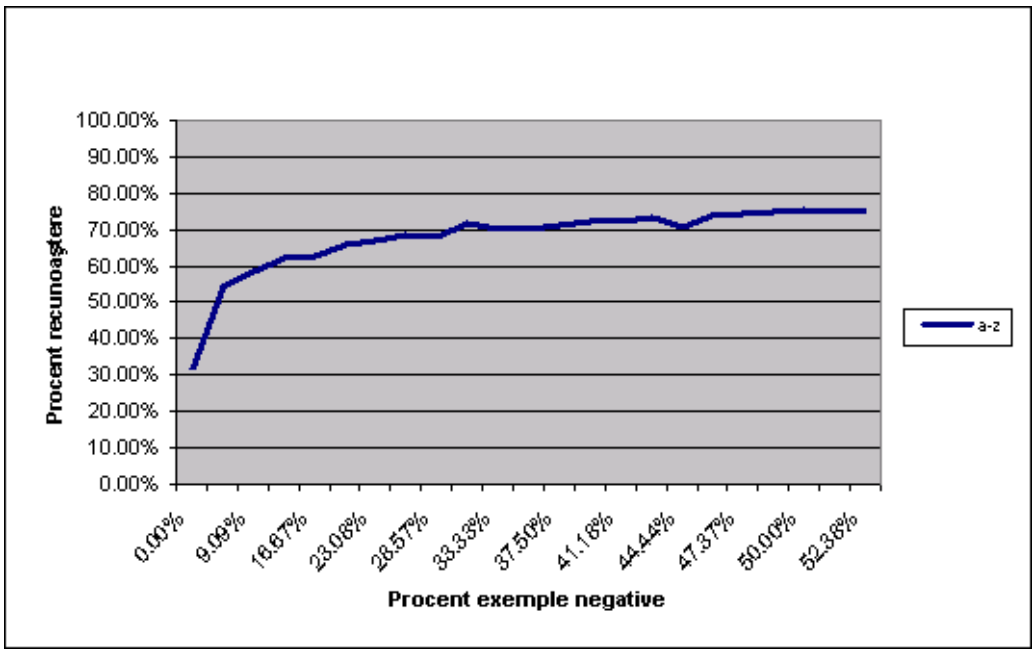


Figura 37. Reprezentarea grafică a procentelor de recunoaştere pentru întreg alfabetul, de către reţeaua neuronală de la experimentul 1

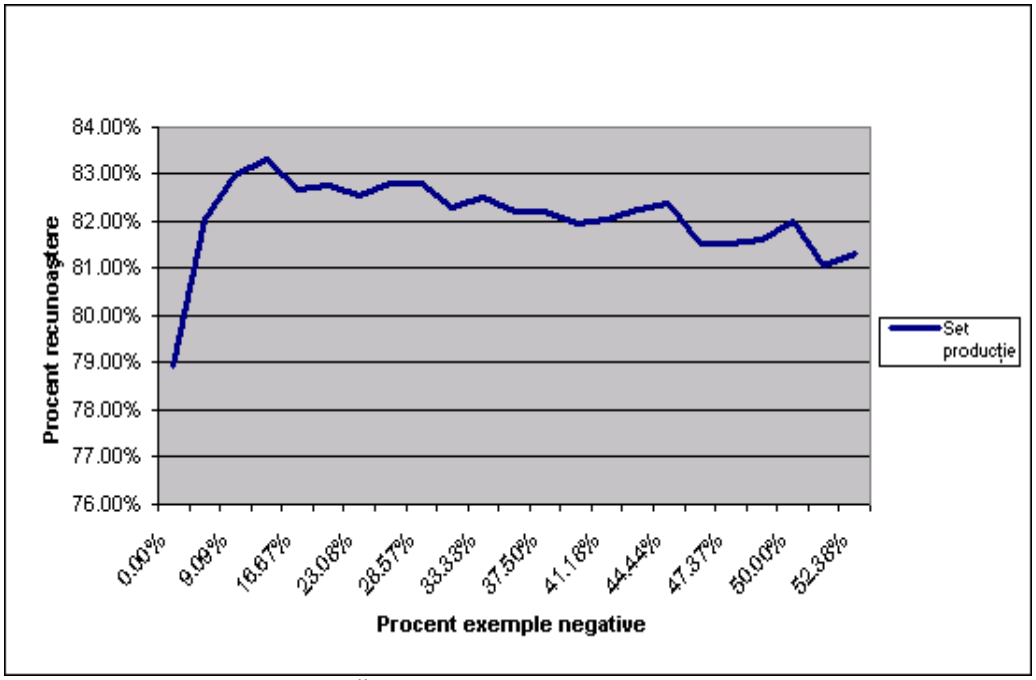


Figura 38. Reprezentarea grafică a procentelor de recunoaştere pentru setul de producţie, de către reţeaua neuronală de la experimentul 1

BUPT

Formula generală de calcul a procentului total pentru setul de producţie este următoarea:

$$P_{prod} = \frac{\sum_{clase rec.} P_{recunoastere} + P_{rec.clase neg.}}{Nr.clase pozitive + 1} \quad (5.1)$$

Reprezentarea grafică a procentului de recunoaştere pentru setul de producţie se poate observa în Figura 38. Uitându-ne atent la acest grafic observăm că are un maxim pentru un set de antrenare format din 13.04% exemple negative(sau 15% din numărul de exemple pozitive).

Acesta este un procent maxim în situaţia în care am folosit un set de exemple negative ce creşte la fiecare pas cu 5% din setul iniţial de exemple pozitive. Pentru a vedea o valoare mai exactă a acestui procent este necesar un al doilea experiment care să utilizeze un pas de discretizare mai fin.

Concluzii

În urma efectuării experimentului 1 (antrenarea unei reţele neuronale în vederea recunoaşterii caracterelor), au fost formulate următoarele concluzii:

- ✓ utilizarea într-un set de antrenare a cel puţin 5% exemple negative îmbunătăţeşte considerabil performanţele reţelei neuronale (cu până la 100% pentru un set de testare ce conţine şi tipare negative)
- ✓ pentru definirea unui set de producţie independent de problemă putem utiliza formula 5.1
- ✓ dacă se doreşte un maxim de performanţă pentru o reţea neuronală, trebuie să utilizăm un număr de exemple pozitive cuprins în intervalul [10% - 20%] din numărul de exemple pozitive

Experimentul 2

Experimentul 2 are ca punct de plecare experimentul 1 şi încearcă găsirea unei valori mai exacte pentru procentul de exemple negative care trebuie utilizat în problema recunoaşterii de caractere din baza de date NIST 19.

Aşa cum s-a observat (vezi Figura 38 şi Tabelul 3) procentul optim de exemple negative găsit în cadrul experimentului 1 este de 13.05% din numărul total de exemple sau 15% din exemplele pozitive. Pentru găsirea unei valori mai exacte, o să antrenăm reţele neuronale ce au setul de exemple negative cuprins între valorile 9.09% şi 16.66% din numărul total de exemple sau 10% şi 20% din exemplele pozitive.

Reţeaua neuronală este identică cu cea de la exemplul 1, şi la fel este şi setul de testare cu toate literele alfabetului (26.000 de caractere) respectiv formula pentru calculul procentului unui set de producţie (vezi formula 5.1)

Pasul de discretizare folosit este de 0.5%, de 10 ori mai mic decât pasul folosit la experimentul 1. Prezentăm direct rezultatele obţinute în cadrul experimentului 2 în Tabelul 4.

Se poate observa că procentele de recunoaştere sunt diferite faţă de cele obţinute la experimentul 1. Acest fapt este datorat iniţializării ponderilor cu valori aleatoare, precum şi faptul că setul de antrenare este obţinut, de asemenea,

BUPT

aleator. Cu toate acestea procentele de recunoaştere obţinute la toate categoriile se situează în acelaşi interval cu procentele obţinute la experimentul 1.

Similar cu exemplul 1, procentul de recunoaştere pentru literele „a", „e", „n" şi „r" scade odată cu creşterea numărului de exemple negative folosit la antrenarea reţelei. De asemenea, procentul de recunoaştere pentru întregul alfabet creşte dacă numărul exemplelor negative folosite la antrenare creşte.

Procentul de recunoaştere pentru setul de producţie are o tendinţă crescătoare până la valoarea de 83.39% (obţinută pentru un set de antrenare ce conţine aproximativ 16% exemple negative), după care tendinţa este descrescătoare.

Se verifică în acest fel datele experimentului 1, care precizau că procentul maxim de recunoaştere obţinut pentru un set de producţie se obţine pentru un număr de exemple negative situat într-un interval de 9%-17%.

Antrenare	Exemple negative %	Litera „a" %	Litera „e" %	Litera „n" %	Litera „r" %	Întreg alfabetul %	Set producţie %
1	9.09	91.30	93.30	95.30	95.20	55.44	83.05
2	9.50	90.70	93.40	95.10	95.60	56.34	82.93
3	9.91	91.60	93.20	94.70	95.40	57.25	83.09
4	10.31	91.20	93.20	95.60	95.50	56.03	83.21
5	10.71	89.70	93.50	95.60	95.30	58.56	83.00
6	11.11	90.50	92.10	94.90	95.20	57.62	82.76
7	11.50	90.30	93.70	94.50	95.60	57.62	83.17
8	11.89	91.00	91.90	95.20	95.80	58.40	83.15
9	12.28	90.80	93.00	95.30	95.40	57.08	83.21
10	12.66	90.70	92.70	94.80	95.20	58.18	83.03
11	13.04	90.10	92.50	95.30	94.90	60.89	83.10
12	13.42	89.20	92.10	94.90	95.20	60.33	82.91
13	13.79	91.00	92.20	95.10	95.50	58.47	83.26
14	14.16	90.30	92.10	94.20	95.10	61.44	83.01
15	14.53	90.70	93.10	93.90	94.90	60.02	83.21
16	14.89	89.70	92.00	95.00	95.10	62.32	83.07
17	15.25	89.40	92.80	95.20	95.00	60.57	83.14
18	15.61	90.80	92.10	94.80	95.10	60.91	83.22
19	15.97	90.70	93.20	95.00	94.80	59.80	83.39
20	16.32	89.10	93.20	94.10	95.20	61.57	83.02
21	16.67	89.20	92.50	94.00	94.90	61.74	82.93

Tabelul 4. Rezultatele obţinute pe baza de date NIST 19 în cadrul experimentului 2

În Figura 39 este reprezentat graficul evoluţiei procentului de recunoaştere pe setul de producţie în funcţie de numărul de exemple negative. Zonele de creştere şi scădere a procentului sunt vizibile mai clar în figură şi se poate observa şi zona de maxim obţinută pentru procentul de recunoaştere.

La finalul celor două experimente desfăşurate pe setul de date NIST 19 am determinat procentul de exemple negative optim pentru antrenare ca fiind 16%. Acelaşi procent, de data aceasta exprimat ca număr de exemple negative din numărul total de exemple pozitive este de 19%.

BUPT

Cu toate acestea la o privire atentă asupra graficelor de evoluţie a procentului de recunoaştere pe seturile de producţie (Figura 38,Figura 39) observăm o oscilare a acestui procent de-a lungul intervalului parcurs. Acest fapt ne determină să recomandăm un interval în care se află procentul optim de exemple negative ce trebuie folosit, în detrimentul unei singure valori.

Astfel, putem spune că intervalul în care este situat procentul optim de exemple negative este 15.5%-16% din numărul total de exemple, sau 18%-19% din numărul total de exemple pozitive.

Pentru verificarea acestor procente se vor efectua experimente de găsire a procentului optim de exemple negative şi pe un alt set de date total diferit de setul de date NIST19.

Numai după aceste experimente putem concluziona intervalul exact în care se găseşte procentul optim de exemple negative.

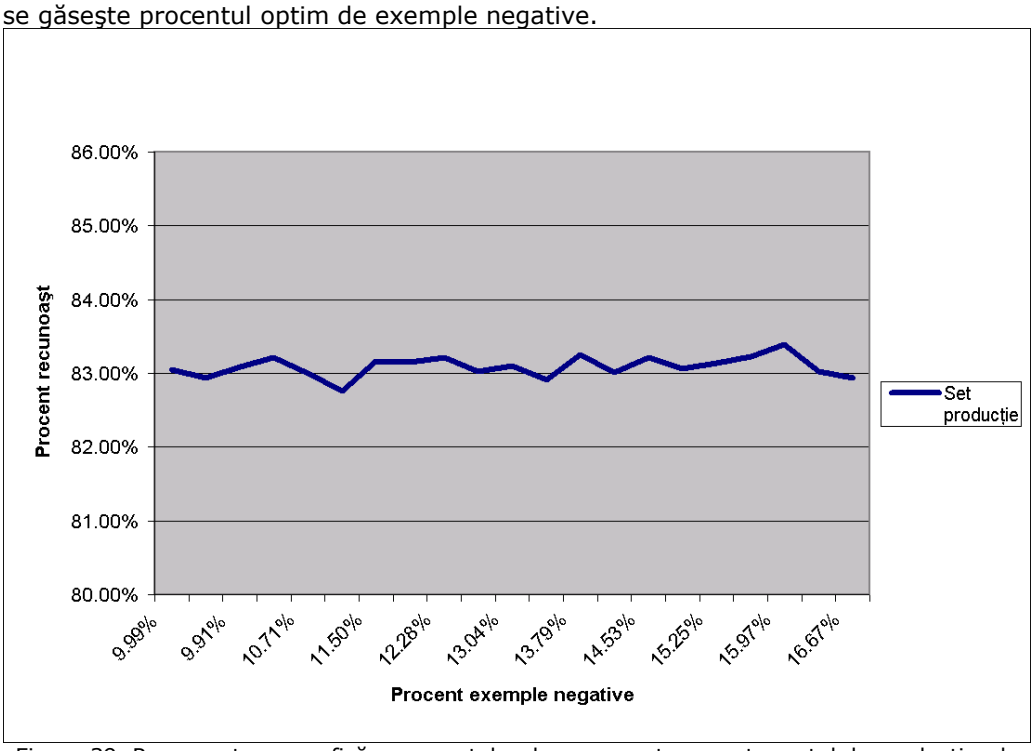


Figura 39. Reprezentarea grafică a procentelor de recunoaştere pentru setul de producţie, de către reţeaua neuronală de la experimentul 2

Concluzii

Concluziile care au fost formulate în urma realizării experimentului 2 (antrenarea unei reţele neuronale pentru recunoaştere de caractere, cu o precizie mai mare în definirea procentului de exemplelor negative din setul de antrenare) sunt prezente în cele ce urmează:

- ✓ procentului de exemple negative (pentru problema recunoaşterii caracterelor) din setul de antrenare este localizat în intervalul [15.5% - 16%] din numărul total de exemple, sau [18% - 19%] din numărul de exemple pozitive

BUPT

5.2.2 Experimente pe setul de date Ocean Data

La fel ca şi în capitolul anterior, s-a încercat găsirea procentului optim de exemplele negative ce trebuie utilizate pentru rezolvarea unei probleme cu ajutorul reţelelor neuronale. În acest experiment am schimbat natura problemei; în locul recunoaşterii de caractere s-a apelat de această dată la estimarea unor mărimi pe baza unor date culese anterior.

Şi de această dată au fost efectuate două experimente, primul a încercat să localizeze intervalul în care se află procentul de exemple negative optim, iar al doilea experiment este pentru calcularea exactă a acestui procent.

Experimentul 3

Pentru următoarele 2 experimente avem un set de antrenare mai mic decât cel folosit la recunoaşterea de caractere. Setul de antrenare iniţial este format numai din exemple pozitive şi se adaugă incremental câte un procent de 5%(din setul iniţial) de exemple negative. Numărul exemplelor pozitive şi negative pentru fiecare antrenare este prezent în Tabelul 5. La fel ca la experimentul 1 avem 23 de antrenări care reuşesc să parcurgă un interval al exemplelor negative situat între 0% şi 52.39% din totalul exemplelor din setul de antrenare, sau între 0% şi 110% din numărul de exemple pozitive utilizat.

Antrenare	Ex. pozitive (număr)	Ex. pozitive (% din total)	Ex. negative (număr)	Ex. negative (% din Ex. pozitive)
1	13.000	100	0	0
2	13.000	95.24	650	5
3	13.000	90.91	1300	10
4	13.000	86.95	1950	15
5	13.000	83.33	2600	20
6	13.000	80.00	3250	25
7	13.000	76.92	3900	30
8	13.000	74.07	4550	35
9	13.000	71.42	5200	40
10	13.000	68.96	5850	45
11	13.000	66.66	6500	50
12	13.000	64.51	7150	55
13	13.000	62.50	7800	60
14	13.000	60.60	8450	65
15	13.000	58.82	9100	70
16	13.000	57.14	9750	75
17	13.000	55.55	10400	80
18	13.000	54.05	11050	85
19	13.000	52.63	11700	90
20	13.000	51.28	12350	95
21	13.000	50.00	13000	100
22	13.000	48.78	13650	105
23	13.000	47.61	14300	110

Tabelul 5. Procentul de exemple pozitive şi exemple negative utilizat pt. Experimentul 3

BUPT

Pasul de incrementare pentru exemplele negative este de 5% (650 de exemple) din numărul iniţial de exemple pozitive (13.000 exemple) cu care s-a pornit acest experiment.

În aceste condiţii avem un set de antrenare care a pornit de la o valoare de 13.000 de exemple şi a ajuns la final cu 27.300 de exemple, marea majoritate fiind exemple negative.

Experimentul constă în determinarea gradului de risc privind apariţia unor valuri foarte mari într-o anumită perioadă a anului. Reţeaua neuronală are ca intrări diverşi parametrii oceanografici (vezi Anexa 2) măsurabili, iar la ieşire avem una din cele 3 categorii de risc privind înălţimea valului:

- risc inexistent – cuprinde valuri a căror înălţime este cuprinsă în intervalul [0 , 0.5] m
- risc scăzut – cuprinde valuri a căror înălţime este cuprinsă în intervalul [0.5 , 1] m
- risc mediu – cuprinde valuri a căror înălţime este cuprinsă în intervalul [1 , 1.5] m

Dacă nici una din ieşirile reţelei nu este activă se consideră că valul are o înălţime cu un grad ridicat de risc. Exemplele negative cu care a fost antrenată reţeaua sunt numai din această ultimă categorie.

Configuraţia reţelei neuronale este următoarea

- stratul de intrare conţine 17 neuroni – am folosit ca intrări toţi parametrii din Anexa 2 mai puţin parametrul 8
- stratul ascuns are 160 de neuroni
- stratul de ieşire are 3 neuroni – corespunzătoare celor 3 nivele de risc pentru valuri

Funcţiile de activare corespunzătoare fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară $f(x) = x$.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

S-au efectuat 23 de antrenări, având pentru prima antrenare un set format numai din exemple pozitive(13.000) şi pentru ultima un set format din 13.000 exemple pozitive şi 14.300 exemple negative.

Rezultatele obţinute în cadrul acestui experiment sunt prezentate în Tabelul 6. La fel ca şi la precedentele experimente, rata de învăţare pentru cele 3 categorii de risc scade odată cu creşterea procentului de exemple negative introduse în setul de antrenare. Scăderea nu este aşa de accentuată ca în cazul experimentului 1, observându-se că avem chiar o categorie de risc (cea cu risc scăzut) la care procentul de recunoaştere a rămas oarecum constant.

Acest fapt este foarte important, deoarece arată diversitatea datelor faţă de cele de la experimentele precedente. Acest aspect se poate observa în Figura 40 unde ratele de recunoaştere pentru cele trei categorii de risc pornesc de la procente foarte diferite între ele. În timp ce pentru risc inexistent avem procente de recunoaştere situate în jurul valorii de 85%, pentru risc scăzut avem 98% iar pentru risc mediu avem 50%.

BUPT

Antrenare	Exemple negative %	Risc inexistent %	Risc scăzut %	Risc mediu %	Set test %	Set producţie %
1	0	87.07	97.92	55.71	43.73	60.17
2	4.76	86.21	94.44	50.00	85.55	79.64
3	9.09	84.48	96.53	54.29	86.82	81.29
4	13.05	85.34	98.61	54.29	88.09	82.29
5	16.67	83.62	95.14	52.86	90.00	81.38
6	20.00	82.76	94.44	48.57	90.82	80.37
7	23.08	81.03	97.22	50.00	91.55	81.12
8	25.93	82.76	94.44	51.43	91.73	81.48
9	28.58	80.17	95.14	50.00	91.91	80.48
10	31.04	80.17	96.53	51.43	91.73	81.42
11	33.33	78.45	97.22	50.00	91.27	80.93
12	35.49	77.59	97.22	48.57	91.45	80.52
13	37.50	79.31	97.22	47.14	91.91	80.50
14	39.40	76.72	95.14	51.43	91.91	80.56
15	41.18	78.45	97.22	50.00	90.00	81.16
16	42.86	77.59	95.83	48.57	92.00	80.27
17	44.44	76.72	94.44	51.43	92.09	80.36
18	45.95	75.00	95.83	47.14	91.27	79.23
19	47.37	78.45	95.14	47.14	91.55	80.15
20	48.72	75.86	95.14	44.29	92.36	78.69
21	50.00	79.31	95.83	42.86	92.45	79.44
22	51.22	75.86	96.53	44.29	91.27	79.10
23	52.39	75.00	95.83	45.71	91.27	79.10

Tabelul 6. Rezultatele obţinute pe baza de date Ocean Data în cadrul experimentului 3

BUPT

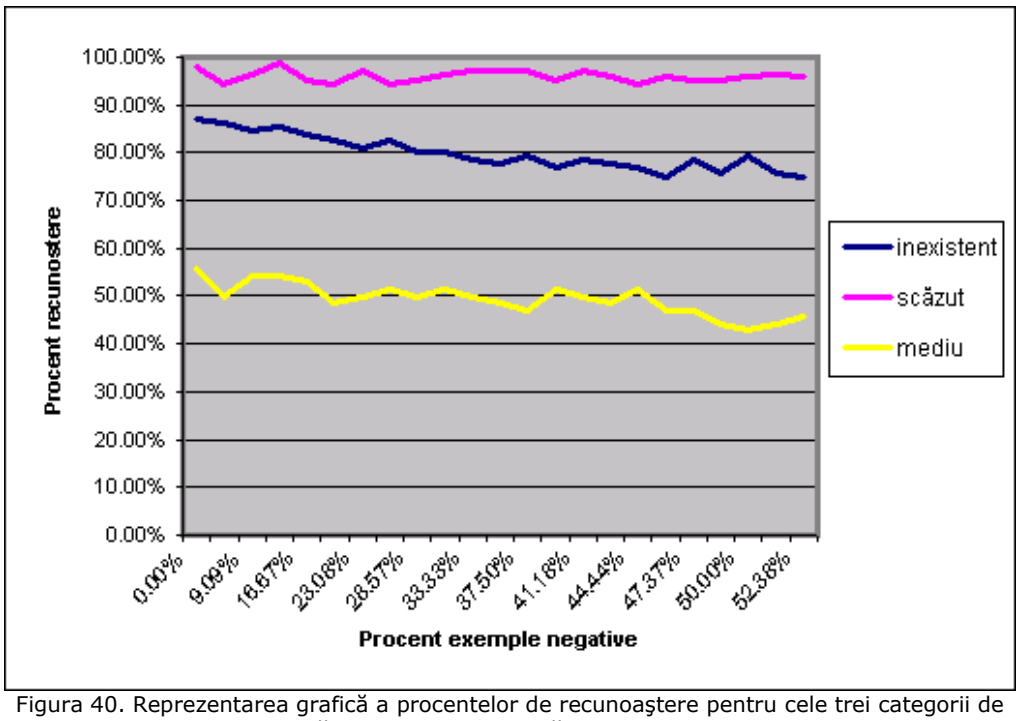


Figura 40. Reprezentarea grafică a procentelor de recunoaştere pentru cele trei categorii de risc de către reţeaua neuronală de la experimentul 3

Graficul setului de testare (Figura 41) se prezintă similar cu cel obţinut la Experimentul 1. Observăm şi aici o creştere accentuată a procentului de recunoaştere pentru primele procente de exemple negative introduse în setul de antrenare şi o stagnare a acestui procent odată cu depăşirea procentului de 30% exemple negative în setul de antrenare.

În privinţa setului de producţie acesta pare identic cu cel generat la Experimentul 1.(Figura 38 şi Figura 42) Acest fapt conduce la concluzia că procentele de exemple negative folosite într-o antrenare de reţele neuronale trebuie să fie la fel indiferent de natura aplicaţiei sau de datele folosite de aceasta.

Procentul maxim de recunoaştere pentru un set de producţie este obţinut pentru o valoare de 13% exemple negative conţinute în setul de antrenare. Dorim o valoare mai exactă a acestui proces pentru setul de date Ocean’s Data, şi de aceea o să executăm încă un experiment la care intervalul de exemple negative folosite la antrenare este [9.09 , 16.67]. Procentele acestea sunt echivalente cu un set de exemple negative situat între 10% şi 20% din numărul de exemple pozitive din setul de antrenare. O să împărţim acest interval în 20, rulând pentru fiecare valoare o antrenare a reţelei neuronale.

BUPT

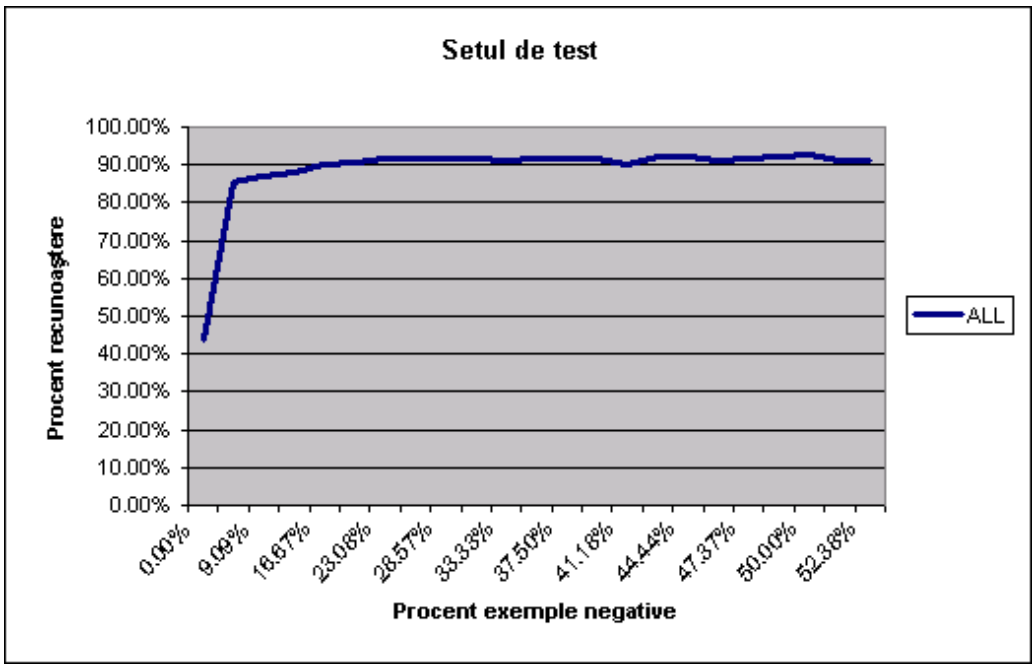


Figura 41. Reprezentarea grafică a procentelor de recunoaştere pentru setul de test, de către reţeaua neuronală de la experimentul 3

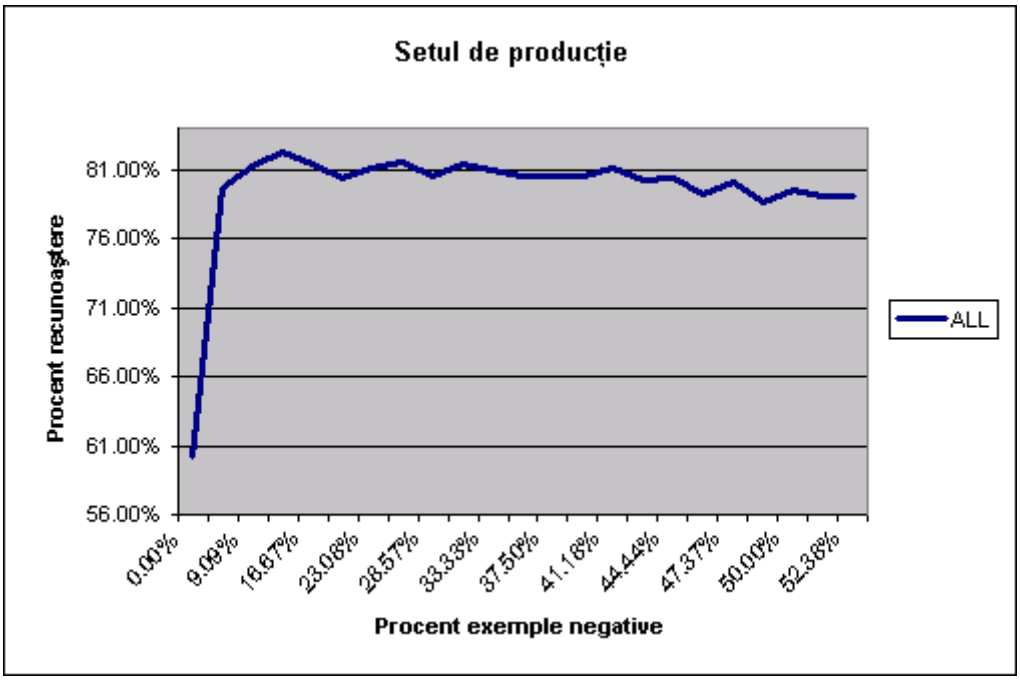


Figura 42. Reprezentarea grafică a procentelor de recunoaştere pentru setul de producţie, de către reţeaua neuronală de la experimentul 3

BUPT

Concluzii

În urma efectuării experimentului 3 (antrenarea unei reţele neuronale pentru detectarea unor valuri periculoase), au fost formulate următoarele concluzii:

- ✓ şi pentru această antrenare, procentul de exemple negative optim rămâne situat în intervalul [10% - 20%] din numărul de exemple pozitive
- ✓ recunoaşterea pe setul de producţie prezintă valori şi formă asemănătoare cu cea de la experimentul 1

Experimentul 4

Experimentul 4 are ca punct de plecare experimentul 3 şi încearcă găsirea unei valori mai exacte pentru procentul de exemple negative care trebuie utilizat în problema recunoaşterii unor tipologii de valuri din baza de date Ocean's Data.

La fel ca şi la Experimentul 2 trebuie să căutăm valoarea exactă a procentului dorit în intervalul [9.09% , 16.67%] – procente din setul de testare, sau [10% , 20%] dacă exprimăm în funcţie de procentele pozitive.

Reţeaua neuronală folosită în acest experiment este identică cu cea de la Experimentul 3. Seturile de antrenare au fost generate astfel încât să avem o împărţire a intervalului de căutare în 20 de părţi. Pentru aceasta pasul de discretizare este de 0.5 % (exemple negative din exemple pozitive), de 10 ori mai mic decât pasul folosit la experimentul 3. Prezentăm direct rezultatele obţinute în cadrul experimentului 4 în Tabelul 7.

Antrenare	Exemple negative %	Risc inexistent %	Risc scăzut %	Risc mediu %	Set test %	Set producţie %
1	9.09	80.17	96.53	55.71	86.18	80.08
2	9.50	79.31	97.22	54.29	87.27	80.24
3	9.91	81.90	97.92	51.43	87.55	80.34
4	10.31	81.03	98.61	54.29	84.00	79.91
5	10.71	81.90	98.61	51.43	85.64	80.06
6	11.11	81.90	98.61	50.00	87.27	79.96
7	11.50	79.31	98.61	47.14	86.45	78.54
8	11.89	79.31	98.61	51.43	87.36	79.93
9	12.28	78.45	97.22	58.57	90.00	81.74
10	12.66	80.17	97.92	54.29	88.00	80.66
11	13.04	80.17	97.92	55.71	87.82	80.95
12	13.42	71.55	98.61	58.57	88.09	80.01
13	13.79	77.59	98.61	50.00	87.09	78.98
14	14.16	76.72	97.92	54.29	88.91	80.28
15	14.53	78.45	97.92	51.43	87.36	79.45
16	14.89	75.00	97.22	51.43	89.00	79.29
17	15.25	78.45	97.22	61.43	88.64	82.20
18	15.61	70.69	95.83	58.57	89.64	79.91
19	15.97	75.86	95.14	51.43	87.82	78.56
20	16.32	77.59	95.83	48.57	88.00	78.55
21	16.67	79.31	95.83	51.43	89.82	80.08

Tabelul 7. Rezultatele obţinute pe baza de date Ocean Data în cadrul experimentului 4

BUPT

Şi la acest experiment procentele de recunoaştere sunt diferite faţă de cele de la experimentul 3. Aşa cum am explicat anterior acest lucru se datorează iniţializării ponderilor cu valori aleatoare, precum şi faptului că setul de antrenare este obţinut de asemenea aleator. Cu toate acestea procentele de recunoaştere obţinute la toate categoriile se situează în acelaşi interval cu procentele obţinute la experimentul 1.

Pentru o analiză mai clară, am reprezentat grafic evoluţia procentului de recunoaştere pentru setul de producţie în Figura 43.

Se poate uşor observa că procentul de recunoaştere are un maxim de 82.20% pentru valoarea de 15.25% exemple negative din setul de antrenare(numărul de exemple negative este echivalent cu 18% din numărul de exemple pozitive).

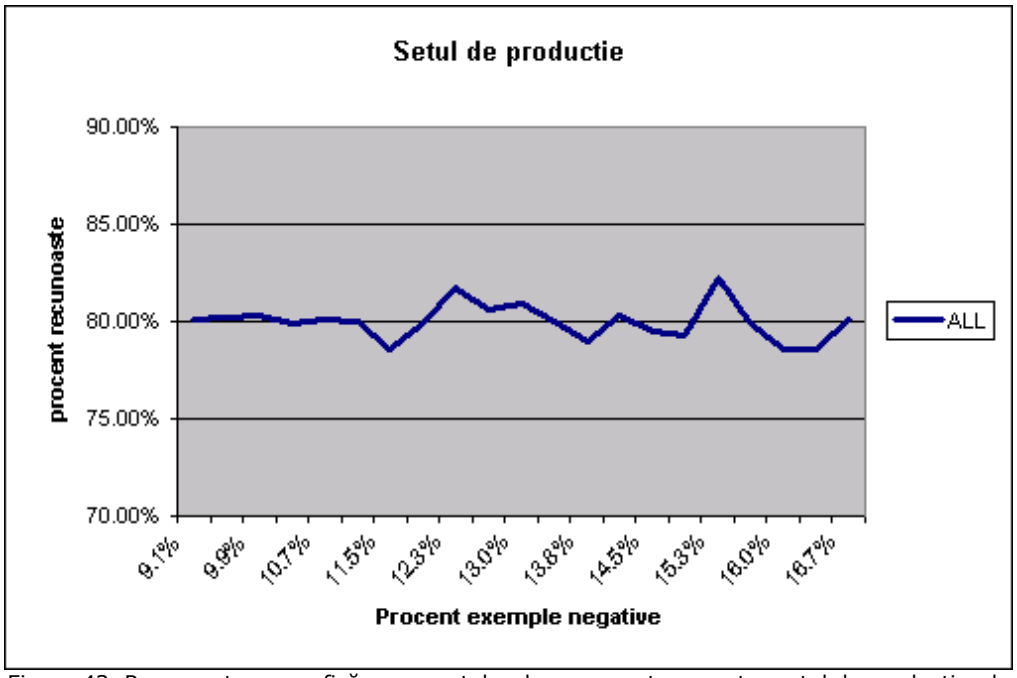


Figura 43. Reprezentarea grafică a procentelor de recunoaştere pentru setul de producţie, de către reţeaua neuronală de la experimentul 4

Procentul de recunoaştere pentru setul de producţie are o tendinţă crescătoare până la valoarea de 82.20%, după care tendinţa este descrescătoare.

Se verifică în acest fel datele experimentului 3, care precizau că procentul maxim de recunoaştere obţinut pentru un set de producţie se obţine pentru un număr de exemple negative situat într-un interval de 9%-17%.

O altă observaţie care confirmă datele obţinute la experimentul 3 este aceea că procentul de recunoaştere pentru întregul set creşte dacă numărul exemplelor negative folosite la antrenare creşte.

BUPT

Concluzii

În urma efectuării experimentului 4 (antrenarea unei reţele neuronale pentru detectarea unor valuri periculoase), au fost formulate următoarele concluzii:

- ✓ pentru această problemă, intervalul [17% - 19%] (din numărul de exemple pozitive) reprezintă o delimitare mai clară a intervalului în care este situat procentul optim de exemple negative ce trebuie utilizat la antrenare.

5.2.3 Determinarea procentului de exemple negative optim ce trebuie utilizat la antrenarea reţelelor neuronale

Pentru determinarea procentului de exemple negative optim ce trebuie utilizat la antrenarea reţelelor neuronale au fost desfăşurate 4 experimente. Două dintre acestea au folosit baza de date NIST 19 (bază de date ce conţine caractere scrise de mână), iar celelalte două au folosit baza de date Ocean's Data (bază de date ce conţine diverse măsurători aplicate asupra valurilor). Acest fapt a asigurat experimentelor noastre o larga diversitate a datelor, fapt ce conduce la o relevanţă semnificativă a rezultatelor obţinute.

Pentru găsirea procentului de exemple negative, pe fiecare bază de date s-au efectuat două experimente: primul experiment încearcă localizarea acestui procent la nivel de interval, iar al doilea experiment reuşeşte găsirea unei valori pentru acest procent (prin împărţirea mai fină a intervalului de la primul experiment).

Rezultatele obţinute pe cele două baze de date sunt(menţionăm că toate valorile procentuale reprezintă procente din setul total de antrenare):

- NIST 19 – intervalul găsit este [9% - 17%], procentul găsit fiind de 16%
- Ocean's Data – intervalul găsit este [9% - 17%], procentul găsit fiind de 15.25%

Cele două valori găsite sunt foarte asemănătoare chiar dacă seturile de antrenare sunt total diferite. Pentru a întări această observaţie prezentăm mai jos câteva comparaţii între rezultatele obţinute pe cele două seturi de valori experimentale furnizate de experimente.

În Figura 44 avem desenate graficele obţinute de setul de testare pentru ambele baze de date. Chiar daca valorile procentuale sunt diferite, forma celor două este identică. Acest lucru exprimă clar tendinţa de creştere a procentului de recunoaştere pentru un set de testare, atunci când procentul de exemple negative din setul de antrenare creşte. Creşterea aceasta este asimptotică, fiind mai accentuată pentru valori ale procentului de exemple negative până la 20% din totalul exemplelor din setul de antrenare.

BUPT

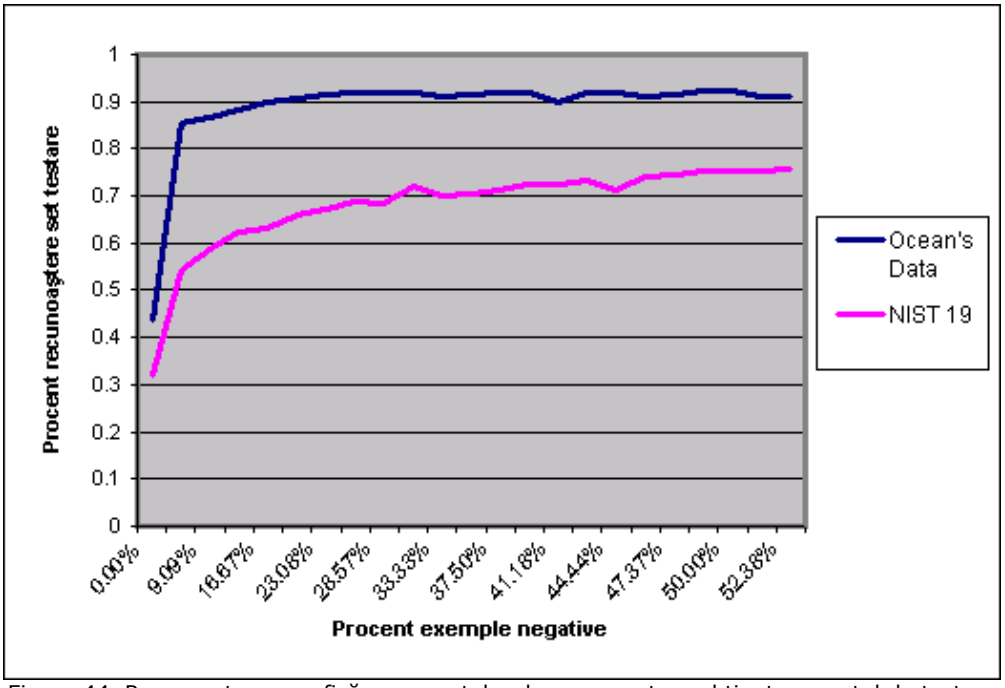


Figura 44. Reprezentarea grafică a procentelor de recunoaştere obţinute pe setul de testare pentru cele două seturi de date

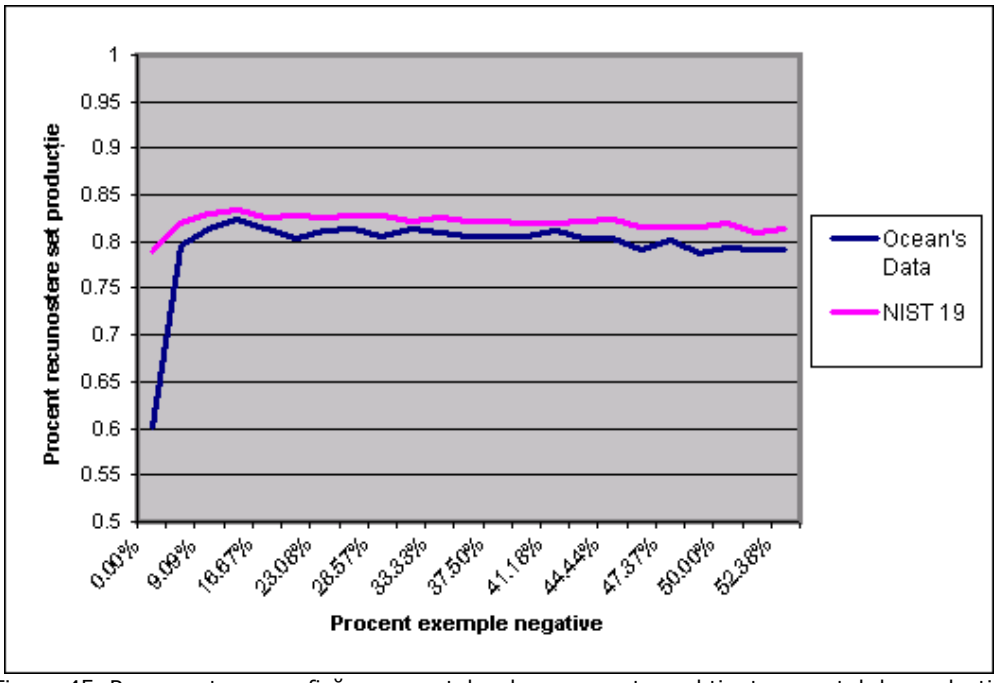


Figura 45. Reprezentarea grafică a procentelor de recunoaştere obţinute pe setul de producţie pentru cele două seturi de date

O altă comparaţie interesantă este făcută pe procentele de recunoaştere obţinute pe un set de producţie. Spre deosebire de setul de testare, cel de producţie a fost alcătuit după o formulă propusă de autor (5.1) şi redă foarte fidel distribuţia unor exemple din clase necunoscute cu exemplele din clase cunoscute de către reţeaua neuronală.

Astfel, în Figura 45 se pot observa asemănările dintre cele două grafice pentru cele două seturi. Deşi pentru procentele de recunoaştere obţinute pe setul de testare avem o diferenţă de până la 30% (pentru anumite configuraţii), pentru setul de producţie, această diferenţă nu este mai mare de 4%. În acest fel obţinem o validare certă pentru formula ce descrie setul de producţie.

În plus se poate observa cum procentul de recunoaştere variază în limita a 5% pentru valori ale setului de antrenare cu conţinut mai mare de 5% exemple negative. În acest context, putem spune că ponderea cea mai mare asupra variaţiei procentului de antrenare este dată de folosirea exemplelor negative în setul de testare, în detrimentul unui set de antrenare ce foloseşte numai exemple pozitive (variaţii de până la 25% a procentului de recunoaştere).

Cu toate acestea, pentru cei care doresc un maxim de randament(procent de recunoaştere) recomandăm folosirea unui procent situat în intervalul [15% - 16%] din numărul total de exemple din setul de antrenare (sau [17% - 19%] din numărul total de exemple pozitive).[84]

Concluzii

Pe baza informaţiilor prezentate anterior, pentru experimentele 1-4 au fost formulate următoarele concluzii:

- ✓ folosirea a cel puţin 5% exemple negative în setul de antrenare al unei reţele neuronale îmbunătăţeşte performanţele acesteia cu până la 25% pentru un set de testare ce conţine şi tipare negative
- ✓ pentru definirea unui set de producţie independent de problemă putem utiliza formula 5.1(graficele de la două probleme diferite au formă şi valori asemănătoare)
- ✓ dacă se doreşte un maxim de performanţă pentru o reţea neuronală, trebuie să utilizăm un număr de exemple pozitive cuprins în intervalul [17% - 19%] din numărul de exemple pozitive
- ✓ procentele de recunoaştere pe clase individuale sunt puţin mai mici (cu până la 5%) atunci când utilizăm exemple negative în setul de antrenare

5.2.4 Modelul matematic pentru procesul de antrenare cu exemple negative

În subcapitolul anterior am încercat, pe baza experimentelor, să găsim procentul optim de exemple negative ce trebuie utilizat la antrenarea unei reţele neuronale.

În continuare, vom justifica matematic afirmaţia făcută în prezenta lucrare, a faptului că procentul de exemple negative ce trebuie să folosit în cazul antrenării unei reţele neuronale trebuie să fie cuprins între 10 – 20% din totalul exemplelor utilizate pentru a obţine performanţe maxime.

BUPT

În acest scop se face apel la o tehnică de analiză regresională prin intermediul căreia sunt analizate datele rezultate din procesul de antrenare a trei reţele neuronale pentru un număr de trei seturi de date şi anume : un set de date destinat recunoaşterii literelor, a datelor furnizate de un sonar şi a datelor rezultate din analizele medicale pentru diabet. Cele trei tipuri de date vor fi în continuare numite generic litere, sonar şi diabet.

Etapele sunt următoarele:

- determinarea unui model corelaţional pentru fiecare din cele trei tipuri de date.
- demonstrarea faptului ca acestea aparţin aceleiaşi categorii de modele.
- definirea unui model general al procesului de recunoaştere cu exemple negative.
- testarea modelului pe o nouă bază de date pentru verificarea veridicităţii acestuia.

Pentru analiza acestora s-a recurs la utilitarul DataFit[88] şi Origin6.0.[89]

Analiză date de tip litere, sonar, diabet.

In cadrul prezentei analize, datele utilizate sunt cele rezultate în urma procesului de recunoaştere. Acestea se referă la procentul de exemple negative utilizat în procesul de antrenare şi la o valoare normalizată dată de raportul dintre procentul de recunoaştere obţinut cu exemple negative şi procentul de recunoaştere obţinut cu exemple pozitive. Normalizarea a fost necesară deoarece procentele de recunoaştere variază în funcţie de natura aplicaţiei şi de tiparele folosite la antrenarea reţelei.

Pentru cele trei tipuri de date, rezultatele experimentale sunt prezentate în tabelul de mai jos (Tabelul 8).

Scopul este găsirea unei funcţii de următorul tip:

Recunoaştere normalizată = f(Procent exemple negative)

În urma interpolării, pentru fiecare dintre cele 3 funcţii s-au obţinut formele din figurile următoare(Figura 46, Figura 47 şi Figura 48)

Procent exemple negative	Recunoaştere normalizată DIABET	Recunoaştere normalizată SONAR	Recunoaştere normalizată LITERE
0	1	1	1
5	1.108482	1.114285714	1.039523689
10	1.221101	1.250892857	1.050924753
15	1.307723	1.296428571	1.055611857
16	1.279962	1.313749999	1.054725108
17	1.278027	1.307142856	1.054091715
18	1.297856	1.307142856	1.053204966
19	1.32343	1.339285714	1.056371928
20	1.307462	1.349999999	1.047377755
25	1.307775	1.214285714	1.048391183
30	1.290598	1.239285714	1.045604256
35	1.283814	1.226428571	1.049024576
40	1.267533	1.244285714	1.048771219

BUPT

45	1.285275	1.246428571	1.042310616
50	1.226556	1.241071428	1.045097542
55	1.258814	1.228571428	1.041677223
60	1.235901	1.196428571	1.041297188
65	1.237989	1.210714285	1.038130225
70	1.22186	1.227678571	1.039016975
75	1.204118	1.197321428	1.041803902
80	1.212989	1.209821428	1.043577401
85	1.206205	1.166071428	1.032809729
90	1.223947	1.148214285	1.032936407
95	1.208292	1.149107142	1.033696478
100	1.193776	1.096428571	1.038510261

Tabelul 8. Valorile normalizate pentru cele 3 baze de cunoştinţe

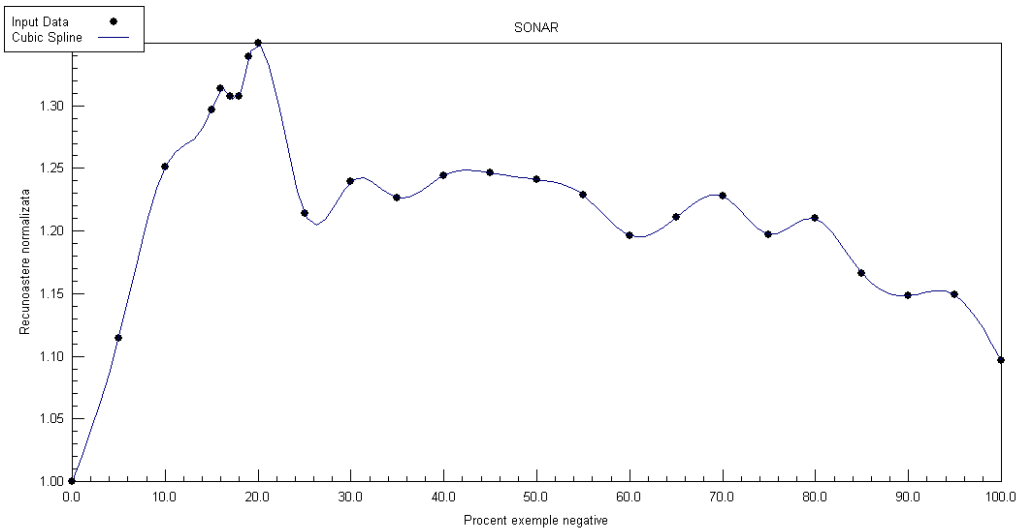


Figura 46. Interpolarea procentelor de recunoaştere normalizate pentru SONAR

Recunoaştere normalizată SONAR = f(Procent exemple negative)

BUPT

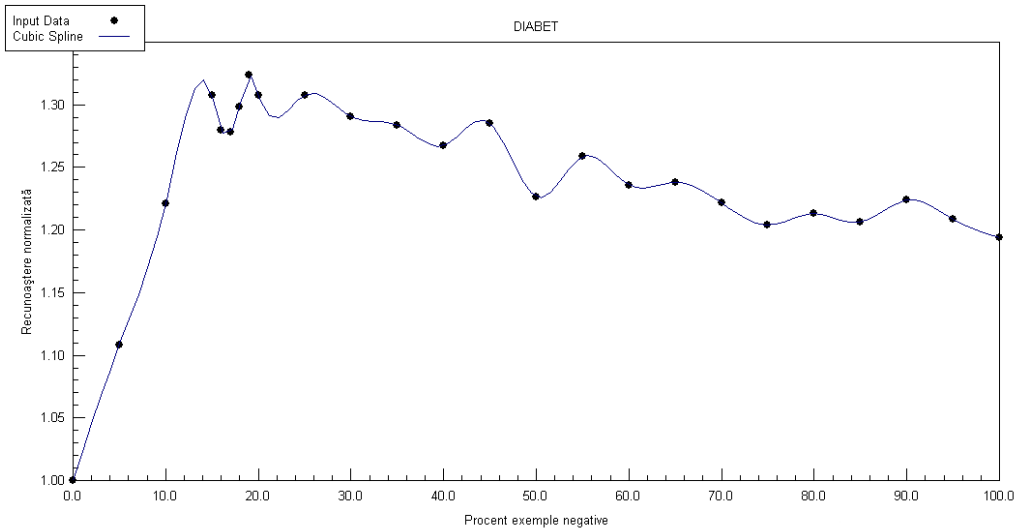


Figura 47. Interpolarea procentelor de recunoaştere normalizate pentru DIABET

Recunoaştere normalizată DIABET = f(Procent exemple negative)

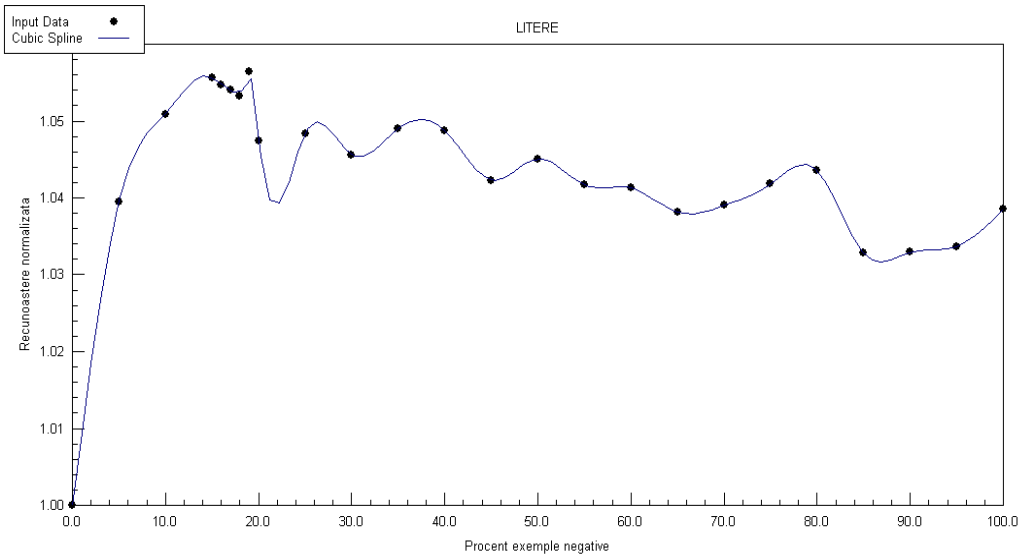


Figura 48. Interpolarea procentelor de recunoaştere normalizate pentru LITERE

Recunoaştere normalizată LITERE = f(Procent exemple negative)

Din analiza celor trei funcţii de interpolare se observă că acestea au aproximativ aceeaşi formă, şi prezintă o creştere a ratei de recunoaştere maximă pentru un interval de exemple negative cuprins între 10% şi 20% din numărul de exemple pozitive utilizat pentru antrenare.

BUPT

O primă concluzie care rezultă este aceea că antrenarea cu exemple negative induce un comportament similar în procesul de recunoaştere. În cursul acestui proces observăm prezenţa unui maxim de instanţe recunoscute pentru fiecare dintre cele 3 exemple.

Intervalul de exemple negative în care acesta se situează este de [10, 20] % din numărul de exemple pozitive din setul de antrenare. Următoarea etapă este aceea de a vedea în ce măsură acest comportament similar poate fi definit printr-un model, şi dacă da, în ce măsură acest model este corect din punct de vedere statistic.

Determinarea modelului procesului de antrenare cu exemple negative

Pentru determinarea modelului comportamental al procesului de antrenare cu exemple negative, se propune utilizarea unei tehnici corelaţionale pentru a defini legăturile între datele prezente în acelaşi set. Fiecare dintre aceste legături este analizată pentru a vedea în ce măsură poate conduce la obţinerea unui model matematic credibil.

În continuare vom detalia procesul de analiză efectuat pentru fiecare set de date în parte.

Modelul matematic al procesului de antrenare pentru SONAR, în urma efectuării unei analize corelaţionale, are următoarea formă.

Definiţia modelului	$f(x) = a*x^4 + b*x^3 + c*x^2 + d*x + e$
Numărul de observaţii	25
Numărul de observaţii ratate	0
Tipul modelului	Neliniar
Limita de iteraţii neliniare	250
Eroarea standard pentru procesul de estimare	3.53E-02
Coeficienţii de determinări multiple(R^2)	0.8367
Credibilitatea modelului	83.67%

Modelul rezultat prezintă o credibilitate de 0.836 pentru o încredere de 95%. Forma acestuia este următoarea:

$$f(x) = -6.91*10^{-8}*x^4 + 1.52*10^{-5}*x^3 - 1.14*10^{-3}*x^2 + 3.179*10^{-2}*x + 1.0126 \quad (5.2)$$

Din datele obţinute pentru acest model se poate observa calitatea deosebit de bună pe care acesta o posedă, cu un model comportamental de 0.836 în condiţiile unei încrederi de 95%.

Modelul matematic al procesului de antrenare pentru DIABET, în urma efectuării unei analize corelaţionale, are următoarea formă.

BUPT

Definiţia modelului	$f(x) = a*x^4 + b*x^3 + c*x^2 + d*x + e$
Numărul de observaţii	25
Numărul de observaţii ratate	0
Tipul modelului	Neliniar
Limita de iteraţii neliniare	250
Eroarea standard pentru procesul de estimare	1.73E-02
Coeficienţii de determinări multiple(R^2)	0.9487
Credibilitatea modelului	94.86%

Modelul rezultat prezintă o credibilitate de 0.948 pentru o încredere de 95%. Forma acestuia este următoarea:

$$f(x) = -4.76*10^{-8}*x^4 + 1.15*10^{-5}*x^3 - 9.65*10^{-4}*x^2 + 0.0302*x + 0.9997 \quad (5.3)$$

Din datele obţinute pentru acest model se poate observa calitatea deosebit de bună pe care şi acesta o posedă, cu un model comportamental de 0.948 în condiţiile unei încrederi de 95%.

Modelul matematic al procesului de antrenare pentru LITERE, în urma efectuării unei analize corelaţionale, are următoarea formă.

Definiţia modelului	$f(x) = a*x^4 + b*x^3 + c*x^2 + d*x + e$
Numărul de observaţii	25
Numărul de observaţii ratate	0
Tipul modelului	Neliniar
Limita de iteraţii neliniare	250
Eroarea standard pentru procesul de estimare	5.787E-03
Coeficienţii de determinări multiple(R^2)	0.784
Credibilitatea modelului	78.48%

Modelul rezultat prezintă o credibilitate de 0.784 pentru o încredere de 95%. Forma acestuia este următoarea:

$$f(x) = -8.72*10^{-9}*x^4 + 2.03*10^{-6}*x^3 - 1.602*10^{-4}*x^2 + 4.638*10^{-3}*x + 1.0109 \quad (5.4)$$

Din datele obţinute pentru acest model se poate observa calitatea bună pe care acesta o posedă, cu un model comportamental de 0.784 în condiţiile unei încrederi de 95%.

Determinarea unui model general al procesului de antrenare cu exemple negative

În acest moment ne propunem să verificăm dacă modelele determinate în pasul anterior sunt reprezentative pentru procesul de antrenare cu exemple

BUPT

negative. Dacă vom obţine un răspuns afirmativ pentru această verificare, dorim să ştim şi dacă este posibilă obţinerea unui model general prin intermediul căruia să fie determinat numărul de exemple negative pentru un caz particular existent.

Intr-o primă instanţă vom verifica dacă cele trei modele fac parte din aceeaşi clasă. Pentru aceasta se va utiliza un test *t* , verificarea fiind făcută pentru setul de parametri (a, b, c, d, e) ce caracterizează modelele. Testul *t* este făcut pentru a verifica dacă două modele diferă în mod semnificativ.

Verificarea se va face cu ajutorul utilitarului Origin 6.0 şi se utilizează parametrii de la două modele succesive. Datele care au stat la baza analize sunt prezentate în tabelul de mai jos.

	a	**b**	**c**	**d**	**e**
SONAR	-6.91E-08	1.52E-05	-0.00114	0.0318	1.0126
DIABET	-4.76E-08	1.15E-05	-0.000965	0.0302	0.9997
LITERE	-8.72E-09	2.03E-06	-0.000160	0.00464	1.10109
Valoarea medie	-4.180E-08	9.57E-06	-0.000755	0.0222093	1.00773

Tabelul 9. Valorile coeficienţilor pentru cele 3 modele studiate

Rezultatele obţinute sunt prezentate pentru fiecare dintre cele 2 analize efectuate asupra coeficienţilor.

Pentru bazele de date SONAR şi DIABET am obţinut rezultatele prezente în tabelul de mai jos. Analiza a fost făcută pentru un nivel de credibilitate de 95% şi putem spune că cele două modele caracterizează aceleaşi populaţii de procese de antrenare.

Baza de date	Media	Variaţia	Numărul de elemente
SONAR	0.20866	0.20217	5
DIABET	0.20579	0.19714	5
t = -0.01014 p= 0.99216			

Pentru DIABET şi LITERE, şi beneficiind de un nivel de credibilitate de 95%, rezultă conform rezultatelor din tabelul de mai jos că cele două modele caracterizează aceleaşi populaţii de procese de antrenare.

Baza de date	Media	Variaţia	Numărul de elemente
SONAR	0.20579	0.19714	5
DIABET	0.20308	0.20394	5
t = -0.00958 p= 0.99259			

Concluzia rezultată în urma acestui studiu este că cele trei modele descriu acelaşi fenomen (caracterizează acelaşi proces de antrenare cu exemple negative) şi că pot reprezenta baza de plecare pentru determinarea unui model global al acestui tip de antrenare a unei reţele neuronale.

BUPT

Pentru construirea acestuia se pleacă de la valorile medii ale parametrilor ce definesc cele trei modele discutate anterior, valori care au fost prezentate într-un tabel anterior (Tabelul 9).

Modelul general rezultat prezintă următoarea formă:

$$f(x) = -4.18067 * 10^{-8} * x^4 + 9.57666 * 10^{-6} * x^3 - 7.5507 * 10^{-4} * x^2 + 2.2209 * 10^{-2} * x + 1.000773333 \quad (5.5)$$

Testarea modelului general pe o nouă bază de date

Utilizând acest model general am ales o nouă bază de date pentru a testa veridicitatea acestuia. Deoarece am făcut numeroase experimente cu reţeaua ce încearcă determinarea apariţiei unor valuri periculoase (VALURI), am ales aceste date pentru a testa modelul general.

Rezultatele obţinute de acesta pe noua bază de date sunt prezentate în Figura 49. Se poate observa că acesta reuşeşte localizare precisă a procentului de exemple negative optim.

De asemenea, toate valorile de recunoaştere ale reţelei prezente pentru un procent de exemple negative situate între [0% - 100%] sunt interpolate cu succes de către modelul general.

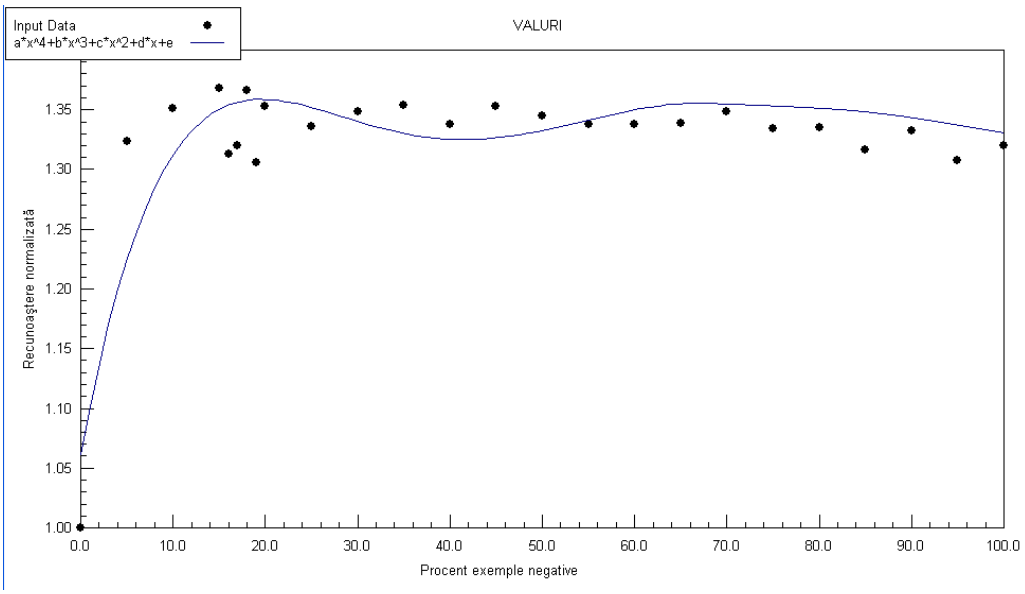


Figura 49. Modelul general pentru antrenarea cu exemple negative aplicat pentru VALURI

Concluzii

Încheiem această analiză prin formularea următoarelor concluzii referitoare la determinarea unui model general de antrenare cu exemple negative:

- ✓ cele trei modele analizate (LITERE, DIABET şi SONAR) descriu acelaşi fenomen (caracterizează acelaşi proces de antrenare cu exemple negative) şi reprezentă baza de plecare pentru

BUPT

determinarea unui model global al acestui tip de antrenare a unei reţele neuronale.

- ✓ modelul general obţinut are forma prezentată în formula 5.5
- ✓ testarea acestuia a fost făcută pe baza de date VALURI şi s-a reuşit localizarea precisă a procentului optim de exemple negative

5.3 Experimente efectuate pe reţele neuronale artificiale de tip multilayer perceptron cu 1 neuron block

În capitolul anterior am observat că antrenarea cu exemple negative pregăteşte reţeaua neuronală pentru memorarea unor caracteristicilor nedorite. Acest lucru este foarte benefic asigurând astfel o robusteţe a reţelei neuronale precum şi o îmbunătăţire semnificativă a procentului de recunoaştere. Din păcate, la tipologia clasică a reţelei neuronale de tip multilayer perceptron, nici un neuron de pe stratul de ieşire nu este direct responsabil de recunoaşterea unor tipare nedorite. Acest lucru se realizează în majoritatea cazurilor printr-o inactivitate totală a neuronilor de ieşire, situaţie similară şi cu cazul în care reţeaua neuronală nu recunoaşte tiparul de la intrare.

Pentru a diferenţia clar un tipar de intrare negativ de un tipar de intrare nerecunoscut de reţea, avem nevoie de unul sau mai mulţi neuroni care să ne spună acest lucru.

Pentru aceasta am definit o nouă topologie de reţea neuronală de tip multilayer percepton şi anume reţea neuronală de tipul multilayer perceptron cu neuroni bloc.

În acest capitol vom antrena în paralel reţele multilayer perceptron clasice împreună cu reţele multilayer perceptron cu 1 neuron **block**.

Vom realiza diverse comparaţii între acestea în multiple situaţii şi cu seturi de antrenare diferite. Se va realiza şi un studiu cu privire la influenţa noii tipologii de reţea asupra procentului de recunoaştere, în funcţie de numărul de clase prezente la ieşire. La final, vom încerca să rezumăm toate datele obţinute în câteva concluzii despre rezultatele obţinute de noua topologie.

5.3.1 Experimente pe NIST 19

Au fost făcute 2 experimente pe această bază de date pentru a studia modul de comportament şi performanţele obţinute de noua topologie în procesul de recunoaştere. Primul experiment a măsurat performanţele obţinute de reţeaua multilayer perceptron cu neuron **block** atunci când numărul de exemple negative prezent în setul de antrenare variază între [5% - 100%]. Astfel, s-au efectuat mai multe antrenări de reţele neuronale cu un set de antrenare conţinând diverse procente de exemple negative. Pentru fiecare set de antrenare s-au antenat 2 reţele: o reţea multilayer perceptron clasică şi o reţea multilayer perceptron cu un neuron **block**.

Al doilea experiment a testat modul în care reţeaua multilayer perceptron cu neuron **block** se comportă atunci când numărul claselor recunoscute la ieşire variază. Şi pentru acest experiment s-au antrenat 2 tipuri de reţele pentru fiecare situaţie posibilă(numărul de clase de la ieşire a variat între 1 şi 10).

BUPT

Ambele experimente utilizează exemple negative în seturile de antrenare.

Experimentul 5

Aşa cum am precizat mai sus, pentru acest experiment s-au realizat multiple seturi de antrenare având ponderi diferite pentru exemplele negative. Deoarece pentru fiecare set de la intrare am comparat rezultatele obţinute de reţeaua multilayer perceptron cu neuron **block** cu o reţea multilayer perceptron clasică, am utilizat seturile de antrenare folosite în cadrul Experimentului 1 şi Experimentului 2.

Reamintim că experimentul pleacă iniţial de la un set de antrenare ce conţine un număr echivalent cu 5% exemple negative(din numărul de exemple pozitive) şi adaugă incremental acest număr la exemplele negative până la obţinerea unor procente de exemple negative şi pozitive egale în setul de antrenare.

Numărul exemplelor pozitive şi negative pentru fiecare antrenare este prezent în Tabelul 2, mai puţin prima înregistrare pentru care nu au fost folosite exemple negative. De asemenea precizăm că pentru valori ale procentelor exemplelor negative cuprinse în intervalul [15% - 20%] (din numărul de exemple pozitive) au fost făcute mai multe antrenări. Astfel, în acest interval variaţie exemplelor negative în setul de antrenare a fost de 1% şi nu de 5% cât este în afara intervalului. Acest lucru s-a datorat găsirii în capitolul precedent a valorii optime de exemple negative cuprinse în intervalul [17% - 19%] (din numărul de exemple pozitive).

Deoarece acest experiment este făcut pentru a compara performanţele unei reţele multilayer perceptron clasică cu una cu neuron block, am dorit să utilizăm rezultatele obţinute în cadrul Experimentului 1, şi de aceea caracterele ce trebuie recunoscute de reţelele neuronale au rămas aceleaşi ca la Experimentele 1 şi 2: „a”, „e”, „n” şi „r”.

Reamintim configuraţia reţelei de la Experimentul 1 ca fiind de tip multilayer perceptron cu 1 strat ascuns. Configuraţia acesteia este următoarea:

- stratul de intrare conţine 1024 neuroni – am folosit ca intrări desene de 32x32 pixeli cu literele respective
- stratul ascuns are 700 de neuroni
- stratul de ieşire are 4 neuroni – fiecare dintre cei 4 neuroni este responsabil cu recunoaşterea unui caracter

Funcţiile de activare corespunzătore fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

În cadrul acestui experiment această reţea va purta denumirea de reţea multilayer perceptron clasică. A doua reţea folosită poartă denumirea de reţea multilayer perceptron cu un neuron **block** şi are următoarea configuraţie:

- stratul de intrare conţine 1024 neuroni – am folosit ca intrări desene de 32x32 pixeli cu literele respective
- stratul ascuns are 700 de neuroni
- stratul de ieşire are 5 neuroni – 4 neuroni responsabili cu recunoaşterea celor 4 caractere, plus un neuron suplimentar(neuron block) responsabil cu recunoaşterea tiparelor negative.

Funcţiile de activare corespunzătore fiecărui dintre cele 3 straturi sunt:

BUPT

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

Antrenarea a pornit cu un set de 4 caractere (a,e,n, şi r), fiecare având câte 5000 de instanţe(exemple pozitive) precum şi câte 200 de caractere(literele b,d,h,l şi t), acestea din urmă având rol de exemple negative. Numărul exemplelor negative a scăzut la 40 fiecare, în intervalul [15% - 20%] pentru o observare mai fină a evoluţiei procentului de recunoaştere.

Rezultatele obţinute de către reţeaua multilayer clasică au fost prezentate în Tabelul 3 şi Tabelul 4. Rezultatele obţinute de către reţeaua neuronală cu un neuron block sunt prezentate în Tabelul 10.

Pentru a oferi o imagine de ansamblu am reprezentat grafic procentele de recunoaştere obţinute de reţeaua multilayer perceptron clasică şi reţeaua multilayer perceptron cu 1 neuron **block** pentru literele *„a”* şi *„r”* în Figura 50 (graficele pentru *„e”* şi *„n”* sunt similare şi nu le-am mai reprezentat).

Antrenare	Exemple negative %	Litera „a” %	Litera „e” %	Litera „n” %	Litera „r” %	Întreg alfabetul %	Set producţie %
	0	91.10	94.40	96.30	96.10	31.95	78.94
1	4.76	93.30	95.70	96.40	96.10	55.30	83.05
2	9.09	92.70	94.70	95.00	95.80	60.83	83.63
3	13.04	93.20	93.40	94.70	95.70	64.30	83.90
4	13.79	92.50	93.20	95.20	95.60	62.78	83.87
5	14.53	92.70	93.10	94.80	96.00	64.33	83.97
6	15.25	92.70	93.50	94.90	95.60	64.43	84.12
7	15.97	93.40	93.10	94.60	95.60	64.11	84.13
8	16.67	92.10	93.40	95.20	95.30	66.52	84.00
9	20.00	92.00	92.50	94.60	94.80	68.08	83.81
10	23.08	91.20	93.20	94.80	94.80	71.61	83.89
11	25.93	91.09	92.90	94.70	95.20	71.61	83.92
12	28.57	91.90	92.30	94.10	95.00	70.14	83.91
13	31.03	91.80	91.70	93.40	94.90	70.31	83.70
14	33.33	91.30	92.20	93.40	94.00	71.73	83.62
15	35.48	90.20	92.50	94.00	94.50	71.40	83.60
16	37.50	90.80	91.30	92.60	94.80	69.83	83.36
17	39.39	90.70	90.80	93.50	93.80	72.63	83.29
18	41.18	91.10	90.70	92.70	93.90	73.09	83.20
19	42.86	91.40	91.60	93.20	93.90	74.21	83.59
20	44.44	90.80	91.80	93.50	94.00	73.00	83.61
21	45.95	91.00	91.20	92.60	93.70	74.17	83.32
22	47.37	91.00	91.20	92.10	93.30	75.02	83.17
23	48.27	90.80	90.20	92.50	93.80	73.86	83.06
24	50.00	91.10	91.30	92.40	94.50	74.46	83.47

Tabelul 10. Rezultatele obţinute pe baza de date NIST 19 în cadrul experimentului 5

BUPT

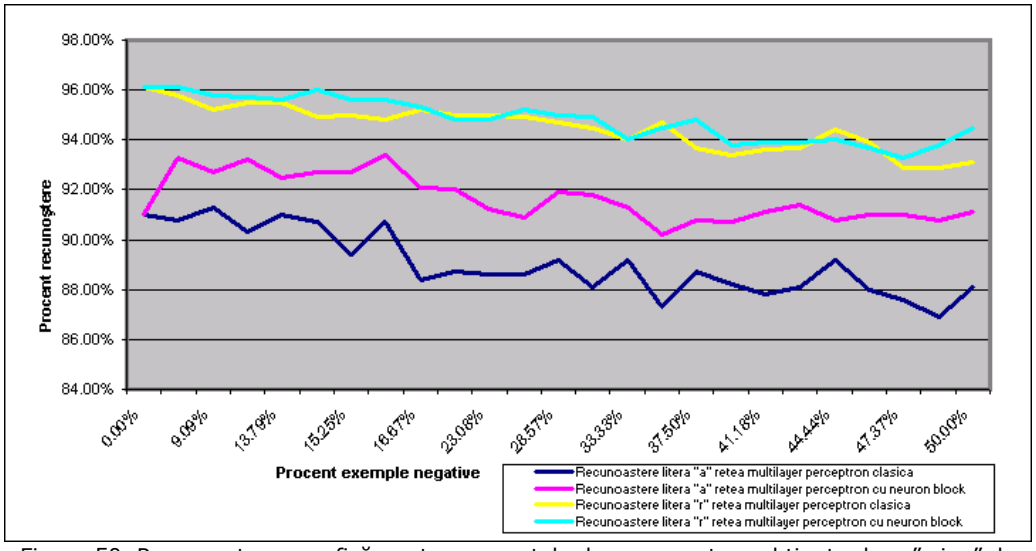


Figura 50. Reprezentarea grafică pentru procentele de recunoaştere obţinute de „a” şi „r” de către reţeaua multilayer perceptron şi reţeaua multilayer perceptron cu 1 neuron block

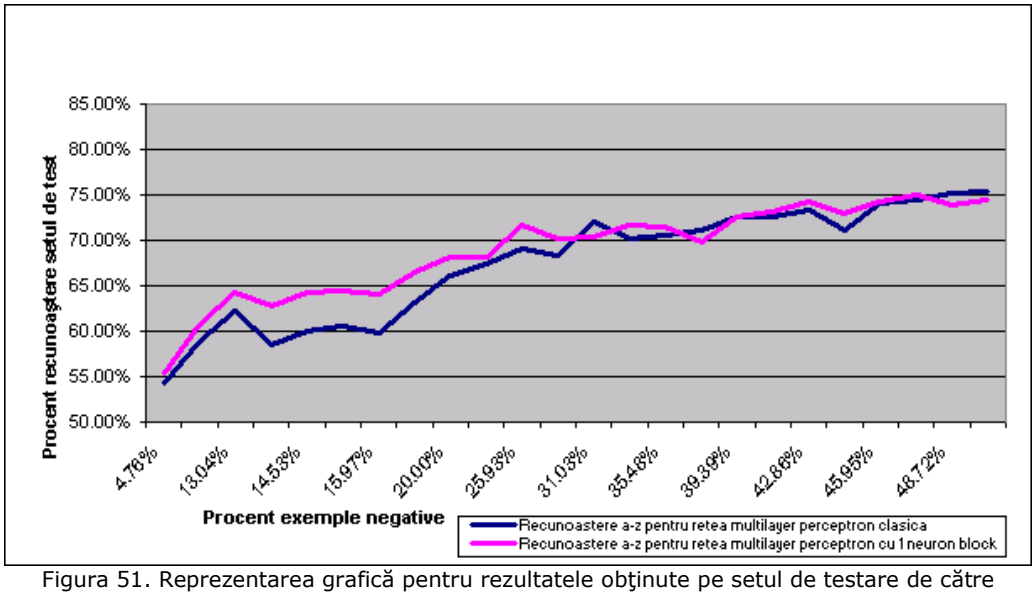


Figura 51. Reprezentarea grafică pentru rezultatele obţinute pe setul de testare de către reţeaua multilayer perceptron şi reţeaua multilayer perceptron cu 1 neuron block

Se poate observa cu uşurinţă faptul că procentele obţinute de reţeaua multilayer perceptron cu neuron **block** pentru fiecare clasă sunt net superioare celor obţinute de reţeaua multilayer perceptron clasică. Observăm de asemenea o reală îmbunătăţire a recunoaşterii unor tipare care sunt asemănătoare cu altele şi pentru care recunoaşterea e mai dificil de făcut. (De ex. litera *„a”* se confunda cu literele *„e”* şi *„o”*)

Astfel, antrenarea unei reţele multilayer perceptron cu neuron **block** oferă rezultate mai bune decât antrenarea unei reţele multilayer perceptron clasice pentru fiecare din clasele ce trebuie recunoscute la ieşire.

BUPT

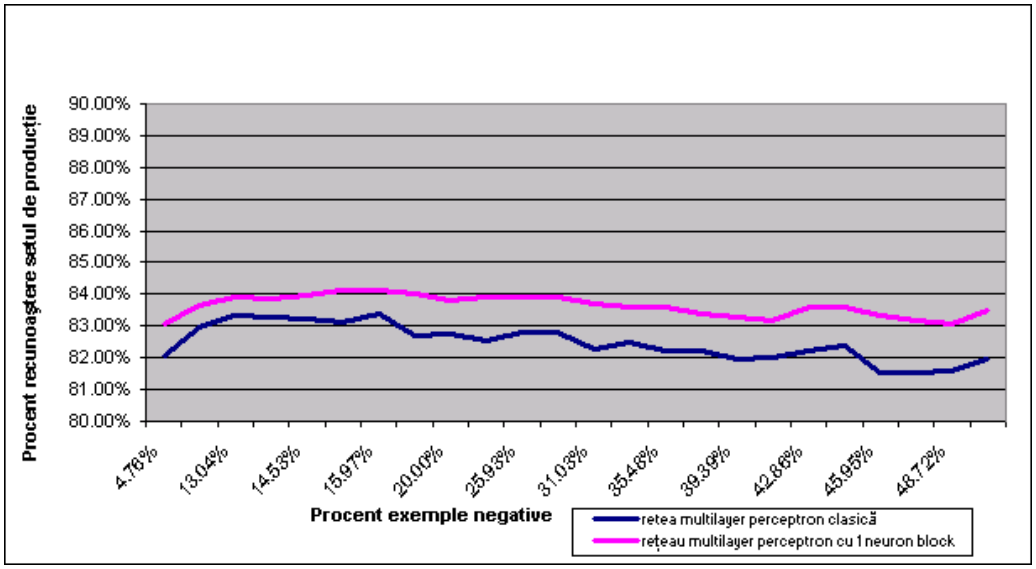


Figura 52. Reprezentarea grafică pentru rezultatele obţinute pe setul de producţie de către reţeaua multilayer perceptron şi reţeaua multilayer perceptron cu 1 neuron block

Pentru a analiza procentele obţinute de cele două reţele pe seturile de testare şi producţie, am construit graficele din Figura 51 şi Figura 52. Se poate observa şi aici că procentele obţinute de către reţeaua multilayer perceptron cu 1 neuron **block** sunt superioare celor obţinute de reţeaua multilayer perceptron clasică. Se observă astfel că acestea diferă cu valori cuprinse între 0% şi 5%.

O altă observaţie este aceea că toate procentele de recunoaştere pentru setul de producţie sunt mai mari la reţeaua multilayer perceptron cu 1 neuron **block** decât reţeaua multilayer perceptron clasică. Acest lucru denotă o maturitate a noii arhitecturi ce se prezintă astfel superioară celei clasice.

Pe lângă procentele de recunoaştere mai mari, reţeaua multilayer perceptron cu un neuron **block** oferă şi o claritate a ieşirilor. Astfel putem distinge acum cu uşurinţă între intrările care nu fac parte din clasele ce trebuie recunoscute de reţea şi intrările care nu sunt recunoscute de reţea. Acest lucru poate oferi noi posibilităţi de analiză a datelor şi clasificare mai sigură a ieşirilor reţelei.

În continuare a fost studiat modul în care reţeaua multilayer perceptron cu neuron **block** reacţionează atunci când numărul claselor de ieşire creşte.

Concluzii

În urma efectuării experimentului 5 (antrenarea unei reţele multilayer perceptron cu 1 neuron **block** în vederea recunoaşterii caracterelor, şi comparaţie între aceasta şi o reţele multilayer perceptron clasică), au fost formulate următoarele concluzii:

- ✓ procentele obţinute de noua topologie de reţea (multilayer perceptron cu 1 neuron **block**) pe un set de producţie sunt cu până la 5% mai bune decât o reţea multilayer perceptron clasică
- ✓ procentele individuale pe clase de recunoaştere sunt de asemenea mai mari decât cele obţinute de o reţea multilayer perceptron clasică antrenată fără exemple negative; lucru care nu a putut fi obţinut de reţeaua multilayer perceptron clasică antrenată şi cu

BUPT

exemple negative (reamintim că acest aspect a fost relevat în experimentele 1-4)

- ✓ o mai mare claritate a ieşirilor la reţeaua multilayer perceptron cu neuron **block**; astfel tiparele negative sunt identificate cu uşurinţă ca o nouă clasă prezentă la ieşire

Experimentul 6

În acest experiment vom varia numărul de clase prezente la ieşirea reţelei de la 1 la 10 clase. Vom încerca să evidenţiem astfel comportamentul reţelei multilayer perceptron cu 1 neuron **block** faţă de o reţea multilayer clasică atunci când numărul claselor de la ieşire creşte.

Configuraţiile celor 2 reţele sunt asemănătoare, ca singură diferenţă remarcăm numai neuronul suplimenta de pe stratul de ieşire prezent la reţeaua multilayer perceptron cu 1 neuron **block**.

Deşi variază de-a lungul celor 10 antrenări, configuraţiile acestea se încadrează în următorul tipar:

- stratul de intrare conţine 1024 neuroni – am folosit ca intrări desene de 32x32 pixeli cu literele respective
- stratul ascuns are între 650 şi 660 de neuroni, variind cu 1 pentru fiecare nou tipar adăugat
- stratul de ieşire are între 1 şi 10 neuroni pentru reţeaua multilayer perceptron clasică şi între 2 şi 11 neuroni pentru reţeaua multilayer perceptron cu neuron **block**.

Funcţiile de activare corespunzătore fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

Cele 10 tipare folosite la ieşire reprezintă recunoaşterea următoarelor litere: „e”, „a”, „n”, „r”, „b”, „d”, „h”, „l”, „t” şi „q” (folosite în această ordine). Numărul de tipare pozitive de antrenare a fost fixat la 20.000, având astfel 20.000 de tipare pentru litera „e” la primul experiment şi terminând cu câte 2.000 pentru fiecare dintre cele 10 litere prezente.

Tiparele negative au fost în număr de 3000 (literele „c”, „f”, „i”, „k” şi „m”) reprezentând 15% din numărul total de tipare pozitive.

Aşa cum am precizat mai sus au fost făcute 20 de antrenări, câte 10 pentru fiecare dintre cele 2 reţele. Testarea s-a efectuat pe un set de producţie cu 26.000 de exemple, conţinând 26 de litere a câte 1.000 de tipare fiecare.

Procentele de recunoaştere obţinute au fost calculate conform formulei (5.1) şi sunt prezente în Tabelul 13 şi reprezentate vizual în Figura 53.

În Tabelul 11 şi Tabelul 12 pot fi observate valorile procentelor de recunoaştere pentru fiecare din cele 10 clase de ieşire utilizate în cadrul experimentului.

De asemenea, pentru o mai bună vizualizare a rezultatelor am reprezentat grafic procentele de recunoaştere pentru litera „e” (a fost prezentă în toate experimentele) obţinute de cele 2 reţele de-a lungul experimentelor în Figura 54.

BUPT

Nr. cls.	Lit. „e” %	Lit. „a” %	Lit. „n” %	Lit. „r” %	Lit. „b” %	Lit. „d” %	Lit. „h” %	Lit. „l” %	Lit. „t” %	Lit. „q” %
1	99.0									
2	95.3	96.1								
3	95.1	93.7	96.5							
4	93.9	92.7	95.5	95.2						
5	87.9	91.5	94.4	95.6	95.3					
6	87.2	90.0	93.4	95.3	94.5	93.8				
7	85.9	88.0	88.2	94.4	93.6	93.9	92.6			
8	82.6	87.8	89.0	92.5	93.0	92.6	90.4	81.0		
9	81.6	86.3	88.0	92.9	92.3	92.4	90.9	73.7	81.7	
10	81.3	78.9	87.1	91.7	93.1	91.6	89.3	71.7	82.7	89.0

Tabelul 11. Procentele de recunoaştere obţinute de reţeaua multilayer perceptron cu 1 neuron block

Nr. cls.	Lit. „e” %	Lit. „a” %	Lit. „n” %	Lit. „r” %	Lit. „b” %	Lit. „d” %	Lit. „h” %	Lit. „l” %	Lit. „t” %	Lit. „q” %
1	99.0									
2	95.4	95.3								
3	93.3	90.7	95.8							
4	93.1	90.2	95.0	94.8						
5	82.9	88.4	92.4	94.7	93.8					
6	81.5	84.1	91.9	93.7	93.4	92.2				
7	82.2	82.8	87.1	94.2	91.1	91.1	89.3			
8	74.7	82.2	86.4	92.6	91.2	89.1	87.8	79.5		
9	74.4	80.7	85.1	90.9	89.9	88.9	88.6	71.5	77.3	
10	74.2	72.8	84.6	91.4	89.6	86.3	89.4	65.8	77.5	82.9

Tabelul 12. Procente de recunoaştere obţinute de reţeaua multilayer perceptron clasică

Nr. clase	Recunoaştere reţea multilayer perceptron cu 1 neuron block (%)	Recunoaştere reţea multilayer perceptron clasică (%)
1	73.92 %	74.52 %
2	82.72 %	82.33 %
3	86.91 %	85.42 %
4	88.24 %	87.35 %
5	88.98 %	87.02 %
6	89.51 %	87.18 %
7	89.24 %	86.37 %
8	86.91 %	84.03 %
9	85.46 %	81.90 %
10	84.77 %	80.72 %

Tabelul 13. Procentele de recunoaştere obţinute pe setul de producţie de către cele 2 reţele

BUPT

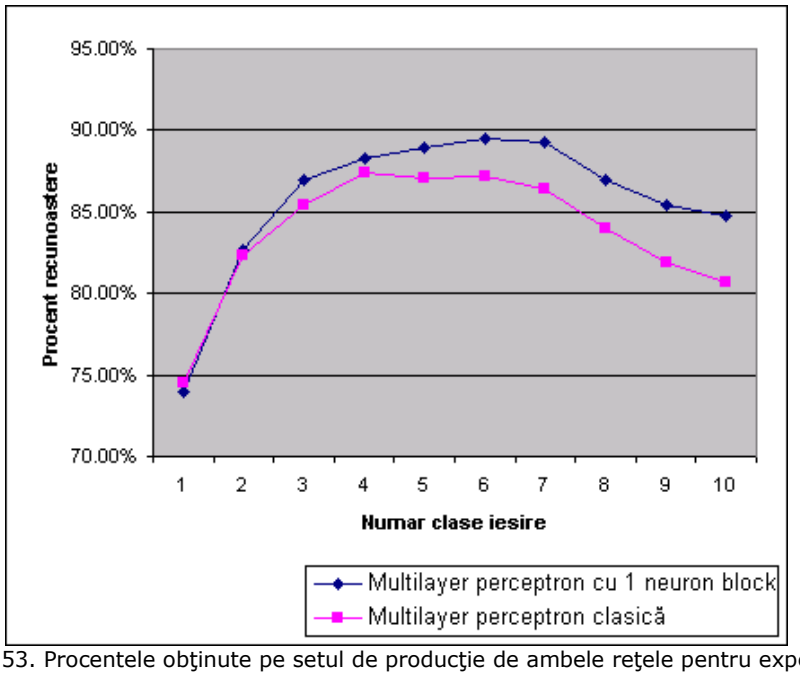


Figura 53. Procentele obţinute pe setul de producţie de ambele reţele pentru experimentul 6

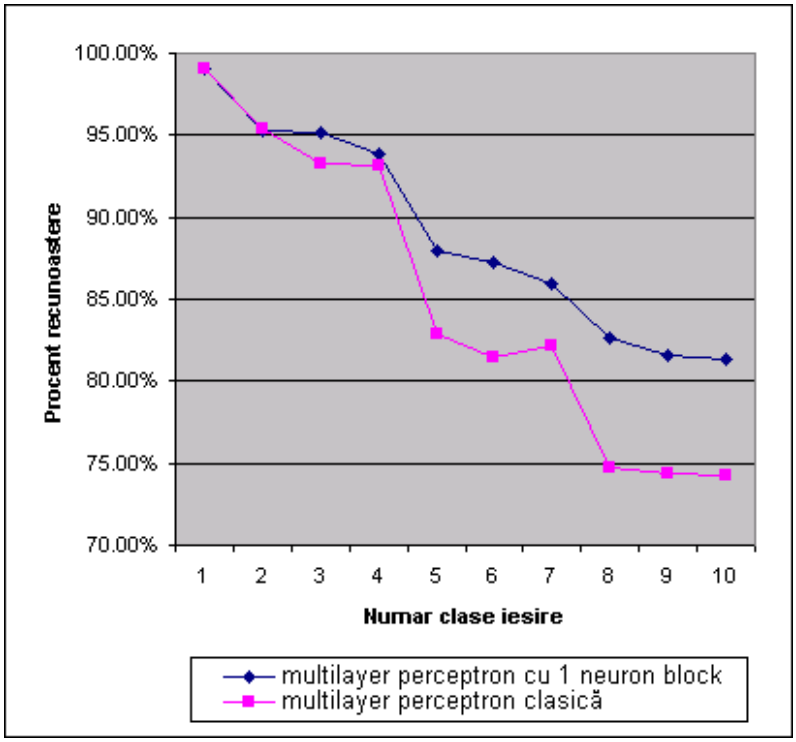


Figura 54. Procentele de recunoaştere pentru litera „e” în cadrul experimentului 6

BUPT

Se poate uşor observa că pentru întreg setul de producţie, reţeaua multilayer perceptron cu neuron **block** a obţinut rate mai mari decât cea clasică pentru situaţiile în care avem cel puţin 2 clase diferite la intrare. Cu cât numărul de clase creşte, diferenţa se accentuează ajungând până la 4% pentru antrenarea unei reţele ce are 10 clase diferite la intrare.

Acest aspect recomandă folosirea unei reţele multilayer cu neuron **block** atunci când avem mai multe clase de intrări. Cu cât numărul de clase de intrare este mai mare, cu atât este de dorit folosirea acestui tip de reţea.

Dacă analizăm comportamentul acestei reţele şi la nivel de clasă particulară, observăm şi aici o îmbunătăţire semnificativă faţă de o reţea multilayer obişnuită. Astfel, în Figura 54 se poate observa o superioritate clară a reţelei multilayer cu neuron **block** şi la nivel de clasă de recunoaştere.

De exemplu, pentru litera „e" am obţinut procente de recunoaştere superioare de până la 7%. Acest aspect este unul deosebit de important şi merită subliniat faptul că reţeaua multilayer cu un neuron **block** prezintă procente de recunoaştere superioare faţă de reţeaua multilayer clasică şi la nivel individual de clasă de intrare. Cu cât numărul de clase de la intrare este mai mare, cu atât diferenţa dintre cele două reţele este mai accentuată.

La fel ca şi în experimentele de la capitolul anterior am dori să verificăm rezultatele obţinute pentru antrenarea unei reţele multilayer perceptron cu neuron **block** pe cât mai multe seturi de date. Din această cauză, următorul capitol este dedicat experimentelor făcute pe setul Ocean's Data de reţele multilayer cu neuron **block**.

Concluzii

Concluziile care au fost formulate în urma realizării experimentului 6 (antrenarea unei reţele neuronale de tip multilayer perceptron cu 1 neuron **block** utilizând mai multe clase de ieşire) sunt prezentate în cele ce urmează:

- ✓ pentru întreg setul de producţie, reţeaua multilayer perceptron cu neuron block a obţinut rate mai mari decât reţeaua multilayer perceptron normală atunci când sunt cel puţin 2 clase diferite la intrare
- ✓ cu cât numărul de clase de la ieşire creşte, diferenţa se accentuează ajungând până la 4% pentru antrenarea unei reţele ce are 10 clase diferite la intrare
- ✓ şi la nivel de clasă individuală se poate observa o mai bună recunoaştere a reţelei multilayer perceptron cu neuron **block** faţă de reţeaua multilayer perceptron normală

5.3.2 Experimente pe Ocean's Data

Şi pe acest set de date s-au efectuat numai experimente pentru a observa cum anume sunt influenţate reţelele multilayer perceptron cu neuron **block** de numărul de exemple negative din setul de antrenare.

Experimente legate de influenţa numărului de clase prezent la ieşire asupra unei reţele multilayer perceptron cu neuron **block**, nu au putut fi executate. Explicaţia rezidă în numărul mic de clase de intrare prezent în acest set, ceea ce face ca datele să fie neconcludente.

BUPT

Experimentul 7

Similar cu experimentul 5, şi pentru acest experiment au fost realizate mai multe seturi de antrenare cu ponderi diferite pentru procentul de exemple negative conţinut. Pentru fiecare din acest set au fost antrenare două tipuri de reţele: o reţea multilayer perceptron cu neuron **block** şi o reţea multilayer perceptron normală. Din această cauză, seturile de antrenare utilizate pentru fiecare antrenare sunt cele de la exemplul 3.

Numărul exemplelor pozitive şi negative este cel din Tabelul 5 (mai puţin prima înregistrare), iar pentru intervalul [15% - 20%] (procente exemple negative din numărul de exemple pozitive) s-au generat seturi de antrenare la care procentul de exemple negative variază cu 1% din numărul de exemple pozitive. Am făcut acest lucru deoarece, acest interval conţine procentul optim de exemple negative recomandat la antrenarea unei reţele neuronale şi dorim o analiză mai atentă în acest caz.

Tema experimentului este similară cu cea a Experimentul 3 şi încearcă determinarea gradului de risc privind apariţia unor valuri foarte mari într-o anumită perioadă a anului.

Configuraţia reţelei neuronale de tipul multilayer perceptron cu neuron **block** este următoarea

- stratul de intrare conţine 17 neuroni – am folosit ca intrări toţi parametrii din Anexa 2 mai puţin parametrul 8
- stratul ascuns are 160 de neuroni
- stratul de ieşire are 4 neuroni – avem 3 dintre aceştia răspunzători cu nivele de risc pentru valuri, iar al 4-lea fiind neuronul block (responsabil cu detecţia tiparelor negative de la intrare)

Funcţiile de activare corespunzătoare fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară $f(x) = x$.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

Prima antrenare utilizează un set de 13.000 de exemple pozitive şi 650 exemple negative, iar ultima are 13.000 exemple pozitive şi 13.000 exemple negative. La fiecare pas de antrenare numărul exemplelor negative a crescut cu 650, iar în intervalul [15% - 20%] (exemple negative din numărul de exemple pozitive) numai cu 130, pentru o observare mai fină a evoluţiei procentului de recunoaştere.

Rezultatele obţinute de către o reţea multilayer perceptron clasică sunt prezentate în Tabelul 6 şi Tabelul 7. Pentru acelaşi set de date, rezultatele obţinute de către reţeaua multilayer perceptron cu un neuron **block** sunt prezente în Tabelul 14.

BUPT

Antrenare	Exemple negative %	Risc inexistent %	Risc scăzut %	Risc mediu %	Set test %	Set producţie %
	0	85.34	94.44	64.29	28.00	61.93
1	4.76	87.07	96.53	60.00	86.18	82.85
2	9.09	78.45	99.31	58.57	88.18	82.00
3	13.04	87.93	99.31	61.43	89.82	85.48
4	13.79	77.59	98.61	64.29	90.18	83.53
5	14.53	81.90	98.61	57.14	90.36	82.82
6	15.25	78.45	97.92	62.86	90.73	83.34
7	15.97	77.59	97.22	65.71	91.64	84.16
8	16.67	82.76	95.83	64.29	91.09	84.39
9	20.00	89.66	95.83	65.71	91.82	86.66
10	23.08	83.62	96.53	54.29	92.18	82.96
11	25.93	74.14	95.14	60.00	92.45	81.73
12	28.57	79.31	97.22	58.57	93.09	83.22
13	31.03	73.28	98.61	57.14	92.27	81.64
14	33.33	78.45	99.31	62.86	92.09	84.47
15	35.48	70.69	98.61	50.00	91.82	79.63
16	37.50	77.59	97.92	45.71	92.00	79.82
17	39.39	65.52	97.92	54.29	92.00	79.36
18	41.18	64.66	97.22	52.86	92.00	78.55
19	42.86	66.38	97.22	51.43	92.00	78.66
20	44.44	68.97	97.22	45.71	91.73	77.91
21	45.95	67.24	95.83	50.00	92.00	77.98
22	47.37	66.38	97.92	41.43	91.64	76.33
23	48.27	66.38	98.61	42.86	91.82	76.90
24	50.00	68.97	94.44	41.43	91.82	76.18

Tabelul 14. Rezultatele obţinute pe baza de date Ocean's Data în cadrul experimentului 7

Pentru o bună vizualizare a datelor (la fel ca şi la restul experimentelor) am reprezentat grafic aceste valori în figurile următoare. Aceste date au fost comparate cu cele obţinute în cadrul experimentelor 3 şi 4. Reamintim că cele 2 experimente (3 şi 4) au măsurat performanţele obţinute de o reţea multilayer perceptron antrenată cu un set ce conţine şi exemple negative.

În Figura 55 au fost reprezentate procentele obţinute pentru fiecare grad de risc în parte, de cele 2 tipuri de reţele. Se poate uşor observa că reţeaua multilayer perceptron cu neuron **block** este superioară celei clasice, obţinând pentru toate categoriile procente superioare.

BUPT

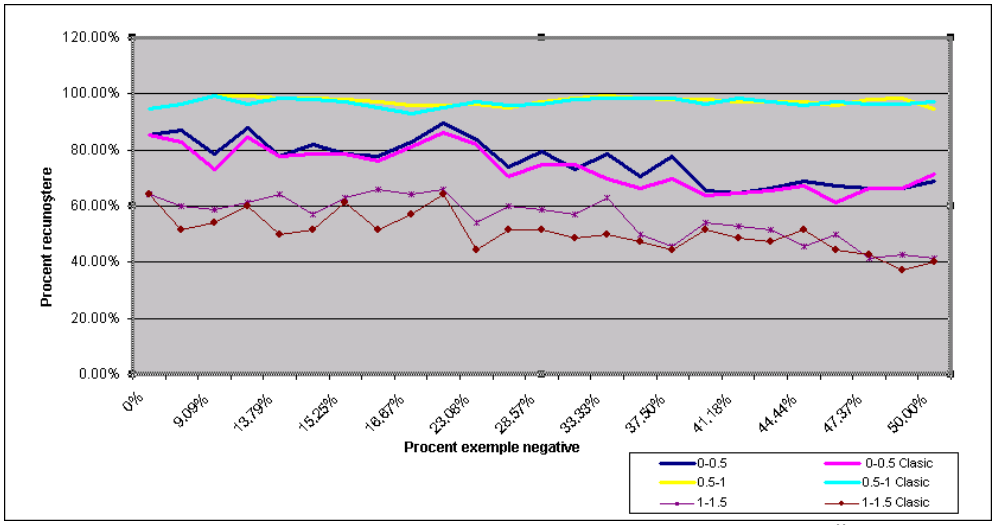


Figura 55. Procentele de recunoaştere pentru valorile de risc obţinute de către o reţea multilayer perceptron cu neuron block şi o reţea multilayer perceptron clasică

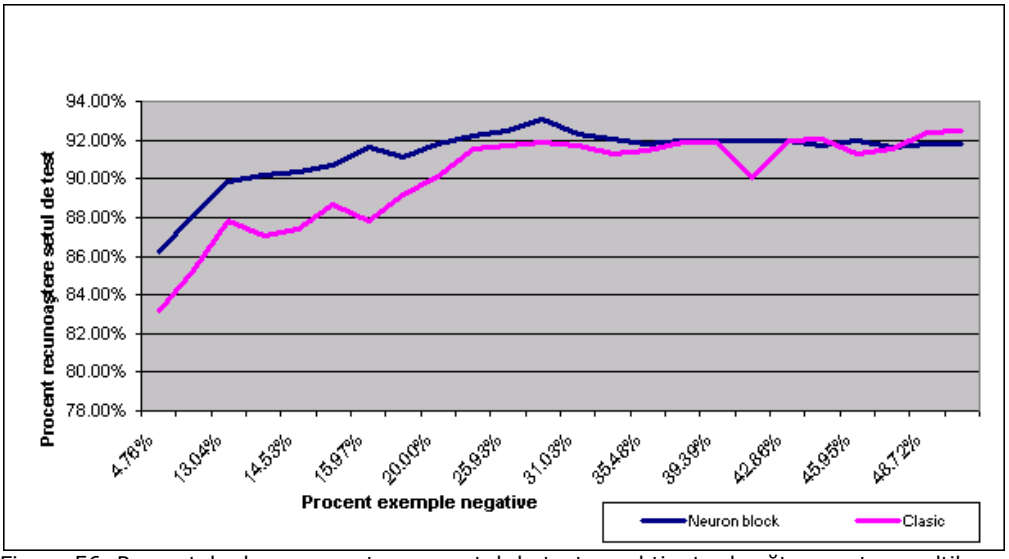


Figura 56. Procentele de recunoaştere pe setul de testare obţinute de către o reţea multilayer perceptron cu neuron block şi o reţea multilayer perceptron clasică

Ambele figuri (Figura 56 şi Figura 57) conţin rezultate comparative obţinute de reţeaua multilayer perceptron cu neuron **block** faţă de reţeaua multilayer perceptron normală pe două seturi de date (un set de producţie şi un set de testare).

BUPT

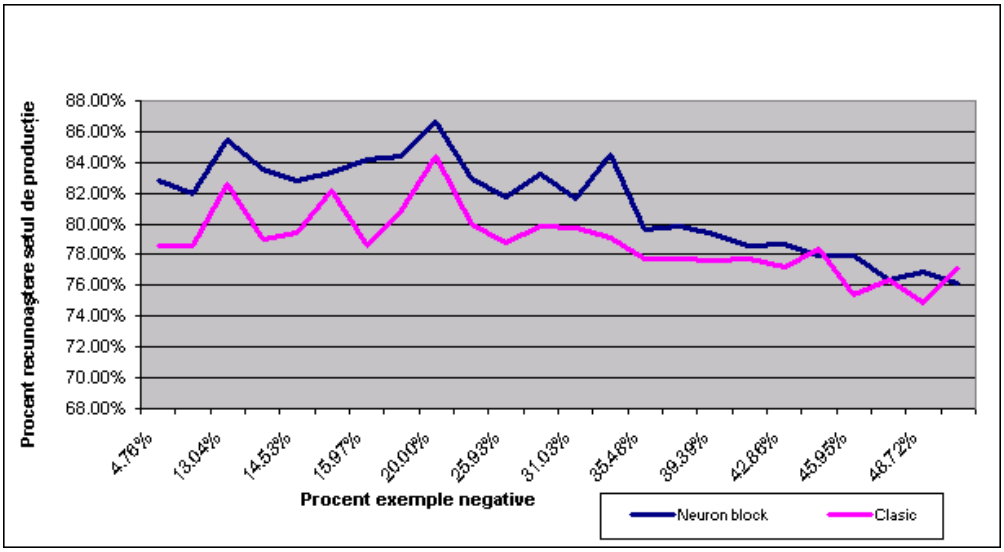


Figura 57. Procentele de recunoaştere pe setul de producţie obţinute de către o reţea multilayer perceptron cu neuron block şi o reţea multilayer perceptron clasică

Din cele trei figuri se observă clar superioritatea reţelei multilayer perceptron cu neuron **block** faţă de o reţea multilayer perceptron normală. Aceasta a obţinut valori mai mari pentru toate situaţiile în care setul de antrenare conţine până la 35% procente negative(din numărul exemplelor pozitive). După acest prag, diferenţele dintre cele două antrenări se reduc.

Faptul că reţeaua multilayer perceptron cu neuron **block** obţine procente de recunoaştere superioare celei clasice, denotă o maturitate a arhitecturii şi o comportare mai bună.

Aşa cum a reieşit şi din experimentul 5, o reţea multilayer perceptron cu neuron **block** oferă o mai bună claritate a ieşirilor. Astfel putem distinge acum cu uşurinţă între intrările care nu fac parte din clasele ce trebuie recunoscute de reţea şi intrările care nu sunt recunoscute de reţea. Acest lucru poate oferi noi posibilităţi de analiză a datelor şi o clasificare mai sigură a ieşirilor reţelei.

Concluzii

În urma efectuării experimentului 7(antrenarea unei reţele multilayer perceptron cu neuron **block** în vederea depistării unor valuri periculoase, şi comparaţie între aceasta şi o reţele multilayer perceptron clasică), au fost formulate următoarele concluzii:

- ✓ procentele obţinute de noua topologie de reţea (multilayer perceptron cu 1 neuron **block**) pe un set de producţie sunt cu până la 8% mai bune decât o reţea multilayer perceptron clasică
- ✓ şi pentru acest set de date procentele individuale pe clase de recunoaştere sunt mai mari decât cele obţinute de o reţea multilayer perceptron clasică antrenată fără exemple negative
- ✓ o mai mare claritate a ieşirilor la reţeaua multilayer perceptron cu neuron **block**; astfel tiparele negative sunt identificate cu uşurinţă ca o nouă clasă prezentă la ieşire

BUPT

5.3.3 Avantajele utilizării unei reţele multilayer perceptron cu neuron block faţă de o reţea neuronală de tip multilayer perceptron simplă

Acest subcapitol s-a ocupat de analizarea unei noi tipologii de reţea multilayer perceptron obţinută prin adăugarea unui neuron pe ultimul strat al unei reţele neuronale normale.

Ideea de adăugare a unui neuron suplimentar pe stratul de ieşire a fost puternic influenţată de prezenţa exemplelor negative în setul de antrenare. Pentru aceste exemple s-a dorit crearea unui neuron care să fie responsabil cu recunoaştere tiparelor negative prezente la intrare. Din această cauză l-am numit neuron **block** (responsabil cu blocarea anumitor tipare).

Experimentele desfăşurate de-a lungul acestui subcapitol au scos în evidenţa **rolul major pe care acesta îl deţine în cadrul procesului de antrenare prin simplificarea metodei de arbitrare** desfăşurată la nivelul ultimului strat. Neuronul(neuronii) câştigători erau desemnaţi prin aplicarea unor funcţii de activare care stabileau dacă un neuron este sau nu activ.

Inactivitatea tuturor neuronilor de pe stratul de ieşire induce o stare confuză deoarece poate fi văzută ca o nerecunoaştere a tiparului de la intrare sau poate fi o clasificare a acestuia ca un exemplu negativ.

Odată cu introducerea unui neuron suplimentar, procesul de arbitrare se simplifică simţitor. Avem astfel un neuron de ieşire şi pentru exemplele negative care pot apare la intrarea reţelei, eliminându-se astfel situaţia confuză de mai sus. De asemenea, desemnarea unui neuron ca fiind câştigător este foarte simplă, luând în acest caz neuronul cu ieşirea cea mai mare.

În exemplele rulate în acest subcapitol s-a observat **o îmbunătăţire simţitoare a procesului de recunoaştere cu până la 5%** pentru o astfel de reţea. Un alt aspect interesant a fost acela că **procentele de recunoaştere pentru reţeaua cu neuron block sunt superioare în orice situaţie** celei clasice. Acest aspect denotă o maturitate a arhitecturii tradusă printr-o uniformitate în procesul de recunoaştere. [66]

Alt aspect important este oferit de procentele de recunoaştere obţinute de fiecare clasă. Dacă privim cu atenţie toate datele observăm că acestea **sunt superioare celor obţinute de o reţea neuronală care a fost antrenată fără exemple negative**. Acest aspect este datorat simplităţii metodei de arbitrare şi considerăm că este o realizare importantă oferită de reţeaua cu neuron block.

Se poate afirma la finalul acestui subcapitol că **o reţea neuronală de tipul multilayer perceptron cu neuron block este superioară uneia clasice din toate punctele de vedere**. Rezultatele obţinute de aceasta sunt cu până la 5% mai bune, iar acest lucru este obţinut doar printr-o simplă modificare a stratului de ieşire(acesta trebuie să conţină un neuron **block**).

5.4 Experimente efectuate pe reţele multilayer perceptron cu mai mulţi neuroni block

În capitolul anterior am observat că utilizarea unei reţele multilayer perceptron cu un neuron **block** pe stratul de ieşire conduce la obţinerea unor procente de recunoaştere superioare faţă de cele obţinute de o reţea similară clasică (fără neuron block). Am dori să experimentăm în continuare acest fenomen prin

BUPT

introducerea mai multor neuroni block pe stratul de ieşire al reţelei multilayer perceptron. Fiecare dintre aceştia vor avea o funcţie de activare diferită şi din această cauză presupunem că vor manifesta comportamente diferite în procesul de recunoaştere.

Experimentele următoare utilizează 2 neuroni suplimentari pe stratul de ieşire. Fiecare dintre aceştia utilizează funcţii de activare diferite, iar activarea cel puţin a unuia face ca tiparul de la intrare să fie considerat un exemplu negativ.

5.4.1 Experimentul 8 (baza de date NIST 19)

Acest experiment este foarte asemănător cu experimentul 5, singura diferenţă constând în introducerea a încă unui neuron block pe stratul de ieşire.

Aşa cum am procedat până acum, şi pentru acest experiment s-au realizat multiple seturi de antrenare. Diferenţa dintre acestea constă în numărul de exemple negative conţinut de fiecare set în parte. Deoarece am dori o comparaţie cu datele obţinute la experimentul 5, am decis sa folosim exact aceleaşi seturi de date folosite de către acel experiment.

Astfel, antrenarea a pornit cu un set de 4 caractere (a,e,n, şi r), fiecare având câte 5000 de instanţe(exemple pozitive) precum şi câte 200 de caractere(literele b,d,h,l şi t), acestea din urmă având rol de exemple negative. Numărul exemplelor negative a scăzut la 40 fiecare, în intervalul [15% - 20%] pentru o observare mai fină a evoluţiei procentului de recunoaştere.

Reţeaua multilayer perceptron cu mai mulţi neuroni **block** antrenată pentru acest experiment are următoarea configuraţie:

- stratul de intrare conţine 1024 neuroni – am folosit ca intrări desene de 32x32 pixeli cu literele respective
- stratul ascuns are 700 de neuroni
- stratul de ieşire are 6 neuroni – 4 neuroni responsabili cu recunoaşterea celor 4 caractere; 2 neuroni suplimentari(neuron **block**) responsabil cu recunoaşterea tiparelor negative.

Funcţiile de activare corespunzătore fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică (pentru cei 4 neuroni responsabili cu detectarea literelor şi unul din neuronii suplimentari) şi funcţia tangentă hiperbolică pentru celălalt neuron block

Rezultatele obţinute de către reţeaua multilayer perceptron cu doi neuroni **block** sunt prezentate în Tabelul 15.

Datele sunt asemănătoare cu cele obţinute la experimentul 5. Acest lucru poate fi observat şi în Figura 58 şi Figura 59. În ambele figuri procentele de recunoaştere pentru reţeaua multilayer perceptron cu 1 neuron **block** sunt asemănătoare cu cele ale reţelei multilayer perceptron cu 2 neuroni **block**.

BUPT

Antrenare	Exemple negative %	Litera „a” %	Litera „e” %	Litera „n” %	Litera „r” %	Întreg alfabetul %	Set producţie %
	0	91.10	94.40	96.30	96.10	31.95	78.94
1	4.76	92.70	96.20	96.90	96.20	55.74	83.91
2	9.09	92.00	95.40	95.80	95.80	59.35	83.75
3	13.04	92.30	94.30	95.30	95.10	63.17	83.90
4	13.79	92.00	94.10	95.10	95.40	63.55	84.22
5	14.53	91.90	93.80	94.90	96.20	63.61	84.34
6	15.25	92.00	94.30	95.50	95.90	64.20	84.63
7	15.97	92.40	93.70	95.30	95.60	65.14	84.49
8	16.67	91.80	93.50	95.00	95.30	66.54	83.89
9	20.00	91.50	93.00	95.10	94.70	69.23	84.15
10	23.08	91.00	93.90	95.20	95.30	70.15	84.41
11	25.93	91.20	93.00	94.80	95.40	70.01	84.36
12	28.57	91.50	93.00	94.80	94.70	71.39	84.24
13	31.03	91.10	91.80	93.80	94.50	71.13	83.77
14	33.33	90.60	93.00	93.60	94.00	72.31	83.84
15	35.48	89.90	92.10	93.90	94.30	69.31	82.68
16	37.50	90.00	92.10	93.10	93.70	71.70	82.74
17	39.39	90.40	91.50	93.70	93.40	72.63	82.85
18	41.18	90.50	91.60	92.80	93.60	75.01	82.99
19	42.86	91.00	92.30	92.90	93.70	73.20	83.68
20	44.44	90.20	92.60	94.00	94.40	72.48	83.93
21	45.95	90.30	91.90	93.50	93.90	74.31	83.64
22	47.37	90.40	92.00	93.00	92.90	76.27	83.41
23	48.27	90.00	90.90	93.10	92.90	76.63	83.12
24	50.00	90.50	91.20	92.50	93.10	76.30	83.19

Tabelul 15. Rezultatele obţinute la experimentul 8

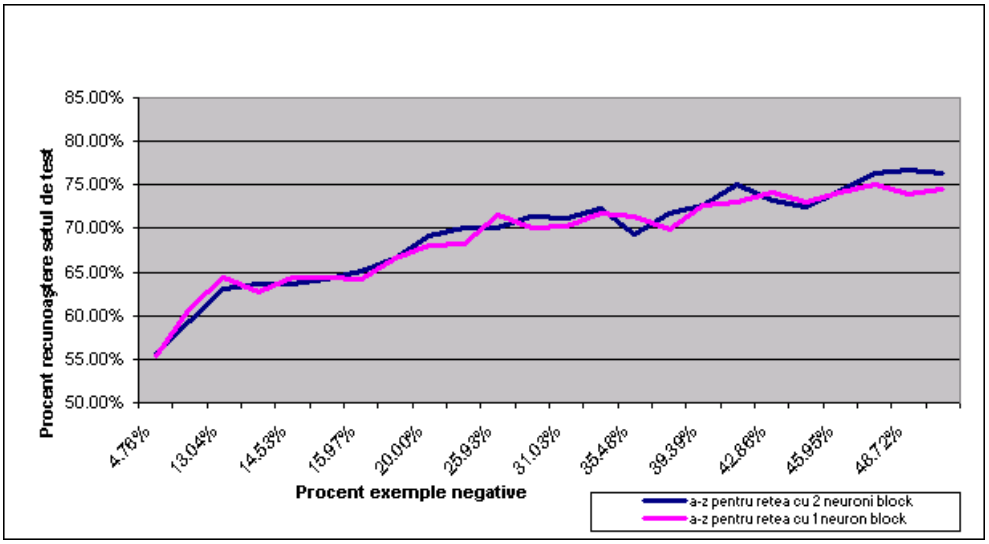


Figura 58. Procentele de recunoaştere obţinute pe un set de test pentru o reţele multilayer perceptron cu 1 şi 2 neuroni block pe stratul de ieşire

BUPT

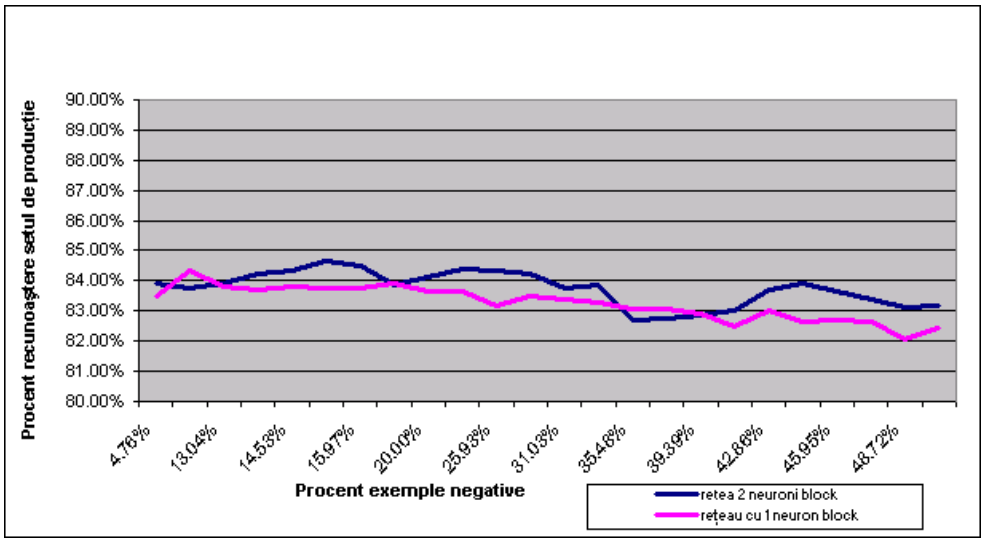


Figura 59. Procentele de recunoaştere obţinute pe un set de producţie pentru reţele multilayer perceptron cu 1 şi 2 neuroni block pe stratul de ieşire

Această „asemănare" a procentelor de recunoaştere pentru cele 2 tipuri de reţele este prezentă şi la nivel de clasă de recunoaştere. Astfel, reprezentând procentele de recunoaştere obţinute de clasele „a" şi „r" în Figura 60 se observă o situaţie similară.

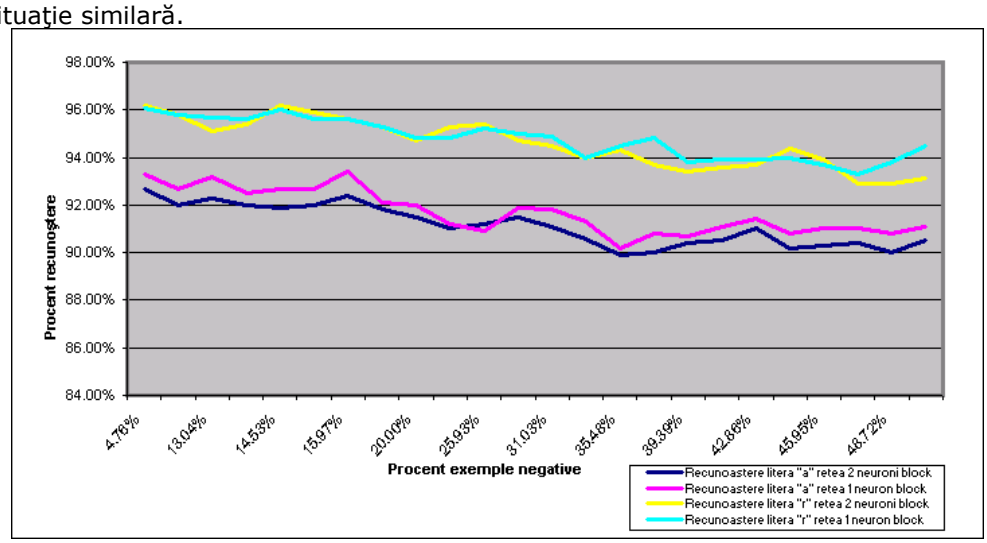


Figura 60. Procentele de recunoaştere obţinute de către literele „a" şi „r" pentru reţele multilayer perceptron cu 1 şi 2 neuroni block pe stratul de ieşire

Se observă că adăugarea celui de-al doilea neuron **block** nu a adus nici o îmbunătăţire a performanţelor. Rata de recunoaştere a rămas aceeaşi cu cea de la reţeaua multilayer perceptron cu 1 neuron block.

Următorul experiment are acelaşi scop ca şi cel curent, numai că a fost efectuat pe o altă bază de date.

BUPT

Concluzii

În urma efectuării experimentului 8(antrenarea unei reţele multilayer perceptron cu doi neuron **block** în vederea recunoaşterii caracterelor, şi comparaţie între aceasta şi reţeaua multilayer perceptron cu un neuron **block**), au fost formulate următoarele concluzii:

- ✓ pentru acest set de date nu s-a observat nici o deosebire clară(la nivel de performanţă pe setul de producţie) între reţeaua multilayer perceptron cu un neuron **block** şi reţeaua multilayer perceptron cu doi neuroni **block**
- ✓ recunoaşterea la nivel de clasă individuală de la ieşirea reţelei cu doi neuroni **block** prezintă valori asemănătoare cu cele obţinute de reţeaua multilayer perceptron cu un neuron **block**

5.4.2 Experimentul 9 (baza de date Ocean's Data)

Şi pentru acest experiment vom folosi pentru comparaţie datele obţinute anterior la antrenarea unei reţele multilayer perceptron cu 1 neuron **block**. Astfel, vom prelua setul de antrenare şi rezultatele folosite în cadrul experimentului 7, singura modificare fiind introducerea celui de-al doilea neuron **block** pe stratul de ieşire.

Configuraţia reţelei multilayer perceptron cu doi neuroni **block** obţinute este următoarea:

- stratul de intrare conţine 17 neuroni – am folosit ca intrări toţi parametrii din Anexa 2 mai puţin parametrul 8
- stratul ascuns are 160 de neuroni
- stratul de ieşire are 5 neuroni – avem 3 dintre aceştia răspunzători cu nivele de risc pentru valuri, iar 2 sunt responsabili cu detectarea tiparelor negative)

Funcţiile de activare corespunzătoare fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară $f(x) = x$.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică pentru primii 4 neuroni; pentru neuronul block, nou introdus avem o funcţie de activare de tip tangentă hiperbolică.

Antrenarea a pornit cu un set de 13.000 de exemple pozitive şi 650 exemple negative ajungând la 13.000 exemple pozitive şi 13.000 exemple negative. La fiecare pas de antrenare numărul exemplelor negative a crescut cu 650, iar în intervalul [15% - 20%] (exemple negative din numărul de exemple pozitive) numai cu 130, pentru o observare mai fină a evoluţiei procentului de recunoaştere.

Rezultatele obţinute de către multilayer perceptron cu doi neuroni **block** sunt prezentate în Tabelul 16.

Pentru o bună analiză am comparat aceste rezultate cu cele obţinute la experimentul 7. Reprezentările vizuale ale rezultatelor obţinute de cele două tipuri de reţele multilayer perceptron(cu 1 şi 2 neuroni **block** pe stratul de ieşire) pot fi observate în Figura 61 şi Figura 62.

BUPT

Antrenare	Exemple negative %	Risc inexistent %	Risc scăzut %	Risc mediu %	Set test %	Set producţie %
	0	85.34	94.44	64.29	28.00	61.93
1	4.76	84.48	97.92	58.57	84.55	82.85
2	9.09	85.34	97.22	61.43	87.27	82.00
3	13.04	84.48	96.53	62.86	90.00	85.48
4	13.79	81.90	95.14	65.71	90.73	83.53
5	14.53	82.76	93.75	64.29	90.36	82.82
6	15.25	78.45	95.14	65.71	90.00	83.34
7	15.97	79.31	94.44	64.29	90.09	84.16
8	16.67	78.45	91.67	61.43	89.73	84.39
9	20.00	86.21	93.75	60.00	90.45	86.66
10	23.08	87.93	95.83	57.14	91.36	82.96
11	25.93	88.79	97.22	58.57	91.45	81.73
12	28.57	78.45	98.61	61.43	92.73	83.22
13	31.03	76.72	99.31	62.86	93.18	81.64
14	33.33	77.59	98.61	60.00	92.55	84.47
15	35.48	76.72	97.22	58.57	92.73	79.63
16	37.50	73.28	96.53	54.29	93.18	79.82
17	39.39	67.24	95.83	55.71	92.55	79.36
18	41.18	68.97	95.14	52.86	92.00	78.55
19	42.86	69.83	96.53	50.00	91.36	78.66
20	44.44	68.10	95.83	42.86	91.00	77.91
21	45.95	68.97	93.75	44.29	91.09	76.30
22	47.37	66.38	97.22	45.71	91.82	77.04
23	48.27	68.97	96.53	40.00	91.36	75.98
24	50.00	68.10	95.83	42.86	91.09	76.21

Tabelul 16. Rezultatele obţinute la experimentul 9

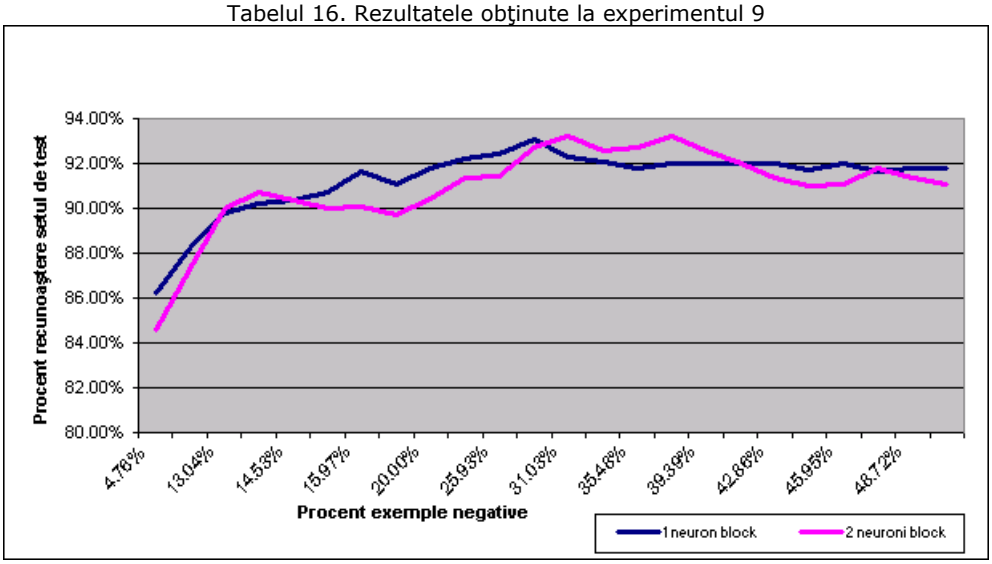


Figura 61. Procentele de recunoaştere obţinute pe un set de test pentru reţele multilayer perceptron cu 1 şi 2 neuroni block pe stratul de ieşire

BUPT

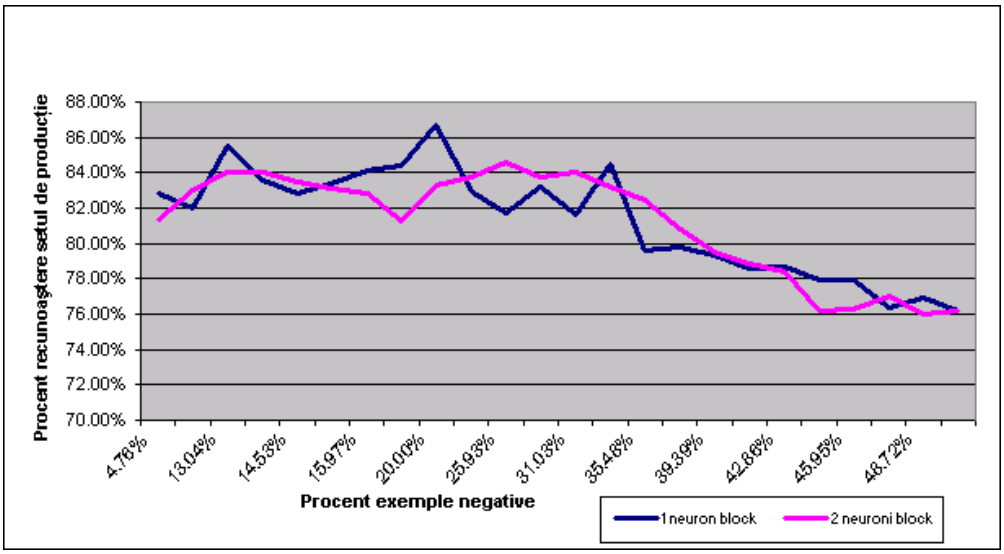


Figura 62. Procentele de recunoaştere obţinute pe setul de producţie pentru reţele de tip multilayer perceptron cu 1 şi 2 neuroni block pe stratul de ieşire

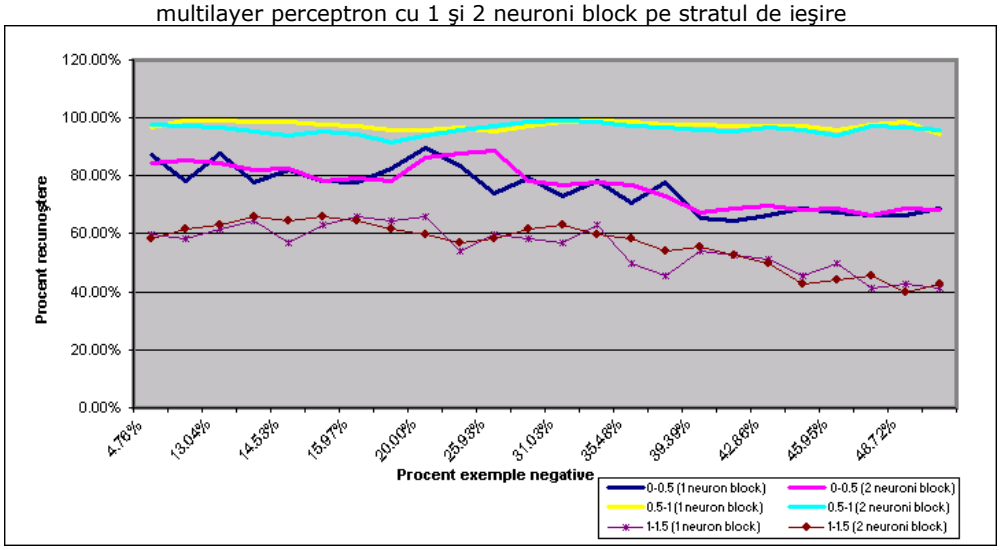


Figura 63. Procentele de recunoaştere obţinute pe clase de recunoaştere de reţele de tipul multilayer perceptron cu 1 şi 2 neuroni block pe stratul de ieşire

La fel ca la experimentul 8, observăm că introducerea unui neuron suplimentar pe stratul de ieşire nu a condus la obţinerea unor rezultate superioare. Am reprezentat grafic şi procentele de recunoaştere pe fiecare clasă de la ieşire (vezi Figura 63), dar lucrurile sunt similare cu cele discutate mai sus.

Aşa cum s-a observat din cele două experimente, introducerea celui de-al doilea neuron suplimentar pe stratul de ieşire, cu o altă funcţie de activare decât primul, nu conduce la îmbunătăţirea performanţelor reţelei neuronale de tip multilayer perceptron.

BUPT

Rezolvarea acestei probleme constă într-o împărţire mai riguroasă a exemplelor de antrenare în clase negative şi asocierea unui neuron block pentru fiecare din aceste clase. O propunere în acest fel ar fi introducerea dinamică a câte unui neuron block pentru fiecare clasă de exemple negative de la intrare.

Deocamdată, asocierea a mai mult de 1 neuron block pentru detectarea exemplelor negative nu reuşeşte să îmbunătăţească performanţele unei reţele neuronale.

Concluzii

În urma efectuării experimentului 9(antrenarea unei reţele multilayer perceptron cu doi neuron **block** în vederea detectării unor valuri periculoase, şi comparaţie între aceasta şi reţeaua multilayer perceptron cu un neuron **block**), au fost formulate următoarele concluzii:

- ✓ pentru acest set de date nu s-a observat nici o deosebire clară(la nivel de performanţă pe setul de producţie) între reţeaua multilayer perceptron cu un neuron **block** şi reţeaua multilayer perceptron cu doi neuroni **block**
- ✓ recunoaşterea la nivel de clasă individuală de la ieşirea reţelei cu doi neuroni **block** prezintă valori asemănătoare cu cele obţinute de reţeaua multilayer perceptron cu un neuron **block**
- ✓ pentru creşterea performanţelor, s-a propus pentru studiu, o împărţire mai riguroasă a exemplelor negative în mai multe clase şi asocierea unui neuron **block** pentru fiecare clasă nou formată

5.5 Concluzii

Acest capitol a fost dedicat în întregime experimentelor care au fost efectuate pentru testarea ideilor prezentate în Capitolul 4.

S-a început prin **evidenţierea importanţei pe care exemplele negative o au în cadrul procesului de antrenare**. Astfel, cu ajutorul unui set de antrenare în care numărul de exemple negative a fost mărit succesiv, s-a observat o îmbunătăţire remarcabilă a procentului de recunoaştere obţinut de reţeaua neuronală.

Pentru a putea compara rezultatele obţinute prin antrenarea mai multor reţele diferite, **a fost propusă o formulă de calcul generală a procentului de recunoaştere pentru un set de producţie** (Formula 5.1). În acest mod, toate experimentele ulterioare au ca punct comun valori ce pot fi comparabile.

În urma efectuării mai multor experimente pe două baze de date diferite, s-a observat un comportament comun al procentului de recunoaştere pe setul de producţie. Acesta lua valori maxime în intervalul [10% - 20%] (număr exemple negative din numărul de exemple pozitive).

Pentru a confirma acest lucru, s-a procedat la antrenarea unor reţele pe alte două baze de date recunoscute în acest domeniu: sonar şi diabet.

Rezultatele obţinute din experimentele pe setul de caractere, sonar şi diabet, au reprezentat punctul de plecare pentru generarea unui model general al procentului de recunoaştere în funcţie de numărul de exemple negative folosit.

Cu ajutorul programului DataFit s-a generat câte un model pentru fiecare bază de date folosită. Cele 3 modele au avut aceeaşi formă, dar coeficienţi diferiţi.

BUPT

Verificarea apartenenţei la acelaşi fenomen pentru seturile de coeficienţi a fost făcută printr-o analiză statistică cu ajutorul testului *t.*

Utilizând programul Origin6.0, s-a demonstrat că seturile de coeficienţi descriu acelaşi fenomen deoarece ele nu diferă în mod semnificativ. De asemenea programul a furnizat şi setul de parametrii ce caracterizează modelul general.

Modelul general a fost verificat pe baza de date Ocean's Data obţinându-se rezultate favorabile. **Modelul astfel generat este viabil şi reuşeşte să aproximeze evoluţia procentului de recunoaştere pe setul de producţie în funcţie de procentul de exemple negative folosit la antrenare**.

Al doilea set de experimente a fost efectuat pentru testarea noii metode de învăţare propusă în această teză: învăţarea pe baza corecţiei erorii cu exemple negative.

Pentru aceasta s-au antrenat mai multe reţele neuronale de tip multilayer perceptron la care s-a introdus un neuron adiţional pe stratul de ieşire. Deoarece acesta este responsabil cu semnalarea exemplelor negative prezente la intrare l-am numit neuron **block**.

În acest mod s-a realizat **o simplitate a metodei de arbitraj desfăşurată la nivelul ultimului strat**. Neuronul/neuronii câştigători erau desemnaţi de o funcţie de activare. Prin introducerea neuronului block putem considera ca fiind câştigător, neuronul cu valoarea de ieşire maximă.

Astfel se elimină şi confuzia generată de inactivitatea tuturor neuronilor pe stratul de la ieşire: nerecunoaştere a tiparului sau clasificare a acestuia ca fiind un exemplu negativ.

Modificarea astfel creată a reuşit **o îmbunătăţire substanţială, atât a procentului de recunoaştere general (cu până la 5%) cât şi a procentelor de recunoaştere individuale fiecărei clase de la ieşire**.

În continuarea experimentelor a fost testată o extensie a tehnicii de învăţare propusă anterior. Pe stratul de ieşire au fost adăugaţi doi neuroni suplimentari cu funcţie de activare diferită, dorindu-se o detecţie mai bună a modelelor negative.

Rezultatele au fost însă asemănătoare celor obţinute de reţeaua cu un neuron block, nici o performanţă notabilă nefiind înregistrată. Principala problemă a fost cauzată de „omogenitatea" exemplelor negative de la intrare. O împărţire a acestora în mai multe clase şi alocarea unui neuron pentru fiecare ar putea fi, în opinia autorului, o rezolvare a problemei.

BUPT

6. ÎMBUNĂTĂŢIREA TIMPULUI DE ANTRENARE A RNA PRIN FOLOSIREA CARACTERISTICILOR DATELOR DE INTRARE

Aşa cum am arătat în capitolul 1, reţelele neuronale artificiale sunt structuri ale căror proprietăţi le fac potrivite pentru rezolvarea unor probleme complexe de clasificare a tiparelor. Cu toate acestea, se întâmpină dificultăţi în aplicarea lor în procesul de rezolvare a unor probleme reale. Cea mai mare dificultate constă în găsirea unui algoritm care să poată calcula un set optimal de ponderi într-un timp relativ scurt.

Din această cauză, literatura de specialitate este plină de tehnici care ajută la creşterea vitezei de antrenare pentru o reţea neuronală. Aceste se grupează în două mari categorii:

- Tehnici care se ocupă de creşterea vitezei de rulare a algoritmului de antrenare; în această categorie avem tehnici de rulare a algoritmului pe diverse reţele de calculatoare[67], tehnici care rulează pe GPU(Graphics Unit Processor)[68], etc.
- Tehnici care se ocupă de optimizarea algoritmului de antrenare; în această categorie avem tehnici de utilizare a algoritmilor genetici pentru generarea iniţială a ponderilor[69], tehnici de iniţializare a ponderilor utilizând informaţie anterioară despre datele ce urmează a fi procesate[70], etc.

Din mulţimea de tehnici care încearcă optimizarea algoritmului de antrenare, ne interesează acele tehnici care utilizează elemente şi concepte de data mining. Domeniul Data mining este un domeniu recunoscut pentru performanţele deosebite în găsirea unor cunoştinţe în volume mari de date (KNN – Knowledge discovery in databases). În cadrul unui proces de tip KNN are loc o identificare a unor date valide, posibil utile în viitoarele prelucrări, şi totodată se încearcă identificarea unor tipare în datele de studiu.

UN proces de tip KNN este reprezentat în Figura 64.

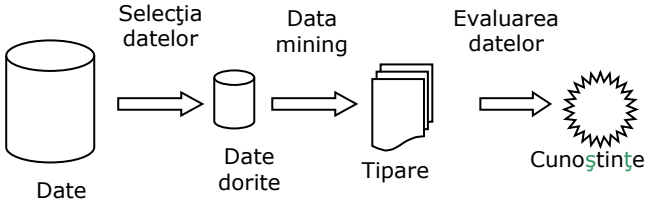


Figura 64. Paşii ce alcătuiesc un proces de tip KNN

BUPT

Scopul final al unui proces de tip KNN este extragerea anumitor caracteristici din datele prezentate la intrare astfel încât acestea să fie reprezentative pentru problema ce trebuie rezolvată.[71]

Prin înlăturarea tuturor acelor date care sunt irelevante sau redundante (faţă de noţiunile ce trebuie învăţate), procesul de descoperire a caracteristicilor este mult mai uşor şi mai rapid. Acest proces poartă denumirea de *selecţie a caracteristicilor dintr-un set de date.* Acest proces este de regulă bine definit şi poate fi urmărit pe toată durata desfăşurării lui.

În funcţie de natura căutării, procesul de selecţie a caracteristicilor trebuie sa-şi definească 4 elemente[72]:

- Punctul de start – este primul element care trebuie definit. Algoritmul porneşte prin selecţia unui punct în spaţiul format dintr-o mulţime de caracteristici. Această selecţie este importantă deoarece va afecta direcţia căutării. O posibilitate întâlnită deseori este pornirea de la o mulţime de caracteristici vidă şi adăugarea unei caracteristici în urma execuţiei unui pas din algoritm (poartă denumirea de căutare înainte). O altă posibilitate este pornirea cu mulţimea tuturor caracteristicilor şi eliminarea unei caracteristici la fiecare pas de execuţie (căutarea înapoi). Există şi a treia posibilitate, şi ea presupune alegerea unui punct undeva la mijloc şi navigarea din acel punct către o margine a mulţimii.
- Modul de organizare al căutării. Tehnicile de căutare euristice sunt mult mai rapide şi oferă rezultate mai bune decât cele exhaustive. Singura problemă a acestora este că nu garantează rezultatul obţinut ca fiind cel optim.
- Strategia de evaluare. Acest element este de departe cel mai important factor în întreg procesul de selecţie a caracteristicilor. Rolul lui este de a preciza modul în care se evaluează un număr de caracteristici. O primă metodă de evaluare este cea euristică în care mulţimea de caracteristici este analizată prin prisma caracteristicile generale ale datelor. A doua metodă presupune un algoritm de inducţie ce este folosit pentru analizarea mulţimii de caracteristici.
- Criteriul de oprire. Decizia de încetare pentru selecţia celor mai bune caracteristici dintr-o mulţime este luată în următoarele cazuri: putem să ne oprim din procesul de adăugare/ştergere de caracteristici, atunci când nu mai există nici o caracteristică care să îmbunătăţească într-un fel selecţia curentă; putem revizui setul curent atâta timp cât valoarea acestuia nu scade; putem continua generarea noului set până când toate caracteristicile ce pot fi adăugate s-au epuizat, în acest caz selectăm cel mai bun set obţinut până atunci.

Mai mulţi autori menţionează utilizarea unor tehnici de selecţie a celor mai bune caracteristici ca fiind un mod de optimizare a procesului de rezolvarea a unei probleme. Vom prezenta în continuare câteva din tehnicile curente de selecţie a caracteristicilor şi care au fost aplicate cu succes în rezolvarea unor probleme care au implicat reţele neuronale. La sfârşitul capitolului vom face şi o comparaţie între rezultatele obţinute de către aceştia folosind diverse tehnici şi rezultatele obţinute utilizând tehnica prezentată în acest capitol.[73]

BUPT

În 1997 Jain şi Zongker definesc problema selecţiei de caracteristici în felul următor[74]:

Dându-se un set de caracteristici, trebuie selectate un număr dintre acestea care obţin cele mai bune rezultate când este aplicat un anume sistem de clasificare.

Pe lângă reducerea costurilor în problema de recunoaştere (deoarece se reduce numărul de caracteristici care trebuie să fie colectate) are loc, în anumite cazuri, şi o mai bună clasificare datorată reducerii numărului de exemple ce trebuie procesat. Aceştia au studiat un număr mare de algoritmi propuşi pentru selecţia unui număr de caracteristici, şi au ales ca fiind cel mai bun, algoritmul *„Sequential Forward Floating Selection”* (SFFS) dezvoltat de Pudil şi alţii.[75]

În 2003, Kim şi Street propun o nouă metodă, denumită ELSA (Evolutionary Local Selection Algorithms) şi o utilizează în decizii manageriale. Această metodă utilizează reţelele neuronale artificiale pentru a descoperi diverse combinaţii de caracteristici. În final, serviciile sau produsele achiziţionate de partea managerială trebuiau să aibă numai acele caracteristici evidenţiate anterior de către ELSA.[76] Autorii concluzionează că un model ce combină ELSA cu reţelele neuronale artificiale produce rezultate satisfăcătoare numai atunci când partea managerială are un scenariu decizional creat anterior.

În 2007 Gigli împreună cu alţi cercetători, prezintă o nouă arhitectură pentru un algoritm de data mining dezvoltat în vederea împărţirii pe clase a unui set de exemple date[77]. Algoritmul integrează o librărie de extragere de caracteristici împreună cu data mining şi fuziune de date, pentru a optimiza şi automatiza soluţia de clasificare prezentată în final.

Ei descriu cum selecţia de caracteristici şi algoritmii de data mining se combină cu ajutorul algoritmilor genetici utilizând o sursă de date comună. Totodată, pentru surse de date multiple, ei propun utilizarea unora dintre cele mai bune tehnici de fuziune de date, acesta realizându-se tot cu ajutorul algoritmilor genetici.

Toate aceste studii relevă faptul că aplicarea unei tehnici de tip selectare de caracteristici la un set de date conduce la obţinerea unui set de date mai mic şi mai reprezentativ pentru problema ce trebuie rezolvată. Acest set restrâns conservă toate caracteristicile necesare în vederea rezolvării problemei propuse şi optimizează timpul şi resursele ce sunt necesare. De asemenea, există cazuri în care datele selecţionate printr-o astfel de tehnică obţin rezultate mai bune, lucru datorat acurateţei de care acestea dau dovadă.

Astfel, antrenarea unei reţele neuronale utilizând un set de caracteristici ale datelor de intrare pare simplă şi foarte naturală. Această idee derivă şi din faptul că o mare parte din datele de intrare ale reţelei neuronale, utilizate la antrenarea ei, sunt recurente sau conţin informaţii care nu sunt necesare în acest proces. Trebuie precizat că procesul de selectare de caracteristici dintr-un set de date care să reprezinte cu succes datele iniţiale este încă o provocare şi nu este 100% rezolvat. Credem că domeniul care este cel mai apropiat de o rezolvare totală este data mining.

Această ipoteză se bazează pe faptul că data mining este un domeniu care prelucrează cantităţi foarte mari de date şi de aceea, calitatea acestor date este foarte importantă. Astfel, caracteristici neimportante pot avea un efect negativ asupra prelucrărilor ulterioare. Din această cauză domeniul data mining pare cel mai potrivit pentru o rezolvare totală a acestei probleme.[78]

BUPT

6.1 Selectarea caracteristicilor unui set de antrenare utilizând tehnici de data mining

Pentru a putea obţine un set de caracteristici ale setului de antrenare vom încerca iniţial să găsim corelaţii în setul de date. Utilizând tehnici de data mining, trebuie să găsim un algoritm care să fie capabil de selecţia celor mai bune caracteristici dintr-un set de date.

Această tehnică presupune două componente:

- o tehnică de evaluare a caracteristicilor
- un algoritm de căutare pentru a selecta cele mai bune caracteristici.

Tehnica de evaluare folosită este CFS (Correlation based Feature Selection) şi este un algoritm care combină o formula de evaluare ce conţine o tehnică de măsură a corelaţiilor dintre caracteristici cu o strategie de căutare de tip euristic.

Tehnica are la bază ipoteza că un set bun de caracteristici trebuie să conţină numai elemente care sunt puternic corelate între ele şi slab corelate (sau chiar deloc) cu elemente conţinute în alte seturi.

Dacă avem un set de test în care cunoaştem valoarea corelaţiei dintre fiecare componentă şi variabila exterioară reprezentată de acea componentă, şi de asemenea, intercorelarea dintre fiecare două componente este cunoscută, putem defini valoarea corelaţiei dintre setul respectiv şi variabila exterioară după formula:

$$r_{zc} = \frac{k\overline{r_{zi}}}{\sqrt{k + k(k-1)\overline{r_{ji}}}} \tag{6.1}$$

unde:

- r_{zc} este corelaţia dintre setul respectiv şi variabila exterioară
- k este numărul de componente
- $\overline{r_{zi}}$ este media aritmetică a corelaţiilor dintre componente şi variabila exterioară
- $\overline{r_{ji}}$ este media aritmetică a intercorelaţiilor dintre componente

CFS poate fi văzut ca un simplu algoritm de filtrare care grupează în mulţimi mai multe caracteristici în funcţie de o funcţie euristică de evaluare a corelaţiei. Caracteristicile principale din setul obţinut sunt strâns corelate între ele şi slab corelate cu alte seturi.

Astfel, caracteristicile irelevante trebuie ignorate deoarece ele au o slabă corelare cu setul respectiv. Caracteristicile redundante trebuie eliminate din listă chiar dacă ele sunt puternic corelate cu alte caracteristici, deoarece ele deja aparţin altor caracteristici.

Acceptarea unei caracteristici va depinde exclusiv de puterea de predicţie a cât mai multor clase din setul de test, clase care nu mai pot să fie revendicate de alte caracteristici conform ecuaţiei (6.1).

Experimentele desfăşurate pe seturi de date au arătat că CFS a reuşit să identifice cu rapiditate caracteristicile importante pentru acele seturi care să nu

BUPT

depindă puternic de alte caracteristici. De asemenea, algoritmul elimină cu succes toate acele caracteristici redundante, care nu sunt relevante sau sunt parţial distruse, acţionând astfel în detrimentul prelucrărilor ulterioare a datelor.

Pentru algoritmul de căutare a fost utilizat BestFirst, un algoritm de tip greedy combinat cu strategie backtracking.[80]

Algoritmul greedy utilizează tehnica alpinistului(hill climbing) având ca scop maximizarea/minimizarea unei funcţii *f(x)* în care *x* ia valori într-un spaţiu discret. Dacă reprezentăm valorile pe care poate să le ia *x* fiind nişte noduri într-un graf, arcele care unesc două astfel de noduri reprezintă similitudinile dintre acele două noduri(stări).

Pentru rezolvarea unei probleme se pot aplica mai multe metode ale acestui algoritm. Selecţia anterioară a atributelor (FS – Forward selection) este una dintre cele mai simple şi mai des folosite metode. Are ca punct de plecare o mulţime vidă şi adaugă câte un atribut la fiecare pas de execuţie până când toate atributele au fost adăugate. Adăugarea este făcută astfel încât valoarea atributelor adăugate până în acel moment să fie maximă (greedy). Astfel, la fiecare pas FS selectează atributul care reuşeşte cel mai bine să fie corelat cu restul atributelor. Adăugarea unui atribut la fiecare pas de rulare face ca atributele deja adăugate la setul curent să nu mai poată fi scoase.

O altă metodă este cea de eliminare a atributelor (BE – Backward elimination). Această metodă este similară cu FS, singura diferenţă fiind aceea că BE porneşte cu toate atributele şi elimină câte un atribut din set la fiecare pas. În acelaşi mod ca şi FS, un atribut eliminat într-un pas nu mai poate fi adăugat într-un pas anterior.

Faptul că un atribut este adăugat/scos printr-o acţiune definitivă este un lucru riscant deoarece atributul poate ulterior să-si dovedească valoarea într-un pas viitor al algoritmului. Din această cauză o metodă mai sigură este una hibridă, în care, la fiecare pas, la algoritm putem adăuga sau şterge un atribut. Şi la pornire avem o libertate mai mare relativ la mulţimea de start. Astfel, putem porni fie cu o mulţime goală, fie completă sau cu un număr aleatoriu de elemente, deoarece putem adăuga sau scoate ulterior ce element dorim.

Algoritmul va parcurge întregul graf, trecând dintr-un nod în alt nod, la fiecare pas urmărindu-se maximizarea(minimizarea) funcţiei *f* până când este atins un punct de maxim(minim).

Algoritmul se termină atunci când nu se mai poate face nici un fel de îmbunătăţire asupra funcţiei *f*. În Figura 65 se poate observa reprezentarea grafică pentru o funcţie care are un singur minim local. Din păcate acest gen de funcţii sunt foarte rar întâlnite în lumea reală, cele mai frecvente fiind cazurile în care predomină funcţiile cu mai multe puncte locale de minim/maxim (Figura 66).

Dacă o funcţie are mai multe puncte de maxim(minim) local, metoda nu garantează obţinerea optimului global, ea furnizând doar optime locale care depind de punctul iniţial ales. Acest fapt se datorează situaţiei în care algoritmul găseşte un punct de optim local şi trebuie să stabilească dacă acesta este şi cel global. Scopul final al algoritmului este găsirea unui punct optim local care să fie cât mai aproape de punctul global.

Ca un dezavantaj pentru tehnicile de căutare de tip alpinist este faptul că nu reuşesc un echilibru între cele 2 obiective principale ale unei tehnici de căutare:

- *exploatarea* celor mai bune soluţii găsite până în momentul respectiv
- *explorarea* spaţiului de căutare

BUPT

Astfel, tehnicile de tip alpinist exploatează cea mai bună soluţie găsită pentru o posibilă îmbunătăţire, dar neglijează explorarea unei mari porţiuni din spaţiul de căutare

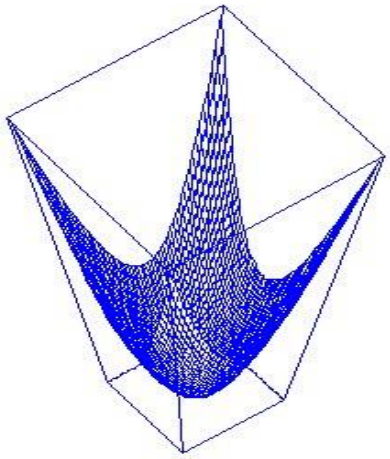


Figura 65. Reprezentarea grafică pentru o funcţie cu un singur minim local

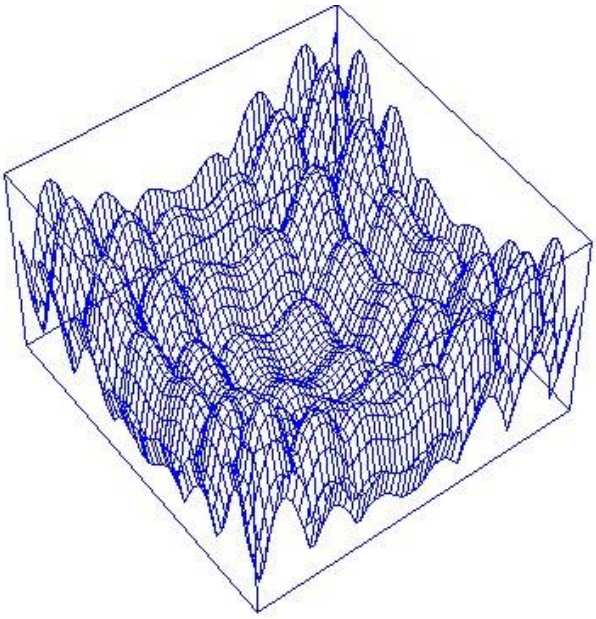


Figura 66. Reprezentarea grafică pentru o funcţie cu mai multe puncte de maxim locale

Combinând cele două componente, tehnica de evaluare a caracteristicilor şi algoritmul de căutare vom încerca să experimentăm căutarea celor mai bune caracteristici pe un set de date folosit în experimentele anterioare.

BUPT

În capitolul precedent o mare parte din experimente au avut ca date de intrare seturi de caractere oferite de baza de date internaţională NIST 19. Fiecare caracter este reprezentat pe 1024 de pixeli (o matrice de 32x32 pixeli), un pixel putând lua valori de 0 sau 1.

O mare parte din cei 1024 de pixeli nu sunt folosiţi în procesul de recunoaştere. Din această cauză aplicarea unui algoritm pentru extragerea unor caracteristici ale datelor de intrare şi aplicarea acestor caracteristici ca şi intrări pentru reţeaua neuronală, sperăm să furnizeze rezultate mai rapide şi, poate şi o îmbunătăţire a performanţelor obţinute de reţeaua neuronală.

6.2 Experimente folosind caracteristici ale datelor de intrare

6.2.1 Programul Weka

Pentru selecţia caracteristicilor am utilizat unul din cele mai folosite programe în domeniul data mining, programul Weka (Waikato Environment for Knowledge Analysis)[81][82].

Weka este o colecţie de algoritmi de învăţare utilizaţi preponderent în domeniul data mining. Weka conţine unelte de dezvoltare pentru[83]:

- preprocesare de date
- clasificare, regresie de date
- clusterizare
- reguli de asociere şi vizualizare

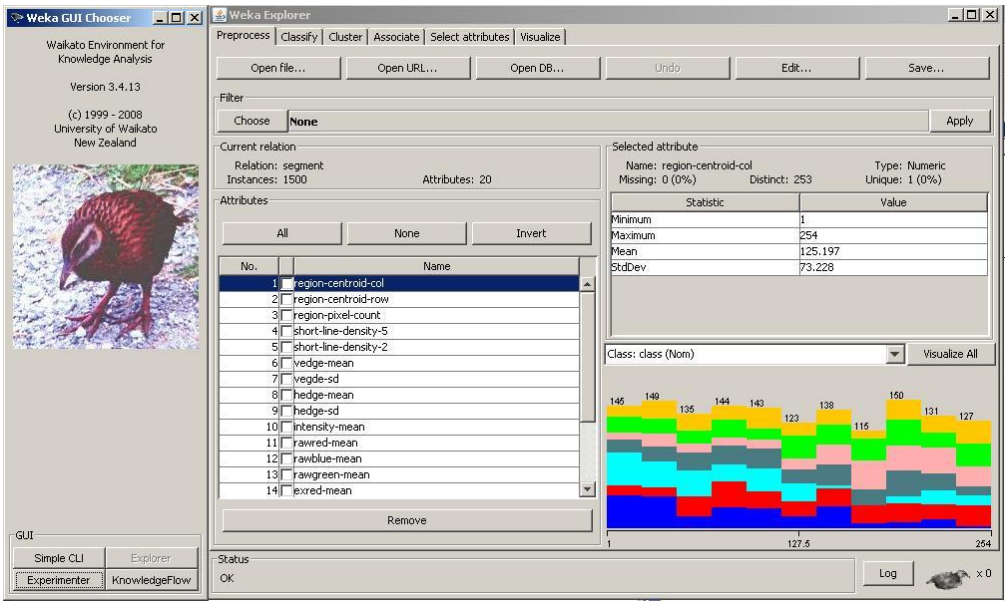


Figura 67. Fereastra principală a programului Weka împreună cu fereastra de explorer

BUPT

În Figura 67 se poate observa interfaţa principală a programului împreună cu fereastra de Explorer. Utilizarea acestuia este destul de facilă şi nu presupune noţiuni avansate de data mining.

6.2.2 Experimente

Experimentele derulate în acest capitol încearcă determinarea a două aspecte privind antrenarea reţelelor neuronale cu caracteristici ale datelor de intrare:

- Acurateţea caracteristicilor obţinute. Vom încerca să determinăm dacă antrenând o reţea neuronală artificială cu caracteristici ale datelor de intrare obţinem rate de recunoaştere comparabile cu cele obţinute de reţeaua antrenată cu datele originale.
- Timpul de antrenare. Se încearcă determinarea timpilor de antrenare pentru reţeaua neuronală antrenată cu caracteristici, timpi care ar trebui să fie de câteva ori mai mici decât la reţeaua antrenată cu date normale.

Experimentul 9

Deoarece pentru o comparaţie cu o reţea clasică, am dorit să utilizăm rezultatele obţinute în cadrul Experimentului 1, şi Experimentul 9 a constat în antrenarea unei reţele neuronale în recunoaşterea caracterelor „a”, „e”, „n” şi „r”.

Un prim pas a fost alegerea celor mai bune caracteristici ale datelor din setul de intrare. Utilizând programul Weka am obţinut un număr de 100 de caracteristici, numărul acestora fiind prezentat în Anexa 4.[86] Repartiţia acestora în matricea de 32x32 de puncte se poate observa şi din Figura 68. Se poate observa foarte uşor cum o serie de puncte nu au fost selectate, prezenţa acestora nefiind necesară la antrenare.

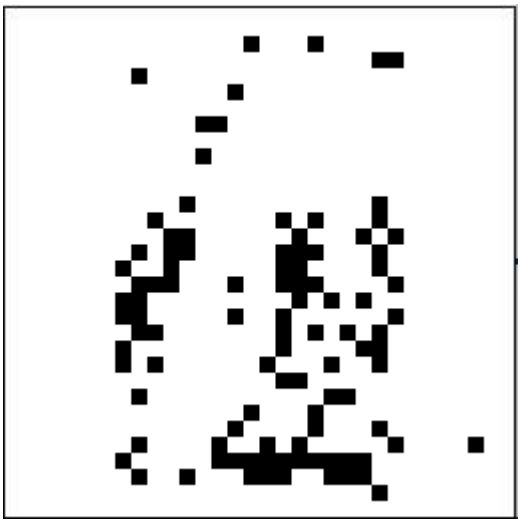


Figura 68. Repartiţia punctelor selectate din matricea de 32x32 ca fiind cele mai bune caracteristici pentru caracterele „a”, „e”, „n” şi „r”

BUPT

Fiecare dintre caracteristicile selectate determină schimbări majore în ieşirile reţelei neuronale. Astfel am exemplificat în Figura 69 modul în care caracteristicile 523 şi 658 influenţează recunoaşterea literei „a” de către reţeaua neuronală.

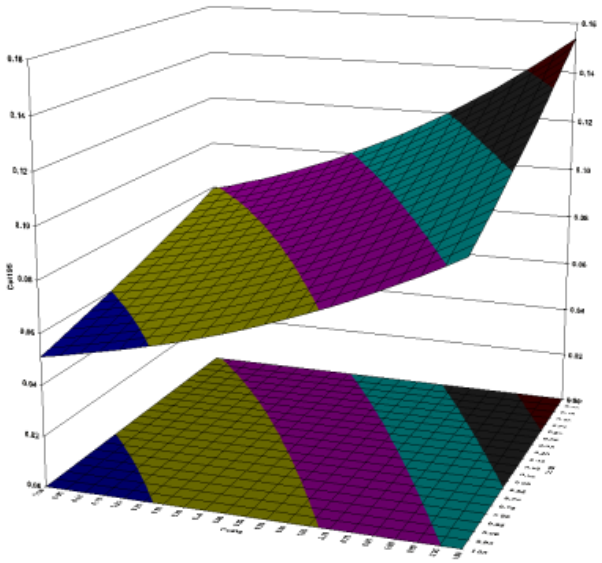


Figura 69. Reprezentarea grafică a influenţei exercitate de caracteristicile 523 şi 658 asupra literei „a”

O primă concluzie este reducerea masivă a setului de intrare; dacă la prima reţea avem 1024 de neuroni de intrare, pentru a doua numărul acestora a scăzut la 100 de neuroni.

Reamintim configuraţia reţelei de la Experimentul 1 ca fiind de tip multilayer perceptron cu 1 strat ascuns. Configuraţia acesteia este următoarea:

- stratul de intrare conţine 1024 neuroni – am folosit ca intrări desene de 32x32 pixeli cu literele respective
- stratul ascuns are 700 de neuroni
- stratul de ieşire are 4 neuroni – fiecare dintre cei 4 neuroni este responsabil cu recunoaşterea unui caracter

Funcţiile de activare corespunzătore fiecărui dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

În cadrul acestui experiment această reţea va purta denumirea de reţea neuronală clasică. A doua reţea folosită poartă denumirea de reţea neuronală antrenată cu caracteristici ale datelor de intrare şi are următoarea configuraţie:

- stratul de intrare conţine 100 neuroni – am folosit numai cele 100 de intrări selectate de Weka ca fiind cele mai bune caracteristici pentru setul de intrare.
- stratul ascuns are 178 de neuroni

BUPT

- stratul de ieşire are 4 neuroni – 4 neuroni responsabile cu recunoaşterea celor 4 caractere.

Funcţiile de activare corespunzătore fiecăruia dintre cele 3 straturi sunt:

- pentru stratul de intrare avem ca funcţie de activare – funcţia liniară f(x) = x.
- pentru stratul ascuns avem ca funcţie de activare – funcţia logistică
- pentru stratul de ieşire avem ca funcţie de activare – funcţia logistică

Antrenarea a fost făcută pentru un set ce conţine 16% exemple negative. Am preluat astfel concluziile obţinute în urma antrenării reţelelor neuronale pentru determinarea procentului optim de exemple negative.

Setul de antrenare a fost compus din 23.200 de litere, din care 20.000 de exemple pozitive (fiecare dintre cele 4 litere având câte 5000 de instanţe) şi 3200 de exemple negative. Setul de test a fost format din 26.000 de litere, fiecare literă având câte 1000 de instanţe. Setul de producţie a fost calculat conform formulei (5.1).

Rezultatele obţinute de cele două reţele se pot observa în Tabelul 17 şi sunt reprezentate vizual în Figura 70.

Reţea	Exemple negative %	Litera „a” %	Litera „e” %	Litera „n” %	Litera „r” %	Întreg alfabetul %	Set producţie %
1	15.97	90.70	93.20	95.00	94.80	59.80	83.39
2	15.97	90.80	93.20	95.30	94.70	59.88	83.89

Tabelul 17. Procentele de recunoaştere obţinute în cadrul Experimentul 9

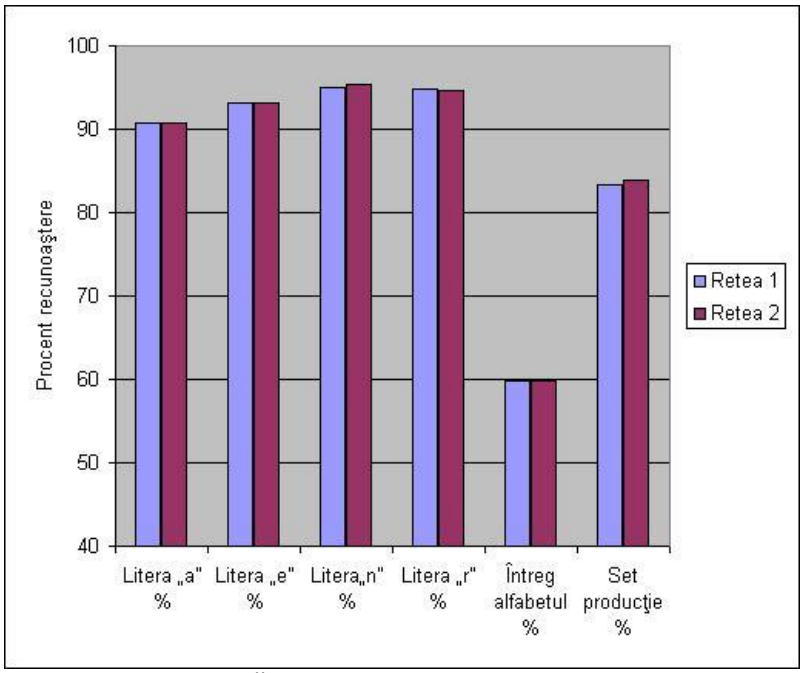


Figura 70. Reprezentarea grafică a procentelor de recunoaştere obţinute la Experimentul 9

BUPT

Observaţia imediată este aceea că procentele de recunoaştere pentru cele două reţele neuronale sunt foarte apropiate, existând şi cazuri în care reţeaua neuronală antrenată cu caracteristici ale datelor de intrare s-a comportat mai bine decât prima reţea.

Acest aspect denotă robusteţea algoritmului de selecţie a caracteristicilor folosit şi confirmă încă o data faptul că un set de caracteristici poate reprezenta cu succes întreg setul de date original.

Cazurile în care am obţinut rate de recunoaştere mai bună (pentru literele „a", „n" şi „r") reprezintă situaţii posibile în care un set de caracteristici se comportă mai bine decât setul original.

Situaţiile în care setul de antrenare prezintă zgomote sau tipare greşite sunt întâlnite destul de frecvent în problemele de recunoaştere conducând astfel la o antrenare greoaie şi o îngreunare a procesului de recunoaştere.

Al doilea obiectiv urmărit la acest experiment este legat de timpul necesar pentru întreg procesul de antrenare. Am monitorizat timpul în care se execută mai multe epoci de antrenare, iar rezultatele sunt prezentate în Tabelul 18.

Numărul de epoci	Reţea 1 (sec)	Reţea 2 (sec)
1	898	84
4	3612	338
6	5398	505
10	9008	841
22	19795	1845
30	26994	2518
38	38202	3194
40	35921	3362
50	44898	4200
55	49391	4619
60	53878	5043
65	58370	5463
70	62857	5881
76	68247	6385
80	71840	6720
85	76333	7142
90	80818	7562
95	85311	7981
100	89799	8400

Tabelul 18. Timpii de antrenare obţinuţi în cadrul Experimentul 9

Se poate observa diferenţa uriaşă între timpii necesari antrenării pentru reţeaua neuronală iniţială şi reţeaua neuronală antrenată numai cu caracteristici ale datelor de intrare.

Pentru o mai clară diferenţiere a acestor timpi, am reprezentat grafic în Figura 71 rezultatele obţinute şi astfel, se poate uşor observa că timpul necesar antrenării reţelei 2 este de aproape 11 ori mai mic decât timpul necesar antrenării reţelei 1.

BUPT

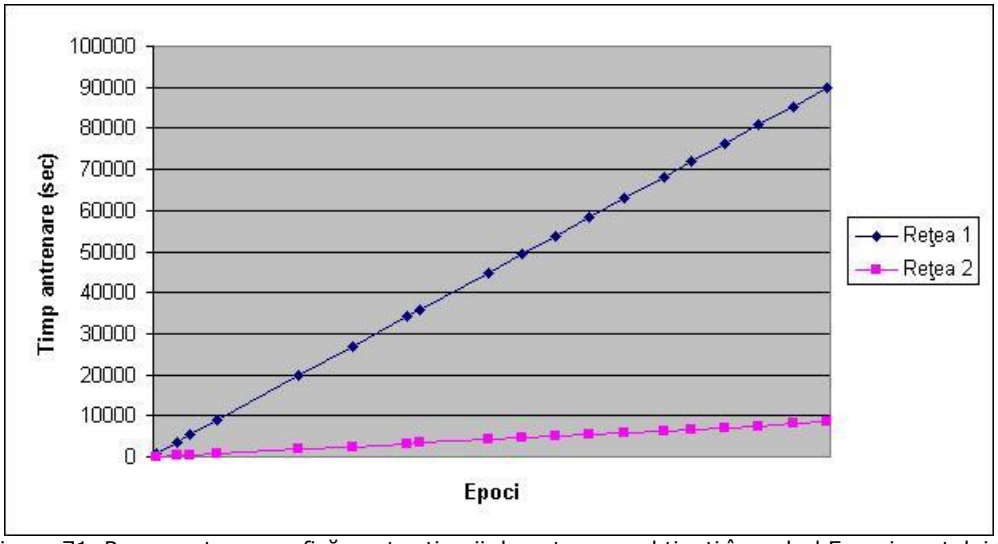


Figura 71. Reprezentarea grafică pentru timpii de antrenare obţinuţi în cadrul Experimentului 9

Extragerea caracteristicilor importante dintr-un set de antrenare al reţelelor neuronale poate este o operaţie care conduce la o reducere importantă a timpilor de antrenare păstrând neschimbate performanţele reţelei.

Pentru validarea clară a acestei prime concluzii, la fel ca şi la restul experimentelor, dorim să repetăm acest tip de experiment şi cu alte baze de date diferite ca şi conţinut de aceasta.

Experimentul 10

Setul de baze de date ales pentru acest experiment este unul foarte cunoscut şi asupra căruia s-au făcut mai multe experimente de acest fel. [62]

Trebuie precizat că au fost selectate numai câteva din bazele de date conţinute de acest set, deoarece s-a dorit şi o comparaţie cu alte studii efectuate asupra acestor baze de date.

Datele folosite pentru comparaţie au fost obţinute prin folosirea următoarelor tehnici de selecţie a caracteristicilor[76][87]:

- utilizând o simplă reţea neuronală
- metoda Bagging
- metoda Ada Boost
- GEFS (Genetic Ensemble Feature Selection)
- MEE (Meta-Evolutionary Ensembles)

Utilizând câteva din bazele de date din acest set am aplicat procedura de selecţie a celor mai bune caracteristici dezvoltată în acest capitol şi cu acestea am antrenat mai multe reţele neuronale.

Rezultatele obţinute se pot observa în Tabelul 19 alături de rezultatele obţinute de alte metode de selecţie a celor mai bune caracteristici.

Se poate observa cu uşurinţă cum metoda propusă de noi a furnizat cele mai bune rezultate, reuşind să întreacă mare parte din celelalte metodele. Rezultatele obţinute certifică astfel metoda de selecţie a celor mai importante caracteristici dintr-un set de date.

BUPT

Baza de date	Reţea neuronală simplă	Bagging	Ada Boost	GEFS	MEE	Reţeaua noastră
Credit-a	84.3	86.2	84.3	86.8	86.4	87.2
Credit-g	71.7	75.8	74.7	75.2	75.6	78.2
Diabetes	76.4	77.2	76.7	77.0	76.8	78.99
Hepatitis	81.5	82.2	80.3	83.3	84.9	84.9
Hypo	93.8	93.8	93.8	94.1	93.9	94.8
Iris	95.9	96.0	96.1	96.7	96.5	96.6
Segment	92.3	94.6	96.7	96.4	93.2	96.4
Sick	95.2	94.3	95.5	96.5	99.3	99.5
Sonar	80.5	83.2	87.0	82.2	85.2	90.0

Tabelul 19. Bazele de date alese şi rezultatele obţinute în cadrul experimentului 10

6.3 Concluzii

În acest capitol a fost prezentată **o metodă de îmbunătăţire a timpului de antrenare a RNA prin utilizarea caracteristicilor datelor de intrare**. Procedura constă într-o preprocesare iniţială a datelor pentru selecţia celor mai importante caracteristicilor ale acestora. Operaţia de preprocesare poate fi văzută ca o operaţie de filtrare a datelor de intrare. În urma acestei operaţii o parte din datele de intrare (în special cele redundante şi cele nefolosite în procesul de antrenare) sunt eliminate, ajungându-se astfel la un set minimal de date, util în rezolvarea problemei. Operaţia de selecţie de caracteristici a fost obţinută cu ajutorul tehnicilor de data mining.

Tehnica propusă este formată din două componente:**o tehnică de evaluare a caracteristicilor şi un algoritm de căutare pentru a selecta cele mai bune caracteristici**.

Tehnica de evaluare folosită este CFS (Correlation based Feature Selection) . Acesta este un algoritm format din două componente principale: o formula de evaluare ce conţine o tehnică de măsură a corelaţiilor dintre caracteristici şi o strategie de căutare de tip euristic.

Tehnica de evaluare are ca rezultat final furnizarea unui set de caracteristici puternic corelate între ele şi slab corelate (sau chiar deloc) cu elemente conţinute în alte seturi.

Pentru **algoritmul de căutare a fost utilizat BestFirst, un algoritm de tip greedy combinat cu strategie backtracking**. Scopul principal al acestuia este găsirea punctului de maxim local care să fie cât mai apropiat de cel general. Acesta trebuie să „aleagă” cel mai bun set de caracteristici furnizat de către tehnica de evaluare precedentă.

Setul de caracteristici furnizat de operaţia de preprocesare este folosit în continuare pentru antrenarea reţelei neuronale.

Rezultatele obţinute în urma antrenării unei reţele neuronale cu aceste caracteristici sunt foarte bune, **deoarece ratele de recunoaştere rămân egale**, şi în unele cazuri sunt chiar superioare, celor obţinute de reţele neuronale antrenate cu datele de intrare iniţiale.

Câştigul cel mai mare a fost reflectat însă, de **micşorarea semnificativă a timpului de antrenare, acesta ajungând să fie de 11 ori mai redus faţă de o antrenare normală.** Este un aspect deosebit de important în procesul de antrenare

BUPT

al reţelelor neuronale, deoarece timpul de antrenare este o problema majoră atunci când seturile de antrenare sunt foarte mari, sau când structura reţelei neuronale este mare.

Experimentele au fost efectuate pe mai multe seturi de date, dorindu-se astfel o verificare a metodei, iar rezultatele obţinute au fost comparate cu rezultatele obţinute de către alte tehnici de selecţie de caracteristici.

Din această cauză considerăm că tehnica de extragere de caracteristici propusă de noi ajută într-o măsură foarte mare în procesul de antrenare al reţelelor neuronale. Astfel, aceasta reuşeşte cu succes la păstrarea intactă a tuturor caracteristicilor setului de antrenare şi reuşeşte să reducă semnificativ timpul necesar antrenării reţelelor neuronale.

BUPT

7. CONCLUZII

7.1 Concluzii

Teza propune o nouă abordare privind o mai bună utilizare a exemplelor negative în procesul de antrenare al reţelelor neuronale, fapt care a condus la dezvoltarea unei noi tehnici de învăţare: învăţarea pe baza corecţiei erorii cu exemple negative.

Punctul de plecare al prezentei lucrări este situaţia dificilă şi problemele întâlnite de toate aplicaţiile ce folosesc reţele neuronale şi au la dispoziţie un set restrâns de antrenare. Incapabile să obţină un număr mai mare de tipare de antrenare (cele mai multe aplicaţii sunt proiectate pentru condiţii speciale), acestea au fost nevoite să-şi îndrepte atenţie spre zona tiparelor negative.

Folosirea exemplelor negative la antrenarea reţelelor neuronale nu este o practică nouă, însă aceasta nu a fost exploatată suficient fiind oarecum minimizată şi nefiind folosită în mod curent.

Din această cauză, **o primă metodă propusă în această teză este legată de folosirea în mod curent a exemplelor negative în procesul de antrenare**. Pentru aceasta au fost prezentate două exemple de aplicaţii care au folosit un set mixt de antrenare compus din exemple pozitive şi exemple negative. S-a putut astfel observa că **folosirea unui număr de exemple negative într-un procent mai mare de 5% din numărul de exemple pozitive conduce la o îmbunătăţire de până la 25% a procentului de recunoaştere** pentru un set de producţie.

S-a demonstrat în continuare că folosirea unui număr de exemple negative cuprins într-un **interval de [10% - 20%] din numărul de exemple negative duce la maximizarea performanţele** reţelei neuronale.

Rezultatele acestea, împreună cu cele obţinute şi pe alte baze de date au stat la baza **definiri unui model matematic general folosit în antrenarea unei reţele neuronale cu exemple negative**.(Formula 5.5) Acesta a fost testat cu succes şi este primul model de acest fel din literatura de specialitate.

Metoda următoare face un pas mai departe şi propune o modificare a reţelei astfel încât aceasta să fie capabilă să recunoască exemplele negative într-o manieră similară celor pozitive. Se evidenţiază astfel cazul în care o reţea neuronală care nu este 100% sigură cărei clase de la ieşire aparţine tiparul de la intrare, poate să fie 100% sigură că acesta nu aparţine nici unei clase.

Ideea a venit din domeniul învăţării automate, unde utilizarea exemplelor negative face parte integrantă a procesului de definire al unui nou concept, şi presupune o tratare egală a exemplelor pozitive şi negative.

În acest context se propune o modificare a reţelei neuronale, prin introducerea pe ultimul strat a unui neuron suplimentar responsabil cu depistarea tiparelor negative prezente la intrare. Acesta l-am numit neuron **block** şi devine activ numai atunci când la intrare este prezent un tipar negativ. Se elimină în acest mod şi situaţia confuză prezentă până acum, în care inactivitatea totală a neuronilor

BUPT

de pe stratul de ieşire putea să fie tradusă fie prin nerecunoaşterea tiparului de la intrare, fie prin apartenenţa acestuia la o clasă ce nu face parte din ieşirile reţelei.

O altă îmbunătăţire semnificativă **este simplitatea procesului de arbitraj** de la ieşirea reţelei atunci când numai un neuron poate să fie activ. Beneficiind de un neuron responsabil cu tiparele negative de la intrare, putem considera acum neuronul câştigător ca fiind cel cu valoarea cea mai mare. Activarea unui neuron nu mai este condiţionată de depăşirea unui prag minim, ci doar de valoarea acestuia.

Experimentele derulate cu o astfel de reţea modificată au obţinut procente de recunoaştere mai mari cu până la 5% faţă de reţelele neuronale fără un neuron **block**. Un aspect interesant a fost acela **că procentele de recunoaştere pentru reţeaua cu neuron block sunt tot timpul superioare celor obţinute de o reţea clasică**. Acest lucru denotă o maturitate a arhitecturii folosite tradusă printr-o uniformitate în procesul de recunoaştere.

Un alt aspect important relevat de către experimente a fost şi o îmbunătăţire a procentelor de recunoaştere pentru fiecare clasă individuală. Se observă aici **îmbunătăţiri de până la 2% faţă de procentele obţinute de o reţea neuronală antrenată fără tipare negative**. Se poate astfel observa, cum simplitatea metodei de arbitraj conduce la o îmbunătăţire a funcţionalităţii reţelei.

Modul în care această tehnică utilizează exemplele negative în procesul de antrenare şi faptul că s-au adus şi modificări reţelei pentru a putea evidenţia clar aceste exemple, au făcut posibil definirea unei noi tehnici de învăţare şi anume: învăţarea pe baza corecţiei erorii cu exemple negative.

Ultimă metodă de îmbunătăţire a performanţelor reţelelor neuronale prin folosirea exemplelor negative, propusă în această teză, poate să fie văzută ca o extensie a noii tehnici de antrenare şi propune folosirea mai multor neuroni **block** pe stratul de ieşire, fiecare dintre aceştia având o funcţie de activare diferită.

O funcţie de activare este capabilă să recunoască doar o parte din aspectele specifice unei problemei, deoarece ea rămâne *sensibilă* numai la anumite informaţii prezente în procesul de antrenare. Utilizarea mai multor funcţii de activare ar trebui să conferă reţelei capabilităţi de recunoaştere superioare datorate însumării tuturor aspectelor relevate de fiecare funcţie în parte.

Experimentele însă, nu au relevat **nici o îmbunătăţire semnificativă a performanţelor reţelelor neuronale cu mai muţi neuroni block faţă de cele obţinute de reţele cu un singur neuron block.** O posibilă îmbunătăţire a fost sugerată printr-o împărţire mai riguroasă a tiparelor de antrenare negative în mai multe clase şi asocierea unui neuron *block* fiecărei clase. Procesul ar putea avea flexibilitate prin introducerea dinamică a nodurilor odată cu descoperirea încă unei clase în tiparele de antrenare negative.

Multiplele antrenări efectuate în cadrul acestei teze au scos în evidenţă timpul extrem de lung care este necesar în anumite probleme de antrenare (majoritatea experimentelor au durat în jur de 8 zile de antrenare continuă pe un procesor de 2.4 Ghz cu 4 nuclee de procesare). Pentru acest fel de probleme, care conţin un set numeros de antrenare sau un număr mare de intrări a fost propusă o **tehnică de îmbunătăţire a timpului de antrenare a RNA prin folosirea caracteristicilor datelor de intrare.**

Procedura constă într-o preprocesare iniţială a datelor de intrare, pentru a putea selecta cele mai importante caracteristici ale acestora. În urma acestei *filtrări,* o parte din datele de intrare (cele redundante şi cele nefolosite în procesul de

BUPT

antrenare) sunt eliminate, ajungându-se astfel la un set minimal ce păstrează caracteristicile setului original.

Tehnica este derivată din domeniul *data mining* şi este formată din **două componente: o tehnică de evaluare a caracteristicilor şi un algoritm de căutare pentru selectarea celor mai bune caracteristici.**

Rezultatele obţinute cu această tehnică sunt foarte bune, **deoarece timpul de antrenare a fost redus semnificativ(de 11 ori mai mic în unele cazuri)** faţă de o antrenare normală. Valorile pentru noile rate de recunoaştere sunt egale cu cele obţinute anterior şi am prezentat cazuri în care aceste sunt chiar superioare. Aceste rezultate certifică utilizarea cu succes a noii tehnici în problemele de antrenare a unei reţele neuronale de tipul perceptron cu mai multe straturi.

7.2 Contribuţii

Definirea unei noi tehnici de învăţare: învăţarea pe baza corecţiei erorii cu exemple negative. Noua tehnică utilizează o reţea neuronală de tip multilayer perceptron ce are un neuron suplimentar pe ultimul strat. Pe acesta l-am denumit neuron **block** şi este responsabil cu recunoaşterea tiparelor negative prezente la intrare. Această tehnică reuşeşte o îmbunătăţire a performanţelor reţelei cu până la 5% faţă de performanţele obţinute anterior. Se dezvoltă astfel situaţii în care reţeaua neuronală nefiind 100% sigură cărei clase de la ieşire îi aparţine intrarea, reuşeşte să fie 100% sigură că nu aparţine nici unei clase. Neuronul suplimentar rezolvă şi situaţia confuză generată de inactivitatea tuturor ieşirilor (tipar negativ sau reţeaua nu reuşeşte să găsească ieşirea potrivită) şi ajută la simplificarea procesului de arbitrare desfăşurat pe ultimul strat.

Definirea unui model matematic pentru antrenarea cu exemple negative funcţie de procentul de exemple negative prezent în setul de antrenare. Acest model este primul de acest fel şi reuşeşte să determine pentru numărul de exemple negative intervalul [10% - 20%] (din numărul de exemple pozitive) ca fiind optim pentru antrenarea reţelei neuronale. Totodată conduce la mărirea setului de antrenare, lucru extrem de important pentru diferite probleme speciale.

Demonstrarea faptului că un singur neuron block este suficient pe stratul de ieşire atâta timp cât tiparele negative sunt tratate omogen. Prin antrenarea cu mai mulţi neuroni block s-au obţinut performanţe egale cu cele obţinute de o reţea cu un singur neuron block. Se sugerează o împărţire a intrărilor negative în mai multe clase şi asocierea acestora cu câte un neuron block.

Accelerarea procesului de recunoaştere prin antrenarea reţelelor cu caracteristici ale datelor de intrare în cazul în care setul de antrenare este mare sau avem multe intrări. S-au folosit tehnici de data mining pentru a reuşi o extragere a celor mai importante caracteristici pentru datele de intrare. Acestea au fost folosite la antrenarea reţelelor obţinând timpi de 10 ori mai mici în condiţiile în care performanţele s-au menţinut. Trebuie menţionat faptul că au existat situaţii în care, prin această operaţie de „filtrare" performanţele reţelei s-au îmbunătăţit.

Realizarea unui program ajutător pentru antrenarea reţelelor neuronale de tip multilayer perceptron.

BUPT

Definirea unei formule pentru un set de producţie universal, capabil de comparaţii între diverse probleme.

7.3 Continuări posibile

Aplicarea neuronului block şi asupra unor configuraţii de reţele diferite de cele de tip multilayer perceptron. În acest fel se poate observa dacă exemplele negative au un rol major şi în dezvoltarea altor tipuri de reţele.

Studierea comportării unei reţele cu mai mulţi neuroni block, atunci când exemplele negative de la intrare se împart în mai multe clase. Împărţirea acestora într-un mod dinamic şi modificarea arhitecturii pentru fiecare clasă de exemple negative nou introdusă.

Noi metode de integrare a exemplelor negative în procesul de antrenare al RNA. Acest lucru va reuşi o apropiere şi mai mare între RNA şi cele biologice.

BUPT

ANEXA 1 – NIST 19

Baza de date NIST 19 conţine un număr impresionant de caractere. Prezentăm mai jos toate clasele care apar în această bază de date (Tabelul 20) şi care este numărul de apariţii pentru fiecare. Se poate observa că numărul de apariţii este diferit de la o clasă la alta şi poate lua o plajă largă de valori (Tabelul 21). Acest fapt este datorat operaţiei de segmentare a textului, în urma căreia mai multe clase au fost eliminate din baza de date.

Câmpurile care sunt în dreptul coloanei „Constituţie” şi sunt goale fie nu au suportat operaţia de segmentare, fie nu apar în textul constituţional din formularul ce trebuia completat.(Figura 35)

Literă mare	Literă mică	Literă mare	Constituţie	
			Literă mare	Literă mică
0	a	A	A	a
1	b	B	B	b
2	c	C	C	
3	d	D	D	d
4	e	E	E	e
5	f	F	F	
6	g	G		g
7	h	H	H	h
8	i	I	I	
9	j	J	J	
	k	K		
	l	L	L	l
	m	M	M	
	n	N	N	n
	o	O	O	
	p	P	P	
	q	Q		q
	r	R	R	r
	s	S	S	
	t	T	T	t
	u	U	U	
	v	V	V	
	w	W	W	
	x	X		
	y	Y	Y	
	z	Z		

Tabelul 20. Clasele prezente în NIST 19

BUPT

Literă mare	Literă mică	Literă mare	Constituţie	
			Literă mare	Literă mică
40363	3210	3033	4436	8467
44704	3136	3026	1500	2876
40072	3286	3344	8489	
41112	3111	3020	2321	8749
39154	3084	2892	2893	25639
36606	2961	3053	7569	
39937	2966	2964		1310
41893	3253	2925	748	6964
39579	3152	4490	9504	
39533	2213	2958	1430	
	2957	2850		
	4454	3375	2511	13399
	3109	3077	7410	
	3150	3128	6460	10166
	3215	3176	25963	
	2816	3127	6617	
	2648	3018		851
	3113	3061	2821	13312
	3136	3129	21143	
	3058	2981	8415	18169
	3312	3143	11461	
	3378	3237	2196	
	3164	3122	2379	
	3292	3203		
	2746	2966	2575	
	3176	3165		

Tabelul 21. Numărul de instanţe pentru fiecare clasă din NIST 19

BUPT

ANEXA 2 – FORMATUL DATELOR MARTHA'S VINEYARD COASTAL OBSERVATORY [ENG]

Field 1	Year	Year of record
Field 2	Yday	1.5 is noon on January 1
Fields 3-7	MO DA HH MM SS	start of 15 or 20 minute burst GMT
Field 8	wave_height[m]	4*sqrt(sum(Snn(f)*dF)),where f < 0.5 Hz and Snn is the spectral intensity from the highest bin below 0.85*depth (from pressure),then taken to the surface using linear wave theory
Field 9	wave_period[seconds]	1/fbar, where fbar = sum(Snn*f)/sum(Snn)
Field 10	wave_dir_[degrees]	compass wave direction from which waves in peak are coming (PUV), as observed at 4.7 meters above bottom
Field 11	temp_adcp_mean[degC]	temperature at 2.11 meters above bottom
Field 12	vel1_adcp_mean[cm/s]	near-bottom mean velocity (3.7 meters above bottom)
Field 13	vel1_adcp_dir[cm/s]	near-bottom velocity direction
Field 14	vel2_adcp_mean[cm/s]	near-surface mean velocity (10.7 meters above bottom)
Field 15	vel2_adcp_dir[cm/s]	near-surface velocity direction
Field 16	tide_paros[m]	corrected for atm press using the Vaisala pressure. A nominal height was subtracted, as follows: Paros pressure - 11.7 meters (to 6/02) Paros pressure - 11.8 (7/02 to present)
Field 17	water_temp[degC]	mean water temperature (CTD) at 12.7 meters depth
Field 18	salinity[ppt]	salinity (CTD) at 12.7 meters depth
Field 19	wave_height_swell[m]	swell height (swell is observed from 0.05 to 0.15 Hz)
Field 20	wave_height_windwave [m]	wind wave height (wind wave is observed from 0.15 to 0.25 Hz)
Field 21	wave_period_swell [secs]	1/fbar in swell region
Field 22	wave_period_wind [secs]	1/fbar in wind wave region
Field 23	wave_directn_swell [degs]	PUV wave direction in swell region
Field 24	wave_directn_windwave [degs]	PUV wave direction in wind wave region

Tabelul 22. Formatul datelor de la Observatorul Martha's Vineyard

BUPT

ANEXA 3 – BAZELE DE DATE CUPRINSE ÎN SETUL DE ANTRENARE DE LA UNIVERSITATEA CARNEGIE MELLON

Este unul din cele mai complete seturi de antrenare pentru reţele neuronale. Are în componenţa sa următoarele baze de date (vezi Tabelul 23)

Baza de date	Numărul de elemente	Complexitatea (1..5 în care 5 cea mai mare)
Segment	160000	5
Parity	50000	2
Protein	20000	4
Sonar	2000	4
Hypo	100000	4
Vowel	3000	5
Xor	4	1
Diabetes 1	768	4
Diabetes 2	768	4
Diabetes 3	768	4
Mushroom 1	9124	3
Mushroom 2	9124	3
Mushroom 3	9124	3
Thyroid	7200	4
Hepatitis	7200	4
Sick	7200	4
Credita	8000	4
Creditg	8000	4
Iris	900	3

Tabelul 23. Bazele de date şi numărul de instanţe din setul de antrenare de la Universitatea Carnegie Mellon

BUPT

ANEXA 4 – CARACTERISTICILE SELECTATE DE PROGRAMUL WEKA PENTRU LITERELE A,E,N ŞI R

În urma prelucrărilor cu programul Weka, din cele 1024 de intrări asociate pentru fiecare literă au fost selectate numai 100 de intrări considerate a fi reprezentative pentru literele „a”, „e”, „n” şi „r”. Fiecare număr reprezintă pixelul ales. Pixelul 0 este considerat în stânga sus şi direcţia de înaintare este spre dreapta.

Pixeli selectaţi	Pixeli selectaţi	Pixeli selectaţi
80	84	120
121	137	175
237	238	301
396	408	426
434	436	440
459	460	467
471	473	489
491	492	498
499	500	504
520	523	530
531	536	553
554	555	559
562	563	564
569	584	585
595	597	599
616	617	623
626	633	649
650	658	660
662	664	680
690	695	696
712	714	721
725	728	754
755	777	789
790	816	820
847	852	856
873	878	881
883	889	904
910	911	912
913	914	915
916	917	918
919	937	940
944	945	946
950	951	952
985		

Tabelul 24. Caracteristicile selectate de programul Weka pentru literele „a”, „e”, „n” şi „r”, reprezentate iniţial printr-o matrice de 32x32pixeli

BUPT

BIBLIOGRAFIE

[1] Haykin S.: „Neural Networks. A comprehensive Foundation”, Macmillan College Publishing, New York 1994

[2] Pora, E.: „Omul şi natura”, Editura Dacia, Cluj-Napoca, pag. 20-25, 1975

[3] Feng-Hsiung, H.: „Behind Deep Blue: Building the Computer that Defeated the World Chess Champion”, Princeton University Press, 2002

[4] McCulloch, W.S., Pitts, W.: „A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics, Vol. 5, pag. 115 – 133, 1943

[5] Hodgkin, A.L., Huxley, A.F.: „A quantitative description of membrane current and its application to conduction and excitation in nerve”, J Physiology, Vol. 117:500-44, 1952

[6] Mitchell, T.M.: „Machine Learning”, McGraw-Hill International Edit, Secţiunea 4.1.1, pag. 82, 1997

[7] Ramon y Cajal, S.: „Histologie du systeme nerveux de l’homme et des vertebrates”, Oxford University Press, New York, 1995

[8] Hebb, D.O.: „The organization of behavior; a neuropsychological theory”, Wiley-Interscience, New York 1949

[9] Rosenblatt, F.: „The perceptron: a probabilistic model for information storage and organization in the brain”, Psychological Review, Vol. 65, Nr. 6, pag. 386-408, 1958

[10] Kohonen, T.: „Correlation matrix memories”, IEEE Transaction on Computers, C-21, pag. 353-359, 1972

[11] Clarke, A.C.: „2001 A Space Odyssey”, New American Library, United States, 1968

[12] Asimov, I.: „The Bicentennial Man”, Ballantine Books, United States, 1976

[13] Hopfield, J.: „Neural Network and physical systems with emergent collective computational abilities”, Proceedings of the National Academy of Science of the USA, Vol. 9, 1982

[14] Treleaven, P.C., Lime, I.G.: „Japan’s Fifth Generation Computer Systems”, IEEE Computer, Vol. 15, Nr. 8, pag. 79-99, August, 1982

[15] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: „Learning internal representations by error propagation”, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, pag. 318-362, 1986

[16] Minsky, M.L., Papert, S.: „Perceptrons: An introduction to Computational Geometry”, The MIT Press, 1969

[17] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: „Learning representations by backpropagating errors.”, Nature, 1986

[18] Hornik, K., Stinchcombe, M., White, H.: „Multilayer feedforward networks are universal approximations”, Neural Networks, pag. 359-366, 1989

BUPT

[19] Funahashi, K.-I.: „On the approximate realization of continuous mappings by neural networks", Neural Networks, pag. 192-193, 1989
[20] Cybenko, G.: „Approximation by superposition of a sigmoidal function", Mathematics of Control, Signals and Systems, pag. 303-314, 1989
[21] Hartman, E.J., Keeler, J.D., Kowalsky, J.M.: „Layered neural networks with Gaussian hidden units as universal approximations", Neural Computation, pag. 210-215, 1990
[22] Orr, Mark J. L.: „Introduction to Radial Basis Function Network", Centre for Cognitive Science, Edinburg, 1996
[23] Kosko, B.: „Hidden patterns in combined and adaptive knowledge networks", International Journal of Approximate Reasoning, vol. 2, pag. 377-393, 1988
[24] Tiponut, V., Căleanu, C.: „Reţele neuronale. Arhitecturi si algoritmi.", Ed. Politehnica Timişoara, Colecţia Prelucrarea semnalelor, 2001
[25] Widrow, B., Hoff, M.E.: „Adaptive switching circuits", IRE WESCON Convention Record, Partea 4, pag. 96-104, DUNNO, 1960
[26] Rabunal, J. R.: „Artificial Neural Networks in Real-life Applications", Idea Group Pub, 2005
[27] Cernăzanu, G.C.: „Tehnici de antrenare a reţelelor neuronale", referat I pentru doctorat, 2005
[28] Behnke, Sven: „Hierarchical Neural Networks for Image Interpretation", Springer, pag. 45, 2003
[29] Yuan, J. L.; Fine, T. L.: „Neural-network design for small training sets of high dimension", IEEE transactions on neural networks, vol. 9, nr. 2, pag. 266-280, 1998
[30] Koike, K., Matsuda, S., Suzuki, T., Ohmi, M.: „Neural Network-Based Estimation of Principal Metal Contents in the Hokuroku District, Northern Japan, for Exploring Kuroko-Type Deposits", Natural Resources Research, Netherlands, Springer, vol. 11, nr. 2/Iunie, pag. 135-156, 2002
[31] Qun, Z., Principe, J. C.: „Improve ATR performance by incorporating virtual negative examples", Proceedings of the International Joint Conference on Neural Networks, pag. 3198-3203, 1999
[32] Tesauro, G., Kephart, O., Sorkin, G. B.: „Neural Networks for Computer Virus Recognition", IEEE Expert, vol. 11, nr. 4, pag. 5-6, August, 1996
[33] Christel, M., Stevens, S., Wactlar, H.: „Informedia Digital Video Library", Proceedings of the ACM Multimedia '94 Conference, pag. 480-481, 1994
[34] Nyogi, P., Girosi, F., Poggio, T.: „Incorporating prior information in machine learning by creating virtual examples", Proceedings of the IEEE 86(11), pag. 2196-2209, 1998
[35] Abu-Mostafa, Y.S.: „Hints", Neural Computation, vol. 7, pag. 639-671, 1995
[36] Qun, Z., Principe, J.C.: „Incorporating virtual negative examples to improve SAR ATR", Proceedings of the SPIE – The International Society for Optical Engineering, vol. 4053, pag. 354-360, 2000
[37] Nilsson, N.: „Learning Machines: foundations of trainable pattern-classifying systems", McGraw-Hill, Inc., 1965

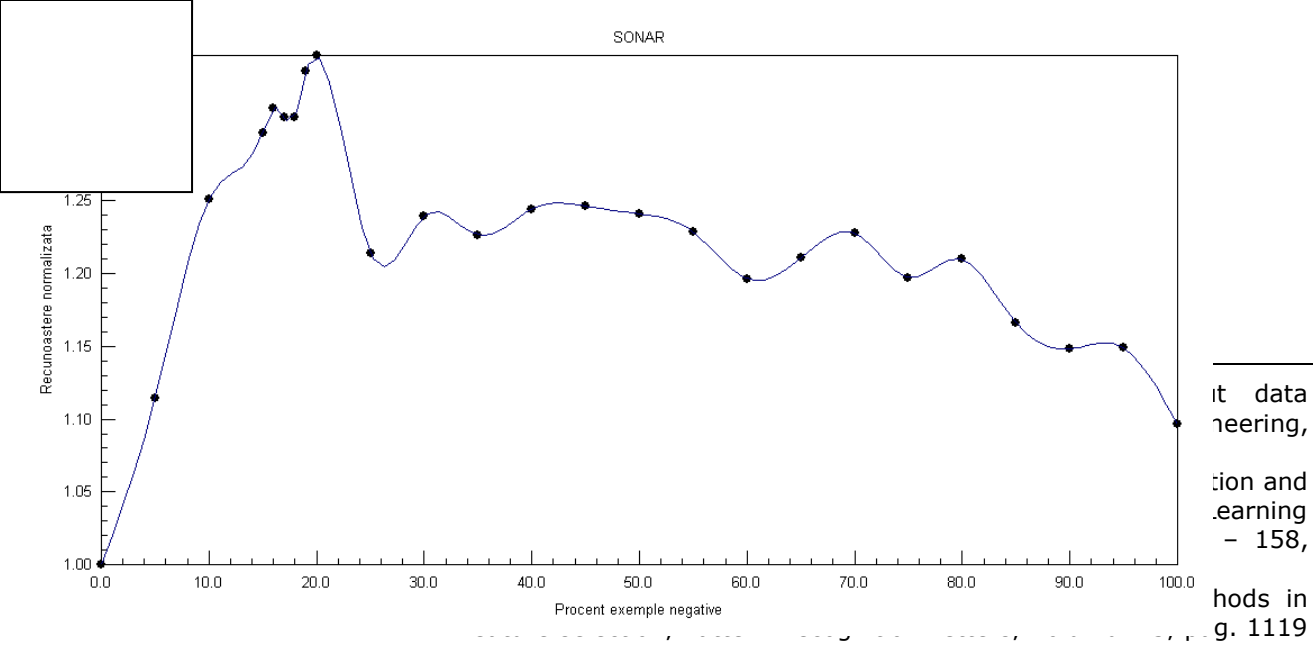
BUPT

[38] Quinlan, J.R.: „Induction of decision tree.", Machine Learning, vol. 1, 1986
[39] Shannon, C.: „A mathematical theory of communication.", Bell System Technical Journal, vol. 27, pag. 379-423, 1948
[40] Winston, P. H.: „Learning structural descriptions from examples", In P.H. Winston editor, 1975
[41] Winston, P. H.: „The psychology of Computer Vision", McGraw-Hill, New York, 1975
[42] Winston, P. H.: „Artificial Intelligence, 3rd edition Reading", Addison Wesley, MA, 1992
[43] Luger, G.: „Artificial Intelligence :Structures and Strategies for Complex Problem Solving", (Fifth Edition) Addison Wesley, 2005
[44] Negnevitsky, M.: „Artificial Intelligence: A Guide to Intelligent Systems (2nd Edition)", Addison Wesley, England, 2005
[45] Bratko, I.: „PROLOG – Programming for Artificial Intelligence", (2nd Edition) Addison Wesley, England, 1993
[46] Mitchell, T.M.: „An analysis of generalization as a search problem", Proceedings IJCAI, vol. 6, 1979
[47] Mitchell, T.M.: „Generalization as search.", Artificial Intelligence, vol. 18(2), pag. 203-226, 1982
[48] Mitchell, T.M.: „Version spaces – an approach to concept learning", Report No. STAN-CS-78-711,Computer Science Dept., Stanford University, 1978
[49] Alecsander, I.; Morton, H.: „An Introduction to Neural Computing", Chapman & All, London, 1990
[50] Kim, H. Y.; K.T., Lim; Nam, Y.S.: „Handwritten numeral string recognition using neural network classifier trained with negative data", Frontiers in Handwriting Recognition, pag. 395 – 400, 2002
[51] Raudys, S; Somorjai, R; Baumgartner, R: „Reducing the overconfidence of base classifiers when combining their decisions", 4th International Workshop on Multiple Clasifier Systems (MCS 2003), vol. 2709, pag. 65-73, 2003
[52] Hosom, J.P., Villiers, J., s.a.l: „Training Hidden Markov Model/Artificial Neural Network (HMM/ANN) Hybrids for Automatic Speech Recognition (ASR)", Center for Spoken Language Understanding (CSLU), 2006
[53] Debevec, P.: „A Neural Network for Facial Feature Location", CS283 Course Project, UC Berkeley, 2001
[54] Tikhonov, A.; Arsenin, V.: „Solutions of ill-posed problems", W.H. Winston, 1977
[55] Girosi, F. & Poggio; B., T. & Caprile: „Extensions of a theory of networks for approximation and learning:outliers and negative examples.", Advances in Neural Information Processing Systems 3, R.P. Lippmann, J.E.Moody and D. S. Touretzky, pag. 750-756, San Mateo, CA, 1991
[56] Anderson, Dave; McNeil, George: „Artificial Neural Networks Technology", Data & Analysis Center for Software, 1992
[57] Lacassie, Juan Pablo; Ruiz del Solar, Javier; Roser, Barry; Hervé, Francisco: „Visualization of Volcanic Rock Geochemical Data and Classification with Artificial Neural Networks", Springer, Netherlands, vol. 38, nr. 6 / August, pag. 697-710, 2006

BUPT

[58] Alippi, C.: „Weight update in back-propagation neural networks: the role ofactivation functions", IEEE International Joint Conference on Neural Networks, vol. 1, pag. 560 – 565, 1991
[59] Ward Systems Group, Inc.: „NeuroShell 2", http://www.wardsystems.com/neuroshell2.asp, 2000
[60] National Institute of Standards and Technology: „NIST Handprinted Forms and Characters Database", http://www.nist.gov/srd/nistsd19.htm, 2007
[61] Martha's Vineyard Coastal Observatory: „Ocean Data", http://www.whoi.edu/mvco/data/oceandata.html
[62] Carnegie Mellon University: „Neural Networks Benchmark Collection", http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/bench/0.html
[63] Lopes, N., Ribeiro, B.: „An efficient gradient-based learning algorithm applied to neural networks with selective actuation neurons", Neural, Parallel & Scientific Computations, Vol. 11, Issue 3, pag. 253-272, 2003
[64] Holt, J.L., Baker, T.E.: „Back propagation simulations using limited precision calculations", Neural Networks, Vol. 2, Issue 2, pag. 121-126, 1991
[65] Ghaffari, M., Hall, E.L.: „Experimental approach for the evaluation of neural network classifier algorithms", Congres Intelligent robots and Computer Vision XXI: algorithms, techniques and active vision, vol. 5267, pag. 250-256, 2003
[66] Cernazanu, C., Holban, S.: „Improving neural network performances – training with negative examples", International Conference on Telecommunications and Networking/International Conference on Industrial Electronics, Technology and Automation, University of Bridgeport, Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics, pag. 49-53, 2008
[67] Babii, S., Cretu, V., Petriu E.M.: „Performance Evaluation of Two Distributed BackPropagation Implementations", Neural Networks, IJCNN 2007, pp. 1578-1583, 2007
[68] Zhongwen, L., Zhongwen, L., Hongzhi, L., Xincai, W.: „Artificial neural network computation on graphic process unit", Neural Networks, IJCNN 2005, pp. 622-626, 2005
[69] Siddique, M.N.H., Tokhi, M.O.: „Training neural networks: backpropagation vs. genetic algorithms", Neural Networks, 2001, IJCNN, pp. 2673-2678, 2001
[70] Nguyen, D., Widrow, B.: „Improving the learning speed of 2-layer neural networks by choosing initial values of adaptive weights", Neural Networks, IJCNN, 1990, pp. 21-26, Volume. 3, 1990
[71] Gorea, D.: Dynamically Integrating Knowledge in Applications. An Online Scoring Engine Architecture, Advances in Electrica land Computer Engineering, Suceava, Romania, Volume 8, pag. 44-49, 2008
[72] Langley, P.: „Selection of relevant features in machine learning", Proceedings of the AAAI Fall Symposium on Relevance, AAAI Press, 1994

BUPT



– 1125, Noiembrie 1994

[76] Kim, Y., Street, W.N., Menczer, F., Roussell, G.J.: „ Feature selection in data mining", L. Wang Editor, Data Mining: Opportunities and Challenges, Idea Group Publishing, pag. 80-105, 2003

[77] Gigli, G., Bosse, I., Lampropoulos, G.A.: „A optimized architecture for classification combining data fusion and data mining", Information Fusion, Volumul 8, pag. 366 – 378, 2007

[78] Guyon, I., Elisseeff, A.: „An introduction to variable and feature selection", Journal of Machine Learning Research, pag. 1157 – 1182, Volumul 3, 2003

[79] Hall, M.: „Correlation – based Feature Selection for Machine Learning", Ph. D. diss., Hamilton, NZ, Waikato University, Department of Computer Science, 1998

[80] Boyan, J., Moore, A.: „Learning evolution functions to improve optimization by local search", Journal of Machine Learning Research, Volume 1, pag. 77 – 102, 2000

[81] http://www.cs.waikato.ac.nz/ml/weka: „Weka3, Data mining Software in java", The University of Waikato, 2008

[82] http://weka.sourceforge.net/wiki/index.php: „Performing attribute selection", 2008

[83] Witten, I.H., Frank, E.: „Data mining: Practical Machine Tools and Techniques (Second Edition)", Morgan Kaufmann, 2005

[84] Cernazanu, C., Holban, S.: „Determining the optimal percent of negative examples used in training the multilayer perceptron neural networks", International Conference on Neural Networks, pag. 114-119, Prague, 2009

[85] Teodorescu, R., Cernazanu, C., Cretu, V.: „The use of the medical ontology for a semantic-based fusion system in Biomedical Informatics Application to Alzheimer Disease", IEEE 4th International Conference on Intelligent Computer Communication and Processing, pag. 254-268, Cluj Napoca, 2008

[86] Cernazanu, C., Holban, S.: „Training Neural Networks Using Input Data Characteristics", Development and Application Systems – Abstract Book, pag. 309 – 312, Romania, 2008

[87] Opitz, D.: „Feature Selection for Ensembles", Proceedings of 16th National Conference on Artificial Intelligence, pag. 379 – 384, Orlando, Florida, 1999

[88] DataFit 0.0: „Oakdale Engineering Data Fit", Oakdale Engineering, http://www.oakdaleengr.com

[89] Origin 6.0: „Introduction to Origin Workspace", TechSmith Camtasia Studio, http://www.originlab.com