On Learning Perceptrons with Binary Weights

Mostefa Golea[*] **Mario Marchand**[†]
Ottawa-Carleton Institute for Physics
University of Ottawa
Ottawa, Ont., Canada K1N 6N5

Submitted to *Neural Computation*, third draft, Dec 22, 1992

Abstract

We present an algorithm that PAC learns any perceptron with binary weights and arbitrary threshold under the family of *product distributions*. The sample complexity of this algorithm is of $O((n/\epsilon)^4 \ln(n/\delta))$ and its running time increases only linearly with the number of training examples. The algorithm does not try to find an hypothesis that agrees with all of the training examples; rather, it constructs a binary perceptron based on various probabilistic estimates obtained from the training examples. We show that, under the restricted case of the *uniform distribution* and zero threshold, the algorithm reduces to the well known *clipped Hebb rule*. We calculate exactly the average generalization rate (*i.e.* the learning curve) of the algorithm, under the uniform distribution, in the limit of an infinite number of dimensions. We find that the error rate decreases *exponentially* as a function of the number of training examples. Hence, the average case analysis gives a sample complexity of $O(n \ln(1/\epsilon))$; a large improvement over the PAC learning analysis. The analytical expression of the learning curve is in excellent agreement with the extensive numerical simulations. In addition, the algorithm is very robust with respect to classification noise.

[*]e-mail: 050287@acadvm1.uottawa.ca
[†]e-mail: mmmsj@acadvm1.uottawa.ca

1 Introduction

The study of neural networks with binary weights is well motivated from both the theoretical and practical points of view. Although the number of possible states in the weight space of a binary network is finite, the capacity of the network is not much inferior to that of its continuous counterpart (Barkai and Kanter 1991). Likewise, the hardware realization of binary networks may prove simpler.

Although networks with binary weights have been the subject of intense analysis from the *capacity* point of view (Barkai and Kanter 1991; Köhler *et al.* 1990; Krauth and Mézard 1989; Venkatesh 1991), the question of the *learnability* of these networks remains largely unanswered. The reason for this state of affairs lies perhaps in the apparent strength of the following distribution-free result (Pitt and Valiant 1988): learning perceptrons with binary weights is equivalent to Integer Programming and so, it is an NP-Complete problem. However, this result does not rule out the possibility that this class of functions is learnable under some *reasonable distributions*.

In this paper, we take a close look at this possibility. In particular, we investigate, within the PAC model (Valiant 1984; Blumer *et al.* 1989), the learnabilty of single perceptrons with binary weights and arbitrary threshold under the family of *product distributions*. A distribution of examples is a product distribution if the the setting of each input variable is independent of the settings of the other variables. The result of this investigation is a polynomial time algorithm that PAC learns binary perceptrons under any product distribution of examples. More specifically, the sample complexity of the algorithm is of $O((n/\epsilon)^4 \ln(n/\delta))$, and its running time is linear in the number of training examples. We note here that the algorithm produces hypotheses that are not necessarily consistent with all the training examples, but that nonetheless have very good generalization ability. These type of algorithms are called "inconsistent algorithms" (Meir and Fontanari 1992).

How does this algorithm relate to the learning rules proposed previously for learning binary perceptrons? We show that, under the *uniform distribution* and for binary perceptrons with zero-threshold, this algorithm reduces to the clipped Hebb rule (Köhler *et al.* 1990) (also known as the majority rule (Venkatesh 1991)).

To understand the typical behavior of the algorithm, we calculate exactly, under the uniform distribution, its average generalization rate (*i.e.* the learning curve) in the limit of an infinite number of input variables. We find that, on average, the generalization rate converges *exponentially* to 1 as a function of the number of training examples. The sample complexity in the average case is of $O(n \ln(1/\epsilon))$; a large improvement over the PAC learning analysis. We calculate also the average generalization rate when learning from noisy examples and show that the algorithm is very robust with respect to classification noise. The results of the extensive simulations are in very good agreement with the theoretical ones.

2 Definitions

Let I denote the set $\{-1, +1\}$. A *perceptron* g on the *instance space* I^n is specified by a vector of n *weight* values w_i and a single *threshold* value θ. For an input vector $\mathbf{x} = (x_1, x_2, ..., x_n) \in I^n$, we have:

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{i=n} w_i x_i > \theta \\ -1 & \text{if } \sum_{i=1}^{i=n} w_i x_i \leq \theta \end{cases} \tag{1}$$

A perceptron is said to be *positive* if $w_i \geq 0$ for $i = 1, ..., n$.

We are interested in the case where the weights are binary valued (± 1). We assume, without loss of generality (w.l.o.g.), that θ is integer and $-n - 1 \leq \theta \leq n$.

An *example* is an input-output pair $< \mathbf{x}, g(\mathbf{x}) >$. A *sample* is a set of examples. The examples are assumed to be generated randomly according to some unknown probability distribution D which can be any member of the family $\mathcal{D}$ of all *product distributions*. Distribution D belongs to $\mathcal{D}$ if and only if the setting of each input variable x_i is chosen *independently* of the settings of the other variables. The *uniform distribution*, where each x_i is set independently to ± 1 with probability $1/2$, is a member of $\mathcal{D}$.

We denote by $P(A)$ the probability of event A and by $\hat{P}(A)$ its empirical estimate based on a given finite sample. All probabilities are taken with respect to the product distribution D on I^n. We denote by $E(x)$ and $Var(x)$ the expectation and variance of the random variable x.

If a, $b \in \{-1, +1\}$, we denote by $P(g = b | x_i = a)$ the conditional probability that $g = b$ given the fact that $x_i = a$.

The *influence* of a variable x_i, denoted $Inf(x_i)$, is defined as

$$\begin{aligned} Inf(x_i) &= P(g = +1 | x_i = +1) - P(g = +1 | x_i = -1) \\ &- P(g = -1 | x_i = +1) + P(g = -1 | x_i = -1) \end{aligned} \tag{2}$$

Intuitively, the influence of a variable is positive (negative) if its weight is positive (negative).

3 PAC Learning Single Binary Perceptrons

3.1 The Learning Model

In this section, we adopt the PAC learning model (Valiant 1984; Blumer *et al.* 1989). Here the methodology is to draw, according to D, a sample of a certain size labeled according to an unknown target perceptron g and then to find a "good" approximation g' of g. The error of the hypothesis perceptron g', with respect to the target g, is defined to be $P(g' \neq g) = P(g'(\mathbf{x}) \neq g(\mathbf{x}))$, where $\mathbf{x}$ is drawn according to the *same* distribution D used to generate the training sample.

An algorithm PAC learns from examples the class G of binary perceptrons, under a family $\mathcal{D}$ of distributions on I^n, if for every $g \in G$, any $D \in \mathcal{D}$ and any $0 < \epsilon, \delta < 1$, the algorithm runs in time polynomial in (n, ϵ, δ) and outputs, with probability at least $1 - \delta$, an hypothesis $g' \in G$ that makes an error at most ϵ with g.

3.2 The Learning Algorithm

We assume that the examples are generated according to a (unknown) product distribution D on $\{-1, +1\}^n$ and labeled according to a target binary perceptron g given by eq.(1). The learning algorithm proceeds in three steps:

1. Estimating, for each input variable x_i, the probability that it is set to $+1$. If this probability is too high (too low), the variable is set to $+1$ (-1). Note that setting a variable to a given value is equivalent to neglecting this variable because any constant can be absorbed in the threshold.

2. Estimating the weight values (signs). This is done by estimating the *influence* of each variable.

3. Estimating the threshold value.

To simplify the analysis, we introduce the following notation. Let $\mathbf{y}$ be the vector whose components y_i are defined as

$$y_i = w_i \times x_i \tag{3}$$

Then eq.(1) can be written as

$$g(\mathbf{y}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{i=n} y_i > \theta \\ -1 & \text{if } \sum_{i=1}^{i=n} y_i \leq \theta \end{cases} \tag{4}$$

In addition, we define $Inf(y_i)$ by:

$$\begin{aligned} Inf(y_i) &= P(g=+1|y_i=+1) - P(g=+1|y_i=-1) \\ &- P(g=-1|y_i=+1) + P(g=-1|y_i=-1) \end{aligned} \tag{5}$$

Note that if $D(\mathbf{x})$ is a product distribution on $\{-1,+1\}^n$, then so is $D(\mathbf{y})$.

Lemma 1 *Let g be a binary perceptron. Let x_i be a variable in g. Let $a \in \{-1,+1\}$. Let g' be a perceptron obtained from g by setting x_i to a. Then, if $P(x_i = -a) \leq \frac{\epsilon}{2n}$,*

$$P(g \neq g') \leq \frac{\epsilon}{2n}$$

Proof: follows directly from the fact that $P(g \neq g') \leq P(x_i = -a)$.□

Lemma 1 implies that we can neglect any variable x_i for which $P(x_i = \pm 1)$ is too high (too low). In what follows, we consider only variables that have not been neglected.

As we said earlier, intuition suggests that the influence of a variable is positive (negative) if its weight is positive (negative). The following lemma strengthens this intuition by showing that there is a measurable *gap* between the two cases. This gap will be used to estimate the weight values (signs).

Lemma 2 *Let g be a perceptron such that $P(g=+1)$, $P(g=-1) > \rho$, where $0 < \rho < 1$. Then for any product distribution D,*

$$Inf(x_i) \begin{cases} > \frac{\rho}{n+1} & \text{if } w_i = +1 \\ < -\frac{\rho}{n+1} & \text{if } w_i = -1 \end{cases}$$

Proof: We first note that from the definition of the influence and eq. 3 and 5, we can write:

$$Inf(x_i) = \begin{cases} +Inf(y_i) & \text{if } w_i = +1 \\ -Inf(y_i) & \text{if } w_i = -1 \end{cases}$$

We exploit the independence of the input variables to write

$$\begin{aligned} Inf(y_i) &= P(\sum_{j\neq i} y_j + 1 > \theta) - P(\sum_{j\neq i} y_j - 1 > \theta) \\ &\quad -P(\sum_{j\neq i} y_j + 1 \leq \theta) + P(\sum_{j\neq i} y_j - 1 \leq \theta) \\ &= 2P(\sum_{j\neq i} y_j = \theta) + 2P(\sum_{j\neq i} y_j = \theta + 1) \end{aligned} \tag{6}$$

One can also write

$$\begin{aligned} P(g=+1) &= P(\sum_j y_j > \theta) \\ &= P(y_i=+1) \times P(\sum_{j\neq i} y_j + 1 > \theta) + P(y_i=-1) \times P(\sum_{j\neq i} y_j - 1 > \theta) \\ &\leq P(\sum_{j\neq i} y_j + 1 > \theta) = \sum_{r=\theta}^{n} P(\sum_{j\neq i} y_j = r) \end{aligned} \tag{7}$$

Likewise,

$$\begin{aligned} P(g=-1) &= P(\sum_j y_j \leq \theta) \\ &= P(y_i=1) \times P(\sum_{j\neq i} y_j + 1 \leq \theta) + P(y_i=-1) \times P(\sum_{j\neq i} y_j - 1 \leq \theta) \\ &\leq P(\sum_{j\neq i} y_j - 1 \leq \theta) = \sum_{r=-n}^{r=\theta+1} P(\sum_{j\neq i} y_j = r) \end{aligned} \tag{8}$$

Let $p(r)$ denote $P(\sum_{j\neq i} y_j = r)$. From the properties of the generating function associated with product distributions, it is well known (Ibragimov 1956; MacDonald 1979) that $p(r)$ is always *unimodal* and reaches its maximum at a given value of r, say r_{max}. We distinguish two cases:
$\underline{\theta \geq r_{max}}$: in this case, using eq (7)

$$\begin{aligned} P(g=+1) &\leq \sum_{r=\theta}^{n} P(\sum_{j\neq i} y_j = r) \\ &\leq (n-\theta+1) \times p(\theta) \end{aligned} \tag{9}$$

Using eq. (6) and eq. (9), it easy to see that

$$Inf(y_i) \geq \frac{2P(g=1)}{n-\theta+1} > \frac{\rho}{n+1}$$

$\underline{\theta \leq r_{max}-1}$: in this case, using eq (8)

$$\begin{aligned} P(g=-1) &\leq \sum_{r=-n}^{r=\theta+1} P(\sum_{j\neq i} y_j = r) \\ &\leq (n+\theta+2) \times p(\theta+1) \end{aligned} \tag{10}$$

Using eq. (6) and eq. (10), it easy to see that

$$Inf(y_i) \geq \frac{2P(g=-1)}{n+\theta+2} > \frac{\rho}{n+1}$$

□

So, if we estimate $Inf(x_i)$ to within a precision better than the gap established in lemma 2, we can determine the value of w_i with enough confidence. Note that if θ is too large (small), most of the examples will be negative (positive). In this case, the influence of any input variable is very weak. This is the reason we require $P(g=+1)$, $P(g=-1) > \rho$.

The weight values obtained in the previous step define the weight vector of our hypothesis perceptron g'. The next step is to estimate an appropriate threshold for g', using these weight values. For that, we appeal to the following lemma.

Lemma 3 *Let g be a perceptron with a threshold θ. Let g' be a perceptron obtained from g by substituting r for θ. Then, if $r \leq \theta$,*

$$P(g \neq g') \leq 1 - P(g=+1|g'=+1)$$

Proof:

$$\begin{aligned} P(g \neq g') &\leq P(g = -1|g' = +1) + P(g = +1|g' = -1) \\ &= 1 - P(g = +1|g' = +1) + P(g = +1|g' = -1) \\ &= 1 - P(g = +1|g' = +1) \end{aligned}$$

The last equality follows from the fact that $P(g = +1|g' = -1) = 0$ for $r \leq \theta$.□

So, if we estimate $P(g = +1|g' = +1)$ for $r = -n-1, -n, -n+1, ...$ and then choose as a threshold for g' the *least* r for which $P(g = +1|g' = +1) \geq (1 - \epsilon)$, we are guaranteed to have $P(g \neq g') \leq \epsilon$. Obviously, such an r exists and is always $\leq \theta$ because $P(g = +1|g' = +1) = 1$ for $r = \theta$.

A sketch of the algorithm for learning single binary perceptrons is given in fig. 1.

Theorem 1 *The class of binary perceptrons is PAC learnable under the family of product distributions.*

Proof: Using Chernoff bounds (Hagerup and Rub 1989), one can show that a sample of size $m = \frac{[160n(n+1)]^2}{\epsilon^4} \ln \frac{32n}{\delta}$ is sufficient to ensure that:

- $|\hat{P}(g = a) - P(g = a)| \leq \epsilon/4$ with confidence at least $1 - \delta/2$.
- $|\hat{P}(x_i = a) - P(x_i = a)| \leq \epsilon/4n$ with confidence at least $1 - \delta/4n$.
- $|\hat{Inf}(x_i) - Inf(x_i)| \leq \frac{\epsilon}{4(n+1)}$ with confidence at least $1 - \delta/8n$.
- $|\hat{P}(g = +1|g' = +1) - P(g = +1|g' = +1)| \leq \epsilon/4$ with confidence at least $1 - \delta/16n$.

Combining all these factors, it is easy to show that the hypothesis g' returned by the algorithm will make an error at most ϵ with the target g, with confidence at least $1 - \delta$.

Since it takes m units of time to estimate a conditional probability using a sample of size m, the running time of the algorithm will be of $O(m \times n)$.□

4 Reduction to the Clipped Hebb Rule

The perceptron with binary weights and zero-threshold has been extensively studied by many authors (Krauth and Mézard 1989; Köhler *et al.* 1990; Opper *et al.* 1990; Venkatesh 1991). All these studies assume a uniform distribution of examples. So, we come to ask how the algorithm of fig. 1 relates to the learning rules proposed previously.

To answer this, let us first rewrite the influence of a variable as:

$$Inf(x_i) = \frac{P(x_i = +1|g = +1)}{P(x_i = +1)} - \frac{P(x_i = -1|g = +1)}{P(x_i = +1)} + \frac{P(x_i = -1|g = -1)}{P(x_i = -1)} - \frac{P(x_i = +1|g = -1)}{P(x_i = +1)}$$

and observe that under the uniform distribution, $P(x_i = +1) = P(x_i = -1)$. Next, we notice that in the algorithm of fig. 1, each weight w_i is basically assigned to the sign of $\hat{Inf}(x_i)$. Hence apart from ϵ and δ, the algorithm can be summarized by the following rule:

$$\begin{aligned} w_i &= \text{sgn}\left(\hat{Inf}(x_i)\right) \\ &= \text{sgn}\left(\sum_\nu g(\mathbf{x}^\nu) x_i^\nu\right) \end{aligned} \tag{11}$$

where $\mathrm{sgn}(x) = +1$ when $x > 0$ and -1 otherwise and x_i^ν denotes the ith component of the νth training example.

Equation 11 is simply the well known *clipped Hebb rule* (Opper *et al.* 1990), also called the *majority rule* in (Venkatesh 1991). Since this rule is just the restriction of the learning algorithm of fig. 1 for uniform distributions, theorem 1 has the following corollary:

Corollary 1 *The clipped Hebb rule PAC learns the class of binary perceptrons with zero thresholds under the uniform distribution.*

5 Average Case Behavior in the Limit of Infinite n

The bound on the number of examples needed by the algorithm of fig. 1 to achieve a given accuracy with a given confidence is overly pessimistic. In our approach, this overestimate can be traced to the inequalities present in the proofs of lemma 2 and 3 and to the use of the Chernoff bounds (Hagerup and Rub 1989). To obtain the typical behavior of the algorithm we will calculate analytically, for any target perceptron, the average *generalization rate* (*i.e.* the learning curve). By generalization rate we mean the curve of the generalization ability as a function of the size of the training set m. The central limit theorem will tell us that the average behavior becomes the typical behavior in the limit of infinite n and infinite m with $\alpha = m/n$ kept constant.

As it is generally the case (Vallet 1989; Opper *et al.* 1990; Opper and Haussler 1991), we limit ourselves, for the sake of mathematical simplicity, to the case of uniform distribution and zero threshold. Therefore, we will calculate the average generalization rate of the clipped Hebb rule (hereafter CHR) (eq. 11) for both noise-free and noisy examples.

5.1 Zero Noise

Let $\mathbf{w}^t = (w_1^t, w_2^t, \cdots\ w_n^t)$ be the target weight vector and let $\mathbf{w} = (w_1, w_2, \cdots\ w_n)$ be the hypothesis weight vector constructed by the CHR with m training examples. The generalization rate G is defined to be the probability that the hypothesis agrees with the target on a random example $\mathbf{x}$ chosen according to the uniform distribution. Let us start by defining the following sums of random variables:

$$X = \sum_{i=1}^{n} w_i x_i \tag{12}$$

$$Y = \sum_{i=1}^{n} w_i^t x_i \quad , \tag{13}$$

The generalization rate is given by

$$G = P\left[\mathrm{sgn}(X) = \mathrm{sgn}(Y)\right] \tag{14}$$

$$= P\left[XY > 0\right] \tag{15}$$

where we have assumed w.l.o.g. that n is an odd number. Since $\mathbf{x}$ is distributed uniformly, we easily find that:

$$E(X) = E(Y) = 0 \tag{16}$$

$$Var(X) = Var(Y) = n \tag{17}$$

$$n \times \rho = E(XY) = \sum_{i=1}^{n} w_i w_i^t \tag{18}$$

where $-1 \leq \rho \leq +1$ is defined to be the normalized *overlap* between the target and the hypothesis weight vector.

According to the central limit theorem, in the limit $n \to \infty$, X and Y will be distributed according to a bivariate normal distribution with moments given by eq. 16, 17 and 18. Hence, for fixed $\mathbf{w}^t$ and $\mathbf{w}$, the generalization rate G is given by:

$$G = 2 \int_0^\infty dx \int_0^\infty dy \; p(x, y)$$

where the joint probability distribution $p(x, y)$ is given by

$$p(x, y) = \frac{1}{2\pi n \sqrt{(1 - \rho^2)}} \times \exp\left[-\frac{x^2}{2n} - \frac{(y - \rho x)^2}{2n(1 - \rho^2)} \right]$$

This integral easily evaluates to give:

$$G(\rho) = 1 - \frac{1}{\pi} \arccos \rho \tag{19}$$

So, as $n \to \infty$, the generalization rate depends only on the angle between the target and the hypothesis weight vectors.

Now, to average this result over the all training samples of size m, we argue that for large n, the distribution of the random variable ρ becomes sharply peaked at its mean. Denoting the average over the training samples by $<< \ldots >>$, this amounts to approximating $<< G(\rho) >>$ by $G(<< \rho >>)$ as $n \to \infty$.

Using eq. 18, we can write (for a fixed $\mathbf{w}^t$):

$$<< \rho >> \;=\; \frac{1}{n} \sum_{i=1}^{n} w_i^t \, << w_i >> \tag{20}$$

$$=\; \frac{1}{n} \sum_{i=1}^{n} (2p_i - 1) \tag{21}$$

where p_i is the probability that $w_i^t w_i = +1$. We introduce the independent random variables $\xi_i^\nu = w_i^t x_i^\nu$, and use eq. 11 to write:

$$w_i^t w_i = \operatorname{sgn}\left[\sum_{\nu=1}^{m} \operatorname{sgn}\left(\xi_i^\nu \sum_{j=1}^{n} \xi_j^\nu \right) \right] \tag{22}$$

Let us define the new random variables η_i^ν

$$\eta_i^\nu = \operatorname{sgn}\left(\xi_i^\nu \sum_{j=1}^{n} \xi_j^\nu \right) \tag{23}$$

With that, p_i can be written as

$$p_i = P\left[\left(\sum_{\nu=1}^{m} \eta_i^\nu \right) > 0 \right] \tag{24}$$

Let q be the probability that $\eta_i^\nu = +1$. From eq. 23, we can write q as:

$$q \;=\; P\left[\left(\sum_{j \neq i}^{n} \xi_i^\nu \xi_j^\nu \right) > -1 \right]$$

$$
\begin{aligned}
&= \sum_{k=(n-1)/2}^{n-1} \left(\begin{array}{c} n-1 \\ k \end{array} \right) \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-1-k} \\
&= \frac{1}{2} + \frac{1}{2^n} \left(\begin{array}{c} n-1 \\ \frac{n-1}{2} \end{array} \right) \\
&= \frac{1}{2} + \frac{1}{\sqrt{2\pi n}} \text{ as } n \to \infty \qquad (25)
\end{aligned}
$$

where, by using Stirling's formula, we have kept only the leading term in $1/\sqrt{n}$ as $n \to \infty$. Hence, in this limit, each η_i^ν has unit variance and a mean of $2/\sqrt{2\pi n}$. Since η_i^ν and $\eta_i^{\mu \neq \nu}$ are statistically independent, the central limit theorem tells us that, when $m \to \infty$, the variable

$$
Z = \sum_{\nu=1}^{m} \eta_i^\nu
$$

becomes a Gaussian variable with mean $\mu_z = m \times \sqrt{2/\pi n}$ and variance m. Hence, as $m \to \infty$ and $\alpha = m/n$ is kept constant, eq. 24 becomes:

$$
\begin{aligned}
p_i &= \frac{1}{\sqrt{2\pi m}} \int_0^\infty dz \ \exp\left[-\frac{(z-\mu_z)^2}{2m} \right] \qquad (26) \\
&= \frac{1}{2} + \frac{1}{2} \ \text{erf}\left(\sqrt{\alpha/\pi} \right) \qquad (27)
\end{aligned}
$$

Hence, using eq. 19, 21 and 27, we have finally:

$$
\begin{aligned}
<< \rho >> &= \text{erf}\left(\sqrt{\alpha/\pi} \right) \qquad (28) \\
<< G >> &= 1 - \frac{1}{\pi} \arccos\left[\text{erf}\left(\sqrt{\alpha/\pi} \right) \right] \qquad (29)
\end{aligned}
$$

This results is independent of the target $\mathbf{w}^t$.

The average generalization rate and normalized overlap are plotted in fig. 2 and compared with numerical simulations. We see that the agreement with the theory is excellent, even for moderate values of n. Notice that the agreement is slightly better for $<< \rho >>$ than it is for $<< G >>$. This illustrates the difference between $<< G(\rho) >>$ and $G(<< \rho >>)$.

To compare this average-case analytic result to the bounds given by PAC learning, we use the fact that we can bound $\text{erf}(z)$ by an exponential (Abramowitz and Stegun 1972) and thus bound the error rate $1 - << G >> = \epsilon_{rr}$ by:

$$
\epsilon_{rr} < \ \exp\left(-\frac{\alpha}{2\pi} \right) \qquad (30)
$$

That is, the error rate decreases *exponentially* with the number of examples and, on average, a training set of size of $O(n \ln(1/\epsilon))$ is sufficient to produce an hypothesis with error rate ϵ. This is an important improvement over the bound of $O((n/\epsilon)^4 \ln(n/\delta))$ given by our PAC learning analysis.

Thus, the CHR is a striking example of a very simple "inconsistent" algorithm that does not always produce hypotheses that agrees with all the training examples, but nonetheless produce hypotheses with outstanding generalization ability. Moreover, the exponential convergence outlines the computational advantage of learning binary perceptrons using binary perceptrons. In fact, if one allows real weights, no algorithm can outperform the Bayes optimal algorithm (Opper and Haussler 1991). The latter's error rate improves only *algebraically*, approximately as $0.44/\alpha$.

On the other hand, for consistent learning rules that produce perceptrons with binary weights, a phase transition to perfect generalization is known to take place at a critical value of α (Sompolinsky *et al.* 1990; Gyorgyi 1990). Thus, these rules have a slightly better sample complexity than the CHR. Unfortunately, they are much more computationally expensive (with a running time that generally increases exponentially with the number of inputs n). Since it is an "inconsistent" learning rule, the CHR does not exhibit a phase transition to perfect generalization. We think that the exponential convergence is the reminiscence of the "lost" phase transition.

An interesting question is how the CHR behaves when learning binary perceptrons on product distributions. To answer this, we first note that the CHR works by exploiting the *correlation* between the state of each input variable x_i and the classification label (eq. 11). Under the uniform distribution, this correlation is positive if $w_i^t = +1$ and negative if $w_i^t = -1$. This is no more true for product distributions: one can easily craft some malicious product distributions where, for example, this correlation is negative although $w_i^t = +1$. The CHR will be fooled by such distributions because it does no take into account the fact that the settings of the input variables do not occur with the same probability. The algorithm of fig. 1 fix this problem by taking this fact into consideration, through the conditional probabilities.

Finally, it is important to mention that binary perceptrons trained with the CHR on examples generated uniformly will perform well even when tested on examples generated by non-uniform distributions, as long as these distributions are *reasonable* (for a precise definition of reasonable distributions, see (Bartlett and Williamson 1990)).

5.2 Classification Noise

In this section we are interested in the generalization rate when learning from noisy examples. We assume that the classification label of each training example is flipped independently with some probability σ. Since the object of the learning algorithm is to construct an hypothesis $\mathbf{w}$ that agrees the most with the underlying target $\mathbf{w}^t$, the generalization rate G is defined to be the probability that the hypothesis agrees with the *noise-free* target on a new random example $\mathbf{x}$.

The generalization rate for a fixed $\mathbf{w}$ and $\mathbf{w}^t$ is still given by eq. 19. To calculate the effect of noise on $<< \rho >>$, let us define q' as the probability that $\eta_i^\nu = +1$ in the presence of noise whereas q denotes this probability in the noise-free regime (*i.e.* eq. 25). These two probabilities are related by:

$$q' = q(1-\sigma) + (1-q)\sigma \tag{31}$$

$$= \frac{1}{2} + \frac{1-2\sigma}{\sqrt{2\pi n}} \text{ as } n \to \infty \tag{32}$$

where we have used eq. 25 for the last equality. This leads to the following expressions for the normalized overlap and the generalization rate, in the presence of noise:

$$<< \rho >> = \text{erf}\left((1-2\sigma)\sqrt{\alpha/\pi}\right) \tag{33}$$

$$<< G >> = 1 - \frac{1}{\pi}\arccos\left[\text{erf}\left((1-2\sigma)\sqrt{\alpha/\pi}\right)\right] \tag{34}$$

One can see that the algorithm is very robust with respect to classification noise: the average generalization rate still converges exponentially to 1 as long as $\sigma < 1/2$. The only difference with the noise-free regime is the presence of the prefactor $(1-2\sigma)$.

The average generalization rate for different noise levels σ is plotted in fig. 3. We see that the numerical simulations are in excellent agreement with the theoretical curves.

6 Summary

We have proposed a very simple algorithm that PAC learns the class of perceptrons with binary weights and arbitrary threshold under the family of product distributions. The sample complexity of this algorithm is of $O((n/\epsilon)^4 \ln(n/\delta))$ and its running time increases only linearly with the sample size.

We have shown that this algorithm reduces to the clipped Hebb rule when learning binary perceptrons with zero threshold under the uniform distribution. We have calculated exactly its learning curve in the limit $n \to \infty$ where the average behavior becomes the typical behavior. We have found that the error rate converges exponentially to zero and have thus improved the sample complexity to $O(n \ln(1/\epsilon))$. The analytic expression of the learning curve is in excellent agreement with the numerical simulations. The algorithm is very robust with respect to random classification noise.

Acknowledgments: This work was supported by NSERC grant OGP0122405.

References

[1] Abramowitz M. and Stegun I. A., "Handbook of Mathematical Functions", Dover Publ., 1972, eq. 7.1.13.

[2] Barkai, E. & Kanter, I., "Storage Capacity of a Multilayer Neural Network with Binary weights", *Europhys. Lett.*, Vol. 14, 1991, 107–112.

[3] Bartlett, P. L. and Williamson, R. C., "Investigating the Distribution Assumptions in the PAC Learning Model", in *Proc. of the 4th Workshop on Computational Learning Theory*, Morgan Kaufman, 1991, 24–32.

[4] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, K., "Learnability and the Vapnik-Chervonenkis Dimension", *J. ACM*, Vol. 36, (1989), 929–965.

[5] Gyorgyi G., "First-order transition to perfect generalization in a neural network with binary synapses", *Phys. Rev. A*, Vol. 41, (1990), 7097–7100.

[6] Hagerup, T., & Rub, C., "A Guided Tour to Chernoff Bounds", *Info. Proc. Lett.*, Vol. 33, (1989), 305–308.

[7] Ibragimov I. A., "On the composition of unimodal distributions", *Theory of Probability and its applications*, Vol. 1, (1956) 255–260.

[8] Köhler H.,Diederich S., Kinzel W., and Opper M., "Learning Algorithm for a Neural Network with Binary Synapses", *Z. Phys. B*, Vol. 78, (1990), 333–342.

[9] Krauth W., Mézard M., "Storage Capacity of Mémory Networks with Binary Couplings", *J. Phys. France*, Vol. 50, (1989), 3057–3066.

[10] MacDonald D. R., "On local limit theorems for integer-valued random variables", *Theory of Probability and Statistics Acad. Nauk.*, Vol. 3 (1979), 607–614.

[11] Meir R., Fontanari J. F., "Calculation of learning curves for inconsistent algorithms", *Phys. Rev. A*, Vol. 92, (1992), 8874–8884.

[12] Opper M,, Haussler H., "Generalization Performance of Bayes Optimal Classification Algorithm for Learning a Perceptron", *Phys. Rev. Lett.* Vol. 66, (1991), 2677–2680.

[13] Opper M., Kinzel W., Kleinz J., Nehl R., "On the Ability of the Optimal Perceptron to Generalize", *J. Phys. A: Math. Gen.*, Vol. 23, (1990), L581–L586.

[14] Pitt, L. & Valiant, L.G., "Computational Limitations on Learning from Examples", *J. ACM*, Vol. 35, 1988, 965–984.

[15] Sompolinsky H., Tishby N, Seung H. S., "Learning from Examples in Large Neural Networks", *Phys. Rev. Lett*, Vol. 65, (1990), 1683–1686.

[16] Vallet, F., "The Hebb Rule for Learning Linearly Separable Boolean Functions: Learning and Generalization.". *Europhys. Lett*, Vol. 8, (1989), 747–751.

[17] Valiant, L.G., "A Theory of the Learnable", *Comm. ACM*, Vol. 27, 1984, 1134–1142.

[18] Venkatesh, S., "On Learning Binary Weights for Majority Functions", in *Proc. of the 4th Workshop on Computational Learning Theory*, Morgan Kaufman, 1991, 257–266.

Algorithm *LEARN-BINARY-PERCEPTRON*(n,ϵ,δ)
Parameters: n is the number of input variables, ϵ is the accuracy parameter and δ is the confidence parameter.
Output: a binary perceptron g' defined by a weight vector $(w_1, \ldots, w_n)$ and a threshold r.
Description:

1. Call $m = \frac{[160n(n+1)]^2}{\epsilon^4} \ln \frac{32n}{\delta}$ examples. This sample will be used to estimate the different probabilities. Initialize g' to the constant perceptron -1.
2. (Are most examples positive?) If $\hat{P}(g = +1) \geq (1 - \frac{\epsilon}{4})$ then set $g' = 1$ and return g'.
3. (Are most examples negative?) If $\hat{P}(g = +1) \leq \frac{\epsilon}{4}$ then set $g' = -1$ and return g'.
4. Set $\rho = \frac{\epsilon}{2}$.
5. (Is $P(x_i = +1)$ too low or too high ?) For each input variable x_i:
   (a) Estimate $P(x_i = +1)$.
   (b) If $\hat{P}(x_i = +1) \leq \frac{\epsilon}{4n}$ or $1 - \hat{P}(x_i = +1) \leq \frac{\epsilon}{4n}$, neglect this variable.
6. (Determine the weight values) For each input variable x_i:
   (a) If $\hat{Inf}(x_i) > \frac{1}{2}\frac{\rho}{n+1}$, set $w_i = 1$.
   (b) Else if $\hat{Inf}(x_i) < -\frac{1}{2}\frac{\rho}{n+1}$, set $w_i = -1$.
   (c) Else set $w_i = 0$ (x_i is not an influential variable).
7. (Estimating the threshold) Initialize r (the threshold of g') to $-(n + 1)$.
   (a) Estimate $P(g = +1 | g' = +1)$.
   (b) If $\hat{P}(g = +1 | g' = +1) > 1 - \frac{1}{4}\epsilon$, go to step 8.
   (c) $r = r + 1$. Go to step 7a.
8. Return g' (that is $(w_1, ..., w_n; r)$).

Figure 1: An algorithm for learning single binary perceptrons on product distributions.

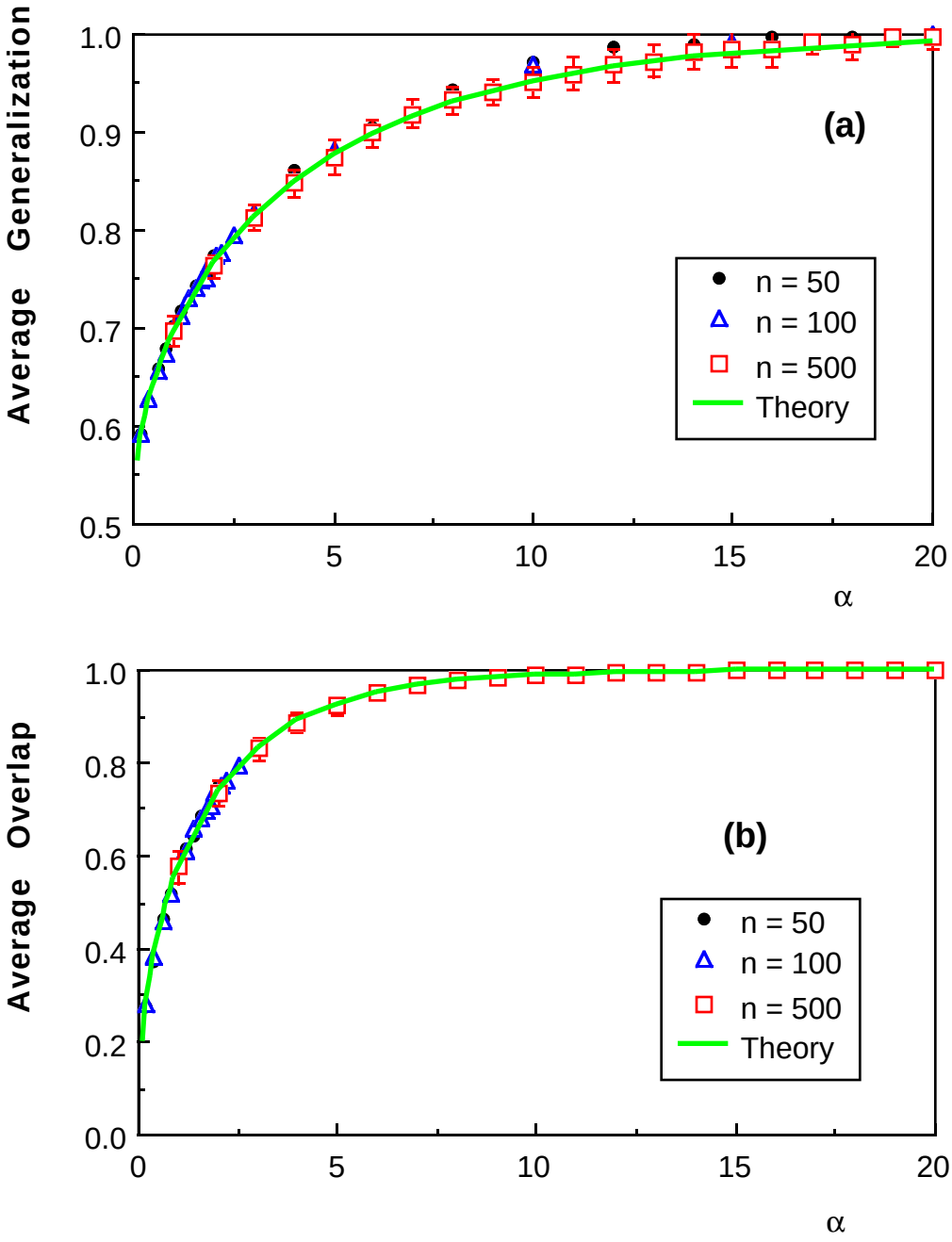

Figure 2: (a) The average generalization rate $<< G >>$ and (b) the average normalized overlap $<< \rho >>$ as a function of normalized number of examples $\alpha = m/n$. Numerical results are shown for $n = 50, 100, 500$. Each point denotes an average over 50 different training samples and the error bars denote the standard deviations.

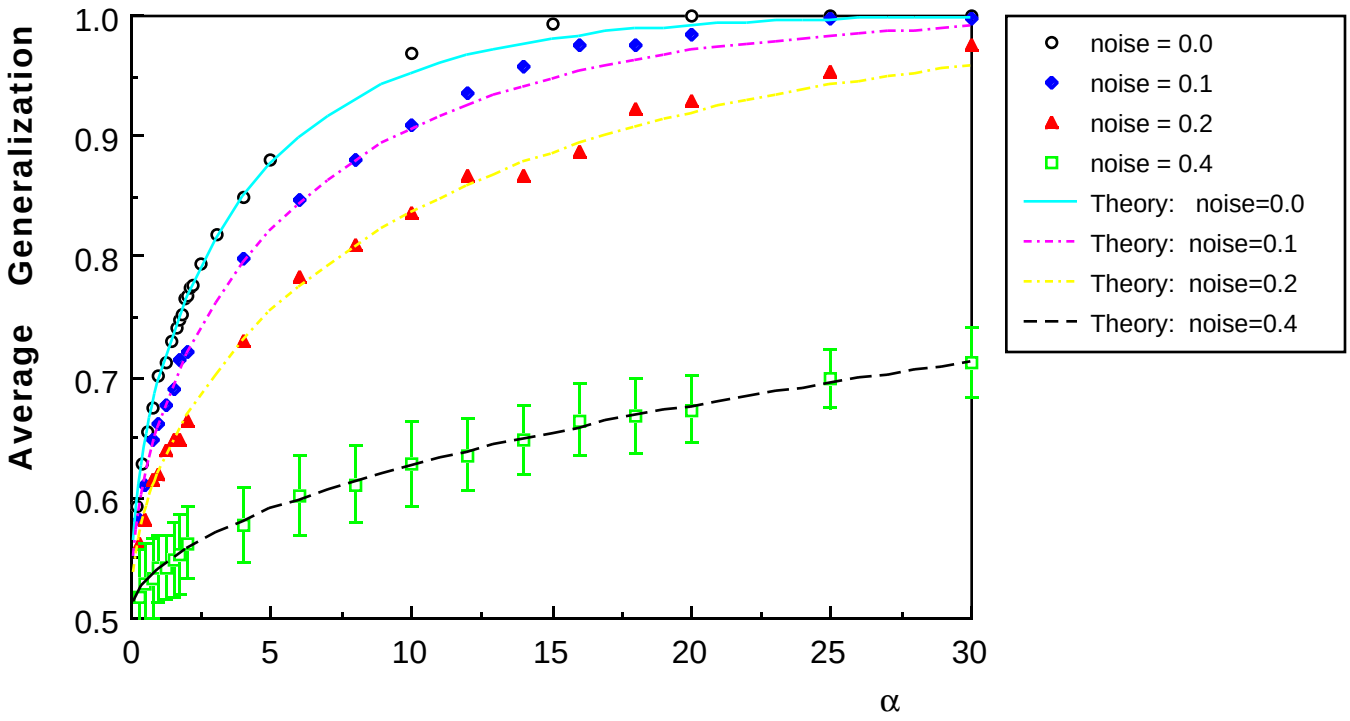

Figure 3: The average generalization rate $<< G >>$ for different noise levels σ. Numerical results are shown for $n = 100$. Each point denotes the average over 50 different simulations (*i.e.* 50 different noisy training sets). The error bars (indicated only for $\sigma = 0.4$ for clarity) denote the standard deviations.