SI Text

1 The belief propagation (BP) algorithm

Until recently, no efficient algorithm was known to solve the random classification problem for binary weights with an extensive number of patterns ($p \propto N$). Recently, [2] applied a BP algorithm to the binary perceptron and showed that it can find a solution up to a value of $\alpha \approx 0.7$. The algorithms presented in this work have been derived from the BP algorithm. Hence, for the sake of completeness, we give here a brief description of the BP algorithm applied to the ± 1 perceptron problem.

Consider the set $\mathcal{W}^*$ of all the (unknown) synaptic weights vectors which properly implement the input/output mapping of the patterns. A uniform sampling of this set defines a probability space space over the set $\mathcal{W}^*$ of all perfect classifiers. Over this space we are interested in single marginals, that is in the probabilities

$$p_i^{\pm} = P_{\mathcal{W}^*}\left(w_i = \pm 1\right) = \left|\{\mathbf{w} \in \mathcal{W}^* : w_i = \pm 1\}\right| / \left|\mathcal{W}^*\right| \tag{1}$$

that the single synapses take a certain binary value in a randomly chosen solution (here $|\cdot|$ denotes number of element of a finite set).

The computation of these marginals constitutes the first step in the process of finding the optimal synaptic weights, as in other optimization problems to which a similar strategy has been applied with success [3, 5]. Once the marginals are computed, one typically proceeds iteratively by fixing the synaptic weights accordingly.

In this scheme, the learning problem is thus converted in the problem of computing marginals, which unfortunately in the worst-case is a procedure taking exponential time in the number of synapses. However, under some weak correlations assumption, it is possible to write a closed set of equations for the marginals which can be solved efficiently by iteration. This procedure is well known in physics under the name of Bethe-Peierls approximation or cavity method (where it is used to evaluate the equilibrium properties of mean-field models) and in information theory as BP [6].

In turn, the iteration scheme can be implemented as a distributed computation, a fact which opens the possibility of implementing a dynamical scheme governed by local rules which actually solves the equation and hence provides the marginals we are interested in. This feature is what will allow us to identify versions of BP which are simple enough to be considered of potential biological interest, in the spirit of other algorithms which

have been recently proposed [7].

For simplicity, just as in the main text, we will assume $\Xi_- = \emptyset$, without loss of generality. The BP approach consists first in finding the marginal probabilities for synaptic weights w_i on perfect classifiers of restricted problems:

- $p_{i\to a}^{w_i}$ is the probability for synaptic weight w_i for a weight vector that satisfies a restricted classification problem in which variable i does not participate in classification of pattern a,

$$p_{i\to a}^{w_i} = P_{\mathcal{W}_{i\to a}^*}(w_i)$$

where $P_{\mathcal{W}}$ is the uniform measure over $\mathcal{W}$ and

$$\mathcal{W}_{i\to a}^* = \left\{ \mathbf{w} \in \mathcal{W} : \sum_j w_j \xi_j^b > 0 \, \forall b \neq a, \sum_{j\neq i} w_j \xi_j^a > 0 \right\}$$

- $\eta_{a\to i}^{w_i}$ is the probability for synaptic weight w_i for a weight vector that satisfies a restricted classification problem in which variable i participates in classification of pattern a only,

$$\eta_{a\to i}^{w_i} = P_{\mathcal{W}_{a\to i}^*}(w_i)$$

where again P is defined as in Eq.1 and

$$\mathcal{W}_{a\to i}^* = \left\{ \mathbf{w} \in \mathcal{W} : \sum_{j\neq i} w_j \xi_j^b > 0 \, \forall b \neq a, \sum_j w_j \xi_j^a > 0 \right\}$$

BP equations for these variables read [2]:

$$\eta_{a\to i}^{w_i} \propto \sum_{\{w_j : j\neq i\}} \prod_{j\neq i} p_{j\to a}^{w_j} \Theta\left[\sum_j w_j \xi_j^a \right] \tag{2}$$

$$p_{i\to a}^{w_i} \propto \prod_{b\neq a} \eta_{b\to i}^{w_i} \tag{3}$$

where $\Theta[x]$ denotes the Heaviside function ($\Theta[x] = 1$ if $x \geq 0$, $\Theta[x] = 0$ otherwise), i, j indices run over $1, \ldots, N$ and a, b are pattern indices. The $\propto$ symbol indicates normalization prefactors that ensure $\eta_{a\to i}^{+} + \eta_{a\to i}^{-} = 1$ and $p_{i\to a}^{+} + p_{i\to a}^{-} = 1$.

On a solution of Eqs. 2-3, BP estimation of marginals in Eq. 1 can be computed as:

$$p_i^{w_i} \propto \prod_a \eta_{a\to i}^{w_i} \tag{4}$$

The standard way to solve Eqs. 2-3 is by iteration. Call $S = (\{\eta_{a\to i}\}, \{p_{a\to i}\})_{a,i}$ and consider $f : S \mapsto f(S)$ the function defined by right-hand sides of Eqs. 2-3. Build the sequence S_t as the iteration $S_t = f^{(t)}(S_0)$ from some initial condition S_0 (e.g. random or uniform), until the distance of two consecutive terms $\|S_{t+1} - S_t\|$ is zero or small enough. Then evaluate Eq. 4.

A modified version of the equations with a reinforcement term introduced in [2] replacing the right-hand term in Equation 3 by $p_i^{w_i} \prod_{b\neq a} \eta_{b\to i}^{w_i}$ drive the system to converge to a polarized solution in which $p_i^{w_i}$ is either 1 or 0, i.e. a single configuration. With weak correlation assumptions, the reinforced equations in terms of $h = \tanh^{-1}(p^+ - p^-)$ and $u = \eta^+ - \eta^-$ become in a leading order approximation (see full derivation in [2]):

$$h_i^{t+1} = \sum_{t'\leq t} \sum_b u_{b\to i}^t \tag{5}$$

$$m_i^{t+1} = \tanh\left(h_i^{t+1}\right) \tag{6}$$

$$u_{a\to i}^t = \frac{1}{\sqrt{N}} f\left(\frac{1}{\sqrt{N}} \sum_{j\neq i} \xi_j^a m_j^t, \frac{1}{N} \sum_{j\neq i} \left(1 - (m_j^t)^2\right)\right) \tag{7}$$

where

$$f(a,b) = \frac{1}{\sqrt{b}} \frac{G\left(\frac{a}{\sqrt{b}}\right)}{H\left(-\frac{a}{\sqrt{b}}\right)} \tag{8}$$

and the auxiliary functions G and H are defined by:

$$G(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{9}$$

$$H(x) = \int_x^\infty G(y)\, dy \tag{10}$$

The idea is that in the course of the learning process the 'hidden variables' h_i go progressively towards large positive or negative values, and hence variables m_i become closer and closer to $+1$ or -1. Hence, at the end of the learning process, the synaptic weights can be set to the sign of m_i.

These equations can describe an on-line learning protocol by switching to asynchronous update, choosing a time scale τ defined by $n\alpha\tau = t$, and picking randomly at time τ a pattern ξ^τ from the set Ξ, giving:

$$m_i^\tau = \tanh(h_i^\tau) \tag{11}$$

$$h_i^{\tau+1} = h_i^\tau + \frac{\xi_i^\tau}{\sqrt{N}} f\left(\frac{1}{\sqrt{N}}\sum_{j\neq i}\xi_j^\tau m_j^\tau, \frac{1}{N}\sum_{j\neq i}\left(1-(m_j^\tau)^2\right)\right) \quad (12)$$

This (on-line) algorithm is fast and solves the learning process up to large values of α [2], but it has a number of unrealistic features

- Each synapse needs to keep a memory of two analog variables: m_i and h_i;
- It is unclear what could be the neurobiological implementation of the two arguments of the function f.

This algorithm can be simplified in order to get rid of such features, while keeping a high capacity and fast convergence. This is done heuristically by crudely simplifying f as a Heaviside step function of the first argument, and replace the tanh with a sign function:

$$\begin{aligned} m_i^\tau &= \text{sign}\left(h_i^\tau\right) \\ h_i^{\tau+1} &= h_i^\tau + \xi_i^\tau \Theta\left[-\sum_{j\neq i}\xi_j^\tau m_j^\tau\right] \end{aligned}$$

where we removed two inessential factors $N^{-\frac{1}{2}}$. As a last step, we identify the m_i fields with the synaptic weights w_i and avoid the ambiguous $h_i = 0$ case by initializing all the h_i's to odd values and introducing a factor 2 in their update term, so that they can only assume odd values:

$$\begin{aligned} h_i^{\tau+1} &= h_i^\tau + 2\xi_i^\tau \Theta\left[-\sum_{j\neq i}\xi_j^\tau w_j^\tau\right] \\ w_i^{\tau+1} &= \text{sign}\left(h_i^{\tau+1}\right) \end{aligned}$$

In this way, we are left with a single discrete variable for each synapse, and the updating signal is much simpler.

2 BP on 0,1 perceptron

The derivation of the **BPI** algorithm for the 0,1 perceptron from the BP scheme closely follows the track of the previous section. The main differences are that in this case the threshold θ is different from 0 and that we need to consider patterns belonging to both the sets Ξ_+ and Ξ_-.

The 0,1 perceptron model is defined in the main text; here we recall that we consider each pattern to have on average a fraction f of active inputs ($\xi_i^a = 1$), and the same fraction of active outputs ($\sigma^a = 1$). We refer to f as the 'coding level'. Both the maximal capacity and the optimal threshold depend on this parameter; we will postpone this issue to the end of this section, the following derivation being general.

The on-line BP-inspired equations (equivalent to Eqs. 11-12 for the ± 1 perceptron) are:

$$
\begin{aligned}
m_i^\tau &= \tanh\left(h_i^\tau\right) && (13)\\
h_i^{\tau+1} &= h_i^\tau + \xi_i^\tau f_{01}\left(\sigma^\tau, \frac{1}{2}\sum_{j\neq i}\xi_j^\tau\left(1+m_j^\tau\right), \frac{1}{4}\sum_{j\neq i}\xi_j^\tau\left(1-(m_j^\tau)^2\right)\right) && (14)\\
w_i^{\tau+1} &= \frac{1}{2}\left(\mathrm{sign}\left(h_i^{\tau+1}\right)+1\right) && (15)
\end{aligned}
$$

where

$$
f_{01}\left(\sigma, a, b\right) = \frac{1}{2\sqrt{b}}\left(\frac{G\left(\left(a+\sigma-\theta\right)/\sqrt{b}\right)}{\sigma-\left(2\sigma-1\right)H\left(\left(a+\sigma-\theta\right)/\sqrt{b}\right)}\right) \qquad (16)
$$

As before, we only need to keep the internal variables h_i, updating them at each time step upon presentation of a pattern (ξ^τ, σ^τ).

Once again, Eqs. 13-16 can be discretized in a crude way by substituting m_i by its sign and the function f_{01} by a step function, so that the internal hidden variables h_i can only take integer values; we further restrict them to take odd values, and the equations become:

$$
\begin{aligned}
h_i^{\tau+1} &= h_i^\tau + 2\xi_i^\tau\left(2\sigma^\tau-1\right)\Theta\left[-\left(2\sigma^\tau-1\right)\left(\sum_{j\neq i}\xi_j^\tau w_j^\tau-\theta\right)\right]\\
w_i^{\tau+1} &= \frac{1}{2}\left(\mathrm{sign}\left(h_i^{\tau+1}\right)+1\right)
\end{aligned}
$$

We now resort to the issue of optimally choosing the threshold θ. This can be computed by means of the replica method, as done in [8]. In the dense coding case $f = 0.5$, the maximal theoretical capacity is $\alpha_{\max} \approx 0.59$, which can be obtained by optimally setting the threshold as $\theta \approx 0.16N$. For lower values of α, the optimal threshold (in terms of the number of solutions to the learning problem) is higher, and reaches $0.25N$ at $\alpha = 0$. However, the algorithm performance is not much affected by the value of θ,

and setting it to the value correnspondingto $\alpha_{\max}$ has proven to be optimal even at lower values of α, with respect to both capacity and convergence time.

When varying f the picture is similar; furthermore, the ratio between the optimal value of θ (taken at $\alpha_{\max}$) with respect to the average number of active inputs in each pattern fN is almost constant, going from 0.32 for $f = 0.5$ to 0.30 for $f = 3 \cdot 10^{-3}$. Thus, with these settings, about 30% of the synapses will be active after learning in all cases.

3 Binary vs K state synapses

In order to make the problem of learning with binary synapses tractable, we ended up 'hiding' a multi-state variable inside each synapse. This raises naturally the question of the practical usefulness of such a device: from the architectural point of view, it may be questionable wether it is better to use a binary device with K hidden states than one with K visible states; in fact, the latter has a greater theoretical capacity. However, the **BPI** algorithms can be superior either when the learning phase and the recalling phase are totally distinct or in presence of noise or unreliable devices.

The hidden variables are only necessary during learning; thus, the overhead required for storing and managing the hidden variables may be limited to that period. Note that this was already possible using the original **BP** algorithm, but the **BPI** version is both faster (it has a better scaling with the dimentions of the problem N) and much easier to implement. In an on-line setting, in which learning has to occur in real time, noise resistance is the primary reason for using binary synapses; this issue is discussed in section 4.4.

Interestingly, the rule set we propose for **BPI** becomes useful even when using a device with a limited number of visible states K: in this case, the learning problem rapidly becomes hard from the algorithmic point of view as N gets large. The binary case is the extreme example of this situation; as we have shown in Fig. 3 in the main text, the **SP** algorithm may perform worse than the **SBPI** algorithm with the same number of states in such a situation. Since the capacity of the visible-state device has to be greater than that of the binary device, the reduced efficiency is due to the **SP** algorithm. We found that some efficiency could be recovered by using a modified version of this algorithm, in which an analog of the rule R2 for **BPI** was added. The modified perceptron algorithm **MP** was defined as:

Compute $I = \xi^\tau \cdot w^\tau$, then

(R1) If $I > \theta_m$, do nothing

(R2) If $0 < I <= \theta_m$ then:

a) If $w_i^\tau \xi_i^\tau \geq 1$, then $w_i^{\tau+1} = w_i^\tau + 2\xi_i^\tau$

b) Else do nothing

(R3) If $I \leq 0$ then $w_i^{\tau+1} = w_i^\tau + 2\xi_i^\tau$.

This is very similar to the **BPI** algorithm, in which the h_i's are replaced by the w_i's and the secondary threshold is $\theta_m \neq 1$. The **SP** algorithm can be recovered by setting $\theta_m = 0$.

Fig. 5 shows that both convergence speed and storage capacity are higher with **MP** $\neq 0$; the optimal value is different for different tasks (cfr. Fig. 5 A and B), and has a strong dependence on the number of states K (not shown). With the proper settings, this algorithm reaches slightly higher capacities than **BPI** even with very few states K compared to the number of synapses N, though being still more sensitive to noise (see section 4.4).

4 Simulation results

In this section, we report evidence for some results cited in the main text.

4.1 Optimal value of the parameter p_s

As cited in the main text, we found that there is a tradeoff between the maximal capacity which can be achieved by the **SBPI** algorithm and the convergence speed, depending on the value of the parameter p_s, i.e. depending on the probability of applying rule R2 when the classification of a pattern is just barely correct. We defined the optimal value of p_s to be the one which minimizes the average number of presentations per pattern required for learning at a given α, and performed some tests using 10 samples at $N = 20001$, varying α by steps of 0.05 and p_s by steps of 0.1; as can be seen in Fig. 6, we found an almost linear relation.

4.2 Distribution of hidden states

Fig. 7 A shows the final distribution histogram of the hidden variables h_i for one sample with $N = 64001$ after learning with $\alpha = 0.3$, for the **BPI** algorithm. When the number

of allowed states is infinite, the distribution has the shape of two bell-like curves. The width of the distribution is proportional to $\sqrt{N}$, as shown in Fig. 7 B. A $\sqrt{N}$ scaling comes naturally from an unsupervised application of rule R3 (i.e. applying it regardless of whether an error is made or not) with a set of αN random patterns. However, an unsupervised learning rule would lead to a Gaussian distribution centered on 0. Supervision (leading to applying rules R2 and R3 conditioned by the value of the total input I) leads to the polarization of the distribution around two symmetric peaks at finite values of h, but leaves the $\sqrt{N}$ scaling unchanged. Introducing an upper and lower bound on h leads to the appearance of two peaks in the distributions at these bounds. These bounds stop the synapses that would otherwise tend to go to very large positive or negative values. If the bounds are large enough, this has no adverse effect on learning because those synapses that reach such large values of h never change sign during the learning process. Reducing further the number of states starts to affect the shape of the whole distribution when the value of the bounds becomes smaller than the location of the peaks of the distribution in the unbounded case. At this point the whole distribution changes, and the convergence time starts to change compared to the unbounded case.

4.3 Optimal value of the number of hidden states K

In order to determine the optimal number of internal states K for a given number of synapses N, we performed some test with N ranging from 1001 to 32001 and looked for the value of K which maximized capacity. Fig. 8 shows that the optimal number of internal states K scales roughly like $\sqrt{N}$, both in the ± 1 and in the 0,1 scenarios.

4.4 Robustness against noise

In this section, we focus on the robustness of the binary perceptron with respect to noise wich might affect the hidden states. We tested two different situations: one in which noise is added during the learning process and afterwards, and another one in which it is only applied after learning has occurred. The first setting mimics the situation in which the multi-stable elements representing the internal states are not reliable on the learning time scale; the latter represents a situation in which learning sessions occur on much faster time-scales compared to the time during which the stored memories have to be available for recalling.

We compared a binary device with hidden states (implementing **SBPI**) with a per-

ceptron with visible states implementing a standard perceptron algorithm **SP** and the modified version described in section 3, **MP**. For proper comparison, all of these devices had the same number of synapses $N = 4001$ and the same overall number of stable states (K hidden states of **BPI** were compared to K visible states of the standard perceptron). The optimal value (the one maximizing robustness) of the secondary threshold for the **MP** algorithm was found to be $\theta_m \approx 30$ for the bounded case $K = 100$ and $\theta_m \approx 180$ for the unbounded case.

Protocol 1. We added gaussian noise to the multi-stable states during the learning process, once after each presentation of the whole pattern set. The process was carried on even after perfect learning was eventually achieved. We generated random numbers according to a normal distribution with standard deviation z, truncated them towards 0, doubled them and added them to the states value (truncation is needed in order to keep the state values integer, doubling to keep them odd). Thus, using $z = 1$ for example, each synapse had a 68% probability of staying unchanged, a 28% probability of making one step upwards or downwards, etc. Each run consisted in 10000 presentations per pattern; as a measure of robustness, we averaged the number of errors made by each device in the last 1000 presentations, a time at which it has reached its asymptotic value. The results are shown in Fig. 9 A-B. The binary device shows a higher resistance to noise: even at the lowest noise level, $z = 1$, the K-visible state device was unable to keep the error rate to 0.

Protocol 2. Each simulation was divided into a short learning period (200 presentations per pattern) and a longer recalling period during which noise was applied and memories were tested without any further learning. The protocol for noise application was the following: at each iteration, each synapse had a fixed probability $p_Z = 0.1$ to switch one state up or down with equal probability. After each iteration, the whole pattern set was probed and the corresponding number of errors recorded. Note that the time scale of the recalling period is arbitrary with respect to that of the learning period. Results are shown in Fig. 9 C-D. We found that the binary device with K hidden states was remarkably more robust than the K-visible state device, especially at short times. Of course, in the limit of very long times all three rules perform equally badly, since all memory of the stored patterns is erased, but at any finite time the system with binary synapses is significantly better.

References

[1] G. Parisi, M. Mezard, and R. Zecchina (2002) *Science* **297**, 812-815

[2] A. Braunstein and R. Zecchina. Learning by message-passing in networks of discrete synapses. *Phys. Rev. Lett.*, 96:030201, 2006.

[3] A. Braunstein, R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina. Polynomial iterative algorithms for coloring and analyzing random graphs. *Phys. Rev. E*, 68:036702, 2003.

[4] M. Mezard and R. Zecchina (2002) *Phys. Rev. E* **66**, 056126

[5] A. Braunstein, M. Mezard and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Structures and Algorithms*, 27:201–226, 2005.

[6] David Mackay. *Information Theory, Inference, and Learning Algorithms*. Combridge University Press, 2003.

[7] S. Fusi, P. J. Drew, and L. F. Abbott. Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611, Feb 2005.

[8] H. Gutfreund and Y. Stein Capacity of neural networks with discrete synaptic couplings. *J. Phys. A: Math. Gen.*, 23:2613-2630, 1990.