

Training a perceptron by a bit sequence: storage capacity

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

1996 J. Phys. A: Math. Gen. 29 7965

(<http://iopscience.iop.org/0305-4470/29/24/020>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 171.66.16.71

The article was downloaded on 02/06/2010 at 04:07

Please note that [terms and conditions apply](#).

Training a perceptron by a bit sequence: storage capacity

M Schröder[†], W Kinzel[†] and I Kanter[‡]

[†] Institut für Theoretische Physik, Universität Würzburg, Am Hubland, D-97074 Würzburg, Germany

[‡] Department of Physics, Bar-Ilan University, Ramat-Gan 52900, Israel

Received 10 July 1996

Abstract. A perceptron is trained by a random bit sequence. In comparison with the corresponding classification problem, the storage capacity decreases to $\alpha_c = 1.70 \pm 0.02$ due to correlations between input and output bits. The numerical results are supported by a signal-to-noise analysis of Hebbian weights.

1. Introduction

Artificial neural networks are successful in predicting time series (Weigand and Gershenfeld 1993). Given a sequence of real numbers, a multilayer network is able to learn from N consecutive numbers the following one. After learning a part of the sequence, the network is able to generalize. If N consecutive numbers are taken from the part of the sequence which the network has not learned, the network can predict the following number to some extent.

Using methods and models of statistical mechanics, training from a set of examples and generalization of neural networks has been studied intensively (Hertz *et al* 1991, Kinzel and Oppen 1991, Oppen and Kinzel 1996). Most work has been concentrated on perceptrons and binary classification problems. A set of N -dimensional input vectors is classified by a perceptron. A different perceptron is trained by this set of examples; after the training process the network is able to generalize: it has some overlap to the weights of the perceptron which has generated the examples. If the classification is not performed by a different perceptron but is assigned randomly, the network can still learn a certain amount of examples. The maximum number of examples which can be classified by a perceptron is related to the storage capacity to the corresponding attractor networks (Gardner 1988).

Only recently has this approach been extended to time series analysis (Eisenstein *et al* 1995). A perceptron was trained from a series of bits which was produced by a different perceptron. Hence, the generation of time series by a neural network is also interesting in this context and only recently has an analytic solution of a stationary time series generated by a perceptron been found (Kanter *et al* 1995).

It turns out that a perceptron can predict bit sequences very well, if those are taken from stationary time series produced by a different perceptron (Eisenstein *et al* 1995). Already a small training set leads to a perfect prediction of the rest of the sequence, at least for $N \rightarrow \infty$. However, the overlap between a learning and a generating network is very small.

In this paper we study the analogy of the storage-capacity problem in the context of bit sequences. A set of P consecutive bits, which are randomly chosen, is repeated periodically

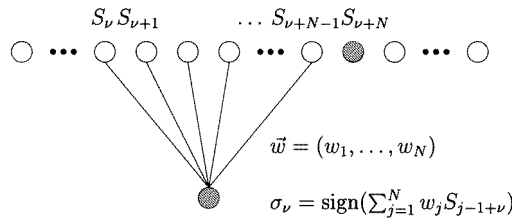


Figure 1. A perceptron learning a periodic time series. The desired output of the perceptron (marked) is the next bit of the series and therefore part of other input patterns as well.

(or placed on a ring). A perceptron with $N < P$ is trained on this bit sequence, where the output bit is given by the bit which follows the N input bit. Hence, the only difference to the examples used for the classification problem are correlations between the input and output. The output bit is contained in the input of N examples.

In section 2 we introduce the bit sequence, which we use for training a perceptron which is simulated in section 3. Section 4 presents a signal-to-noise analysis of the Hebbian learning rule. A general Boolean function is considered in section 5, and section 6 contains a summary and the conclusions.

2. Bit sequence

P bits $S_i \in \{-1, 1\}$; $i = 1, \dots, P$ are chosen randomly and independently. This sequence is repeated periodically from $i = -\infty$ to $i = \infty$ (or placed on a ring, equivalently). N consecutive bits are used as an input to a perceptron with weights $w_j \in \mathbb{R}$; $j = 1, \dots, N$ (see figure 1):

$$\sigma_{\nu} = \text{sign} \sum_{j=1}^N w_j \xi_j^{\nu} \quad \text{with } \xi_j^{\nu} = S_{j-1+\nu}. \quad (1)$$

The problem we are interested in here is the following. Can we find a weight vector $\underline{w} = (w_1, \dots, w_N)$ which reproduces the next bits in the sequence? That is

$$\sigma_{\nu} = S_{\nu+N} \quad \text{for all } \nu \in \mathbb{N}. \quad (2)$$

In particular we are interest in the maximal number $P_c(N)$ of bits which can be reproduced correctly by a perceptron for $N \rightarrow \infty$; as usual we define

$$\alpha = P/N \quad \alpha_c = \lim_{N \rightarrow \infty} \frac{P_c(N)}{N}. \quad (3)$$

There exist mathematical theorems about the number configuration $\{\sigma_{\nu}\}$ which can be realized by (1), which are already more than 140 years old (Schläfli 1852, Cover 1965). If the P input vectors $\underline{\xi}^{\nu} = (\xi_1^{\nu}, \dots, \xi_N^{\nu})$ are in a general position (i.e. if any subset of N vectors is linearly independent), then the number $C(P, N)$ of possible configurations $\{\sigma_{\nu}\} \in \{+1, -1\}^P$ is given by

$$C(P, N) = 2 \sum_{i=0}^{N-1} \binom{P-1}{i}. \quad (4)$$

In our case of the random bit sequence we expect the input vectors $\underline{\xi}^{\nu}$ to be in a general position. For $P \leq N$ one obtains $C(P, N) = 2^P$; hence, any bit sequence with $P \leq N$ can be perfectly predicted by a perceptron. For $P < 2N$ there is still a large fraction of

configurations which is given by (1); this fraction goes to one for $N \rightarrow \infty$. This means that for random configurations $\{\sigma_v\}$ the probability to map them by a perceptron is one in the limit of $N \rightarrow \infty$. For $P > 2N$ this probability is zero. Hence, for a perceptron and random examples one finds $\alpha_c = 2$ (Gardner 1988).

However, in our case the configurations $\{\sigma_v\}$ are not randomly chosen but taken from the input vectors. Each output bit σ_v appears in N input vectors $\underline{\xi}^{v+1}, \dots, \underline{\xi}^{v+N}$ also. There are correlations between the input vectors and the output bits. In addition, only the *fraction* of configuration σ_v which cannot be reproduced by a perceptron goes to zero for $N \rightarrow \infty$ and $N < P < 2N$; their *number* is still increasing exponentially with N . For instance, for $N = 100$ and $\alpha = 1.8$ (4) gives about 10^{54} configurations which are not linearly separable, that is 6.7% of all of the possible 2^{180} ones. Conversely, for $P > 2N$ the number of configurations which can be reproduced by a perceptron still increases exponentially with N , although their fraction disappears. Hence, it is not obvious, whether the patterns given by a bit sequence belong to the first or second class, which means whether $\alpha_c < 2$ or $\alpha_c > 2$.

In the uncorrelated case the storage capacity α_c has been calculated using the replica method (Gardner 1988). Correlations between the input vectors do not change the result $\alpha_c = 2$. Only if there is a bias for the output bits *and* for the input bits the storage capacity α_c increases with the bias. If the patterns are anticorrelated α_c can be lower than $\alpha_c = 2$ also (López *et al* 1995).

Calculating the version space of weights in terms of replicas is hard, since the input and output bits are correlated. Recently some progress has been reported (Saad 1996).

Here, we have studied the bit sequence numerically.

3. Perceptron: simulations

To calculate the storage capacity α_c of the perceptron being trained by a random bit sequence, we have used two methods.

(i) We have used several routines which try to minimize the number of errors and indicate whether they succeeded or not. Hence, we obtained a fraction $f(\alpha, N)$ of patterns for which the routine could find a solution. The capacity $\alpha_c(N)$ is defined by $f(\alpha_c, N) = \frac{1}{2}$. Obviously, we obtain a lower bound for the true α_c , only. The results did not depend on the actual algorithm within the expected error bounds. We have used a routine that minimizes the 'linear cost function' $E = \sum_{v=0}^P \theta(1 - E^v)(1 - E^v)$ with $E^v = \frac{1}{N} \sum_{j=1}^N w_j \xi_j^v \sigma^v$ (without constraining the vector \underline{w}). Preliminary results have been reported by Bork (1994).

(ii) The other estimate uses the median learning time (Priel *et al* 1994). For random patterns the average learning time τ_a of the perceptron algorithm diverges as $\tau_a^{-1/2} \sim (\alpha_c - \alpha)$ for $\alpha \rightarrow \alpha_c$ (Oppen 1988). We use this power law in our case, also. The median τ_m of the distribution of learning times calculated for $\alpha < \alpha_c$ and α_c is obtained from a fit to the power-law divergence. This method has the advantage that one does not have to determine whether a pattern cannot be learned at all. If the number of learning steps is larger than the median the algorithm can stop; this saves a large amount of computer time.

Figure 2 and table 1 show the results of the simulations. In the uncorrelated case both of the methods already give the exact result $\alpha_c = 2$ within the statistical error and for $N = 100$. If we use the input from the bit sequence, but random output bits, the results agree with $\alpha_c = 2$, also. However, if in addition we use the output bits from the bit sequence we obtain $\alpha_c = 1.70 \pm 0.02$. Hence, the correlations between output bits and input vectors decrease the storage capacity. For the perceptron it is harder to learn a random bit sequence than a

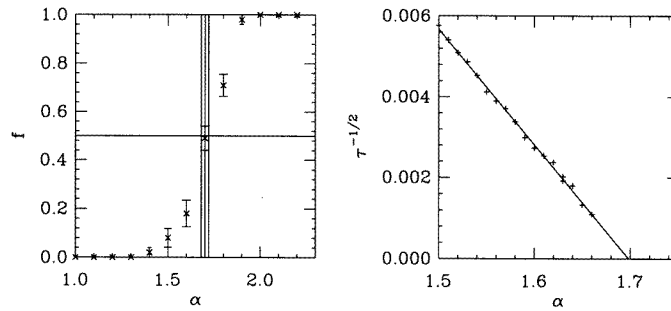


Figure 2. Left-hand side: the probability f of a bit sequence to be not linearly separable as a function of $\alpha = P/N$. The sequence is constructed from P random bits which are repeated periodically. The simulations are performed for a perceptron with $N = 100$ input bit and f is averaged over 50 sets of patterns at least. Right-hand side: the median learning time to the power of $-\frac{1}{2}$ as a function of α . The size of the perceptron is $N = 400$, and τ is averaged over 1000 sets of patterns. The line is a least square fit to the data.

Table 1. The storage capacity of a perceptron learning different tasks. Measured with (method 1) half-error and (method 2) median learning-time method.

	Method 1	Method 2
Random $N = 100$	1.99 ± 0.01	1.995 ± 0.01
Time series $N = 100$	1.85 ± 0.025	1.82 ± 0.02
Time series $N = 400$		1.82 ± 0.02
Ring $N = 100$	1.7 ± 0.025	1.69 ± 0.01
Ring $N = 400$		1.7 ± 0.02
Ring (rnd out) $N = 100$	1.98 ± 0.05	1.99 ± 0.01
Ring (rnd out) $N = 400$		1.98 ± 0.02
Magnetization $m = 0.4$	$N = 100$	
Ring	1.95 ± 0.05	1.95 ± 0.03
Random	2.25 ± 0.05	

random classification problem. This is due to the correlations between the input and output but not due to the correlations between the input vectors.

If a perceptron which has learned a bit sequence perfectly is used as a bit generator, then any initial state of N bits taken from the sequence reproduces the complete sequence. Hence the sequence is an attractor of the bit generator. However, we found that the basin of attraction is very small. If only one bit is flipped in the initial state then there is a high probability that the generator runs into a different sequence.

We have also studied two additional problems.

(i) The P random bits are not repeated periodically but the perceptron is trained with a string of $N + P$ random bits. Hence, there are still P patterns but an output bit belongs only to part of the other input patterns. On average the correlations are weaker. Indeed, we find that the storage capacity $\alpha_c = 1.82 \pm 0.02$ is larger than the one for the periodic sequence.

(ii) With a bias $m = \frac{1}{P} \sum_{i=1}^P S_i$ in the bit sequence, the storage capacity increases. This is similar to the random classification problem (Gardner 1988).

4. Perceptron: Hebbian learning rule

In order to get some insight from analytic calculations we now consider the Hebbian learning rule

$$\underline{w} = \frac{1}{N} \sum_{v=1}^P \sigma_v \underline{\xi}^v. \tag{5}$$

Output bits σ_v and input vectors $\underline{\xi}^v$ are taken from a bit sequence $\{S_i\}$, (1) and (2). It is known that the Hebbian weights cannot map the examples perfectly. However, the training error can be calculated from a signal-to-noise analysis (see for instance Hertz *et al* 1991). The sign of the following stability E^v shows whether an example is classified correctly

$$E^v = \sigma^v \underline{w} \underline{\xi}^v = \frac{1}{N} \sum_{i=1}^N \sum_{\mu=1}^P \sigma^v \sigma^\mu \xi_i^v \xi_i^\mu. \tag{6}$$

The fraction of negative values of E^v defines the training error.

We calculated the first two moments $\langle E \rangle$ and $\langle E^2 \rangle$ of E^v , where $\langle \dots \rangle$ means an average over the distribution of the examples, i.e. over all realizations of the bit sequence. If all bits σ_v and ξ_i^v are random one has

$$\langle \sigma^v \sigma^\mu \rangle = \delta_{v\mu} \quad \langle \xi_i^v \xi_j^\mu \rangle = \delta_{v\mu} \delta_{ij}. \tag{7}$$

This gives

$$\langle E \rangle = 1 \quad \langle E^2 \rangle = 1 + \alpha.$$

In the limit $N \rightarrow \infty$ the values of E^v are Gaussian distributed with mean one and standard deviation $\sqrt{\alpha}$. However, for the periodic bit sequence, (1) and (2), the values of σ^v and ξ_j^v are taken from the random bits S_i . For instance σ^v is identical with ξ_j^μ for $j = 1, \dots, N$ and $\mu = v + N + 1 - j$. Taking this into account we find for $1 < \alpha < 2$:

$$\langle E \rangle = \begin{cases} 1 + \frac{1}{N} & \text{for } P \text{ even} \\ 1 & \text{for } P \text{ odd} \end{cases} \tag{8}$$

$$\langle E^2 \rangle = \begin{cases} 2 + \alpha + \frac{6 - \alpha}{N} - \frac{4}{N^2} & \text{for } P \text{ even} \\ 2 + \alpha - \frac{2}{N} & \text{for } P \text{ odd.} \end{cases}$$

For $\alpha > 2$ the results above for odd P hold for even ones, also. Hence for $N \rightarrow \infty$ the standard deviation of the E^v values is $\sqrt{1 + \alpha}$ instead of $\sqrt{\alpha}$ of the uncorrelated case. The correlations increase the noise relative to the signal. Assuming a Gaussian distribution of the E^v values in the limit $N \rightarrow \infty$, which is supported by our numerical simulation, we obtain the training error ε_t as

$$\varepsilon_t = \phi \left(-\frac{1}{\sqrt{1 + \alpha}} \right) \tag{9}$$

with the error function

$$\phi(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy. \tag{10}$$

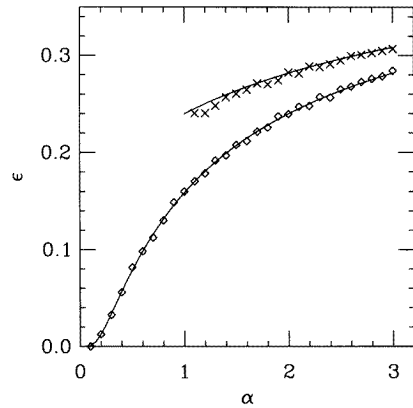


Figure 3. The training error of Hebbian weights for different topologies. The inputs are chosen binary. \diamond ; random patterns and \times ; patterns from a random bit sequence with a periodic boundary condition. The simulations were done for $N = 200$ and averaged over 100 samples each. The curves show the theoretical results.

If the random bits are not repeated periodically, but arranged linearly as discussed above, the moments depend on the number ν of the pattern. If $\nu = 1$ is the first and $\nu = P$ is the last pattern we define

$$\gamma = \begin{cases} \nu/N & \text{for } \nu \leq N \\ 1 & \text{for } \nu > N. \end{cases} \quad (11)$$

In this case the training error depends on γ and we find

$$\varepsilon_t = \phi\left(-\frac{1}{\sqrt{\alpha + \gamma^2}}\right). \quad (12)$$

Figure 3 shows the training error $\varepsilon_t(\alpha)$ for the uncorrelated bits and the periodic bit sequence. In the latter case ε_t is averaged over the patterns. The correlations of the bit sequence increase the training error, in agreement with the decrease of the storage capacity shown in the previous section.

5. General Boolean function

Up to now we have restricted our map to a perceptron. We expect that multilayer networks can reproduce a larger bit sequence, in accordance to the higher storage capacity of the committee machine (Priel *et al* 1994). In this section we study the storage capacity of a general Boolean function $b : \{+1, -1\}^N \rightarrow \{+1, -1\}$, which is the size of the random bit sequence with period P which can be reproduced by any Boolean function b , i.e.

$$b(S_\nu, \dots, S_{\nu+N-1}) = S_{\nu+N} \quad \nu = 1, \dots, P. \quad (13)$$

Since we have the freedom to choose for any input configuration $(S_\nu, \dots, S_{\nu+N-1})$ an arbitrary output bit $S_{\nu+N}$, our problem reduces to the question if all the input configurations are different from each other. If all $(S_\nu, \dots, S_{\nu+N-1})$ are different then we can define a Boolean function which maps each of those states to the corresponding bit $S_{\nu+N}$. For the rest of the $2^N - P$ input states we have the freedom to choose an arbitrary output bit; hence, in this case, there are $2^{(2^N - P)}$ many Boolean functions which map the bit sequence correctly.

If two of the input configurations $(S_\nu, \dots, S_{\nu+N-1})$ are identical there is still a probability of $\frac{1}{2}$ that the two output bits are different, also. To get an analytic estimate for the size of a random bit sequence which can be reproduced by a Boolean function we neglect correlations between the input configurations. That means we consider P configurations

(S_1^v, \dots, S_N^v) ; $v = 1, \dots, P$ where all of the bits S_i^v are chosen randomly and independently. We want to calculate the probability f that all of the P states are pairwise different. There are 2^N many possible states. The first configuration $v = 1$ can be any of those states. The second one can take any of the $2^N - 1$ remaining states, etc. Hence, the number C of allowed configurations is

$$C = 2^N(2^N - 1)(2^N - 2) \cdots (2^N - P + 1) \quad (14)$$

which gives

$$\ln C = \sum_{v=1}^P \ln(2^N - v + 1) = \sum_{v=1}^P \left[N \ln 2 + \ln \left(1 - \frac{v-1}{2^N} \right) \right] \quad (15)$$

$$= PN \ln 2 + \sum_{v=1}^P \ln \left(1 - \frac{v-1}{2^N} \right). \quad (16)$$

If $P \ll 2^N$ we can expand \ln and obtain

$$\ln C \simeq PN \ln 2 - \frac{1}{2^N} \frac{P(P-1)}{2}. \quad (17)$$

Since the total number of all possible configurations is 2^{PN} , the function of the allowed ones is

$$f \simeq \exp \left[-\frac{P(P-1)}{2^{N+1}} \right]. \quad (18)$$

We define the average period P_c by $f(P_c) = \frac{1}{2}$ and obtain for large N

$$P_c = \sqrt{2 \ln 2} 2^{N/2}. \quad (19)$$

Hence we expect that the average length of the bit sequence which can be reproduced by a Boolean function scales as $\sqrt{2^N}$. In fact our problem is similar to the random map, where the average cycle length has the same scaling properties (Harris 1960, Derrida and Flyvbjerg 1987).

The configurations taken from a random bit sequence are correlated, since consecutive configurations are obtained by shifting a window of N bits over the sequence. However, our numerical simulations show that these correlations do not change the scaling law (19). For a given sequence with P bits the size N of the window is increased until this sequence can be reproduced by a Boolean function. N_c is defined as the window size N where 50% of the sequences are reproduced. In figure 4, P is shown as a function of N_c . For $P \leq 17$, N_c is determined by exhaustive enumeration. For larger P values N_c is estimated from up to 10^5 random samples. The log-linear plot shows that the data are consistent with

$$P = 1.6 \times \sqrt{2^{N_c}}. \quad (20)$$

The comparison with (19) shows that the correlations seem to change the prefactor from 1.17 to 1.6, but the number still increases with $\sqrt{2^N}$, the size of the input space.

6. Summary

A perceptron of N input bits has been trained by a random bit sequence with a period P . Each output bit is contained in N input vectors. These correlations decrease the storage capacity to $\alpha_c = 1.7 \pm 0.02$ compared to $\alpha_c = 2$ for uncorrelated output bits. For the corresponding bit generator the bit sequence has a tiny basin of attraction.

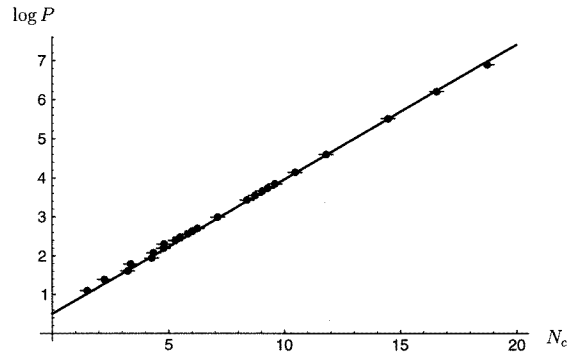


Figure 4. The length of a cycle that is learnable by a Boolean function as a function of N_c . The values up to $P = 17$ are exact. The values up to $P = 100$ are averaged over 100 000 samples, for $P = 251$ over 50 000, for $P = 503$ over 1000 and for $P = 1007$ over 100 samples. The errorbars are given. The line shows $P = \exp(0.5)2^{0.5N_c}$.

An analysis of Hebbian weights shows that a bit sequence gives a larger noise-to-signal ratio than a random classification problem. This result is in agreement with the lower storage capacity.

If a general Boolean function is trained by the random bit sequence, the maximal period P scales as $\sqrt{2^N}$, the size of the input space.

Acknowledgments

This work has been supported by the Deutsche Forschungsgemeinschaft and the MINERVA Center of Physics of the Bar-Ilan University. We thank Georg Reents for valuable discussions.

References

- Bork A 1994 Zeitreihenanalyse *Diploma Thesis* Institut für Theoretische Physik der Universität Würzburg, Würzburg
- Cover T M 1995 *IEEE Trans. Electron. Comput.* **EC-14** 326
- Derrida B and Flyvbjerg 1987 *J. Physique* **48** 971
- Eisenstein E, Kanter I, Kessler D A and Kinzel W 1995 *Phys. Rev. Lett.* **74** 6
- Gardner E 1988 *J. Phys. A: Math. Gen.* **21** 257
- Harris B 1960 *Ann. Math. Stat.* **31** 1045
- Hertz J, Krogh A and Palmer R G 1991 *Introduction to the Theory of Neural Computation* (Redwood City, CA: Addison-Wesley)
- Kanter I, Keeler D A, Priel A and Eisenstein E 1995 *Phys. Rev. Lett.* **75** 2614
- Kinzel W and Oppen M 1991 in *Models of Neural Networks* ed E Domany, J L van Hemmen and K Schulten (Berlin: Springer) p 149
- López B, Schröder M and Oppen M 1995 *J. Phys. A: Math. Gen.* **28** L447
- Oppen M 1988 *Phys. Rev. A* **38** 3824
- Oppen M and Kinzel W 1996 in *Models of Neural Networks* vol III, ed E Domany, J L van Hemmen and K Schulten (New York: Springer) p 151
- Priel A, Blatt M, Grossmann T and Domany E 1994 *Phys. Rev. E* **50** 577
- Schläfli L 1852 Theorie der vielfachen Kontinuität *Gesammelte Mathematische Abhandlungen* ed Steiner-Schläfli-Komitee (Basel: Birkhäuser) p 171
- Saad D 1996 Private communication
- Weigand A S and Gershenfeld N A ed 1993 *Time Series Prediction, Forecasting the Future and Understanding the Past* (Santa Fe, NM: Santa Fe Institute)