J.N. Hwang, J.J. Choi, S. Oh and R.J. Marks II, "Query based learning applied to partially trained multilayer perceptrons", IEEE Transactions on Neural Networks, Vol. 2, pp.131-136, (1991).

Query-Based Learning Applied to Partially Trained Multilayer Perceptrons

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

***Abstract*—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 (*query*), to give a correct classification. An efficient query learning procedure is to provide the good training data to the oracle at low cost. 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 steepness of the multidimensional decision surfaces. Using the boundary point and gradient information, conjugate input pairs are generated and presented to an oracle for proper classification. These new data are 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. An application example to power system security assessment is given.**

I. Introduction

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

Query-based learning requires asking a partially trained classifier to respond to the question, "What don't you yet understand?". The response of the query is then taken to the *oracle*. The oracle, for a price, will respond with the correct classification for a given data point. Examples of oracles include computationally intensive simulators, costly experimentation, or a human expert. The properly classified points from the oracle are then introduced as additional training data for the classifier. The use of queries through such a systematic training data generation mechanism can be viewed as *interactive learning*. On the other hand, only the use of available (or randomly generated) data, is *passive learning*. In certain problems, when applied properly, queries can significantly increase the resulting classification accuracy with a small amount of additional complexity introduced [1].

Manuscript received May 29, 1990; revised September 21, 1990. This work was supported in part through grants from the National Science Foundation under Contract no. 62-6273 and the Washington Technology Center under Contract no. 09-1051.

J.-N. Hwang, S. Oh, and R. J. Marks II are with the Department of Electrical Engineering, FT-10, University of Washington, Seattle, WA 98195.

J. J. Choi was with the Department of Electrical Engineering, University of Washington, Seattle. He is now with Boeing Computer Services, Seattle, WA 98195.

IEEE Log Number 9040685

In this paper, we consider the use of queries in the training of a partially trained multilayer perceptron. We propose to use the inversion algorithm, which allows generation of the network input (or inputs) that can produce any specified output vector. For binary classifiers, inversion of a network midway between the two classifications (i.e., 0 and 1) results in a classification boundary. This boundary is the locus of input vectors that, with respect to the *neural network's representation* of the training data, is highly confusing. In addition, for each boundary point, we can generate the classification gradient. The gradient provides a useful measure of the steepness of the multidimensional decision profile. Using the boundary point and gradient information, conjugate input pairs are generated, which are supposed to carry the most important information (maximum confusion) in locating the correct decision boundaries [2], [3]. These points of confusion are presented to the oracle for proper classification. The choice of whether or not to use an oracle is dependent on the degree of training and classification complexity. Our approach to query-based learning works best when the network's performance is being fine tuned.

The organization of this paper is as following. In Section II, the backpropagation learning and network inversion algorithms are briefly discussed. Section III introduces the boundary search procedure from network inversion and the recursive formula for gradient computation. A simple toy problem for binary classification based on this query learning is discussed in Section IV. Finally, we apply this query learning to a power system security assessment problem in Section V.

II. Learning and Inversion of a Multilayer Perceptron

The forward system dynamics in the retrieving phase of an L-layer perceptron can be described by the following iterative equations (for $1 \le i \le N_{l+1}$, $0 \le l \le L-1$):

$$u_i(l+1) = \sum_{j=1}^{N_l} w_{ij}(l+1)a_j(l) + \theta_i(l+1) = \sum_{j=0}^{N_l} w_{ij}(l+1)a_j(l)$$

$$a_i(l+1) = f(u_i(l+1)) \tag{1}$$

where $a_j(l)(u_j(l))$ denotes the activation value (net input) of the jth neuron at the lth layer; $\theta_j(l)$ (or $w_{i0}(l)$) denotes the bias of the jth neuron at the lth layer; $w_{ij}(l)$ denotes the weight value linked between the ith neuron at the lth layer and the jth neuron at the $(l-1)$th layer; and f is the nonlinear activation function.

A. Backpropagation Network Learning

The learning phase of a multilayer perceptron uses the backpropagation learning rule, an iterative gradient descent algorithm designed to minimize the mean squared error E between

1045-9227/91/0100-0131$01.00 © 1991 IEEE

the desired target vector $\{t_i\}$ and the actual output vector $\{a_i(L)\}$, [4], [5]:

$$w_{ij}(l) \Leftarrow w_{ij}(l) - \eta \frac{\partial E}{\partial w_{ij}(l)} = w_{ij}(l) - \eta \frac{\partial E}{\partial a_i(l)} \frac{\partial a_i(l)}{\partial w_{ij}(l)} = w_{ij}(l) - \eta \delta_i(l) \frac{\partial a_i(l)}{\partial w_{ij}(l)} \quad (2)$$

where

$$E = E\left(\{w_{ij}(l)\}, \{a_j(0)\}\right) = \frac{1}{2} \sum_{i=1}^{N_L} \left(t_i - a_i(L)\right)^2 \quad (3)$$

and the backpropagated error signal $\delta_i(l)$ can be recursively calculated (see Fig. 1);

$$\delta_i(l) = \frac{\partial E}{\partial a_i(l)} = \sum_{j=1}^{N_{l+1}} \frac{\partial E}{\partial a_j(l+1)} \frac{\partial a_j(l+1)}{\partial a_i(l)} = \sum_{j=1}^{N_{l+1}} \delta_j(l+1) \frac{\partial a_j(l+1)}{\partial a_i(l)} \quad (4)$$

with the initialization error signal $\delta_i(L) = \partial E / \partial a_i(L) = -(t_i - a_i(L))$.

B. Network Inversion

The inversion of a network will generate the input vector $\{a_j(0)\}$ that can produce a desired output vector. By taking advantage of the *duality* between the weights and the input activation values in minimizing the mean squared error E [see (3)], the iterative gradient descent algorithm can again be applied to obtain the desired input vector [6]:

$$a_j(0) \Leftarrow a_j(0) - \eta \frac{\partial E}{\partial a_j(0)} = a_j(0) - \eta \delta_j(0). \quad (5)$$

The idea is similar to the backpropagation 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 backpropagates the error signals to the input layer to update the activation values of input units so that the output error is decreased (see Fig. 1). In order to avoid the input activation values $\{a_j(0)\}$ from growing without limits, a small modification of the updating rule was usually made.

$$u_j(0) \Leftarrow u_j(0) - \eta \frac{\partial E}{\partial u_j(0)} = u_j(0) - \eta \frac{\partial E}{\partial a_j(0)} \frac{\partial a_j(0)}{\partial u_j(0)} = u_j(0) - \eta \delta_j(0) \frac{\partial a_j(0)}{\partial u_j(0)} \quad (6)$$

where $u_j(0) = f^{-1}(a_j(0))$ is a "pseudo" net input created to allow flexible gradient descent search without limiting the dynamic ranges (e.g., usually we assume $0 \le a_j(0) \le 1$).

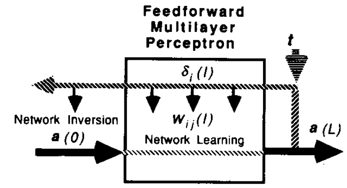

Fig. 1. Illustration of learning with and inversion of a multilayer perceptron.

III. Boundary Search and Gradient Computation

Neural networks have been shown to train better with boundary vicinity data in certain applications [3]. The trained neural network gives an *parametric representation* of the true classification or *concept* [7]. The representation's classification boundary indicates the region of maximum classification ambiguity; and the representation's classification gradient measures the steepness there [8], [9].

A. Boundary Search for Regions of Maximum Ambiguity

Without loss of generality and for simplicity of illustration, a binary (two-class) classification example is given first. We want to train a multilayer perceptron with a single output neuron to classify two types of patterns, i.e., $a_1(L)$ is 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 trained network with several randomly selected initial input data points. The inversion algorithm will progress along some trajectory (gradient descent search) and gradually move each initial input data toward one specific boundary point [8], [9].

To illustrate inversion, a two-layer perceptron with a single hidden layer of 10 neurons was trained with 50 randomly selected two-dimensional training data for octagonal region classification, where the classification target is "1" inside the octagon and "0" without [see Fig. 2(a)]. A perspective plot of a true classification profile is shown in Fig. 2(b). After 5000 iterations of the training based on the 50 available randomly selected training data, the neural network was trained to possess the representation profile shown in Fig. 3(a). There are 57 classification boundary points, corresponding to an output of 0.5, created using the inversion algorithm. These are shown in Fig. 3(b).

For classification of multiple classes, where each class (say, the kth) is presented by the kth output neuron, the *boundary* (or the region of maximum ambiguity) for the kth class can also be determined by searching the points (in the input space) which give rise to 0.5 activation value for the kth output neuron regardless the values of other output neurons. In other words, the E measure in (6) is computed based only on the square difference between the target value and actual value of kth output neuron.

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

Consider the example where three-layer perceptron (two input neurons, two hidden layers with 5 and 7 neurons, and three output neurons) was trained for a four-class (three rectangles and one background) classification problem, as shown in Fig. 4(a). More specifically, each two-dimensional data input will be classified as one of the four different classes in terms of three output neurons' activation values, i.e., $A = 100$, $B = 010$, $C = 001$, and $D = 000$. There are 450 uniformly sampled, ran-

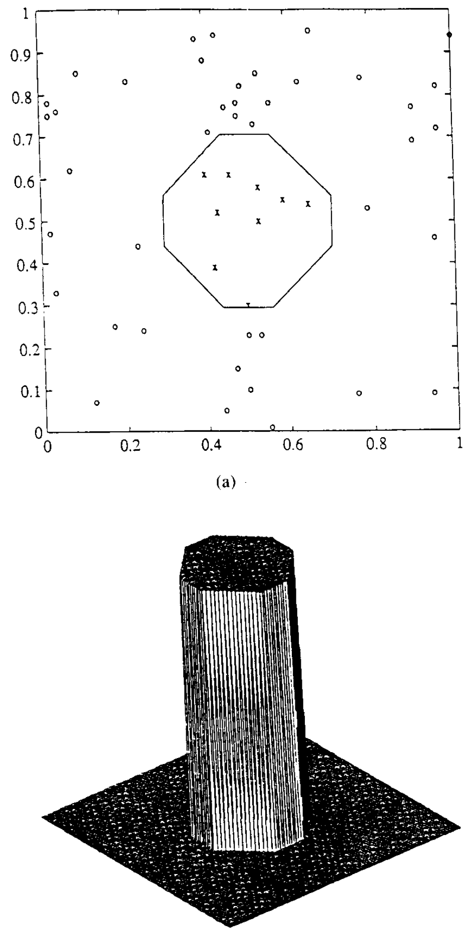

Fig. 2. (a) Fifty randomly selected two-dimensional training data are selected for octagonal region classification. (b) A perspective plot of the octagonal region classification profile.

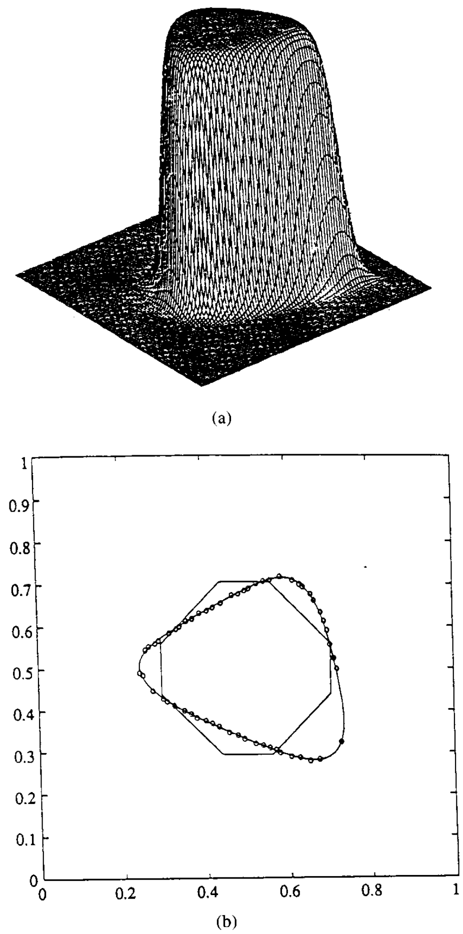

Fig. 3. (a) A neural network was trained with the data in Fig. 1(a) for 5000 iterations. This is the trained neural network's representation of the octagonal region classification profile. (b) Fifty-seven classification boundary points corresponding to the generalization thresholded at 0.5. These are the points of highest confusion.

domly selected two-dimensional data used for the training. After 10 000 iterations, 100 classification boundary points were created for each rectangle corresponding to an output value of 0.5 at each specific neuron regardless the output values of other neurons [see Fig. 4(b)].

B. Gradient Computation for Steepness of Classification Profile

After a neural network is trained, the parametric mapping relationship between the input and output is established through the weights. For each point in the input space, we can compute the gradient $\rho_{kj}(0)$ of each output neuron (e.g., the kth) with respect to each input neuron (e.g., the jth). At the boundary, this gradient is a measure of the steepness there. This gradient [8], [9],

$$\varrho_{kj}(0) = \frac{\partial a_k(L)}{\partial a_j(0)} \tag{7}$$

can be recursively computed (based on a simple chain rule) from

$$\varrho_{ij}(l) = \sum_{m=1}^{N_{l+1}} \varrho_{im}(l+1) f'(u_m(l+1)) w_{mj}(l+1), \quad 1 \le l \le L-1 \tag{8}$$

where the initial values are given as $\varrho_{ij}(L) = 1$ if $i = j$; 0 otherwise.

Fig. 5 illustrates the magnitude of the gradients of 30 points from the 0.5 contour of Fig. 3(a), i.e.,

$$|\text{gradient}| = \sqrt{\sum_j \varrho_{1j}(0)^2}.$$

IV. Conjugate Pairs for Refining the Boundary

For each inverted boundary point, a conjugate training data pair based on the magnitude of gradient can be created [see Fig. 6(a)]. More specifically, two points lying on opposite sides of

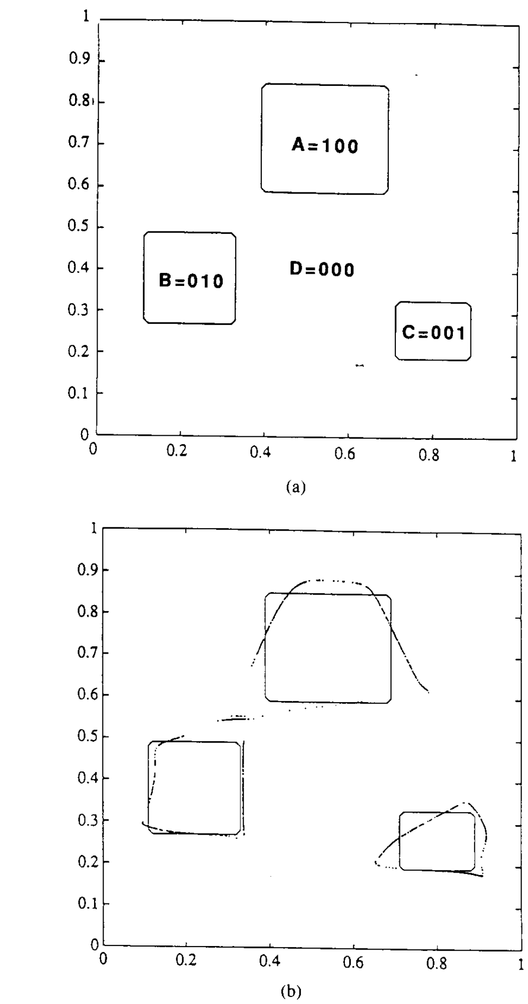

Fig. 4. (a) A four-class classification problem consists of three rectangular regions. (b) After 10 000 iterations of the training, 100 classification boundary points were created for each rectangle.

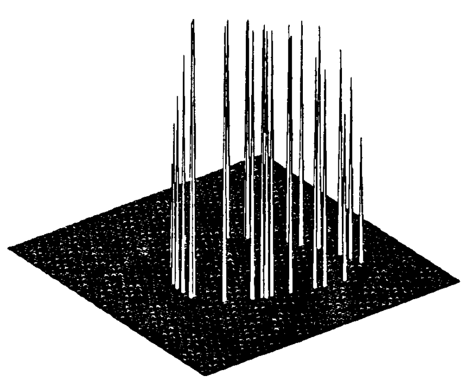

Fig. 5. The magnitude of the gradients of 30 points from the 0.5 contour of Fig. 3(a).

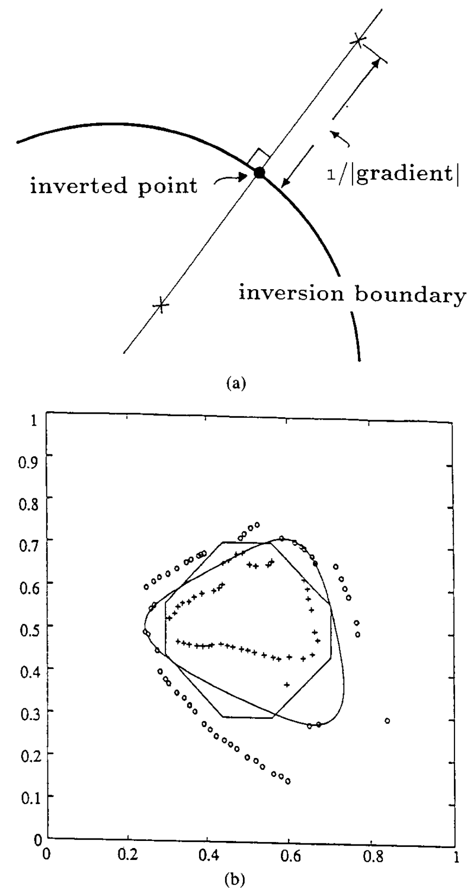

Fig. 6. (a) For each inverted boundary point, a conjugate training data pair based on the magnitude of gradient were created. (b) Conjugate pairs which have different classifications and the boundary points whose conjugate pairs are classified in the same class.

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 $1/|\text{gradient}|$. This idea is very similar in concept to the use of *close-opposed* pairs in locally trained piecewise linear classifiers [2]. This perpendicular line can be parametrically represented as:

$$\frac{x_1 - z_1}{\varrho_{k1}(0)} = \frac{x_2 - z_2}{\varrho_{k2}(0)} = \cdots = \frac{x_n - z_n}{\varrho_{kn}(0)}.$$

Use of only the boundary points for additional training information typically leads to a biased emphasis of one side of boundary. We therefore adopt the two-sided conjugate pairs for boundary fine-tuning. The motive behind the selection of the conjugate pairs stems from our desire to increase the boundary steepness between two distinct classes by narrowing the regions of ambiguity. The oracle will provide true classification (1 or 0) for the newly generated conjugate pair 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 the new training

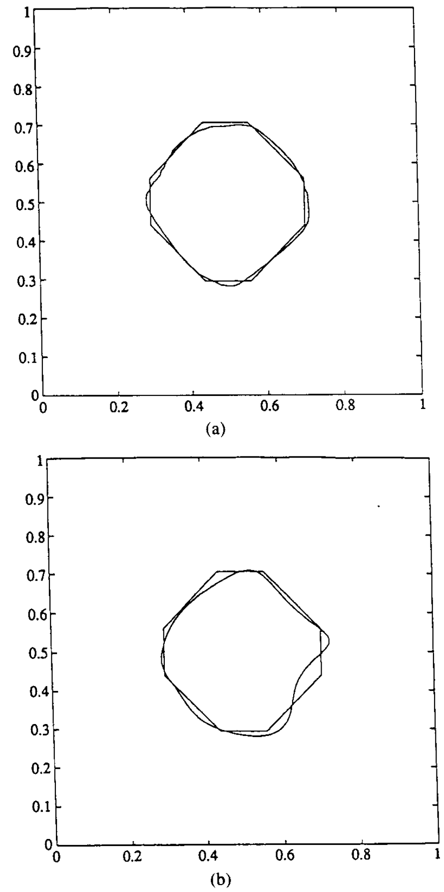

Fig. 7. Illustration of the result of using query-based learning. (b) Learning based on purely randomly selected training data with a larger size of training data.

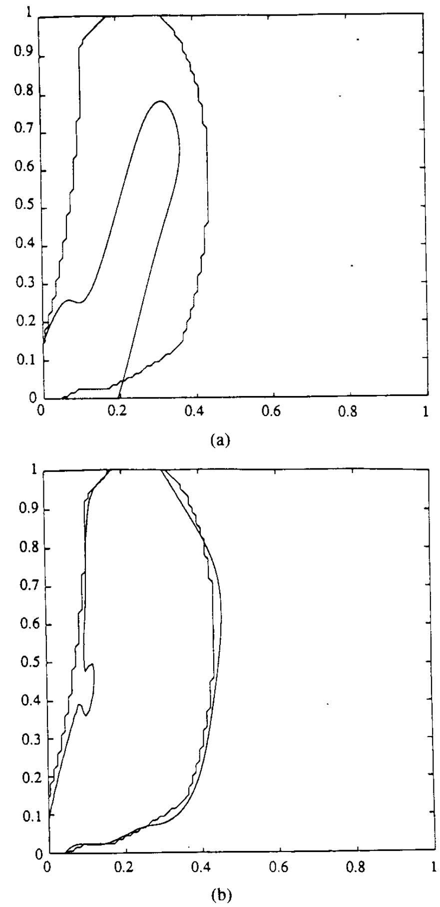

Fig. 8. The boundary trace result for the third randomly selected two-dim slice out of a four-dim power security assessment problem. (a) Neural network generalization after being trained by 5000 randomly selected training data. (b) Corresponding neural network generalization after being trained by 5000 query-selected training data.

data. To illustrate, we continue with the octagonal example, a set of 137 training data were newly generated and used to retrain the network along with the originally randomly selected 50 training data. Fig. 6(b) shows the created conjugate pairs which have different classifications and the boundary points whose conjugate pairs are classified in the same class.

Fig. 7(a) shows the dramatically improved result after retraining based on the 187 (= 50 + 137) data points. The result of query learning is strikingly better in comparison with conventional learning using 500 points of randomly selected training data [see Fig. 7(b)]. To get the best performance, the above query-based procedure can be repetitively applied in the training so that the classification accuracy can be gradually improved.

V. Application to Power System Security Problem

One of the main aspects of *power system security* is static or steady-state security. This is defined as the ability of the power system, after a disturbance such as a line break or other rapid load change, to reach a safe state that does not violate any operating constraints [10].

Defining multidimensional static-security regions for even a small power network is a computationally demanding task. It involves the solution of a nonlinear programming problem with a large number of variables and an equally large number of limit constraints which define the feasible region of operation [11]. In addition, the amount of memory required to store the security status under each probable network configuration is equally prohibitive. This leads us to the solution by artificial neural networks [12].

A three-layer perceptron consists of four inputs, two hidden layers (with 20 and 10 units separately), and one output is used to train a four-dimensional power system security assessment problem, where a four-dimensional input vector (representing the voltages of the power lines) is tested to determine whether the power plant is in a "secure" status or not.

To illustrate how well the query-based learning identifies the true boundary, a randomly selected two-dimensional slice of the four-dimensional power security assessment problem is used for graphical presentation. Fig. 8(a) shows a two-dimensional slice of the true boundary and the resulting representation boundary trained with 5000 randomly selected training data. On the other

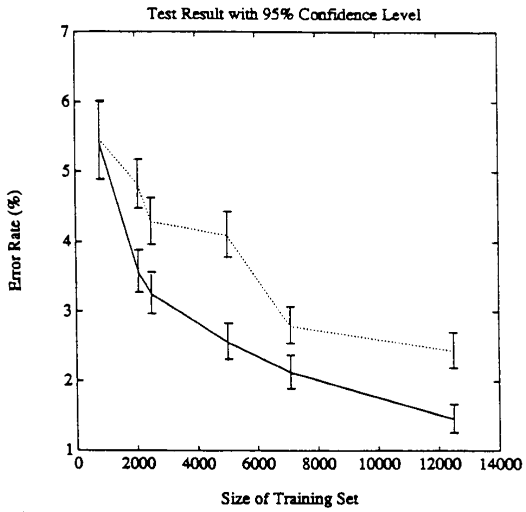

Fig. 9. The misclassification rate as a function of the training data set cardinality.

hand, Fig. 8(b) shows the corresponding boundary trace that trained with same amount of query-selected training data, which consist of 500 initially randomly selected training data and 4500 conjugate training pairs data via two consecutive steps of queries from the oracle. The oracle is a computationally intensive software simulator based on the algorithm presented in [13]. Fig. 9 shows the misclassification rate of 20 000 independent test data as a function of the training data set cardinality. The dotted line indicates the improvement of query-based learning and the solid line indicates the improvement of randomly selected data learning. Note that, in the last stage of boundary fine-tuning (the most difficult stage for pattern recognition tasks), the query learning is able to improve the accuracy drastically with a significantly smaller amount of additional query data compared to the improvement by randomly selected data. To be more specific, with 5000 query training data, we are able to achieve almost the same performance with 13 000 randomly selected training data.

VI. Conclusion

In many cases of interest, evaluation of classification solutions can be very complicated and time consuming. Query learning starts with a partially trained artificial neural network. Data point locations with high information content can be chosen through a boundary search and gradient computation iterative procedure. These points of uncertainty or confusion are then evaluated by an oracle to determine their status. The answered query clarifies this confusion. The clarification is incorporated into the training process to further refine the performance of the neural network. The empirical result of our query-learning examples illustrates that, in certain cases, a much more favorable performance results as compared with that of conventional learning based on purely randomly selected training data. This can be true at even a significantly higher training data set cardinality.

Acknowledgment

Valuable discussions with Professors M. A. El-Sharkawi, L. Atlas, and R. Ladner at the University of Washington are gratefully acknowledged.

References

[1] L. Atlas *et al.*, "Training connectionist networks with queries and selective sampling," in *Advances in Neural Information Processing Systems*. Los Altos, CA: Morgan Kaufman, 1990, pp. 566–573.

[2] J. Sklansky and L. Michelotti, "Locally trained piecewise linear classifiers," *IEEE Trans. PAMI*, vol. 2, no. 2, pp. 101–111, Mar. 1980.

[3] M. Wann, T. Hediger, and N. N. Greenbaun, "The influence of training sets on generalization in feed-forward neural networks."in *Proc. Int. Joint Conf. Neural Networks*, San Diego, CA, June 1990, pp. III 137–142.

[4] 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)*. Cambridge, MA: M.I.T. Press, 1986, chap. 8.

[5] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavior science," Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

[6] A. Linden and J. Kindermann, "Inversion of multilayer nets," in *Proc. Int. Joint Conf. Neural Networks*, Washington DC, June 1989, pp. 425–430.

[7] L. G. Valiant, "A theory of the learnable," *Commun.* ACM, vol. 27, pp. 1134–1142, Nov. 1984.

[8] J. N. Hwang, J. J. Choi, S. Oh, and R. J. Marks II, "Classification boundaries and gradients of trained multilayer perceptrons," in *Proc. Int. Symp. Circuits Syst.* New Orleans, LA, May 1990, pp. 3256–3259.

[9] —, "Query learning based on boundary search and gradient computation of trained multilayer perceptrons," in *Proc. Int. Joint Conf. Neural Networks*, San Diego, CA, June 1990, pp. III 57–62.

[10] L. H. Fink, "Power system security assessment," in *Proc. IEEE Conf. Decision Control*, pp. 478–480, Dec. 1984.

[11] A. S. Debs, *Modern Power System Control and Operations*. Boston, MA: Kluwer, 1988.

[12] R. J. Marks II, M. J. Damborg, M. A. El-Sharkawi, L. E. Atlas, M. Aggoune, and D. A. Cohn, "Artificial neural networks for power system static security assessment," in *Proc. Int. Symp. Circuits Syst.* Portland, OR, May 1989, pp. 490–494.

[13] H. W. Dommel and T. F. Tinney, "Optimal power flow solutions," *IEEE Trans. PAS*, vol. 87, pp. 1866–1876, Oct. 1968.