Classification through Hyperplane Fitting with Feedforward Neural Networks

Georg Dorffner
Austrian Research Institute for Artificial Intelligence
Neural Networks Group
Schottengasse 3, A-1210 Vienna, Austria
georg@ai.univie.ac.at
and
Dept. of Medical Cybernetics and Artificial Intelligence
University of Vienna

Abstract

This paper introduces and demonstrates a novel (or at least unusual) way of using feedforward neural networks for classification, inspired by a technique known from statistics. Usually, hidden units of a network are considered to define hyperplanes which *separate* clusters or sub-classes. The network defined in this paper uses the hyperplanes to *represent* clusters or sub-classes of points by fitting them as in regression. An update and gradient descent learning rule is defined, and results from XOR and the benchmark sonar data are used for illustration. The resulting network is reminiscent of some kind of "inverse" competitive learning or LVQ, and have different qualitative properties than multilayer perceptrons (e.g. linear separability is replaced by "linear regressability").

Keywords: Feedforward neural networks, hyperplanes, regression, classification, gradient descent, winner-take-all

1 Introduction

Among the many ways of depicting neural networks like multilayer perceptrons (MLPs), one of the most popular ones is the characterization in terms of hyperplanes splitting up the input data space, especially when they are used for classification (e.g. [Lippmann 1987]). The weighted sum (including bias term) of a hidden unit clearly defines an $n-1$-dimensional hyperplane in n-dimensional

space. These hyperplanes are seen as decision boundaries, many of which together can carve out complex regions necessary for classification of complex data. Only few people question or analyse this assumption. For instance, [Pratt & Christensen 1994] show that when training an MLP with backpropagation, the resulting hyperplanes sometimes do in fact separate data points clearly as one would suspect, but in many cases they do not. They explain this apparently paradoxical behavior through the continuous sigmoid function which can permit a distinction of classes even if the hidden units are not perfect separators. [Smieja 1992] analyses the dynamics of hyperplanes during backpropagation in a physical analogy and does come to the conclusion that they result in optimally separating data points.

In this paper we demonstrate that there is another quite distinct way of viewing and employing simple neural networks. In this method, the units' hyperplanes are used to *represent* data points rather than separate them. A learning rule is given, and some experiments with this learning rule are described. It turns out that this rather novel (or at least unusual) way of using neural networks enriches the set of classification methods when one is looking for techniques especially tuned to a particular application (i.e. a particular data distribution).

2 Hidden units as representing hyperplanes

It is well-known in statistics that classification cannot only be achieved by hyperplanes separating the classes, but also through hyperplanes *representing* the classes (or sub-classes), pretty much like in regression ([Bock 1974, Mosteller & Tukey 1977]). A hyperplane representing a sub-class of data points is defined as the hyperplane for which the sum of the quadratic orthogonal distances of all points is at a minimum. In the two-dimensional case this is identical to the regression line approximating that sub-class. Classification is then achieved by (a) assigning a class label to each hyperplane, and (b) for each data point choosing the closest hyperplane. This is in contrast to other ways of classification such as variations of nearest-neighbor or Voronoi tesselations, where a single point represents such sub-classes. On the other hand, the case of representing hyperplanes could be considered a more general case, if one permits hyperplanes of lesser dimensionality, i.e. defined through fewer degrees of freedom (e.g. lines in three-dimensional space), which would include single points, as well. As is pointed out in [Bock 1974], in this more general case the decision boundaries between the sub-classes (i.e., the boundaries where two hyperplanes are equally far in terms of orthogonal distance) are not necessarily hyperplanes again, but more general quadratic surfaces such as cones. Thus, this method of classification is not directly equivalent to classification through separating hyperplanes.

These observations led us to define representing hyperplanes in a neural network unit to construct such a classifier from a layered network. The difficult

problem in the above approach is to automatically find a sub-classification into clusters of data points and their appropriate representing hyperplanes. Viewing the method in the context of neural networks, especially through applying a gradient descent rule similar to backpropagation ([Werbos 1974, Rumelhart et al. 1986]), finding hyperplanes can be done through a learning algorithm. The crucial observation is that the net input y of a multilayer perceptron unit (i.e., the result of the propagation rule, which is the weighted sum of input values) is proportional to the orthogonal distance of the data point to the hyperplane defined by the weights and the bias of that unit. Thus, choosing the closest hyperplane can be done by choosing the unit with smallest absolute value of the net input, i.e., the smallest weighted sum. This somewhat reverses the usual way of picking a "winner" in a neural network layer, which is based on looking for the largest dotted product of the weight and input vectors ([Grossberg 1976]) or the smallest Euclidean distance ([Kohonen 1984]). To turn this into a "real" winner-take-all procedure, a Gaussian-type of transfer function can be chosen which assign an activation of 1 to a net input of 0, and asymptotically goes to 0 for increasing absolute values of the net input.

Learning can then be done in a way very similar to learned vector quantization (or LVQ,[Kohonen 1984]). The hidden layer of units comprising the hyperplanes is divided into as many clusters as there are classes to be recognized, and each of them is assigned to one of the classes. For each point in the training set the cluster corresponding to its class is considered, and winner-take-all is performed in the above manner. Then a learning rule (to be described below) is applied to the weights which turns and shifts the corresponding hyperplane such as to minimize the summed squared distance of all points leading to the same winner. For this to work, learning must proceed in the "batch" mode, i.e., weight increments are summed up and added to the weights after an entire epoch. Otherwise, minimization would always be done with respect to a single point, ignoring the structure of the sub-class to be fitted. After learning, novel input points are presented to the network and winner-take-all is performed on the whole hidden layer. The input is then assigned the class attached to the cluster the winner lies in. From this we can see that an output layer degenerates to fusing the activations of all units of a cluster, i.e. by introducing one unit for each class with positive weights to its corresponding cluster. Of course, winner-take-all can also be omitted leading to distributed responses. In this case, hidden-to-output weights can be trained by a simple delta rule, as is done for many variations of radial basis function networks (e.g. [Broomhead & Lowe 1988]).

3 The update and learning rules

An $n-1$-dimensional hyperplane H_j, defined by a unit j (using the weighted sum as propagation rule, as in the multilayer perceptron) is defined in n-dimensional

space through the equation

$$\sum_{i=1}^{n} w_{ij} x_i + c_j = 0 \tag{1}$$

where c_j is a constant and the vector $(w_{ij}), i = 1..n$ is orthognal to the hyperplane. It can be shown easily that the net input y of this unit

$$y_j = \sum_{i=1}^{n} w_{ij} x_i - \theta_j \tag{2}$$

with bias $\theta_j = -c_j$ is proportional to the orthogonal distance $d(H_j, \vec{x})$ of any point $\vec{x}$ to the hyperplane. To choose a hyperplane H_j to represent (or characterize – [Bock 1974]) a sub-class $A_{kl} \subseteq C_k$ (where C_k is one of m classes of the classification problem) one picks the one with the following property

$$\sum_{\{i|\vec{x}_i \in A_{kl}\}} d(\vec{x}_i, H_j)^2 \rightarrow \min \tag{3}$$

i.e., one that forms a multivariate regression of the sub-class. With the help of hyperplanes representing sub-classes, classification can be done for any arbitrary point $\vec{x}_i$ by choosing the hyperplane H_j with smallest absolute value of its orthogonal distance $d(\vec{x}_i, H_j)$ and returning the class k to which sub-class A_{kl} has been assigned to. Assuming a Gaussian transfer function for all hyperplane units, e.g.

$$x_j = e^{-\frac{y_j^2}{\sigma^2}} \tag{4}$$

with a freely chosen σ for all units, finding the closest hyperplane is identical to winner-take-all among the hyperplane units using activations x_j. To permit several sub-classes A_{kl} per class C_k, the layer of hyperplane units must be divided into m clusters.

The crucial point now is to find appropriate hyperplanes through learning. For this we assume a gradient descent rule which minimizes the squared orthogonal distances of all points I it represents, as in eq. 3. This is identical to minimizing the sum of squared net inputs for all those points. Similar to [Rumelhart et al. 1986] we can thus define the weight increment for learning as

$$\Delta w_{ij} \propto -\frac{\partial}{\partial w_{ij}} \left(\sum_{\{I|\vec{x}_i^{(I)} \in A_{kl}\}} \sum_{i=1}^{n} w_{ij} x_i^{(I)} - \theta_j \right)^2 \tag{5}$$

$$= -\eta \left(\sum_{\{I|\vec{x}_i^{(I)} \in C_k\}} \sum_{i=1}^{n} w_{ij} x_i^{(I)} - \theta_j \right) x_i \tag{6}$$

$$= -\eta y_j x_i \tag{7}$$

with a learning rate η (subsuming the factor 2 from the derivative). What results is a kind of anti-Hebbian rule using the net input of the postsynaptic unit and the activation of the pre-synaptic one. Similarly, it can be derived that

$$\Delta\theta_j = \eta y_j \tag{8}$$

These rules must be applied to the winner l in the k-th cluster representing sub-class A_{kl}. The assumption is that each hyperplane is changed so as to form a regression of all the points that are currently closest to it. This is a crucial assumption which might lead to non-optimal results if, through unfortunate initial weights, the hyperplane attempts to fit points that would rather be fitted by a different plane. This appears to be akin to local minima problems in regular backpropagation. In any case, learning must be done in the "batch" mode adapting weights only after computing increments for an entire epoch. Since the size of the weight vector influences the proportional dependency between orthogonal distances and the net input, normalization of the weight vectors after each epoch is advisable.

Of course, other methods to arrive at appropriate hyperplanes are conceivable, as well. Any estimation technique from regression theory (e.g. [Mosteller & Tukey 1977, Bock 1974]) can be used directly, before hyperplanes are translated into network weights. If appropriate sub-classifications are known, prinicipal component analsysis for each sub-class can lead to good approximations.

4 An example: XOR

The XOR mapping of two binary inputs onto one output is the simplest paradigmatic example to illustrate the function of a certain network type. It is interesting to view the network type described above in this context. This is done for instructive reasons and not as a proof for the power of this type of network.

With regular multilayer perceptrons, the well-known observation is that XOR is not linearly separable, therefore requiring hidden units and a non-linear transfer function. In the context of representing hyperplanes, the question instead is, whether points of one class can be *linearly fitted* well enough to distinguish them from points of the other class. In the case of XOR, the question can be answered in the positive. The points (1,0) and (0,1) of the class 1 can be perfectly represented by the straight line connecting them. A unit, whose weights form this line, with a (steep) Gaussian transfer function can thus respond maximally to these two points, and output a 0 (or a value very close to it) for the complementary class. It is clear, that non-linearity is also need here – in this case the Gaussian transfer function – but the interesting aspect is that the XOR can be solved without hidden units.

The next question that arises in this context is whether XOR is learnable with this network. It is well known that multilayer perceptrons trained with backprop-

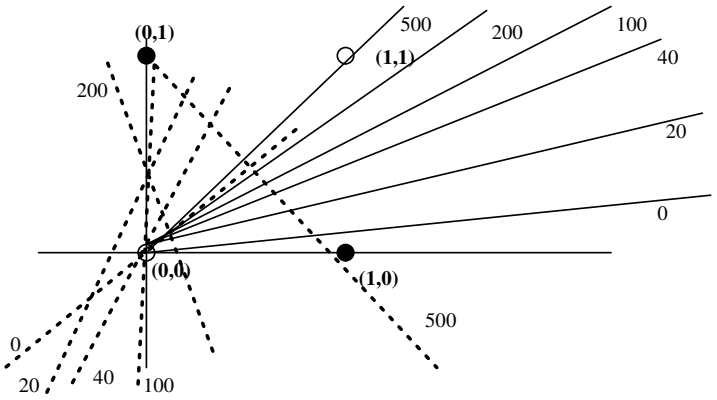

Figure 1: The evolution of hyperplanes (lines) during learning of the XOR problem. The figure next to each line indicates the number of epochs to reach this position

agation can consume extremely long learning times to approximate XOR. To test the learnability, a network with representing hyperplanes was implemented. A layer of two units was used, one for each class. (It should be noted that one unit would be sufficient, but this is the straight-forward application of the above-mentioned learning strategy. No additional output units are needed, since there is only one unit per class.) The network was trained with the above rule and a learning rate of $\eta = 0.01$ and converged within about 500 learning steps to a near to optimal representation of the problem. Figure 1 shows how the lines evolved through training.

5 A real-world example: Sonar data

To test the presented network type with real-world data, the well-known benchmark of [Sejnowski & Gorman 1988] was used. 208 sonar readings, encoded through 60 numbers, are to be classified into two groups ("mines" and "rocks"). A training set of 102 readings (angle dependent case), was taken here to train several networks with representing hyperplanes. The learning rate was varied between 0.001 and 0.5 and revealed erratic behavior at rates above 0.2 (constant increase of weights). The number of hidden units was varied between 4 and 20 (i.e., between 2 and 10 possible sub-classes per class). Both learning with and without normalization was tested, revealing little a definite advantage for normalization. Figure 2 depicts results (in terms of percentage correct as a function of epochs) for a network with 10 hidden units. While the training set could be fitted with up to 90 %, generalization hardly reached levels of 65 % (slightly better than naive guessing). Especially cases of class 2 (mines) were hard to predict.

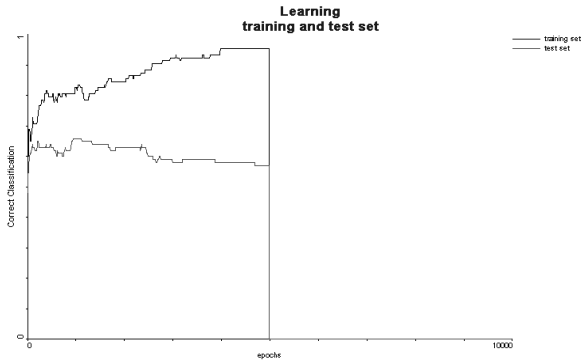

Figure 2: Percentage of correct classified cases for training set (upper curve) and test set (lower curve) of the sonar data as a function of epochs

6 Discussion

The results from the sonar data show that the learning rule indeed finds hyperplanes to fit the data, but also that generalization is very poor. We suggest that the reason for this might be one of the following.

- the sonar data might not be suited for this method
- the parameters might not have been tuned well enough yet.
- the gradient descent rule might be insufficient for finding hyperplanes which are good for generalization

Our intent was not to introduce a novel learning algorithm and prove that it is better than other ones. Our intent was to introduce another way of using neural network to implement traditional statistical techniques, in accordance with the analysis in the other direction by [Sarle 1994] and [Ripley 1992]. We do this in the realm of suggesting that for different types of data distribution different types of estimation techniques, and thus different types of neural networks, might be appropriate. Even though it can be proven that the well-known types multilayer perceptron and radial basis function network are universal function approximators ([Hornik et al. 1989, Kurkova 1992]), there is no proof as of yet as to which type is efficient in practical applications, is learnable, or generalizes well for given data distributions. Therefore, the larger the choice of different methods, the greater the probability of finding an appropriate method for a given application. The network presented in this paper enriches this set of possible methods.

7 Conclusion

In this paper, we demonstrated how neural networks can be trained so as to make the units' hyperplanes represent sub-classes in data space, rather than separate them. We devised a simple learning rule to minimize the summed orthogonal distance of all data points represented by a hyperplane. We illustrated the procedure with the famous XOR mapping, and empirically proved its validity by applying it to the benchmark sonar data. Even though for those data generalization is very poor, this new method appears to offer itself for efficient classification if the data distribution can be found to meet criteria of linear regressability.

8 Acknowledgements

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Minsitry for Science, Research, and the Arts.

References

[Bock 1974] Bock H.H.: *Automatische Klassifikation*, Vandenhoeck & Ruprecht, Goettingen, 1974.

[Broomhead & Lowe 1988] Broomhead D.S., Lowe D.: Multivariable Functional Interpolation and Adaptive Networks, *Complex Systems*, 2,321-355, 1988.

[Grossberg 1976] Grossberg S.: Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors, *Biological Cybernetics*, 21, 145-159, 1976.

[Hornik et al. 1989] Hornik K., Stinchcombe M., White H.: Multi-layer Feedforward Networks are Universal Approximators, *Neural Networks* 2, 359-366,1989.

[Kohonen 1984] Kohonen T.: *Self-Organization and Associative Memory*, Springer, Berlin, 1984.

[Kurkova 1992] Kurkova V.: Universal Approximation Using Feedforward Neural Networks with Gaussian Bar Units, in Neumann B.(ed.), *Proceedings of the Tenth European Conference on Artificial Intelligence* (ECAI92), Wiley, Chichester, UK, pp.193-197, 1992.

[Lippmann 1987] Lippmann R.P.: An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, 4(2)4-22., 1987.

[Mosteller & Tukey 1977] Mosteller F., Tukey J.W.: *Data Analysis and Regression - a second course in statistics*, Addison-Wesley, Reading, MA, 1977.

[Pratt & Christensen 1994] Pratt L.Y., Christensen A.N.: Relaxing the Hyperplane Assumption in the Analysis and Modification of Back-propagation Neural Networks, in Trappl R.(ed.), *Cybernetics and Systems '94*, World Scientific Publishing, Singapore, pp.1711-1718, 1994.

[Ripley 1992] Ripley B.D.: Statistical Aspects of Neural Networks, Department of Statistics, University of Oxford, 1992.

[Rumelhart et al. 1986] Rumelhart D.E., Hinton G.E., Williams R.J.: Learning Internal Representations by Error Propagation, in Rumelhart D.E., McClelland J.L.(eds.), *Parallel Distributed Processing*, MIT Press, Cambridge, MA, 1986.

[Sarle 1994] Sarle W.S.: Neural Networks and Statistical Models, Proceedings of the Nineteenth Annual SAS Users Group International Conference, 1994.

[Sejnowski & Gorman 1988] Sejnowski T.J., Gorman R.P.: Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets, *Neural Networks*, 1(1)pp.75-88, 1988.

[Smieja 1992] Smieja F.: Hyperplane "Spin" Dynamics, Network Plasticity and Back-Propagation Learning, GMD, Institut fuer Angewandte Informationstechnik, Arbeitspapier Nr.634, 1992.

[Werbos 1974] Werbos P.: Beyond regression: New tools for prediction and analysis in the behavioral sciences, Harvard University, Ph.D. Dissertation, 1974.