A Time Delay Radial Basis Function Network for Phoneme Recognition

Michael R. Berthold *Intel Corporation, Santa Clara, CA95052, USA*

(to appear in Proceedings of the International Conference on Neural Networks, Orlando 1994)

***Abstract*— This paper presents the Time Delay Radial Basis Function Network (TDRBF) for recognition of phonemes. The TDRBF combines features from Time Delay Neural Networks (TDNN) and Radial Basis Functions (RBF). The ability to detect acoustic features and their temporal relationship independent of position in time is inherited from TDNN. The use of RBFs leads to shorter training times and less parameters to adjust, which makes it easier to apply TDRBF to new tasks.**

The recognition of three phonemes with about 750 training and testing tokens each was choosen as an evaluation task. The results suggest an equivalent performance of TDRBF and TDNN presented in [7], but TDRBF require much less training time to reach a good performance and in addition have a clear indication when the minimum error is reached, therefore no danger of overtraining exists.

I. Introduction

To detect features not only present in the input pattern at one point in time, but also features of the input pattern changing over time, is essential for successful recognition of speech. Conventional methods based on Multi Layer Perceptrons often use the Time Delay Neural Network (TDNN) which was first used for the recognition of speech by Waibel et al (see [7] for details). But long training times and multiple parameters which need careful adjustment make it hard to apply this network to new tasks. In contrast Radial Basis Function (RBF) based networks do not require special adjustments and the training time is much shorter than the time required to train a TDNN. But RBF networks are neither shift invariant in time, nor can they detect features in time.

The desire to combine the advantages of Radial Basis Functions with the time shift invariance of the TDNN lead to the Time Delay Radial Basis Function Network presented in this paper.

II. The Architecture

To allow a RBF to detect features over time it is necessary to present inputs not only of one point in time but of several. This leads to the structure shown in figure 1. The RBF gets in-

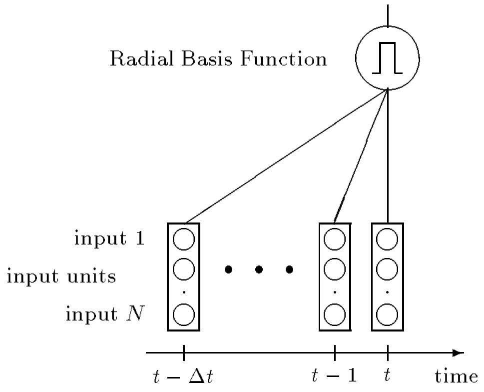

Figure 1: A RBF "looking" onto a window in time

put from several frames at different points in time. In this structure the RBF itself does not care about different frames or inputs, it handles the whole input data as one big vector, computes the distance to its prototype and applies an activation function which, in this case, is a signum function with variable radius[1] (see also

The author is now with: Forschungszentrum Informatik, Gruppe ACID (Prof. D. Schmid) Haid–und–Neu–Strasse 10–14, 76131 Karlsruhe, Germany (eMail: berthold@fzi.de)

[1] An attempt to use Gaussian and triangular activation functions did not result in performance improvements. The Signum function is also much easier (and faster!) to use.

[3]):

$$d(t) = \sum_{\tau=0}^{\Delta t} \sum_{i=0}^{n-1} (o_i^\tau - \mu_{i,\tau})^2$$

$$a(t) = \begin{cases} 1 & d(t) \leq r \\ 0 & else \end{cases}$$

$d(t)$ denotes the euclidian distance of the input data (o_i^τ: i–th component of the input vector at time τ) to the reference vector ($\mu_{i,\tau}$) of this RBF. $a(t)$ is the final activation, which depends on the radius r of the signum function.

One obvious problem with this structure is the inability to be invariant against shifts in time. If the pattern presented to the network is equal to the stored prototype but only shifted by one frame, the RBF will not respond as desired. Therefore a second layer which is simply integrating over time is used. Figure 2 shows the structure of the resulting network.

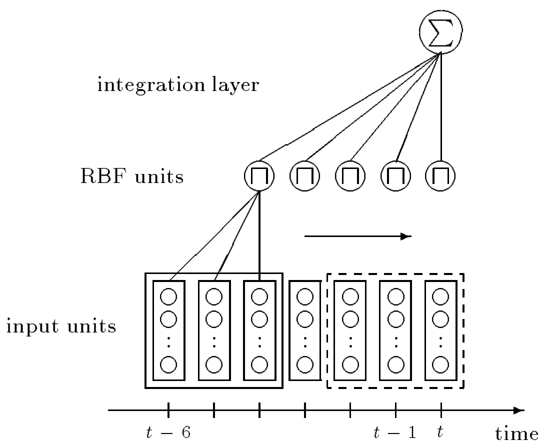

Figure 2: The structure of a 3(5)–TDRBF Network, for simplicity only one RBF for one class is shown. In practice several classes might be used and the training algorithm would have committed probably more than one RBF per class.

The network still contains RBFs looking onto small windows in time, but the second layer sums up activations of several instances of these RBFs over time. The "$x(y)$" notation used in this paper specifies the sizes of these windows in time. x indicates the number of frames each RBF takes as an input, y represents the size of the window in time the second layer is integrating.

This structure leads to the desired response to a pattern even if the pattern is shifted in time against the stored prototype. The input pattern will match the prototype of one of the instances of the corresponding RBF. The activation function of the integration layer is important for the success of this approach. Here a simple "weighted sum + sigmoid"–method was used, but one could also imagine a "winner take all" function which only picks the best response for each class. In our experiments the first approach produced the best results.

The only adjustable parameter of this network is the relation between the time–span of the Radial Basis Functions and the window size of the integration layer. As is shown later this "magic number" can be found by a few simple tries. Because for most speech problems the size of the time window is in the order of 10–20 the number of experiments required to determine the structure of the network is usually below 5–10. And due to the usage of RBFs, the training is done much faster than it is in the case of a Back–Propagation type network.

III. Training

Training this new architecture requires some thoughts about how the integration layer is being used during training. In the following experiments this layer was only used to pass the information to which class a certain window in time belongs down to the RBF layer. In one series of experiments the weights were initially set to 1.0 and were then trained using normal BackProp which did not improve the overall performance of the network significantly. All instances of RBFs in this window in time are then trained to classify the pattern correctly. Several training methods to achieve this are known, one could use an unsupervised clustering algorithm first and do some sort of supervised training during a second phase. It would also be possible to use the "growing cell structure" approach proposed by Fritzke (see [1]) to generate the first layer of the network. The algorithm used in this approach was RCE–training (see [4] and [5]) which introduces new prototypes when necessary and adjusts the radii of existing prototypes to avoid conflicts. RCE is one of the algorithms supported by Intels Ni1000 (see [2]), a Radial Basis Function Coprocessor. This form of training makes the TDRBF much easier to use than a TDNN. It is obvious when training is finished, because the algorithm does not commit any new prototypes. The danger of overtraining, known from BackProp, no longer exists.

IV. Testing with the BDG-task

To allow a comparison, the same phoneme recognition task was used for testing the TDRBF which was used for the introduction of the TDNN (see also [7]). The BDG–task uses a large vocabulary database of 5240 common japanese words. These words were uttered in isolation by one Japanese native male speaker. All utterances were digitized at a 12 kHz sampling rate. The database was then split into a training and a testing set, from which the actual phonetic tokens were extracted. The training tokens were then randomized within each class and the two sets were built by alternating each class to be learned. The phonemes to recognize were the three stop consonants "b", "d" and "g". Each "piece" of speech contained 15 frames, each of them containing 16 Melscale coefficients. The network to recognize these chunks of speech needs from 1 to 15 frames of input for the RBFs (depending on the size of the time window for this layer) and accordingly 15 to 1 frames for the integration layer (to cover the whole input sequence of 15 frames) which finally provides three outputs corresponding to the three phonemes. The only adjustable parameter is the relation of the number of time delays in the RBF layer compared to the number of time delays in the integration layer, which is uncritical to optimize. Figure 3 illustrates the main structure of the BDG-TDRBF.

The bottom layer has 16 inputs per time frame, which are Melscale Coefficients (a logarithmically compressed frequency spectrum). Each of the RBFs in the middle layer (for simplicity only one RBF per class is drawn, the number of RBFs in this layer changes during training, when new prototypes are committed) takes the current and the past 6 frames as an input, resulting in prototypes of dimension 7 $*$ 16 = 112. The responses of these RBFs are integrated by the last layer, which takes values from the previous and the past eight frames and produces the final output, for each of the three phonemes one value per frame.

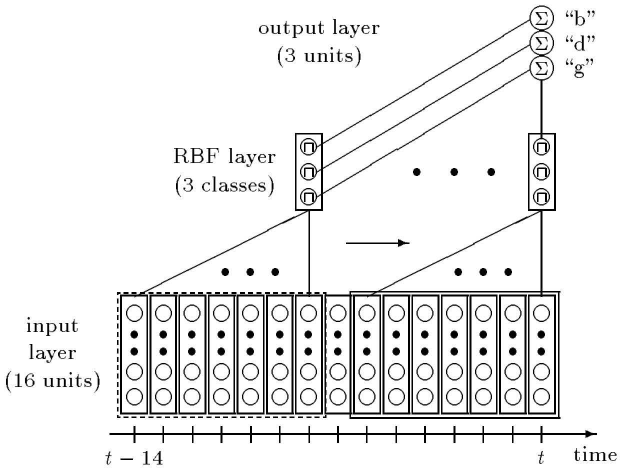

Figure 3: The Structure of the 7(9)–TDRBF used for the BDG–task. The number of RBFs in the second layer grows during training. Here only one RBF per class is drawn.

V. Performance

A. RBF Layer Window Size

Figure 4 shows how the size of the window which is moved over the input signal influences the performance of the whole network. If the number of time delays in the RBF layer is chosen too small, the performance is poor because the network has no opportunity to find features over time. On the other hand, if the number of time delays is chosen too large, the shift invariance over time is lost and the performance degrades also. The optimum performance lies somewhere in between and is easy to spot.

For this specific task, the interesting features are spread out over somewhere between 6 to 9 frames in time. Note that the error–rate for the training data set is a perfect 0% for the interesting networks and the best results for the test data are similar to those achieved with the TDNN mentioned in [7]. All experiments took less than 10 runs over the training data to complete.

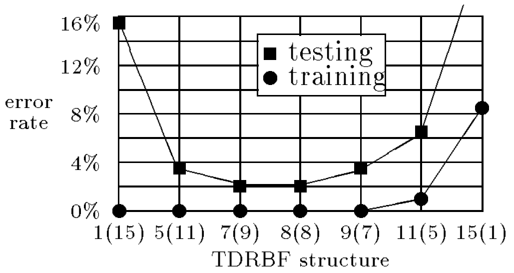

Figure 4: Performance vs. window size

B. *Training the Integration Layer*

For the previous examples the integration layer was used only to sum activations produced by RBFs for different points in time. One obvious experiment was to train these 1.0–weights. To do so, we used common Error Back Propagation training. Because this includes only a very small number of weights (for this example 3 — 45, depending on the structure of the net) it did not take too long to train this part of the networks. For this experiment we used the 7(9)–network from above. The error rate on the training data stayed at 0% and the error on the test data decreased by 0.3% which corresponds to two more patterns classified correctly. This trend held for other experiments as well, the training of the integration layer always improved performance but never substantially. Nevertheless it seems to be an interesting approach to improve performance, worthwhile to investigate further.

VI. Conclusion and Summary

In this paper a Time Delay Radial Basis Function Network (TDRBF) for the recognition of phonemes was presented. The main advantage over comparable approaches recognizing phonemes with Neural Network based systems is the decrease in time needed to train the network and the absence of hard to optimize parameters.

The TDRBF combines advantages of TDNNs, which are the shift invariance and the ability to recognize features in time, and the training speed and ease to use of RBF–based networks.

It was shown that TDRBF perform as well as TDNN on a phoneme recognition task used in [7] (TDRBF: 98.3% vs. TDNN: 98.5% recognition rate). The TDRBF needed less than 10 epochs to finish training, whereas the TDNN needs around at least 50 epochs to converge. In addition training epochs for TDRBF are faster than one epoch for TDNN, especially in the beginning when the commitment of new prototypes has just started.

VII. Acknowledgment

I would like to thank Mark Holler and Simon Tam for the opportunity to work on this project during my time at Intel. I would also like to thank Jay Diamond for many fruitful discussions. And last, but not least I would like to thank Alex Waibel for introducing me to this interesting field and for providing the BDG–database.

References

[1] B. Fritzke: "Growing Cell Structures – a self–organizing network in k dimensions", ANNs II, I. Aleksander & J. Taylor, eds., Amsterdam, pp. 1051–1056

[2] M. Holler, et al: "A 40,000 Patterns/s Classification Coprocessor with Learning Capability", Proc. Hot Chips V, Stanford, August 1993.

[3] J. Moody, C. Darken: "Fast Learning in Networks of locally–tuned processing units", Touretzky, Hinton and Sejnowsky, editors, Connectionist Models Summer School, 1988, pp. 133–143

[4] D. Morgan, P. Scofield: "Neural Networks for Speech Processing", Kluwer Academic Publishers, Boston, 1991, pp. 63–69

[5] D. Reilly, L. Cooper, C. Elbaum: "A Neural Model for Category Learning", Biol. Cybern., vol. 45, 1982, pp. 35–41

[6] E. Singer, R. Lippmann: "A Speech Recognizer using Radial Basis Function Neural Networks in an HMM Framework", Proc. ICASSP, San Francisco, March 1992.

[7] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang: "Phoneme Recognition Using Time–Delay Neural Networks", Proc. ICASSP, Vol 37, No. 3, March 1989