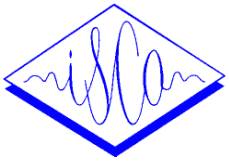

Fourth ISCA ITRW on
Speech Synthesis (SSW-4))
Perthshire, Scotland
August 29 - September 1, 2001

A Multi-lingual System for the Determination of Phonetic Word Stress Using Soft Feature Selection by Neural Networks

Horst-Udo Hain, Hans Georg Zimmermann

Siemens AG, Corporate Technology, 81730 Munich, Germany
{horst-udo.hain,georg.zimmermann}@mchp.siemens.de

Abstract

Any TTS system requires a routine to determine the transcription of out of vocabulary (OOV) words. This transcription contains three information: the phoneme sequence, the position of syllable boundaries and the position of word stress. In the TTS system "Papageno", the phonemes and syllable boundaries are determined by a neural network proposed in [1]. In the same paper also a second network for word stress determination was proposed. A similar architecture is used here, enhanced by a diagonal matrix between the input and the hidden layer penalised by weight decay. Weight decay is a strategy to limit the growth of a weight unless it is really necessary. It can be used to improve the generalisation ability of the network.

Keywords: multi-lingual TTS, neural networks, language independent, weight decay

1. Introduction

One problem for multi-lingual TTS systems is the determination of the transcription for words not contained in the phonetic dictionary. In the TTS system "Papageno", parts of the unknown word are looked up in the dictionary and the transcriptions found there are connected to the complete transcription. Gaps between the parts are filled using a neural network. If the beginning of the word was found in the dictionary, then the word stress of this part is used for the complete word. If not, then the word stress is determined by a second neural network.

The architecture proposed in [1] was changed to achieve better results. The input in the old network were the first syllables of the word. Now the first phonemes are used (see figure 1). This network is independent from errors made by the determination of the syllable boundaries. The output has as many nodes as input phonemes are used. The output node with the maximum value determines the stressed phoneme, of course only if it belongs to a stressable phoneme.

But during the training of neural networks it is often hard to find the appropriate number of free parameters

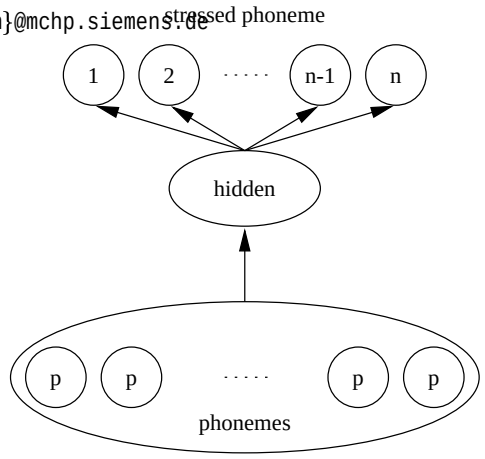

Figure 1: Normal neural network for word stress prediction.

for the given task. This number strongly depends on the architecture and the training data. If the network is too complex and there are not enough training examples, the network will be over-fitting the data. In the opposite case an under-fitting occurs.

One strategy could be to start with a huge network and then to prune weights that are useless or even disturbing. This was used in the first approach with quite good results [1]. But weight pruning has to be done in a separate step and is not a part of the training procedure. It has to be done in many iterations and is time consuming. In opposite to that weight decay is done online during the training.

2. Basics of Pruning

In large networks with a large number of free parameters there are three classes of weights:

1. weights that have learned something reasonable,
2. weights that have learned something apparently reasonable (over-fitting), and
3. weights with no importance which do not disturb on the training set.

One task for the training process is to find out which class a weight belongs to. There are several possibilities to achieve this. In this paper, weight pruning and weight decay are proposed.

2.1. Weight Pruning with EBD

The background of weight pruning is the question whether a weight has any influence to the network or not, or in other words, does it have any consequence for the error function if a weight is set to zero [2]. Therefore the Taylor approximation of the error function

$$E(w+\Delta w) = E(w) + E'(w)\Delta w + \frac{1}{2}\Delta w^T E''(w)\Delta w \quad (1)$$

is used to calculate the values $E(0)$ (by $\Delta w = -w$) and $E(w_{min})$ (by $E(w_{min}) = \min_{\Delta w} E(w+\Delta w)$). The difference of both values is the test value t:

$$t = E(0) - E(w_{min}) \quad (2)$$

There are two ranges of interest for this test value:

$t > 0$ If t is significant greater than zero, then the weight is necessary and cannot be deleted.

$t \sim 0$ If t is nearly zero, then the weight is a candidate for the deletion.

One pruning algorithm for weight pruning is "Early Brain Damage" (EBD). This is an adaption of the "Optimal Brain Damage" (OBD) algorithm with the advantage that it is applyable before the training has reached a local minimum.

Weight pruning is performed after a number of iterations. In the late stopping approach it is assumed that there is already a structure within the weights, so they are not initialised after the pruning step. It is also important to prune not too many weights at a time to ensure that the learned structure isn't destroyed. As shown in section 6.2, a rate of 2 percent per pruning iteration works well, but 5 percent might be too much.

2.2. Basics of Weight Decay

Weight decay is a strategy to find the not disturbing weights [3]. It decreases the complexity of a network by limiting the weight to prevent it from growing too large unless it is really necessary. This is done by adding a penalty term to the error function:

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum w^2 \,, \quad (3)$$

where E_0 is the error function and λ is a parameter to control the penalisation of large weights. For a gradient descent algorithm, the updated weight w_+ is calculated as

$$\begin{aligned} w_+ &= w - \eta\frac{\partial E}{\partial w} \\ &= w - \eta\lambda w - \eta\frac{\partial E_0}{\partial w} \\ &= (1-\eta\lambda)w - \eta\frac{\partial E_0}{\partial w} \end{aligned} \quad (4)$$

There are two terms in equation (4). The second term is the gradient caused by E_0. If there is no gradient because the weight is one of the useless weights (class 3), then only the first term is used to update the weight. This term has the form $w_k = (1-\eta\lambda)^k w_0$ with $0 < (1-\eta\lambda) < 1$ and pushes the weight to zero after a number of iterations.

Weight decay has two positive effects on generalisation: it suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem and it can suppress some of the effects of static noise on the targets. In connection with weight pruning, e. g. with EBD-Pruning, weight decay "'prepares"' the weights of class 2 for the pruning because they are rarely used and therefore decreased to nearly zero. This results in a small test value during the pruning step and the weights are eliminated.

3. Network Enhanced by Weight Decay

In the new architecture, a fourth cluster `wd_input` is inserted between `input` and `hidden` cluster (see figure 2). The connection between `input` and `wd_input` is realised by a diagonal matrix. The values on the diagonal are initialised with 1.

Only for this connection the weight decay penalty term is added to the error function. Thereby a kind of input pruning is achieved.

There is no interest in setting a bias to linear models by weight decay. Weight decay is used for a special task, and this is to select important features for the word stress determination. We choose $\lambda = 0.001$ which is of the size of the smallest learning rate. Thus the learning is not dominated by the weight decay (cf. equation (4)).

The implementation of weight decay in this way leads to a preprocessing that determines how important the input information is for the solution of the problem. This is a soft feature selection, because the algorithm itself makes the decision. There is no "hard" decision using expert knowledge.

4. Training Data

The system in [1] was trained for the languages English, German and Dutch using the CELEX dictionaries [4]. The same dictionaries were used for the new architecture. To determine the context size, a statistic about the

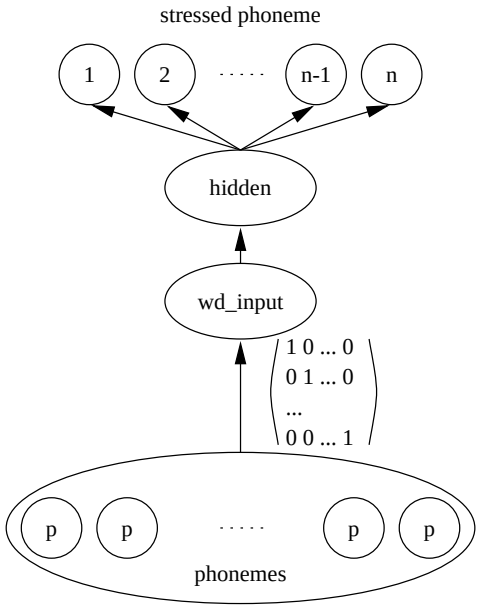

Figure 2: Neural network for word stress prediction with weight decay.

n	German	Dutch	English
1	19.16	11.64	5.16
2	33.93	33.77	45.85
3	11.75	14.51	18.70
4	10.55	10.00	12.45
5	9.48	9.27	7.50
6	6.16	6.82	4.49
7	3.72	4.95	2.79
8	2.22	3.54	1.60
9	1.39	2.39	0.69
10	0.87	1.56	0.44
11	0.48	0.84	0.21
12	0.17	0.42	0.09
13	0.076	0.17	0.04
14	0.031	0.08	0.008
15	0.012	0.03	0.003
16	0.005	0.03	
17		0.004	
18	0.001	0.004	
19		0.0007	
20		0.0003	

Table 1: Distribution of stressed phonemes for German, Dutch and English.

stressed phonemes is used. The statistics are shown in table 1.

The table contains the information how often the n-th phoneme was stressed. For instance for German in 34 percent of the words the word stress is on the second phoneme. For Dutch and German, all positions stressed more than one percent are included in the training data which results in a context of 9 phonemes for German and 10 for Dutch. Due to the small dictionary for English, the context was expanded to 11 to increase the number of patterns.

5. Training Procedure

The training of the network is automated using the Tcl interface of SENN [5]. The learning algorithm is VarioEta, the patterns are randomly selected. The set of patterns is divided into 70 percent for training and 30 percent for testing. In a first step the weights are initialised with random values (except the diagonal connection). The training starts with a learning rate $\eta = 0.1$ which is decreased to 0.01 and 0.001 if there was no progress on the test set during the last 20 iterations. Progress means that the so called signal for the output nodes has been increased. A signal is a mechanism in SENN to count the number of correct handled patterns. One can define a measure which determines whether the calculated output for a pattern is the desired one or not. The number of the correct handled patterns is counted and divided by the total number of patterns. The value of a signal lies in the range $[0,1]$ where 1 means that all patterns are handled right. If the signal increases, the weights are saved to file.

After 20 iterations without progress with $\eta = 0.001$ the weight pruning starts. In every pruning step 2 or 5 percent of the weights are deleted using EBD-Pruning. Weights with a negative test value are not inserted. After the pruning the remaining weights are not initialised. The training continues with $\eta = 0.01$ and 0.001 again for 20 iterations without progress. A value of 0.1 could be too large and possibly destroy the structure within the network.

The training is interrupted by an external process that tests the saved weights on the complete dictionary. If the results do not increase for a larger number of iteration, the training script is stopped.

For weight decay there is no difference in the controlling training script because weight decay is a part of the learning algorithm. As mentioned in section 2.2, weight decay means to add a penalty term to the error function. There are no changes in the script necessary. To keep the results comparable, even the identical sets for training and testing are used. The only difference between the two approaches is the definition of the network architecture in the topology file.

6. Results

Most of the tests were performed for English because it has the smallest dictionary and therefore the smallest number of patterns which leads to a faster training. The results for English are compared with German and Dutch.

6.1. The Influence of the Weight Pruning

A very interesting fact is that in this architecture the weight pruning increases the performance of the NNs. Without weight pruning, only 95.1 percent correct words are achieved for the NN with 75 hidden nodes. At this point the NN contains 40500 weights for 35859 training patterns. The NN with 200 hidden nodes has even 108000 weights, 3 times more than patterns available. The NN has the chance to over-fit the data.

During the pruning unneeded weights are deleted as described in section 2.1. The adaptation to the noise contained in the data becomes more and more difficult, and the less the number of weights is, the more the algorithm has to use the remaining weights to concentrate on the important information. The dependency of the achieved results on the remaining weights is shown in figure 3 for the network with 200 hidden nodes and a pruning rate of 2 percent (`h200 p2`).

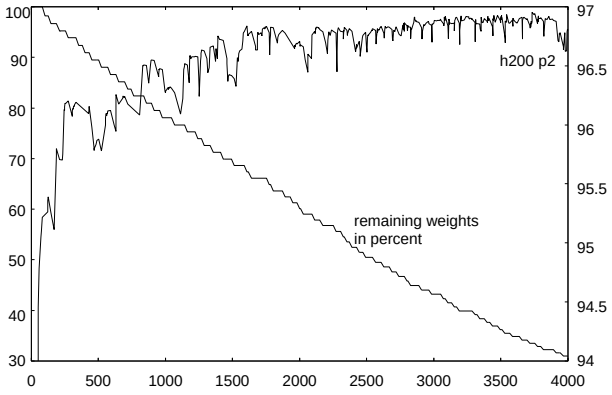

Figure 3: Dependency of the achieved results on the number of remaining weights for `h200 p2`. On the left side the axis for the declining line shows the remaining weights in percent. The right axis shows the correct stressed words in percent.

6.2. Without Weight Decay

The first part of the test was performed without weight decay. Two different numbers of hidden nodes (75 and 200) and also two pruning rates (2 and 5 percent) have been used. The results are shown in in table 2 and figure 4.

The table and the picture show that the best result is achieved with 200 hidden nodes, as expected. Both lines show the same behaviour till iteration 1500, but from then on there is no progress for the training with 5 percent pruning. This pruning rate is too much for this architecture. Too many weights are deleted during the pruning step, so the structure that was found during the training is destroyed too rapidly.

NN	pruning rate	correct words (%)	in iteration	alive weights total	alive weights in %
h75	2	96.42	2353	17671	43.6
h75	5	96.50	844	23588	58.2
h200	2	96.95	3732	36167	33.5
h200	5	96.83	1657	14405	13.3

Table 2: Best results for English, with 2 and 5 percent pruning

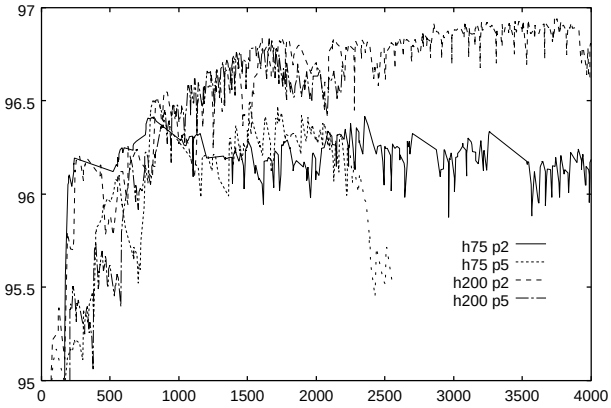

Figure 4: Correct stressed words (in percent) with 75 and 200 hidden nodes and 2 and 5 percent pruning rate. `h75` stands for 75 hidden nodes, `p2` for 2 percent pruning. The two lower lines show the results for 75 hidden nodes, the upper lines for 200 hidden nodes.

6.3. With Weight Decay

Similar tests have been performed with weight decay. As one can see in table 3 and figure 5, weight decay not only increases the achieved results, but also the training speed.

network	percent correct	after iteration
h75	96.42	2353
h75 wd	97.20	1339
h200	96.95	3732
h200 wd	97.17	1169

Table 3: Best results for English, with and without weight decay.

To be able to compare the results, the lines `h75` and `h200` again show the results for 75 and 200 hidden nodes without weight decay and 2 percent pruning rate (cf. figure 4). The two other lines show the results with weight decay. The best result for 75 hidden nodes without weight decay is 96.4 percent which is achieved after 2353 iterations. 96.4 percent with weight decay are achieved after only 502 iterations, and the best result is 97.2 percent af-

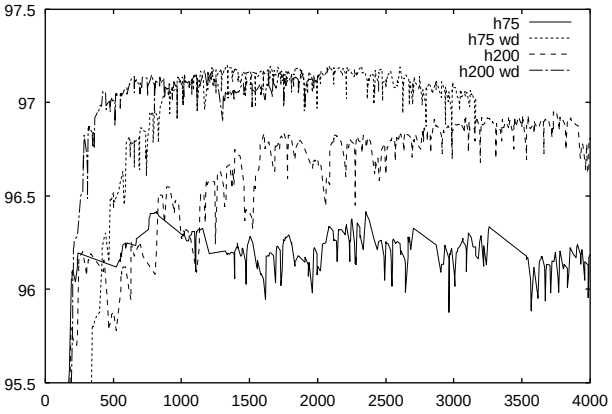

Figure 5: Correct stressed words (in percent) with and without weight decay. `h75` stands for 75 hidden nodes without weight decay, `h75 wd` for the results with weight decay. The two lower lines show the results without weight decay, the upper lines with weight decay. The pruning rate is always 2 percent.

ter 1339 iterations. The result for 200 nodes (97.17 percent) is similar to that with 75. The training is faster with 200 nodes compared to the speed with 75 nodes.

Table 4 shows the results for German and Dutch. For German, the results are improved by 0.4 percent, for Dutch by 1.5 percent.

	network	percent correct	after iteration
German	h100	96.95	3697
	h100 wd	97.39	1582
Dutch	h200	93.47	1529
	h200 wd	94.99	856

Table 4: Results for German and Dutch, with and without weight decay.

6.4. Results of the Input Pruning

As mentioned in section 3, weight decay on the diagonal matrix between `input` and `wd_input` can be interpreted as an input pruning. In figure 6 the development of the weights of the first input phoneme for English is shown. All weights are initialised with 1. Some of them are decreased close to zero (nodes 2, 8, 10, 16, 18, 19, 31, 38, 41 – 45 and 47), some of them increased up to 1.6 (nodes 1, 12 and 46). The nodes set close to zero (0 ± 0.1) can be treated as pruned because there input value is always multiplied with a very small value. Table 5 shows the results for English with 11 input phonemes.

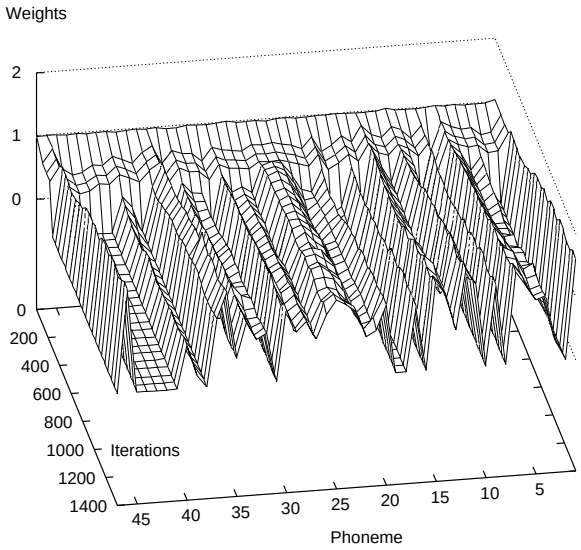

Figure 6: Development of the weights of the first input phoneme for English due to weight decay. The x-axis shows the number of the input phoneme (1 – 47), the y-axis the number of iterations, and the z-axis the value of the diagonal connection for the given phoneme between `input` and `wd_input`. Some of the weights are pushed to zero, some other are increased up to 1.6.

pho-	Context										
neme	1	2	3	4	5	6	7	8	9	10	11
aU						x	x	x	x	x	x
V							x	x	x	x	x
U	x					x		x		x	x
OI	x			x			x	x	x	x	x
8						x	x	x	x	x	x
tS		x		x		x	x	x	x	x	
h		x			x	x	x	x	x	x	x
g							x	x	x	x	x
dZ		x				x	x		x	x	x
Z	x	x	x	x	x	x	x	x	x	x	x
T		x		x	x	x	x	x	x	x	x
R	x	x	x	x	x	x					
D	x	x	x	x	x	x	x	x	x	x	x
9	x					x		x		x	x

Table 5: Pruned phonemes for English. The *x* in a column means that the weight decay connection for this context has a value close to zero. Only those phonemes are shown that are pruned in at least five positions.

Most of the phonemes are pruned at the end of the context (e. g. `/aU/` or `/V/`), one is pruned in the beginning (`/R/`), and some are pruned in nearly all input nodes (`/tS/`). There are two phonemes that are completely faded out: `/Z/` and `/D/`. That means that this

architecture with this training data is not able to gain any information from these nodes that is usable for the decision about the stress position.

In German four phonemes and in Dutch five phonemes are completely faded out.

pho-	Context								
neme	1	2	3	4	5	6	7	8	9
9	x		x			x	x	x	x
Z	x	x	x	x	x	x	x	x	x
j		x	x	x	x	x	x	x	x
x	x				x	x	x	x	x
pf		x			x	x	x	x	x
tS	x	x	x	x	x	x	x	x	x
dZ	x	x	x	x	x	x	x	x	x
ks	x					x	x	x	x
kv	x	x	x	x	x	x	x	x	x

Table 6: Pruned phonemes for German

pho-	Context									
neme	1	2	3	4	5	6	7	8	9	10
M			x		x	x	x	x	x	x
E:	x	x	x						x	x
<	x	x	x	x	x	x	x	x	x	x
*	x	x	x	x	x	x	x	x	x	x
(	x	x	x	x	x	x	x	x	x	x
!	x	x	x	x	x	x	x	x	x	x
g	x	x	x	x	x	x	x	x	x	x
dZ	x	x	x	x	x	x	x	x	x	
Z	x	x	x	x			x	x	x	x
S	x	x	x			x	x	x	x	x

Table 7: Pruned phonemes for Dutch

In total 26 to 30 percent of the weight decay connections are pushed to zero (cf. table 8).

	connections	pruned	percent
English	517	150	29
German	423	114	27
Dutch	440	118	27

Table 8: Number of values within the diagonal connection between input and wd_input in the range of $-0.001 < x < 0.001$.

7. Conclusion

In this paper the influence of weight pruning and weight decay on the training of a neural network for word stress determination is analysed. Weight pruning is a method that can handle over-fitting, and weight decay can be used to analyse the importance of the input features.

Using weight pruning only, the results on correct stressed words can be improved by some percent. But the training is slow and it takes a lot of iterations to achieve good results.

By adding the weight decay connection, a preprocessing of the input data is performed. This is a soft feature selection, because during the training the algorithm decides which input information is interesting and which is not. No expert knowledge and no manual data examination is necessary.

This kind of input pruning improves the results due to the fact that the disturbing input is faded out by the weight decay connection. The noise contained in the input data now has a lesser influence, and the training algorithm can focus on the important information. As shown in tables 3 and 4, the number of training iterations is decreased significantly. This leads to a faster training. Another positive side effect is that the pruned input nodes (phonemes) can be deleted from the network. Thereby the number of connections and thus the memory usage and computation time is decreased.

The application of weight decay improved the results for the word stress determination on word level for English from 97.0 to 97.2 percent, for German from 97.0 to 97.4 percent, and for Dutch from 93.5 to 95.0 percent.

8. References

[1] Horst-Udo Hain: A Hybrid Approach for Grapheme-to-Phoneme Conversion Using a Combination of Partial String Matching and a Neural Network. *ICLSP 2000, Beijing*

[2] Ralph Neuneier and Hans Georg Zimmermann: How to Train Neural Networks. *In: G. B. Orr and K.-R. Müller (Eds.): Neural Networks: Tricks of the Trade. Springer, 1998.*

[3] Anders Krogh and John A. Hertz: A Simple Weight Decay Can Improve Generalisation. *In: J. E. Moody, S. J. Hanson and R. P. Lippmann, eds.: Advances in Neural Information Processing Systems 4. San Mateo CA, 1995, p. 950-957*

[4] G. Burnage: CELEX: a guide for users, Technical Report, University of Nijmegen, Center for Lexical Information, Nijmegen. 1990

[5] Senn Version 3.0 User Manual. Internal Technical Report, SIEMENS AG. 1998