Adaptive Response Function Neurons

R.B. Ollington and P. W. Vamplew
School of Computing, University of Tasmania, GPO Box 252-100, Hobart 7001, Tasmania, Australia
{Robert.Ollington, Peter.Vamplew}@utas.edu.au

Abstract

Biological neurons that show a locally tuned response to input may arise from the network topology of interneurons in the system. By considering such a sub-network, a learning algorithm is developed for the on-line learning of the centre, width and shape of locally tuned response functions. The response function for each input is trained independently, resulting in a very good fit for the presented data. Two example networks utilising these neurons were considered. The first was a completely supervised network while the second utilised a Kohonen-like training scheme for the hidden layer. The adaptive response function neurons (ARFNs) were able to achieve excellent class separation while maintaining good generalisation with relatively few neurons.

1. Introduction

Biological neurons are typically modelled using a linear response function. The electrical potential of a neuron is calculated as the weighted sum of its inputs, with the weights representing the synaptic efficiencies of the input connections. While the biological system is considerably more complex than this it can be assumed that the response functions of the majority of neurons are at least monotonic.

Biological neurons do exist, however, that respond in a selective way to input. This may be due to either the physical properties of some sensory neurons or to the topology of the network containing the locally responsive neuron [1]

Certain classes of artificial neural network also contain neurons that are locally responsive to certain input levels. These include self-organising maps (SOMs)[2] and radial basis function (RBF) networks [1]. The output of a SOM neuron is typically a distance measure from the supplied input to a stored exemplar, while the response function of RBF neurons is typically a gaussian.

The outputs of SOM neurons are compared to find a winning neuron, hence the critical parameters for a SOM are just the stored exemplars, ie. the centres of the response functions. The locations of the centres of the response functions are initially random. For each example presentation the winning neuron (that neuron whose exemplar is closest to the input) and some surrounding "neighbourhood" of neurons have their exemplars shifted towards the given input pattern.

The outputs of RBF neurons, on the other hand, are passed onto a second layer of neurons, which are typically trained using a supervised gradient descent rule. Thus both the centres and widths of the response functions of RBF neurons are important. Many techniques have been proposed for determining appropriate centres and widths of the basis functions of RBF networks.

One solution is to find RBF centres by applying a clustering algorithm such as K-means and determining appropriate widths using techniques such as "P-nearest neighbours"[1, 3]. In order to produce more compact RBF networks Leonardis and Bischof[4] propose a method of pruning based on the minimum description length (MDL) principle. None of these training methods can be employed on-line.

The resource-allocating network (RAN) of Platt[5] adds neurons if the network error is high and adjusts the centres of existing neurons if the error is low. The width of the gaussian response functions is reduced as new neurons are added. While the RAN can be trained on-line it has the disadvantage of having an indeterminate network size.

By considering a biologically plausible sub-network for the formation of locally-tuned neurons, a training method was developed that can be used on-line. The training algorithm independently adjusts the centres, widths and shapes of locally-tuned response functions for each input to the neuron.

2. Adaptive Response Function Neurons

Within the field of artificial neural networks (ANNs) a frequency model of biological neurons is commonly used. The output of such a neuron represents the firing frequency of the neuron. The activation function is typically a sigmoid and the input response is usually linear with individually adjustable weights representing synaptic efficiencies. This model will be used to develop the adaptive response function neuron (ARFN).

Consider a cortical neuron that receives input from both an exitatory and an inhibitory interneuron. Now suppose that each of these interneurons is excited by a common cortical input (see Figure 1). With appropriate choices for thresholds the output neuron, which we shall call the

ARFN, will have a guassian like response to the cortical input. Equation 1 gives the input response function, R(x), for the ARFN.

$$R(x) = \frac{s_e}{1+e^{g_e x - t_e}} + \frac{s_e}{1+e^{g_i x - t_i}} \qquad (1)$$

where s_e and s_i are the synaptic efficiencies of the interneuron→ARFN connections for the excitatory and inhibitory interneurons respectively; g_e and g_r are the synaptic efficiencies of the input→interneuron connections; and t_e and t_i are the synaptic efficiencies of the threshold→interneuron connections.

In Figure 1 we see that there are six synaptic connections that could be modified. Two of these (g_i and g_e) are from the input to the two interneurons. Modifying the synaptic efficiencies of these neurons would effectively modify the slopes (gain) of the corresponding sigmoid activation functions of the interneurons. These could potentially be modified independently to create an asymmetrical output response function.

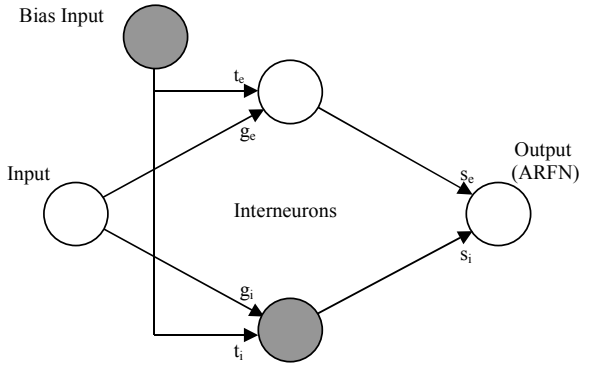

Figure 1: A neuron arrangement to implement a guassian-like response function. White neurons are excitatory, grey neurons are inhibitory. t_e, t_i, g_e, g_i, s_e and s_i are the synaptic efficiencies of the indicated connections.

Another two synapses (t_e and t_i) occur between the inhibitory bias input and the interneurons. Modifying the synaptic efficiencies of these neurons would alter the threshold of the two sigmoids. This would adjust the centre and width of the output response function.

Finally the synaptic connections (s_e and s_i) between the interneurons and the ARFN could be modified. It is not clear that modification of these synapses would perform any useful function, therefore these synapses have been ignored in the development of the ARFN and will be omitted in the following discussion.

Figure 1 shows the network topology for a single input ARFN. For an ARFN with multiple inputs, each input has its own pair of interneurons, which allow independent response functions to develop. The inhibitory bias input is shared by all interneurons.

2.1. Training Thresholds

Training the thresholds of the interneurons is straightforward. In the case of the excitatory neuron, if the response of the interneuron is high the threshold should be trained up, if it is low it should be trained down. The opposite should occur for the inhibitory interneuron. These are Hebbian[6] learning rules as shown in equations (2) and (3) below.

$$\Delta t_e = \eta_t (r_e - \alpha) \qquad (2)$$
$$\Delta t_i = \eta_t (\alpha - (1-r_i)) \qquad (3)$$

where η_t is the training rate for thresholds; r_e and r_i are the outputs of the excitatory and inhibitory interneurons respectively; and α is a parameter determining the equilibrium position for the training rule.

In terms of a possible biological implementation, it is assumed that if the neuron is to be trained then the bias input is set high and if the neuron is not to be trained then the bias input is set low. This could be achieved by feedback connections after some form of competition has determined those neurons to be trained.

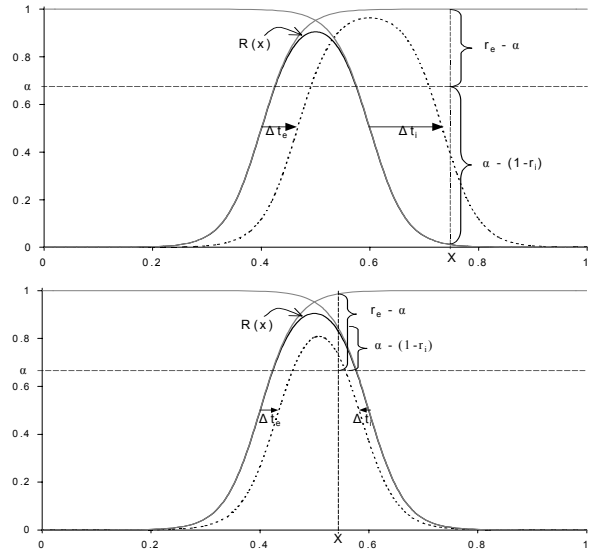

Figure 2: Effect of threshold training on the width and centre of the response function, R(x), for α>0.5 and input X. r_e is the response function of the excitatory interneuron, $1-r_i$ is the inverse of the response function of the inhibitory interneuron. (a) For the input X shown, $\alpha-(1-r_i) > r_e-\alpha > 0$, hence the thresholds of the interneurons both increase resulting in the expansion of R(x) (dotted line). (b) For the input X shown, $\alpha-(1-r_i) < 0 < r_e-\alpha$, hence the thresholds of the interneurons move towards the input resulting in the contraction of R(x) (dotted line).

If the parameter α in equations (2) and (3) above is greater than 0.5, then as well as adjusting the centre of the response function, the width will also be adjusted in an intuitive way. As shown in Figure 2, if the output of both the excitatory and inhibitory interneuron is either high or low then the response function will expand as well as moving the centre of the response function towards the input value. If the output of the excitatory interneuron is high and the output of the inhibitory neuron is low the response function will contract towards the input. If the network is consistently trained on a small range of inputs the width of the response function will be small whereas if the input range is wide then the width of the response function will be large.

Unfortunately as the width of the response function decreases the amplitude will also reduce until the excitatory and inhibitory sigmoids completely cancel each other. To avoid this the gains of the sigmoids must increase as the width of the response function decreases.

2.2. Training Gains

If the output of the excitatory interneuron is less than 0.5 then decreasing the gain of the sigmoid (decreasing the synaptic efficiency of the input→interneuron connection) will increase the response of the neuron to that input value. Similarly if the output is greater than 0.5 then increasing the gain of the sigmoid will increase the response of the neuron to that input value. Since we want the output of the neuron to increase for a particular input value after training we could devise learning rules for the input→interneuron synapses as shown in equations (4) and (5) below.

$$\Delta g_e = \eta_g (r_e - 0.5) \quad (4)$$
$$\Delta g_i = \eta_g ((1-r_i) - 0.5) \quad (5)$$

where η_g is the training rate for gain; and g_e and g_i are the synaptic weights of the gain connections for the excitatory and inhibitory interneurons respectively.

Unfortunately these intuitive rules do not produce desirable behaviour. If these rules are used the excitatory interneuron places too much importance on outlying high inputs and vice versa for the inhibitory interneuron. The modified rules in equations (6) and (7) overcome this problem.

$$\Delta g_e = \eta_g (r_e - 0.5) (\beta - r_e) \quad (6)$$
$$\Delta g_i = \eta_g ((1-r_i) - 0.5) (\beta - (1-r_i)) \quad (7)$$

where $\beta > 0.5$.

Together with the threshold training described above these rules were found to produce a surprisingly good fit to the presented data.

3. Results

The ARFN was first tested using a supervised learning scheme for classification problems. One ARFN was created for each output category and the appropriate ARFN was trained for each input example. The values chosen for α and β were 0.9 and 0.95 respectively.

The network was trained to differentiate three iris species from four measurements of the adult plant[7, 8]. The data set consisted of 150 samples, 50 for each category. 50% of the data was used as a training set and 50% was used as a test set. The supervised ARF network achieved a mean accuracy of 96.0% on the training data and 93.2% on the test data.

To help visualise the response functions the ARFN was also trained on the entire data set and the resultant response functions were plotted along with the frequency distribution of the input data for each category. The response functions for one category are shown in Figure 3.

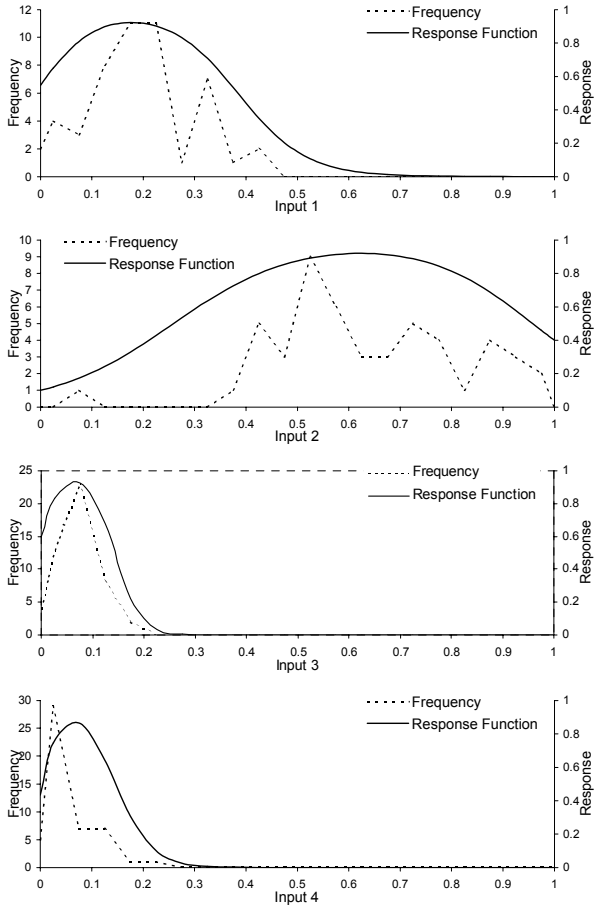

Figure 3. The response functions for each input of an ARFN trained on category one of the iris data set. The bin width of the frequency distributions is 0.05 units.

These figures clearly show the potential of ARFNs for data classification. However a single layer, supervised training model is unlikely to produce acceptable results for more complicated data sets.

3.1. Self Organisation

To improve performance on more complex data an additional output layer was added to the network. The ARF layer was now trained using a Kohonen[2] training scheme with fixed neighbourhood size to determine those neurons to be trained. After training the ARF layer the output layer was trained using gradient descent. This was compared to a similar network using fixed-width gaussian response functions in the hidden layer.

Figure 4 shows the response functions learned by three neurons that were trained on the iris data using the self-organising training scheme.

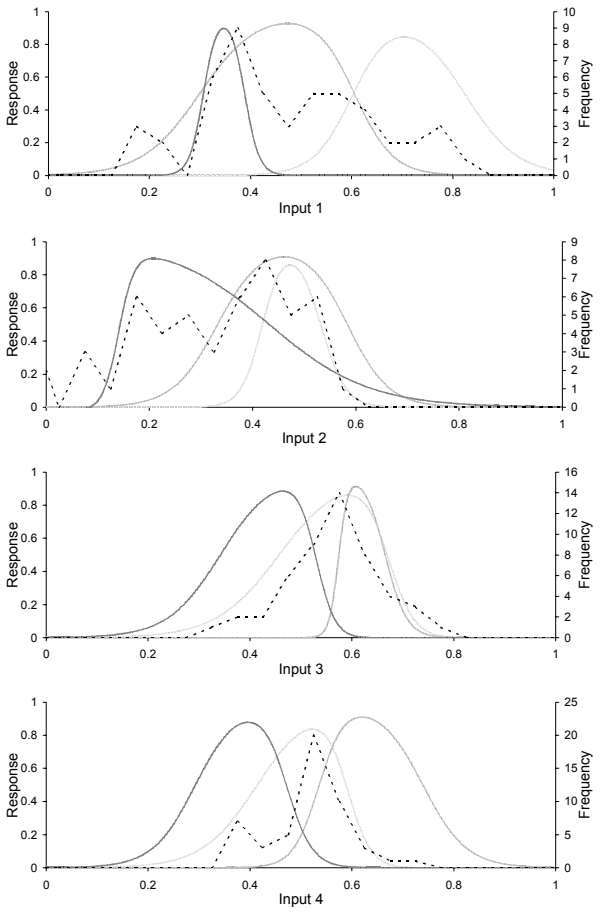

Figure 4. The response functions for three ARFNs whose output was highly correlated with category two of the iris data set. The frequency distributions (dashed line) for category two are also shown with bin widths of 0.05 units.

A variety of response function shapes and sizes are learnt by the network. In particular note the sharp cut-offs learnt by some neurons. In theory this would allow for greater separability of classes without the need for large numbers of finely tuned neurons. However, both the ARF network and the fixed-width network performed well on this relatively simple data set with no significant differences between them.

The networks were then tested on a more difficult data set. The weed data set uses seven attributes of weed seeds to identify one of ten possible weed species[9]. This is a small data set of only 398 examples, hence generalisation is typically poor for this data. Figure 5 shows the performance of each network on the weed data set.

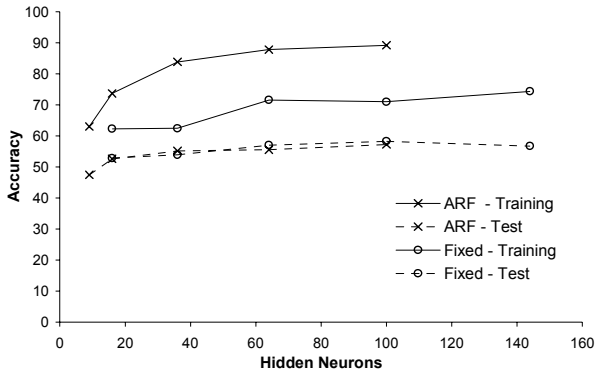

Figure 5. The performance of ARFNs and fixed-width guassian neurons on the weed data set. Results for the training and test data are shown.

The ARF network is considerably more accurate on the training data although both networks show a similar ability to generalise to the test data. These results show that fewer ARFNs are required to achieve the same level of class separation on the test data, without any loss of ability to generalise to the training data.

4. Conclusion

The adaptive response function neuron (ARFN) presented is able to achieve a better fit to the presented data than a neuron using a fixed-width gaussian response function. A network of these neurons, trained using a Kohonen training scheme was able to learn sharp distinctions in the data without requiring large numbers of finely tuned neurons. Unlike many methods for adjusting response function widths, ARFNs may be continually updated on-line and may learn asymmetrically shaped response functions.

Aside from practical applications, ARFNs also provide some biological justification for other networks using local response functions. The ARFN is not a model of any particular biological system. However it is certainly

possible, given the neuron types and numbers available, that such neurons could exist in the neocortex or archicortex. Using only simple Hebbian-like training rules, ARFNs are able to adapt the width, shape and centres of locally-tuned response functions. This suggests that adaptive locally-tuned neurons are not only biologically plausible but also highly likely to occur naturally.

ARFNs as presented should be compatible with almost any training method and the Kohonen method is presented purely by way of example. Further research will need to be carried out to determine the best method for training ARFNs.

Acknowledgments

Data sets were obtained from the UCI Repository of machine learning databases[10].

References

[1] J. Moody and C. J. Darken, Fast Learning in Networks of Locally-Tuned Processing Units, *Neural Computation*, Vol. 1, pp. 281-294, 1989

[2] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, 1995

[3] L. Bruzzone and D. F. Prieto, A Technique for the Selection of Kernel-Function Parameters in RBF Neural Networks for Classification of Remote-Sensing Images, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 37, pp. 1179-1184, 1999

[4] A. Leonardis and H. Bischof, An efficient MDL-based construction of RBF networks, *Neural Networks*, Vol. 11, pp. 963-973, 1998

[5] J. Platt, A Resource-Allocating Network for Function Interpolation, *Neural Computation*, Vol. 3, pp. 213-225, 1991

[6] D. O. Hebb, *The Organisation of Behaviour*, Wiley, 1949

[7] B. V. Dasarthy, Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments., *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, pp. 67-71, 1980

[8] G. W. Gates, The Reduced Nearest Neighbor Rule., *IEEE Transactions on Information Theory*, Vol. pp. 431-433, 1972

[9] S. Waugh and A. Adams, Comparison of inductive learning of classification tasks by neural networks, 1993

[10] P. M. Murphy and D. W. Aha, UCI Repository of machine learning databases, Technical Report University of California, Department of Information and Computer Science, Irvine, CA, 1994