J.N. Hwang, J.J. Choi S. Oh & R.J. Marks II, "Classification boundaries and gradients of trained multilayer perceptrons", Proc. 1990 IEEE International Symposium on Circuits and Systems, (1-3 May, 1990, New Orleans , Louisiana) pp. 3256-3259.

Classification Boundaries and Gradients of Trained Multilayer Perceptrons*

Jenq-Neng Hwang, Jai J. Choi†, Seho Oh, Robert J. Marks II

Department of Electrical Engr., FT-10
University of Washington
Seattle, WA 98195

Abstract

This paper presents a novel approach for query based neural network learning. Consider a layered perceptron partially trained for binary classification. The single output neuron is trained to be either a 0 or a 1. A test decision is made by thresholding the output at, say, $\frac{1}{2}$. The set of inputs that produce an output of $\frac{1}{2}$, forms the classification boundary. We adopted an inversion algorithm for the neural network that allows generation of this boundary. In addition, for each boundary point, we can generate the classification gradient. The gradient provides a useful measure of the sharpness of the multi-dimensional decision surfaces. Using the boundary point and gradient information, conjugate input pair locations are generated and presented to an oracle for proper classification. This new data is used to further refine the classification boundary thereby increasing the classification accuracy. The result can be a significant reduction in the training set cardinality in comparison with, for example, randomly generated data points.

1 Introduction

Multilayer perceptrons are feed-forward neural networks, which have one or more layers of *hidden neurons* between the input and output layers. When a network is trained, it is used to generate output response vector given the input test vector. The converse problem of generating input vectors corresponding to a given output vector is referred to as *inversion*. The inversion of a network midway between two classifications results in a classification boundary. This boundary is the locus of vector that, with respect to the neural network's representation of the training data, is highly confusing. Evaluation of the gradient at these points is a measure of the classification sensitivity there.

The system dynamics in the retrieving phase of an L-layer neural net can be described by the following equations:

$$u_i(l+1) = \sum_{j=1}^{N_l} w_{ij}(l+1)a_j(l) + \theta_i(l+1) \quad (1)$$

$$a_i(l+1) = f(u_i(l+1)) \quad 1 \le i \le N_{l+1}, \quad 0 \le l \le L-1 \quad (2)$$

where $a_j(l)$ denotes the activation value of the j^{th} neuron at the l^{th} layer and f is the nonlinear activation function.

Learning of the Multilayer Perceptrons The learning phase of a multilayer perceptron uses the back propagation (BP) learning rule, an iterative gradient descent algorithm designed to minimize the mean squared error between the the desired target values and the actual output values [1]:

$$w_{ij}(l) \Longleftarrow w_{ij}(l) - \eta \frac{\partial E}{\partial w_{ij}(l)} \quad (3)$$

where $E = \frac{1}{2} \sum_{i=1}^{N_L} (t_i - a_i(L))^2$.

*This work is supported through grants from the National Science Foundation and the Washington Technology Center. Discussions with L. Atlas, M. A. El-Sharkawi, D. Cohen and R. Ladner are gratefully acknowledged.

†J. J. Choi was supported, in part, by a developmental grant from Physio Control Inc.

Inversion of the Multilayer Perceptrons The inversion of a network will generate the input $\{a_j(0)\}$ (or inputs) that can produce the desired output vector (see Figure 1). By taking advantage of the *duality* between the weights and the activations in minimizing the mean squared error between the the desired target values and the actual output values, the iterative gradient descent algorithm can also be applied to obtain the desired input.

$$a_j(0) \Longleftarrow a_j(0) - \eta \frac{\partial E}{\partial a_j(0)} \quad (4)$$

The idea is similar to the BP algorithm, where the error signals are propagated back to tell the weights the manner in which to change in order to decrease the output error. The inversion algorithm back-propagates the error signals down to the input layer to update the activation values of input units so that the output error is decreased [2].

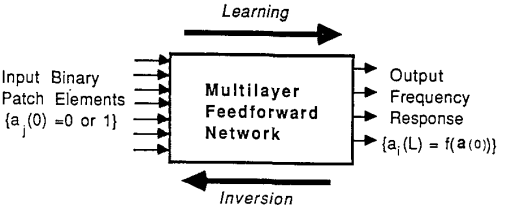

Figure 1: Learning and inversion of a multilayer perceptron

2 Boundary Search and Gradient Computation

Boundary Search for Region of Maximum Ambiguity After a multilayer perceptron is well trained by the BP algorithm, it is very instructive to locate the *classification boundary* (or the region of maximum ambiguity). Without loss of generality and also for simplicity of illustration, only a binary classification (two class) example is given below. We want to train a single output neuron, $a_1(L)$, to be either "0" or "1". A good strategy is to evaluate the boundary corresponding to an output value of 0.5. This can be done by inverting the network with several randomly selected initial input data points. The inversion algorithm will take some trajectory and gradually move each initial input data toward one specific boundary point.

A neural network with a single hidden layer of 10 neurons was trained with 50 randomly selected 2-dimensional data for hexagonal classification region (see Figure 2(a)). A perspective plot of the classification region is shown in Figure 2(b). The classification target is "1" inside the hexagon and "0" without. After 3000 iterations of the training, 30 classification boundary points corresponding to an output of 0.5 were computed using

the inversion algorithm (see Figure 3(a)). We would expect this boundary becomes closer to the target hexagon as the size of the training set increases (or sometimes the length of the training iteration increases). For classification of multiple classes, where each class (say, the k-th) is represented by the k-th output neuron, the *boundary* (or the region of maximum ambiguity) for the k-th class can be also located by searching the points (in the input space) which give rise to 0.5 activation value for the k-th output neuron regardless the values of other output neurons. To be more specific, the E measure in Eq. 4 is computed based only on the square difference between the target value and actual value of k-th output neuron;

$$E = \frac{1}{2}(t_k - a_k(L))^2.$$

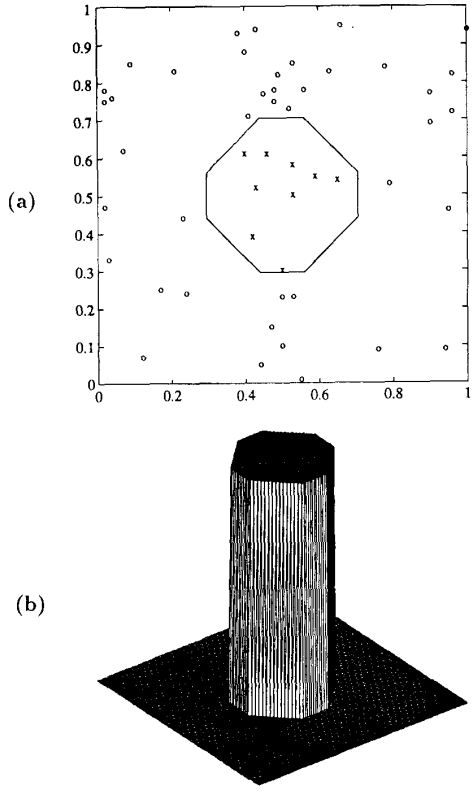

Figure 2: (a) A neural network for hexagonal classification region. (b) A perspective plot of the classification region.

Gradient Computation for Sensitivity of Functional Mapping After the neural network is well trained, the functional mapping relationship between the input and output is established through the weights. For each point in the input space, we can compute the gradient, $\rho_{kj}(L)$, of each output neuron (*e.g.* the k-th), with respect to each input neuron (*e.g.* the j-th),. This gradient is a measure of the sensitivity of the functional mapping relationship. This gradient,

$$\varrho_{kj}(L) = \frac{\partial a_k(L)}{\partial a_j(0)}, \tag{5}$$

can be recursively computed from

$$\varrho_{ij}(l) = f'(u(l)) \sum_{k=1}^{N_{l-1}} w_{ik}(l)\varrho_{kj}(l-1), \quad 2 \le l \le L \tag{6}$$

where the initial values are given as $\varrho_{kj}(1) = f'(u_k(1))w_{kj}(1)$.

Figure 3(b) illustrates the magnitude of gradient of each inverted boundary point shown in Figure 3(a).

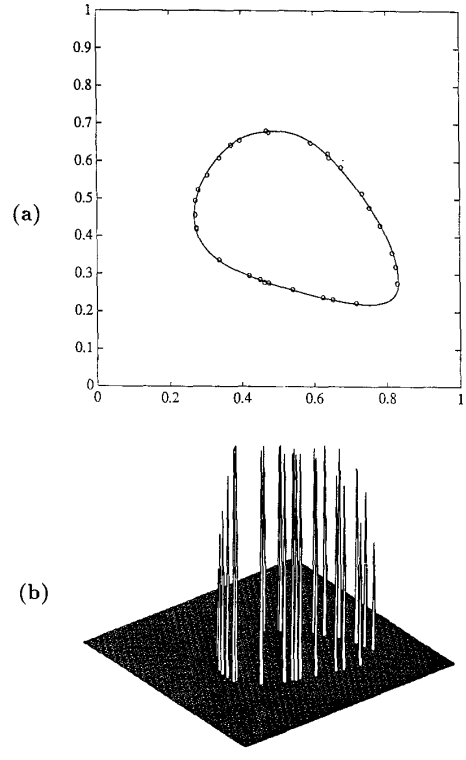

Figure 3: (a) 30 classification boundary points corresponding to an output of 0.5 were created. (b) The magnitude of gradient of each inverted boundary point.

3 Applications to Query Learning

In many machine learning applications, the source of the training data can be modeled as an *oracle*. An oracle has the ability, when presented with an example, to give a correct classification. A cost, however, is typically associated with this query. The study of queries in classifier training paradigms is therefore a study of the manner in which oracles can provide the good classifier training data at low cost.

One method of query learning makes use of a partially trained classifier. The classifier locates points of confusion which the classifier has difficulty in classifying. These data points are then taken to the oracle for proper classification. The properly classified points are then introduced as additional training data for the classifier. The use of queries through systematic data generation mechanism can be viewed as an *interactive* learning. On the other hand, the use of only available (or randomly generated) data, is a *passive learning*. If properly done, the use of queries can reduce the cost of data drastically from the case where examples are generated at random [3].

The neural network hexagonal classification region example given in Figure 2(a) is again used for the query learning illustration. There are 50 randomly selected training data used for training. After 5000 iterations of the training, 57 classification boundary points corresponding to an output of 0.5 were created using the inversion algorithm and is shown in Figure 4(a). For each inverted boundary point, a conjugate training data pair based on the magnitude of gradient were also created (see Figure 4(b)). More specifically, two points lying on opposite sides of the line passing through the boundary point and perpendicular to the boundary surface are located, with their distances to the corresponding boundary point equal to $1/|\varrho_{kj}(L)|$. The motive behind the selection of this conjugate pair comes from the desire to sharpen the boundary between two distinct classes by narrowing the regions of ambiguity. The oracle will provide true classification (1 or 0) for those newly generated query points. If all three points (the boundary point as well as the conjugate pair) fall into the same class, we neglect the conjugate pair and keep only the boundary point as part of the new training data. Otherwise, we choose all three points to be part of training data. This results in a total of 137 newly generated training data, which will be used to retrain the network along with the originally randomly selected 50 data. Figure 5(a) shows the created conjugate pairs which have different classifications and the boundary points whose conjugate pairs are classified as the same classes. Figure 5(b) shows the dramatic improvement of this query learning. The result of query learning was much more favorably as compared with that of conventional learning based on purely randomly selected training data with the same data set cardinality (see Figure 6(a)) or even a larger size (500 points) of training data (see Figure 6(b)). This query learning has also been applied to real world power security problems, which involves the solution of a fairly complicated nonlinear programming problem [4]. It is shown that as the training size increases, the relative advantage of the query based system increases significantly.

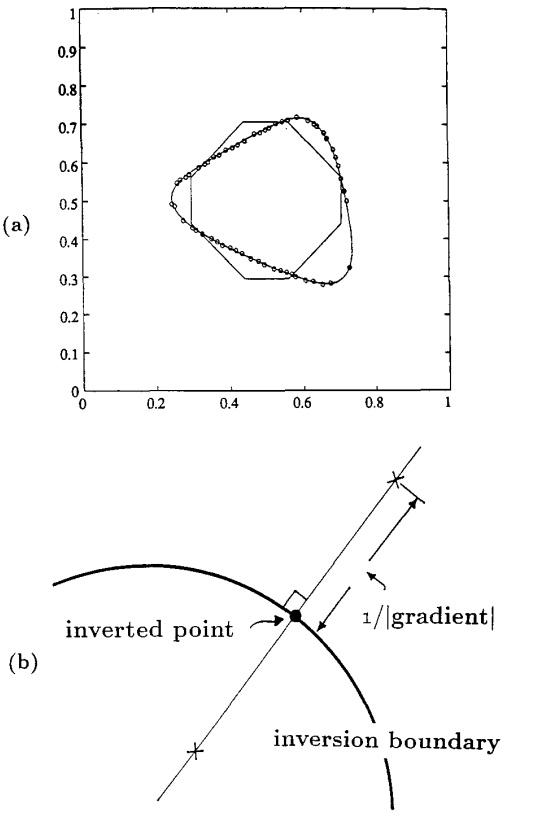

Figure 4: (a) 57 Classification boundary points. (b) A conjugate training data pair based on gradient were created.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing (PDP): Exploration in the Microstructure of Cognition (Vol. 1)*, Chapter 8, pages 318–362. MIT Press, Cambridge, Massachusetts, 1986.

[2] A. Linden and J. Kindermann. Inversion of multilayer nets. In *Proc. Int'l Joint Conf. on Neural Networks*, II 425-430, Washington D.C., June 1989.

[3] D. Cohn, L. E. Atlas, R. Ladner, R. J. Marks II, M. El-Sharkawi. M. Aggoune, and D. C. Parks. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems*, Denver, November 1989.

[4] J. N. Hwang, J. J. Choi, S. Oh, R. J. Marks II. Query learning based on boundary search and gradient computation of trained multilayer perceptrons. Submitted to *Int'l Joint Conference on Neural Networks*, San Diego, June 1990.

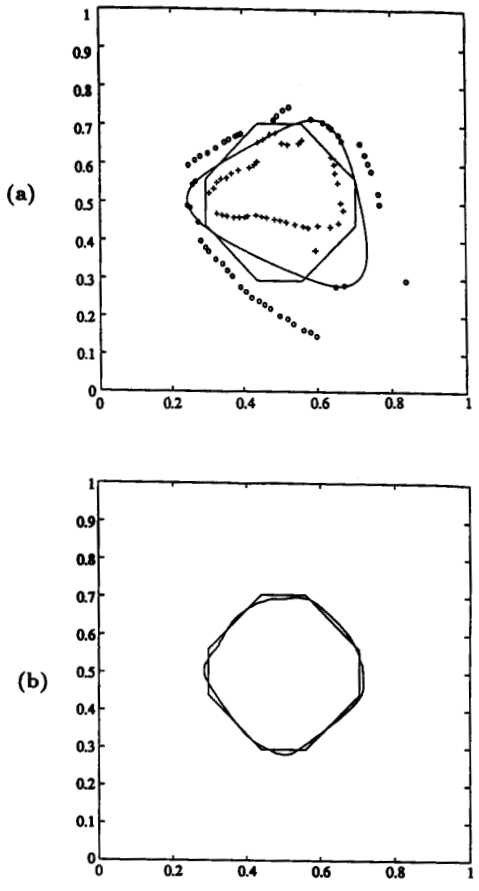

Figure 5: (a) The conjugate pairs which have different classifications and the boundary points whose conjugate pairs are classified as the same classes. (b) The dramatic improvement of the query learning.

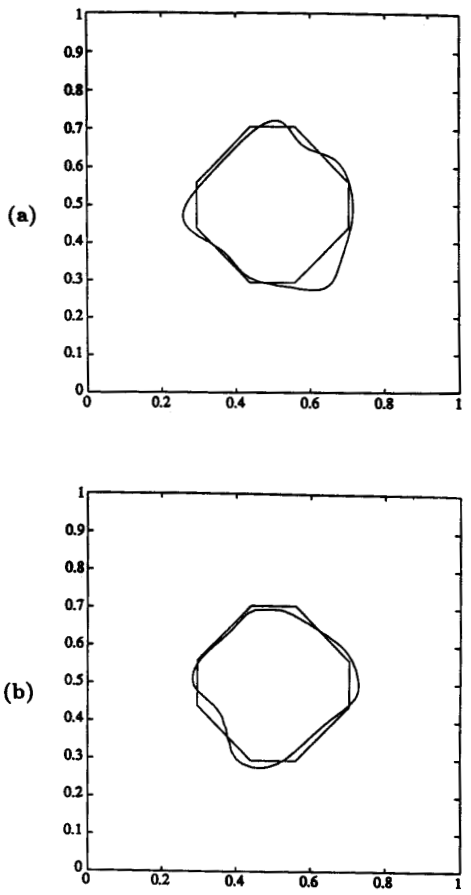

Figure 6: (a) Learning based on purely randomly selected training data with same data set cardinality as query based learning. (b) Learning based on purely randomly selected training data with a larger size of training data.