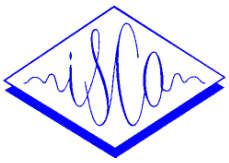

3rd European Conference on
Speech Communication and Technology
EUROSPEECH'93
Berlin, Germany, September 19-23, 1993

TRAINING OF A TIME-DELAY NEURAL NETWORK FOR SPEECH RECOGNITION BY SOLVING STIFF DIFFERENTIAL EQUATIONS

Veronika Bappert, Matthias Jobst

Lehrstuhl für allg. Elektrotechnik u. Akustik, Ruhr-Universität Bochum, D-44780 Bochum, Germany

10.21437/Eurospeech.1993-299

ABSTRACT

Time-Delay Neural Networks for robust speech recognition are usually trained using the back-propagation learning procedure. Since this learning procedure contains well known disadvantages like occasional instability or no guarantee of convergence, etc., we propose an algorithm which expresses the back-propagation in terms of solving a system of ordinary stiff differential equations. Tests have shown that the problem concerning the training of the TDNN is mathematically stiff, so that a special procedure suggested by Gear for solving stiff differential equations could be applied. Instead of optimizing the weights of the network by using fixed steps of small size, the stability of the differential equation solver allows computational steps of variable sizes that still follow the direction of the gradient's steepest descent. For the purpose of training speech recognizers, this procedure does not necessarily result in an acceleration of the computation time but it ensures the convergence without an adjustment of the learning rate and independent of the initial weight values.

Keywords: Time-Delay Neural Network, back-propagation, stiff differential equations, speech recognition

1. INTRODUCTION

Neural networks have the ability to recognize trained patterns and, additionally, to categorize similar patterns in accordance to this knowledge. Therefore, they seem to be predestinated to be applied to speech recognition tasks, but the architecture of the standard multi-layer perceptron is not designed to handle dynamically varying signals as in the case of speech. As a special neural network for speech recognition, the Time-Delay Neural Network (TDNN) was developed by Hinton et al. [1]. This architecture enables the network to recognize phonemes independent of their temporal location within the speech signal and mostly independent of their duration. Like the standard multi-layer perceptron the TDNN is usually trained using the back-propagation learning procedure. But although this kind of supervised learning is widespread, problems concerning the convergence to the global minimum, the convergence speed, and occasional instabilities are not yet solved. Therefore, we suggest to adapt an algorithm for solving differential equations to the modified back-propagation procedure for TDNNs which avoids these problems. Originally, this proceedure was successfully applied to small multi-layer perceptrons by Filkin and Owens [2]. The following section describes the algorithm and, afterwards, the modifications concerning the TDNN are illustrated. The last section demonstrates first results of speech recognition experiments and draws conclusions for the future work.

2. DESCRIPTION OF THE ALGORITHM

The supervised training of the neuronal network modifies the strength of the connections between units in such a way that there will be a correlation between the input patterns and the activities of the output neurons. During each learning iteration P patterns are presented to the network. It uses these input vectors to generate output vectors o and then compares them with the desired outputs l. The measure of the total error E is the sum of the squared errors of each pattern:

$$E = \sum_{0}^{P} \left| \vec{l}^{p} - \vec{o}^{p} \right|^{2} \qquad (1)$$

The back-propagation algorithm is a gradient search technique which performs the steepest descent on the surface of the error space. The weights and the error space are connected by the following equation [3]:

$$\Delta w_{ij} \approx \left(-\frac{\partial E_p}{\partial w_{ij}} \right) = f'\left(\sum_k w_{ik} \cdot o_k\right) \cdot o_j \cdot (l_i - o_i) \qquad (2)$$

The term w_{ij} denotes the weighted connection from unit j to unit i. The activation function f is set to the commonly used sigmoid nonlinearity:

$$f(y) = (1 + e^{-\beta y})^{-1}$$

After one iteration the new weights are determined as follows:

$$w_{t+1} = w_t + \eta \cdot \Delta w + \alpha \cdot (w_t - w_{t-1}) \qquad (3)$$

where η is the learning rate and the momentum term $\alpha \cdot (w_t - w_{t-1})$ considers the past directions of the weight changes. The equations describing the weight changes are discrete functions of time. But, assuming the training being continuous in time, we may replace equation (2) by ordinary differential equations (ODE):

$$\frac{d\vec{w}_i}{dt} = \left(-\frac{\partial E}{\partial \vec{w}_i}\right) = \Delta \vec{w}_i = \vec{g}(\vec{w}_i) \qquad (4)$$

The vector $\vec{w}_i$ denotes all connections to unit i.

An explicit method for solving ODEs is e.g. the Euler-Cauchy method:

$$\vec{w}_{t+1} = \vec{w}_t + h \cdot \vec{g}(\vec{w}_t) \qquad (5)$$

where h is the stepsize which is equivalent to the learning rate. This method corresponds to the back-propagation algorithm described in (3) setting $\alpha = 0$. The momentum term could be considered by using a multistep method. Since $\vec{g}(\vec{w}_i)$ equals Δw_{ij}, as could be seen above, equation (2) can be applied for the evaluation of the weight changes.

In some physical and technical problems differential equations occur being "stiff". They are called stiff when the different components of the system they describe decay at different rates. This means that some components reach a steady state very fast and, therefore, a small stepsize is required to achieve a given accuracy. Other components of the same system, however, reach this state very slowly which would allow a greater stepsize with the same level of accuracy [4]. For reasons of stability, a standard method like the Runge-Kutta method, would restrict the stepsize of the integration to the small one. Yet, special differential equations solvers developed for such stiff problems offer more efficient solutions.

The stiffness of a system is determined by the range of the eigenvalues λ_i. Computing the Jacobian matrix, the ratio of the largest eigenvalue to the smallest eigenvalue has to be larger than one:

$$\frac{max|\lambda_i|}{min|\lambda_i|} \gg 1 \qquad (6)$$

Filkin and Owens investigated several multi-layer perceptrons combined with different training data. They report that all these combinations turned out to be stiff [2]. In our case, stiffness also occurred in all the TDNNs we tested. Thus, a special method by Gear [5] for the integration of stiff differential equations was implemented. He defines the term stiff stability, which is less restrictive than the A-stability, allowing a higher order error term and an adaptive stepsize which follows the curvature of the error surface.

For solving nonlinear differential equations iterative methods can be applied which are either implicit or explicit. Since Gear's algorithm is an implicit predictor-corrector method, each iteration of the solution is first predicted and subsequently the accuracy of this value is improved by one or more corrector steps. A corrector equation of an order greater than one corresponds to the momentum term added in equation 3. As described above, the training of the neural network first requires the prediction of the new weights:

$$\vec{w}_i^{(t+1,0)} = \vec{w}_i^{(t)}$$

The accuracy of the weighted connections $\vec{w}_i^{(t+1)}$ is improved using the corrector equation:

$$\vec{w}_i^{(t+1,\upsilon+1)} = \left(\sum_k^n \alpha_k \cdot \vec{w}_i^{(t-k)}\right) + h \cdot \beta_n \cdot \vec{g}(\vec{w}_i^{(t+1,\upsilon)})$$

n denotes the order of the error term. The coefficients α_k and β_n related to the chosen order can be looked up, e.g., in [5]. They derive from conditions given by the characteristic polynomial and the order of the equation. h is the stepsize and ν the number of iterations. Usually, three iterations are sufficient [4].

The disadvantage of Back Propagation lies in the fixed learning rate used for the iteration of the weights, as the velocity of the training depends on this value. On the one hand, a large stepsize risks the instability of the procedure. On the other hand, the training is very time-consuming if a quite small stepsize is chosen. In contrast to that, the differential equation solver allows a variable stepsize h, which is limited only by the condition for the convergence of the corrector iteration as estimated by:

$$\left\| h \cdot \beta_n \cdot \left(\frac{\partial g_k}{\partial w}\right) \right\| < 1 \qquad (7)$$

||...|| is a matrix norm, e.g. the euclidean norm. The matrix $(\partial g_k / \partial w)$ is the Jacobian matrix of the system of differential equations. Since the convergence depends on the stepsize the condition is checked during the training. As soon as it is no longer given the stepsize has to be reduced to a suitable value. But if a larger stepsize is possible it should not be changed in any case. Usually, the order of the corrector equation is higher than one and, thus, each variation of the stepsize requires the re-calculation of several equally-spaced steps. Therefore, before changing the stepsize, the advantage of the larger stepsize and the additional effort for computing the new equally-spaced steps should be compared roughly.

3. ADAPTATION OF THE ALGORITHM TO THE TDNN

As described above, in contrast to the standard multi-layer perceptron the architecture of the TDNN is designed especially for the classification of time varying signals because it considers not only the current features but also the history of these features. In the following, a short description of a TDNN is given which corresponds to the one we used for our tests with the new learning algorithm (Fig. 1). The input layer consists of 15x17 units, so that 15 successive feature vectors, each containing 17 spectral coefficients, serve as input to the network. This layer is connected to a first hidden layer of 13x9 units, and this layer again is connected to a second hidden layer of 9x3 units. Since the TDNN should classify three different diphones, the output layer holds three units.

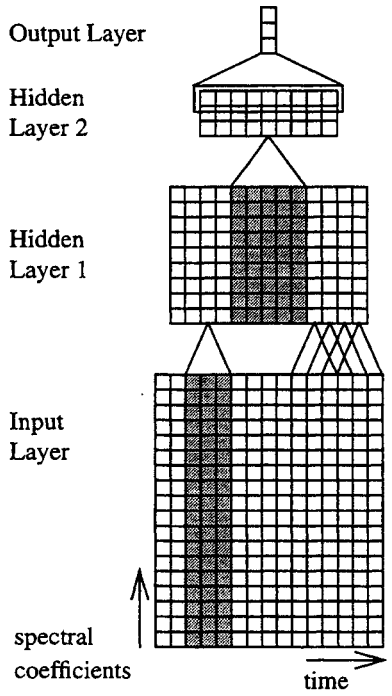

Figure 1: The TDNN architecture

Each vector of the first hidden layer consisting of 9 units is fully connected to one feature vector and two delayed versions of this vector of the input layer. Thus, the weights form a three frames wide window shifting along the input layer to connect the input vectors with the first hidden layer. Similar to this connection the first hidden layer and the second hidden layer are connected by a five-frames-wide window. Finally, the activation of the output units is obtained by integrating the activations of corresponding units in the second hidden layer. The training of the TDNN requires that each shift of the window which connects two layers is learned separately. But after each iteration the average of these weight changes is calculated to achieve one window of connections and, in this way, to achieve a correct classification unaffected by temporal shifts of the phoneme.

Since the application of the back-propagation to the TDNN required no remarkable modifications, the adaptation of our proposed algorithm did not either. The computation of the Jacobian matrix used to determine the appropriate stepsize depends on the amount of adjustable weights (eqn. 7). But compared to the neural networks tested by Owens and Filkin, our TDNNs require about forty times as many weighted connections and, thus, much more computation time to evaluate the stepsize. Therefore, we introduced some modifications to accelerate the training.

Principally, the correct determination of the stepsize needs the computation of the matrix norm, considering all training patterns and all weights. To reduce this enormous computational expenditure and, concurrently, to keep the convergence of the training we tried to base the calculation of the matrix values on constantly varying parts of the training patterns. To ensure that we obtained sufficient accuracy, we simultaneously controlled the convergence of the corrector iterations. In case that the iterations did not converge, the stepsize is reduced.

Inspired by an approach of Gear, who suggests to reduce the stepsize computations when the step did not increase since the last time, we implemented an option to reduce the number of stepsize calculations. The regular back-propagation uses a small fixed learning rate to guarantee the stability of the training, usually ca. 0.25. The stiff differential equation solver should provide the training with a larger stepsize considering equation 7. But the computational effort to obtain this stepsize is very large. In the case that the stepsize of the stiff algorithm is smaller, we assume stability and keep this stepsize for several iterations until the training progress is comparable to the one of the regular back-propagation. During these iterations the convergence is controlled.

4. FIRST RESULTS AND CONCLUSION

For the first performance test we used only a small database of the three diphones 'ba', 'da', 'ga'. These diphones were uttered in isolation by two male speakers and one female speaker. The training data consisted of 15 versions and the test data of five versions of each diphone and each speaker. Thus, there were 135 training patterns and 45 test patterns. All diphones were recorded in an acoustically treated room and were digitized at 44.1 kHz. Since humans are known to be the best speech recognition system, we selected a kind of preprocessing, namely, the loudness coefficients in accordance to the procedure after Zwicker, that is a rough simulation of our auditory system. It should have similar discriminating abilities and, thus, it should improve the performance of the speech recognizer.

At the beginning, we tested the stiffness of our TDNN using

equation 6. Compared to a small multi-layer perceptron designed for the XOR-problem, the stiffness of the TDNN is very high. Unfortunately, our expectations regarding a large stepsize were not fulfilled. Whereas the stiff equation solver applied to the XOR-problem or the encoding problem [3] allowed stepsizes of 10 and more, the training of the TDNN could only proceed with values of less than 0.01. This value is comparable to the learning rate applied by Rumelhart et al. to the encoding problem, but is still extremely small. A reason for this small stepsize is that the error space of the XOR- and the encoding problem is less complex than that of the speech-recognition task. Because of the small stepsizes our procedure needed much more iterations than the usual back-propagation applied to the TDNN. Therefore, the effort for the computation of the stepsize and for the solving of the differential equation seems of no use. Nevertheless, the decisive advantage of this algorithm lies in the ensured convergence to the global minimum. The training of a neural network applying the usual back-propagation requires several trials with different learning rates to achieve stability. In addition to that, the training should be repeated several times with different initializations of the weighted connections in order to avoid that the training is stuck in a local minimum. Since the stiff differential equation solver makes use of the Jacobian matrix, it considers the local curvature of the error surface and therefore, it guarantees convergence. Simultaneously, the stability of the training is ensured, because the stepsize is chosen in accordance with the condition for the convergence. We tested the learning procedure by starting the training with different initializations of the weights and could verify that in all cases the training converged. The recognition rates were similar to best ones achieved with the usual back-propagation, namely -speaker dependent- 100% of the test corpora and about 97% of the training corpora. Therefore, we are convinced that safe convergence and the guaranteed stability justify the additional computational effort.

The most important task for the immediate future is to reduce the computation time of the presented version of the back-propagation procedure. Then, this algorithm based on solving differential equations would provide an appropriate alternative to the usual back-propagation.

ACKNOWLEDGEMENTS

This work was partly supported by the Federal German Ministry for Research and Technology (BMFT) under grant No. IN 108 B/5. Thanks are due to Prof. Blauert for providing continuous encouragement.

REFERENCES

[1] K.J. Lang, A.H. Waibel, G. E. Hinton (1990): "A Time-Delay Neural Network Architecture for Isolated Word Recognition", in Neural Networks, Vol. 3, pp. 23-43

[2] Owens, A.J. & Filkin, D.L. (1989):"Efficient Training of the Back-Propagation Network by Solving a System of Stiff Ordinary Differential Equations"; IEEE-IJCNN 89, Bd. 2, S. 373-380.

[3] Rumelhart, D.E. & McClelland, J.L., ed. (1986). Parallel distributed processing, Volume 1: Foundations; MIT Press Cambridge.

[4] Engeln-Müllges, G. & Reutter, F. (1987). Numerische Mathematik für Ingenieure; BI Wissenschaftsverlag, Zürich.

[5] Gear, C.W. (1971). Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, NJ.

[6] Paulus, E. & Zwicker, E. (1972): "Programme zur automatischen Bestimmung der Lautheit aus Terzpegeln oder Frequenzgruppenpegeln", in Acoustica, Vol.27, Heft 5; S. 253-266.