Comparison of Neuron Specific and Fully Connected Centroid Layers in Hybrid Multilayer Perceptron Networks

Mikko Lehtokangas
Tampere University of Technology
Signal Processing Laboratory
P.O.Box 553, FIN-33101 Tampere, Finland
Phone: +358-3-3653881, Fax: +358-3-3653095
email: mikkol@cs.tut.fi

ABSTRACT: In recent studies certain hybrid neural network architectures have been investigated for modelling purposes. These hybrids are based on the multilayer perceptron network which forms the baseline structure. However, in addition to the usual sigmoidal hidden layers the first hidden layer has been selected to be an adaptive centroid layer. Each unit in this exceptional hidden layer incorporates a centroid vector that is located somewhere in the space spanned by the input variables. Two practical centroid layer implementations have been considered. In the first one, each of the units in the centroid layer is connected only to one of the hidden units in the next layer. Thus the centroid layer is neuron specific and it has as many units as the next hidden layer. In the second one, each of the units in the centroid layer is connected to all of the hidden units in the next layer. In this case the centroid layer is fully connected and it usually does not have the same number of units as the next hidden layer. In recent studies, comparisons with conventional multilayer perceptron networks have demonstrated that both of these hybrids can yield improved performance especially in terms of fast learning and compact network structure. In this work we compare the two hybrid approaches, and discuss their pros and cons. In the comparison we have used maximum covariance based weight initialization technique to boost the performance of each method to extreme. Based on the experiments, the fully connected centroid layer seem to lead to faster learning with smaller structure. However, with fully connected centroid layer the model selection is more difficult since there are at least two hidden layers for which the number of units need to be selected.

KEYWORDS: hybrid multilayer perceptron network, adaptive centroid layer, weight initialization, classification

INTRODUCTION

Multilayer perceptron networks are one of the most widely used neural network paradigms, Haykin (1999). However, there are many types of modelling problems where it is difficult to train multilayer perceptron networks with conventional techniques like backpropagation, Haykin (1999). As an example, in classification problems the data forms usually clusters of different shapes. According to these clusters, the decision boundaries can be very complex with the result that it is hard to get multilayer perceptron network to converge into adequate solution. Due to these problems more advanced methods have been developed and applied for training multilayer perceptron networks. These include second order training methods adopted from the optimization theory, Fletcher (1990), and network initialization methods, Lehtokangas et. al. (1998). In addition more complex feedforward network type structures have been proposed like the cascade-correlation structure, Fahlman and Lebiere (1990). Considering especially classification problems, the radial basis function networks represent a completely different approach, Moody and Darken (1989). The radial basis function networks are based on kernel functions which make them well suited for problems where input data forms distinct clusters. However, there are also certain locality problems when training radial basis function networks, Moody and Darken (1989), and the number of hidden units can grow to be very large.

In recent studies we have investigated hybrid multilayer perceptron network structures for classification purposes, Lehtokangas and Saarinen (1997) and Lehtokangas and Saarinen (1998). In these hybrids a multilayer perceptron network forms the baseline structure. This baseline structure has been then modified so that the first hidden layer performs a special centroid transformation for the input data. This transformation is similar to the hidden layer transformation in radial basis function networks. The difference is that instead of using kernel functions, only centroid vectors are used in

these special units. The centroid vectors are located somewhere in the space spanned by the input variables. Two practical centroid layer implementations have been considered. In the first one, each of the units in the centroid layer is connected only to one of the hidden units in the next layer, Lehtokangas and Saarinen (1997). Thus the centroid layer is neuron specific and it has as many units as the next hidden layer. In the second one, each of the units in the centroid layer is connected to all of the hidden units in the next layer, Lehtokangas and Saarinen (1998). In this case the centroid layer is fully connected and it usually does not have the same number of units as the next hidden layer. Comparisons with conventional multilayer perceptron networks have demonstrated that both of these hybrids can yield improved performance especially in terms of fast learning and compact network structure. In this work we compare these two hybrid approaches, and discuss their pros and cons. In the comparison we have used maximum covariance based weight initialization technique to boost the performance of each method to extreme, Lehtokangas et. al. (1998). Based on the experiments, the fully connected centroid layer seem to lead to faster learning with smaller structure. However, with fully connected centroid layer the model selection is more difficult since there are at least two hidden layers for which the number of units need to be selected.

EFFECT OF CENTROID TRANSFORMATION

Before introducing the hybrid structures in detail, let us illustrate the effect of centroid transformation by a simple example. In this example the centroid transformation is defined as computing the Euclidean distance between the network inputs and each centroid vector. Clearly, this transformation is similar to the one used in radial basis function networks. Now, it is widely known that classification of circular shape clusters like ellipsoid depicted in Fig. 1a is rather difficult task for a multilayer perceptron network. Quite a many hidden units would be required to get even adequate performance. As a result, in such a problem one would prefer to use for example radial basis function networks. However, for the sake of simplicity radial basis function type models usually use radially symmetric kernel functions. Therefore radial basis function type models have in many cases also difficulties when the clusters are not radially symmetric. These difficulties are increased by the fact that usually the kernel centers are obtained with some sort of clustering algorithm. Many popular simple clustering methods are quite vulnerable to local minima. Moreover, we usually do not know the correct number of clusters which also has significant effect on the clustering. As a result, in many simple problems (like ellipsoid) radial basis function networks may also require quite a many hidden units for accurate classification.

Now, considering multilayer perceptron networks with additional centroid layer, Fig. 1 depicts a simple centroid transformation. In Fig. 1a is the original data where the goal is to separate an ellipsoid shape cluster (black area) from the other data (white area). Then Fig. 1b shows the data after the centroid transformation. The transformation was obtained by using centroid layer with two centroid units. For the sake of simplicity the centroid vectors were obtained with simple k-means clustering. Clearly, after centroid transformation the problem is virtually linearly separable (Fig. 1b). Therefore, in this simple one cluster case the hidden layers following the centroid layer have a very easy task to do the final separation. Obviously in more difficult problems the data may not be quite linearly separable after the centroid layer. This is because we usually have the wrong number of centroid units in traditional sense (traditionally we would have one unit for each circular cluster) and the centroid vectors are usually not located in the center of the circular clusters (we do not know the correct number of clusters; this tends to move centroid vectors away from cluster centers during clustering). However, as our example in Fig. 1 demonstrate, the centroid transformation tend to remove circular shapes from the decision boundary. Therefore, even though the data may not be linearly separable after the centroid transformation, the remaining task can be much easier in the sigmoidal hidden layers perspective. Note that in the radial basis function net-

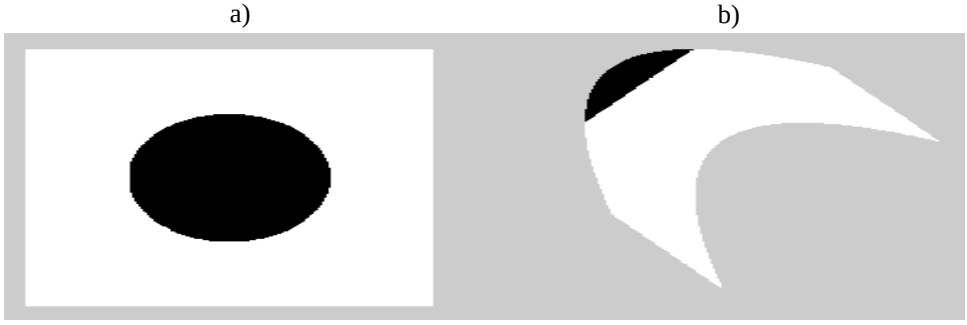

Figure 1: Example of centroid transformation: a) original data, and b) data after centroid transformation. The black and write areas denote the two classes.

works it is assumed that the problem is linearly separable after the single hidden layer. However, as the training problems described above suggest, in many cases this assumption does not hold. As a result, multilayer perceptron network with centroid layer can be seen as an radial basis function related model where the above assumption has been omitted, and the possible remaining nonlinearities are handled by using sigmoidal hidden layers. Note also that because we do not have the linear separability assumption after the centroid layer, the problem of finding exactly the correct cluster centers is not so critical any more since the imperfections can be compensated with the following sigmoidal hidden layers.

HYBRID MULTILAYER PERCEPTRON NETWORKS

In this section we briefly recall the two hybrid approaches to enable structural and computational comparison. In both designs the first and last layers are the input and output layers, respectively. The second layer is the centroid layer where each unit incorporates a centroid vector. Between the centroid and output layers are the sigmoidal hidden layers which are equal to the hidden layers in basic multilayer perceptron. For the sake of simplicity, in the following we consider structures where there are only one sigmoidal hidden layer.

Let us first consider the design where each centroid unit is neuron specific, Lehtokangas and Saarinen (1997). In this case the outputs of such a centroid unit are the component wise squared differences between the centroid vector elements and inputs of the network. Further, the outputs of a centroid unit are connected to only one hidden unit in the next layer. Hence the number of centroid units is equal to the number of hidden units in the next layer. In this case the network function is

$$o_1 = f\left(v_0 + \sum_{j=1}^{q} v_j \, f\left(w_{0j} + \sum_{i=1}^{p} w_{ij} \, (x_i - c_{ij})^2 \right) \right), \tag{1}$$

in which c_{ij} are the centroid vector parameters, w_{ij} are the weights between the centroid and sigmoidal hidden layers, and v_j are the weights between the sigmoidal hidden (there are q hidden units) and output layer. Also, x_i are the network inputs (there are p inputs), o_1 is the network output, and f is the activation function. Here we chose the activation function to be the hyperbolic tangent function. In the other design the centroid layer is fully connected to the next layer, Lehtokangas and Saarinen (1998). That is, now each centroid unit has only one output which is connected to all units in the next layer. The output is obtained as the squared Euclidean distance between a centroid vector and network input vector. The network function for this second design is

$$o_2 = f\left(v_0 + \sum_{k=1}^{r} v_k \, f\left(w_{0k} + \sum_{j=1}^{q} w_{jk} \sum_{i=1}^{p} (x_i - c_{ij})^2 \right) \right), \tag{2}$$

in which c_{ij}, w_{jk}, v_k, x_i, f and p are as above. In addition, q is the number of centroid units, r is the number of sigmoidal hidden units and o_2 is the network output. As can be seen the biggest difference between the two designs is that in (1) the centroid units are tied to sigmoidal hidden units in the next layer. So only the value for q need to be determined before training. In (2) each hidden layer has different number of units, so values for both q and r need to be determined before training. Hence, the model selection problem is more difficult with (2) than with (1) even though computationally (1) and (2) are similar. Computational cost of training is also an important aspect, which will be discussed next.

In both of the above designs the training of the connection weights w and v can be accomplished with standard backpropagation type training rules. Hence we can restrict ourselves to make comparison of the training rules for the centroid parameters only. Because we consider classification problems, the natural choice for training cost function is the relative entropy cost function, Haykin (1999). Now, many popular training methods are based on the first-order gradient descent techniques. To be able to use these generic training rules for adjusting the centroid vectors, we need to find out the derivative of the cost function as follows

$$\frac{\partial RE}{\partial c_{ij}} = \sum_{l=1}^{n} \frac{\partial RE_l}{\partial c_{ij}} \, . \tag{3}$$

For design (1) we obtain

$$\frac{\partial RE_{1l}}{\partial c_{ij}} \propto (x_{il} - c_{ij}) \; (d_l - o_{1l}) \, w_{ij} \, f_{jl}' \, v_j \, . \tag{4}$$

In above, d is the desired output and f' is the derivative of the activation function in the sigmoidal hidden layer. Similarly, for design (2) we get

$$\frac{\partial RE_{2l}}{\partial c_{ij}} \propto (x_{il} - c_{ij})\ (d_l - o_{2l}) \sum_k w_{jk}\ f_{kl}'\ v_k\ . \tag{5}$$

Obviously, for the second design the training rule is little more complex but not dramatically. The difference here is again due to the fact that in the second design the centroid layer and the next hidden layer are fully connected instead of being neuron specific like in the first design. We can summarize that the first design is in general more simpler and computationally less demanding, but the second design is potentially more flexible due to the increased degree of freedom (layers are not tied).

EXPERIMENTS

In this section the performance of the above described hybrid designs are empirically investigated. We have used two benchmark problems. The first problem is the two spirals problem, Fahlman and Lebiere (1990), where the goal is to separate two interlocked spirals. This problem has been found relatively hard to solve with sigmoidal feedforward neural networks. The second problem is the channel equalization problem described in Kantsila et al. (to appear). There the goal is to equalize a burst of bits transmitted through a fixed communication channel having 20db signal-to-noise ratio. It should be emphasized that the two-spirals problem demonstrates only the convergence while the equalization problem demonstrates also generalization performance. In all the experiments the RPROP, Riedmiller (1994), optimization procedure was used for the cost function minimization. In addition, maximum covariance weight initialization, Lehtokangas et.al. (1998), was applied to boost the performance of both methods. Each of the simulations were repeated ten times.

According to some experimentation, suitable numbers for the hidden units were found to be as follows: Fig. 2a) q=27 (163 network parameters), 2b) q=5 and r=14 (109 network parameters), 3a) q=6 (37 network parameters), and 3b) q=3 and r=5 (32 network parameters). The convergence in the two spirals problem (Fig. 2) is remarkable. With neuron specific centroid layer 600 to 1500 training epochs are required, and with fully connected centroid layer only 100 to 600 training epochs are required. Note also the small amount of parameters with the latter method. Earlier studies have reported that several thousands of epochs are required (even tens of thousands) with over 160 network parameters, Fahlman and Lebiere (1990). Here, on average the first design required 429 Mflops (floating point operations) for the training and the second design only 93 Mflops. Considering the equalization problem (Fig. 3), there the convergence seem to be similar with both designs in terms of training epochs. However, we found that with the first design the training took 1.6 Mflops and the resulted generalization error was 1.61%. For the second design the training took 0.9 Mflops and the resulted generalization error was 1.64%. Hence, both designs generalized equally well, but the second design (fully connected) required considerable less computation. This is partly due to the smaller amount of parameters required but also in the second design the maximum covariance weight initialization can be implemented much more efficiently.

CONCLUSIONS

Neuron specific and fully connected centroid layers in hybrid multilayer perceptron network were considered. We first pointed out the problems that multilayer perceptron networks have in classifying circular shapes. Then we briefly illustrated the centroid transformation for removing/reducing the circular shapes. This transformation was then added to the multilayer perceptron network as a so called centroid layer. The resulting structure is a kind of hybrid multilayer perceptron. Two different hybrid designs (neuron specific and fully connected) were described and compared. The neuron specific hybrid design was found to be simpler in terms of network structure and training rule. However, experimental results demonstrated, that the fully connected design yields faster learning in practise. Note that maximum covariance weight initialization was used to boost the training performance. Both designs generalized equally well. As a results, the multilayer perceptron network with fully connected centroid layer seem to be potentially more viable approach.

ACKNOWLEDGEMENTS

This work has been supported by the Academy of Finland.

REFERENCES

Fahlman S. and Lebiere C., 1990, "The cascade-correlation learning architecture," In D. Touretzky (Ed.), Advances in Neural Information Processing Systems 2, San Mateo, CA, Morgan Kaufman, pp. 524-532.

Fletcher R., 1990, Practical methods of optimization, 2nd edition, Wiley, Chichester.

Haykin S., 1999, Neural networks: a comprehensive foundation, 2nd edition, Prentice Hall, New Jersey.

Kantsila A., Lehtokangas M. and Saarinen J., to appear, “Burst adaptive equalization of binary data,” Journal of Intelligent Systems.

Lehtokangas M. and Saarinen J., 1997, “Classification using multilayer perceptron networks with adaptive neuron specific preprocessing units,” Proceedings of International Conference on Signal and Image Processing, New Orleans, pp. 112- 116.

Lehtokangas M. and Saarinen J., 1998, “Centroid based multilayer perceptron networks,” Neural Processing Letters, vol. 7, pp. 101-106.

Lehtokangas M., Salmela P., Saarinen J. and Kaski K., 1998, “Weight initialization techniques,” In C. Leondes (Ed.), Algorithms and Architectures, Series on Neural Network Systems Techniques and Applications, Academic Press, San Diego.

Moody J. and Darken C., 1989, “Fast learning in networks of locally-tuned processing units,” Neural Computation, vol. 1, pp. 281-294.

Riedmiller M., 1994, “Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptive learning algorithms,” International Journal on Computer Standards and Interfaces, Special Issue on Neural Networks, vol. 5.

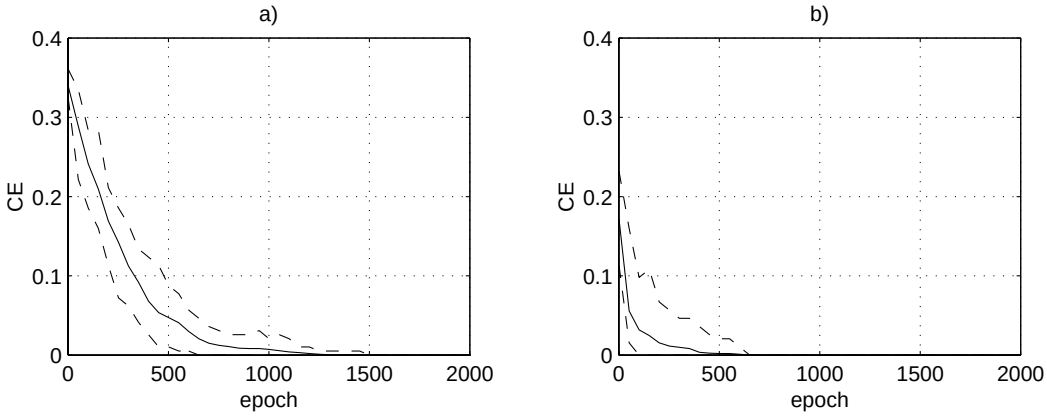

Figure 2: Classification error as a function of training epochs for the two-spirals problem; a) MLP with neuron specific centroid units, and b) MLP with fully connected centroid units. The solid line represent the average curve, and the dashed lines represent the minimum and maximum curves, respectively.

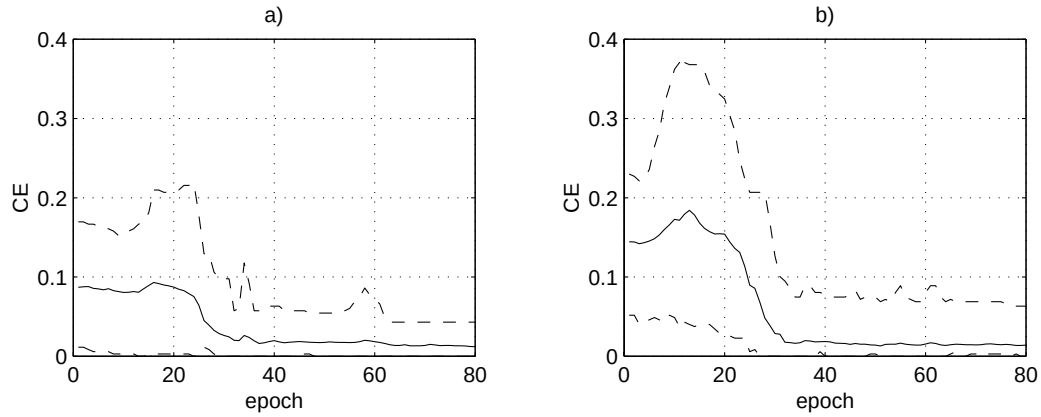

Figure 3: Classification error as a function of training epochs for the *independent test set* of the equalization problem; a) MLP with neuron specific centroid units, and b) MLP with fully connected centroid units. The solid line represent the average curve, and the dashed lines represent the minimum and maximum curves, respectively.