Average Case Analysis of the Clipped Hebb Rule for Nonoverlapping Perceptron Networks

Mostefa Golea
Ottawa-Carleton Institute for Physics
University of Ottawa
Ottawa, Ont., Canada K1N 6N5
golea@physics.uottawa.ca

Mario Marchand
Ottawa-Carleton Institute for Physics
University of Ottawa
Ottawa, Ont., Canada K1N 6N5
mmmsj@acadvm1.uottawa.ca

Abstract

We investigate the clipped Hebb rule for learning different multilayer networks of nonoverlapping perceptrons with binary weights and zero thresholds when the examples are generated according to the uniform distribution. Using the central limit theorem and very simple counting arguments, we calculate exactly its learning curves (*i.e.* the generalization rates as a function of the number of training examples) in the limit of a large number of inputs. We find that the learning curves converge *exponentially* rapidly to perfect generalization. These results are very encouraging given the simplicity of the learning rule. The analytic expressions of the learning curves are in excellent agreement with the numerical simulations, even for moderate values of the number of inputs.

From: Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, pp. 151–157, Santa-Cruz CA, July 26–28, 1993. ACM Press.

1 Introduction

Neural networks with binary weights have attracted much attention recently [1, 4, 6, 7, 13]. This was motivated by both theoretical and practical reasons. First, because the number of possible states in the weight space of a binary network is finite, its properties may differ drastically from these of a network with continuous weights [4, 12]. Second, the hardware realization of binary networks may prove simpler.

The *generalization* ability of neural networks with binary weights has been studied extensively using the *statistical mechanics approach* [4, 7, 12]. Although this approach has yielded some impressive results, it has its shortcomings. In particular, it neglects the *computational* aspect of the learning process by assuming a stochastic training algorithm, similar to a finite Monté Carlo process, that leads at long times to a Gibbs distribution [12]. Unfortunately, stochastic training algorithms generally require prohibitively long convergence times. So, despite this intensive study, the fundamental question of whether or not there exist efficient algorithms for learning this class of networks remains largely unanswered. Since learning even single binary perceptrons is intractable in the distribution free sense [10], we are lead to consider the existence of learning algorithms that work well under some *reasonable distributions* of examples.

Perhaps the simplest algorithm that we can imagine for learning binary networks is the *clipped Hebb rule* [6] (also called the majority rule in [13]). This rule is local, homogeneous, easy to implement, and simple enough to be biologically plausible. Moreover, it observes each training example only once. Recently, we investigated the average case behavior of this rule when learning *single* perceptrons with binary weights under the uniform distribution [2]. We found that its *learning curve*, *i.e.*, the average generalization as a function of the size of the training set, converged *exponentially* to perfect generalization. These finding were confirmed by extensive simulations.

In this paper, we take this investigation one step further and look at the average behavior of the clipped Hebb rule when learning *networks* of *nonoverlapping* binary

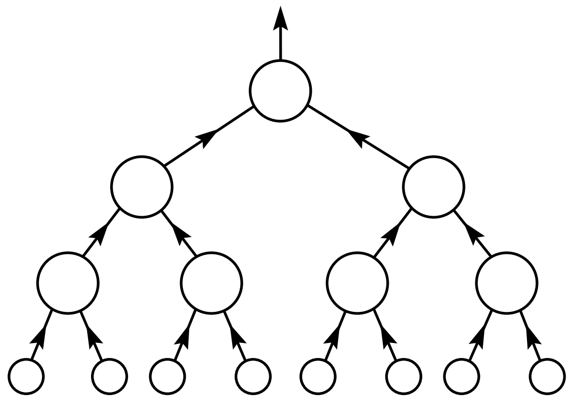

Figure 1: A multilayer network of nonoverlapping binary perceptrons. Note that each node has one and only one outgoing connection. All weights in the network are binary valued (± 1). The hidden nodes and the output node are binary valued perceptrons.

perceptrons. A network is nonoverlapping if each node, including the inputs, has one and only one outgoing connection (fig. 1). This is referred to, within the computational learning community, as the μ or the read-once restriction. Such networks have been investigated, recently, within the PAC learning framework [3, 5].

We derive the analytical expressions for the average generalization rate of the clipped Hebb rule, in the limit of large number of inputs, when learning 1) a two-layer network of nonoverlapping binary perceptrons and 2) a multilayer network of nonoverlapping binary perceptrons. We find that the generalization rates still converge exponentially rapidly to perfect generalization with respect to the number of training examples. The results of the numerical simulations are in very good agreement with the theoretical predictions.

2 Definitions

Let X denote the set $\{-1,+1\}^n$. We are interested in learning a *target function* f^* that maps from the set X (the input space) into $\{-1,+1\}$. We assume f^* is a layered network of nonoverlapping binary perceptrons (fig. 1). For an input vector $\mathbf{x} \in X$, we take x_i to be the state of the input node i of the network.

A useful property of nonoverlapping networks is that one can assume, without loss of generality (w.l.o.g.), that all the weights, except those coming directly from the input nodes, are positive [5] [1]. From now on, we assume this is the case and concentrate only on learning the input level weights.

[1]Any other case reduces to this one by an analog to De Morgan's laws that allows us to push negation of weights down to the input level.

For nonoverlapping networks, each input node i has one and only one input level weight J_i. We define $\mathbf{J}$ to be the weight vector obtained from the collection of all these input level weights. Then, each possible setting of the weight vector $\mathbf{J}$ defines a mapping function f. We denote by $\mathbf{J}^*$ the weight vector associated with f^*. We call $\mathbf{J}^*$ the *target weight vector* and the corresponding network the *target network*. Each perceptron (hidden unit) in the target network is referred to as a *target perceptron*.

The training examples are input vectors $\{\mathbf{x}^l\}_{l=1,\ldots,m}$, generated according to the uniform distribution D on X, and labeled according to the target function (network) f^*. An example is said to be positive (negative) if $f^*(\mathbf{x}) = +1$ (-1).

Knowing the target network architecture and using the training examples, the goal of the learning algorithm is to find a setting of $\mathbf{J}$ that approximate the most the target function. The network corresponding to the $\mathbf{J}$ found by the algorithm is called the *hypothesis network*. Each perceptron in the hypothesis network is referred to as an *hypothesis perceptron*.

Let $\sigma^l \equiv f^*(\mathbf{x}^l)$. The clipped Hebb rule, for the network, can be simply written as

$$J_i = \mathrm{sgn}(\sum_{l=1}^{m} \sigma^l x_i^l) \qquad i = 1,\ldots,n \tag{1}$$

where $\mathrm{sgn}(a) = +1$ if $a > 0$ and -1 otherwise. Because the results of this paper are independent of $\mathbf{J}^*$, we assume from now on that $J_i^* = 1$ for $i = 1,\ldots,n$.

We will denote by $P(A)$ the probability that the event A occurs, and by $P(A \mid B)$ the conditional probability that event A occurs given the fact that event B has been observed. All probabilities are taken with respect to the uniform distribution D on X.

3 Learning a Two-layer Network of Nonoverlapping Binary Perceptrons

Let the target function f^* be a two-layer network of k nonoverlapping perceptrons with binary weights and zero thresholds. That is,

$$f^* = \mathrm{sgn}\left(\sum_{j=1}^{k} g_j^*\right)$$

This is the so-called nonoverlapping Committee Machine [11]. We assume, w.l.o.g., that k is odd. Then, a negative (positive) example is classified negative (positive) by at least $(k+1)/2$ target perceptrons.

We denote by $\mathbf{Ir}(\mathbf{x})$ the vector $(g_1(\mathbf{x}),\ldots,g_k(\mathbf{x}))$. This is called the *internal representation* of $\mathbf{x}$. Obviously, $\mathbf{Ir}(\mathbf{x})$ depends on the setting of $\mathbf{J}$. We denote by $\mathbf{Ir}^*(\mathbf{x})$ the *target* internal representation, *i.e.*, the one corresponding to $\mathbf{J}^*$.

As usual, we let the number of training examples depend on the number of weights, and we write $m = \alpha n$. We are interested in the limit where $n \to \infty$ and α is finite.

Let R_j denote the overlap between the weight vectors associated with the target perceptron g_j^* and the corresponding hypothesis perceptron g_j:

$$R_j = \frac{\sum_{i \in S_j} J_i^* J_i}{|S_j|} \tag{2}$$

where S_j denotes the set of indices of variables connected to g_j^* $(j = 1, \ldots, k)$.

Let $G_j(\alpha)$ denote the generalization rate of the hypothesis perceptron g_j, *i.e.*the probability that g_j agrees with the target perceptron g_j^* on a new random example $\mathbf{x}$. It is well known that, under the uniform distribution, $G_j(\alpha)$ depends only on R_j and is given by [9]

$$G_j(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}(\overline{R_j}) \tag{3}$$

where the overbar denotes the average with respect to all training sets of size αn.

If we assume for simplicity that each perceptron is connected to the same number of inputs n/k, then R_j and G_j are the same for all the perceptrons. Let us put $R_j \equiv R$ and $G_j \equiv G$.

Let $y_i^l = \sigma^l x_i^l$ and $Y_i = \sum_{l=1}^{\alpha n} y_i^l$. Eq. 1 can be written now as

$$J_i = \mathrm{sgn}\left(\sum_{l=1}^{\alpha n} y_i^l\right) = \mathrm{sgn}(Y_i) \tag{4}$$

Let us define q such that

$$P(y_i^l = +1) = \frac{1}{2} + \frac{q}{\sqrt{n}} \quad \text{as } n \to \infty.$$

where q is independent of i and l. Note here that $\frac{q}{\sqrt{n}}$ reflects the correlation between the state of each input node and the output node. This correlation is positive if $J_i^* = +1$ and negative if $J_i^* = -1$. The clipped Hebb rule exploits this correlation to determine the sign (value) of J_i.

The random variable Y_i is simply the sum of αn independent and identically distributed, ± 1 random variables. Thus, according to the central limit theorem, as $n \to \infty$, Y_i will be distributed according to a normal distribution with mean $\mu = 2\alpha n \frac{q}{\sqrt{n}}$ and variance $\sigma = \sqrt{\alpha n}$. Hence,

$$\begin{aligned} \overline{J_i} = P(Y_i > 0) - P(Y_i < 0) &= \frac{2}{\sqrt{\pi}} \int_0^{\mu/\sqrt{2}\sigma} e^{-t^2} dt \\ &\equiv \mathrm{erf}(q\sqrt{2\alpha}) \end{aligned} \tag{5}$$

where erf() denotes the error function. This yields,

$$\overline{R} = \mathrm{erf}(q\sqrt{2\alpha}) \tag{6}$$

To specify the value of q, assume that x_i is connected to perceptron g_j^*. Then, from the definition of y_i, we can write

$$\begin{aligned} & P(y_i = 1) \equiv P(\sigma x_i = 1) \\ = \;& P(f^*(\mathbf{x}) = 1) \\ \times \;& [P(g_j^* = 1 | f^*(\mathbf{x}) = 1) \times P(x_i = 1 | g_j^* = 1) \\ + \;& P(g_j^* = -1 | f^*(\mathbf{x}) = 1) \times P(x_i = 1 | g_j^* = -1)] \\ + \;& P(f^*(\mathbf{x}) = -1) \\ \times \;& [P(g_j^* = 1 | f^*(\mathbf{x}) = -1) \times P(x_i = -1 \mid g_j^* = 1) \\ + \;& P(g_j^* = -1 | f^*(\mathbf{x}) = -1) \times P(x_i = -1 | g_j^* = -1)] \end{aligned} \tag{7}$$

The different probabilities in the above equation can be evaluated fairly easily. For example, $P(x_i = 1 \mid g_j^* = 1)$ is is simply the probability that an input variable in g_j^* is set to $+1$ given the fact that g_j^* has at least $(n/k+1)/2$ of its inputs set to $+1$. Under the uniform distribution, this is given by

$$\begin{aligned} P(x_i = 1 \mid g_j^* = 1) &= \frac{\sum_{r=\frac{n/k+1}{2}}^{n/k} \binom{n/k}{r} \frac{r}{n/k}}{\sum_{r=\frac{n/k+1}{2}}^{n/k} \binom{n/k}{r}} \\ &= \frac{1}{2} + \frac{\binom{n/k-1}{\frac{n/k-1}{2}}}{2^{n/k}} \end{aligned}$$

Thus, after few manipulations, eq. 7 evaluates to

$$P(y_i = 1) = 2\frac{\binom{k-1}{\frac{k-1}{2}}}{2^k} \frac{\binom{n/k-1}{\frac{n/k-1}{2}}}{2^{n/k}}$$

As n and $n/k \to \infty$, this reduces to

$$P(y_i = 1) = \frac{1}{2} + 2\frac{\binom{k-1}{\frac{k-1}{2}}}{2^k} \frac{\sqrt{k}}{\sqrt{2\pi n}} \tag{8}$$

And hence

$$q = 2\frac{\binom{k-1}{\frac{k-1}{2}}}{2^k} \frac{\sqrt{k}}{\sqrt{2\pi}} \tag{9}$$

Deriving an expression for G (and the overall generalization of the hypothesis network) for an arbitrary k is a difficult task. In the following, we concentrate on the two limiting cases: the case $k = 3$ and the case of large k. Note that the learning curves for an arbitrary k will lay in between these corresponding to the two limiting cases.

3.1 The Case of $k = 3$

For $k = 3$, eq. 9 yields $q = \frac{\sqrt{3}}{2}\frac{1}{\sqrt{2\pi}}$. Putting this value back in eqs. 6 and 3, we get

$$\overline{R} = \mathrm{erf}\left(\frac{\sqrt{3}}{2}\sqrt{\frac{\alpha}{\pi}}\right) \tag{10}$$

$$G(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\text{erf}\left(\frac{\sqrt{3}}{2}\sqrt{\frac{\alpha}{\pi}}\right)\right) \tag{11}$$

Let us compare this result with the generalization rate of the same rule when learning single perceptrons with binary weight, which is given by [2]:

$$G_{\text{single}}(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\text{erf}\left(\sqrt{\alpha/\pi}\right)\right) \tag{12}$$

Eq. 11 is equivalent to eq. 12, with an $\alpha_{\text{eff}} = \frac{3}{4}\alpha$. That is to say, if we take the single perceptron as reference, the *effective* fraction of training examples contributing to the learning process now is around 3/4. This is due to the decrease in the correlation between the states of the input nodes and the output node. But nonetheless, $G(\alpha)$ still converges exponentially to 1.

The overall generalization rate, denoted $G_T(\alpha)$, is the probability that the hypothesis network will classify correctly a new positive (negative) example. This probability depends on the target internal representation of the example tested, more precisely on how many target perceptrons classify this example as positive/negative. For a positive example $\mathbf{x}$, we have two possibilities:

1) $\mathbf{x}$ is classified positive by 2 target perceptrons, say by g_1^* and g_2^*, and negative by the remaining target perceptron, g_3^*. The hypothesis network can fail to classify this example correctly only if

$$\begin{array}{llll} & g_1(\mathbf{x}) \neq g_1^*(\mathbf{x}), & g_2(\mathbf{x}) \neq g_2^*(\mathbf{x}), & g_3(\mathbf{x}) = g_3^*(\mathbf{x}) \\ \text{or} & g_1(\mathbf{x}) \neq g_1^*(\mathbf{x}), & g_2(\mathbf{x}) = g_2^*(\mathbf{x}), & g_3(\mathbf{x}) = g_3^*(\mathbf{x}) \\ \text{or} & g_1(\mathbf{x}) = g_1^*(\mathbf{x}), & g_2(\mathbf{x}) \neq g_2^*(\mathbf{x}), & g_3(\mathbf{x}) = g_3^*(\mathbf{x}) \\ & \text{or} & g_j(\mathbf{x}) \neq g_j^*(\mathbf{x}) \quad \text{for } j = 1,2,3 & \end{array}$$

This can happen with a probability

$$G(\alpha)[1-G(\alpha)]^2 + 2G(\alpha)^2[1-G(\alpha)] + [1-G(\alpha)]^3$$

2) $\mathbf{x}$ is classified positive by all 3 target perceptrons. The hypothesis network can fail to classify this example correctly only if at least two of its perceptrons fail to do so. This can happen with a probability

$$3G(\alpha)[1-G(\alpha)]^2 + [1-G(\alpha)]^3$$

The same argument holds for negative examples. Taking into account the probability that an example is classified positive (negative) by r target perceptrons, we get

$$\begin{aligned} G_T(\alpha) = 1 \quad &- \quad \frac{3}{2}G(\alpha)^2[1-G(\alpha)] - \frac{3}{2}G(\alpha)[1-G(\alpha)]^2 \\ &- \quad [1-G(\alpha)]^3 \end{aligned} \tag{13}$$

The analytical expressions for $\overline{R}$, $G(\alpha)$, and $G_T(\alpha)$ are plotted in fig.2 along with the simulation results. Again, the agreement is excellent. One can also see that $G_T(\alpha)$ tends exponentially to perfect generalization.

The argument of this section may be used in principle to derive an expression for $G_T(\alpha)$ for any value of k. However, it becomes too complicated to follow for $k \geq 7$. Thus, we will look simply at the other end of the spectrum, *i.e.*, large k.

3.2 The Case of Large k

Here we are interested in the case where $k \to \infty$ (but n is still larger than k such that $n/k \to \infty$). In this case, eq. 9 reduces to

$$q \sim \frac{2}{\sqrt{2\pi}}\frac{1}{\sqrt{2\pi}}. \tag{14}$$

Putting this value back in eqs. 6 and 3, we get

$$\overline{R} = \text{erf}\left(\sqrt{\frac{2}{\pi}}\sqrt{\frac{\alpha}{\pi}}\right) \tag{15}$$

$$G(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\text{erf}\left(\sqrt{\frac{2}{\pi}}\sqrt{\frac{\alpha}{\pi}}\right)\right) \tag{16}$$

Comparing this to the case of single binary perceptrons (eq. 12), we see that a fraction $2/\pi$ ($\simeq 0.63$) of the training examples contribute to the learning process, and that $G(\alpha)$ still converges exponentially to 1.

To determine the overall generalization, let $\mathbf{x}$ be a random input and let

$$a = \sum_{j=1}^{k} g_j^*(\mathbf{x}) \qquad b = \sum_{j=1}^{k} g_j(\mathbf{x})$$

For different inputs $\mathbf{x}$, a and b are correlated Gaussian variables with

$$\overline{a} = \overline{b} = 0 \qquad \overline{a^2} = \overline{b^2} = k \qquad \overline{ab} = k \times \overline{\rho}$$

where ρ, the overlap between $\mathbf{Ir}^*(\mathbf{x})$ and $\mathbf{Ir}(\mathbf{x})$, is given by

$$\rho = \frac{\sum_{j=1}^{k} g_j^*(\mathbf{x})g_j(\mathbf{x})}{k}$$

By definition,

$$G_T(\alpha) \equiv P(ab > 0)$$

which, as for a single perceptron, depends only on the average overlap $\overline{\rho}$. Thus $G_T(\alpha)$ is again given by eq. 3

$$G_T(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}(\overline{\rho})$$

We now evaluate $\overline{\rho}$ (remember that the overbar denotes the average with respect to the training set). For that, let

$$h_j(\mathbf{x}) = g_j^*(\mathbf{x})g_j(\mathbf{x}) \qquad j = 1, \ldots, k$$

Then,

$$P(h_j(\mathbf{x}) = +1) = G(\alpha) \qquad \overline{h_j(\mathbf{x})} = 2G(\alpha) - 1$$

This yields

$$\overline{\rho} = 2G(\alpha) - 1 = 1 - \frac{2}{\pi}\cos^{-1}(\overline{R}) \tag{17}$$

So, the overall generalization is given by

$$\begin{aligned} G_T(\alpha) &= 1 - \frac{1}{\pi}\cos^{-1}(2G(\alpha) - 1) & (18) \\ &= 1 - \frac{1}{\pi}\cos^{-1}\left(1 - \frac{2}{\pi}\cos^{-1}(\overline{R})\right) & (19) \end{aligned}$$

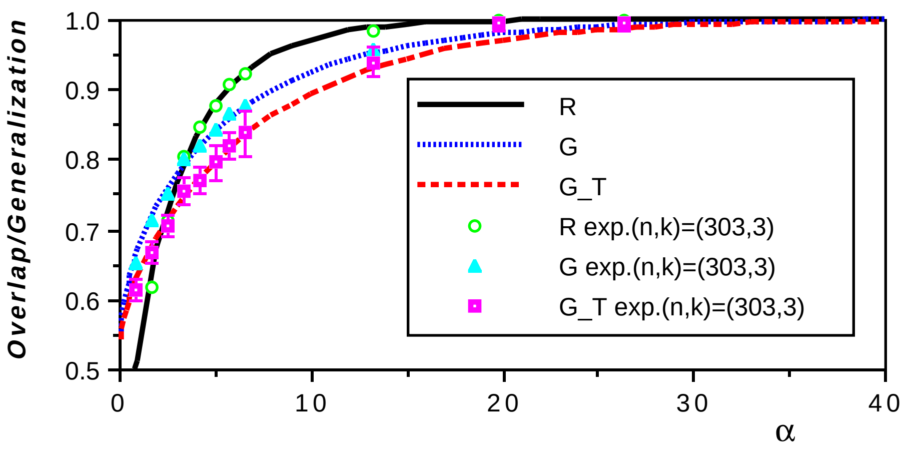

Figure 2: Learning a two-layer network of 3 nonoverlapping binary perceptrons connected to the same number of inputs. Shown are the average overlap R, the generalization rate of each perceptron G, and the overall generalization rate G_T. The points are the results of the simulations for $n = 303$. Each point denotes an average over 25 different training samples. The error bars, shown only for one curve for clarity, denote the standard deviations.

It is interesting to see that, for large k, the generalization rate of a majority of nonoverlapping perceptrons behaves as that of a single perceptron, with a modified overlap $1 - \frac{2}{\pi}\cos^{-1}(\overline{R})$. Eq. 19 has been also derived in [7, 11], using a different method.

The analytical expressions for $G(\alpha)$ and $G_T(\alpha)$ are plotted in fig.3 along with the simulation results. There is a noticeable deviation from the theoretical predictions; the reason for this is that, in the simulations, k and n/k are not sufficiently large. On the other hand, one can see that as k and n/k become larger, the simulations results tend towards the theoretical curves. One can also see that $G_T(\alpha)$ tends exponentially to perfect generalization.

Finally, the nonoverlapping Committee Machine with binary weights has been studied extensively using the statistical mechanics approach [7, 11]. This approach assumes a stochastic training algorithm, similar to a finite Monté Carlo process, that leads at long times to a Gibbs distribution of weights [12]. Under this assumption, it is found that a phase transition to perfect generalization does occur at a critical value of α. Thus, such training algorithms have a better sample complexity than the clipped Hebb rule. However, the time complexity of the clipped Hebb rule is only $O(n \times m)$, whereas stochastic training algorithms generally require prohibitively long convergence times.

4 Extension to Multilayer Networks of Nonoverlapping Binary Perceptrons

Let f^* be a layered network of nonoverlapping binary perceptrons (fig. 1). Let H denote the number of hidden layers and k_h the number of perceptrons in layer h ($h = 1, \ldots, H$). Assume that the number of nodes in layer $h-1$ is much greater than the number of nodes in layer h. That is

$$n \to \infty$$

$$n/k_1 \to \infty$$

$$k_{h-1}/k_h \to \infty \qquad h = 2, \ldots, H$$

Assume, for simplicity, that perceptrons in the same layer are connected to the same number of nodes in the previous layer. Let $G_h(\alpha)$ denotes the generalization rate of a perceptron (hidden unit) in layer h.

Using the arguments of the previous section that led to eq. 14, one can show that each hidden layer will contribute a factor $2/\sqrt{2\pi}$ to q. Thus,

$$q \sim \left(\frac{2}{\sqrt{2\pi}}\right)^H \frac{1}{\sqrt{2\pi}} \tag{20}$$

With this, eq. 15 reads now

$$\overline{R} = \operatorname{erf}\left(\left(\sqrt{\frac{2}{\pi}}\right)^H \sqrt{\frac{\alpha}{\pi}}\right) \tag{21}$$

$G_1(\alpha)$ is still given by

$$G_1(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}(\overline{R}) \tag{22}$$

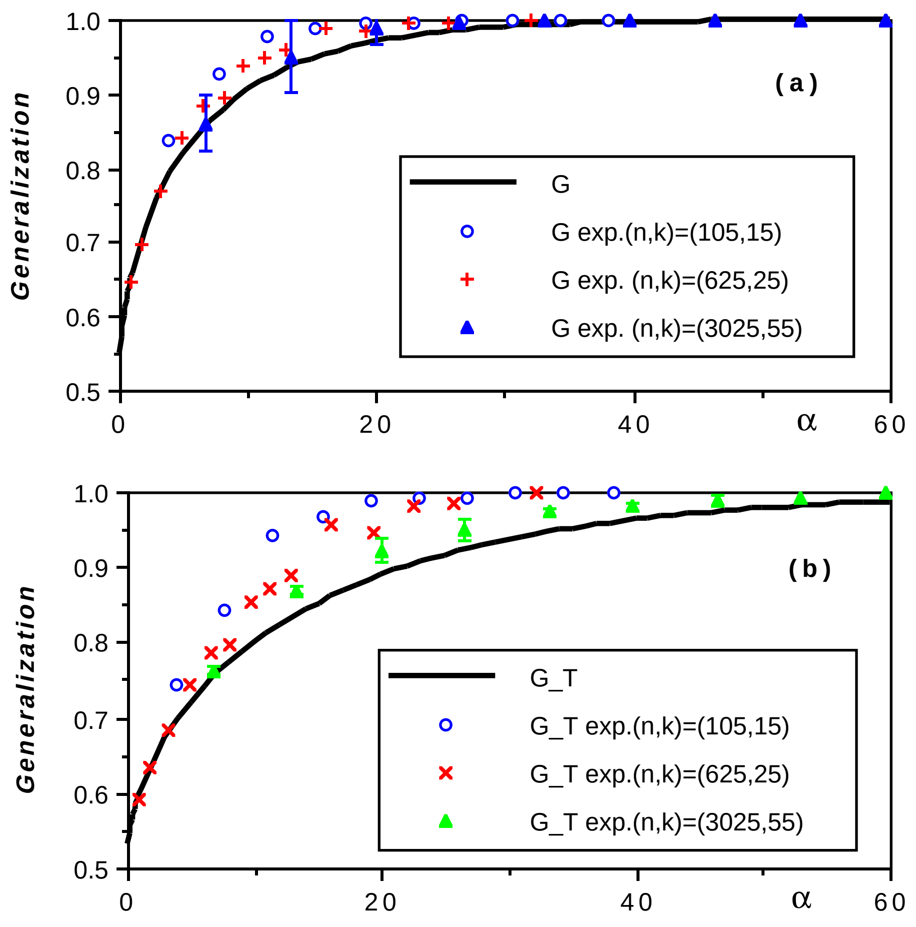

Figure 3: Learning a two-layer network of k nonoverlapping binary perceptrons connected to the same number of inputs. The case of large k. Shown are (a) the generalization rate of each perceptron G, and (b) the overall generalization rate G_T. The points are the results of the simulations for the indicated values of $(n{,}k)$. Each point denotes an average over 25 different training samples. The error bars, shown only for one curve for clarity, denote the standard deviations.

Applying the arguments that led to eq. 18 recursively, that is from one layer to the next, we get

$$G_h(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(2G_{h-1}(\alpha) - 1\right) \qquad h = 2, \ldots, H. \tag{23}$$

$$G_T(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(2G_H(\alpha) - 1\right) \tag{24}$$

Finally, we note that eq. 21 can be written as

$$\overline{R} = \mathrm{erf}(\sqrt{\alpha_{\mathrm{eff}}/\pi}) \tag{25}$$

where

$$\alpha_{\mathrm{eff}} = \left(\frac{2}{\pi}\right)^H \alpha$$

Compared to the single binary perceptron case, α_{eff} reflects the *effective* number of examples contributing to the learning process. This effective number decreases as $(2/\pi)^H$. This may explain the observation made in [7] that the critical value of α at which the phase-transition (for consistent rules) occurs scales as $(\pi/2)^H$.

5 Conclusion

We have investigated the clipped Hebb rule for learning different networks of nonoverlapping binary perceptrons under the uniform distribution. We have calculated exactly the learning curves of this rule in the limit $n \to \infty$, where the average behavior becomes the typical one. Our results indicate that the clipped Hebb rule does indeed learn this class of networks. Specifically, the generalization rates converge exponentially to perfect generalization as a function of the number of training examples. The analytic expression of the learning curves are in excellent agreement with the numerical simulations. These results are very encouraging given the simplicity of the learning rule. In particular, this shows that simple neural networks with binary weights may be learnable under simple distributions of examples.

We note here that the clipped Hebb rule produces hypotheses that are not necessarily consistent with all the training examples, but that nonetheless have very good generalization ability. These type of algorithms, called "inconsistent algorithms" [8], are very important because, in many situations, there is no hypothesis consistent with all the training examples. This may be due to the intrinsic difficulty of the problem or to the examples being noisy. The clipped Hebb rule in particular is very robust with respect to classification noise [2].

An interesting question is how the clipped Hebb rule behaves under product distributions. To answer this, we note that the clipped Hebb rule works by exploiting the *correlation* between the state of each input variable x_i and the classification label (eq. 1). Under the uniform distribution, this correlation is positive if $J_i^* = +1$ and negative if $J_i^* = -1$. This is no more true for product distributions: one can easily craft some malicious product distributions where, for example, this correlation is negative although $J_i^* = +1$.

Finally, we note that throughout this paper, we have assumed that the architecture is known in advance. Whereas this is in line with most of the neural network research, it is hardly justifiable in practice. it would be very interesting to be able to calculate the learning curve(s) of an algorithm that can find both the weight values and the architecture of even the simplest networks such as unions of nonoverlapping perceptrons [3].

Acknowledgments: This work was supported by NSERC grant OGP0122405.

References

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

[2] Golea M. and Marchand M., "On Learning Perceptrons with Binary Weights", *To appear* in *Neural Computation.*

[3] Golea M., Marchand M., and Hancock T., "On Learning μ-Perceptron Networks with Binary Weights", to *appear* in *Advances in Neural Information Processing Systems*, Vol.5, 1992.

[4] Gyorgyi G., "First-order Transition to Perfect Generalization in a Neural Network with Binary Synapses", *Phys. Rev. A*, Vol. 41, (1990), 7097–7100.

[5] Hancock T., Golea M., and Marchand M., "Learning Nonoverlapping Perceptron Networks From Examples and Membership Queries", TR-26-91, Center for Research in Computing Technology, Harvard University. *To appear* in *Machine Learning*.

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

[7] Mato G. and Parga N., "Generalization Properties of Multilayerd Neural Networks", *J. Phys. A: Math. Gen.*, Vol. 25, (1992), 5047–5054.

[8] Meir R., Fontanari J. F., "Calculation of Learning Curves for Inconsistent Algorithms", *Phys. Rev. A*, Vol. 45, (1992), 8874–8884.

[9] 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.

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

[11] Schwarze H. and Hertz J., "Generalization in a Large Committee Machine", *Europhys. Lett.*, Vol. 20, (1992), 375-380.

[12] Seung H. S., Sompolinsky H., and Tishby N., "Statistical Mechanics of Learning from Examples", *Phys. Rev. A*, Vol. 45, (1992), 6056–6091.

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