ELSEVIER

Pattern Recognition Letters 18 (1997) 1293–1300

Pattern Recognition Letters

Investigating redundancy in feed-forward neural classifiers

Aarnoud Hoekstra [*,1], Robert P.W. Duin

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

Abstract

In this article we will focus on how we can investigate (read visualise) the clustering behaviour of neurons during training. This clustering property has already been investigated before, by Annema, Vogtländer and Schmidt. However, we will present a different approach in visualisation illustrated by experiments performed on two-class problems. © 1997 Elsevier Science B.V.

Keywords: Self-organising map; Feed-forward neural classifier training; Symmetry breaking

1. Introduction

Training neural networks is a difficult task. In general it is very hard to understand what is happening inside a network. Furthermore, many parameters have to be chosen in order to solve a particular problem. In this article we will focus on investigating the training behaviour of feed-forward neural classifiers. Moreover we are interested in how the weights of such a classifier behave during training. It is generally assumed that during training a learning rule directs the weights of a network in such a way that it solves a problem as well as possible. The number of weights may be much larger than the number of samples used to find the weights. This causes some kind of redundancy in the network in the sense that some weights, or neurons as we will assume, perform the same operation. Consequently the neurons, or weights belonging to a neuron, are somehow clustered in the weight space. Furthermore, since network training starts from approximately the same weights, neurons start off as a large cluster and then (slowly) break down in smaller clusters which solve the problem. This behaviour of the weights, referred to as symmetry breaking, can also be observed in the mean squared error curve during network training. In this curve changes of the error occur. Annema (1994) and Vogtländer (1994) showed that these changes coincide with the change of direction of the decision function of a hidden unit, i.e. 'breaking' of symmetry. Our experiments confirm this behaviour in a visualisation of the neuron weight space.

Usually the initial weights in a network are chosen small and centered around 0. For a feed-forward network with sigmoidal output neurons this means that all the neurons have approximately the same output (somewhere around 0.5) and implement approximately the same function. In Fig. 1 this is graphically illustrated for two data sets. It shows the decision function of the different hidden neurons at the point of initialization. Most of the neurons perform the same function, i.e. their decision functions

[*] Corresponding author.

[1] This work was partially supported by the Foundation of Computer Science in the Netherlands (SION) and the Dutch Organization for Scientific Research (NWO).

are clustered. Consequently there is a large redundancy in the network. It is expected that during training these clusters of neurons break apart in smaller clusters, thereby reducing the redundancy in the network. However, if all redundancy is removed from the network, it may become overtrained. The question is: can we visualise the behaviour of the weights? If so, is it possible to detect the symmetry breaking property of a neural network and is it possible to detect cases of undertraining and overtraining? In this article we will address these questions by presenting a mapping tool and illustrate the behaviour using two-class classification problems.

It is assumed here that we investigate feed-forward neural networks consisting of a single hidden layer. The neurons in the hidden layer span a space by their weights, the neuron weight space. The dimensionality of this space depends on the dimensionality of the input space and the output space. In the experiments to be presented, the neurons span a four-dimensional space. During training the hidden neurons move through that space. When a sufficient number of neurons is available, some neurons will perform the same task indicating that there is some redundancy in the network, especially at the initialization point of the network.

The neuron weight space, referred to as weight space, can be considered as a feature space in which the neurons, i.e. their corresponding weight vectors, are data points. In order to find out whether they form clusters or not, cluster analysis has to be performed. This can be done by traditional clustering methods like hierarchical clustering or k-means clustering. However, one is also interested in what the weight space looks like. Are there regions which are never visited by the training algorithm? This requires a method to map the high dimensional weight space onto a lower, easy to visualise, one. When a traditional method is used, one needs a method to display the clusters found, which is in general hard to do. We will show that by using the self-organising map (SOM) (Kohonen, 1989) we are able to do both: project the weight space and observe the clustering properties of the weights. The SOM can be used to map a high dimensional space onto a lower dimensional one in such a way that the topological properties of that space are preserved as well as possible. This enables one to get an indication of what the space looks like. Moreover, by determining the trajectories of the weights on the SOM grid it can be shown that the weights indeed break into smaller and smaller clusters.

A technique introduced in (Kraaijveld et al., 1995), is used to create an image of the scatter of the data in the feature space using the mapping found by the network. After training, the distances of neighbouring neurons in the SOM grid are determined and displayed as gray value images. This is done by calculating the Euclidean distance of a grid neuron prototype vector to the prototype vectors of its four neighbouring neurons. The gray value for the grid neuron under investigation is now set to the maximum distance found among the four calculated inter-neuron distances. In this way a gray value image of the resulting mapping can be found. Neurons having a low gray value will lie close together in the original space and will therefore belong to the same cluster. A large value for a neuron implies a large distance between the neurons in the weight space and therefore the neurons probably belong to different clusters. By also inserting the trajectory of the neurons one is able to get an indication of what the neuron weight space looks like and how it is searched.

2. Experiments

In order to investigate the behaviour of the weights of a neural network in the weight space, we performed several experiments. The neural networks were trained on two different data sets, one perfectly separable and one overlapping data set. Fig. 1 shows the two data sets. On the left the perfectly separable data set is depicted. It was derived from (Annema, 1994) and (Vogtländer, 1994), who also performed experiments on redundancy in neural classifiers. The overlapping data set was derived from (Hoekstra and Duin, 1996). This set which was used to ensure overtrained neural classifiers, are of particular importance since it is expected that all neurons behave differently.

All networks used in the experiments consist of two inputs, six hidden neurons to ensure enough freedom for the network to adapt, and one output neuron since both problems are two class problems. For each of these data sets different networks with

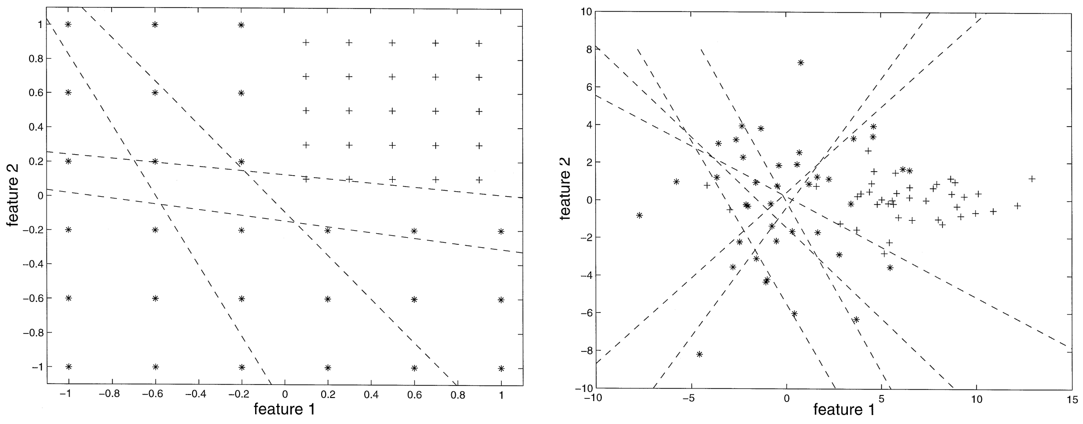

Fig. 1. Two networks consisting of six hidden neurons, for two different data sets, at the initialization point. The dotted line denotes a decision function of a hidden neuron. The clustering behaviour of the neurons is clearly visible due to decision functions having the same direction.

different learning methods were trained, *standard backpropagation* using our C-library (Hoekstra et al., 1996), *Levenberg–Marquardt* and *fast backpropagation* using our Matlab PRTools toolbox (Duin, 1995) and the neural network toolbox. The reason for choosing different learning rules is to investigate the influence of the learning rule in redundancy reduction. A SOM was trained on the weights of the different networks. This was done in the following way. At certain time steps the weights of the hidden units of the networks are encoded in a four-dimensional vector and saved in a file, the training set of the SOM. Fig. 2 shows how the encoding works.

After training the feed-forward networks we have a set of four-dimensional weight vectors which constitute the training set for the SOM. These vectors are normalized in order to prevent large weights from dominating the SOM training process. This procedure normalizes the weights such that the variance is scaled to 1. For each collection of weights a SOM was trained. Note, however, that due to the different learning methods weight files have unequal lengths. The self-organising maps consist of a two-dimensional grid of 150×150 neurons, to ensure enough detail. Since it is not known in advance how the weight space is searched, a large map is chosen. The maps were trained for a large number of epochs (500 000) to ensure convergence. After training the distances of neighbouring neurons in the grid were calculated and displayed in a gray value image. In Fig. 3 such an image is shown. It can clearly be seen that there are regions of different gray levels. The darker areas in the picture are places where the neurons have a small neighbouring distance. The next two subsections show the experimental results for the two data sets.

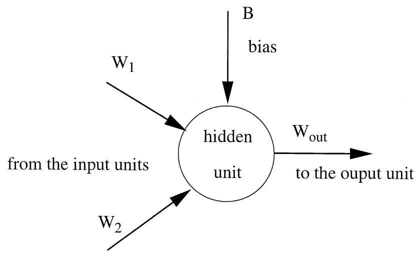

Fig. 2. Encoding of the weights of a hidden neuron. This four-dimensional vector, $(W_1, W_2, B, W_{\text{out}})$, is used as a training pattern for the SOM.

2.1. Separable data set

Here, for the three different training methods, ten different initializations of networks were used. As can be observed from Fig. 1, the data set can be separated with at least two hidden neurons. However, we used networks with six hidden neurons to ensure enough redundancy and it is therefore expected that the weights of the networks still remain clustered after a satisfactory solution is obtained. Figs. 3–5 show the results for the backpropagation rule, the Levenberg–Marquardt and fast backpropagation rule, respectively. As can be observed, there

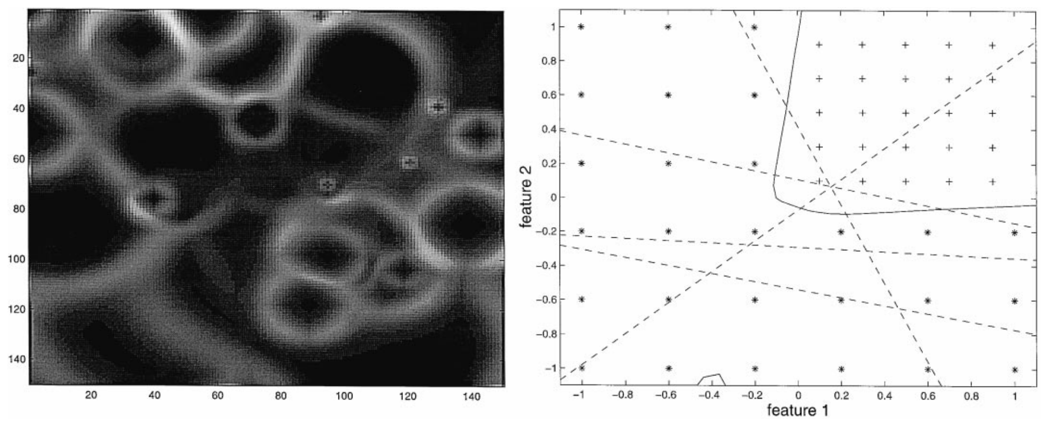

Fig. 3. The SOM image of the weights resulting from the of the standard backpropagation training with momentum on the ‘Annema’ set are displayed on the left. On the right the final decision function for a network is displayed. The solid line depicts the output function and the dashed lines are the decision functions implemented by the hidden neurons.

is a significant difference between the different rules on the same training data. The SOM for the Levenberg–Marquardt training rule in Fig. 4 (left picture) shows the flattest surface (almost black everywhere) indicating that the neurons are always clustered together. Apparently this learning rule quickly finds a satisfactory solution and almost always in the same manner. The experiments show that the Levenberg–Marquardt rule usually finds a solution after one iteration. This is probably due to the fact that it is a second-order method and that the data set is easily separable. Note that the surface is the result of averaging the results of ten different network initializations.

The surface for the standard backpropagation rule with momentum shows far more variation. During training the weights can differ quite dramatically causing a rough surface. The isolated black regions in the map (Fig. 3) show that the ten networks have been trained quite long resulting in regions to a far distance of the initialization. After a certain time the weights do not alter much anymore and start to circulate in the map. In Fig. 9 this map is shown again, but now with the trajectories of the six hidden neurons. These trajectories are for a particular realization of one of the ten trained networks, i.e. one training session of one neural network. The starting point of the weight trajectory is indicated by the

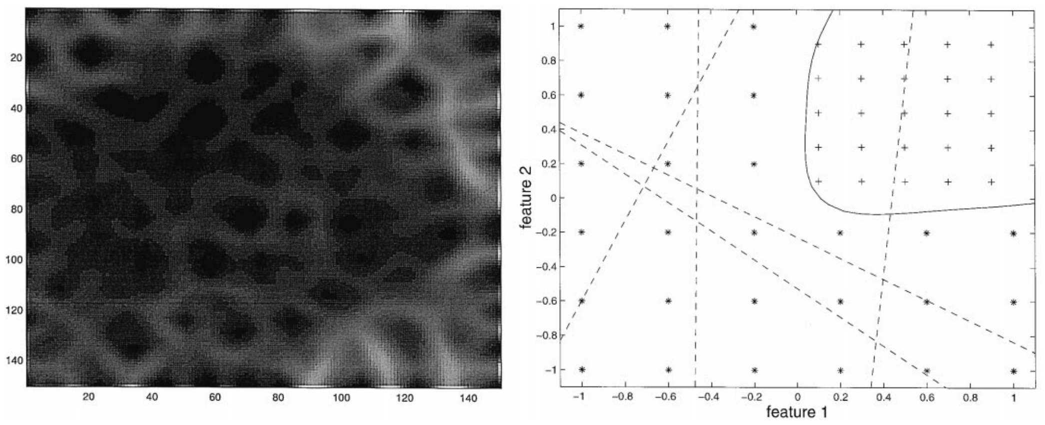

Fig. 4. The SOM image of the weights resulting from the Levenberg–Marquardt training on the ‘Annema’ set are displayed on the left. On the right the final decision function for a network is displayed. The solid line depicts the output function and the dashed lines are the decision functions implemented by the hidden neurons.

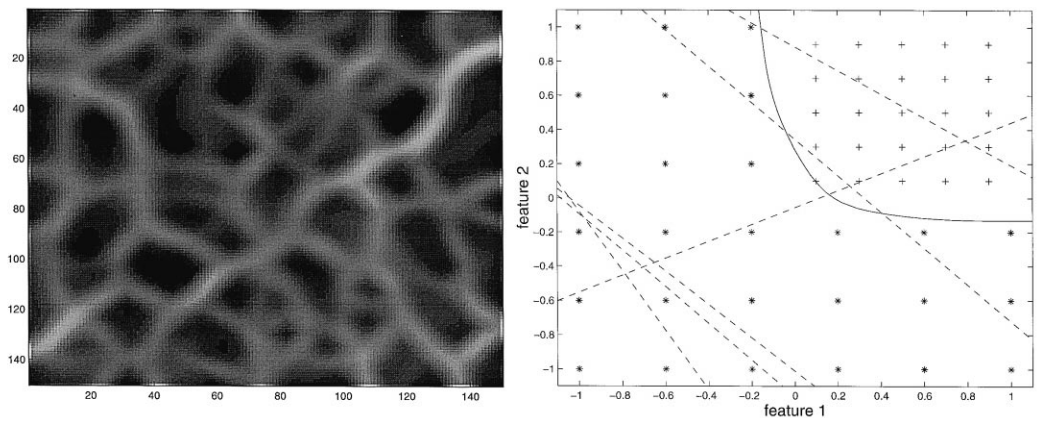

Fig. 5. The SOM image of the weights resulting from the fast backpropagation training on the 'Annema' set are displayed on the left. On the right the final decision function for a network is displayed. The solid line depicts the output function and the dashed lines are the decision functions implemented by the hidden neurons.

double circle. From this point the weights move over the map towards a more or less stable situation. It should be noted that the map is an average over many weight configurations. It can be seen that some trajectories stay close together and split at a certain time step, indicated by a circle. This splitting of the paths coincides with a change in the mean squared error (MSE) and therefore with a decrease in redundancy. The MSE curve shown in Fig. 8 shows the drops in error corresponding to redundancy reduction. However, the danger still exists that the network becomes overtrained.

2.2. Overlapping data set

The overlapping data set (the right picture in Fig. 1) is expected to cause more redundancy reduction since it easily causes the network to become overtrained when using a constant MSE as a stopping criterion. Here we trained ten networks using two

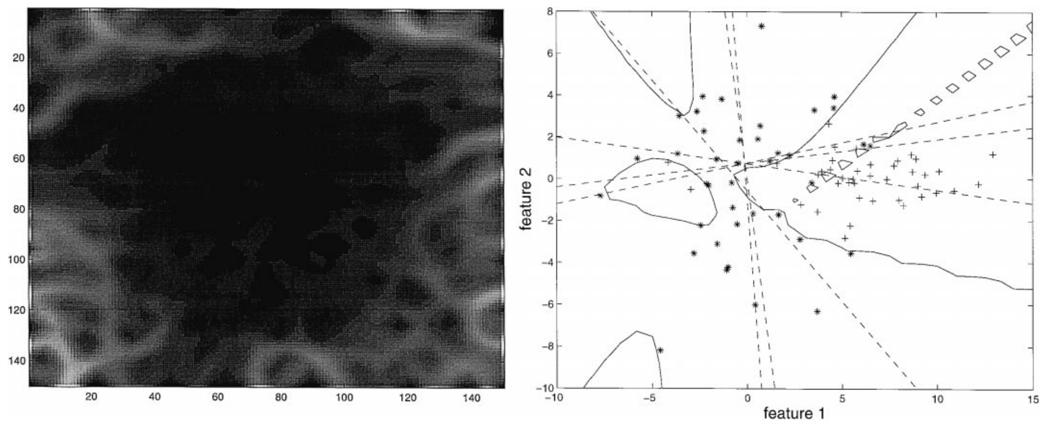

Fig. 6. The SOM image of the weights resulting from the Levenberg–Marquardt rule on the overlapping data set are displayed on the left. On the right the final decision function for a network is displayed. The solid line depicts the output function and the dashed lines are the decision functions implemented by the hidden neurons.

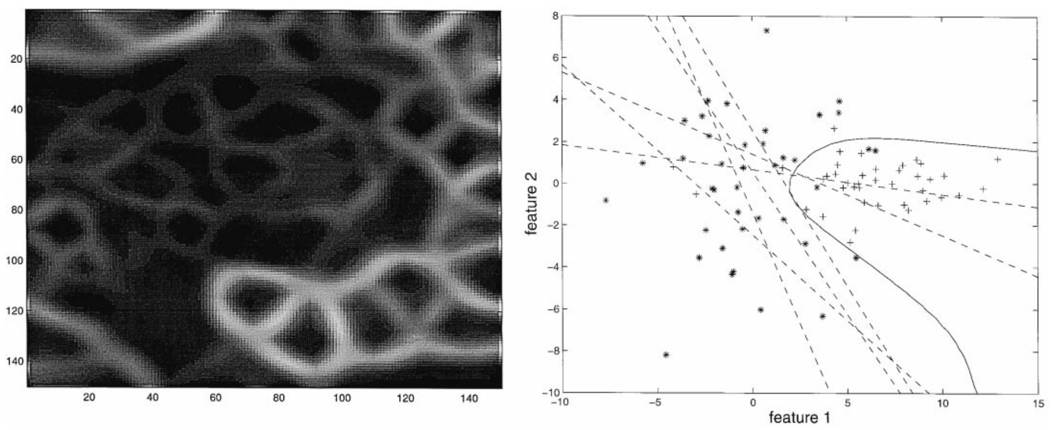

Fig. 7. The SOM image of the weights resulting from the fast backpropagation method on the overlapping data set are displayed on the left. On the right the final decision function for a network is displayed. The solid line depicts the output function and the dashed lines are the decision functions implemented by the hidden neurons.

learning methods, the *Levenberg–Marquardt* rule and *fast backpropagation*, both available through the neural network toolbox in Matlab. Figs. 6 and 7 show the results for both methods. The Levenberg–Marquardt clearly causes an overtrained situation (right part of Fig. 6), which was expected. However, this does not cause any ‘sharp’ regions in the projected weight space which is quite flat, i.e. a large dark region. Probably all the realizations of the networks are overtrained and therefore there seems to be no structure in the weight space present, i.e. no clear clusters.

Fig. 7 shows a well trained neural network. However, not all realizations are well trained. There are also cases of overtraining, which cause separate clusters in the weight space. This is in contrast to the Levenberg–Marquardt method which skips those regions due to its second order nature. The behaviour of the fast backpropagation is clearly shown: there are flat areas and small isolated areas. Its behaviour can be compared to the standard backpropagation method used in the previous subsection. Also in that case one notices small isolated areas to which the weights have converged.

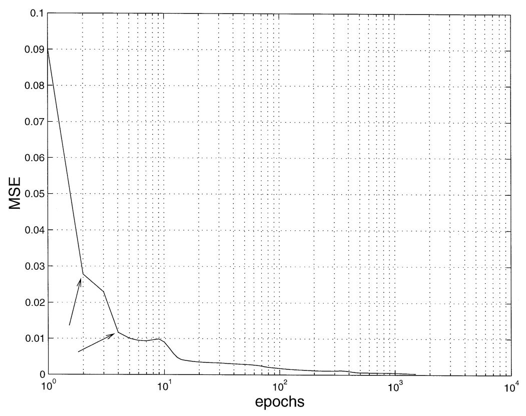

Fig. 8. The MSE curve of the network of which the neuron weight trajectories have been plotted in Fig. 9. The arrows indicate symmetry breaking in the network and correspond to the circles drawn in Fig. 9.

3. Conclusions

By using the SOM for mapping the high dimensional weight space onto a two dimensional grid, visualisation of the weight space is possible. It can clearly be seen that there are different regions of density in that space. The resulting SOM is determined by the weights of the different networks, in our case ten. This may not be a sufficient number because another network may explore another part of the space. Experiments, however, do indicate that the weights roughly explore the same paths in space, although when trained quite long the paths lie far from each other. Furthermore, by using a SOM mapping, some information is lost in the mapping. The symmetry breaking effect present in neural networks can be observed and coincides with the changes in the mean squared error during training.

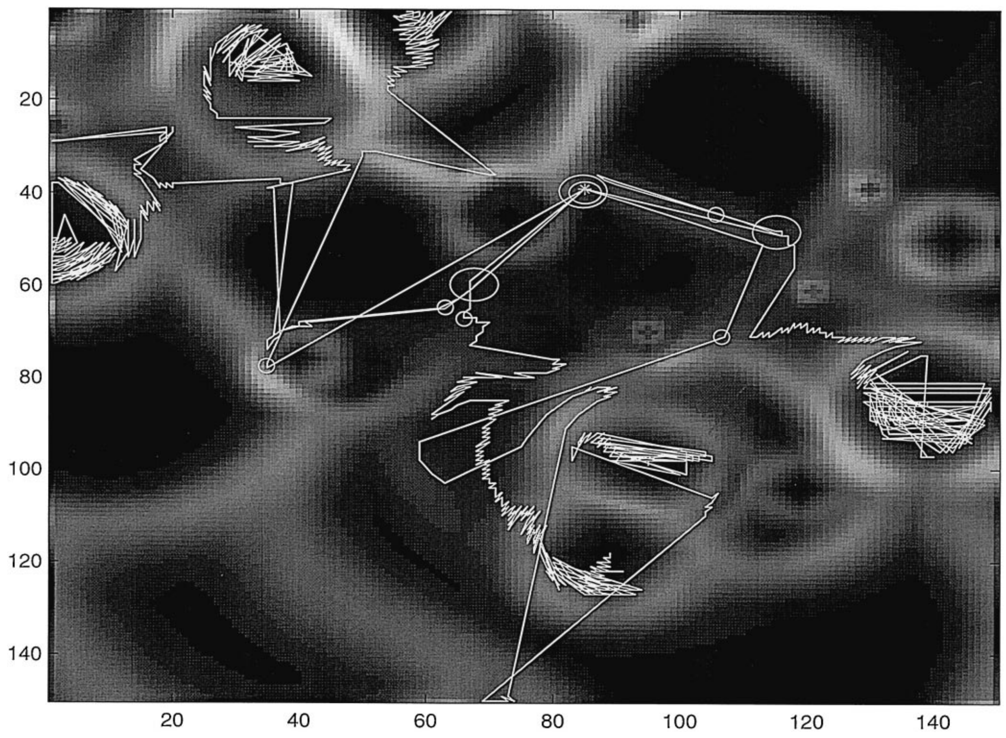

Fig. 9. The SOM image of the weights resulting from the backpropagation rule with momentum on the ‘Annema’ set. In this figure the trajectories of the six hidden neurons for a particular network realization are shown. The redundancy reduction property is shown by the different paths that split. The double circle indicates the initialization point of the network, whereas a single circle indicates symmetry breaking which corresponds to the MSE drops in Fig. 8. A small single circle refers to the second drop whereas a large circle refers to the first drop.

Experiments show that the weights of the different networks explore different trajectories in the neuron weight space. Initially some of the trajectories remain in each others neighbourhood, but after some time their paths diverge, signalling reduction of redundancy.

Furthermore, there is clearly a difference in the behaviour of different learning rules. The Levenberg–Marquardt is a fast method which leads to a fast solution and roughly the same weight configuration in each experiment, reflected in large flat areas in the visualization. Apparently the method is not very sensitive to the initialization. Backpropagation methods explore much larger areas and find different weight configurations as may be observed in the presented figures. However, these methods are much more sensitive to an initialization, which directs the initial direction for searching the weight space. In contrast, a Levenberg–Marquardt method finds a solution far more quickly.

Discussion

Kittler: You showed your results for a fully connected network. Do you know how the cluster tendency depends on the connectivity?

Hoekstra: No, I have not studied that, it might be a problem. In the network everything is fully connected so I have for each hidden unit the same dimensionality of the vector. In cases of not fully connected networks, you should do something with missing values. So you have to align all the vectors, such that they have equal size. And I don't know how that behaves, since that creates subspaces.

Mao: The results are very interesting. They reveal some links between radial basis networks and the standard feed-forward network. In the radial basis function network the hidden layer also has some clustering properties in the original feature space. In your case it is in the weight space, so there is a difference. You have discovered this cluster tendency of the weight vectors by using several different algorithms. Is there any theoretical analysis to reveal why this cluster tendency should exist, and is there a certain way to force the learning algorithm to form clusters by some regularization techniques? For example, in a paper by Oja, a few years ago, the method called the soft weight sharing method was described. They basically apply a mixture of two Gaussians as a prior distribution of weights and then they use a back-propagation algorithm to train and force the weights into two clusters.

Hoekstra: I am not familiar with the reference that you quote, that is, I have seen it but I have not gone into its details.

Sklansky: I suggest another approach to determine the number of neurons in the first hidden layer. The idea is to partition the data into clusters, for instance by K-means and then to look for hyperplanes that separate clusters in opposite classes. The minimum number of hyperplanes is easy to find and that minimum number is a very good choice of the number of neurons in the first hidden layer.

Hoekstra: Yes, but our viewpoint was somewhat different. We saw some phenomena happening in training, and we just wanted to know what that was. So we are not all that interested in the original data, but in what is happening in the network. Of course, a possible application is to find the number of nodes. But we just started out with exploring the phenomena.

Roli: You said that you performed the clustering algorithm in the weight space. The feature space is related to the weights of neurons. You know there is a problem well known in literature due to symmetry of the weights, in the sense that you can have weights with very different values, whereas the related neurons perform the same operation.

Hoekstra: Yes, that may be the case. I did do some normalization on the weights, to prevent the weights from exploding.

Kappen: I have a comment. There is some theoretical work in this direction by David Saad and Sarah Solla. They have used statistical mechanics techniques to look at symmetry breaking in the hidden layer, specifically in a committee machine, where the second layer weights are fixed and the symmetry breaking is in the first layer.

Hoekstra: I am not a physicist, and thus I am not familiar with statistical mechanics, but I have read some early work on symmetry breaking in perceptrons.

References

Annema, A.J., 1994. Analysis, Modeling and Implementation of Analog Integrated Neural Networks. PhD thesis, Twente University.

Duin, R.P.W., 1995. PRTOOLS a Matlab toolbox for pattern recognition. Pattern Recognition group, Delft University of Technology.

Hoekstra, A., Duin, R.P.W., 1996. On the non-linearity of pattern classifiers. In: Proceedings of the 13th ICPR, Vienna, pp. 271–275. IEEE Computer Society Press, Los Alamitos. Vol. 3, track D: Parallel and Connectionist Systems.

Hoekstra, A., Kraaijveld, M.A., Ridder, D. de, Schmidt, W.F., 1996. The Complete SPRLIB & ANNLIB. Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology.

Kohonen, T., 1989. Self-Organization and Associative Memory. 3rd edition, Springer-Verlag, Heidelberg,

Kraaijveld, M.A., Mao, J., Jain, A.K., 1995. A non-linear projection method based on kohonen's topology preserving maps. IEEE transactions on neural networks 6 (6), 548–559.

Vogtländer, A.C., 1994. The learning behaviour of multilayer networks used as classifiers. Master's thesis, Faculty of Applied Physics, Delft University of technology.