COMPARISON OF RECURRENT NEURAL NETWORK ARCHITECTURES FOR SPEAKER VERIFICATION

David Shrimpton and Brett D. Watson

Department of Electrical and Computer Engineering
University of Queensland, Australia

ABSTRACT - Recurrent Neural Networks (RNN) have shown promise in the area of automatic speech recognition. In this paper we examine the application of RNN architectures to the problem of text dependent automatic speaker identity verification.

INTRODUCTION

A recurrent neural network architecture contains feedback as well as feed forward connections. It is hypothesised that the presence of feedback connections will enable RNNs to perform better than multilayer perceptrons (MLPs) when the input is a dynamical time series, such as in speech processing.

The standard backpropagation algorithm used to train MLPs must be modified in order to train RNNs. There are two techniques for modifying the algorithm: In the first technique the output of a node is calculated at time t by using the outputs at time $t-1$ from the connecting nodes. This circumvents the problem caused by a recurrent connection from the same node. The value on the recurrent connection used in calculating the weighted sum at time t, is the output of the same node from time $t-1$. This technique is referred to as backpropagation in time, or, unfolding of the network in time. It was first proposed by Rumelhart, Hinton & Williams (1986). Variations of this algorithm have been used by Watrous & Shastri (1987) and by Morgan & Scofield (1991), in speech recognition.

The second modified backpropagation algorithm is called recurrent backpropagation (RBP) . It was proposed independently by Pineda (1987 & 1989) and Almeida (1987 & 1988). In RBP the outputs at a particular time instant are calculated by relaxing the network by an iterative numerical method. The process is essentially the solution of a set of simultaneous non-linear equations. The weight changes are then calculated by gradient descent. An efficient method of calculating the weight changes is to form a transposed error propagation network by:

1. Reversing the connections of the original network.
2. Removing the original inputs.
3. Using the error at each node as the new inputs.
4. Replacing the sigmoid transfer function at each node with a linear transfer function which is the value of the derivative of the sigmoid function of the original network.

The new transposed network is also relaxed and the values of the outputs are used in calculating the weight changes.

In RBP, the outputs of the net at a time instant are independent of the outputs at any previous time instant and depend only on the current inputs. Hence a network is not trained on an input sequence, but only on a particular set of input values. If a sequence of input sets is presented to the net, the outputs for each set are independent and will not vary with reordering of the input sets. In this sense, an RNN trained with RBP, offers no advantage over a multilayer perceptron.

In contrast in the backpropagtion through time algorithm, the state of the network at any time instant depends on the states at all previous times. A sequence of inputs, presented in a different order will produce a different output sequence.

In this paper, time dependence was built into RNNs trained with RBP by using the stored outputs of nodes from previous time steps as inputs at each time step.

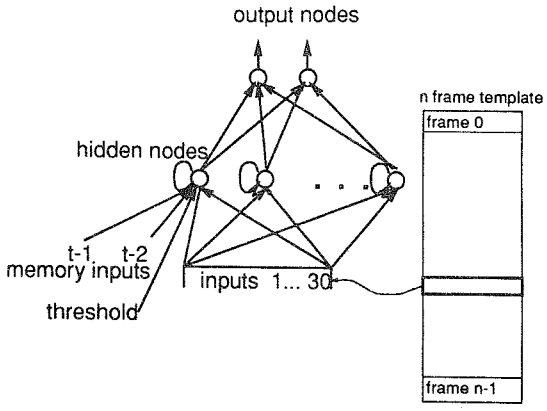

Figure 1: Recurrent Neural Network Architecture

These modified RNNs were used for text dependent speaker identity verification using isolated digit utterances. A comparison was made of the performance of two RNN architectures and two multilayer perceptrons trained on the same data. One of the RNNs and one of the MLPs had time delayed inputs as described below. .

METHOD

Feature Extraction

The speech data consisted of isolated digit utterances. These were sampled at 16 kHz and blocked into 32 msec frames, overlapped by 3/4 of a frame. The features extracted from each Hamming windowed frame were the first 10 LPC cepstra, delta cepstra and delta delta cepstra (30 features). Input values were normalised to between -1 and 1 and then multiplied by a gain of 50. The normalisation procedure was the same as that used by Iso and Watanabe (1990) and was with respect to the mean and range of a feature over all frames in a template according to the formula: $(x - mean)/range$

Network Architecture

Four variations of a 22 node network were investigated. Each network had a 2 node output layer and a 20 node hidden layer. Each output node had trainable connections from each node in the hidden layer and each hidden node had trainable connections from each of the 30 inputs. Each of the 22 nodes had a trainable connection from a constant threshold of value 1. All nodes had sigmoid transfer functions. The nets grouped into two pairs, one pair with a single recurrent connection on each hidden node and the other without. Within each pair one net had memory inputs and the other did not. The memory inputs consisted of the output values of the same node at the four preceding time steps. There were trainable connections on each memory input. The configuration for a network with a single recurrent connection on each hidden node and two memory inputs is shown in figure 1.

Training Method

The training scheme used was similar to that used by Oglesby and Mason (1990) to train a multilayer perceptron for speaker identification. A separate net was trained for each digit for each speaker. The training data consisted of 36 templates from two equal groups: templates from the specified person (6 different templates repeated 3 times) and templates from a rejection group (6 different templates from each of 3 speakers). The templates from the target and rejection sets were alternated in the training set. The supervised learning method was used, with one output node being trained to give a rising exponential output sequence and the other to give a falling exponential sequence, to templates from the target group. The supervisory desired output patterns were reversed for the rejection templates.

Separate nets were trained for each of the ten digits for each of ten speakers. For each target speaker, the rejection set of three speakers was chosen randomly from the set of nine other speakers.

Testing Methods

To calculate false rejection error rates, twelve utterances of each digit, for each of the ten speakers were used. To calculate false acceptance error rates, a total of 237 utterances of each digit, from 34 speakers were used. The 237 utterances were comprised of 18 each from 9 speakers and 3 each from 25 other speakers. The utterances used in testing were different from those used in training.

The networks were tested by applying test templates and measuring over the sequences the mean output of the target node. The error rates were determined by combining the results for all ten digits .

RESULTS

	FA				FR			
SPKR	RNME	MPME	RNNM	MPNM	RNME	MPME	RNNM	MPNM
A	4.2	3.4	0.0	0.0	8.3	33.3	16.7	16.7
B	1.3	0.0	0.0	0.0	25.0	41.7	33.3	50.0
C	0.8	0.4	0.0	0.0	16.7	16.7	33.3	41.7
D	0.0	1.7	0.0	0.0	0.0	0.0	0.0	0.0
E	22.4	12.7	24.5	16.9	16.7	25.0	16.7	16.7
F	5.1	3.8	0.4	2.1	0.0	16.7	0.0	0.0
G	0.8	0.8	0.4	0.4	0.0	0.0	0.0	0.0
H	3.4	5.1	0.8	1.7	33.3	25.0	16.7	25.0
I	10.1	2.5	13.1	8.4	0.0	8.3	0.0	0.0
J	0.4	0.0	0.4	0.8	25.0	25.0	16.7	16.7
AVERAGE	4.8	12.5	4.0	3.0	12.5	19.2	13.3	16.7

Table 1: 4-Way Comparison of Percentage False Acceptance/Rejection

Table 1 shows the results of the first experiment performed. The abbreviations used are:

- SPKR Speaker
- RNME Recurrent Net with Memory
- MPME Multilayer Perceptron with Memory
- RNNM Recurrent Net, No Memory
- MPNM Multilayer Perceptron, No Memory

The table gives a comparison of the false acceptance and false rejection error rates between the ten speakers used and between the four architectures tested. The thresholds used to calculate error rates were fixed for all speakers and networks. A marked variation in performance between the speakers was observed.

Figure 2 is a plot of the false acceptance rate averaged over the ten speakers against the corresponding false rejection rate for the same threshold . The nets were trained for 40 epochs . The error rates were determined by combining results from 10 utterances. There were small differences between the performance of the different architectural variations trialed. The best results were obtained for the multilayer perceptron while the worst were obtained for the multilayer perceptron with memory.

Figure 3 shows a comparison of the error rates for 15, 25 and 40 epochs of training for a single architecture - the recurrent net with memory. A decrease in error rates was obtained with increasing training period and it was unclear from the data whether or not the optimal training point was reached.