Arbeitspapiere der GMD 735

Byoung-Tak Zhang

Learning by Incremental Selection of Critical Examples

March 1993

GESELLSCHAFT FÜR MATHEMATIK
UND DATENVERARBEITUNG MBH

Abstract

The problem of selecting critical examples for training neural networks is discussed. Based on Bayesian statistics and information theory we derive a measure of criticality and present a computationally efficient method for selecting critical examples. This leads to a new type of learning scheme in which the network is trained incrementally on an increasing number of self-selected examples, instead of on all the available data at the outset. Experimental results show that the selective incremental learning method finds a critical subset of the given examples and thereby achieves considerable improvement in training speed and generalization performance if a large amount of data is known in advance.

1 Introduction

There are a number of reasons why the training data should be controlled in neural networks. One is that data collection is usually combined with costly experimentation or time-consuming interaction with the environment. Another is that the training time generally increases as the number of examples increases (Abu-Mostafa 1989). A third reason for controlling data is that there is no guarantee that the generalization performance is improved by increasing only the cardinality of the training data set. Often the opposite is observed (Press *et al.*, 1986).

In general, one should choose those examples which are most likely to help the network solve the problem. Volper and Hampson (1987) show that the perceptron learning can be significantly accelerated by utilizing specific instances. For classification problems, these examples are the border patterns, i.e. the patterns that lie closest to the separating hyperplane. Ahmad and Tesauro (1989) study the generalization accuracy as a function of training set and suggest a heuristic for selecting border patterns for the majority function. The analysis of Huyser and Horowitz (1989) shows that training patterns must be drawn from the critical set containing all the border patterns of the function if perfect generalization is to be guaranteed. For real applications, however, it is practically impossible to estimate in advance the exact utility of examples, since the ultimate criticality of examples depends not only on the task, but also on the network configuration which has a highly nonlinear structure.

Recently, information-based objective functions for data selection have been addressed by MacKay (1992). He approaches the problem from the Bayesian framework, a branch where the problem of active data sampling has a long history. The statistical interpretation of network learning was also studied by Tishby *et al.* (1989). Their analysis enables the evaluation of the probability of a correct prediction of an independent example, after training the network on a given training set. However, they did not suggest methods of how to select the critical examples efficiently.

In previous work we have proposed an incremental learning method for selecting and generating critical examples using the search principles of genetic algorithms (Zhang and Veenker 1991a; Zhang and Veenker 1991b). The work described below is motivated by the Bayesian framework on active data selection and extends our earlier work in theory and experiments.

The paper is organized as follows. In Section 2 we derive a quantitative measure of criticality of examples. In Section 3 the learning algorithm is described that finds a critical subset of the given examples for an arbitrary task while the training progresses. This is followed by empirical results and conclusions in Sections 4 and 5, respectively.

2 Selection Criterion

We want to build a model of an input-output relation, γ, by a supervised learning neural network. Let D be the example set for the relation $\gamma : X \rightarrow Y$, i.e.

$$D_N = \{(\mathrm{x}_p, \mathrm{y}_p) \mid \mathrm{x}_p \in X,\ \mathrm{y}_p \in Y,\ \mathrm{y}_p = \gamma(\mathrm{x}_p), p = 1, ..., N\} \tag{1}$$

where N is the number of examples. We train the weights W of the network using the data D_N by minimizing the additive error function

$$E(D_N|W) = \sum_{p=1}^{N} E(\mathrm{y}_p|\mathrm{x}_p, W). \tag{2}$$

The error function commonly used for an example is the sum of squared errors between the target and the computed output. For a two-layer feedforward network it has the form:

$$E(\mathrm{y}_p|\mathrm{x}_p, W) = \sum_j \left(y_{pj} - f_j \left(\sum_h w_{jh} f_h \left(\sum_i w_{hi} x_{pi} \right) \right) \right)^2 \tag{3}$$

where the indices i, j, and h run over the input, output, and hidden units, respectively. f_k denotes the activation function of the k-th unit and w_{ki} the weight from unit i to k.

Given the network configuration W, we can assign the likelihood that training examples are related through the network as (Tishby *et al.* 1989):

$$P(D_N|W) = \prod_{p=1}^{N} P(\mathrm{y}_p|\mathrm{x}_p, W) \tag{4}$$

where

$$P(\mathrm{y}_p|\mathrm{x}_p, W) = \frac{\exp(-\beta E(\mathrm{y}_p|\mathrm{x}_p, W))}{Z(\beta)}. \tag{5}$$

Here β is a positive constant which determines the sensitivity of the probability to the error value and $Z(\beta) = \int \exp(-\beta E(\mathbf{y}|\mathbf{x}, W))dy$ is a normalizing constant.

Now we want to improve the performance of the network by training on an $(N+1)$-th data point. This example should be selected so that the information gain of the network is maximized when we add the new example $(\mathbf{x}_{N+1}, \mathbf{y}_{N+1})$ to the existing training set. Let the probability distributions of the parameters before and after the example $\mathbf{y}_{N+1}$ is learned be $P_N(W)$ and $P_{N+1}(W)$. According to information theory (Kullback 1959), the mean information for discrimination between $P_N(W)$ and $P_{N+1}(W)$ is given by

$$I(P_{N+1}, P_N) = \int P_{N+1}(W) \ln \frac{P_{N+1}(W)}{P_N(W)} dW. \qquad (6)$$

The greater the value of $I(P_{N+1}, P_N)$, the less resemblance there is between the two distributions and the more information we have gained about W. Given a fixed distribution $P_N(W)$, the maximum information gain is thus achieved by maximizing the difference of $P_{N+1}(W)$ from $P_N(W)$. This is done, because of the relation (5) between the likelihood and the error of the data, by selecting the example whose addition to D_N leads to the greatest $E(D_{N+1}|W)$ with the current parameters W, i.e. the example that maximizes

$$\Delta E_{N+1} = E(D_{N+1}|W) - E(D_N|W), \qquad (7)$$

and training the network on this example to reduce the error.

This example can be found by presenting the candidates $\mathbf{x}_c$ to the partially trained network and computing their errors $E(\mathbf{y}_c|\mathbf{x}_c, W)$ and selecting the k-th example satisfying

$$E(\mathbf{y}_k|\mathbf{x}_k, W) = \max_c \{ E(\mathbf{y}_c|\mathbf{x}_c, W) \}. \qquad (8)$$

In situations where we have only to select one from a large data set, we need only the relative error of each example, not the value itself. Formally we define the *criticality* of an example ($\mathbf{x}_c$ with respect to the network W as:

$$e_W(c) = \frac{E(\mathbf{y}_c|\mathbf{x}_c, W)}{\dim(\mathbf{y}_c)} = \frac{1}{m}\sum_{j=1}^{m} (y_{cj} - f_j(\mathbf{x}_c; W))^2, \qquad (9)$$

where f_j denotes the actual output of unit j and m, the number of the output units, is used to normalize the value. The measure $e_W(c)$ has a value $0 \le e_W(c) \le 1$ if the sigmoid output function $f_j(a) = \frac{1}{1+e^{-a}}$ is used.

Observe that the criticality of an example is defined with respect to the current model W and the accuracy of the model is dependent on the criticality of the training examples used so far. At first glance, this recursive relationship between the training set and the network seems harmful, but it gives us a natural procedure for finding a critical subset of given data, as will be described in the next section.

3 Selective Incremental Learning

Learning proceeds incrementally; that is, the network is trained on an small example set which is increased after training by additional examples. Formally we consider two disjoint subsets of the given *learning set*: a training set and a candidate set. The *training set*, denoted by $\Psi = \{\psi_p\}$, is defined to be the set of data that are actually used to train the network. The rest of the given examples will be referred to as the *candidate set* and denoted by $\Phi = \{\varphi_q\}$. The training set is initialized with a small set of randomly chosen seed examples. The objective of learning is to find a training set that is as small as possible yet is large enough to achieve perfect generalization to the candidate set. The learning process consists of iteration of the training of the network and the expansion of the training set.

In the training phase, the weights of the network are updated using only the examples in the training set Ψ. For the purpose of this discussion, we consider feedforward networks of sigmoid activation functions. The principle described, however, should be equally applicable to other types of networks and learning rules. The weights of the network are initialized randomly with values from the interval $-\omega \leq w_{ji} \leq +\omega$. Each time an example $(\mathbf{x}_p, \mathbf{y}_p)$ is presented to the network, the weights are modified by

$$\Delta w_{ji}(t) = -\epsilon(t)\frac{\partial E_p}{\partial w_{ji}} + \eta(t)\Delta w_{ji}(t-1) \tag{10}$$

with $\epsilon(t)$ and $\eta(t)$ denoting the stepsize and the momentum factor at time t, respectively. The error gradient $\frac{\partial E_p}{\partial w_{ji}}$ of the p-th example is approximated by the error backpropagation procedure (Rumelhart *et al.*, 1986).

The weight adjustment is repeated until the total sum of errors of the current training set is reduced to a specified performance level, i.e.

$$E(\Psi|W) \leq \frac{h(n+m)}{\tau}. \tag{11}$$

Here n, m and h are the number of input, output and hidden units, respectively. The parameter τ is a constant, typically $100 \le \tau \le 200$, and determines the allowable error tolerance per connection. This is based on the observation that the learning capacity is closely related with the total number of adjustable connections in the network (Denker *et al.* 1987).

In the selection phase, the examples in the current candidate set are tested for their criticality (9). Note that $e_W(c)$ can be calculated in a simple way by presenting the c-th candidate example to the current network and observing its error, followed by normalization of the value. Having computed the criticality, a number π of examples φ_c are selected from the candidate set Φ and moved to the training set Ψ:

$$\Psi \leftarrow \Psi \cup \{\varphi_c\}, \qquad \Phi \leftarrow \Phi - \{\varphi_c\}. \tag{12}$$

Note that the training set increases by π, while the candidate set decreases by the same amount.

Using the expanded training set Ψ, the next cycle of training and selection is performed. The algorithm terminates when the specified performance level is achieved or the candidate set Φ is empty.

4 Experimental Results

The above selective incremental learning method (SEL) has been applied to several discrete problems (e.g. majority, contiguity, and parity) and continuous functions such as the forward and inverse kinematics of a robot arm (Zhang 1992). In most applications a moderate amount of training examples was sufficient to achieve a reasonable performance. In this section we briefly describe two of them: a benchmark problem and a more realistic application.

4.1 Four-Quadrant Function

The capability of the algorithm to select critical examples is demonstrated on the following function:

$$y(x_1, x_2) = \begin{cases} 1 & \text{if } (x_1 < 0.5 \text{ and } x_2 > 0.5) \text{ or } (x_1 > 0.5 \text{ and } x_2 < 0.5) \\ 0 & \text{otherwise,} \end{cases}$$

where $0.1 \leq x_1, x_2 \leq 0.9$. This function, called four-quadrant, is a real-input variant of the XOR problem. Due to its graphical characteristics the problem allows an easy analysis of the learning results. The graph of the function is depicted in Figure 1. Also shown is the total of 400 (20×20) examples which was provided the algorithm initially.

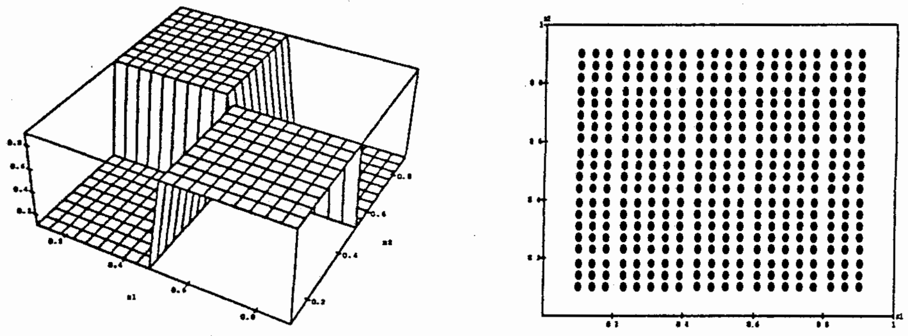
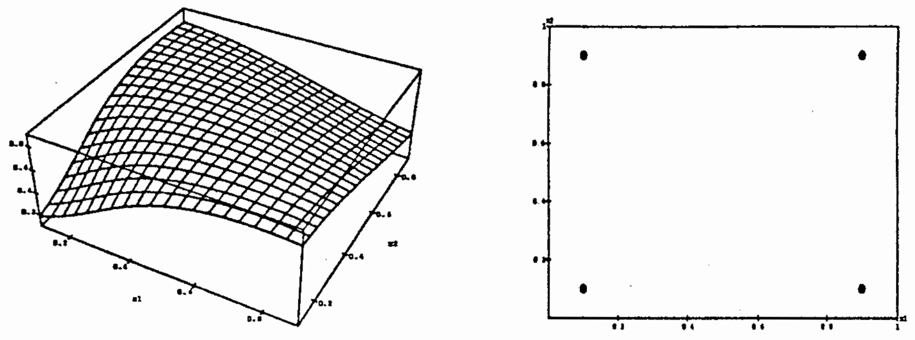


Figure 1: The four-quadrant function and the given data points

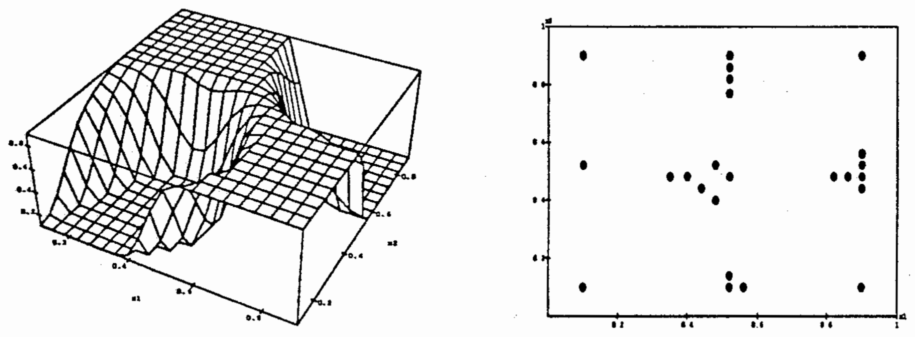
In this experiment a feed-forward network with four hidden units was used. The training set was initialized with four seed examples shown in Figure 2. In each selection step, additional four examples ($\pi = 4$) were added to the existing training set. The examples were selected by the criticality measure (9). The figure also shows the graphes of the approximated function and the corresponding training points used up to the corresponding training time. Approximation results suggest that the selected examples are very effective for learning the function. Moreover, the preferred examples are those that lie on the separating lines of output 0 and 1. These are the border patterns and known as critical to solving this kind of problem (Ahmad and Tesauro 1989; Huyser and Horowitz 1989).

4.2 Handwritten Digit Recognition

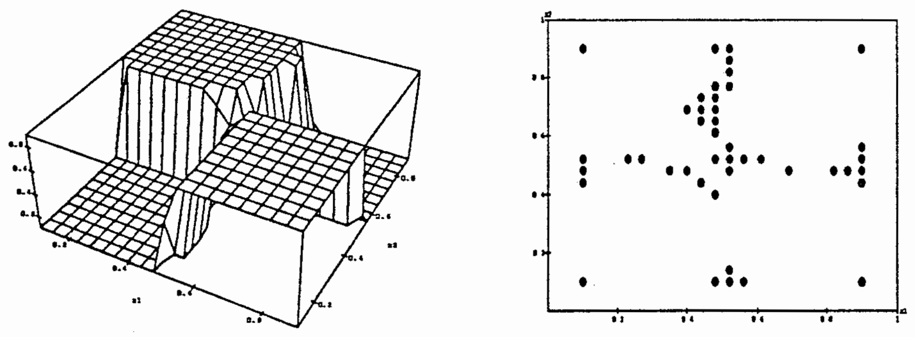
The performance of the algorithm was evaluated on the recognition of handwritten digits. The data was collected from 10 persons. Each person wrote by hand 680 digits on two sheets of paper. The digits were read by a scanner to be converted in 15×10 bitmaps. One sheet of each writer was chosen randomly to build the learning set (3400 digits). The other sheets of them were collected



$s = 0 \quad (N = 4)$



$s = 5 \quad (N = 24)$



$s = 10 \quad (N = 44)$

Figure 2: Selective incremental learning of the four-quadrant function s denotes the selection step. The learning started with four seed examples and in each selection step four new examples are added to the existing training set. The graphs show the intermediate results of approximation at $s = 0, 5$ and 10 and the training points selected.

to build the test set (3400 digits). Some examples for the bitmap patterns are shown in Figure 3.

We used a feed-forward network with a 150-40-10 architecture. The weights were initialized with random values from the interval $[-0.1, +0.1]$. 10 digits of 0 to 9 were randomly chosen from the learning set to initialize the training set. In each selection phase, 200 new examples were chosen to expand the training set. In each adaptation phase the network was trained until the total sum of errors for the training set dropped below $\lambda = \frac{1}{150}h(n+m) = \frac{1}{150}40(150+10) = 42.7$ where n, m and h are the number of the input, the output and the hidden units, respectively. This precision corresponds to an average error per example of $E_p = 0.01$.

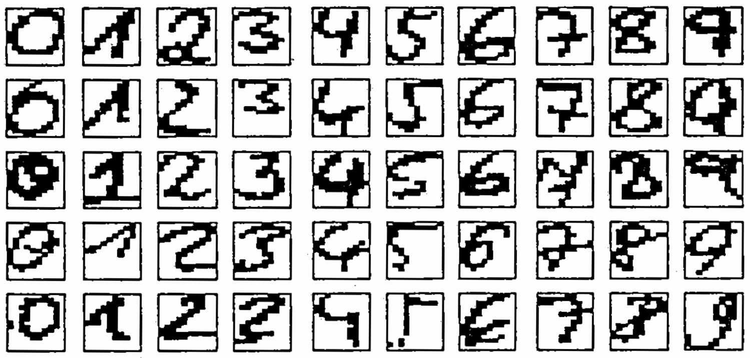
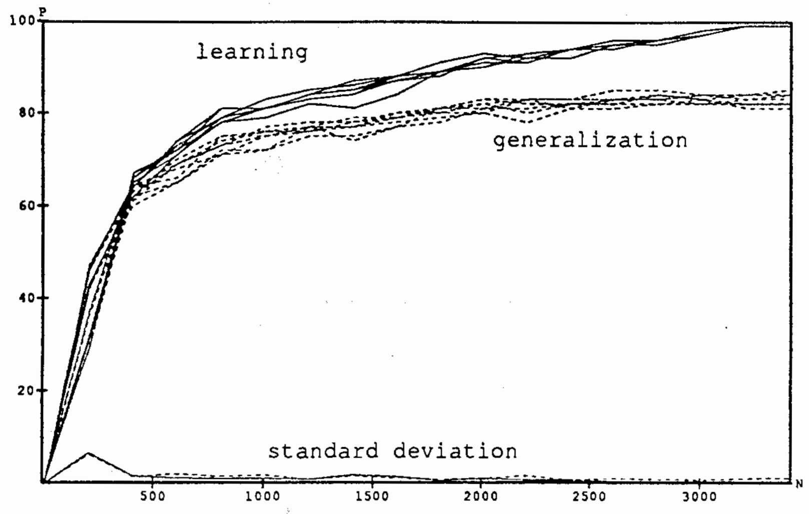


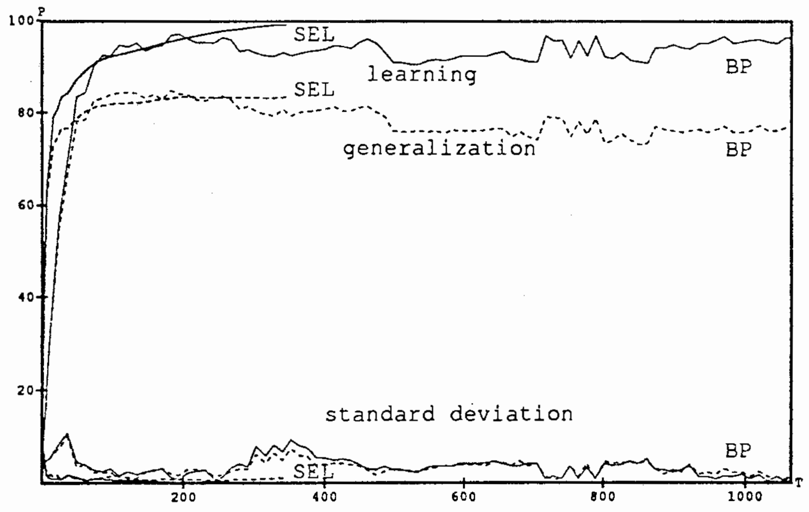
Figure 3: Some of the test digits

The performance of the SEL method for ten runs is shown in Figure 4 (a) as a function of the training set size N . The learning performance was calculated within the learning set (the training set plus the candidate set) while the generalization performance was measured on the test set. One can see a rapid improvement in learning and generalization performance at the early stages until the training set size of $N = 1000$ and a slow but steady increase afterwards. This indicates that the algorithm has found at this point a moderately critical subset of the given data. The standard deviation suggests that the network performance is very robust against the initial weight values and the seed examples.

For comparison, we trained the same size of network using an online back-propagation procedure (BP). All the learning parameters were the same as those used in the adaptation phase of SEL. The SEL method converged to a 99% performance, on average, four times faster than the simple BP training. For both algorithms the maximum generalization accuracy was about 85%



(a)



(b)

Figure 4: Results on digit recognition. (a) The learning and generalization performance of SEL as a function of the training set size, for 10 runs. (b) Performance of SEL compared with BP as a function of the cpu time, averaged over 10 runs each.

for the data used. To illustrate the difference, the average performances of both learning procedures for ten runs each are presented in Figure 4 (b) as a function of the training time measured in cpu minutes on a Sparcstation. It can be seen that although BP improves its performance relatively fast at the early stages and converges at the end, the convergence was very unstable and therefore very slow. The simple BP method seems to concentrate too much on the statistical average of all the available patterns, which occasionally shows good performance but generally gets stuck in a local optimum. On the other hand, SEL improves its learning and generalization performance monotonically and converges faster than BP, achieving its maximum generalization accuracy at convergence time. BP shows overfitting phenomena which make it difficult to predict the generalization performance. The monotonicity of generalization in SEL can be very useful since it helps decide whether to gather more data or not to improve the performance.

5 Concluding Remarks

We have proposed a criterion for measuring the criticality of examples for a given network and presented a method which selects a critical subset of given examples during training the network. This approach has led to a new form of supervised learning in which the learning process starts with a small training set which incrementally grows by selected examples after training. The underlying idea behind this selective incremental learning is that the performance of the network can best be improved by being trained on a good training set and the quality of the training set can best be enhanced by taking account of the current state of the network. In effect, the incremental learning performs a kind of adaptive global-to-local search not only on the weight space but on the example space as well, as opposed to the simple backpropagation training procedure that pushes the network to load all the data at the same time, leading usually to a delayed convergence.

The experimental results on digit recognition have shown that the method finds a critical subset of given examples, and that it converges significantly faster and generalizes more robustly than a simple backpropagation procedure. We expect the incremental learning will in general be more efficient than a non-incremental one if the given data contains enough information for correct generalization. However, we have found in another set of experiments that the incremental learning can be more expensive than a simple backpropagation if the available data is too sparse or already critical.

The work can be extended in two main ways. In one way, the criticality measure may be used for further purposes. Although it was originally constructed for selecting examples, the measure can also be used to generate new useful examples. The ability of creating new examples enables for a learning system to adapt itself to an unknown environment. Preliminary results on this point were reported in (Zhang and Veenker 1991b), where new examples are created by recombining existing critical examples through genetic operators. Hwang *et al.* (1991) and Baum (1991) have also addressed the generation of examples. In the other way, one can combine the idea of incremental data selection with architecture optimization algorithms, especially with the constructive algorithms such as Ash (1989) and Fahlman (1990). In a series of experiments reported in (Zhang 1992) we could optimize the number of hidden units of a feed-forward network using only a critical subset of examples and thereby could improve the optimization efficiency of constructive algorithms, instead of providing them with all known data set at the outset.

Acknowledgements

We thank Professor Gerd Veenker and Knut Möller of the University of Bonn, Heinz Mühlenbein, Jörg Kindermann, Frank Śmieja, Alexander Linden and the other members of the learning systems research group of GMD for stimulating discussions and comments.

References

[1] Abu-Mostafa, Y.S. 1989. The Vapnik-Chervonenkis Dimension: Information versus Complexity in Learning, *Neural Computation*, 1(3):312–317, MIT Press.

[2] Ahmad, S., and Tesauro, G. 1989. Scaling and generalization in neural networks: a case study. In *Proc. 1988 Connectionist Models Summer School*, pp. 3-10, Morgan Kaufmann.

[3] Ash, T. 1989. Dynamic node creation in backpropagation networks, *Connection Science*, 1(4):365–375.

[4] Baum, E.B. 1991. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Trans. on Neural Networks*, 2(1):5-19.

[5] Denker, J. *et al.* 1987. Large automatic learning, rule extraction, and generalization. *Complex systems*, 1:877-922.

[6] Fahlman, S.E. 1990. The cascade-correlation learning architecture. In *Advances in Neural Information Processing 2*, D. S. Touretzky, ed., pp. 524-532, Morgan Kaufmann.

[7] Huyser, K.A., and Horowitz, M.A. 1989. Generalization in connectionist networks that realize Boolean functions. In *Proc. 1988 Connectionist Models Summer School*, pp. 191-200, Morgan Kaufmann.

[8] Hwang, J.N. *et al.* 1991. Query-based learning applied to partially trained multilayer perceptrons. *IEEE Trans. on Neural Networks*, 2(1):131-136.

[9] Kullback, S. 1959. *Information Theory and Statistics*, Wiley, New York.

[10] MacKay, D.J.C. 1992. *Bayesian methods for adaptive models.* Ph.D. thesis, Caltech, Pasadena, CA.

[11] Press, W.H., Flannery, B., Teukolsky, S.A. and Vetterling, W.T. 1986. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press.

[12] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland, eds., Vol. I, pp. 318-362, MIT Press.

[13] Tishby, N., Levin, E., and Solla, S.A. 1989. Consistent inference of probabilities in layered networks: predictions and generaliation. *Proc. IJCNN-89*, Vol. II, pp. 403–409, IEEE.

[14] Volper, D.J., and Hampson, S.E. 1987. Learning and using specific instances, *Biological Cybernetics*, 57:57-71.

[15] Zhang, B.T. 1992. *Learning by Genetic-Neural Evolution: Active Adaptation to Unknown Environments with Self-developing Parallel Networks* (in German), DISKI-16, Infix-Verlag. also available as Informatik Berichte Nr. 93, Inst. for Comp. Sci., Univ. of Bonn.

[16] Zhang, B.T., and Veenker, G. 1991a. Focused incremental learning for improved generalization with reduced training sets. In *Artificial Neural Networks: Proc. ICANN-91*, T. Kohonen *et al.* eds., Vol. I, pp. 227-232, North-Holland.

[17] Zhang, B.T., and Veenker, G. 1991b. Neural networks that teach themselves through genetic discovery of novel examples. In *Proc. IJCNN-91*, Vol. I, pp. 690-695, IEEE.