International Journal of Neural Systems, Vol. 5, No. 2 (June, 1994) 000–000

Letter to the Editor

A STORAGE ALGORITHM FOR TWO-LAYERED NEURAL NETWORKS

R. MONASSON*
Dipartimento di Fisica, Università di Roma "La Sapienza", 00185 Roma, Italy†

Received 15 March 1994
Accepted 2 May 1994

A learning algorithm for the two-layered committee machine is proposed. The proof of its convergence in a finite time is given. Its efficiency is compared to the simple exhaustive enumeration of the internal representations of the training set.

1. Introduction

One of the main features of the perceptron is the existence of a learning algorithm with a proof of convergence.[1] More precisely, given a training set of hetero-associations (input pattern, corresponding output) to be realised, this algorithm always finds synaptic weights ensuring a correct mapping in a finite time (provided that such couplings exist). It is however well-known that the computational abilities of the perceptron are rather poor (e.g. impossibility of solving nonlinear problems such as the XOR function).[2] One must therefore resort to more complex neural networks, including layers of hidden units (whose values are not specified in the training set) between the input cells and the output.[3] Despite the existence of efficient but non-exact learning algorithms,[3,4] the important question of finding a learning algorithm with a proof of convergence remains open. The key point is the occurrence of a huge number of possible choices for the hidden unit states, which are called the internal representations of the training set. It probably makes learning in a multi-layered network a non-polynomial problem with respect to the size of the training set. In this paper, a storage algorithm for the committee machine[5] is presented in Sec. 2. Its convergence towards a solution in a finite time is proven. Section 3 briefly compares its performances to the simpler approach of an exhaustive search in the space of internal representations.

2. Algorithm

The algorithm described below aims at storing patterns in the so-called fully connected committee machine[5] with one hidden layer: for any input pattern ξ_i^μ $(i = 1 \dots N)$, the network computes the states η_l^μ $(l = 1 \dots K)$ of the hidden units with the usual perceptron-like rule

$$\eta_l^\mu = \mathrm{sign}(\mathbf{J}_l . \boldsymbol{\xi}^\mu) \tag{1}$$

where the vector $\mathbf{J}_l$ denotes the set of the $K \times N$ continuous couplings J_{li} between the hidden neurons η_l and the input cells ξ_i. The output σ^μ corresponds to the most frequent sign among the hidden cells

$$\sigma^\mu = \mathrm{sign}\left(\sum_{l=1}^{K} \eta_l^\mu\right) . \tag{2}$$

In order to simplify the notation, we hereafter fix $K = 3$ (the algorithm may easily be extended to any odd value of K— see Sec. 3). Moreover, the majority rule (1–2) being an odd function of the input pattern, one can impose without any restriction that all the pairs $(\boldsymbol{\xi}^\mu, \sigma^\mu)$, $\mu = 1 \dots P$, which are to be stored

* E-mail: monasson@roma1.infn.it.
† Previous address: Laboratoire de Physique Théorique de l'ENS, 24 rue Lhomond, 75231 Paris Cedex 05, France (unité propre de recherche du CNRS rattachée à l'Université Paris-Sud).

fulfill the condition $\sigma^\mu = +1$. All patterns $\boldsymbol{\xi}^\mu$ are normalised to the unity.

Let t be an integer. We now build a set of 2^t synaptic weights $(\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3)$, each of them being labelled by an ordered sequence of t pairs (number of hidden units, number of patterns) according to the following process:

1. At time $\tau = 0$, let us define one single sequence $\mathcal{S} = \emptyset$ and initialise the corresponding synaptic vectors
$$\mathbf{J}_1^{\mathcal{S}} = \mathbf{J}_2^{\mathcal{S}} = \mathbf{J}_3^{\mathcal{S}} = \mathbf{0}\,.$$
2. At time τ, we have 2^τ sequences of length τ and their corresponding synaptic vectors. Let us assume that none of these 2^τ stores the training set perfectly (otherwise the algorithm has found a solution and stops). Therefore, for every sequence $\mathcal{S}$, there exists at least one pattern $\boldsymbol{\xi}^{\mu_{\mathcal{S}}}$ and two different hidden units $a_{\mathcal{S}}$ and $b_{\mathcal{S}}$ such that
$$\mathbf{J}_{a_{\mathcal{S}}}^{\mathcal{S}}.\boldsymbol{\xi}^{\mu_{\mathcal{S}}} \leq \mathbf{J}_{b_{\mathcal{S}}}^{\mathcal{S}}.\boldsymbol{\xi}^{\mu_{\mathcal{S}}} \leq 0\,.$$
The new synaptic couplings corresponding to the two sequences $\mathcal{S}' = \mathcal{S} + (a_{\mathcal{S}}, \mu_{\mathcal{S}})$ and $\mathcal{S}'' = \mathcal{S} + (b_{\mathcal{S}}, \mu_{\mathcal{S}})$ of length $\tau + 1$ are defined by ($\forall l = 1, 2, 3$)
$$\mathbf{J}_l^{\mathcal{S}'} = \mathbf{J}_l^{\mathcal{S}} + \delta_{l,a_{\mathcal{S}}}\boldsymbol{\xi}^{\mu_{\mathcal{S}}} \ \text{ and } \ \mathbf{J}_l^{\mathcal{S}''} = \mathbf{J}_l^{\mathcal{S}} + \delta_{l,b_{\mathcal{S}}}\boldsymbol{\xi}^{\mu_{\mathcal{S}}}$$
where δ is the Kronecker function. We are now supplied with $2^{\tau+1}$ sequences of length $\tau+1$ and with the $2^{\tau+1}$ corresponding synaptic weights.
3. τ is increased by one and we go back to step 2 if $\tau < t$, stop otherwise.

We can now make the following statement:

If there exists one set of couplings $\mathbf{J}_1^*, \mathbf{J}_2^*, \mathbf{J}_3^*$ (each of the three vectors being normalised to the unity) storing all the patterns with a stability

$$\delta = \min_{1\leq\mu\leq P}[\text{median}(\mathbf{J}_1^*.\boldsymbol{\xi}^\mu, \mathbf{J}_2^*.\boldsymbol{\xi}^\mu, \mathbf{J}_3^*.\boldsymbol{\xi}^\mu)] \qquad (3)$$

strictly positive, then the algorithm described above finds in a finite time a set of couplings storing all the patterns and, more precisely, the length of the corresponding sequence is less than $t_c = E(3/\delta^2)+1$, where E denotes the integer part.

Let us prove this assertion. First, we note that, according to (3), a set of three synaptic vectors $\mathbf{J}_l$, $l = 1, 2, 3$, stores all the patterns if and only if the stability δ is positive. In order to prove that the algorithm is able to find a solution in a finite time, we need $\delta > 0$ (cf. Ref. 1).

For every pattern $\boldsymbol{\xi}^\mu$, we define the set

$$\mathcal{D}_\mu^* = \{l = 1,\, 2,\, 3 | \mathbf{J}_l^*.\boldsymbol{\xi}^\mu \geq \delta\}\,, \qquad (4)$$

for the two or three hidden units l, of the local fields $\mathbf{J}_l^*.\boldsymbol{\xi}^\mu$ of which are larger than the stability δ. Using the previous algorithm, for any integer t, we obtain 2^t sequences $\mathcal{S}$ (and the corresponding synaptic couplings). Among all of them, let us call $\mathcal{R}$ the sequence such that at each stage $\tau \to \tau+1$ (with $\tau < t$) one updates $\mathcal{R}(\tau) = \mathcal{R}$ into $\mathcal{R}(\tau+1) = \mathcal{R}'$ if $a_{\mathcal{R}} \in \mathcal{D}_{\mu_{\mathcal{R}}}^*$ (in this case, one defines $l_{\mathcal{R}} = a_{\mathcal{R}}$) or into $\mathcal{R}(\tau+1) = \mathcal{R}''$ otherwise (defining then $l_{\mathcal{R}} = b_{\mathcal{R}}$). The crucial point is that, in the latter case, we necessarily have $b_{\mathcal{R}} \in \mathcal{D}_{\mu_{\mathcal{R}}}^*$ since $\mathcal{D}_{\mu_{\mathcal{R}}}^*$ contains at least two distinct elements! Thus, in both cases, $l_{\mathcal{R}} \in \mathcal{D}_{\mu_{\mathcal{R}}}^*$: at each time step, the perceptron $l_{\mathcal{R}}$ is updated in the "right" direction.

The rest of the proof is straightforward. Let us suppose that the algorithm has not stopped before $\tau = t$. Defining the three ratios

$$c_l(\tau) = \frac{\mathbf{J}_l^{\mathcal{R}(\tau)}.\mathbf{J}_l^*}{|\mathbf{J}_l^{\mathcal{R}(\tau)}|}$$

(for instance $c_l(\tau) = 1$ if $\mathbf{J}_l^{\mathcal{R}(\tau)} = \mathbf{0}$), one may easily derive some lower bounds for the c_l's. Between times τ and $\tau+1$, the two functions c_l with $l \neq l_{\mathcal{R}}$ remain of course unchanged. Using (4), we immediately obtain for $l = l_{\mathcal{R}}$

$$\begin{aligned}\mathbf{J}_l^{\mathcal{R}(\tau+1)}.\mathbf{J}_l^* &= \mathbf{J}_l^{\mathcal{R}(\tau)}.\mathbf{J}_l^* + \mathbf{J}_l^*.\boldsymbol{\xi}^{\mu_{\mathcal{R}}} \\ &\geq \mathbf{J}_l^{\mathcal{R}(\tau)}.\mathbf{J}_l^* + \delta \geq \delta n_l(\tau+1) \\ |\mathbf{J}_l^{\mathcal{R}(\tau+1)}|^2 &= |\mathbf{J}_l^{\mathcal{R}(\tau)}|^2 + 2\mathbf{J}_l^{\mathcal{R}(\tau)}.\boldsymbol{\xi}^{\mu_{\mathcal{R}}} + |\boldsymbol{\xi}^{\mu_{\mathcal{R}}}|^2 \\ &\leq |\mathbf{J}_l^{\mathcal{R}(\tau)}|^2 + 1 \leq n_l(\tau+1)\end{aligned}$$

where $n_l(\tau)$ is the number of updatings of the perceptron numbered l during the first τ time steps. We therefore deduce that $c_l(\tau) > \delta\sqrt{n_l(\tau)}$ $(1 \leq l \leq 3)$ and, according to the bounds $|c_l(t)| \leq 1$, that

$$t = \sum_{l=1}^{3} n_l(t) \leq \frac{3}{\delta^2}\,.$$

This proves that the algorithm must necessarily stop at a time t lower than t_c. Therefore, among all the 2^{t_c} sets of synaptic vectors generated by the algorithm, one of them at least perfectly stores the whole training set.

3. Discussion

It is quite clear that the time needed to ensure the convergence to a solution scales exponentially with the size P of the training set. This may be understood as the necessity for the algorithm to "see" at least once each pattern and thus $t_c \propto P$ (this rough argument is indeed too strong: only a finite fraction of the P patterns have the lowest stability δ but the previous scaling of t_c remains correct). In other terms, when the number N of input neurons becomes large, the typical stability δ of the training set scales as $\frac{1}{\sqrt{N}}$ which, according to the linear relation $P = 3N\alpha$, gives back $t_c \propto P$. Defining Δ as the maximum stability $\delta\sqrt{N}$ over the solutions $\mathbf{J}_l^*$, one sees that the maximal number of operations (i.e. updatings) required by the algorithm is

$$t_1 = e^{Na_1}\, ; \quad a_1 = \frac{3\ln 2}{\Delta^2}\, .$$

Therefore, the situation appears hopeless close to the critical capacity α_c since the optimal stability Δ achievable by the network vanishes. The performance of the algorithm is however much better for small sizes α of the training set, i.e. when Δ is large.

One may wish to compare this result to the performance of a simple exhaustive scanning of all possible internal representations of the training set. For every pattern to be mapped onto $\sigma = +1$, there are four correct choices of internal representations η_l : $(+ + +)$, $(- + +)$, $(+ - +)$ and $(+ + -)$. Let us suppose that the training set is stored with an adequate choice of the weights $\mathbf{J}_l^*$. Among the 4^P possible sets $\{\eta_l^\mu\}$ of internal representations of the $\{\boldsymbol{\xi}^\mu\}$'s, one of them at least is thus linearly separable by the three perceptrons (e.g. $\eta_l^\mu = \mathrm{sign}(\mathbf{J}_l^*.\boldsymbol{\xi}^\mu)$). The learning procedure is as follows. One starts from a given set of internal representations, $\{\eta_l^\mu = +1\}$ for instance. One then trains in parallel the three perceptrons $\mathbf{J}_l$ so as to map the input patterns $\boldsymbol{\xi}^\mu$ onto these corresponding representations η_l^μ. If the task is not achieved after a time (number of updatings per perceptron) t_p, a new set of P internal representations is drawn and so on. If we choose the halting criterion such that $t_p = 1/\varepsilon^2$ where

$$\varepsilon = \max_{\{\mathbf{J}_l^*\}} \left\{ \min_{1\le l\le 3} \left[\min_{1\le \mu\le P} |\mathbf{J}_l^*.\boldsymbol{\xi}^\mu| \right] \right\}$$

is the largest local stability over all solutions $\mathbf{J}_l^*$ (as compared to δ which represents a global stability for the three hidden units), we are sure that a set of synaptic vectors storing all input patterns will be found through the exhaustive enumeration of all possible internal representations. In other words, if this algorithm has been running for a time (total number of updatings) $3t \times 4^P$ without having found any solution, we can affirm that there are no weights $\mathbf{J}_l^*$ storing the training set with a local stability larger or equal to $1/\sqrt{t}$. Cover's computation[6] teaches us that when $P < N$ there is no need for such an exhaustive enumeration of the internal representations: any of them is always implementable (provided that the input pattern are in general position). We can therefore bound the time required to find a solution (provided that there exist some weights $\mathbf{J}_l^*$ storing the patterns with a local stability ε) by

$$t_2 = \frac{3}{\varepsilon^2} e^{Na_2}\, ; \quad a_2 = \begin{cases} 0 & \text{if } \alpha < \dfrac{1}{3} \\ 6\alpha \ln 2 & \text{if } \alpha > \dfrac{1}{3} \end{cases}$$

(this bound t_2 can be improved, especially in the range $N < P < 2N$).

That learning procedure presents however a major drawback. Clearly, ε is not physically relevant for the storage in a committee machine. One can indeed encounter the following situation: ε can be very small and, at the same time, δ (obtained with the synaptic weights optimising ε) large. Then there exists a pattern $\boldsymbol{\xi}^\mu$ for which at least two local fields $\mathbf{J}_l^*.\boldsymbol{\xi}^\mu$ are positive and large ($\geq \delta$) while the third local field is very small $\mathbf{J}_{l_0}^*.\boldsymbol{\xi}^\mu = \pm\varepsilon$. Hence this pattern is correctly stored by the optimal configuration of the weights and is stable : slight modifications of any of the $\mathbf{J}_l$ cannot affect the output since the sign of η_{l_0} does not matter. However, training the perceptron $\mathbf{J}_{l_0}$ requires a large number of updatings, which diverges when $\varepsilon \to 0$. To avoid this problem, one cannot perform independent trainings of the input perceptrons. In any "physical" storage algorithm, the only source of learning time divergence should be the vanishing of the optimal stability Δ.

The above algorithm may be generalised to any two-layered committee machine including $K = 2k + 1$ hidden units. One finds that the upper bound over the sequence length becomes $t_c = E((2k+1)/\delta^2) + 1$ and the number of such sequences is $(k+1)^{t_c}$. It would be interesting to see whether the algorithm presented here could be extended to other architectures, e.g. the parity machine. Moreover, it would be worth knowing how its speed could be improved by the use of more efficient perceptron algorithms, as for instance the so-called Adatron.[7]

Acknowledgments

I am deeply grateful to N. Parga for discussions concerning this work. I also thank N. Brunel, M. Mézard and J. P. Nadal for their remarks and suggestions.

References

1. F. Rosenblatt, *Principles of Neurodynamics*, (New York: Spartan, 1962).
2. M. L. Minsky and S. A. Papert, *Perceptrons* (Cambridge MA: MIT Press, 1969).
3. J. Hertz, A. Krogh and R. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, 1991), and references therein.
4. D. E. Rumelhart and J. L. McClelland, *Parallel Dristributed Processing* (Cambridge MA: MIT Press, 1986); T. Grossman, R. Meir and E. Domany, *Complex Systems* **2** (1988), 555.
5. E. Barkai, D. Hansel and H. Sompolinsky, *Phys. Rev. A* **45** (1992), 4146; A. Engel, H. M. Kohler, F. Tschepke, H. Vollmayr and A. Zippelius, *Phys. Rev. A* **45** (1992), 7590; G. Mato, C. Mourkazel and N. Parga, *Maximum capacity of a double-layered neural network*, Bariloche preprint (1991).
6. T. Cover, *IEEE Trans. Electron. Comput.* **14** (1965), 326.
7. J. K. Anlauf and M. Biehl, *Europhys. Lett.* **10** (1989), 687.