Calculation of the Learning Curve of Bayes Optimal Classification Algorithm for Learning a Perceptron With Noise

Manfred Opper
Institut fuer Theoretische Physik
Justus-Liebig-Universitaet Giessen
Giessen, Germany
maopper@dgihrz01.bitnet

David Haussler
Computer and Information Sciences
U.C. Santa Cruz
Santa Cruz, CA 95064
haussler@cis.ucsc.edu

Abstract

The learning curve of Bayes optimal classification algorithm when learning a perceptron from noisy random training examples is calculated exactly in the limit of large training sample size and large instance space dimension using methods of statistical mechanics. It is shown that under certain assumptions, in this "thermodynamic" limit, the probability of misclassification of Bayes optimal algorithm is less than that of a canonical stochastic learning algorithm, by a factor approaching $\sqrt{2}$ as the ratio of number of training examples to instance space dimension grows. Exact asymptotic learning curves for both algorithms are derived for particular distributions. In addition, it is shown that the learning performance of Bayes optimal algorithm can be approximated by certain learning algorithms that use a neural net with a layer of hidden units to learn a perceptron.

1 Introduction

Extending a line of research initiated by Elizabeth Gardner [Gar88, GD88], 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 [DSW+87] [HLW88] [BH89] [VJP89] [LTS89] [GT90] [HS90] [STS90] [OKKN90] has focused on quantifying what is known in the neural net literature as the *generalization performance* of learning algorithms. This is the probability that the learning algorithm will correctly predict the classification of a new random instance, after it has seen a certain number of random classified instances, called *training examples*. In other literature, this is referred to as the *probability of a mistake* or the *expected 0-1 loss*.

Most neural net learning algorithms make predictions on novel instances 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 *Gibbs algorithm*[1], was studied from a statistical mechanics perspective in [GT90, HS90, STS90], and in a more abstract setting in [LW89] (as the *randomized weighted majority algorithm*) and [HKS91]. For noise-free training examples, the extreme "zero temperature" version of this algorithm simply chooses a hypothesis at random from among those that are consistent with all the training examples, as in [Maa91]. Here we apply similar methods from statistical physics to study *Bayes optimal classification algorithm*, a special case of the *weighted majority algorithm* [Lit89, LW89, Vov90] (see also [DMW88]). Further investigation of the Bayes and Gibbs algorithms appears in [HKS91], from both an information theory and a Vapnik-Chervonenkis theory perspective.

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 target function is modeled by assuming an *a priori probability distribution* over possible target functions. When there is noise in the examples, *a priori* information about the nature of this noise is also incorporated. One then seeks a learning algorithm that will give the best *average* generalization performance (i.e. minimum average loss) on target functions and noise processes selected according to this *a priori* distribution. This is what Bayes optimal classification algorithm does. The performance of Bayes algorithm provides the natural standard against which other algorithms may be compared.

In this paper we derive expressions for the average generalization performance for both Bayes algorithm and the Gibbs algorithm for the simplest neural network: the single layer perceptron. We assume that a target perceptron is selected at random according to a prior distribution, and that noisy training examples are generated from this target, where classification label of each example is flipped independently with some probability $0 \leq \lambda \leq 1/2$. The noise-free case ($\lambda = 0$) was investigated previously in [OH91].

[1] This algorithm was called the *Boltzmann algorithm* in [OH91].

In the noisy case, one can measure the probability of misclassification as either the probability that a mistake is made in predicting the noisy label, or the probability that a mistake is made predicting the underlying classification label, before the noise is added. Both of these are with respect to the random choice of the target concept, the random choice of the training instances, the random noise added to the training instances, the random choice of the test instance, and any internal randomization in the algorithm. Let us adopt the latter probability as our notion of the probability of misclassification here and in the following paragraph. (We look at both notions in this paper.) Let N denote the dimension of the instance 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 Gibbs algorithm is a factor $\sqrt{2}$ from the optimal performance of Bayes algorithm, asymptotically as α becomes large. Here we make minimal assumptions on the the *a priori* density on the weight space used to select the target function, and on the density used to select the training examples.

When the latter densities are chosen to be uniform on the surface of the sphere, then we can give explicit formulae for the probability of misclassification as a function of α. For large α and noise-free training examples, this probability is approximately $0.44/\alpha$ for Bayes algorithm and $0.62/\alpha$ for the Gibbs algorithm. These results show that the general upper bounds derived in [HKS91] for the performance of the Bayes and Gibbs algorithms on hypothesis spaces of finite VC dimension are tight to within a relatively small constant in this case. When the training examples are corrupted by random classification noise with rate λ, the asymptotic probability of misclassification for the Gibbs algorithm is given by $C(\lambda)/\alpha$, where the function $C(\lambda)$ is (actually $C(\lambda)(1-2\lambda)^2$) is as plotted in Figure 4. As mentioned above, we show that the probability of misclassification for Bayes algorithm is less by a factor of $\sqrt{2}$.

It turns out that 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 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 just takes a majority vote of the hidden units. For $n = 1$ we get the Gibbs algorithm, and as $n \to \infty$, the performance of these algorithms approaches Bayes optimal.

2 Results

2.1 Basic Definitions

In the simple binary classification learning problem we consider, one tries to learn a *target function* f^* that maps from a set X (the *instance 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 instance $\vec{x}$. Each possible setting of the vector of weights $\vec{w}$ in the neural network defines a classification function $f_{\vec{w}}$ from X into $\{-1, +1\}$. For the case of the single layer perceptron

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

where $\mathrm{sign}\,(x) = +1$ if $x > 0$ and $\mathrm{sign}\,(x) = -1$ if $x \leq 0$. The division by $\sqrt{N}$ is not really necessary here, but it will be convenient later, since we restrict the weight vector to the sphere $\vec{w} \cdot \vec{w} = N$, and this effectively normalizes $\vec{w}$ to unit length.

Learning a perceptron was recently investigated by several authors from a statistical mechanics perspective [VJP89, Gyo90b, GT90, OKKN90]. 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 instances $\mathbf{x}^{m+1} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{m+1})$ is selected at random. Noisy classification labels $\sigma^{m+1}(\vec{w}^*, \eta^{m+1}) = \sigma^{m+1} = (\sigma_1, \sigma_2, \ldots, \sigma_{m+1})$ are generated, where $\sigma_k = \eta_k f_{\vec{w}^*}(\vec{x}_k)$, $k = 1, 2, \ldots, m+1$, and $\eta^{m+1} = (\eta_1, \eta_2, \ldots, \eta_{m+1})$ is a sequence of independent identically distributed $\{\pm 1\}$-valued random noise variables. We assume that $\eta_k = -1$ with probability λ and $\eta_k = +1$ with probability $1 - \lambda$, where $0 \leq \lambda \leq 1/2$. Thus the sign of each classification label is flipped independently with probability λ. We call λ the *noise rate*.

Given only training examples

$$(\vec{x}_1, \sigma_1), (\vec{x}_2, \sigma_2), \ldots, (\vec{x}_m, \sigma_m)$$

and instance $\vec{x}_{m+1}$, the learning algorithm must predict the label σ_{m+1}. The generalization error $\epsilon(m, \lambda)$ of the learning algorithm is the probability that it predicts wrong, as a function of the training sample size m and the noise rate λ. Note that in this formulation the object is to predict the noisy classification label. Clearly no algorithm can achieve a success rate better than λ at this task.

An alternate formulation is to consider the generalization error to be the probability that the prediction of the learning algorithm on instance $\vec{x}_{m+1}$ differs from the underlying noise-free classification $f_{\vec{w}^*}(\vec{x}_{m+1})$. We denote this probability by $\delta(m, \lambda)$. It is clear that

$$\begin{aligned} \epsilon(m, \lambda) &= (1-\lambda)\delta(m, \lambda) + \lambda(1 - \delta(m, \lambda)) \\ &= \lambda + (1 - 2\lambda)\delta(m, \lambda), \end{aligned} \tag{1}$$

so either one of these quantities is easily obtained from the other.

2.2 Posterior Density

For now, let us assume that the instance sequence $\mathbf{x}^{m+1}$ 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 $d\mu(\vec{w}^*)$, and the sequence of noise events η^{m+1}, which is chosen as described above. For each $m \geq 1$ let

$$\Delta(\vec{w}, \sigma^m) = \Delta(\vec{w}, \sigma^m(\vec{w}^*, \eta^m))$$

be the number of times in the first m examples that $\sigma_i = \eta_i f_{\vec{w}^*}(\vec{x}_i) \neq f_{\vec{w}}(\vec{x}_i)$, i.e. the number of labels in $\sigma^m = (\sigma_1, \ldots, \sigma_m)$ that are predicted incorrectly by the hypothesis represented by $\vec{w}$.

Fix a constant $\beta > 0$. The constant β plays the role of an inverse temperature from a statistical mechanics viewpoint; large β represents low temperature and small β represents high temperature. From a learning point of view, β will govern the tradeoff between goodness of fit to the sample data and *a priori* plausibility of the hypothesis. Large β will force the algorithm, when predicting σ_{m+1}, to use hypotheses that fit the first m examples well (i.e. those with low Δ), even if they are *a priori* unlikely to be the target. Small β will allow the combined effects of the *a priori* most likely hypotheses to carry more weight, so long as they don't have too large a Δ.

Specifically, for each m define the *posterior density*

$$d\mu_m(\vec{w}) = d\mu(\vec{w}) e^{-\beta\Delta(\vec{w}, \sigma^m(\vec{w}^*, \eta^m))}/Z_m,$$

where

$$\begin{aligned} Z_m = Z(\sigma^m) &= Z(\mathbf{x}^m, \sigma^m(\vec{w}^*, \eta^m)) \\ &= \int e^{-\beta\Delta(\vec{w}, \sigma^m(\vec{w}^*, \eta^m))} d\mu(\vec{w}) \end{aligned}$$

is the normalizing constant for this density. In statistical mechanics, Z_m is called the *partition function*. Note that in the posterior density, the (unnormalized) weight of a hypothesis is reduced exponentially in proportion to the number of times it is incorrect on the training sequence, as in the weighted majority algorithm [LW89]. For noise rate λ, we can easily show that when $\beta = \ln((1-\lambda)/\lambda)$, the density $d\mu_m$ is the correct Bayesian posterior density over all possible target vectors $\vec{w}$, assuming the prior density is $d\mu(\vec{w})$ and we are given the examples $(\vec{x}_1, \sigma_1), (\vec{x}_2, \sigma_2), \ldots, (\vec{x}_m, \sigma_m)$. (This is also shown in [Lit89]). This means that for any (measurable) set of weight vectors W, $\int_{\vec{w} \in W} d\mu_m(\vec{w})$ is the conditional probability that the target vector was chosen from W, given the observed examples $(\vec{x}_1, \sigma_1), (\vec{x}_2, \sigma_2), \ldots, (\vec{x}_m, \sigma_m)$. To see this, first calculate the unconditional probability of the label sequence σ^m by conditioning over possible targets $\vec{w}$, weighted by the prior. This gives

$$\begin{aligned} \Pr(\sigma^m) &= \int \lambda^{\Delta(\vec{w}, \sigma^m)} (1-\lambda)^{m - \Delta(\vec{w}, \sigma^m)} d\mu(\vec{w}) \\ &= (1-\lambda)^m \int e^{-\beta\Delta(\vec{w}, \sigma^m)} d\mu(\vec{w}) \\ &= (1-\lambda)^m Z_m. \end{aligned} \tag{2}$$

Therefore

$$\begin{aligned} \Pr(\vec{w}^* \in W | \sigma^m) &= \frac{\Pr(\sigma^m | \vec{w}^* \in W) \Pr(\vec{w}^* \in W)}{\Pr(\sigma^m)} \\ &= \frac{\int_{\vec{w} \in W} \lambda^{\Delta(\vec{w}, \sigma^m)} (1-\lambda)^{m - \Delta(\vec{w}, \sigma^m)} d\mu(\vec{w})}{(1-\lambda)^m Z_m} \\ &= \frac{\int_{\vec{w} \in W} e^{-\beta\Delta(\vec{w}, \sigma^m)} d\mu(\vec{w})}{Z_m} \\ &= \int_{\vec{w} \in W} d\mu_m(\vec{w}). \end{aligned}$$

In the limit when $\beta = \infty$ (the *zero temperature limit*), the posterior density $d\mu_m$ is zero everywhere except on the set $\{\vec{w} : f_{\vec{w}}(\vec{x}_k) = f_{\vec{w}^*}(\vec{x}_k), \text{ for } k = 1, \ldots, m\}$ of all weight vectors that are *consistent* with the first m training examples. This is called the *version space* in the AI literature [Mit82]. Thus in the zero temperature limit, all hypotheses that contradict even one training example are eliminated from consideration. The volume of the remaining version space, as a fraction of the volume of the original hypothesis space, is given by the partition function Z_m, which in this case is just the measure of the version space under the *a priori* density $d\mu$. (For the zero temperature case this volume is denoted by V_m in [OH91, HKS91].) Even in the finite β case, it is useful to think of Z_m as a kind of *a posteriori* volume measure on the hypothesis space, which decreases as the number m of training examples increases.

2.3 Gibbs and Bayes Algorithms

In order to make its prediction on the instance $\vec{x}_{m+1}$, the *Gibbs algorithm* chooses a hypothesis $\vec{w}$ at random according to the posterior density $d\mu_m$ on the hypothesis space and predicts according to this hypothesis. This is the stochastic learning algorithm discussed in [GT90, STS90, LW89].

Let

$$Z_m^{right} = \int_{\{\vec{w} : f_{\vec{w}}(\vec{x}_{m+1}) = \sigma_{m+1}\}} e^{-\beta\Delta(\vec{w}, \sigma^m(\vec{w}^*, \eta^m))} d\mu(\vec{w}),$$

i.e. the *a posteriori* volume (after the first m examples) of those hypotheses that predict correctly on the $m+1$st example. Let

$$Z_m^{wrong} = \int_{\{\vec{w} : f_{\vec{w}}(\vec{x}_{m+1}) \neq \sigma_{m+1}\}} e^{-\beta\Delta(\vec{w}, \sigma^m(\vec{w}^*, \eta^m))} d\mu(\vec{w}),$$

i.e. the *a posteriori* volume of those that predict wrong. Clearly

$$Z_m = Z_m^{right} + Z_m^{wrong}.$$

Note also that

$$Z_{m+1} = Z_m^{right} + e^{-\beta} Z_m^{wrong}.$$

Hence

$$Z_m - Z_{m+1} = (1 - e^{-\beta}) Z_m^{wrong}.$$

Thus, since the Gibbs algorithm chooses its hypothesis at random according to the posterior density $d\mu_m$, it makes a mistake in predicting σ_{m+1} with probability

$$\frac{Z_m^{wrong}}{Z_m} = \frac{1}{1 - e^{-\beta}} \left(1 - \frac{Z_{m+1}}{Z_m} \right). \tag{3}$$

A similar formulation has been obtained in [LTS89] and [LW89].

The average generalization error of the Gibbs algorithm, when the target vector $\vec{w}^*$ is chosen at random by $d\mu(\vec{w}^*)$ and the noise sequence η^{m+1} is generated randomly with noise rate λ, but the first $m+1$ instances $\mathbf{x}^{m+1} = (\vec{x}_1, \ldots, \vec{x}_{m+1})$ are fixed, is thus given by

$$\begin{aligned} & \epsilon_{Gibbs}(\mathbf{x}^{m+1}, \lambda) \\ &= \frac{1}{1 - e^{-\beta}} \left\langle 1 - \frac{Z(\mathbf{x}^{m+1}, \sigma^{m+1})}{Z(\mathbf{x}^m, \sigma^m)} \right\rangle_{\vec{w}^*, \eta^{m+1}} \\ &= \frac{1}{1 - e^{-\beta}} \left\langle 1 - \frac{Z_{m+1}}{Z_m} \right\rangle_{\vec{w}^*, \eta^{m+1}}, \end{aligned} \tag{4}$$

where $\langle \rangle_{\vec{w}^*, \eta^{m+1}}$ denotes integration over $d\mu(\vec{w}^*)$ and average over the noise values $\eta_1, \ldots \eta_{m+1}$.

The present formulation treats integrations over $\vec{w}$ and averages over $\vec{w}^*$ and η^{m+1} in a nonsymmetric way. To facilitate the subsequent calculations we can remove this asymmetry by replacing the average over targets and noise by an equivalent average over all possible labelings $\sigma^m = (\sigma_1, \sigma_2, \ldots, \sigma_m) \in \{\pm 1\}^m$. We do so by fixing $\beta = \ln((1-\lambda)/\lambda)$ as above. Note that this implies that

$$1 - \lambda = \frac{1}{1 + e^{-\beta}}. \tag{5}$$

Then since $Z_m = Z(\sigma^m)$, from Equation (2) we have

$$\Pr(\sigma^m) = \frac{Z(\sigma^m)}{(1 + e^{-\beta})^m}. \tag{6}$$

Thus from Equation (4) we get

$$\begin{aligned} & \epsilon_{Gibbs}(\mathbf{x}^{m+1}, \lambda) \\ &= \frac{1}{1 - e^{-\beta}} \sum_{\sigma^{m+1} \in \{\pm 1\}^{m+1}} \Pr(\sigma^{m+1}) \left(1 - \frac{Z_{m+1}}{Z_m} \right) \\ &= \frac{1}{(1 + e^{-\beta})^{m+1} (1 - e^{-\beta})} \cdot \\ & \qquad \sum_{\sigma^{m+1} \in \{\pm 1\}^{m+1}} Z(\sigma^{m+1}) \left(1 - \frac{Z(\sigma^{m+1})}{Z(\sigma^m)} \right). \end{aligned} \tag{7}$$

If the goal is to maximize the probability of a correct prediction, it turns out that the Gibbs 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 comparing this with the corresponding posterior probability for -1. One then chooses the output with largest posterior probability. Since the noise rate λ is at most one half, this clearly maximizes the probability of a correct prediction. This strategy is known as *Bayes optimal classification algorithm* or *Bayes algorithm* for short. In its general form, where β is not necessarily set according to the noise rate λ, the algorithm is called the weighted majority algorithm [Lit89, LW89].

It is clear that Bayes algorithm makes an mistake in predition only when $Z_m^{wrong} \geq Z_m^{right}$. A little algebra shows that this is equivalent to

$$\frac{1 + e^{-\beta}}{2} - \frac{Z_{m+1}}{Z_m} \geq 0.$$

Thus the average generalization error for Bayes algorithm is the expectation of[2]

$$\Theta \left(\frac{1 + e^{-\beta}}{2} - \frac{Z_{m+1}}{Z_m} \right)$$

when the target $\vec{w}^*$ and the noise sequence η^{m+1} are randomly chosen, where $\Theta(x)$ is the unit step function, i.e. $\Theta(x) = 1$ if $x \geq 0$, $\Theta(x) = 0$ if $x < 0$.

As above, we can write this in a more symmetric form:

$$\begin{aligned} & \epsilon_{Bayes}(\mathbf{x}^{m+1}, \lambda) \\ &= \left\langle \Theta \left(\frac{1 + e^{-\beta}}{2} - \frac{Z_{m+1}}{Z_m} \right) \right\rangle_{\vec{w}^*, \eta^{m+1}} \\ &= \frac{1}{(1 + e^{-\beta})^{m+1}} \cdot \\ & \qquad \sum_{\sigma^{m+1} \in \{\pm 1\}^{m+1}} Z(\sigma^{m+1}) \Theta \left(\frac{1 + e^{-\beta}}{2} - \frac{Z(\sigma^{m+1})}{Z(\sigma^m)} \right) \end{aligned} \tag{8}$$

When learning a perceptron, the hypothesis used by Bayes algorithm cannot itself be represented by a perceptron. However, we can construct a network with one hidden layer for which the average generalization 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 Gibbs algorithm and make them hidden units. Each hidden unit output $\sigma(i)$, $i = 1, \ldots, n$, is passed to

[2] Here we assume that the probability that $\frac{1+e^{-\beta}}{2} = \frac{Z_{m+1}}{Z_m}$ is zero, so it doesn't matter what value we choose for $\Theta(0)$. A more general treatment is given in [HKS91].

a fixed output perceptron that computes the majority function $\sigma_{maj} = \mathrm{sign}\left(\sum_{i=1}^{n} \sigma(i)\right)$ as the final output. Using Equation (3), for a fixed target concept $\vec{w}^*$, the probability that k perceptrons make the right prediction on $\vec{x}_{m+1}$ and $n-k$ the wrong prediction is given by the binomial distribution

$$\begin{aligned} a_k &= \binom{n}{k}\left(1-\frac{1-Z_{m+1}/Z_m}{1-e^{-\beta}}\right)^k\left(\frac{1-Z_{m+1}/Z_m}{1-e^{-\beta}}\right)^{n-k} \\ &= \frac{1}{(1-e^{-\beta})^n}\binom{n}{k}\left(\frac{Z_{m+1}}{Z_m}-e^{-\beta}\right)^k\left(1-\frac{Z_{m+1}}{Z_m}\right)^{n-k} \end{aligned}$$

Thus the probability that the majority vote makes a mistake in prediction is $\sum_{k=0}^{\lfloor n/2 \rfloor} a_k$. It follows that the average generalization error of the learning algorithm with n hidden units is given by

$$\epsilon_n(\mathbf{x}^{m+1},\lambda) = \left\langle \sum_{k=0}^{\lfloor n/2 \rfloor} a_k \right\rangle_{\vec{w}^*,\eta^{m+1}} \tag{9}$$

As $n \to \infty$, $\sum_{k=0}^{\lfloor n/2 \rfloor} a_k$ converges to

$$\Theta\left(\frac{1}{2}-\frac{1-Z_{m+1}/Z_m}{1-e^{-\beta}}\right) = \Theta\left(\frac{1+e^{-\beta}}{2}-\frac{Z_{m+1}}{Z_m}\right)$$

whenever $\frac{1-Z_{m+1}/Z_m}{1-e^{-\beta}} \neq 1/2$. Taking the expectation with respect to $\vec{w}^*$ and η^{m+1}, as in (9), we recover in this limit the average generalization error of Bayes algorithm given in (8). Thus by varying the number n of hidden units, we define a sequence learning of algorithms with average generalization errors $\epsilon_n(\mathbf{x}^{m+1},\lambda)$, $n = 1,3,5,\ldots$ with $\epsilon_1(\mathbf{x}^{m+1},\lambda) = \epsilon_{Gibbs}(\mathbf{x}^{m+1},\lambda)$ and $\epsilon_\infty(\mathbf{x}^{m+1},\lambda) = \epsilon_{Bayes}(\mathbf{x}^{m+1},\lambda)$.

2.4 Calculation of the Learning Curves

In the following we now assume that the instances $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{m+1}$ are drawn independently at random from an N-dimensional probability distribution. In particular, we assume that for each i and j, where $1 \leq i,j \leq N$, the cartesian components $x_k(i)$ and $x_k(j)$ of $\vec{x}_k$, $k = 1,2,\ldots,m+1$, are i.i.d. random variables with zero mean and finite covariance C_{ij}. This means that the coordinate system is chosen such that the mean of the distribution on the instance space is at the origin. For each n, the average generalization error for the n hidden unit learning algorithm is denoted by $\epsilon_n(m,\lambda) = \left\langle \epsilon_n(\mathbf{x}^{m+1},\lambda)\right\rangle_{\mathbf{X}^{m+1}}$, where $\langle\rangle_{\mathbf{X}^{m+1}}$ denotes the average over the random selection of $\vec{x}_1,\ldots,\vec{x}_{m+1}$.

For each m let

$$Y_m = Y(\mathbf{x}^{m+1},\sigma^{m+1}(\vec{w}^*,\eta^{m+1})) = \frac{Z(\mathbf{x}^{m+1},\sigma^{m+1})}{Z(\mathbf{x}^m,\sigma^m)}$$

From Equation (9) it is clear that for each n, the average generalization error $\epsilon_n(m,\lambda)$ can be written as the expectation of a polynomial function of Y_m. This expectation can be calculated from the positive integer moments of the random variable Y_m. To facilitate this calculation, we abbreviate the the rth moment of Y_m by

$$y(m,r) = \left\langle \left\langle Y^r(\mathbf{x}^{m+1},\sigma^{m+1}(\vec{w}^*,\eta^{m+1}))\right\rangle_{\vec{w}^*,\eta^{m+1}}\right\rangle_{\mathbf{X}^{m+1}}$$

Averaging directly over possible label sequences σ^{m+1} we obtain

$$\begin{aligned} y(m,r) &= \left\langle \sum_{\sigma^{m+1}\in\{\pm1\}^{m+1}} \Pr(\sigma^{m+1})\frac{Z^r(\mathbf{x}^{m+1},\sigma^{m+1})}{Z^r(\mathbf{x}^m,\sigma^m)}\right\rangle_{\mathbf{X}^{m+1}} \\ &= \frac{1}{(1+e^{-\beta})^{m+1}}\left\langle \sum_{\sigma^{m+1}\in\{\pm1\}^{m+1}} \frac{Z^{r+1}(\mathbf{x}^{m+1},\sigma^{m+1})}{Z^r(\mathbf{x}^m,\sigma^m)}\right\rangle_{\mathbf{X}^{m+1}} \end{aligned} \tag{10}$$

for $r = 0,1,2\ldots$, using Equation (6).

The specific calculation of $y(m,r)$ for the perceptron and subsequent reconstruction of $\epsilon_n(m,\lambda)$ is somewhat lengthy, so we only sketch it. First one finds that for the perceptron,

$$\begin{aligned} &Z^{r+1}(\mathbf{x}^{m+1},\sigma^{m+1}) \\ &= \int \prod_{a=1}^{r+1} d\mu(\vec{w}_a) \prod_{k=1}^{m+1}\prod_{a=1}^{r+1}(e^{-\beta}+(1-e^{-\beta})\Theta((\vec{w}_a\cdot\vec{x}_k)\sigma_k/\sqrt{N})) \end{aligned} \tag{11}$$

The expression $\frac{Z^{r+1}(\mathbf{x}^{m+1},\sigma^{m+1})}{(1+e^{-\beta})^{(m+1)(r+1)}}$ can be interpreted as the probability that $r+1$ independent random *replicas* of the network weights and noise values give the same output $\sigma_1,\ldots,\sigma_{m+1}$ on inputs $\vec{x}_1,\ldots,\vec{x}_{m+1}$.

We use Equations (10) and (11) to calculate $y(m,r)$. However, we can calculate $y(m,r)$ exactly only in the limit of large instance space dimension N, and then only by making some further assumptions on the distribution over the instance space and the prior density $d\mu$ on the hypothesis space. In particular, we assume that for almost all sets of $r+1$ vectors $\vec{w}_a$, $a = 1,\ldots,r+1$, drawn randomly and independently according to $d\mu$, the random vector with components $u_a(\vec{x}) = \vec{w}_a\cdot\vec{x}/\sqrt{N}$, $a = 1,\ldots,r+1$, obeys the multivariate *central limit theorem*, i.e. for large N, the joint probability density of the u_a is approximately a multivariate Gaussian distribution, and in the limit as $N \to \infty$ it is exactly Gaussian. Since we are assuming also that the mean of the distribution over $\vec{x}$ is the origin, for all $\vec{w}$ the means of the corresponding random variables u_a are zero. These assumptions are satisfied for a wide class of distributions. For example, they hold asymptotically for large N when the distribution over the instance space is discrete and uniform over the Boolean cube $\{\pm1\}^N$, and the density $d\mu$ is a symmetric function of the cartesian components $w(j)$,

with finite variance. This includes a uniform distribution on the sphere of radius $\sqrt{N}$, as well as a discrete, uniform distribution on the Boolean cube. The same results hold for the case when the distribution on the instance space is also continuous and uniform on the sphere.

For $\vec{w}_1, \ldots, \vec{w}_{r+1}$ drawn randomly from $d\mu$, let the $r+1$ by $r+1$ matrix-valued random variable Q be defined by $Q_{ab} = N^{-1}(\vec{w_a} C \vec{w_b})$, for $1 \leq a, b, \leq r+1$, where $C = \{C_{ij}\}$ is the N by N covariance matrix for the distribution on the instance space. Under our Gaussian assumption, Q_{ab} is the covariance of the random variables $u_a(\vec{x}) = \vec{w_a} \cdot \vec{x}/\sqrt{N}$ and $u_b(\vec{x}) = \vec{w_b} \cdot \vec{x}/\sqrt{N}$. We can now carry out the average over the $m+1^{st}$ instance in (10) and its two possible outputs $\sigma_{m+1} = \pm 1$ explicitly. The resulting expression can be reexpressed as an average over targets and noise. We find

$$y(m,r) = 2(2\pi)^{-\frac{r+1}{2}}(1+e^{-\beta})^{-1} \cdot \int_{-\infty}^{\infty} \prod_{a=1}^{r+1} du_a (e^{-\beta} + (1-e^{-\beta})\Theta(u_a)) \cdot \left\langle (\det Q)^{-\frac{1}{2}} \exp(-\frac{1}{2}\sum_{a,b} u_a (Q^{-1})_{ab} u_b) \right\rangle_Q , \quad (12)$$

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

$$\int \prod_{a \leq b} dQ_{ab} f(\{Q_{ab}\}) \Phi(\{Q_{ab}\})$$

and the density[3]

$$\Phi = \left\langle \left\langle \int \prod_{a=1}^{r+1} d\mu_m(\vec{w_a}) \prod_{a \leq b} \delta(Q_{ab} - N^{-1}(\vec{w_a} C \vec{w_b})) \right\rangle_{\vec{w}^*, \eta^m} \right\rangle_{\mathbf{X}^m}$$

is the joint probability density, averaged over all target concepts, noise and training instances, 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.

Let $\alpha = m/N$. We will calculate the limit $y(\alpha, r)$ of $y(m,r)$ as $m, N \to \infty$ with α held constant. To do this, we make the crucial assumption of *replica symmetry* [MPV87]. This means that for any pair a, b of distinct replicas and for a typical sequence of labels, $N^{-1}(\vec{w_a} C \vec{w_b})$ becomes self-averaging, i.e. nonfluctuating, and converges to a fixed single orderparameter $Q_1(\alpha)$, which only depends on the probability distributions of $\vec{x}$ and $\vec{w}^*$.

Replacing also the diagonal elements Q_{aa} by the value $Q_0(\alpha)$, under this assumption the whole average in (12) reduces to the value at $Q_{ab} = Q_1(\alpha)$ and $Q_{aa} = Q_0(\alpha)$ respectively.

Let

$$q(\alpha) = Q_1(\alpha)/Q_0(\alpha)$$

and

$$\gamma(\alpha) = \sqrt{q(\alpha)/(1-q(\alpha))}.$$

Inserting the replica symmetric ansatz into (12), one finally obtains

$$y(\alpha, r) = \frac{2}{1+e^{-\beta}} \int_{-\infty}^{\infty} Dt \, [e^{-\beta} + (1-e^{-\beta}) H(t\gamma(\alpha))]^{r+1} \quad (13)$$

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

Note that the distribution of instances and targets enter the result only through $q(\alpha)$. Before calculating $q(\alpha)$ for a specific distribution on the instance space and prior measure $d\mu(\vec{w})$, we derive a general relation between ϵ_{Gibbs} and ϵ_{Bayes}.

First, let $F(Y_m)$ be any polynomial of Y_m (such as the polynomial $\sum_{k=0}^{\lfloor n/2 \rfloor} a_k$ from Section 2.3) and for every $\alpha > 0$ let $\bar{F}_\alpha$ denote the limit of the expectation of $F(Y_m)$ as $m, N \to \infty$ with $\alpha = m/N$ held constant. This expectation is with respect to the random choice of the instances, target concept and noise values. Equation (13) enables us to calculate $\bar{F}_\alpha$. In particular, if

$$F(Y_m) = c_0 + c_1 Y_m + \ldots + c_n Y_m^n$$

then

$$\bar{F}_\alpha = c_0 y(\alpha, 0) + c_1 y(\alpha, 1) + \ldots + c_n y(\alpha, n).$$

It then follows from (13) that

$$\bar{F}_\alpha = \frac{2}{1+e^{-\beta}} \int_{-\infty}^{\infty} Dt \, \hat{H}(t,\alpha) F\left(\hat{H}(t,\alpha)\right), \quad (14)$$

where

$$\hat{H}(t,\alpha) = e^{-\beta} + (1-e^{-\beta}) H(t\gamma(\alpha)).$$

This result is easily extended to any function F that can be approximated by polynomials, including the Θ function.

To apply this result, let $\epsilon_n(\alpha, \lambda)$ denote the limit of $\epsilon_n(m, \lambda)$ as $m, N \to \infty$ with $\alpha = m/N$, and similarly for $\epsilon_{Gibbs}(\alpha, \lambda)$ and $\epsilon_{Bayes}(\alpha, \lambda)$. Then from Equations (9) and (14), using the fact that $H(-x) = 1 - H(x)$ and $\lambda = \frac{e^{-\beta}}{1+e^{-\beta}}$, after some lengthy algebra we obtain

[3] Here we use Dirac's δ function, which is the generalized function defined as the kernel that (for well behaved functions f) yields $f(x) = \int_{-\infty}^{\infty} dt \, f(t) \delta(x-t)$.

$$\epsilon_n(\alpha,\lambda) = \lambda + (2-4\lambda)\sum_{k=0}^{\lfloor n/2\rfloor}\binom{n}{k}\int_{-\infty}^{\infty}\tilde{H}(t,k,\alpha)Dt, \tag{15}$$

where

$$\tilde{H}(t,k,\alpha) = H^{k+1}(t\gamma(\alpha))(1-H(t\gamma(\alpha)))^{n-k}.$$

For $n=1$ and $n=\infty$ this yields

$$\begin{aligned}\epsilon_{Gibbs}(\alpha,\lambda) &= \lambda + (2-4\lambda)\int_{-\infty}^{\infty} Dt\, H(t\gamma(\alpha))\,[1-H(t\gamma(\alpha))] \\ &= \lambda + (1-2\lambda)\pi^{-1}\arccos(q(\alpha)) \end{aligned}\tag{16}$$

and

$$\begin{aligned}\epsilon_{Bayes}(\alpha,\lambda) &= \lambda + (2-4\lambda)\int_{-\infty}^{\infty} Dt\, H(t\gamma(\alpha))\Theta\,[1-2H(t\gamma(\alpha))] \\ &= \lambda + (1-2\lambda)\pi^{-1}\arccos(\sqrt{q(\alpha)}) \end{aligned}\tag{17}$$

From Equation (1), we can now also calculate the probability that these algorithms incorrectly predict the underlying noise-free classification $f_{\vec{w}^*}(\vec{x}_{m+1})$. We get

$$\delta_{Gibbs}(\alpha,\lambda) = \frac{(\epsilon_{Gibbs}(\alpha,\lambda)-\lambda)}{1-2\lambda} = \pi^{-1}\arccos(q(\alpha)) \tag{18}$$

and

$$\delta_{Bayes}(\alpha,\lambda) = \frac{(\epsilon_{Bayes}(\alpha,\lambda)-\lambda)}{1-2\lambda} = \pi^{-1}\arccos(\sqrt{q(\alpha)}) \tag{19}$$

It should be noted that $q(\alpha)$ implicitly depends on the noise rate λ. However, by eliminating $q(\alpha)$ one arrives at the result

$$\delta_{Bayes}(\alpha,\lambda) = \pi^{-1}\arccos\left(\cos^{1/2}(\pi\delta_{Gibbs}(\alpha,\lambda))\right). \tag{20}$$

Remarkably, this functional relation does not depend on the noise rate λ. From Equation (15) one finds that the same is true for all $\delta_n(\alpha,\lambda) = \frac{(\epsilon_n(\alpha,\lambda)-\lambda)}{1-2\lambda}$ as functions of $\delta_{Gibbs}(\alpha,\lambda)$.

In Figure 1 we show $\delta_n(\alpha,\lambda)$ as a function of $\delta_{Gibbs}(\alpha,\lambda)$ for $n=1,3,7,21,\infty$. For large α, the generalization error $\delta_n(\alpha,\lambda)$ converges to zero for all n. In the same limit the ratio $\delta_{Gibbs}(\alpha,\lambda)/\delta_{Bayes}(\alpha,\lambda)$ 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 generalization error of Bayes algorithm drops much faster from its random guessing value than the generalization error of the Gibbs algorithm. For small α we have

$$0.5 - \delta_{Bayes}(\alpha,\lambda) \approx (\pi(0.5-\delta_{Gibbs}(\alpha,\lambda)))^{\frac{1}{2}}.$$

Finally, we give results for a uniform spherical distribution on the instance space, i.e. $C_{ij}=\delta_{ij}$, and corresponding uniform spherical prior distribution on the target vector $\vec{w}^*$ (see Figure 2). In this case the order parameter $q(\alpha)$ becomes identical to the *Edwards-Anderson* parameter [KS78]. It naturally appears in the calculation of the averaged *free energy* $F(m)$. This is defined by

$$\begin{aligned}\beta F(m) &= -\left\langle \langle \ln Z_m\rangle_{\vec{w}^*,\eta^m}\right\rangle_{\mathbf{X}^m} \\ &= -\frac{1}{(1+e^{-\beta})^m}\left\langle \sum_{\sigma^m\in\{\pm1\}^m} Z(\sigma^m)\ln Z(\sigma^m)\right\rangle_{\mathbf{X}^m}\end{aligned}\tag{21}$$

Using the replica trick, this is given by

$$\begin{aligned}\beta F(m) &= -\lim_{n\to1}\frac{\partial}{\partial n}\ln\left\langle \sum_{\sigma^m\in\{\pm1\}^m} Z^n(\sigma^m)\right\rangle_{\mathbf{X}^m}\end{aligned}$$

By a calculation similar to that in [GT90], one finds $q(\alpha)$ by extremizing the expression

$$\begin{aligned} f(\alpha) &= \lim_{m,N\to\infty, m/N=\alpha} -\beta F(m)/N \\ &= \frac{1}{2}\ln(1-q(\alpha)) + \frac{1}{2}q(\alpha) \\ &\quad + \frac{2\alpha}{1+e^{-\beta}}\int_{-\infty}^{\infty}\hat{H}(t,\alpha)\ln\hat{H}(t,\alpha)Dt \end{aligned}\tag{22}$$

where

$$\hat{H}(t,\alpha) = e^{-\beta} + (1-e^{-\beta})H(t\gamma(\alpha)).$$

Plugging this value of $q(\alpha)$ into (18) and (19), and solving for large α, we find that $\delta_{Gibbs}(\alpha,\lambda)\approx C(\lambda)/\alpha$ and $\delta_{Bayes}(\alpha,\lambda)\approx\frac{1}{\sqrt{2}}C(\lambda)/\alpha$ as α goes to ∞, where

$$C(\lambda) = -\frac{\sqrt{2}(1-2\lambda)^{-1}}{\int_{-\infty}^{\infty} z\exp(-\frac{1}{2}z^2)\ln[\lambda+(1-2\lambda)H(z)]dz}. \tag{23}$$

Note that for all λ the asymptotic behavior of the learning curve is $\approx \alpha^{-1}$. A slower convergence ($\approx \alpha^{-\frac{1}{2}}$) of the generalization error has been predicted in [GT90] for a *deterministic* learning algorithm that tries to learn a perceptron with noise by minimizing the number of errors on the first m examples. Though the type of noise discussed in [GT90] is not the same as ours, we expect that the use of a nonzero temperature in the algorithm is essential in obtaining faster convergence. A similar result has been given in [Gyo90b].

The function $C(\lambda)(1-2\lambda)^2$ is shown in Figure 4. Since $C(0) \approx 0.62$, as a special case we obtain the generalization error without classification noise: $\delta_{Gibbs}(\alpha, 0) \approx 0.62/\alpha = 0.62N/m$, as in [OH91]. As already noted, the corresponding Bayes value is smaller by a factor of $\sqrt{2}$.

In Figure 2 the generalization errors for the noise-free case $\lambda = 0$ are depicted as functions of α. As α approaches 0, the deviation $0.5 - \epsilon_{Bayes}(\alpha, 0)$ of the generalization error of Bayes algorithm from purely random guessing is $\approx (2\alpha/\pi^3)^{\frac{1}{2}}$, which has infinite slope at $\alpha = 0$, whereas for the Gibbs algorithm this quantity is linear in α with slope $\approx 2/\pi^2$.

The positive moments of the random variable Y_m contain more information than just the different values of generalization errors of the learning algorithms we have examined. For example, consider the random variable

$$\frac{1}{1-e^{-\beta}}(1-Y_m),$$

which is the probability that the Gibbs algorithm makes the wrong prediction on the $m+1$st classification label for fixed training instances, target function and noise values. The distribution of Y_m gives the fluctuations of this random variable when the training instances, target function and noise values are drawn at random. Using (14), from $q(\alpha)$ we can obtain the complete probability density of Y_m as $m, N \to \infty$ for fixed $m/N = \alpha$. We denote this density by $f(Y_\alpha)$.

We find for the $\lambda = 0$ case that

$$f(Y_\alpha) = \frac{2Y_\alpha}{\gamma(\alpha)} \exp[-\frac{1}{2}t^2(1-\gamma(\alpha)^2)] \tag{24}$$

where t is the solution of $Y_\alpha = H(t\gamma(\alpha))$.

In Figure 5 we show $f(Y_\alpha)$ for 3 values of $q = q(\alpha)$, namely $q = 0.1, 0.3, 0.7$. Here we have dropped the dependence of q on α for convenience.

For small q, f is approximately a Gaussian centered around $Y_\alpha = \frac{1}{2}$:

$$f(Y_\alpha) \approx q^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\left(Y_\alpha - \frac{1}{2}\right)^2 2\pi/q\right]$$

Thus for small q, which means small α when $\lambda = 0$, the volume Z_m is partitioned in two subvolumes of approximately equal size. This means that the learning algorithm (Bayes or Gibbs) does not do significantly better than random guessing.

For all values of $q < \frac{1}{2}$, the density $f(Y_\alpha)$ goes to zero as $Y_\alpha \to 1$. At $q = \frac{1}{2}$, the behavior of the density f changes dramatically, and for all values of $q > \frac{1}{2}$, the density $f(Y_\alpha)$ goes to infinity as $Y_\alpha \to 1$. This means that it is more likely that there is only a small reduction in the volume from Z_m to Z_{m+1}, and thus the learning algorithm has a higher probability of predicting correctly. In fact, for any $\epsilon > 0$, as $q \to 1$ the probability that $Y_m \leq 1 - \epsilon$ converges to zero, giving perfect generalization in this limit.

The largest fluctuations of Y_m appear for $q = \frac{1}{2}$. In this case $\gamma(\alpha) = 1$ and from Equation (24) we simply get

$$f(Y_\alpha) = 2Y_\alpha$$

for $0 \leq Y_\alpha \leq 1$.

We have also estimated the density f for finite m, N by performing numerical simulations on perceptrons. The calculation of Z_m at $\lambda = 0$ for a fixed sample can be easily performed when the prior distribution on the weight vectors $\vec{w}$ is uniform on the Boolean cube $\{\pm 1\}^N$. Using this distribution on the weight vectors, when the instances are distributed uniformly on the sphere, replica symmetry is known to be valid at $\lambda = 0$ for values of α up to ≈ 1.245. Above this value a discontinuous transition to perfect generalization takes place [Gyo90a]. Figures 6 and 7 display results of our simulations for $m = 4, N = 14$ and $m = 16, N = 20$ respectively, averaged over many samples. The smooth curves are the theoretical predictions from Equation (24). The histograms represent the experimental results. The range of Y_α was partitioned into equal width bins. The height plotted for each bin the fraction of times the value of Y_α lands in the bin divided by the width of the bin.

It is also interesting to study the learning curves in the opposite limit $\beta \to 0$ or $\lambda \to \frac{1}{2}$ in more detail. In this case $\delta_{Gibbs}(\alpha, \lambda)$ and $\delta_{Bayes}(\alpha, \lambda)$ have the asymptotic scaling

$$\delta_{Gibbs}(\alpha, \lambda) \approx g_{Gibbs}(\beta^2\alpha)$$

and

$$\delta_{Bayes}(\alpha, \lambda) \approx g_{Bayes}(\beta^2\alpha),$$

where the functions g_{Gibbs} and g_{Bayes} are calculated by expanding (22) for small β.

This high temperature and high noise limit results in a much simpler equation for $q(\alpha)$ when the distribution on instances and prior on weights are both uniform on the sphere:

$$q(\alpha)\sqrt{\frac{1+q(\alpha)}{1-q(\alpha)}} \approx \frac{\alpha\beta^2}{2\pi}.$$

Solving numerically we obtain the results of Figure 3. For $\beta^2\alpha \to \infty$ we find

$$\delta_{Gibbs}(\alpha,\lambda) \approx \frac{4}{\beta^2\alpha} \approx \frac{1}{(1-2\lambda)^2\alpha},$$

which is the dashed line in Figure 3. It follows from this and Equation (1) that

$$\epsilon_{Gibbs}(\alpha,\lambda) \approx \lambda + \frac{2}{\beta\alpha} \approx \lambda + \frac{1}{(1-2\lambda)\alpha}.$$

In [STS90] related results are given for the Gibbs algorithm in a general but somewhat different setting.

3 Conclusion

Our results show that in the limit the relationship between the generalization error of the Bayes and the Gibbs 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 the latter distribution has $\vec{0}$ mean, a central limit theorem holds and the replica symmetry assumptions made above are valid. This also holds for the approximations to Bayes algorithm we have defined using neural nets with a layer of hidden units and a majority gate output unit as well. Our experimental results show the formulas derived in the limit are already fairly accurate for relatively small sample size and instance space dimension, at least for some simple cases. We expect that the quantitative relationships shown in Figure 1 are quite robust, and that in practice, this use of 2-layer nets with majority output may give better generalization performance in many circumstances.

Acknowledgements

Helpful discussions with Haim Sompolinsky, Naftali Tishby and Michael Kearns are greatfully acknowledged. D. Haussler's research was supported by ONR grant N00014-91-J-1162. Part of this work was done while M. Opper was visiting the Physics department of the University of California at Santa Cruz. He would like to thank them for their hospitality.

References

[BH89] E. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[DMW88] Alfredo DeSantis, George Markowski, and Mark N. Wegman. Learning probabilistic prediction functions. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 312–328, San Mateo, CA, 1988. Published by Morgan Kaufmann.

[DSW+87] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield. Automatic learning, rule extraction and generalization. *Complex Syst.*, 1:877–922, 1987.

[Gar88] E. Gardner. The space of interactions in neural networks. *J. Physics A*, 21:257–270, 1988.

[GD88] E. Gardner and B. Derrida. Optimal storage properties of neural network models. *J. Physics A*, 21:271–284, 1988.

[GT90] G. Gyorgyi and N. Tishby. Statistical theory of learning a rule. In K. Thuemann and R. Koeberle, editors, *Neural Networks and Spin Glasses*. World Scientific, 1990.

[Gyo90a] G. Gyorgyi. First order transition to perfect generalization in a neural network with binary synapses. *Phys. Rev. A*, 41:7097, 1990.

[Gyo90b] G. Gyorgyi. Inference of a rule by a neural network with thermal noise. *Phys. Rev. Lett.*, 64:2957, 1990.

[HKS91] D. Haussler, M. Kearns, and R. Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. In *Proceedings of the Fourth Workshop on Computational Learning Theory*, 1991.

[HLW88] D. Haussler, N. Littlestone, and M. Warmuth. Predicting 0,1-functions on randomly drawn points. In *Proceedings of the 29th Annual Symposium on the Foundations of Computer Science*, pages 100–109. IEEE, 1988.

[HS90] D. Hansel and H. Sompolinsky. Learning from examples in a single-layer neural network. *Europhys. Lett.*, 11(7):687–692, 1990.

[KS78] S. Kirkpatrick and D. Sherrington. Infinite ranged models of spin glasses. *Phys. Rev. B*, 17:4384, 1978.

[Lit89] N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, University of Calif., Santa Cruz, 1989.

[LTS89] E. Levin, N. Tishby, and S. Solla. A statistical approach to learning and generalization in neural networks. In R. Rivest, editor, *Proc. 2nd Workshop on Computational Learning Theory*. Morgan Kaufmann, 1989.

[LW89] N. Littlestone and M. Warmuth. The weighted majority algorithm. In *30th Annual IEEE Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[Maa91] Wolfgang Maass. On-line learning with an oblivious environment and the power of randomization. In *Proceedings of the Fourth Workshop on Computational Learning Theory*, 1991.

[Mit82] T. M. Mitchell. Generalization as search. *Art. Intell.*, 18:203–226, 1982.

[MPV87] M. Mezard, G. Parisi, and M.A. Virasoro. *Spin Glass Theory and Beyond*, volume 9 of *Lecture Notes in Physics*. World Scientific, 1987.

[OH91] M. Opper and D. Haussler. Generalization performance of Bayes optimal classification algorithm for learning a perceptron. *Physical Review Letters*, May 1991. to appear.

[OKKN90] M. Opper, W. Kinzel, J. Kleinz, and R. Nehl. On the ability of the optimal percepton to generalize. *J.Phys.A.: Math.Gen.*, 23:L581–L586, 1990.

[STS90] H. Sompolinsky, N. Tishby, and H.S. Seung. Learning from examples in large neural networks. *Phys.Rev.Lett.*, 65:1683–1686, 1990.

[VJP89] F. Vallet, J.Cailton, and P.Refregier. Linear and nonlinear extensions of the pseudo inverse for learning Boolean functions. *Europhys.Lett.*, 9:315–320, 1989.

[Vov90] Volodimir Vovk. Aggregating strategies. In *Proceedings of the 3nd Workshop on Computational Learning Theory*, pages 371–383. published by Morgan Kaufmann, 1990.

Figure captions

Figure 1: Generalization errors of the n-hidden unit network with majority output as a function of $\delta_{Gibbs}(\alpha)$ for $n = 1, 3, 7, 21, \infty$. The case $n = 1$ (diagonal line) corresponds to the Gibbs algorithm itself, and the case $n = \infty$ (lowest curve) corresponds to Bayes optimal algorithm. The results are independent of λ, so this parameter is omitted.

Figure 2: Generalization errors of the n-hidden unit network with majority output at $\lambda = 0$ as a function of α for $n = 1, 3, 7, \infty$, assuming a spherically symmetric distribution on target vectors and instances. The case $n = 1$ (highest solid curve) was recently calculated in [GT90] and corresponds to the Gibbs algorithm. The case $n = \infty$ (lowest solid curve) corresponds to Bayes optimal algorithm.

Figure 3: Learning curves for Gibbs (upper solid line) and Bayes (lower solid line) algorithm in the high temperature limit $\beta \to 0$, assuming a spherically symmetric distribution on target vectors and instances. The dashed line is the asymptotic approximation $g(\beta^2\alpha) \approx \frac{4}{\beta^2\alpha}$ for the Gibbs learning curve.

Figure 4: The function $C(\lambda)(1 - 2\lambda)^2$ defined by Equation (23).

Figure 5: Probability density $f(Y_\alpha)$ for $q(\alpha) = 0.1$ (bell-shaped curve) $q(\alpha) = 0.3$ (flatter curve) and $q(\alpha) = 0.7$ (curve peaked at 1).

Figure 6: Probability density from numerical simulations of perceptrons with weight vectors on Boolean cube, for $m = 4$ and $N = 14$. The smooth curve is the theoretical prediction. See further explaination in text.

Figure 7: Same as Figure 6, but with $m = 16$ and $N = 20$.