ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 165-170

The Filtered Associative Network

L.S. Smith and K. Swingler
Centre for Cognitive and Computational Neuroscience
Department of Computing Science and Psychology
University of Stirling Stirling FK9 4LA, Scotland.

Abstract. A simple sequence recognition network which learns quickly is presented. This 2-layer network is modular, with the first layer being a new type of network, replicated, and the second a form of bidirectional associative memory. Some early results are presented.

1. Introduction.

Networks which process time-varying signals are divided into three types in [1], namely sequence recognition networks, sequence reproduction networks, and temporal association networks. The network described here is in the first of these classes. Its input is a time-varying signal, and its output is the network's classification of that signal. This puts it in the same class as backpropagated networks which use a tapped delay line for translating input at different times into input to different units (of which there are many in the literature), and TDNN [2]. Both of these networks place equal emphasis on the content of each element of the input sequence, and cannot come to any conclusions about the sequence class until the whole sequence has been received. Further, since both of these approaches are based on the backpropagated delta rule, which is a gradient descent algorithm, they are slow to learn.

The approach presented here is different. In particular, learning is not a gradient descent operation. Single-shot learning is possible. The filtered activation network itself is a two layer network, made up of two types of network, a simple single layer filter network (SLFN), and a network for combining the results of the SLFNs. There are usually a number of the former networks making up the first layer of the whole filtered activation network (FAN), but only one of the latter, making up the second layer of the FAN. A detailed description of the whole network is given in [3]. All adaptation takes place in the SLFNs: the second layer network (which is a simple autoassociative form of BAM [4]) interprets the outputs of the first layer networks.

2. The simple single layer filter network (SLFN).

The SLFN recognises a sequence of values from a small vocabulary as being in one of a number of classes. The network has a set of input nodes $I = \{I_0, I_1, ..., I_{m-1}\}$, and a set of output nodes, $H = \{H_0, H_1, ..., H_{n-1}\}$. These are connected together by weights W_{hi}, where $0 \leq h < n$ and $0 \leq i < m$, to form a single layer network. There are inhibitory weights, C_{hk} between the output nodes, where $h \neq k$. In addition there are synaptic filters P_{hi}, with values between 0 and 1, one per W_{hi}. The SLFN is illustrated in figure 1. Local coding is used for both input and output, so that in both in training and

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 165-170

test, exactly one input unit at a time and one output unit will be on. The SLFN can process input sequences of arbitrary length, with each element having one of m (=3 in figure 1) possible values, and can classify such sequences into n (=2 in figure 1) classes.

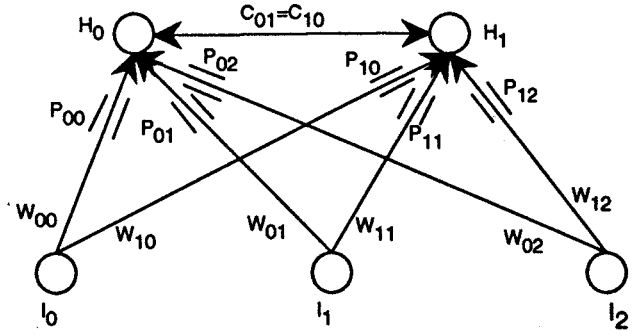

Fig.1. A 3-input 2-output simple single layer filter network.

2.1 Simple Training.

For training, all the W_{hi} are initialised to 0. We assume for now there is exactly one training sequence for each class. For each sequence, the output unit is chosen at the start, and the sequence presented. To learn to classify the sequence (2,1) as class h, the first element is presented to unit I_2, and the weight between it and the output unit selected, W_{h2}, adjusted. The next element is presented to unit I_1, and the weight W_{h1} adjusted. This is repeated for each sequence.

Weight adjustment takes the form of a Hebbian increment, but set up in such a way that earlier elements have more influence. The actual change in weight is

$$\Delta W_{hi} = (1-W_{hi}).I_i.\tau^t \qquad 1$$

where $0<\tau\leq 1$, and t indexes the position in the sequence (starting at 0). The $(1-W_{hi})$ term may be omitted: its effect is to ensure that the weight stays between 0 and 1. Note that the filters P_{hi} are not used in this simple training method. The effect is that input units corresponding to elements early in the sequence have large weights to their classification output, and input units corresponding to elements later in the sequence have smaller weights. On test, when a sequence is presented, the output unit which receives the highest activation is the one whose training sequence most resembles the test sequence, with more importance being given to earlier elements in the sequence.

2.2. Simple Recall.

On starting the recognition of a sequence, the weight values are copied into their

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 165-170

filters. The weights play no further part in the recognition process: they are only used as filter initialisation values. The filters control the amount of activation that can pass through them. This is controlled in two ways: firstly, the total amount of activation that can pass through the filter is defined by the initial value of the filter, and secondly, the amount of activation that can pass through the filter at time t is limited to τ^t. When a test pattern is presented, for input I_i at time step t, the activity that will be sent to an output unit H_h is the maximum of what is left in filter P_{hi} (which will be the original weight W_{hi} if input I_i has not been presented before in this pattern), and τ^t. P_{hi} is decremented by this amount.

Thus (ignoring the factor of $(1-W_{hi})$ in equation 1) the activity at unit H_h for a test pattern which is the same as the training pattern will be $\Sigma_t \tau^t$. For a test pattern which differs from the training pattern, there will be a contribution of τ^t from the t'th element in the test pattern if it is the same as an earlier element in the training pattern, or a contribution of τ^j (for some j>t) if the test pattern has an element the same as a later element in the training sequence, and no contribution at all when the element in the test pattern does not occur in the training pattern, or when the allowance in the filter has been used up. Thus, only the original training pattern, or a pattern which has the initial training pattern as a subsequence at its start can achieve the maximum amount of activation.

The weights C_{hk} are used to emphasise the rather small differences between activations at the units H_h which occur when sequences are similar, in the style of a winner-take-all network. These have been found (empirically) to work well when set to -1/n (n is the number of output units). The excitation from the inputs and the inhibition from the other output units is combined in a nonlinear fashion, so that

$$H_h(t) = H_h(t-1) + (1-H_h(t-1)).Ex(h,t) + H_h(t-1).In(h,t) \qquad 2$$

where Ex(h,t) is the excitatory input to output unit h at time step t, and In(h,t) is the inhibitory input to unit h at time step t. Equation 2 also forces the values of $H_h(t)$ to be between 0 and 1.

2.3. Training on multiple examples.

The algorithm for training described in section 2.1 can perform only single shot training. However, it is often the case that there are a number of examples of a class, and the network is required to find a suitable internal representation for the class in the absence of an ideal case. The algorithm described here reduces to that of section 2.1 when there is only one example for each class; otherwise, it is a gradual learning rule (though still not gradient descent), one which uses the filters to record differences between different example patterns. The algorithm is as follows:

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 165-170

```
Set all weights to 0.                                                    i
For each class to be learned
{
    Choose the output unit to be asserted (H_c)
    For each sequence in this class
    {
    Initialise the filter values for this class (P_ci := W_ci for all i)
        For t := 0 to (length of sequence - 1)
        {
            Select t'th element in this example
            /* assume t'th element activates unit I_r */
            If (P_cr = 0) then
              ΔW_cr := η.τ^t.(1-W_cr)                                    ii
            else if (W_cr < τ^t) then
              P_cr := P_cr - τ^t                                         iii
            else P_cr := P_cr - W_cr ;                                   iv
            if (P_cr < 0) then
            { ΔW_cr := η.τ^t.|P_cr|.(1-W_cr)                             v
              P_cr := 0 }
        } /* single sequence loop */
        For each input unit i
            If (P_ci > 0) then ΔW_ci := -η. W_ci .P_ci ;                 vi
    } /* single class loop */
} /* class loop */
```

Steps i and ii are very much the same as in section 2.1: however, a learning rate parameter, η, has been introduced. Steps iii and iv reduce the filter by an amount equal to the maximal amount of activation that the filter could let through at time step t. Then, if this would leave the filter negative, the filter is reset to 0, and the weight increased (step v). Lastly, if the filter is left at the end with some positive value (that is, if the filter has not been entirely used up in the processing of a sequence), the weight is reduced (step vi). Note that increments in weight have a multiplier of (1-W) and decrements a multiplier of W, so that weights stay between 0 and 1.

The effect of the learning algorithm is that the SLFN is responsive to a range of similar sequences for each class. The choice of η (values of around 0.5 have been found useful) will depend on the similarity of the sequences in each class.

3. The full filtered activation network (FAN).

The complete FAN network consists of a number of SLFNs followed by a nonadaptive network whose function is to select the majority output of the SLFNs. Note that training of the FAN net is simply the same as training of each SLFN. The

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 165-170

only difference is in recall.

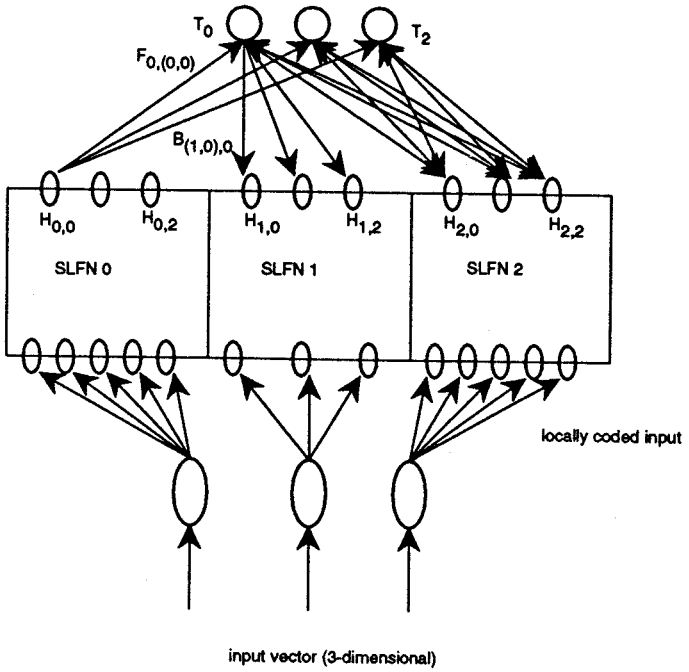

Fig. 2.A complete FAN network. It has three SLFN components as the first layer, each with three classifications. The number of input units may be different in different SLFNs, but the number of final output units must be the same as the number of output units on each SLFN.

The top layer network has weights $F_{x,(i,j)}$ which are set so that each unit $H_{i,j}$ (i.e. the j'th unit in the i'th SLFN) excites unit T_j, and either has no effect or inhibits units T_k ($k \neq j$), and weights $B_{(i,j),x}$ set so that final output unit x excites units $H_{(i,x)}$ (for all i), and inhibits units $H_{(i,y)}$ for $y \neq x$. The outputs of the units T come to reflect those of the units H. The updating of the output units takes place concurrently with the updating of the units H, so that each SLFN can affect other SLFNs during recall.

4. Results.

The network has been tested both on some synthetic examples, and on a real dataset generated from speech data. The synthetic data set consisted of sequences of length 20 of random integers between 0 and 15, on 8 input channels (i.e. 8 SLFNs were used in the first layer). Even with 98% replacement of numbers by 0's, all the sequences were correctly identified. With replacement of numbers by random different numbers, 100% correct identification took place with 78% noise. This performance is due to the

large differences between sequences, and to the use of 8 channels, each trying to identify the same sequence. Using only one channel, 100% performance was obtained with 68% replacement of numbers by 0's, or with 20% replacement of numbers by random other numbers.

The system was also tested with synthetic noisy training data. Sequences were generated as before, but each sequence was degraded with random noise to produce 6 noisy training examples. At up to 30% noise in the training samples, the net succeeded in correctly identifying all the original (undegraded) training sequences.

The net was tested on digitised bandpassed speech signals. These came from the following vowel sounds (separated by hand from their enclosing consonants), as pronounced by a native speaker of English: heed, hid, head, hod, who'd, hud, heard, hoard, hood, had, and hard. These were bandpassed using 8 overlapping filters, and then the values compressed into the range 0-19, and sequences of up to 20 elements long produced. There were 10 examples of each vowel. This was not a proper experiment in speech recognition, but an experiment in using real data from equipment which was to hand. The results were that the system succeeded in recognising 95.5 of the vowels it was trained with. Splitting the data set into two parts (of 5 examples each), and training on the one set, and looking for generalisation on the other, the net scored 52%. In both cases, 6 epochs were used. Using backpropagation (using a 160:10:11 network for 700 epochs), a network succeeded in recognising 92% when the whole dataset was used for training. This fell to 62% (using a 160:8:11 network for 200 epochs) when the data set was split.

5. Conclusions.

The network has been shown to work. Its precise behaviour has not yet been thoroughly characterised, particularly for large numbers of input units, and this is an area needing further research. Clearly, the network can only deal with sequences of limited length, and the exact limitation depends on the accuracy with which weights and filters store their values. From a software viewpoint, this presents few problems, although a true hardware implementation would need to consider these issues. The network needs to know the start of each sequence, and this could be a problem. Nonetheless, this is a simple modular network, and one which can learn quickly.

Acknowledgements: British Telecom funded K. Swingler on this work under their CONNEX 2 research programme.

References

[1] Hertz J., Krogh A., Palmer R.G., *Introduction to the theory of neural computation*, Addison Wesley, 1991.
[2] Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K., *Phoneme recognition using time-delay neural networks*, IEEE Trans ASSP, 37, 328-339, 1989.
[3] Patent Application, *Filtered Activation Network*, filed by British Telecom, 16 July 1992.
[4] Kosko B., *Bidirectional Associative Memories*, IEEE Trans SMC, SMC-18, 42-60.