Generalization Performance of Bayes Optimal Classification Algorithm for Learning a Perceptron

Manfred Opper
Institut für Theoretische Physik, Justus-Liebig-Universität Giessen, D-6300 Giessen, Germany

David Haussler
Department of Computer and Information Sciences, University of California at Santa Cruz, Santa Cruz, California 95064

(Received 7 December 1990)

The generalization error of the Bayes optimal classification algorithm when learning a perceptron from noise-free random training examples is calculated exactly using methods of statistical mechanics. It is shown that if an assumption of replica symmetry is made, then, in the thermodynamic limit, the error of the Bayes optimal algorithm is less than the error of a canonical stochastic learning algorithm, by a factor approaching $\sqrt{2}$ as the ratio of the number of training examples to perceptron weights grows. In addition, it is shown that approximations to the generalization error of the Bayes optimal algorithm can be achieved by learning algorithms that use a two-layer neural net to learn a perceptron.

PACS numbers: 87.10.+e, 02.50.+s, 05.20.−y

Extending a line of research initiated by Gardner,[1,2] exceptional progress has been made in recent years in applying the methods of statistical mechanics to the analysis of the process of learning from random examples, as exemplified in the learning algorithms used to train neural networks. Recent work[3-10] has focused on quantifying the generalization performance of learning algorithms. This is the probability that the learning algorithm will predict correctly the classification of a new random input, after it has seen a certain number of random classified inputs, called *training examples.*

Most neural-net learning algorithms make predictions on novel inputs by selecting a *hypothesis,* represented by couplings or "synaptic" *weights* of a neural network, that performs well on the training examples. A canonical algorithm of this type, which we call the *Boltzmann algorithm,* was studied in Refs. 7–9. Here we apply similar methods from statistical physics to study a different type of learning algorithm, namely, the *Bayes optimal classification algorithm.*

The performance of any learning algorithm will depend on the *target function,* i.e., the input-output mapping to be learned. In the Bayesian approach, variability in the selection of the target function is modeled by assuming an *a priori probability distribution* over possible target functions. One then seeks a learning algorithm that will give the best *average* generalization performance on target functions selected according to this *a priori* distribution. This is what the Bayes optimal classification algorithm does. The performance of the Bayes algorithm provides the natural standard against which other algorithms may be compared.

In this Letter we derive expressions for the average generalization performance for both the Bayes algorithm and the Boltzmann algorithm for the simplest neural network: the single-layer perceptron. Let N denote the dimension of the input space, m the number of training examples, and $\alpha = m/N$. We show that as $m,N \to \infty$ such that α remains constant, the probability of misclassification after m random training examples for the Boltzmann algorithm is a factor of $\sqrt{2}$ from the optimal performance of the Bayes algorithm, asymptotically as α becomes large.

It turns out that the Bayes algorithm is difficult to implement on a neural network, so we also look at a series of learning algorithms that approximate the performance of the Bayes algorithm. These algorithms use a neural network with a layer of n hidden units between the input and output, for $n = 1,3,5,\ldots$, where the output node takes a majority of the hidden units. For $n=1$, we get the Boltzmann algorithm, and as $n \to \infty$, performance of these algorithms approaches the Bayes optimal.

In the simple binary-classification learning problem we consider, one tries to learn a *target function* f^* that maps from a set X (*input space*) into $\{-1,+1\}$. Here we take X to be the N-dimensional space of real vectors and consider each component $x(i)$ of $\vec{x} \in X$ to be the state of an input node for a neural network on input $\vec{x}$. Each possible setting of the vector of weights $\vec{w}$ in the neural network defines a classification function $f_{\vec{w}}$ from X to $\{-1,+1\}$. For the single-layer perceptron

$$f_{\vec{w}}(\vec{x}) = \mathrm{sgn}(\vec{w}\cdot\vec{x}/\sqrt{N}),$$

where we restrict the weight vector to the sphere $\vec{w}\cdot\vec{w} = N$. Learning a perceptron was investigated by various authors.[6,7,10,11] We look at a simple model of learning where we assume that the function f^* can be learned *perfectly* by the neural network, i.e., $f^* = f_{\vec{w}^*}$ for some weight vector $\vec{w}^*$. We will call $\vec{w}^*$ the *target vector.*

In the process of learning we assume that a target vector $\vec{w}^*$ is selected at random and a sequence of inputs $\mathbf{x}^\infty = (\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots)$ is selected at random. Noise-free classification labels $\sigma^\infty = (\sigma_1, \sigma_2, \sigma_3, \ldots)$ are generated by $\sigma_k = f_{\vec{w}^*}(\vec{x}_k)$, $k = 1,2,\ldots$. Given only training examples $(\vec{x}_1,\sigma_1),(\vec{x}_2,\sigma_2),\ldots,(\vec{x}_m,\sigma_m)$ and input $\vec{x}_{m+1}$, the learning algorithm must predict the label σ_{m+1}. The generalization error $\epsilon(m)$ of the learning algorithm is the probability that it predicts wrong, as a function of the training size m.

For now, let us assume that the input sequence $\mathbf{x}^\infty$ is fixed and the only randomization is over the choice of the target vector $\vec{w}^*$, which is chosen according to an *a priori* density $d\mu(\vec{w}^*)$. Let $\mathcal{T}_m(\vec{w}^*) = \{\vec{w}: f_{\vec{w}}(\vec{x}_k) = f_{\vec{w}^*}(\vec{x}_k)$, for $k=1,\ldots,m\}$, i.e., the set of all weight vectors that are *consistent* with the first m training examples. This is called the *version space* in the artificial intelligence literature.[12] The volume of the version space is its measure under $d\mu$, i.e.,

$$V_m(\vec{w}^*) = \int_{\vec{w}\in\mathcal{T}_m(\vec{w}^*)} d\mu(\vec{w})\,.$$

In order to make its prediction on the input $\vec{x}_{m+1}$, the *Boltzmann algorithm* uses a hypothesis $\vec{w}$ chosen at random according to the posterior density on the weight space, given the first m training examples. This density is zero everywhere except on the version space $\mathcal{T}_m(\vec{w}^*)$, where it is equal to the prior density $d\mu(\vec{w})$ divided by the normalizing constant $V_m(\vec{w}^*)$. This procedure is the zero-temperature limit of the stochastic learning algorithm discussed in Refs. 7 and 9. The algorithm makes the correct prediction if $\vec{w}$ is in the set $\mathcal{T}_{m+1}(\vec{w}^*)$. Thus for fixed target $\vec{w}^*$, the probability of making the correct prediction is given by the ratio $V_{m+1}(\vec{w}^*)/V_m(\vec{w}^*)$. A similar formulation has been obtained in Ref. 13. The average generalization error of this algorithm, when $\vec{w}^*$ is chosen at random by $d\mu(\vec{w}^*)$, but the first $m+1$ inputs $\mathbf{x}^{m+1} = (\vec{x}_1,\ldots,\vec{x}_{m+1})$ are fixed, is thus given by

$$\epsilon_{\text{Boltz}}(\mathbf{x}^{m+1}) = \langle 1 - V_{m+1}(\vec{w}^*)/V_m(\vec{w}^*)\rangle_{\vec{w}^*}\,, \qquad (1)$$

where $\langle\,\rangle_{\vec{w}^*}$ denotes integration over $d\mu(\vec{w}^*)$.

The present formulation treats integrations over $\vec{w}$ and averages over $\vec{w}^*$ in a nonsymmetric way. To facilitate the subsequent calculations we can remove this asymmetry by replacing the average over targets by an equivalent average over all possible labelings $\boldsymbol{\sigma}^m = (\sigma_1, \sigma_2,\ldots,\sigma_m) \in \{\pm 1\}^m$. Let $\mathcal{T}(\boldsymbol{\sigma}^m) = \mathcal{T}_m(\vec{w}^*)$ and $V(\boldsymbol{\sigma}^m) = V_m(\vec{w}^*)$, when $\sigma_k = f_{\vec{w}^*}(\vec{x}_k)$ for all $k=1,\ldots,m$. We define $\mathcal{T}(\boldsymbol{\sigma}^m) = \varnothing$ and $V(\boldsymbol{\sigma}^m) = 0$ if the labels in $\boldsymbol{\sigma}^m$ cannot be generated by any weight vector. Then, since $V(\boldsymbol{\sigma}^{m+1})$ is the prior probability of the label sequence $\boldsymbol{\sigma}^{m+1}$,

$$\epsilon_{\text{Boltz}}(\mathbf{x}^{m+1}) = \sum_{\boldsymbol{\sigma}^{m+1}\in\{\pm 1\}^{m+1}} V(\boldsymbol{\sigma}^{m+1})[1 - V(\boldsymbol{\sigma}^{m+1})/V(\boldsymbol{\sigma}^m)]\,. \qquad (2)$$

If the goal is to maximize the probability of a correct prediction, it turns out that the Boltzmann algorithm is not the best algorithm to use. The best possible prediction for σ_{m+1} would be obtained by calculating the total posterior probability, given the first m training examples, of all $\vec{w}$'s that would predict $+1$ on $\vec{x}_{m+1}$ and compare this with the corresponding posterior probability for -1. One then chooses the output with largest posterior probability, which clearly maximizes the probability of a correct prediction. This is known as the *Bayes optimal classification algorithm* or *Bayes algorithm.* It is also called the weighted majority algorithm.[14,15]

Using the Bayes algorithm, an error occurs only if, for the actual output σ_{m+1}, the corresponding volume $V(\boldsymbol{\sigma}^{m+1})$ turns out to be the smaller of the two possible subvolumes of the mth version space $\mathcal{T}(\boldsymbol{\sigma}^m)$, i.e., only if $V(\boldsymbol{\sigma}^{m+1}) \le \frac{1}{2} V(\boldsymbol{\sigma}^m)$. Thus the average generalization error, when the target vector is drawn at random but $\mathbf{x}^{m+1}$ is fixed, is given by

$$\epsilon_{\text{Bayes}}(\mathbf{x}^{m+1}) = \langle\Theta(1 - 2V_{m+1}(\vec{w}^*)/V_m(\vec{w}^*))\rangle_{\vec{w}^*} = \sum_{\boldsymbol{\sigma}^{m+1}\in\{\pm 1\}^{m+1}} V(\boldsymbol{\sigma}^{m+1})\Theta(1 - 2V(\boldsymbol{\sigma}^{m+1})/V(\boldsymbol{\sigma}^m))\,, \qquad (3)$$

where $\Theta(x)$ is the unit step function, i.e., $\Theta(x) = 1$ if $x \ge 0$, $\Theta(x) = 0$ if $x < 0$.

Note that when learning a perceptron, the hypothesis used by the Bayes algorithm cannot itself be represented by a perceptron. However, we can construct a network with one hidden layer for which the average error converges to ϵ_{Bayes} as n, the number of units in the hidden layer, goes to infinity. To do this we train an odd number n of perceptrons *independently* using the Boltzmann algorithm and make them hidden units. Each hidden-unit output $\sigma(i)$, $i=1,\ldots,n$, is passed to a fixed output perceptron that computes the majority function $\sigma_{\text{maj}} = \text{sgn}[\sum_{i=1}^n \sigma(i)]$ as the final output. For a fixed target vector $\vec{w}^*$, the probability that k perceptrons make the right prediction on $\vec{x}_{m+1}$ and $n-k$ make the wrong prediction is then given by the binomial distribution $a_k = \binom{n}{k} Y_m^k (1-Y_m)^{n-k}$, with $Y_m = Y_m(\vec{w}^*) = V_{m+1}(\vec{w}^*)/V_m(\vec{w}^*)$. Thus the error of the majority vote becomes $\sum_{k=0}^{[n/2]} a_k$. As $n \to \infty$ it is easy to see that this expression converges to $\Theta(1 - 2Y_m(\vec{w}^*))$ for all $Y_m \ne \frac{1}{2}$. Taking the expectation with respect to $\vec{w}^*$ we recover the generalization error of the Bayes algorithm from (3). By varying the number n of hidden units, this defines a sequence learning of algorithms with generalization errors $\epsilon_n(\mathbf{x}^{m+1})$, $n = 1,3,5,\ldots$, with $\epsilon_1(\mathbf{x}^{m+1}) = \epsilon_{\text{Boltz}}(\mathbf{x}^{m+1})$ and $\epsilon_\infty(\mathbf{x}^{m+1}) = \epsilon_{\text{Bayes}}(\mathbf{x}^{m+1})$.

In the following we now assume that the instances $\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots$ are drawn independently at random from an N-dimensional probability distribution. In particular, we assume that for each i and j, where $1 \le i,j \le N$, the Cartesian components $x_k(i)$ and $x_k(j)$ of $\vec{x}_k$, $k=1,2,\ldots$, are independent identically distributed random variables with zero mean and covariance C_{ij}. For each n, the average generalization error for the n-hidden-unit learning algorithm is $\epsilon_n(m) = \langle\epsilon_n(\mathbf{x}^{m+1})\rangle_{\mathbf{x}^{m+1}}$, where $\langle\,\rangle_{\mathbf{x}^{m+1}}$ denotes the average over the random selection of $\vec{x}_1,\ldots,\vec{x}_{m+1}$.

This average can be calculated from the positive integer moments of the random variable Y_m. These are

$$y(m,r) = \langle\langle Y_m^r(\mathbf{x}^{m+1},\vec{w}^*)\rangle_{\vec{w}^*}\rangle_{\mathbf{x}^{m+1}} = \left\langle \sum_{\boldsymbol{\sigma}^{m+1}\in\{\pm 1\}^{m+1}} \frac{V^{r+1}(\mathbf{x}^{m+1},\boldsymbol{\sigma}^{m+1})}{V^r(\mathbf{x}^m,\boldsymbol{\sigma}^m)} \right\rangle_{\mathbf{x}^{m+1}} \qquad (4)$$

for $r = 0,1,2,\ldots$. (Here we have also added the explicit dependence of Y_m and V on the inputs $\vec{x}_1,\ldots,\vec{x}_{m+1}$ to our notation.)

For the perceptron,

$$V^{r+1}(\mathbf{x}^{m+1},\sigma^{m+1}) = \int \prod_{a=1}^{r+1} d\mu(\vec{w}_a) \prod_{k=1}^{m+1}\prod_{a=1}^{r+1} \Theta((\vec{w}_a\cdot\vec{x}_k)\sigma_k/\sqrt{N}),$$

which can be interpreted as the probability that $r+1$ independent random *replicas* $\vec{w}_a$, $a=1,\ldots,r+1$, of the network weights represent functions that all give the same outputs $\sigma_1,\ldots,\sigma_{m+1}$ on inputs $\vec{x}_1,\ldots,\vec{x}_{m+1}$.

To enable the calculation of $y(m,r)$ for large N we assume that the distribution of instances and the density $d\mu$ are such that for $N\to\infty$, and almost all $\vec{w}_a$, $a=1,\ldots,r+1$, the weighted inputs to the ath perceptron $u_a=\vec{w}_a\cdot\vec{x}/\sqrt{N}$ obey the *central limit theorem,* i.e., they become Gaussian distributed with zero mean and finite covariance $Q_{ab}=N^{-1}(\vec{w}_a C\vec{w}_b)$, for $1\le a,b\le r+1$. When the weight vectors $\vec{w}_a$ are drawn randomly by $d\mu(\vec{w})$, $Q=\{Q_{ab}\}$ is an $r+1$ by $r+1$ matrix-valued random variable.

Under the Gaussian assumption we can carry out the average over the $(m+1)$st input in (4) and its two possible outputs $\sigma_{m+1}=\pm 1$ explicitly. The remaining sum over the first m output can again be reinterpreted as an average over the targets. The result can be written as

$$y(m,r)=\left\langle 2(2\pi)^{-(r+1)/2}(\det Q)^{-1/2}\int_0^{\infty}\prod_{a=1}^{r+1} du_a \exp\left[-\tfrac{1}{2}\sum_{a,b} u_a (Q^{-1})_{ab} u_b\right]\right\rangle_Q, \tag{5}$$

where we define the auxiliary Q average $\langle f(\{Q_{ab}\})\rangle_Q$ for a function f as

$$\int \prod_{a\le b} dQ_{ab}\, f(\{Q_{ab}\})\Phi(\{Q_{ab}\}).$$

The density

$$\Phi=\left\langle\left\langle \frac{1}{V^{r+1}(\vec{w}^*)}\int_{\vec{w}_a\in\mathcal{V}_m(\vec{w}^*)} \prod_{a=1}^{r+1} d\mu(\vec{w}_a)\prod_{a\le b}\delta(Q_{ab}-N^{-1}(\vec{w}_a C\vec{w}_b))\right\rangle_{\vec{w}^*}\right\rangle_{\mathbf{x}^m}$$

is the joint probability density, averaged over all target vectors and training inputs, for the $r(r+1)/2$ weighted inner products Q_{ab} of $r+1$ vectors $\vec{w}_a$ taken randomly from the posterior distribution after the first m training examples.

To calculate the limit $y(\alpha,r)$ of $y(m,r)$ as $m,N\to\infty$ with $\alpha=m/N$, we make the crucial assumption of *replica* symmetry.[16] This means that for any pair a,b of distinct replicas, and for a typical sequence of labels, $N^{-1}\times(\vec{w}_a C\vec{w}_b)$ becomes self-averaging, i.e., nonfluctuating, and converges to a fixed single order parameter $Q_1(\alpha)$, which only depends on the probability distributions of $\vec{x}$ and $\vec{w}^*$. Replacing also the diagonal elements Q_{aa} by the value $Q_0(\alpha)$, the whole average in (5) reduces to the value at $Q_{ab}=Q_1(\alpha)$ and $Q=Q_0(\alpha)$, respectively.

Let $q(\alpha)=Q_1(\alpha)/Q_0(\alpha)$ and $\gamma(\alpha)=\{q(\alpha)/[1-q(\alpha)]\}^{1/2}$. Inserting the replica-symmetric ansatz into (5), one finally obtains

$$y(\alpha,r)=2\int_{-\infty}^{\infty} Dt\, H^{r+1}(t\gamma(\alpha)) \tag{6}$$

with $Dt=(2\pi)^{-1/2}\exp(-t^2/2)$ and $H(z)=\int_z^{\infty} Dt$.

Before calculating $q(\alpha)$ for a specific distribution of inputs and prior measure $d\mu(\vec{w})$, we derive a general relation between ϵ_{Boltz} and ϵ_{Bayes}. Let $\epsilon_{\text{Boltz}}(\alpha)$ denote the limit of $\epsilon_{\text{Boltz}}(m)$ as $m,N\to\infty$ with $\alpha=m/N$, and similarly for $\epsilon_{\text{Bayes}}(\alpha)$. From (1), (3), and (6) we get

$$\epsilon_{\text{Boltz}}(\alpha)=2\int_{-\infty}^{\infty} Dt[H(t\gamma(\alpha))-H^2(t\gamma(\alpha))]=\pi^{-1}\arccos[q(\alpha)], \tag{7}$$

$$\epsilon_{\text{Bayes}}(\alpha)=2\int_{-\infty}^{\infty} Dt\, H(t\gamma(\alpha))\Theta[1-2H(t\gamma(\alpha))]=\pi^{-1}\arccos\{[q(\alpha)]^{1/2}\}. \tag{8}$$

By eliminating q one arrives at the result

$$\epsilon_{\text{Bayes}}(\alpha)=\pi^{-1}\arccos\{\cos^{1/2}[\pi\epsilon_{\text{Boltz}}(\alpha)]\}. \tag{9}$$

For large α, i.e., small generalization error, the ratio $\epsilon_{\text{Boltz}}(\alpha)/\epsilon_{\text{Bayes}}(\alpha)$ converges to $\sqrt{2}$. On the other hand, for $\alpha=0$, both algorithms have generalization error 0.5, which is equivalent to random guessing. However, as α grows positive, the error of the Bayes algorithm drops much faster from its random guessing value than the error of the Boltzmann algorithm. For small α we have $0.5-\epsilon_{\text{Bayes}}(\alpha)\approx\{\pi[0.5-\epsilon_{\text{Boltz}}(\alpha)]\}^{1/2}$. In Fig. 1 we display $\epsilon_{\text{Bayes}}(\alpha)$ together with the corresponding results for $\epsilon_n(\alpha)$ for $n=3,7,21$ as a function of $\epsilon_{\text{Boltz}}(\alpha)$.

In the case of perceptrons with discrete weights our result holds down to a critical value of $\epsilon_{\text{Boltz}}^{\text{crit}}(\alpha)$, below which a discontinuous transition to perfect generalization $\epsilon_{\text{Boltz}}(\alpha)=0$ is known to take place.[9,17]

Finally, we give results for a uniform spherical distribution of the inputs, i.e., $C_{ij}=\delta_{ij}$, and corresponding uniform spherical prior distribution on the target vector $\vec{w}^*$ (see Fig. 2). In this case the order parameter q becomes identical to the usual *Edwards-Anderson* parameter.[18] It naturally appears in the calculation of the average entropy $S(m)=\langle\langle \ln V_m(\mathbf{x}^m,\vec{w}^*)\rangle_{\vec{w}^*}\rangle_{\mathbf{x}^m}$.

For the spherical distributions this has been calculated in Ref. 7. As the result one finds $q(\alpha)$ by extremizing

$$s(\alpha)=\lim_{N\to\infty} S(m)/N = \tfrac{1}{2}\ln[1-q(\alpha)]+\tfrac{1}{2}q(\alpha) + 2\alpha\int_{-\infty}^{\infty} Dt\, H(t\gamma(\alpha))\ln[H(t\gamma(\alpha))].$$

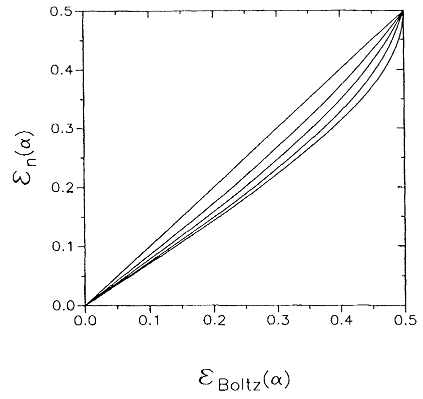

FIG. 1. Generalization errors of the n-hidden-unit network with majority output as a function of $\epsilon_{\mathrm{Boltz}}(\alpha)$ for $n=1,3,7,21,\infty$. The case $n=1$ (diagonal line) corresponds to the Boltzmann algorithm itself, and the case $n=\infty$ (lowest curve) corresponds to the Bayes optimal algorithm.

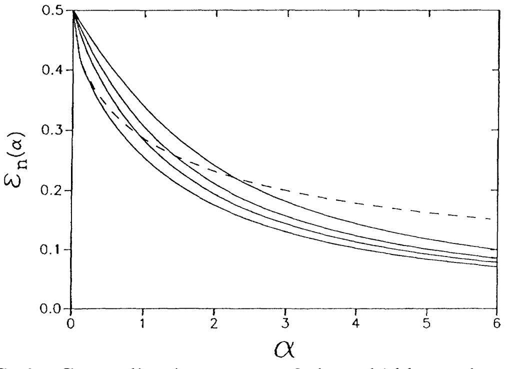

FIG. 2. Generalization errors of the n-hidden-unit network with majority output as a function of α for $n=1,3,7,\infty$, assuming a spherically symmetric distribution on target vectors and inputs. The dashed line is the error of the Hebbian rule as obtained in Ref. 6. The case $n=1$ (highest solid curve) was recently calculated in Ref. 7 and corresponds to the Boltzmann algorithm. The case $n=\infty$ (lowest solid curve) corresponds to the Bayes optimal algorithm.

Plugging this value of $q(\alpha)$ into (7) and (8), and solving numerically, we find that $\epsilon_{\mathrm{Boltz}}(\alpha)\approx 0.62/\alpha$ and $\epsilon_{\mathrm{Bayes}}(\alpha)\approx 0.44/\alpha$ as α goes to ∞. The asymptotic generalization error of the deterministic learning algorithm that finds the hypothesis of "optimal stability," determined to be $\approx 0.57/\alpha$ in Ref. 10, lies between these values. As α approaches 0, the deviation $0.5-\epsilon_{\mathrm{Bayes}}(\alpha)$ of the error of the Bayes algorithm from purely random guessing is $\approx(2\alpha/\pi^3)^{1/2}$, which has infinite slope at $\alpha=0$, whereas for the Boltzmann algorithm this quantity is linear in α with slope $\approx 2/\pi^2$. Remarkably, for this special choice of distributions even the simplest deterministic algorithm, the Hebbian rule $\vec{w}=\sum_{k=1}^{m}\vec{x}_k\sigma_k$, achieves the same asymptotic error for small α.[6,10]

In conclusion, we have shown that the Bayes algorithm performs significantly better than the Boltzmann algorithm when learning a perceptron without noise. Our results can be extended to the case where there is noise in the training examples without significant change in the methodology. They can also be used to investigate the relationship between the generalization error and the average one-step information gain.[7]

Our results show that the relationship between the generalization error of the Bayes and the Boltzmann algorithms is independent of the particular assumptions made about the densities used to choose the target perceptron and to generate the training examples, as long as a central limit theorem holds and the replica-symmetry assumptions made above are valid. This also holds for the approximations to the Bayes algorithm we have defined using neural nets with a layer of hidden units and a majority-gate output unit as well. Thus we expect that the quantitative relationships shown in Fig. 1 are quite robust, and that the use of two-layer nets with majority output may give better generalization performance.

Helpful discussions with Haim Sompolinsky and Naftali Tishby are gratefully acknowledged.

[1]E. Gardner, J. Phys. A **21**, 257 (1988).
[2]E. Gardner and B. Derrida, J. Phys. A **21**, 271 (1988).
[3]J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield, Complex Syst. **1**, 877 (1987).
[4]D. Haussler, N. Littlestone, and M. Warmuth, in *Proceedings of the Twenty-Ninth Annual IEEE Symposium on Foundations of Computer Science* (IEEE, Washington, DC, 1988), p. 100.
[5]E. Baum and D. Haussler, Neural Comput. **1**, 151 (1989).
[6]F. Vallet, J. Cailton, and P. Refregier, Europhys. Lett. **9**, 315 (1989).
[7]G. Gyorgyi and N. Tishby, in *Neural Networks and Spin Glasses,* edited by K. Thuemann and R. Koeberle (World Scientific, Singapore, 1990).
[8]D. Hansel and H. Sompolinsky, Europhys. Lett. **11**, 687 (1990).
[9]H. Sompolinsky, N. Tishby, and H. S. Seung, Phys. Rev. Lett. **65**, 1683 (1990).
[10]M. Opper, W. Kinzel, J. Kleinz, and R. Nehl, J. Phys. A **23**, L581 (1990).
[11]G. Gyorgyi, Phys. Rev. Lett. **64**, 2957 (1990).
[12]T. M. Mitchell, Art. Intell. **18**, 203 (1982).
[13]E. Levin, N. Tishby, and S. Solla, in *Proceedings of the Second Workshop on Computational Learning Theory,* edited by R. Rivest (Morgan Kaufmann, San Mateo, CA, 1989).
[14]N. Littlestone, Ph.D. thesis, University of California at Santa Cruz, 1989 (unpublished).
[15]N. Littlestone and M. Warmuth, in *Proceedings of the Thirtieth Annual IEEE Symposium on Foundations of Computer Science* (IEEE, Washington, DC, 1989), p. 256.
[16]M. Mezard, G. Parisi, and M. A. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).
[17]G. Gyorgyi, Phys. Rev. A **41**, 7097 (1990).
[18]S. Kirkpatrick and D. Sherrington, Phys. Rev. B **17**, 4384 (1978).