14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

NEURAL NETWORKS WITH RANDOM LETTER CODES FOR TEXT-TO-PHONEME MAPPING AND SMALL TRAINING DICTIONARY

Enikő Beatrice Bilcu, and Jaakko Astola

Institute of Signal Processing, Tampere University of Technology
Korkeakoulunkatu 1, Tampere, Finland
emails: bilcub@cs.tut.fi, jta@cs.tut.fi

ABSTRACT

In this paper we address the problem of text-to-phoneme (TTP) mapping implemented by neural networks. One important disadvantage of the neural networks is the convergence interval which can be in some situations very large. Even when the neural networks are trained in off line mode a shorter convergence interval would be of interest due to various reasons. In the TTP mapping, decreasing the number of necessary iterations is equivalent to relaxing the requirements for the dictionary size. In this paper, we show that proper letter encoding can increase the convergence speed of the multilayer perceptron neural network for the task of TTP mapping. Experimental results that compare the performance of several techniques that speed-up the convergence of the multilayer perceptron, in the context of TTP mapping are also presented.

1. INTRODUCTION

One important step in several speech processing applications is the text-to-phoneme mapping. In speech synthesis, TTP mapping is responsible for translation of the written text to the corresponding phonetic transcriptions from which the synthetic speech is then generated. In speech recognition, a dictionary of phonetic transcriptions must be build by mapping the words to their phonetic transcriptions and this is done by TTP mapping.

Several solutions, to the problem of TTP mapping, have been proposed in the open literature. Some of the solutions are based on decision trees [4] while others use neural networks to perform the TTP mapping [3], [5]-[11]. The advantage of the decision trees is that they provide a more accurate mapping for known words (words that were used in the training process). However, for new words that were not used in the training process, neural networks provide superior phoneme accuracy compared to the decision trees [4].

The well known multilayer perceptron (MLP) neural network have been applied with success to the problem of TTP mapping [3], [5]-[11]. When MLP is trained with the back-propagation with the momentum algorithm the adaptive system is simple to implement and has a low computational complexity. The setup of the training process is simple due to the fact that only one parameter, that is constant during the training , controls the stability and the modelling performance of the NN. However, the MLP trained with the back-propagation algorithm possess a slow convergence that represents a major drawback (the importance of a fast training system in the context of TTP mapping will be emphasized later in this paper). In order to deal with this problem several approaches have been introduced.

It has been shown in several publications that the convergence speed of the back-propagation algorithm can be increased if a time-varying learning rate is implemented [7]-[11]. Usually the learning rate is adapted using the output error, and the mechanism of learning rate adaptation increases the computational complexity of the system. Another way to increase the convergence speed of the neural networks is to use orthogonal inputs that can increase the modelling accuracy as well [3]. In this regard, a self-organizing codebook was proposed in [5] with the aim to increase the phoneme accuracy in the TTP mapping task. In this approach, the input letters are encoded using a separate small neural network. Another way to increase the performance of the adaptive system is to apply some transformation at the input of the NN. This was done in [8] by using the discrete cosine transform (DCT) to orthogonalize the NN's inputs. The phoneme accuracy obtained with the MLP neural network for different orthogonal and non-orthogonal letter codes was also studied in [6]. In that paper, it was found that the random real valued codes can give improved recognition rates compared with other orthogonal and non-orthogonal letter codes for small neural network complexities (small number of synaptic weights). However, in [6], the random real valued letter codes were shorter than the orthogonal binary codes. Moreover, the main focus in [6] was the phoneme accuracy and not the size of the training dictionary.

In this paper, we extend the work in [6] and we study the convergence speed of the MLP neural network that uses random codes for the input letters. We show that when the input letters are encoded using vectors with equals lengths, real random codes can provide faster convergence compared to the orthogonal binary codes that are usually implemented. We compare the results obtained using binary and random valued codes with other two approaches that use a time-varying learning rate and transform domain training. In Section 2 we describe with the description of the text-to-phoneme (TTP) mapping problem, and we give implementation details and discuss the importance of having a fast converging TTP mapping system. In Section 3, we describe the neural networks that are used in this paper and the encoding methods implemented for the input letters. Section 4 shows comparative experimental results and Section 5 concludes the paper.

2. TEXT-TO-PHONEME MAPPING

In this section, we give a brief overview of the TTP mapping outlining the necessity to have a fast converging system. In all systems implemented to perform TTP mapping the input consists of a set of letters and the output of the system is represented by the corresponding phonemes. For imple-

Letters	Corresponding binary vectors
a	1 0 0 0 ... 0 0 0
b	0 1 0 0 ... 0 0 0
c	0 0 1 0 ... 0 0 0
⋮	⋮
z	0 0 0 0 ... 0 1 0
\0	0 0 0 0 ... 0 0 1

Table 1: Binary orthogonal letter codes. Each vector has 27 elements of which only one is set to unity.

Letters	Corresponding random real valued vectors
a	-0.43 -1.59 0.59 0.79 ... -0.94 1.47 0.03
b	-1.66 -1.44 -0.64 0.94 ... -0.37 1.13 -0.62
⋮	⋮
z	-1.14 0.69 -0.01 0.23 ... 1.47 -0.07 -0.20
\0	0.14 0.28 2.09 -0.13 ... 0.70 -0.83 -1.08

Table 2: Random real valued letter codes of length 27. The elements of each vector are randomly chosen from the interval $[-1,1]$.

Letters	Corresponding binary vectors of length 5
a	1 0 0 0 0
b	0 1 0 0 0
c	1 1 0 0 0
...	...
\0	1 1 0 1 1

Table 3: Binary non-orthogonal codes.

mentation the input letters are encoded as numerical values. The encoding affects the accuracy and convergence speed of the system and the first problem is to find the best encoding scheme for the input letters.

Moreover, when TTP mapping systems are implemented, they must be trained first on some set of letters for which the phonetic transcriptions are known and after that the system is ready to use. Usually a specialist performs the transcription of the training set, based on some phonetic rules. During the training process, at each iteration, a group of letters (3, 5, 7 letters) are presented at the input of the neural network. The output of the NN is the phoneme that corresponds to the middle input letter. Due to this fact the number of iterations during the training process is equal to the total number of letters in the training dictionary. In order to ensure an increased level of phoneme accuracy, usually a large training dictionary is used. In a large dictionary, the number of repetitions of a certain group of input letters is large enough to be properly learned by the neural network. If the available training dictionary is large enough, then the multilayer perceptron neural network can be trained with good performance using a constant learning rate [12]. However, it is not a trivial task to build a large training dictionary and it can be very time consuming. As a consequence, it would be of practical interest to have some training methodology that ensures convergence in fewer number of iterations such that a smaller training dictionary can be used.

In our experiments we have used the American English

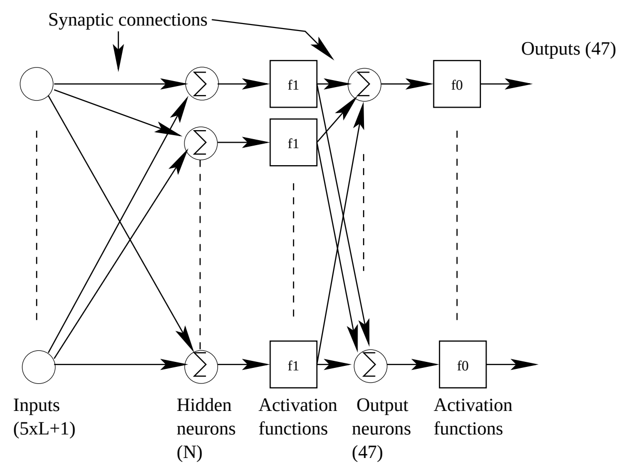
Phonemes	Corresponding binary vector
_	1 0 0 0 ... 0 0 0
aa	0 1 0 0 ... 0 0 0
⋮	⋮
zh	0 0 0 0 ... 0 0 1

Table 4: Orthogonal phoneme codes. Each vector has 47 elements of which only one is set to unity.

Carnegie Mellon University (CMU) pronunciation dictionary that contains around 10^5 words together with their phonetic transcriptions. As one can observe, the total size of the training dictionary that we have used here is huge (around 8×10^4 words). We will see in Section 4, that actually only a small fraction from this dictionary is needed in some circumstances. In order to implement a TTP mapping system based on neural networks the CMU dictionary must be preprocessed first. The following steps were implemented in order to perform pre-processing [12]:

- The words and their phoneme transcriptions were aligned such that one-to-one correspondence was obtained between the letters of each word and their phoneme symbols.
- In order to eliminate the ambiguity that can occur for multiple pronunciations of the same word, only one phonetic transcription was chosen from each entry into the dictionary.
- The whole dictionary was split into two parts. For the first part we have randomly chosen 80% from the whole CMU dictionary (each word with a single phonetic transcription). The training dictionary used in our simulations contains the phonetic transcriptions of 86821 words. The testing set contained the rest 20% (22015 words) of the whole CMU dictionary. The set used for training the NNs, and the set used for testing the NNs did not contain words in common.
- Once we have obtained the training and test sets, they were processed as follows: First, the order of the words in both sets were randomized. Second, each letter in a word is encoded using some numerical vectors. In our paper, we have used three types of letter codes: the binary vectors shown in Tab. 1, the random real codes, such as, shown in Tab. 2 and the binary non-orthogonal codes of length 5 from Tab. 3. The character \0 is introduced to represent the *graphemic null*. The number of letters in the English dictionary is 26, and together with a *graphemic null* we have 27 letters. Therefore, each vector from Tab. 1 representing a letter or space between words, has 27 elements. The random real codes have also length 27 and their elements were chosen from a sequence of real numbers with zero mean random Gaussian distribution and unity variance. A similar encoding scheme, using binary orthogonal codes, was also applied for the phoneme transcriptions. Since English can be represented with 47 phonemes including the *null phoneme* and *pseudo phonemes*, the dimension of the binary vector that encodes the phoneme is 47 (see Tab. 4).

After the database containing the words is processed as described above, the next step is to train the TTP mapping system. Training of the neural networks used in this paper is



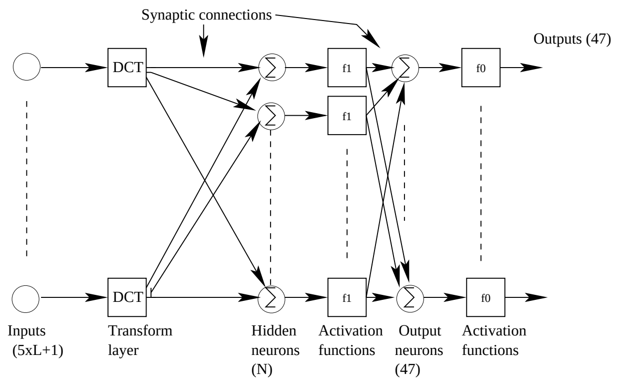


Figure 1: The block diagram of the multilayer perceptron neural network (left) and of the transform domain multilayer perceptron neural network (right).

done in online mode where the synaptic weights are updated at each training iteration [1], [2]. After the NN were trained, the performance in terms of phoneme accuracy is evaluated on the test dictionary. The neural networks and the training algorithms implemented in our experiments are briefly reviewed in the next section. Detailed description of the training algorithms are given in [3], [6]-[9].

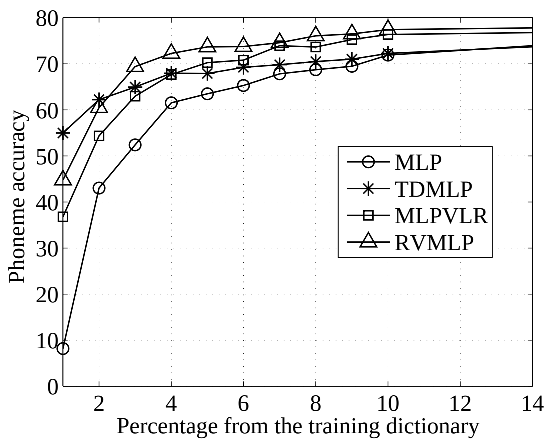


Figure 2: Phoneme accuracy obtained with the different neural networks as function of the size of the training dictionary.

3. NEURAL NETWORKS

In this paper, we study the convergence speed of two neural network structures such as multilayer perceptron neural network and the transform domain multilayer perceptron neural network. Also three types of encoding vectors for the input letters are analyzed and two training algorithms: the error back-propagation with momentum and fixed learning rate and the error back-propagation with momentum and time-varying learning rate.

The block diagram of the MLP neural networks implemented in our experiments is depicted in Fig. 1. The MLP has one input layer, one hidden layer of neurons and one output layer. The number of outputs is 47 which is equal to the length of the phoneme codes. The MLP takes five adjacent letters at the input such that the number of inputs for the MLP was:

$$N_{in} = 5 \times L_{in} + 1 = 136 \quad (1)$$

where L_{in} is the length of the letter codes that is 27 for orthogonal binary codes and for real valued random codes (see Tab. 1 and Tab. 2). In (1) the number of inputs contain also the input bias term this is why the unity is added.

The transform domain multilayer perceptron neural network has also one input layer, one hidden layer of neurons and one output layer. A number of 47 outputs and a number of 26 inputs (25 inputs due to the 5 input letters and one input bias term) are used in the TDMLP neural network.

The activation function used in the hidden layer, of both MLP and TDMLP neural networks was the hyperbolic tangent activation function described by the following formula:

$$f_i^{(1)}(n) = \frac{1 - \exp(y_i^{(1)}(n))}{1 + \exp(y_i^{(1)}(n))} \quad (2)$$

where $y_i^{(1)}(n)$ is the output of the i^{th} hidden neuron at iteration n, $f_i^{(1)}$ is the output after activation function that is propagated in the next layer and $\exp()$ is the exponential function.

At the output both MLP and TDMLP have tangential activation function described by:

$$f_i^{(o)}(n) = \frac{\exp(y_i^{(o)}(n))}{\sum_j^{47} \exp(y_j^{(o)}(n))} \quad (3)$$

where $y_i^{(o)}(n)$ is the output of the i^{th} output neuron at iteration n, $f_i^{(o)}$ is the output of the neural network.

The training algorithm for the TDMLP neural networks is derived from the standard error back-propagation with momentum in which the orthogonal transformation of the neural network inputs is included. The details of this training algorithm are given in [8] and in [10].

The multilayer perceptron neural network was trained with both fixed learning rate and time-varying learning rate. The training algorithm that used a fixed learning rate was the error back-propagation with momentum (see [1], [2] and [10] for more details). For the case of a time-varying learning rate, we have used the training algorithm introduced in [7]. All compared neural networks were fully connected and had equals numbers of synaptic weights.

4. EXPERIMENTS AND RESULTS

In this section, we show the comparative results obtained for the problem of TTP mapping. We emphasize that our main goal here is to study the influence of the training algorithm and input letter encoding on the convergence speed of the neural networks. As we have explained earlier in this paper, in the context of TTP mapping application, a fast converging TTP system enable a smaller training dictionary which in some practical cases can be of interest.

In our experiments, we have used several setups of the TTP mapping system. A MLP neural network that uses binary orthogonal letter codes and it is trained using the error back-propagation with momentum algorithm with a fixed learning rate. We have denoted this by MLP. Another TTP mapping system is composed of a MLP neural network that has random real valued letter codes at the input and it is trained with a fixed learning rate (we denoted this by RVMLP). The third compared system is composed of a MLP neural network that uses binary orthogonal letter codes at the input and it is trained by an algorithm with adaptive learning rate that was also derived from the error back-propagation with momentum algorithm (we denoted this by MLPVLR). Finally the fourth TTP mapping system is based on the transform domain MLP that uses the non-orthogonal binary codes from Tab. 3 to encode the letters and the training algorithms introduced in [8] to update the synaptic weights (this is denoted by TDMLP).

All compared neural networks were trained in online mode on the training dictionary that contained around 8×10^4 words. In order to study the phoneme accuracy for small dictionary sizes, the synaptic weights of all the compared systems were saved at 1%, 2%, $\ldots$, 100% from the training dictionary. The tests were done for all these synaptic weights and the results are shown in Fig. 2. For instance in Fig. 2 the phoneme accuracy at 5% is obtained after training the neural networks with 4000 words. As we can see from this figure, for very small training dictionaries (up to 2000 words) the TDMLP offers better phoneme accuracy compared with the other neural networks. For higher sizes of the training dictionary, the MLP that uses random real valued letter codes provides the best mapping accuracy. For very large size of the training dictionary, the performances of the four neural networks tends to stabilize around the same level of the phoneme accuracy.

5. CONCLUSIONS

In this paper, the problem of text-to-phoneme mapping implemented by neural networks was addressed. We have shown that random real valued codes used for input letters can increase the convergence speed of the multilayer perceptron neural network in the context of TTP mapping application. Although, TTP mapping is usually done in off line mode, a fast converging system bring some advantages. Besides, reduction of the processing (training) time, implementing a fast TTP mapping system greatly reduces the required size of the training dictionary.

REFERENCES

[1] S. Haykin, *Neural Networks - A Comprehensive Foundation*-2nd Ed., Pretince Hall, New York, 1999.

[2] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[3] M. Embrechts and F. Arciniegas, ”Neural Networks for Text-To-Speech Recognition”, *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, vol. 5, 2000, pp. 3582-3587.

[4] J. Suontausta and J. Häkkinen, “Decision Tree Based Text-To-Phoneme Mapping for Speech Recognition”, in *Proc. of the Int. Conf. on Spoken Language Processing*, October 2000.

[5] K. Jensen and S. Riis, “Self-Organizing Letter Code-Book for Text-To-Phoneme Neural Network Model,”in *Proc. of the Int. Conf. on Spoken Language Processing*, October 2000.

[6] E. B. Bilcu, J. Astola, and J. Saarinen, “Comparative Study of Letter Encoding for Text-To-Phoneme Mapping,”in *Proc. of XIII European Signal Processing Conference, EUSIPCO 2005*, Antalya Turkey, September 2005.

[7] E. B. Bilcu, J. Suontausta, and J. Saarinen, “Text-To-Phoneme Mapping Using a Fast Neural Network with Adaptive Learning Rate,”in *Proc. WSEAS 2003*, Crete, Greece, July 2003.

[8] E. B. Bilcu, J. Suontausta, and J. Saarinen, “A New Transform Domain Neural Network for Text-To-Phoneme Mapping,”in *Proc. of the 6th WSEAS Multiconference on Circuits, Systems, Communications and Computers, CSCC 2002*, July 2002, pp. 4591-4596.

[9] E. B. Bilcu, P. Salmela, J. Suontausta, and J. Saarinen, “Application of the Neural Networks for Text-To-Phoneme Mapping, ”in *Proc. of XI European Signal Processing Conference, EUSIPCO 2002*, Toulouse, France, September 2002, pp. 97-100.

[10] E. B. Bilcu, *Neural Networks for Text-To-Phoneme Mapping*-Licentiate thesis, Tampere University of Technology, Tampere, Finland, 2006.

[11] R. A. Jacobs, ”Increased Rates of Convergence Through Learning Rate Adaptation”, *Neural Networks*, vol. 1, 1988, pp. 295-307.

[12] U. Bhattacharya and S. K. Parui, “Self-Adaptive Learning Rates in Backpropagation Algorithm Improve its Function Approximation Performance,”in *Proc. of Int. Conf on Neural Networks*, vol. 5, 1995, pp. 2784-2788.

[13] C-C Yu and B-D Liu, “A Backpropagation Algorithm with Adaptive Learning Rate and Momentum Coefficient,”in *Proc. of the Int. Joint Conf. on Neural Networks, IJCNN 2002*, vol. 2, May 2002, pp. 1218-1223.