COMPARATIVE STUDY OF LETTER ENCODING FOR TEXT-TO-PHONEME MAPPING

Enikő Beatrice Bilcu, Jaakko Astola

Institute of Signal Processing
Tampere University of Technology
Korkeakoulunkatu 1, FIN-33720, Tampere, Finland
email: bilcub@cs.tut.fi, jta@cs.tut.fi

Jukka Saarinen

Multimedia Technologies Laboratory
Nokia Research Center
Visiokatu 1, FIN-33720, Tampere, Finland
e-mail:Jukka.P.Saarinen@nokia.com

ABSTRACT

Text-to-phoneme mapping is a very important preliminary step in any text-to-speech synthesis system. In this paper, we study the performances of the multilayer perceptron (MLP) neural network for the problem of text-to-phoneme mapping. Specifically, we study the influence of the input letter encoding in the conversion accuracy of such system. We show, that for large network complexities the orthogonal binary codes (as introduced in NetTalk) gives better performance. On the other hand in applications that require very small memory load and computational complexity other compact codes may be more suitable. This study is a first step toward implementation a neural network based text-to-phoneme mapping in mobile devices.

1. INTRODUCTION

Speech synthesis and speech recognition has attracted the attention of many researchers during last few decades. One very important preliminary step of any speech synthesis system is the text-to-phoneme (TTP) mapping which converts a written text into its phonetic transcription. From the obtained phonetic transcriptions the synthetic speech is obtained next by means of other processing blocks. Also, in speaker independent speech recognition systems based on phonemes, there is a necessity to define a dictionary in terms of phonetic transcriptions. The transcriptions can be based on a predefined look-up table or some mathematical model that gives the transcription for any given word. Clearly, the former method is more accurate than the latter one. However, the model based approach requires less memory and can be straightforwardly applied for new words. Our paper deals with the application of neural networks (NN) into the text-to-phoneme (TTP) mapping [7].

Neural networks (NNs) has been shown to provide a solution for the problem of phonetic transcription [2]-[9]. In [3], MLP networks are applied to the TTP mapping and the mapping is based on the context information in which the letters occur. In this publication, the letters are encoded using an additional neural network for each input letter. The increase in the phoneme accuracy was not significant but the computational complexity and memory load increased. Also, the approach in [3] requires a more complicated training procedure due to the extra NN. In [9] and [11] we have studied the performances obtained with different neural network structures for the problem of TTP. In that publications we have compared the phoneme accuracy obtained with the MLP, recurrent neural networks and the transform domain MLP for different input context dependency. However, in the above mentioned papers there is no comparative study on the use of different codes for the input letters.

For real time applications, the requirement is to obtain fast networks that uses a small amount of memory and also can provide accurate recognition rates for a relatively small computational complexity. Usually, each letter and each phoneme are represented as binary vectors for the neural network. If these vectors has large dimensions, also the input-output layers of the NN scales correspondingly, that easily results in large number of weights especially if context vectors are used. Although, the context dependence has been shown to increase the phoneme recognition accuracy, the large number of the parameters of such NN is not desired from the application point of view. Moreover, in some applications, such as mobile devices, the memory restrictions and low computational cost will always play an important role. Although the memory capabilities and computational power of such devices increases very fast, there will be more and more applications that must run in parallel. This inevitable trend, makes the memory load and computational power to play an important role in the evaluation of different applications. As a consequence, in this paper we study the performances of the multilayer perceptron (MLP) neural network applied to TTP for several input codes. The purpose of this study is to compare the recognition rates obtained with different orthogonal and non-orthogonal input codes.

2. PROBLEM FORMULATION

In this paper, we address the problem of text-to-phoneme mapping for isolated words. Our goal is to obtain the phonetic transcriptions of a number of isolated words using the MLP neural network. The dictionary used for training and testing the neural networks in our experiments was the Carnegie Mellon University (CMU) pronunciation dictionary. In order to implement TTP with NNs, the data from the dictionary has to be pre-processed. The following steps briefly describe the data pre-processing (more details can be found in [8] and [9]):

- The words and their phoneme transcriptions were aligned such that one-to-one correspondence was obtained between the letters of each word and its phoneme symbols.
- Only one phonetic transcription was chosen from each entry into the dictionary.
- The whole dictionary was split into two parts (a training part containing 80% from the whole CMU dictionary and a testing part containing the rest of 20% words. 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 testing sets, they are processed as follows: first, the order of the words in both sets were randomized. After that, each letter in a word is coded using orthogonal or non-orthogonal vectors (also the *graphemic null* is encoded).
- Similar coding scheme 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 codes the phoneme is 47 such as shown in Tab. 1.

Here we study the performance, in terms of phoneme accuracy, of such system when the input letters are encoded in several different ways. More specifically, we have used the following vector codes for the input letters:

- **Orthogonal binary codes (OBC)** as shown in Tab. 2. The length of a vector corresponding to a single letter has 27 elements (there are 26 letters in the English alphabet and the space between the words). Usually orthogonal vectors are used in order to increase the phoneme accuracy and to speed-up the training of the neural network [3]. A simple straightforward approach is to use the codes from Tab. 2.
- **Non-orthogonal binary codes (NOBC)** as shown in Tab. 3. The vector to encode a single letter has length 5 (5 bits are enough to encode 27 characters). These codes, although non-orthogonal, have the advantage of having a much shorter length than the previous ones. However, the inputs of the NN are correlated in this case, and we should expect the performance of the neural network to decrease.
- **Non-orthogonal codes of -1 and $+1$ (NOC)** as shown in Tab. 4. These codes are obtained from the ones in Tab. 3 replacing the zero bits with -1. The non-orthogonal binary codes from Tab. 3 have non-negative values. Changing the zero bits with -1 we increase the dynamic range of the inputs.
- **Random real valued codes (RC)** In this experiment the codes of the input letters are obtained from a random Gaussian-distributed sequence of real numbers. The length of the codes was 5 and the random sequence has zero mean and unity variance. The aim of using these codes is to study the performance in terms of phoneme accuracy when the inputs of the neural network are uncorrelated and non-binary. Moreover, the elements of each code have positive and negative values in order to expand the dynamic range of the neural network inputs. We should emphasize here that the random codes are generated just once at the beginning of the training procedure and the same codes are used also for testing the NN.
- **DCT codes (DCT)** In this case the letters are encoded as shown in Tab. 3 and transformed using the Discrete Cosine Transform (DCT) prior to application to the NN. Moreover, the training algorithm of the neural network changes due to this fact (see [9] for more details). A block diagram of the transform domain MLP (TDMLP) is depicted in Fig. 2 and a detailed description of the training algorithm can be found in [9]. By using this encoding of the input letters we wanted to compare the phoneme accuracy obtained with the DCT codes and the performance of the NN that uses random real valued codes as described above.

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 1: Orthogonal phoneme codes. Each vector has 47 elements of which only one is set to value of one.

Letters	Corresponding binary vector
a	1 0 0 0 . . . 0 0 0
b	0 1 0 0 . . . 0 0 0
⋮	⋮
y	0 0 0 0 . . . 1 0 0
z	0 0 0 0 . . . 0 1 0
\0	0 0 0 0 . . . 0 0 1

Table 2: Orthogonal letter codes. Each vector has 27 elements of which only one is set to value of one.

Letters	Corresponding binary vectors (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: Non-orthogonal letter codes. Each vector has 5 elements of 0 and 1.

Letters	Corresponding binary vectors (length 5)
a	1 -1 -1 -1 -1
b	-1 1 -1 -1 -1
c	1 1 -1 -1 -1
. . .	. . .
\0	1 1 -1 1 1

Table 4: Non-orthogonal letter codes of $\{-1, 1\}$. Each vector has 5 elements of -1 and $+1$.

To implement our TTP mapping system we have chosen to use the MLP neural network due to its simplicity of implementation. In our experiments we have used three layered neural networks with one input layer, one hidden layer and one output layer. To increase the phoneme accuracy, 5 letters were considered at the input of the network with the middle one being the letter to be transcribed (see [8], [9] and the references therein). A block diagram of such neural network is depicted in Fig. 1.

The MLP neural network was trained using the back-propagation with momentum algorithm. The hidden neurons have hyperbolic tangent activation functions:

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

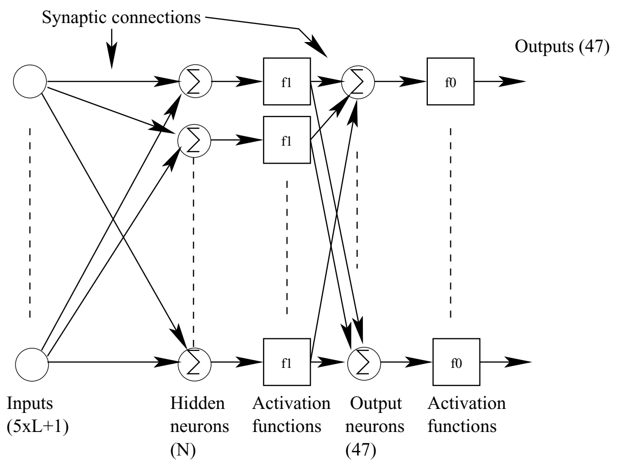

Figure 1: Block diagram of the multilayer perceptron (MLP) neural network.

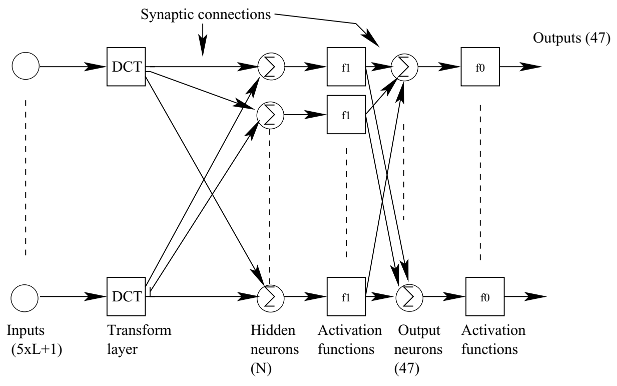

Figure 2: Block diagram of the transform domain multilayer perceptron (TDMLP) neural network.

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.

The output neurons have tangential activation function described by:

$$f_i^{(o)}(n) = \frac{\exp(y_i^o(n))}{\overset{47}{\underset{j}{}}\exp(y_j^o(n))} \tag{2}$$

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 number of synaptic connections between the neurons of the neural network can be computed as follows:

$$M = (5L+1)N + (N+1)O \tag{3}$$

where L is the length of the vector used to encode a single input letter, N is the number of neurons in the hidden layer, and O is the number of output neurons ($O = 47$ in this case).

The same formula (3) can be applied for both MLP and TDMLP neural networks to compute the number of synaptic neurons. The difference is the length of letter codes. For the binary orthogonal codes, from Tab. 2, $L = 27$, while for the other codes $L = 5$. The term $5L + 1$ appears due to the five input letters plus the input bias while the term $N + 1$ appears due to N hidden neurons and the bias in the hidden layer.

From (3) we can see, that if one uses larger vectors to encode the input letters, the number of synaptic connections M will be larger when the number of hidden neurons N is kept constant. This increases the memory and computational load of a system that implements TTP mapping. Although, available memory and computational power of the mobile devices continuously increases, there will be more and more applications to run in parallel. As a consequence the interest to have applications with low complexity will still be of great interest. One alternative to decrease the complexity of the above mentioned TTP mapping system is to reduce the number N of hidden neurons. Anyway this cannot be reduced to much without decreasing the performance of the system. Another way to reduce the memory load and computational complexity is to decrease the number of inputs of the neural network. This can be done either by reducing the number of input letters or by reducing the length of the vectors used to encode the letters. In text-to-phoneme mapping the phoneme accuracy is highly influenced by the number of input letters (context dependence [1], [2], [3]) therefore, the former solution is not of interest. The only way to reduce the complexity is to shorten the letter codes and this is the reason why we studied the above mentioned codes[1].

3. EXPERIMENTAL RESULTS

In this section, we show the performances in terms of phoneme accuracy obtained with the MLP neural network for text-to-phoneme mapping. In our experiments we have implemented and tested five different types of encoding schemes. All compared codes have advantages and disadvantages: the OBC, used in many other implementations [3], [7]-[11], increase the memory load of the neural network. In order to decrease the memory load of a TTP mapping system using neural networks, different types of codes, with shorter length, can be implemented. To study the effect of such short input codes we have selected four different methods for letter encoding. Two of these are non-orthogonal codes whereas in the other two some orthogonalization techniques are implemented. When the DCT codes are used to perform TTP mapping the neural network training is different due to the transform layer. Normalization of the synaptic weight corrections must be introduced into the training algorithm, that makes the training more complicated [9]. However, once trained, the neural network using DCT input codes is used exactly as the standard MLP neural network and have the same computational and memory load.

We emphasize here that in our implementation the problem of isolated word transcription is addressed. As a consequence, the dependence between adjacent words are not taken into account. Therefore, when the first letter of a word is transcribed, the input vector of the NN is formed by concatenation of five letter codes: two codes for \0 (see Tab. 2 to Tab. 4), the code of the current letter and the codes corresponding to letter 2 and 3 from the current word. A similar approach is done when the last letter of a word must be transcribed. In this case the first elements of the input vector correspond to the last three letters of the word (the code of

[1]Actually one could decrease the number of neural network outputs, but this only decreases the number of synaptic weights in the output layer.

the current letter being in the middle of the input vector) and the last elements corresponds to two spaces.

Complexity	NOBC	RC	NOC	DCT
485	62.41	63.83	62.78	62.85
923	71.32	70.87	71.50	72.03
1361	74.63	74.70	74.93	73.94
1799	75.95	75.42	76.28	75.66
2383	76.90	76.84	78.44	77.02
3551	77.62	77.69	79.07	78.73
4427	77.75	77.14	78.83	78.55
5303	77.65	78.05	78.79	78.70
6325	77.83	77.64	79.06	77.93
7347	77.91	77.65	78.95	77.68
8807	75.95	76.50	77.56	76.96
Complexity	OBC			
413	40.75			
962	72.24			
1328	75.11			
1877	77.50			
2426	79.97			
3524	81.37			
4439	82.68			
5354	82.41			
6269	82.74			
7367	82.81			
8831	83.02			

Table 5: Phoneme accuracy obtained with five different input codes: non-orthogonal binary codes (NOBC), random real valued codes, non-orthogonal codes of $\{-1,+1\}$ (NOC), DCT domain codes (DCT) and orthogonal binary codes (OBC).

The phoneme accuracy obtained with the five input letter codes are shown in Tab. 5. Analyzing the results from this table we see that for very small number of synaptic weights (between 400 and 500), the shorter codes (NOBC, RC, NOC and DCT) provide much higher accuracy (the difference is around 20%). For moderate number of synaptic weights (around 1000 and 1300) all codes provide slightly the same performances. When the number of synaptic weights is increased more, the use of the orthogonal binary codes (OBC) increase the phoneme accuracy of the neural network (the difference is between 4 to 5 percents).

From these results, we can conclude that in applications which require a very low memory load, the orthogonal binary codes does not provide good enough performance. In these cases, shorter codes, as the ones presented here, provide higher phoneme accuracy. However, when the neural network complexity if not of importance, the OBC codes, applied to larger neural networks, provide an additional 4 to 5 percents increase in the phoneme accuracy. The above mentioned encoding schemes can be also implemented in more sophisticated TTP mapping systems such as the one proposed in [10] for the multilingual case.

4. CONCLUSIONS

In this paper, we have studied the performance in terms of phoneme accuracy, of a TTP mapping system based on neural networks. Several different orthogonal and non-orthogonal codes were implemented to encode the input letters. We have seen that the binary orthogonal codes as used in NetTalk provide good phoneme accuracy when the number of neurons in the hidden layer is relatively large. This increases the complexity of the system. On the other hand, shorter codes provide much better performance at smaller network complexity. The work and results presented in this paper can be part of the preliminary steps toward practical implementations of such system, in mobile devices.

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] K. Jensen and S. Riis, *"Self-Organizing Letter Code-Book for Text-to-Phoneme Neural Network Model"*, Proceedings of the International Conference on Spoken Language Processing, 2000.

[4] M. Embrechts and F. Arciniegas, *"Neural Networks for Text-to-Speech Phoneme Recognition"*, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 5, 2000, pp. 3582-3587.

[5] M. Adamson and R. Damper, *"A Recurrent Network that Learns to Pronounce English Text"*, Proceedings of the International Conference on Spoken Language Processing, Vol. 3, 1996, pp. 1704-1707.

[6] V. Pagel, K. Lenzo and A. Black, *"Letter to Sound Rules for Accented Lexicon Compression"*, Proceedings of the International Conference on Spoken Language Processing, Vol. 5, 1998, pp. 2015-2018.

[7] N. McCulloch, M. Bedworth and J. Bridle, *"NetSpeak - a re-implementation of NetTalk"*, Computer Speech and Language 2, 1987, p. 289-301.

[8] E. B. Bilcu, P. Salmela and J. Suontausta, *"Application of neural networks for text-to-phoneme mapping"*, Proceedings of the European Signal Processing Conference, 2002, pp....

[9] E. B. Bilcu, J. Suontausta and J. Saarinen, *"A New Transform Domain Neural Network for Text-To-Phoneme Mapping"*, Proceedings of the 6th WSEAS Multiconference on Circuits, Systems, Communications and Computers, 2002, pp.

[10] E. B. Bilcu, J. Astola, J. Saarinen - *"A Hybrid Neural Network Rule/Based System for Bilingual Text-To-Phoneme Mapping,"* - Proceedings of the 14th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2004, Sao Luis, Brazil, September, 2004.

[11] E.B. Bilcu, J. Suontausta and J. Saarinen, *A study on different neural network architectures applied to text-to-phoneme mapping*, in Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, 2003, Vol. 2, 18-20 Sept. 2003, pp: 892 - 896.