Neural Transplant Surgery: An Approach to Pre-training Recurrent Networks

Peter Vamplew and Anthony Adams
Artificial Neural Network Research Group
Computer Science, University of Tasmania
P.Vamplew@cs.utas.edu.au, A.Adams@cs.utas.edu.au

Abstract

Partially-recurrent networks have advantages over strictly feed-forward networks for certain spatiotemporal pattern classification or prediction tasks. However networks involving recurrent links are generally more difficult to train than their non-recurrent counterparts. In this paper we demonstrate that the costs of training a recurrent network can be greatly reduced by initialising the network prior to training with weights 'transplanted' from a non-recurrent architecture.

Introduction

The approaches taken to adapting feed-forward networks to temporal processing can be divided into two main categories: non-recurrent and recurrent. Both of these approaches make use of fixed time-delays on the connections within the network to provide a means of storing temporal information. However in non-recurrent networks all such connections feed into higher layers within the network, whereas nodes in a recurrent network can also have time-delayed links to their own or lower levels. Non-recurrent architectures are generally faster to train than recurrent networks, but have the disadvantages of only storing a finite amount of context, and of having a potentially larger number of free parameters which can adversely affect their generalisability.

Non-recurrent networks

Non-recurrent temporal networks use time-delayed connections to higher levels to store information about previous inputs to the network. The simplest implementation of this concept (which we will label an input windowed network) has such time delays only on the connections between its input and hidden layer nodes (see Figure 1) Each input node is connected to each hidden node by several connections, each with an independent weight and a different time delay (see Figure 1). Usually these time delays will vary by only a single time-step. For example, each input may be connected to each hidden node by three links with time delays of 0, 1 and 2 steps. At each time step the network is evaluated exactly as a normal static network would be. The effect is to provide the network at each stage with the current input, plus a series of previous inputs thereby giving it the temporal information required for the task.

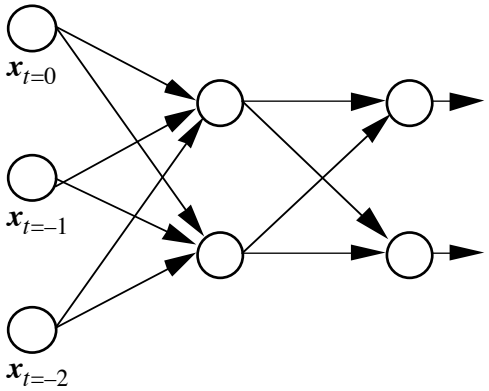

Figure 1. Input windowed network with three time frames, two hidden nodes and two outputs. The inputs at each time frame are vectors.

This style of network can be trained using standard backpropagation and because it essentially ignores the temporal aspects of the data by expanding them into spatial aspects, can generally be trained relatively quickly. However this non-recurrent approach has two weaknesses.

First the temporal context processed by the network is strictly limited to the size of the input window, and hence unable to deal with patterns longer than this window (and in particular with potentially infinite sequences). Second the complete interconnection between the input and hidden layers means the network will have a large number of free parameters when trained on long temporal sequences, which may have a negative effect on both the training time and the generalisation capabilities of the network. One approach to remedying the latter problem is the Time Delay Neural Network [2]. This takes advantage of the fact that often the same features are present at several different times in the input sequence, and therefore the same feature detectors can be applied at different time delays. In a TDNN some of the links between the input and hidden layers share weights, and hence the number of free parameters is reduced (see Figure 2).

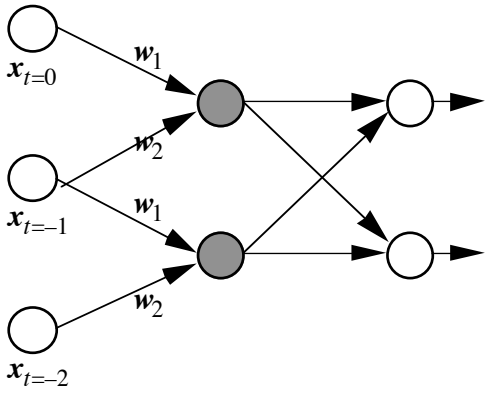

Figure 2. Time Delay Neural Network. In this example each hidden node only sees two time frames and weights are shared.

Recurrent networks

In a recurrent network nodes can have time-delayed links to any other node within the network. For this paper we will consider a limited class of networks where such recurrent links are restricted to a single hidden layer, with each node in that layer recurrently linked to every other node (including itself). Such a network can be trained using a modified version of backpropagation called backpropagation-through-time (BPTT). Note that other training algorithms such as real-time recurrent learning can also be used [3],[5].

The basis of the BPTT algorithm is the 'unrolling' of the recurrent network through several time frames to form a non-recurrent network whose behavious is identical over that finite period of time. The error can then be back-propagated through this non-recurrent architecture, with each weight being updated with the sum of the change calculated for it at each time step. For a more formal definition of the algorithm see [4].

The major problem with this training method is that outputs and relevant inputs may be presented to the network many time-frames apart. This means that the error from the output will effectively be passed through many layers prior to reaching the relevant activations of the input layer, and hence will be much dissipated by the noise inherent in this process. This makes the learning of local spatial features by a recurrent network extremely difficult, and sometimes impossible (particularly when dealing with long temporal sequences).

Neural transplants

Training a recurrent network can be likened to a "chicken-and-egg' problem. It is difficult for the network to learn the temporal aspects of a sequence unless the correct spatial features are being extracted. However at the same time it is difficult for the network to learn what spatial features are important until the temporal aspects have been at least partially learnt. In comparison, an input windowed network finds the task of forming spatial feature detectors much easier as inputs are always presented at the same time as the output they are relevant to (assuming the window is of sufficient size).

The basis of our proposed training scheme is to develop suitable feature detectors within an input windowed network, and then to 'transplant' these detectors into a recurrent network which can then be trained using BPTT. Providing the recurrent net with some knowledge of the spatial features in this manner should aid in the subsequent training to learn the temporal aspects.

The major issue which has to be addressed is how to disconnect the extra inputs used for input windowing without losing the local feature extraction capabilities of the network. The solution we have used is a specialised case of a TDNN architecture in which each hidden unit is connected to only a single time-frame of input information. Due to this limited connection these 'temporally focused' units are forced to learn to extract the important spatial features for each time-frame (see Figure 3).

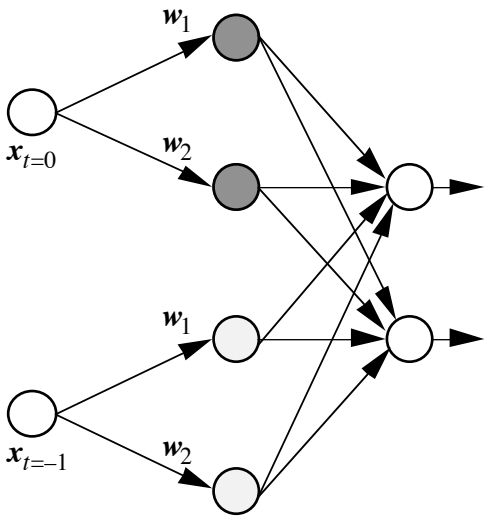

Figure 3. Temporally focused input windowed network. Each pair of hidden nodes has the same input weights but different output weights.

This network is trained using backpropagation until it achieves its maximum level of classification. At this point we convert the network into a recurrent architecture. The hidden units from the TDNN (which we will refer to as spatial units) can be incorporated directly into our new hidden layer. We freeze the input weights of these units, and do not add any recurrent inputs. Instead we add several recurrent units to the hidden layer, which are fully connected to the input layer, and have recurrent links from all other recurrent and spatial units. For the examples given in this paper we have found it beneficial to initialise the weights on all inputs to recurrent units to zero, with the exception of the recurrent links from the spatial units which were randomly initialised. We have also chosen the

number of recurrent units so that we can directly transplant the hidden-layer to output weights from the TDNN, as this was found to have extremely positive effects on training times (see Figure 4).

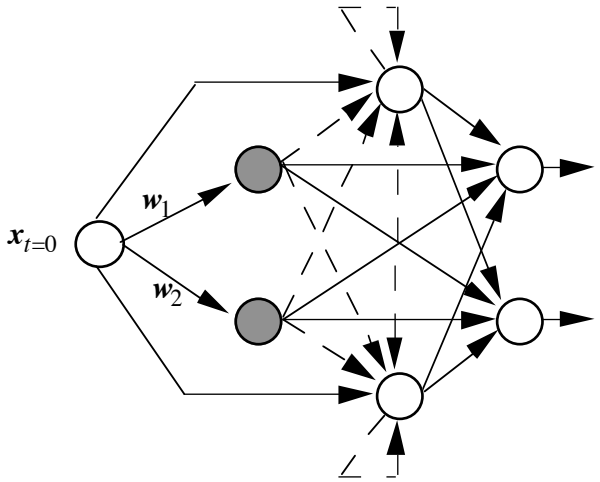

Figure 4. The recurrent network with transplanted weights (solid lines represent non-recurrent links, dashed lines are recurrent links).

Once the network has been initialised with these transplanted weights it is trained using BPTT, modified only to the extent that the spatial units' input weights are not changed during training. Freezing these weights ensured that they didn't 'wander off' whilst the network was adapting the recurrent weights to learn the temporal aspects of the problem. It also had the side benefit of allowing the caching of the spatial units outputs for each training example, thereby reducing the amount of computation involved during the forward pass of training.

Data sets and results

This transplant method has so far been tested on two simplified motion-detection problems, both of which were outlined in [3]. In both the network is presented with eight input values per time-frame, one of which is turned on whilst the others are off. The location of this active cell is moved a single step randomly to the right or left at each time-frame, and the task of the network is to identify the direction of this motion. For the first problem the task is made easier by not allowing wraparound from one end of the eight cells to the other, so that upon reaching the first or eighth cell the direction of movement will always be reversed. For the second problem such wraparound is allowed.

In order to compare the different architectures and training methods we have adopted two conventions proposed in [1]. A straight comparison of the number of epochs or pattern presentations between training methods would be misleading, as the computational costs of each presentation vary between methods. Instead we measure the number of connection crossings, where a connection crossing is recorded every time a weight is crossed on either the forward or backward pass of the training method. It was also found that on occasions the network trained using standard BPTT would settle into a location in weight-space where the error would either cease to fall or even rise. To counter such situations we set a time-out limit of 50,000 pattern presentations. Any network reaching this time-out would be reinitialised with new random weights and training recommenced. The time spent in reaching the time-out was included in the total time required to train that network.

Method	Input windowed	BPTT	Transplant		
			Total	TDNN phase	BPTT phase
Learning rate	0.6	0.4		0.4	0.4
Pattern presentations	530	2410	2270	1200	1070
CCs per presentation	152	172		52	94 + 1800 for caching
Total CCs	80560	414520	164780	62400	102380

Table 1: Average training time over ten runs for each training method, in terms of pattern presentations and connection crossings (CCs), for the motion detection without wraparound

Method	Input windowed	BPTT	Transplant		
			Total	TDNN phase	BPTT phase
Learning rate	0.6	0.2		0.15	0.2
Pattern presentations	600	19440	10420	3240	6980
CCs per presentation	152	172		52	94 + 1800 for caching
Total CCs	91200	3343680	835760	177840	657920

Table 2: Average training time over ten runs for each training method, in terms of pattern presentations and connection crossings (CCs), for the motion detection with wraparound

The averaged results over ten separate training runs of the input windowed, BPTT and transplant method (the latter broken down into the two phases of its training) are recorded in Tables 1 and 2. These results are the best obtained for each method after experimentation to find the appropriate learning rate. As can be seen the input windowed network trains much faster than either of the recurrent networks for both problems. However whilst the transplanted network's training time is much longer than the non-recurrent network, it is significantly faster that that of the BPTT network, particularly on the more complex wraparound detection problem.

Conclusion

The results obtained on the two motion detection tasks show that the training time of a recurrent network can be greatly reduced by initialising the network with weights from a non-recurrent network. It is expected that the method of transplanting the input weights should scale well to more difficult tasks. However the current method of initialising the recurrent and output weights is dependent on the user's knowledge of the problem task. In order to extend the transplant method to real-world problems it will be neccesary to develop a more generalised and automated method of initialising these portions of the network.

References

[1] Fahlman, SE. (1988) Faster-Learning Variations on Backpropagation: An Empirical Study, in *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann.

[2] Lang, K, Waibel, A & Hinton, G (1990) A Time-Delay Neural Network Architecture for Isolated Word Recognition in *Neural Networks*, **3**, 23–43.

[3] Robinson, A. (1989) Dynamic Error Propagation Networks, PhD Thesis, Engineering Department, Cambridge University.

[4] Werbos, P. (1990) Backpropagation Through Time: What It Is and How to Do It" in *Proceedings of the IEEE*, **78**(10), October 1990, 1550–1560

[5] Williams, R & Zipser, D. (1989) A Learning Algorithm for Continually Running Fully Recurrent Neural Networks" in *Neural Computation*, **1**, 270–280