Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

ISOLATED ROMANIAN WORDS RECOGNITION USING NEURAL NETWORKS

Marian P. Briciu[1]

This paper describes an experiment of isolated word recognition in Romanian using the backpropagation algorithm as the main training tool. The paper is organized as follow: in the first section some theoretical aspects of neural networks are reviewed; in the second section is presented the recognition algorithm; the third section presents experimental results based on a small size vocabulary application for isolated Romanian words recognition.

I. INTRODUCTION

One of the most important aspects of machine learning models is how well the model generalizes to the phonemes of speech, or to letters, in that they do not themselves convey meaning but indicate different intensities that are interpreted as meaningful units by *the language* unseen data. Multi-layer perceptron (MLP) neural networks have been very successful in many problems, often providing improved generalization over competing machine learning methods. Neural networks are particularly interesting for speech recognition, which requires massive constraint satisfaction, i.e., the parallel evaluation of many clues and facts and their interpretation in the light of numerous interrelated constraints. The computational flexibility of the human brain comes from its large number of neurons in a mesh of axons and dendrites. The communication between neurons is via the synapse and afferent fibers. There are many billions of neural connections in the human brain. At a simple level it can be considered that nerve impulses are comparable *of the brain.* Neural networks attempt to achieve real-time response and human like performance using many simple processing elements operating in parallel as in biological nervous systems. Models of neural networks use a particular topology for the interactions and interrelations of the connections of the *neural units.*

One of the technical developments sparking the recent resurgence of interest in neural networks has been the popularization of *multi-layer perceptrons* (MLP). A multi-layer perceptron. In contrast to a single-layer perceptron, it has a hidden layer. The hidden layers can be viewed as feature extractors. Each layer has the same computation models as the single-layer perceptron; i.e., the value of each node is computed as a linear weighted sum of the input nodes and passed to a sigmoid type of threshold function.

The back propagation algorithm is a generalization of the minimum mean squared error (MMSE) algorithm. It uses a gradient search to minimize the difference between the desired outputs and the actual net outputs, where the optimized criterion is directly related to pattern classification. With initial parameters for the weights, the training procedure is then repeated to update the weights until the cost function is reduced to an acceptable value or remains unchanged.

In real-world application, these weights are estimated from a large number of training observations in a manner similar to hidden Markov modeling. The weight updates in the Step 3 are accumulated over all the training data. The actual gradient is then estimated for the complete set of training data before the starting of the next iteration. Note that the estimation criterion for neural networks is directly related to classification rather than the maximum likelihood.

II. THEORETICAL ELEMENTS AND RECOGNITION ALGORITHM

During the recognition of isolated words, the neural network transforms input feature vectors into those that correspond to speech, and passes them to speech recognizers. A multi layer perceptron is used to establish the on linear mapping function of speech feature vectors between the testing and the training environments.

Neural network is trained to minimize the accumulated mean squared error, which is the sum of the squared difference between network output and corresponding target.

The main idea of automatic speech recognition is that the recognition system must be "trained" with

[1] Military Technical Academy, Bucharest,
81-83 Av. George Cosbuc, tel. 0040 72 21 31 016, e-mail: briciumarian@yahoo.com

parameters (spectral features) of the words that will be recognized.

There are four basic steps to performing recognition, shown in fig. 1. Each step will be explained briefly here and in more detail later in this document.

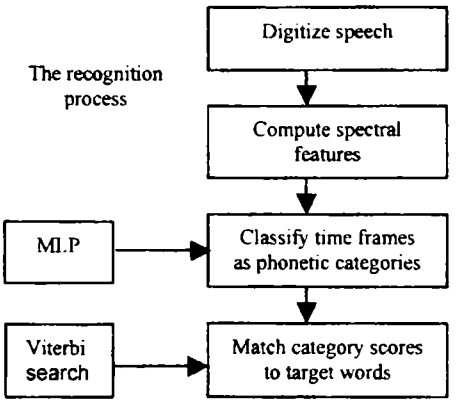

Fig 1. Diagram of the recognition process

First, we digitize the speech that we want to recognize the sampling rate is 16000 samples per second. Second, we compute features that represent the spectral-domain content of the speech (regions of strong energy at particular frequencies). These features are computed every 10 msec, with one 10-msec section called a *frame*. Third, a neural network (also called an ANN, multi-layer perceptron, or MLP) is used to classify a set of these features into phonetic-based categories at each frame. Fourth, a Viterbi search is used to match the neural-network output scores to the target words (the words that are assumed to be in the input speech), in order to determine the word that was most likely uttered.

My database is composed of numbers spoked by a number of speakers and stored with a soundboard.

In order to build the recognition system I used the NICO (Neural Inference COmputation) toolkit. The network topology is very flexible. Units are organized in *groups* and the group is a hierarchical structure, so groups can have sub-groups or other objects as members.

The training tools, feature extraction and target generation tools put some constraints on how I can organize my database. 1) All files corresponding to the same utterance must have the same base name. 2) All files of the same type, e.g., phonetic label files, or feature vector files, must have the same file extension and live in the same directory. If my database is not organized this way I can create some new directories and make symbolic links to the physical locations of the files.

The tool Barkfib implements a standard feature extraction, based on a Bark or Mel-scaled filter-bank computed from the FTT spectrum.

The words target for each 10 ms frame of the feature files can be extracted from the phonetic transcription files of the database. The tool Lab2Targ does this by checking the length of the parameter files and using the time-marks of the transcription files to compute the targets for each phoneme output unit at each frame.

In order to train my network I use the most important training tool of the toolkit, BackPropagation algorithm.

Now, let us look back to the digitized waveform which is converted into a spectral-domain representation; one of two sets of features (Bark or Mel given by Barkfib command) may be used, depending on the recognizer. For the current recognizer, I use twelve mel-frequency cepstral coefficients (MFCC) and one energy feature (for a total of 13 features per frame).

To provide some information about the acoustic content I take a *context window* of features. This means simply taking the frame of interest as well as frames that are -60, -30, 30, and 60 msec away from the frame of interest. This is done to take into consideration the dynamic nature of speech: the identity of a phoneme will often depend not only on the spectral features at one point in time, but it will also depend on how the features change over time.

I send the features in a context window to a neural network for classification (13 features per frame at 10 frames = 130 features). The output of the neural network is a classification of each input frame, measured in terms of the probabilities of phoneme-based categories. By sending context windows for all frames of speech to the neural network, I can build a matrix of the probabilities of phoneme-based categories over time. In my example of neural-network output, the words to be recognized are *unu, doi, trei, patru, cinci, sase, sapte, opt, noua, zero*.

The target-word pronunciations are then expanded into strings of phonetic-based categories, and a Viterbi search is used to find the best path through the matrix of probabilities for each legal string. The output of recognition is the word string that corresponds to this best path.

Once we have a matrix of how the phonetic probabilities change with time, we want to search for the best word. Before doing that, we need to compute the set of legal strings of phonetic categories. This set is dependent on the words we want to recognize and the possible order of words, so we combine pronunciation models for each of our words. Here I have a simple search path that can recognize only isolated spoken words, which have to be preceded and followed by silence. In searching, when we look at a new frame, we transition to a new state if the probability of the new category is greater than the probability of the current category. At the end of the search, we have a score for the most likely category sequence and the path through the categories that was used to generate the best score. We can take this path and easily determine the corresponding word (or word sequence). This word has the best fit to the input data, and it is therefore the word that was most likely uttered.

III. EXPERIMENTAL RESULTS

In order to exemplify the recognition algorithm explained above I made an application of Romanian numbers recognition. I built and trained, with different parameters, a network for words recognition on the acoustic frame level for my database. In all cases I kept the same number of MFCC (12).

First, I have trained the network with a single hidden level, but with different number of neurons in this level, and, in the same time, I have considered the number of the iterations made by the back-propagation algorithm equal to 50. The recognition error rate in this case is shown in table 1. The training takes 45 minutes on an AMD K7 computer at 850 MHz.

Table 1. Recognition error rate resulted after training with one hidden level

Word	Recognition error rate	
	100 neurons in hidden level	150 neurons in hidden level
Unu	7.372e-001	7.142e-001
Doi	6.324e-001	6.113e-001
Trei	7.476e-001	7.282e-001
Patru	8.658e-001	8.398e-001
Cinci	7.476e-001	7.282e-001
Sase	14.567e-001	14.216e-001
Sapte	11.543e-001	11.284e-001
Opt	7.658e-001	7.398e-001
Nouă	8.747e-001	8.284e-001
Zero	7.878e-001	7.264e-001

Second, I have trained the network with two hidden level, with the same number of neurons in both hidden levels but with different number of neurons in first and second training, and I have considered the number of the iterations made by the back-propagation algorithm equal to 50. The recognition error rate in this case is shown in table 2. The training takes 2 hours and 30 minutes on an AMD K7 computer at 850 MHz.

Table 2. Recognition error rate resulted after training with two hidden levels

Word	Recognition error rate	
	100 neurons in first and second hidden levels	150 neurons in first and second hidden levels
Unu	5.359e-001	5.148e-001
Doi	4.560e-001	4.395e-001
Trei	5.589e-001	5.256e-001
Patru	6.688e-001	6.406e-001
Cinci	5.476e-001	5.302e-001
Sase	10.597e-001	10.2406e-001
Sapte	8.683e-001	8.364e-001
Opt	5. 748e-001	5. 288e-001
Nouă	6. 657e-001	6. 394e-001
Zero	5.749e-001	5.296e-001

Third, I have trained the network with three hidden levels, with 100 neurons in all three hidden levels, and, in the same time, and I have considered the number of the iterations made by the back-propagation algorithm equal to 30. The recognition error rate in this case is shown in table 3. This training takes 7 hours and 40 minutes on an AMD K7 computer at 850 MHz.

Table 3. Recognition error rate resulted after training with three hidden levels

Word	Recognition error rate
Unu	4. 683e-001
Doi	3. 750e-001
Trei	4.589e-001
Patru	5. 598e-001
Cinci	4.466e-001
Sase	9.549e-001
Sapte	7.674e-001
Opt	4. 778e-001
Nouă	5. 659e-001
Zero	4.783e-001

IV. CONCLUSIONS

If we compare the results obtained in all cases we can say that the recognition error rate it is improving if we increase the number of hidden levels and the numbers of the neurons in this levels, but we must find the optimum topology of the network. It is better to begin with a low number of neurons and increase it step by step. The same applies to the inclusion of parameters. A better recognition error rate can be obtained if we increase the number of hidden levels, but at a specific point the recognition rate is good enough to stop. The training duration is not a constriction in the recognition process, because it is made just ones with a specific vocabulary and it is not repeated each time we do the recognition.

REFERENCES

[1] L. R. Rabiner and R. W. Schafer "*Digital processing of speech signals.*", Prentice Hall, 1978

[2] A.T. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K.J. Lang, Phoneme recognition using time-delay neural networks, *IEEE Trans. Acoustics, Speech & Signal Processing*, Vol. 37, No. 3, March 1989

[3] D. Dumitrescu and C. Hariton "*Rețele neuronale. Teorie și aplicații.*", Teora, 1996

[4] I. Gavat *et al*, "*Elemente de sinteză și recunoaștereа vorbirii*",Printech, 2000

[5] X.D. Huang, A. Acero and H.-W. Hon, "*Spoken Language Processing. A guide to Theory, Algorithm and System Development*", Prentice Hall, 2001

[6] The Departmemt for Speech, Music, and Hearing at KTH, Stockholm, Sweden: NICO toolkit