Neuron Clustering in Feed Forward Neural Networks

Aarnoud Hoekstra and Robert P.W. Duin

Pattern Recognition Group,
Faculty of Applied Physics, Delft University of Technology,
Lorentzweg 1, 2628 CJ Delft, The Netherlands
aarnoud@ph.tn.tudelft.nl

Keywords: neural networks, symmetry breaking, redundancy, clustering.

Abstract

Training neural networks is a difficult task. In general it is very hard to understand what is happening inside a network. Furthermore, lots of parameters have to be chosen in order to solve a particular problem. In this article we will focus on investing 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 good as possible. The number of weights might 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. Moreover, 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.

In this article we will focus on how we can investigate (read visualize) this clustering behaviour of neurons during training. This clustering property has already been investigated before [Ann94, Vog94, Sch93], however we will present a different approach in visualization illustrated by an experiment performed on a two class problem.

1 Introduction

When training a neural network the weights of the different neurons in the hidden layer, assuming a single hidden layer, follow some path through the weight space. This path is determined by both the training algorithm and the initialization, which determines the starting position of the neurons in the weight space. After training the neurons end up somewhere in the weight space, such that the problem is solved. Generally, 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 where all the weights are approximately equal. Consequently there is a large redundancy in the network. During training these clusters break apart in smaller clusters, thereby reducing the redundancy in the network. However if all redundancy is removed from the network, it may be overtrained. If one is able to visualize the behaviour of the weights one might also be able to see cases of undertraining and overtraining.

The neuron weight space, referred to as weight space, can be considered to be a feature space in which the neurons, i.e. their corresponding weights, 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 a tree or k-means clustering. However, one is also interested in how the weight space looks like and which paths are followed in that space. Are there any regions which are never met or bypassed by the training algorithm? This requires a method which is able to map the high dimensional weight space onto a lower, visualizable, one. We will show that by using the self organising map (SOM) [Koh89] 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 feature space onto a low dimensional one in such a way that the topological properties of that space are preserved. This enables one to get an indication of how the feature space looks like. Moreover, by determining the trajectories of the neuron weight vectors on the SOM grid it can be shown that those weights indeed break into smaller and smaller clusters. A technique introduced by Kraaijveld et. al. [KMJ95], is used to create an image of the mapping of the feature space.

The SOM is topology preserving since the neighbourhood is taken into account during learning. Therefore it is possible to indicate where clusters are in the high dimensional space which was mapped. By calculating the distances between neighbouring neurons (assuming they are 8-connected) in the 2-dimensional grid it is possible to detect clusters. If the distance between neighbouring neurons is large in terms of Euclidean distance, it is less probable that they represent the same cluster. However, if they have a small distance than it is probable that they belong to the same cluster. By displaying these distances as gray values (see figure 4) the topology of the original space is reveal. For figure 4 holds, the darker and area the more distance there is between neighbours. By also inserting the trajectory of the neurons one is able to get an indication of how the feature space, in our case the neuron weight space, is searched during training.

2 Experiments

In order to investigate the behaviour of the weights of a neural network in the weight space, we performed the following experiment. Ten feed forward neural networks, each consisting of a single hidden layer, were trained on a relatively simple two class problem. They were trained using standard backpropagation. The left figure in figure 1 shows the training data, the two classes are denoted by '+' and '$*$'. This data was derived from Annema [Ann94] and Vogtländer [Vog94] who also performed experiments on redundancy in neural classifiers. As can be observed, the two classes are separable with zero error, and require a network which has at least two hidden units. In our experiment we used networks with six hidden units to ensure enough redundancy. To prevent dependency on the initialization of the weights, each network was initialized randomly before training.

By saving the weights of the hidden units each ten epochs during training, the clustering property of the weights can be investigated. These weights are encoded in a 4-dimensional vector, the two input weights, the bias weight of the hidden neuron and its weight to the output neuron. At each time stamp six of such vectors were saved, one for each hidden units. Since a network is trained for a fixed number of epochs, 15,000, the total number of vectors saved is 9,012 (including the initial weights and the final ones). This file of weight vectors serves as a training file for the SOM which is used for displaying the clustering properties of the weights. For each of the ten networks a weight file was made, also a total weight file consisting of all the weights of the ten networks was made.

The SOM is used to investigate both the clustering property of the weights, as well as to get an impression of the weight space. For each feed forward network a SOM was trained, and one on the total collection of weights. Self organising maps having a 2-dimensional square grid of 150×150 units were chosen to ensure enough detail. Since it is not known in advance how the weight space is searched, it is better to choose a large map. The maps were trained for a large number of epochs $(500,000)$ to ensure convergence of the map. After training the distances of neighbouring neurons in the grid were calculated and displayed in a grey value image.

Figure 4 shows the mapping of the total weight space explored by the ten networks onto a 2-dimensional grid. The darker an area in the grid the further away two neighbouring neurons are. There are clearly some clusters visible in the map. Probably some weights end up circling around in a small partition of the weight space and thus forming a cluster. Furthermore, the trajectories of six hidden neurons of a network are drawn in the map. These are depicted by the black lines, one for each hidden neuron, and show how the neuron weights vectors behave during training. Note that the map is determined by all the weight vectors found during the training sessions, and the drawn trajectories represent just one of these sessions. The network training starts at the black blob in the upper right part of the figure. From this point the network weights start off "exploring" the weight space towards a satisfying solution. The picture clearly shows that some weights tend to cluster and then break apart and re-clusters. At points where weights tend to cluster or break apart a circle is drawn. The thickness of the circle indicates the number of epochs the network has trained, this varies from 20 (thin circle) to 9,850 (fattest circle). The network's decision function at those points is depicted in figure 2 and 3. Figure 2 shows the network for the initialization and epoch 20. Due to the small weight values at initialization, the decision function linear. However, shortly after the first few training steps the network weight break apart (in 4 clusters) and thus reducing the redundancy. This is also shown if figure 1 (right) where a significant drop in the error curve can be seen. Actually, only neuron 4 and 5 have become important. In epoch 1,920 (figure 3) the decision function has become quadratic. At the end of the training cycle the neuron weights have almost completely separated, however some units still perform almost the same kind of function.

3 Conclusions

By using the SOM as a mapping technique for mapping the high dimensional weight space onto a two dimensional grid visualization of the weight space is possible. It can clearly be seen that there are different regions of density in that space. However, one has to bear in mind that the SOM is determined by the weights of the different networks, 10 in our case. This might not be a sufficient number because another network might explore another part of the space. But the experiment does 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 breaking of the clustering of the weights is observed in the different trajectories explored by the weights. First some of the trajectories remain in each others neighbourhood, but after some time their paths become different signaling reduction of redundancy.

4 Acknowledgments

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

References

[Ann94] Annema, A.J. *Analysis, Modeling and Implementation of Analog Integrated Neural Networks.* PhD thesis, Twente University, 1994.

[HKRS96] Hoekstra, A, Kraaijveld, M.A., Ridder, D. de, and Schmidt, W.F. *The Complete SPRLIB & ANNLIB.* Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, 1996.

[KHKL95] Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. *SOMPAK The Self-Organising Map program package.* Laboratory of Computer and Information Science, Helsinki University of Technology, Rakentajanaukio 2 C, SF-02150 Espoo, FINLAND, version 3.1 edition, April 7 1995.

[KMJ95] Kraaijveld, M.A., Mao, J., and Jain, A.K. A nonlinear projection method based on kohonen's topology preserving maps. *IEEE transactions on neural networks*, 6(6):548–559, 1995.

[Koh89] Kohonen, T. *Self-Organization and Associative Memory.* Springer-Verlag, Heidelberg, 3rd edition, 1989.

[Sch93] Schmidt, W.F. *Neural pattern classifying systems.* PhD thesis, Delft University of Technology, 1993.

[Vog94] Vogtländer, A.C. The learning behaviour of multilayer networks used as classifiers. Master's thesis, Faculty of Applied Physics, Delft University of technology, 1994.

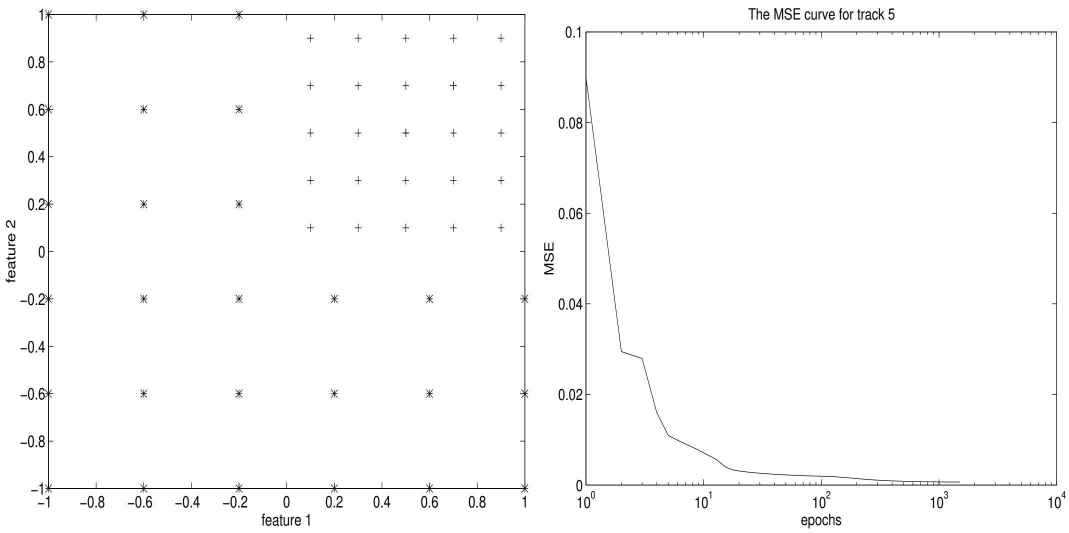

Figure 1: *The left figure depicts the dataset used for the experiment, the crosses and stars denote the different classes. The figure on the right shows the MSE curve for a particular realization of the weight, the drops in error agree with the splitting of paths in the last figure (SOM).*

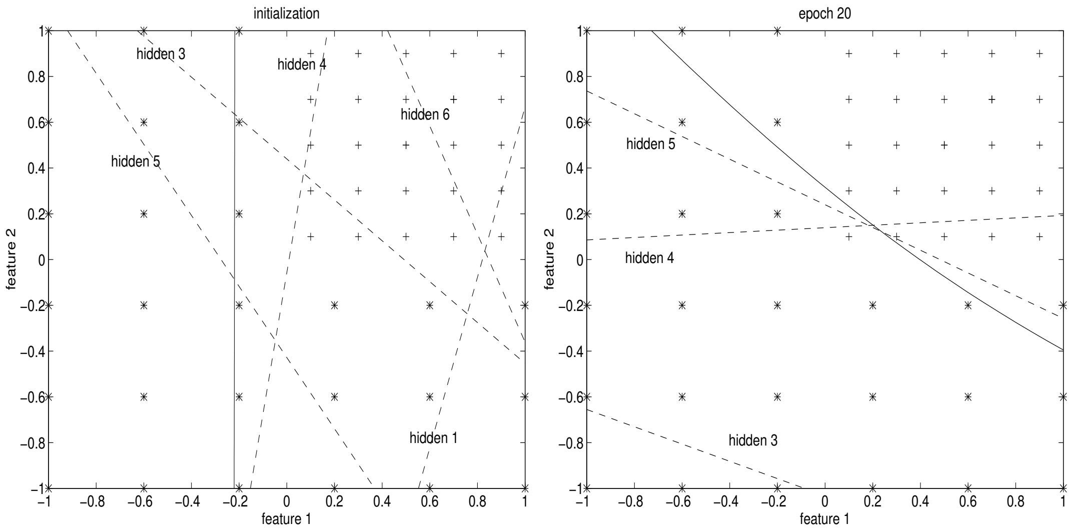

Figure 2: *Both figures show the decision function implemented by the network at different time stamps. The dashed lines with legend show the decision functions of the hidden neurons.*

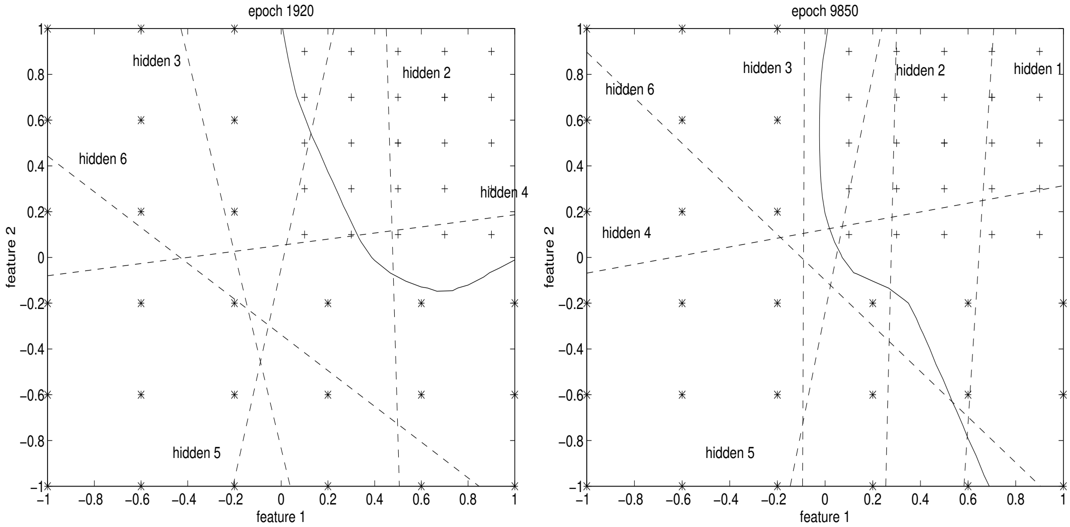

Figure 3: *Both figures show the decision function implemented by the network at different time stamps. The dashed lines with legend show the decision functions of the hidden neurons.*

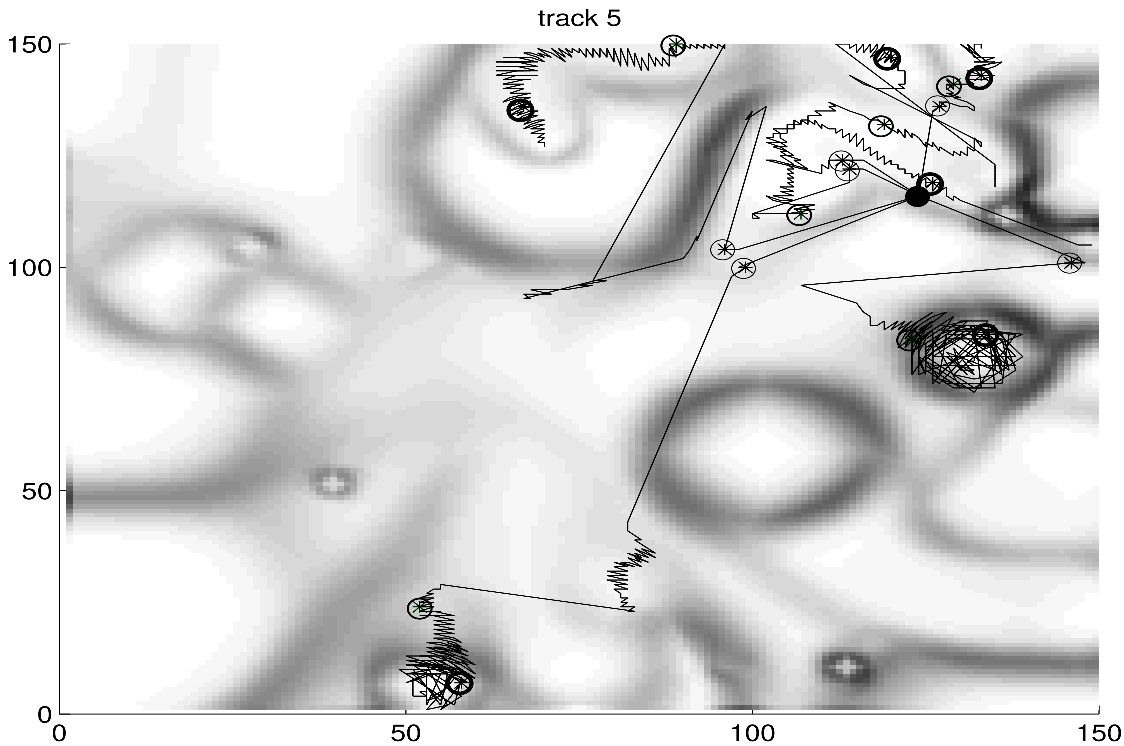

Figure 4: *The SOM image for all the weights, with the trajectories of a particular network drawn. The trajectories show the way the different neurons move through the space. The black circles correspond to the situations depicted in the previous figures (the fatter a circle the longer a network has trained).*