Learning curves of the clipped Hebb rule for networks with binary weights

1993 J. Phys. A: Math. Gen. 26 5751

(http://iopscience.iop.org/0305-4470/26/21/015)

J. Phys. A: Math. Gen. 26 (1993) 5751–5766. Printed in the UK

Learning curves of the clipped Hebb rule for networks with binary weights

Mostefa Golea† and Mario Marchand‡
Ottawa-Carleton Institute for Physics, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5

Received 30 November 1992, in final form 6 July 1993

Abstract. Networks with binary weights are very important from both the theoretical and practical points of view. In this paper, we investigate the clipped Hebb rule for learning different networks of non-overlapping binary perceptrons under uniform distribution. We calculate exactly its learning curves in the limit of a large number of inputs, where the average behaviour becomes the typical behaviour. This calculation is performed using only very simple counting arguments and the central limit theorem. The results indicate that the clipped Hebb rule does indeed learn this class of networks. In particular, the generalization rates converge extremely rapidly, often exponentially, to perfect generalization. These results are very encouraging given the simplicity of the learning rule. The analytic expression of the learning curves are in excellent agreement with the numerical simulations.

1. Introduction

Neural networks with binary weights have attracted much attention recently [1–7]. 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 those of a network with continuous weights [2, 8]. Second, the hardware realization of binary networks may prove to be simpler.

One interesting property of neural networks is their generalization ability. This is defined as the probability that a trained network will predict the correct classification of new examples. The generalization properties of neural networks with binary weights have been studied extensively using the statistical mechanics approach [2, 5, 8, 9]. Although this approach has yielded some impressive results, it has its shortcomings. In particular, it neglects the computational aspect of the learning process. It assumes a stochastic training algorithm, similar to a finite Monte Carlo process, that at long times leads to a Gibbs distribution [8]. Unfortunately, stochastic training algorithms generally require prohibitively long convergence times. So, despite intensive study, the fundamental question of whether or not there exist efficient algorithms for learning this class of networks remains largely unanswered. The reason for this state of affairs perhaps lies in the apparent strength of the following distribution independent result [10]: learning perceptrons with binary weights are equivalent to 0-1 integer programming and hence the problem becomes an NP-complete one. However, this result does not rule out the existence of efficient learning algorithms that work well under some reasonable distributions of examples.

† e-mail: golea@physics.uottawa.ca
‡ e-mail: mmmsj@acadvm1.uottawa.ca

0305-4470/93/215751+16$07.50

Perhaps the simplest algorithm that one may think of for learning binary networks is the clipped Hebb rule [3] (also called the majority rule in [7]). This rule is local, homogeneous and simple enough to be biologically plausible and easy to implement. Moreover, it observes each training example only once and hence its running time increases only linearly with the number of training examples.

Recently, we decided to take a close look at the clipped Hebb rule. We investigated its behaviour when learning single binary perceptrons under a uniform distribution [11]. We showed that within the 'probably approximately correct' (PAC) learning framework [12, 13], the clipped Hebb rule does indeed learn this class of perceptrons. Also, we calculated its learning curve, i.e. the average generalization rate as a function of the size of the training set. We found that the generalization rate converges exponentially to perfect generalization as a function of the number of training examples. These findings were confirmed by extensive simulations.

In this paper, we take this investigation one step further. We look at the average behaviour of the clipped Hebb rule when learning networks of non-overlapping binary perceptrons. A network is non-overlapping if each node, including the inputs, has one and only one outgoing connection (figure 1). This is referred to, within the computational learning community, as the μ or the read-once restriction [14, 15]. Such networks have been recently investigated within the PAC learning framework [16, 17].

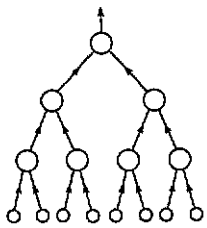

Figure 1. A multilayer network of non-overlapping binary perceptrons. Note that each node has one and only one outgoing connection. All weights in the network are binary valued (± 1). The hidden nodes and the output node are binary-valued perceptrons.

We show that, under the uniform distribution of examples, the clipped Hebb rule does indeed learn this class of networks. We derive expressions for the average generalization rate of this rule, in the limit of large number of inputs, when learning (i) a union of non-overlapping binary perceptrons; (ii) a two-layer network of non-overlapping binary perceptrons; and (iii) a multilayer network of non-overlapping binary perceptrons. This is done using only very simple counting arguments and the central limit theorem. We find that the generalization rates still converge extremely rapidly, often exponentially, to perfect generalization. The results of extensive simulations are in very good agreement with the theoretical predictions.

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 types of algorithm are called 'inconsistent algorithms' [18]. Such algorithms 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 known to be very robust with respect to random classification/input noise [11].

Finally, we should mention that the analysis presented in this paper can be easily extended to handle the case of learning networks of non-overlapping perceptrons with real weights using the Hebb rule [19].

For the sake of completeness and because we will make use of the technique, a short derivation of the single binary perceptron's learning curve is included in this paper.

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 either a single binary perceptron or a network of non-overlapping binary perceptrons (figure 1). For an input vector $x \in X$, we take x_i to be the state of the input node i of the network.

One interesting property of non-overlapping networks is that one can assume, without loss of generality (WLOG), that all the weights, except those coming directly from the input nodes, are positive [17]†. From now on, we assume this is the case and concentrate only on learning the input level weights.

For non-overlapping networks, each input node i has one and only one input level weight J_i. We define J to be the weight vector obtained from the collection of all these input level weights. Then, each possible setting of the weight vector J defines a mapping function f. We denote by J^* the weight vector associated with f^*. We call 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 $\{x^l\}_{l=1,\ldots,m}$, generated according to the uniform distribution D on X, and labelled according to the target function (network) f^*. An example is said to be positive (negative) if $f^*(x) = +1(-1)$.

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

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

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

where $\operatorname{sgn}(a) = +1$ if $a > 0$ and -1 otherwise. Note that the learning rule (equation (1)) does not require that the target values for the internal nodes be provided: only the target values for the output node need to be specified.

Because the results of this paper are independent of J^*, we assume from now on that $J_i^* = 1$ for $i = 1,\ldots,n$.

Before we leave this section, we define several probabilistic quantities that will be useful later. We denote by $P(A)$ the probability that the event A occurs. We denote by $P(A\&B)$ the probability that events A and B do occur simultaneously. We denote by $P(A|B)$ 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 single binary perceptrons

First, we look at the case where the target function (network) f^* is a single perceptron g^* with binary weight vector J^*,

$$f^*(x) \equiv g^*(x) = \begin{cases} +1 & \text{if } \sum_{i=1}^{i=n} J_i^* x_i > 0 \\ -1 & \text{if } \sum_{i=1}^{i=n} J_i^* x_i \leqslant 0. \end{cases} \tag{2}$$

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

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

Let g, with weight vector J, be the perceptron returned by the clipped Hebb learning rule. Let R denote the overlap between J^* and J, i.e.

$$R = \sum_{i=1}^{n} J_i^* J_i / n.$$

The generalization rate, $G(\alpha)$, is defined as the probability that the hypothesis perceptron g agrees with the target perceptron g^* on a new random example x, drawn according to D. It is well known that, under uniform distribution, $G(\alpha)$ depends only on R and is given by [20]

$$G(\alpha) = 1 - (1/\pi)\cos^{-1}(\overline{R}) \tag{3}$$

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

We derive an expression for $\overline{R}$ when learning a binary perceptron using the clipped Hebb rule. For this, let $y_i^l = \sigma^l x_i^l$ and let $Y_i = \sum_{l=1}^{\alpha n} y_i^l$. Then, equation (1) can be written as

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

Now, Y_i is simply the sum of αn independent and identically distributed, ± 1 random variables. Let us define q such that

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

where, as we shall see, q is independent of i and l.

Note here that $q/\sqrt{n}$ reflects the correlation between the state of each input node and the output node (for a random function, $P(y_i = +1) = 1/2$ and $q = 0$). 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.

According to the central limit theorem, as $n \to \infty$, Y_i will be distributed according to a normal distribution with mean $\mu = 2\alpha n q/\sqrt{n}$ and variance $\sigma = \sqrt{\alpha n}$. Hence,

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

where erf denotes the error function. This yields

$$\overline{R} = \frac{\sum_{i=1}^{n} J_i^* \overline{J_i}}{n} = \operatorname{erf}(q\sqrt{2\alpha}). \tag{5}$$

We need to specify the value of q. The reason we did not do this earlier is that we will make use of equation (5), in its general form, later. From the definition of y_i^l, it is easy to

see that $P(y_i^l = +1)$ is simply the probability that an input variable is set to $+1$ when each negative training example x^l is replaced by $-x^l$. Note that, once the negative examples are inverted, each example has at least $(n+1)/2$ of its inputs set to $+1$. Under the uniform distribution, $P(y_i^l = +1)$ is given by

$$\begin{aligned} P(y_i^l = +1) &= \sum_{r=(n+1)/2}^{n} \binom{n}{r} \frac{r}{n} \Big/ \sum_{r=(n+1)/2}^{n} \binom{n}{r} \\ &= \frac{1}{2} + \binom{n-1}{\frac{1}{2}(n-1)} \Big/ 2^n \\ &\sim \frac{1}{2} + \frac{1}{\sqrt{2\pi n}} \qquad \text{as } n \to \infty. \end{aligned} \tag{6}$$

It follows that

$$q = 1/\sqrt{2\pi} \tag{7}$$

$$\overline{R} = \operatorname{erf}\left(\sqrt{\alpha/\pi}\right) \tag{8}$$

$$G(\alpha) = 1 - 1/\pi \cos^{-1}\left(\operatorname{erf}\left(\sqrt{\alpha/\pi}\right)\right) \tag{9}$$

i.e. the generation rate tends to 1 exponentially as a function of the number of training examples. Compared to this, the generalization rates of algorithms that learn binary perceptrons using perceptrons with real weights improve only algebraically as a function of the number of training examples [19–21]. We will see that the exponential convergence remains for more complicated binary networks. This is another reason why we should stick with binary network solutions whenever they exist.

Finally, as reported in [11], the agreement between this section's results and the numerical simulations is excellent, even for moderate values of n.

4. Learning a union of non-overlapping binary perceptrons

Let us now assume that the target function (network) f^* is a union of k non-overlapping binary perceptrons:

$$f^* = g_1^* \vee g_2^* \vee \ldots \vee g_k^*.$$

In other words, the target network is a two-layer network of non-overlapping binary perceptrons where the output node computes an OR function. In this case, a negative example is classified negative by each and every target perceptron. A positive example is classified positive by one or more target perceptrons.

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

Let $G_j(\alpha)$ denote the generalization rate of the hypothesis perceptron g_j, i.e. the probability that g_j agrees with the corresponding target perceptron g_j^* on a new random example x. Let R_j denotes the overlap between the weight vectors associated with g_j^*

and g_j. Assume, for now, that each perceptron is connected to the same number of input variables, n/k. Then

$$R_j = \frac{\sum_{i \in S_j} J_i^* J_i}{n/k} \tag{10}$$

where S_j denotes the set of indices of variables connected to g_j^* $(j = 1, \ldots, k)$. It is easy to see that, under our assumptions, R_j and G_j are the same for all perceptrons. So, let us put

$$R_j \equiv R \qquad \text{and} \qquad G_j \equiv G.$$

Let y_i^l be defined as in the previous section and let us write again

$$P(y_i^l = +1) = \frac{1}{2} + \frac{q_u}{\sqrt{n/k}} \qquad \text{as } n/k \to \infty$$

for some q_u. The $\sqrt{k}$ factor comes from the fact that each perceptron is connected only to n/k inputs. Then, for n and $n/k \to \infty$, $\overline{R}$ is still given by equation (5), with q substituted for $q_u\sqrt{k}$. In other words,

$$\overline{R} = \operatorname{erf}\left(q_u \sqrt{k}\sqrt{2\alpha}\right). \tag{11}$$

Also, $G(\alpha)$ is still given by equation (3),

$$G(\alpha) = 1 - (1/\pi)\cos^{-1}(\overline{R}). \tag{12}$$

To specify the value of q_u, we first remind the reader that $q_u/\sqrt{n/k}$ reflects simply the correlation between the state of each input node and the state of the output node. We use the following intuitive argument to calculate q_u. Let x^l be a positive example with a target internal representation $Ir^*(x^l)$. If $Ir^*(x^l)$ has at least one of its components set to -1, then $-x^l$ is also a positive example. Under the uniform distribution D, x^l and $-x^l$ are equally probable to occur in the training set and so their combined contribution to q_u is null. We are left with the positive examples for which $Ir^*(x^l)$ has all its components set to $+1$, and with the negative examples for which, obviously, $Ir^*(x^l)$ has all its components set to -1. It is easy to see that the contribution of these examples to q_u is exactly q. This is true because, for these examples, $f^*(x^l) = g_j^*(x^l)$, $f^*(-x^l) = g_j^*(-x^l)$ $(j = 1, \ldots, k)$. So, q_u is given by

$$q_u = \frac{2}{2^k} \times q = \frac{2}{2^k} \times \frac{1}{\sqrt{2\pi}} \tag{13}$$

where $2/2^k$ is the probability of drawing an example x^l for which $Ir^*(x^l)$ has all its components set to $+1$ or all set to -1.

Substituting this value of q_u in equation (11), we get

$$\overline{R} = \operatorname{erf}\left(\sqrt{\frac{\alpha}{\pi}} \frac{\sqrt{k}}{2^{k-1}}\right). \tag{14}$$

With this, equation (12) reads

$$G(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\operatorname{erf}\left(\sqrt{\frac{\alpha}{\pi}}\frac{\sqrt{k}}{2^{k-1}}\right)\right). \tag{15}$$

Comparing this to the case of single binary perceptrons (9), we see that only about a fraction of $1/2^{k-1}$ of the training examples in fact contribute to the learning process. This is not due to the inefficiency of the clipped Hebb rule but simply due to the fact that, for most of the positive examples, there is no correlation whatsoever between the state of the output node and the state of the input nodes. In fact, we could use *only* the negative examples in the learning process without any significant loss! It is very likely that any learning algorithm for the union will experience the same difficulty.

We now turn our attention to the the overall generalization rate of the network, $G_T(\alpha)$. This is defined as the probability that the hypothesis network agrees with the target network on a new random example x drawn according to D.

Let $G_T^+(\alpha)$ and $G_T^-(\alpha)$ denote the generalization rates for the positive and negative examples, respectively. The hypothesis network will classify correctly a random negative example if and only if each of its perceptrons does so, i.e. for a negative example,

$$G_T^-(\alpha) = \prod_{j=1}^{k} G(\alpha) = G(\alpha)^k.$$

The probability that the hypothesis network will classify correctly a positive example depends on its target internal representation, i.e. on how many target perceptrons classify this example as positive/negative. Let us consider a positive example x that is classified positive by r target perceptrons, say $g_1^*(x) = 1, \ldots, g_r^*(x) = 1$, and negative by the remaining $k-r$ target perceptrons, $g_{r+1}^*(x) = -1, \ldots, g_k^*(x) = -1$. The hypothesis network can fail to classify this example correctly only if

$$g_j(x) \neq g_j^*(x) \qquad \text{for } j = 1, \ldots, r$$

and

$$g_j(x) = g_j^*(x) \qquad \text{for } j = r+1, \ldots, k.$$

This can happen with a probability $(1-G)^r G^{k-r}$. Taking into account the probability that a positive example is classified positive by r target perceptrons, $G_T^+(\alpha)$ can be written as

$$G_T^+(\alpha) = \sum_{r=1}^{k}\left[\binom{k}{r} \Big/ 2^k - 1\right]\left(1 - (1-G)^r G^{k-r}\right).$$

The overall generalization rate is thus given by

$$G_T(\alpha) = \frac{1}{2^k} \times G_T^-(\alpha) + \left(1 - \frac{1}{2^k}\right) \times G_T^+(\alpha) \tag{16}$$

where $1/2^k$ is the probability that a random drawn example is a negative example and $1 - 1/2^k$ is the probability that a random drawn example is a positive example.

After few manipulations, equation (16) reduces to

$$G_T(\alpha) = 1 - \frac{1}{2^{k-1}}(1 - G(\alpha)^k). \tag{17}$$

The generalization rate $G(\alpha)$ (equation (15)) and the overall generalization rate $G_T(\alpha)$ (equation (17)) are plotted in figure 2, for different values of k. The results of the numerical simulations are also shown. Again, we see that the agreement with the theory is excellent, even for moderate values of n. For small values of k, $G(\alpha)$ converges exponentially to 1 as a function of the number of training examples. But as k increases, $G(\alpha)$ drops very rapidly. On the other hand, the overall generalization, $G_T(\alpha)$, increases very rapidly with k. The latter is due to the fact that the *default* generalization, $G_T(\alpha = 0)$, increases very rapidly with k.

Finally, it is easy to extend the results of this section to the case where the perceptrons are not connected to the same number of inputs. Assume that perceptron g_j^* is connected to n/k_j inputs, Then equations (14), (15) and (17) will become

$$\overline{R_j} = \operatorname{erf}\left(\sqrt{\frac{\alpha}{\pi}}\frac{\sqrt{k_j}}{2^{k-1}}\right) \tag{18}$$

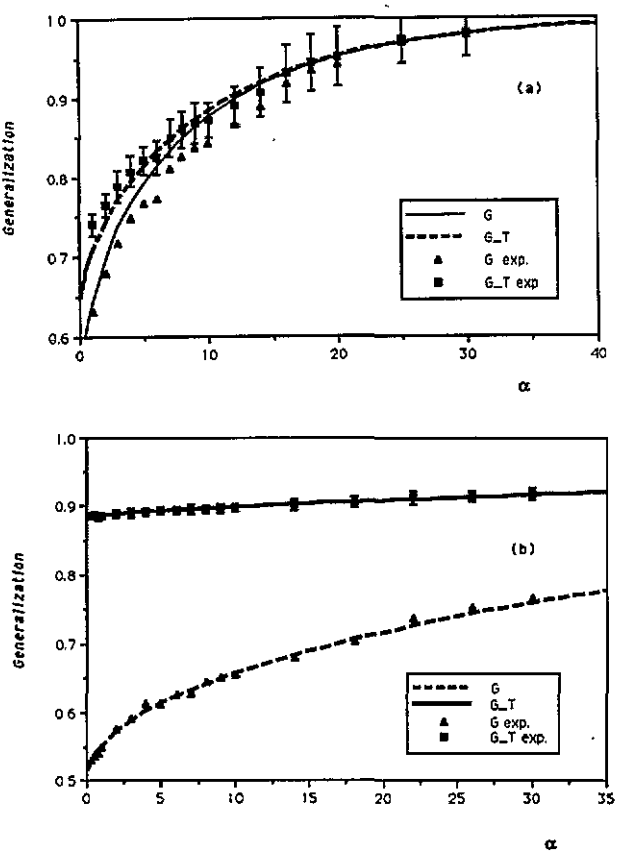

Figure 2. Learning a union of k non-overlapping binary perceptrons connected to the same number of inputs: (a) $k = 2$ and (b) $k = 4$. The average generalization rate of one perceptron in the net, G, and the overall generalization rate, G_T are shown. The points are the results of the simulations for $n = 100$. Each point denotes an average over 25 different training samples. The error bars, shown only for one curve for clarity, denote the standard deviations.

$$G_j(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\operatorname{erf}\left(\sqrt{\frac{\alpha}{\pi}}\frac{\sqrt{k_j}}{2^{k-1}}\right)\right) \tag{19}$$

$$G_T(\alpha) = 1 - \frac{1}{2^{k-1}}\left(1 - \prod_{j=1}^{k} G_j(\alpha)\right). \tag{20}$$

This situation is reproduced in figure 3, for $k = 3$.

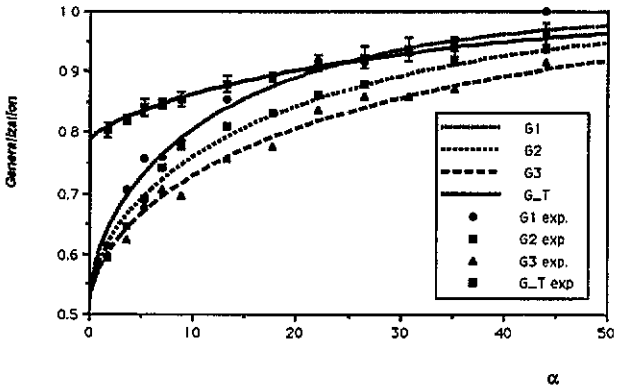

Figure 3. Learning a union of three non-overlapping binary perceptrons connected respectively to n/k_1, n/k_2, and n/k_3, where $k_1 = 4.45098$, $k_2 = 3.02666$, and $k_3 = 2.2475$. The generalization of each perceptron G_i $(i = 1, 2, 3)$, and the overall generalization G_T are shown. The points are the results of the simulations for $n = 227$. Each point denotes an average over 25 different training samples. The error bars, shown only for one curve for clarity, denote the standard deviations.

5. Learning a two-layer network of non-overlapping binary perceptrons

Let us assume that the target function (network) f^* is a two-layer network of k non-overlapping binary perceptrons:

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

In other words, the output node of the target network computes a majority function. This is the so-called non-overlapping committee machine. We assume, WLOG, that k is odd. Then, a negative (positive) example is classified negative (positive) by at least $(k+1)/2$ target perceptrons. Assume again, for simplicity, that each perceptron is connected to the same number of inputs, n/k.

Let $G_j(\alpha) \equiv G$ and $R_j \equiv R$ be defined as in the previous section. Let us again write

$$P(y_i^l = +1) = \frac{1}{2} + \frac{q_m}{\sqrt{n/k}} \qquad \text{as } n/k \to \infty.$$

for some q_m.

Then, for n and $n/k \to \infty$, $\overline{R}$ is still given by equation (5), with q substituted for $q_m\sqrt{k}$. That is

$$\overline{R} = \mathrm{erf}\left(q_m\sqrt{k}\sqrt{2\alpha}\right). \tag{21}$$

Also, $G(\alpha)$ is still given by equation (3),

$$G(\alpha) = 1 - (1/\pi)\cos^{-1}(\overline{R}). \tag{22}$$

We need to determine the value of q_m. Assume that x_i is connected to perceptron g_j^*. First, we note that if $g_j^*(x^l) = -1$, then $P(x_i^l = +1) < P(x_i^l = -1)$ (since we have assumed, WLOG, that $J_i^* = 1$). Likewise, if $g_j^*(x^l) = +1$, then $P(x_i^l = -1) < P(x_i^l = +1)$. Based on this observation, it is easy to see that the contributions to q_m from the following two sources will be negative:

(i) positive examples x^l for which $g_j^*(x^l) = -1$, and

(ii) negative examples x^l for which $g_j^*(x^l) = +1$.

Similarly, the contributions to q_m from the following two sources will be positive:

(i) positive examples x^l for which $g_j^*(x^l) = +1$, and

(ii) negative examples x^l for which $g_j^*(x^l) = -1$.

Moreover, the contribution, in absolute value, of each of these four possibilities to q_m is exactly q. Taking into account the probability that each of the four possibilities mentioned above does occur, we get

$$\begin{aligned} q_m = {} & \left(P(f^*(x^l) = 1 \& g_j^*(x^l) = 1) + P\left(f^*(x^l) = -1 \& g_j^*(x^l) = -1\right)\right) \times q \\ & - \left(P(f^*(x^l) = 1 \& g_j^*(x^l) = -1) + P\left(f^*(x^l) = -1 \& g_j^*(x^l) = 1\right)\right) \times q. \end{aligned} \tag{23}$$

Now,

$$\begin{aligned} P\left(f^*(x^l) = 1 \& g_j^*(x^l) = 1\right) &= P\left(f^*(x^l) = 1\right) \times P\left(g_j^*(x^l) = 1 | P(f^*(x^l) = 1\right) \\ &= \frac{1}{2} \sum_{r=(k+1)/2}^{k} \binom{k}{r} \frac{r}{k} \Big/ \sum_{r=(k+1)/2}^{k} \binom{k}{r} \\ &= \frac{1}{2}\left(\frac{1}{2} + \binom{k-1}{\frac{1}{2}(k-1)} \Big/ 2^k\right). \end{aligned}$$

Similarly,

$$P\left(f^*(x^l) = -1 \& g_j^*(x^l) = -1\right) = \frac{1}{2}\left(\frac{1}{2} + \binom{k-1}{\frac{1}{2}(k-1)} \Big/ 2^k\right)$$

$$P\left(f^*(x^l) = 1 \& g_j^*(x^l) = -1\right) = \frac{1}{2}\left(\frac{1}{2} - \binom{k-1}{\frac{1}{2}(k-1)} \Big/ 2^k\right)$$

$$P\left(f^*(x^l) = -1 \& g_j^*(x^l) = 1\right) = \frac{1}{2}\left(\frac{1}{2} - \binom{k-1}{\frac{1}{2}(k-1)} \Big/ 2^k\right).$$

After few manipulations, this yields

$$q_m = 2\binom{k-1}{\frac{1}{2}(k-1)} \Big/ 2^k q = 2\left[\binom{k-1}{\frac{1}{2}(k-1)} \Big/ 2^k\right]\left(\frac{1}{\sqrt{2\pi}}\right). \tag{24}$$

Finally, we look at the overall generalization rate of the network, $G_T(\alpha)$. Deriving an expression for $G_T(\alpha)$ for an arbitrary k is a difficult task. In the following, we concentrate on the two limiting cases: $k = 3$ and large k. Note that the learning curves for an arbitrary k will lie in between these two limiting cases.

5.1. The case of $k = 3$

For $k = 3$, equation (24) reduces to

$$q_m = \frac{1}{2}\frac{1}{\sqrt{2\pi}}.$$

Putting this value back into equations (21) and (22), we get

$$\overline{R} = \operatorname{erf}\left(\frac{\sqrt{3}}{2}\sqrt{\frac{\alpha}{\pi}}\right) \tag{25}$$

$$G(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\operatorname{erf}\left(\frac{\sqrt{3}}{2}\sqrt{\frac{\alpha}{\pi}}\right)\right). \tag{26}$$

Comparing this to the case of single binary perceptrons (9), we see that three quarters of the training examples contribute to the learning process, and that $G(\alpha)$ still converges exponentially to 1.

The probability that the hypothesis network will classify correctly a new positive (negative) example depends on its target internal representation, more precisely on how many target perceptrons classify this example as positive/negative. For a positive example x, we have two possibilities:

(i) x is classified positive by two target perceptrons, say by g_1^*, g_2^*, and negative by the remaining target perceptron, g_3^*. The hypothesis network can fail to classify this example correctly only if

$$g_1(x) \neq g_1^*(x) \qquad g_2(x) \neq g_2^*(x) \qquad g_3(x) = g_3^*(x)$$

or

$$g_1(x) \neq g_1^*(x) \qquad g_2(x) = g_2^*(x) \qquad g_3(x) = g_3^*(x)$$

or

$$g_1(x) = g_1^*(x) \qquad g_2(x) \neq g_2^*(x) \qquad g_3(x) = g_3^*(x)$$

or

$$g_j(x) \neq g_j^*(x) \qquad \text{for } j = 1, 2, 3.$$

This can happen with a probability

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

(ii) x is classified positive by all three 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

$$G_T(\alpha) = 1 - \tfrac{3}{2}G(\alpha)^2[1 - G(\alpha)] - \tfrac{3}{2}G(\alpha)[1 - G(\alpha)]^2 - [1 - G(\alpha)]^3. \tag{27}$$

The analytical expressions for $\overline{R}$, $G(\alpha)$, and $G_T(\alpha)$ are plotted in figure 4 along with the simulation results. Again, the agreement is excellent. One can also see that $G_T(\alpha)$ tends exponentially to perfect generalization.

The arguments 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 $k \geqslant 7$. Thus, we will look simply at the other end of the spectrum, i.e. large k.

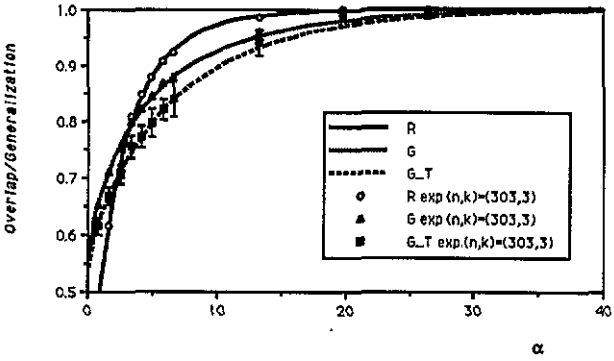

Figure 4. Learning a two-layer network of three non-overlapping binary perceptrons connected to the same number of inputs. The average overlap R, the generalization rate of each perceptron G and the overall generalization rate G_T are shown. The points are the results of the simulations for $n = 303$. Each point denotes an average over 25 different training samples. The error bars, shown only for one curve for clarity, denote the standard deviations.

5.2. *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, equation (24) reduces to

$$q_m \sim \frac{2}{\sqrt{2\pi k}} \frac{1}{\sqrt{2\pi}}. \tag{28}$$

Putting this value back into equations (21) and (22), we get

$$\overline{R} = \operatorname{erf}\left(\sqrt{\frac{2}{\pi}}\sqrt{\frac{\alpha}{\pi}}\right) \tag{29}$$

$$G(\alpha) = 1 - \frac{1}{\pi}\cos^{-1}\left(\operatorname{erf}\left(\sqrt{\frac{2}{\pi}}\sqrt{\frac{\alpha}{\pi}}\right)\right). \tag{30}$$

Again, comparing this to the case of single binary perceptrons (9), we see that a fraction of $2/\pi$ ($\simeq 0.63$) of the training examples contribute to the learning process, and that $G(\alpha)$ still converges exponentially to 1.

To determine the overall generalization, let x be a random input and let

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

For different inputs 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 $Ir^*(x)$ and $Ir(x)$, is given by

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

By definition,

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

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

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

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

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

Then,

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

This yields

$$\overline{\rho} = 2G(\alpha) - 1 = 2(1 - (1/\pi)\cos^{-1}(\overline{R})) - 1 = 1 - (2/\pi)\cos^{-1}(\overline{R}). \quad (31)$$

The overall generalization is then given by

$$G_T(\alpha) = 1 - (1/\pi)\cos^{-1}(\overline{\rho}) = 1 - (1/\pi)\cos^{-1}(2G(\alpha) - 1) \quad (32)$$

$$= 1 - (1/\pi)\cos^{-1}(1 - (2/\pi)\cos^{-1}(\overline{R})). \quad (33)$$

It is interesting to see that, for large k, the generalization rate of a majority of non-overlapping perceptrons behaves like that of a single perceptron, with a modified overlap $1 - (2/\pi)\cos^{-1}(\overline{R})$. Equation (33) has also been derived in [10], using a different method.

The analytical expressions for $G(\alpha)$ and $G_T(\alpha)$ are plotted in figure 5 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.

6. Extension to multilayer networks of non-overlapping binary perceptrons

Let f^* be a layered network of non-overlapping binary perceptrons (figure 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

$$n \to \infty \qquad n/k_1 \to \infty \qquad k_{h-1}/k_h \to \infty \qquad h = 2, \ldots, H.$$

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.

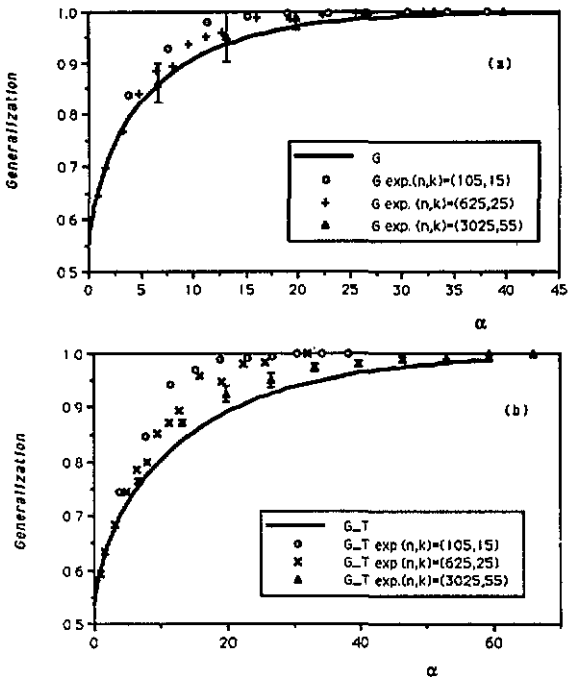

Figure 5. Learning a two-layer network of k non-overlapping binary perceptrons connected to the same number of inputs: the case of large k. The generalization rate of each perceptron G (a) and the overall generalization rate G_T (b) are shown. The points are the results of the simulations for the indicated 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.

Using the arguments in the previous section that led to equation (28), one can show that each hidden layer will contribute a factor $2/\sqrt{2\pi}$ to q_m. Thus,

$$q_m\sqrt{k_1} = \left(\frac{2}{\sqrt{2\pi}}\right)^H \frac{1}{\sqrt{2\pi}}. \tag{34}$$

With this, equation (29) now reads

$$\overline{R} = \operatorname{erf}\left(\left(\sqrt{\frac{2}{\pi}}\right)^H \sqrt{\frac{\alpha}{\pi}}\right). \tag{35}$$

Also, $G_1(\alpha)$ is still given by

$$G_1(\alpha) = 1 - (1/\pi)\cos^{-1}(\overline{R}). \tag{36}$$

Applying the arguments that led to equation (32) recursively, that is from one layer to the next, we get

$$G_h(\alpha) = 1 - (1/\pi)\cos^{-1}(2G_{h-1}(\alpha) - 1) \qquad h = 2, \ldots, H \tag{37}$$

$$G_T(\alpha) = 1 - (1/\pi)\cos^{-1}(2G_H(\alpha) - 1). \tag{38}$$

Finally, we note that equation (35) can be written as

$$\overline{R} = \operatorname{erf}\left(\sqrt{\alpha_{\text{eff}}/\pi}\right) \tag{39}$$

where

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

Compared to the single binary perceptron case (8), 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 [5] that the critical value of α at which the phase transition occurs scales as $(\pi/2)^H$.

7. Conclusion

We have investigated the clipped Hebb rule for learning different networks of non-overlapping binary perceptrons under uniform distribution. We have calculated exactly the learning curves of this rule in the limit $n \to \infty$, where the average behaviour becomes the typical one. Our results indicate that the clipped Hebb rule does indeed learn this class of network. Specifically, the generalization rates converge extremely rapidly, often exponentially, to perfect generalization as a function of the number of training examples. The analytical expressions for the learning curves are in excellent agreement with the numerical simulations.

The generalization abilities of networks of non-overlapping perceptrons with binary weights has been investigated using the statistical mechanics approach [5, 6]. Assuming a stochastic training algorithm that leads, at long times, to a Gibbs distribution of weights, it is found that a phase transition to perfect generalization does occur at a critical value of α [5, 6]. Thus, stochastic training algorithms have a slightly 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.

It is interesting to note that, for networks of non-overlapping binary perceptrons, the expression for the generalization ability of one perceptron in the network is exactly the same as that for the single perceptron except that α is replaced by an α_{eff} that reflects the effective number of examples contributing to the learning process (compare equations (15), (26) and (30) to equation (9)). This effective number will obviously depend on the network architecture, i.e. the number of hidden units and hidden layers, and the function computed at the output node. As long as α_{eff} is not too small compared to α†, we expect the clipped Hebb rule to produce exponentially converging generalization curves.

One serious drawback for the clipped Hebb rule (and the Hebb rule) is the fact that it does not work for networks in which each input may have more than one outgoing connection. The obvious problem in such a situation is that the clipped Hebb rule will assign the same value to all connections coming from the same input, even if they have different values in the target network.

Finally, throughout this paper, we have assumed that the architecture is known in advance. Whereas this is in line with most neural network research, it is hardly justifiable in

† α_{eff} should be of an order greater than $\ln(\alpha)$.

practice. Is there an algorithm that can learn networks of non-overlapping binary perceptrons in terms of finding both the weight values and the network architecture? Note that such an algorithm can still use the clipped Hebb rule to determine the weight values. For some progress in this direction, see [16, 22].

Acknowledgments

This work was supported by NSERC grant OGP0122405. We thank the anonymous referees for their helpful comments. MG would like to thank Sara Solla for helpful suggestions.

References

[1] Barkai E and Kanter I 1991 *Europhys. Lett.* **14** 107–112
[2] Gyorgyi G 1990 *Phys. Rev.* A **41** 7097–100
[3] Köhler H, Diederich S, Kinzel W and Opper M 1990 Z. *Phys.* B **78** 333–42
[4] Krauth W and Mézard M 1989 *J. Physique* **50** 3057–66
[5] Mato G and Parga N 1992 *J. Phys. A: Math. Gen.* **25** 5047–54
[6] Schwarze H and Hertz J 1992 *Europhys. Lett.* **20** 375–80
[7] Venkatesh S 1991 *Proc. 4th Workshop on Computational Learning Theory* (San Mateo, CA: Morgan Kaufman) pp 257–66
[8] Seung H S, Sompolinsky H and Tishby N 1992 *Phys. Rev.* A **45** 6056–91
[9] Timothy L H W and Albrecht R 1992 *Phys. Rev.* A **45** 4102–10
[10] Pitt L and Valiant L G 1988 *J. ACM* **35** 965–84
[11] Golea M and Marchand M 1993 *Neural Comput.* in press
[12] Valiant L G 1984 *Comm. ACM* **27** 1134–42
[13] Blumer A, Ehrenfeucht A, Haussler D and Warmuth K 1989 *J. ACM* **36** 929–65
[14] Pagallo G and Haussler D 1989 A greedy method for learning μDNF functions under the uniform distribution *Technical Report UCSC-CRL-89-12* (Santa Cruz: Department of Computer and Information Science, University of California at Santa Cruz)
[15] Schapire R E 1991 *Proc. 4th Annual Workshop on Computational Learning Theory* (San Mateo, CA: Morgan Kaufman) pp 184–98
[16] Golea M, Marchand M and Hancock T 1992 *Adv. Neural Information Processing Systems* **5** 591–98
[17] Hancock T, Golea M and Marchand M 1993 *Machine Learning* to appear
[18] Meir R and Fontanari J F 1992 *Phys. Rev.* A **45** 8874–84
[19] Vallet F 1989 *Europhys. Lett.* **8** 747–51
[20] Opper M, Kinzel W, Kleinz J and Nehl R 1990 *J. Phys. A: Math. Gen.* **23** L581–6
[21] Opper M and Haussler H 1991 *Phys. Rev. Lett.* **66** 2677–80
[22] Golea M 1993 Average case analysis of an Hebb-type rule that finds the network connectivity submitted