Unreasonable Effectiveness of Learning Neural Networks: From Accessible States and Robust Ensembles to Basic Algorithmic Schemes – Supporting Information

Carlo Baldassi,[1,2] Christian Borgs,[3] Jennifer Chayes,[3] Alessandro Ingrosso,[1,2] Carlo Lucibello,[1,2] Luca Saglietti,[1,2] and Riccardo Zecchina[1,2,4]

[1] *Dept. Applied Science and Technology, Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy*
[2] *Human Genetics Foundation-Torino, Via Nizza 52, I-10126 Torino, Italy*
[3] *Microsoft Research, Cambridge, MA, USA*
[4] *Collegio Carlo Alberto, Via Real Collegio 30, I-10024 Moncalieri, Italy*

CONTENTS

S1. INTRODUCTION AND NOTATION

This Supporting Information text contains all the technical details of the algorithms described in the main text, the techniques and the parameters we used to obtain the results we reported. We also report some additional results and report other minor technical considerations.

Preliminarily, we set a notation used throughout this document which is slightly different from the one of the main text, but more suitable for this technical description.

A. The network model

As described in the main text, we consider an ensemble of y neural networks with K units and binary variables $W_i^{ka} \in \{-1, 1\}$ where $k \in \{1, \dots, K\}$ is the unit index, $i \in \{1, \dots, {}^{N}/_{K}\}$ is the synaptic index and $a \in \{1, \dots, y\}$ is the replica index. Each network has thus N synapses, where N is divisible by K. For simplicity, we assume both K and ${}^{N}/_{K}$ to be odd. The output of each unit is defined by a function $\tau(\xi; W) = \text{sign}\left(\sum_{i=1}^{N/K} W_i \xi_i\right)$. The output of the network is defined by a function $\zeta\left(\left\{\xi^k\right\}_k; \left\{W^k\right\}_k\right) = \text{sign}\left(\sum_{k=1}^{K} \tau\left(\xi^k; W^k\right)\right)$ where ξ^k represents the input to the k-th unit. In the case $K = 1$, this is equivalent to a single-layer network (also known as perceptron). In the case where all ξ^k are identical for each k, this is equivalent to a fully-connected two-layer network (also known as committee machine or consensus machine). If the ξ^k are different for different values of k, this is a tree-like committee machine. Note that, due to the binary constraint on the model, adding weights to the second layer is redundant, since for all negative weights in the second layer we could always flip both its weight and all the weights of the unit connected to it. Therefore, without loss of generality, we just set the weights of the second layer to 1, resulting in the above definition of the output function ζ.

The scalar product between two replicas a and b is defined as $W^a \cdot W^b = \sum_{k=1}^{K} \sum_{i=1}^{N/K} W_i^{ka} W_i^{kb}$. For brevity of notation, in cases where the unit index does not play a role, we will often just use a single index $j \in \{1, \dots, N\}$, e.g. $W^a \cdot W^b = \sum_{j=1}^{N} W_j^a W_j^b$.

B. Patterns

The networks are trained on random input/output associations, i.e. patterns, (ξ^μ, σ_D^μ) where $\mu \in \{1, \dots, \alpha N\}$ is the pattern index. The parameter $\alpha > 0$ determines the load of the network, so that the number of patterns is

proportional to the number of synapses. The inputs are binary vectors of N elements with entries $\xi_i^{k\mu} \in \{-1, +1\}$, and the desired outputs are also binary, $\sigma_D^\mu \in \{-1, +1\}$. Both the inputs and the outputs are extracted at random and are independent and identically distributed (i.i.d.), except in the case of the fully-connected committee machine where $\xi_i^{k\mu} = \xi_i^{k'\mu}$ for all k, k' and therefore we only extract the values for $k = 1$.

We also actually exploit a symmetry in the problem and set all desired outputs to 1, since for each pattern its opposite must have an opposite output, i.e. we can always transform an input output pair (ξ^μ, σ_D^μ) into $(\xi^{\mu\prime}, 1)$, where the new pattern $\xi^{\mu\prime} = \sigma_D^\mu \xi^\mu$ has the same probability as ξ^μ.

C. Energy definition

The energy, or cost, for each pattern is defined as the minimum number of weights which need to be switched in order to correctly classify the pattern, i.e. in order to satisfy the relation $\zeta\left(\left\{\xi^{k\mu}\right\}_k, \left\{W^k\right\}_k\right) = 1$. The total energy is the sum of the energies for all patterns, $E(W) = \sum_{\mu=1}^{\alpha N} E^\mu(W)$.

If the current configuration of the weights W satisfies the pattern, the corresponding energy is obviously 0. Therefore, if the training problem is satisfiable, the ground states with this energy definition are the same as for the easier energy function given in terms of the number of errors.

If the current configuration violates the pattern, the energy can be computed as follows: we need to compute the minimum number c^μ of units of the first level which need to change their outputs, choose the c^μ units which are easiest to fix, and for each of them compute the minimum number of weights which need to be changed. In formulas:

$$E^\mu(W) = \Theta\left(-\Delta_{\text{out}}^\mu\right) \sum_{k=1}^{c^\mu} s_k^\mu \tag{S1}$$

where:

$$\Delta_k^\mu = \xi^{k\mu} \cdot W^k \tag{S2}$$

$$\Delta_{\text{out}}^\mu = \sum_k \text{sign}\left(\Delta_k^\mu\right) \tag{S3}$$

$$s^\mu = \text{sort}\left(\left\{-\frac{1}{2}\left(\Delta_k^\mu - 1\right),\ \forall k : \Delta_k^\mu < 0\right\}\right) \tag{S4}$$

$$c^\mu = \frac{1}{2}\left(-\Delta_{\text{out}}^\mu + 1\right) \tag{S5}$$

where the sort $(\cdot)$ function returns its argument sorted in ascending order. The above auxiliary quantities all depend on W, but we omitted the dependency for clarity of notation.

In the single-layer case $K = 1$ the expression simplifies considerably, since $\Delta_{\text{out}}^\mu = \xi^\mu \cdot W$ and reduces to $E^\mu(W) = \Theta\left(-\Delta_{\text{out}}^\mu\right) \frac{1}{2}\left(-\Delta_{\text{out}}^\mu + 1\right)$.

S2. REPLICATED SIMULATED ANNEALING

We run Simulated Annealing (plus "scoping") on a system of interacting replicas. For simplicity, we trace away the reference configuration which mediates the interaction. Thus, at any given step, we want to sample from a probability distribution

$$\begin{aligned} P\left(\{W^a\}\right) &\propto \sum_W \exp\left(-\beta \sum_{a=1}^y E\left(W^a\right) + \gamma \sum_{a=1}^y \sum_{j=1}^N W_j^a W_j\right) \\ &\propto \exp\left(-\beta \sum_{a=1}^y E\left(W^a\right) + \sum_j \log\left(2\cosh\left(\gamma \sum_{a=1}^y W_j^a\right)\right)\right) \end{aligned} \tag{S6}$$

The reference configuration is traced out in this representation, but we can obtain its most probable value by just computing $\tilde{W}_j = \text{sign} \sum_{a=1}^y W_j^a$. It is often the case that, when the parameters are chosen appropriately,

$E\left(\tilde{W}\right) \leq \langle E\left(W^a\right)\rangle_a$, i.e. that the energy of the center is lower than that of the group of replicas. In fact, we found this to be a good rule-of-thumb criterion to evaluate the choice of the parameters in the early stages of the algorithmic process.

The most straightforward way to perform the sampling (at fixed β and γ) is by using the Metropolis rule; the proposed move is to flip a random synaptic weight from a random replica. Of course the variation of the energy associated to the candidate move now includes the interaction term, parametrized by γ, which introduces a bias that favors movements in the direction of the center of mass of the replicas.

We also developed an alternative rule for choosing the moves in a biased way which implicitly accounts for the interaction term while still obeying the detailed balance condition. This alternative rule is generally valid in the presence of an external field and is detailed at the end of this section. Its advantage consists in reducing the rejection rate, but since the move proposal itself becomes more time consuming it is best suited to systems in which computing the energy cost of a move is expensive, so its usefulness depends on the details of the model.

A. Computing the energy shifts efficiently

Here we show how to compute efficiently the quantity $E\left(W'\right) - E\left(W\right)$ when W' and W only differ in the value of one synaptic weight j and the energy is defined as in eq. (S1). To this end, we define some auxiliary quantities in addition to the ones required for the energy computation, eqs. (S2)-(S5) (note that we omit the replica index a here since this needs to be done for each replica independently):

$$P^+ = \{\mu : \Delta^\mu_{\text{out}} = 1\} \tag{S7}$$

$$P^- = \{\mu : \Delta^\mu_{\text{out}} < 0\} \tag{S8}$$

$$\chi^\mu = \begin{cases} 1 & \text{if } s^\mu < 0 \wedge c^\mu < K \wedge s^\mu_{c^\mu} = s^\mu_{c^\mu+1} \\ 0 & \text{otherwise} \end{cases} \tag{S9}$$

These quantities must be recomputed each time a move is accepted, along with (S2)-(S5). Note however that in later stages of the annealing process most moves are rejected, and the energy shifts can be computed very efficiently as we shall see below.

Preliminarily, we note that any single-flip move only affects the energy contribution from patterns in $P^+ \cup P^-$.

The contribution to the energy shift ΔE^μ for a proposed move $W^k_i \to -W^k_i$ is most easily written in pseudo-code:

Algorithm 1: Energy shift function

```
Function ΔE^μ(μ, k, i, W_i^{kμ})
    if μ ∈ P^+ then
        if ξ_i^μ ≠ W_i^{kμ} then return 0
        if sign(Δ_k^μ) ≠ 1 then return 0
        return 1
    else if μ ∈ P^- then
        if Δ_k^μ > 1 then return 0
        d := −ξ_i^μ W_i^{kμ}
        if Δ_k^μ > 0 ∧ d = 1 then return 0
        if Δ_k^μ = 1 then return 1
        v := −(Δ_k^μ + 1)/2 + 1
        if v > s^μ_{c^μ} then return 0
        if v < s^μ_{c^μ} then return -d
        if d = 1 then return -1
        if χ^μ = 1 then return 0
        return 1
    else
        return 0
    end
end
```

Indeed, this function is greatly simplified in the single-layer case $K = 1$.

B. Efficient Monte Carlo sampling

Here we describe a Monte Carlo sampling method which is a modification of the Metropolis rule when the system uses N binary variables W_j and the Hamiltonian function can be written as:

$$H(W) = E(W) - \frac{1}{\beta}\sum_{j=1}^{N} k_j W_j \tag{S10}$$

where the external fields k_j can only assume a finite (and much smaller than N) set of values. The factor β^{-1} is introduced merely for convenience. Comparing this to eq. (S6), we see that, having chosen a replica index a uniformly at random, we can identify

$$k_j = \frac{1}{2}\left(\log\left(\frac{\cosh\left(\gamma+\gamma\sum_{b\neq a} W_j^b\right)}{\cosh\left(-\gamma+\gamma\sum_{b\neq a} W_j^b\right)}\right)\right) \tag{S11}$$

.

Given a transition probability to go from state W to state W', $P(W\to W')$, the detailed balance equation reads:

$$P(W)\,P(W\to W') = P(W')\,P(W'\to W) \tag{S12}$$

Let us split the transition explicitly in two steps: choosing the index j and accepting the move. The standard Metropolis rule is: pick an index $j\in\{1,\dots,N\}$ uniformly at random, propose the flip of W_j, accept it with probability $\min\left(1, e^{-\beta\Delta E_{W\to W'}-2k_jW_j}\right)$, where $\Delta E_{W\to W'} = E(W')-E(W)$. We want to reduce the rejection rate and incorporate the effect of the field in the proposal instead. We write:

$$P(W\to W') = C(W\to W')\,A(W\to W') \tag{S13}$$

where C is the choice of the index, and A is the acceptance of the move. Usually C is uniform and we ignore it, but here instead we try to use it to absorb the external field term in the probability distribution. From detailed balance we have:

$$\begin{aligned}\frac{A(W\to W')}{A(W'\to W)} &= \frac{p(W')}{p(W)}\frac{C(W'\to W)}{C(W\to W')}\\ &= e^{-\beta\Delta E_{W\to W'}-2k_jW_j}\frac{C(W'\to W)}{C(W\to W')}\end{aligned} \tag{S14}$$

so if we could satisfy:

$$e^{-2k_jW_j}\frac{C(W'\to W)}{C(W\to W')} = 1 \tag{S15}$$

then the acceptance A would simplify to the usual Metropolis rule, involving only the energy shift ΔE. This will turn out to be impossible, yet easily fixable, so we still first derive the condition implied by eq. (S15). The key observation is that there is a finite number of classes of indices in W, based on the limited number of values that W_jk_j can take (in the case of eq. (S11) there are y possible values). Let us call K_c the possible classes, such that $W_j\in K_c \Leftrightarrow W_jk_j = c$, and let us call $n_c = |K_c|$ their sizes, with the normalization condition that $\sum_c n_c = N$. Within a class, we must choose the move j uniformly.

Then $C(W\to W')$ is determined by the probability of picking a class, which in principle could be a function of all the values of the n_c: $P_c\left(\{n_{c'}\}_{c'}\right)$. Suppose now that we have picked an index in a class K_c. The transition to W' would bring it into class K_{-c}, and the new class sizes would be

$$n'_{c'} = \begin{cases} n_{c'}+1 & \text{if } c' = -c\\ n_{c'}-1 & \text{if } c' = c\\ n_{c'} & \text{otherwise}\end{cases}$$

therefore:

$$\frac{C(W'\to W)}{C(W\to W')} = \frac{n_c}{P_c\left(\{n_{c'}\}_{c'}\right)}\frac{P_{-c}\left(\{n'_{c'}\}_{c'}\right)}{n_{-c}+1} \tag{S16}$$

Since the only values of $n_{c'}$ directly involved in this expression are n_c and n_{-c}, it seems reasonable to restrict the dependence of P_c and P_{-c} only on those values. Let us also call $q_c = n_c + n_{-c}$, which is unaffected by the transition. Therefore we can just write:

$$\frac{C\left(W' \to W\right)}{C\left(W \to W'\right)} = \frac{n_c}{q_c - n_c + 1} \frac{P_{-c}\left(q_c - n_c + 1, q_c\right)}{P_c\left(n_c, q_c\right)} \tag{S17}$$

Furthermore, we can assume – purely for simplicity – that:

$$P_c\left(n_c, q_c\right) + P_{-c}\left(q_c - n_c, q_c\right) = \frac{q_c}{N} \tag{S18}$$

which allows us to restrict ourselves in the following to the case $c > 0$, and which implies that the choice of the index will proceed like this: we divide the indices in super-classes $D_c = K_c \cup K_{-c}$ of size q_c and we choose one of those according to their size; then we choose either the class K_c or K_{-c} according to $P_c\left(n_c, q_c\right)$; finally, we choose an index inside the class uniformly at random. Considering this process, what we actually need to determine is the conditional probability of choosing K_c once we know we have chosen the super-class D_c:

$$\hat{P}_c\left(n_c, q_c\right) = \frac{N}{q_c} P_c\left(n_c, q_c\right) \tag{S19}$$

Looking at eq. (S15) we are thus left with the condition:

$$\hat{P}_c\left(n_c + 1, q_c\right) = e^{-2c} \frac{n_c + 1}{q_c - n_c} \left(1 - \hat{P}_c\left(n_c, q_c\right)\right) \tag{S20}$$

Considering that we must have $\hat{P}_c\left(0, q_c\right) = 0$, this expression allows us to compute recursively $\hat{P}_c\left(n_c, q_c\right)$ for all values of n_c. The computation can be carried out analytically and leads to $\hat{P}_c\left(n_c, q_c\right) = \phi\left(n_c, q_c, e^{-2c}\right)$ where the function ϕ is defined as:

$$\phi\left(n, q, \lambda\right) = \lambda \frac{n}{q - n + 1} {}_2F_1\left(1, 1 - n; q - n + 2; \lambda\right) \tag{S21}$$

with ${}_2F_1$ the hypergeometric function. However, we should also have $\hat{P}_c\left(q_c, q_c\right) = 1$, while $\phi\left(q, q, \lambda\right) = 1 - \left(1 - \lambda\right)^q$ and therefore this condition is only satisfied for $c = 0$ (in which case we recover $\hat{P}_c\left(n_c, q_c\right) = \frac{n_c}{q_c}$, i.e. the standard uniform distribution, as expected).

Therefore, as anticipated, eq. (S15) can not be satisfied[1], and we are left with a residual rejection rate for the case $n_c = q_c$. This is reasonable, since in the limit of very large c (i.e. very large γ in the case of eq. (S11)) the probability distribution of each spin must be extremely peaked on the state in which all replicas are aligned, such that the combined probability of all other states is lower than the probability of staying in the same configuration. Therefore we have (still for $c > 0$):

$$\hat{P}_c\left(n_c, q_c\right) = \phi\left(n_c, q_c, e^{-2c}\right)\left(1 - \delta_{n_c, q_c}\right) + \delta_{n_c, q_c} \tag{S22}$$

$$\frac{A\left(W \to W'\right)}{A\left(W' \to W\right)} = e^{-\beta \Delta E_{W \to W'}} \left(1 - \delta_{n_c, q_c}\left(1 - e^{-2c}\right)^{q_c}\right) \tag{S23}$$

where $\delta_{n,q}$ is the Kronecker delta symbol. The last condition can be satisfied by choosing a general acceptance rule of this form:

$$A\left(W \to W'\right) = \min\left(1, e^{-\beta \Delta E_{W \to W'}}\right) a_c\left(n_c, q_c\right) \tag{S24}$$

where

$$a_c\left(n_c, q_c\right) = \begin{cases} 1 - \delta_{n_c, q_c}\left(1 - e^{-2c}\right)^{q_c} & \text{if } c > 0 \\ 1 & \text{if } c \leq 0 \end{cases}$$

[1] Strictly speaking we have not proven this, having made some assumptions for simplicity. However it is easy to prove it in the special case in which $k_j \in \{-1, +1\}$, since then our assumptions become necessary.

In practice, the effect of this correction is that the state where all the variables in class K_c are already aligned in their preferred direction is a little "clingier" than the others, and introduces an additional rejection rate $\left(1-e^{-2c}\right)^{q_c}$ (which however is tiny when either c is small or q_c is large).

The final procedure is thus the following: we choose a super-class D_c at random with probability q_c/N, then we choose either K_c or K_{-c} according to $\hat{P}_c$ and finally pick another index uniformly at random within the class.

This procedure is highly effective at reducing the rejection rate induced by the external fields. As mentioned above, depending on the problem, if the computation of the energy shifts is particularly fast, it may still be convenient in terms of CPU time to produce values uniformly and rejecting many of them, rather then go through a more involved sampling procedure. Note however that the bookkeeping operations required for keeping track of the classes compositions and their updates can be performed efficiently, in $\mathcal{O}(1)$ time with $\mathcal{O}(N)$ space, by using an unsorted partition of the spin indices (which allows for efficient insertion/removal) and an associated lookup table. Therefore, the additional cost of this procedure is a constant factor at each iteration.

Also, the function $\phi(n,q,\lambda)$ involves the evaluation of a hypergeometric function, which can be relatively costly; its values however can be pre-computed and tabulated if the memory resources allow it, since they are independent from the problem instance. For large values of $q-n(1-\lambda)$, it can also be efficiently approximated by a series expansion. It is convenient for that purpose to change variables to

$$x = q - n(1-\lambda)$$
$$\rho = \frac{n\lambda}{x}$$

(note that $\rho \in [0,1]$). We give here for reference the expansion up to x^{-2}, which ensures a maximum error of 10^{-5} for $x \geq 40$:

$$\phi\left(\frac{x\rho}{\lambda}, x\left(1+\rho\frac{1-\lambda}{\lambda}\right), \lambda\right) = \rho\left(1-\frac{(1-\rho)(1-\lambda)}{x}\left(1+\frac{1-(2-3\rho)(1-\lambda)}{x}\left(1+\mathcal{O}\left(\frac{1}{x}\right)\right)\right)\right) \tag{S25}$$

Finally, note that the assumption of eq. (S18) is only justified by simplicity; it is likely that a different choice could lead to a further improved dynamics.[2]

C. Numerical simulations details

Our Simulated Annealing procedure was performed as follows: we initialized the replicated system in a random configuration, with all replicas being initialized equally. The initial inverse temperature was set to β_0, and the initial interaction strength to γ_0. We then ran the Monte Carlo simulation, choosing a replica index at random at each step and a synaptic index according to the modified Metropolis rule described in the previous section, increasing both β and γ, by a factor $1+\beta_f$ and $1+\gamma_f$ respectively, for each $1000y$ accepted moves. The gradual increase of β is called 'annealing' while the gradual increase of γ is called 'scoping'. Of course, since with our procedure the annealing/scoping step is fixed, the quantities β_f and γ_f should scale with N. The simulations are stopped as soon as any one of the replicas reaches zero energy, or after $1000Ny$ consecutive non-improving moves, where a move is classified as non-improving if it is rejected by the Metropolis rule or it does not lower the energy (this definition accounts for the situation where the system is trapped in a local minimum with neighboring equivalent configurations at large β, in which case the algorithm would keep accepting moves with $\Delta E = 0$ without doing anything useful).

In order to compare our method with standard Simulated Annealing, we just removed the interaction between replicas from the above described case, i.e. we set $\gamma_0 = 0$. This is therefore equivalent to running y independent (except for the starting configurations) procedures in parallel, and stopping as soon as one of them reaches a solution.

In order to determine the scaling of the solution time with N, we followed the following procedure: for each sample (i.e. patterns assignment) we ran the algorithm with different parameters and recorded the minimum number of iterations required to reach a solution. We systematically explored these values of the parameters: $\beta_0 \in \{0.1, 0.5, 1, 2, 3, \ldots, 10\}$, $\beta_f \in \{0.1, 0.2, \ldots, 4.9, 5.0\}$, $\gamma_0 \in \{0.1, 0.5, 1, 1.5\}$, $\gamma_f \in \{0, 0.01, 0.02, \ldots, 0.4\}$ (the latter two only in the interacting case, of course). This procedure gives us an estimate for the minimum number of iterations required to solve a typical problem at a given value of N, K and α. We tested 10 samples for each value of (N, K, α). Since the interacting case has 2 additional parameters, this implies that there were more optimization opportunities, attributable to random chance; this however is not remotely sufficient to explain the difference in performance between the two cases: in fact, comparing instead for the typical value of iterations required (i.e. optimizing the average

[2] We did in fact generalize and improve this scheme after the preparation of this manuscript, see [1].

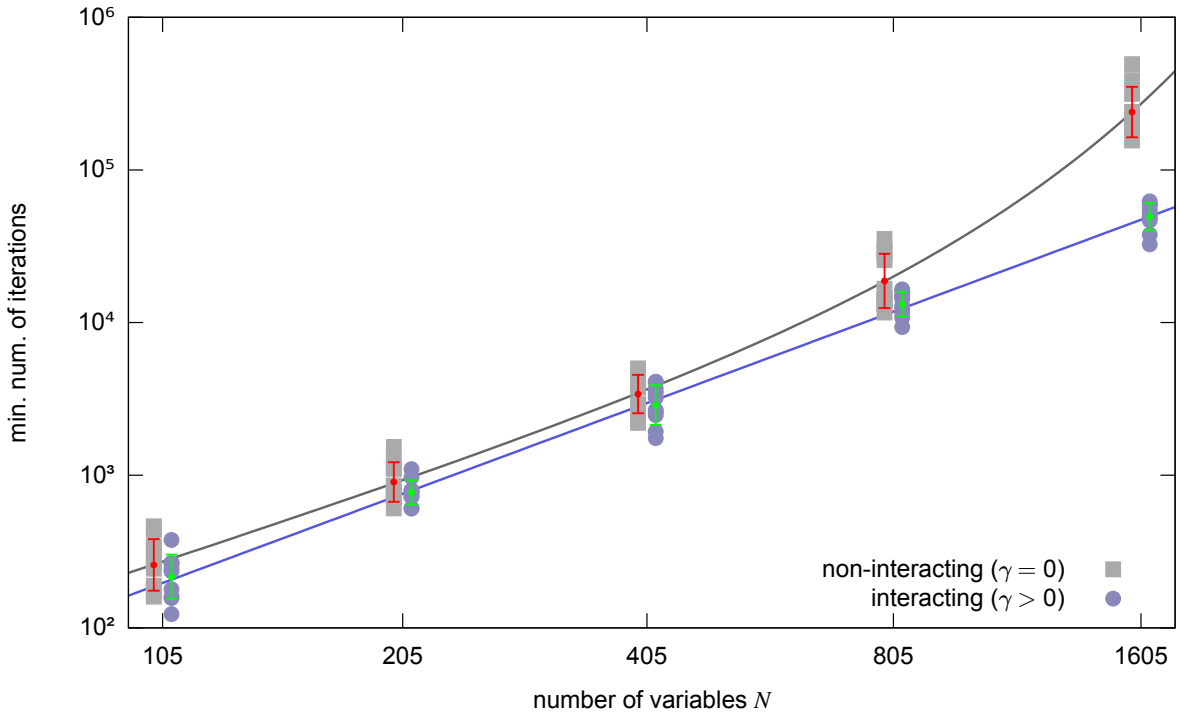


Figure S1: Replicated Simulated Annealing on the fully-connected committee machine, with $K = 5$ hidden units, comparison between the interacting version (i.e. which seeks regions of high solution density) and the non-interacting version (i.e. standard SA), at $\alpha = 0.2$ using $y = 3$ replicas. This is the analogous of figure 2 of the main text for a committee machine, showing similar results. 10 samples were tested for each value of N (the same samples were used for the two curves). The bars represent averages and standard deviations (taken in logarithmic scale) while the lines represent fits. The interacting case was fitted by a function aN^b with $a \simeq 0.02$, $b \simeq 2.0$, while the non-interacting case was fitted by a function $aN^b e^{cN^d}$ with $a \simeq 0.08$, $b \simeq 1.7$, $c \simeq 4.2 \cdot 10^{-5}$, $d \simeq 1.5$. The two data sets are slightly shifted relative to each other for presentation purposes.

iterations over $(\beta_0, \beta_f, \gamma_0, \gamma_f)$) gives qualitatively similar results, since once a range of good values for the parameters is found the iterations required to reach a solution are rather stable across samples.

The results are shown in figure 2 of the main text for the single-layer case at $\alpha = 0.3$ and figure S1 for the fully-connected two-layer case (committee machine) at $\alpha = 0.2$ and $K = 5$. In both cases we used $y = 3$, which seems to provide good results (we did not systematically explore different values of y). The values of α were chosen so that the standard SA procedure would be able to solve some instances at low N in reasonable times (since the difference in performance between the interacting and non-interacting cases widens greatly with increasing α). The results show a different qualitative behavior in both cases, polynomial for the interacting case and exponential for the non-interacting cases. All fits were performed directly in logarithmic scale.

S3. REPLICATED GRADIENT DESCENT

A. Gradient computation

As mentioned in the main text, we perform a stochastic gradient descent on binary networks using the energy function of eq. (S1) by using two sets of variables: a set of continuous variables $\mathcal{W}_i^k$ and the corresponding binarized variables W_i^k, related by $W_i^k = \text{sign}\left(\mathcal{W}_i^k\right)$. We use the binarized variables to compute the energy and the gradient, and apply the gradient to the continuous variables. In formulas, the quantities at time $t + 1$ are related to those at

time t by:

$$\left(\mathcal{W}_i^k\right)^{t+1} = \left(\mathcal{W}_i^k\right)^t - \eta \frac{1}{|m\left(t\right)|} \sum_{\mu \in m(t)} \frac{\partial}{\partial W_i^k} E^\mu \left(W^t\right) \tag{S26}$$

$$\left(W_i^k\right)^{t+1} = \text{sign}\left(\left(\mathcal{W}_i^k\right)^{t+1}\right) \tag{S27}$$

where η is a learning rate and $m\left(t\right)$ is a set of pattern indices (a so-called minibatch). A particularly simple scenario can be obtained by considering a single layer network without replication ($K = 1$, $y = 1$) and a fixed learning rate, and by computing the gradient one pattern at a time ($|m\left(t\right)| = 1$). In that case, $E^\mu\left(W\right) = R\left(-\sum_i W_i \xi_i^\mu\right)$ where $R\left(x\right) = \frac{1}{2}\left(x+1\right)\Theta\left(x\right)$ and the gradient is $\partial_{W_i} E^\mu\left(W\right) = -\frac{1}{2}\xi_i^\mu \Theta\left(-\sum_i W_i \xi_i^\mu\right)$. Since the relation (S27) is scale-invariant, we can just set $\eta = 4$ and obtain

$$\mathcal{W}_i^{t+1} = \mathcal{W}_i^t - 2\xi_i^\mu \Theta\left(-\sum_i W_i^t \xi_i^\mu\right) \tag{S28}$$

where now the auxiliary quantities $\mathcal{W}$ are discretized: if they are initialized as odd integers, they remain odd integers throughout the learning process. This is the so-called "Clipped Perceptron" (CP) rule, which is the same as the Perceptron rule ("in case of error, update the weights in the direction of the pattern, otherwise do nothing") except that the weights are clipped upon usage to make them binary. Notably, the CP rule by itself does not scale well with N; it is however possible to make it efficient (see [2, 3]).

In the two-layer case ($K > 1$) the computation of the gradient is more complicated; it is however simpler than the computation of the energy shift which was necessary for Simulated Annealing (Algorithm 1), since we only consider infinitesimal variations when computing the gradient. The resulting expression is:

$$\partial_{W_i^k} E^\mu\left(W\right) = \begin{cases} -\frac{1}{2}\xi_i^{k\mu} & \text{if } \left(\Delta_{\text{out}}^\mu < 0\right) \wedge \left(1 + 2s_{c^\mu}^\mu \leq \Delta_k^\mu < 0\right) \\ 0 & \text{otherwise} \end{cases} \tag{S29}$$

i.e. the gradient is non-zero only in case of error, and only for those units k which contribute to the energy computation (which turn up in the first c^μ terms of the sorted vector s^μ, see eqs. (S2)-(S5)). Again, since this gradient can take only 3 possible values, we could set $\eta = 4$ and use discretized odd variables for the $\mathcal{W}$.

It is interesting to point out that a slight variation of this update rule in which only the first, least-wrong unit is affected, i.e. in which the condition $\left(1 + 2s_{c^\mu}^\mu \leq \Delta_k^\mu\right)$ is changed to $\left(1 + 2s_1^\mu \leq \Delta_k^\mu\right)$, was used in [4], giving good results on a real-world learning task when a slight modification analogous to the one of [3] was added. Note that, in the later stages of learning, when the overall energy is low, it is very likely that $c^\mu \leq 1$, implying that the simplification used in [4] likely has a negligible effect. The simplified version, when used in the continuous case, also goes under the name of "least action" algorithm [5].

Having computed the gradient of $E\left(W\right)$ for each system, the extension to the replicated system is rather straightforward, since the energy (with the traced-out center) becomes (cf. eqs. (4) and (5) in the main text):

$$H\left(\{W^a\}\right) = \sum_{a=1}^{y} E\left(W^a\right) - \frac{1}{\beta}\sum_{j=1}^{N} \log\left(e^{-\frac{\gamma}{2}\sum_{a=1}^{y}\left(W_j^a - 1\right)^2} + e^{-\frac{\gamma}{2}\sum_{a=1}^{y}\left(W_j^a + 1\right)^2}\right) \tag{S30}$$

and therefore the gradient just has an additional term:

$$\frac{\partial H}{\partial W_i^a}\left(\{W^b\}\right) = \frac{\partial E}{\partial W_i}\left(W\right)\Big|_{W=W^a} - \frac{\gamma}{\beta}\left(\tanh\left(\gamma \sum_{b=1}^{y} W_i^b\right) - W_i^a\right) \tag{S31}$$

Note that the trace operation brings the parameter β into account. Using $\eta' = \frac{\gamma}{\beta\eta}$ as control parameter, the update equation (S26) for a replica a becomes (we omit the unit index k for simplicity):

$$\left(\mathcal{W}_i^a\right)^{t+1} = \left(\mathcal{W}_i^a\right)^t - \eta \frac{1}{|m\left(t\right)|} \sum_{\mu \in m(t)} \frac{\partial E^\mu}{\partial W_i}\left(W\right)\Big|_{W=\left(W^a\right)^t} + \eta'\left(\tanh\left(\gamma \sum_{b=1}^{y}\left(W_i^b\right)^t\right) - \left(W_i^a\right)^t\right) \tag{S32}$$

In the limit $\beta, \gamma \to \infty$, η' stays finite, while the tanh reduces to a sign.

The expression of eq. (S32) is derived straightforwardly, gives good results and is the one that we have used in the tests shown in the main text and below. It could be noted, however, that the two-level precision of the variables used

in the algorithm introduces some artifacts. As a clear example, in the case of a single replica ($y = 1$) or, more in general, when the replica indices W_i^a are all aligned, we would expect the interaction term to vanish, while this is not the case except at $\gamma = \infty$.

One possible way to fix this issue is the following: we can introduce a factor in the logarithm in expression (S30):

$$\log\left(\frac{e^{-\frac{\gamma}{2}\sum_{a=1}^{y}\left(W_j^a-1\right)^2}+e^{-\frac{\gamma}{2}\sum_{a=1}^{y}\left(W_j^a+1\right)^2}}{f\left(W_j^1,\dots,W_j^y\right)}\right) \tag{S33}$$

such that $f\left(W_j^1,\dots,W_j^y\right)=1$ whenever its arguments lie on the vertices of the hypercube, $W_j^a \in \{-1,1\}$. This does not change the Hamiltonian for the configurations we're interested in, but it can change its gradient. We can thus impose the additional constraint that the derivative of the above term vanishes whenever the W_j^a are all equal. There are several ways to achieve this; however, if we assume that the function f has the general structure

$$f\left(W_j^1,\dots,W_j^y\right)=a\left(g\left(W_j^1\right),\dots,g\left(W_j^y\right)\right) \tag{S34}$$

with $g(1)=g(-1)$ and with $a(\dots)$ being a totally symmetric function of its arguments[3], then it can be easily shown that necessarily

$$\frac{\partial}{\partial W_i^a}\log\left(\frac{e^{-\frac{\gamma}{2}\sum_{b=1}^{y}\left(W_j^b-1\right)^2}+e^{-\frac{\gamma}{2}\sum_{b=1}^{y}\left(W_j^b+1\right)^2}}{a\left(g\left(W_j^1\right),\dots,g\left(W_j^y\right)\right)}\right)=\gamma\left(\tanh\left(\gamma\sum_{b=1}^{y}W_i^b\right)-\tanh\left(\gamma y\right)W_i^a\right) \tag{S35}$$

This expression has now two zeros corresponding to the fully aligned configurations of the weights at $+1$ and -1, as desired, and is a very minor correction of the original one used in eq. (S32) (the expressions become identical at large values of γy). In fact, we found that the numerical results are basically the same (the optimal values of the parameters may change, but the performances for optimal parameters are very similar for the two cases), such that this correction is not needed in practice.

An alternative, more straightforward way to fix the issue of the non-vanishing gradient with aligned variables is to perform the trace over the reference configurations in the continuous case (i.e. replacing the sum over the binary hypercube with an integral). This leads to the an expression for the interaction contribution to the gradient of this form: $\gamma\left(\frac{1}{y}\sum_{b=1}^{y}W_i^b-W_i^a\right)$. This, however, does seem to have a very slightly but measurably worse overall performance with respect to the previous ones (while still dramatically outperforming the non-interacting version).

In general, the tests with alternative interaction terms show that, despite the fact that the two-level gradient procedure is purely heuristic and inherently problematic, the fine details of the implementation may not be exceedingly relevant for most practical purposes.

The code for our implementation is available at [6].

B. Numerical simulations details

Our implementation of the formula in eq. (S32) follows this scheme: at each time step, we have the values $T_i = \sum_{b=1}^{y}W_i^b$, we pick a random replica index a, compute the gradient with respect to some $m(t)$ patterns, update the values $\mathcal{W}^a$ and W^a, compute the gradient with respect to the interaction term using T and W^a, and update the values of T and – again – of $\mathcal{W}^a$ and W^a. This scheme is thus easy to parallelize, since it alternates the standard learning periods in which each replica acts independently with brief interaction periods in which the sum T is updated, similarly to what was done in [7].

An epoch consists of a presentation of all patterns to all replicas. The minibatches $m(t)$ are randomized at the beginning of each epoch, independently for each replica. The replicas were initialized equally for simplicity.

In our tests, we kept fixed the learning rates η and η' during the training process, since preliminary tests did not show a benefit in adapting them dynamically in our setting. We did, however, find beneficial in most cases to vary γ, starting at some value γ_0 and increasing it progressively by adding a fixed quantity $d\gamma$ after each epoch, i.e. implementing a "scoping" mechanism as in the Simulated Annealing case (although even just using $\gamma = \infty$ from the start already gives large improvements against the non-interacting version).

All tests were capped at a maximum of 10^4 epochs, and the minimum value of the error across all replicas was kept for producing the graphs.

---

[3] One possibility is using $g(w)=\frac{\cosh(\gamma y w)}{\cosh(\gamma y)}\exp\left(\frac{\gamma y}{2}\left(1-w^2\right)\right)$ and $a(\dots)$ equal to the average of its arguments.

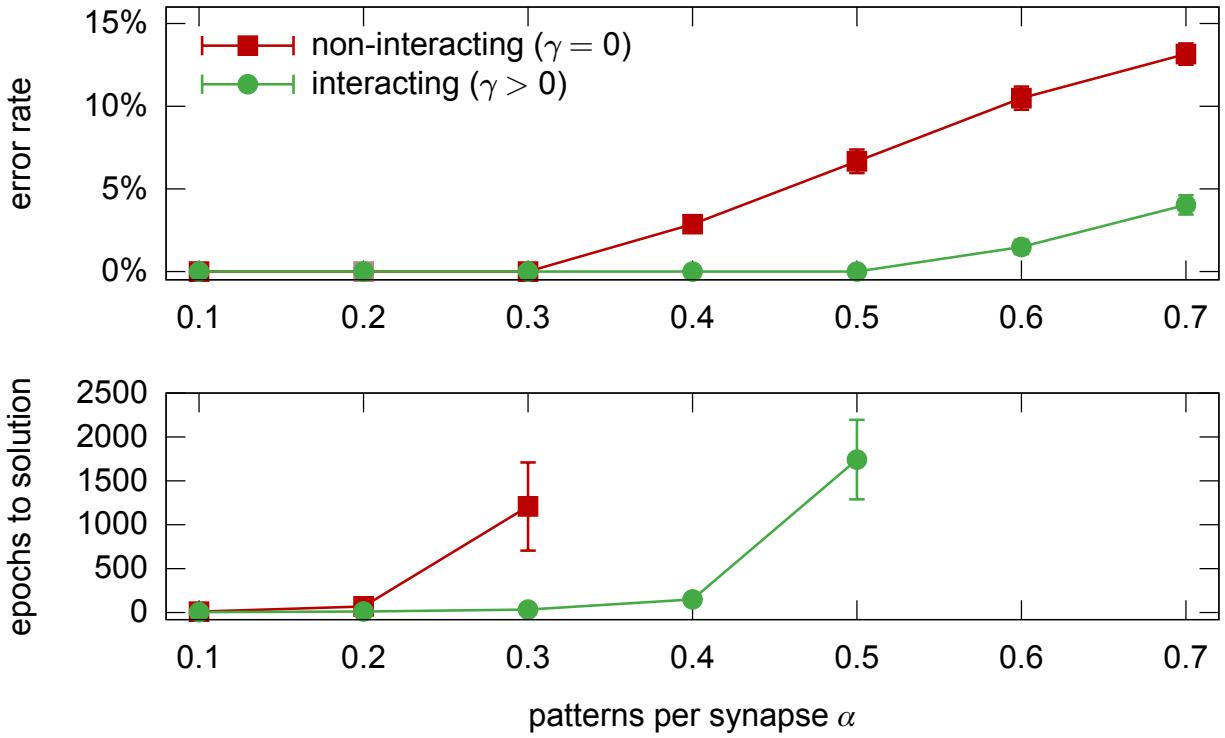


Figure S2: Replicated Stochastic Gradient descent on a fully-connected committee machine with $N = 1605$ synapses and $K = 5$ units in the second layer, comparison between the non-interacting (i.e. standard SGD) and interacting versions, using $y = 3$ replicas and a minibatch size of 10 patterns. Each point shows averages and standard deviations on 10 samples with optimal choice of the parameters, as a function of the training set size. Top: minimum training error rate achieved after 10^4 epochs. Bottom: number of epochs required to find a solution. Only the cases with 100% success rate are shown.

In the interacting case, we systematically tested various values of η', γ_0 and $d\gamma$, and, for each α, we kept the ones which produced optimal results (i.e. lowest error rate, or shorter solution times if the error rates were equal) on average across the samples. Because of the overall scale invariance of the problem, we did not change η.

Figure S2 shows the results of the same tests as shown in figure 3 of the main text for different values of the number of replicas and the minibatch size. The results for the interacting case are slightly worse, but still much better than for the non-interacting case.

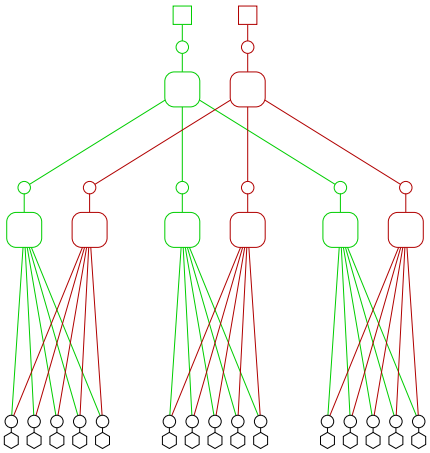
For the perceptron case $K = 1$ we did less extensive tests; we could solve 10 samples out of 10 with $N = 1601$ synapses at $\alpha = 0.7$ in an average of 4371 ± 661 epochs with $y = 7$ replicas and a minibatch size of 80 patterns.

S4. REPLICATED BELIEF PROPAGATION

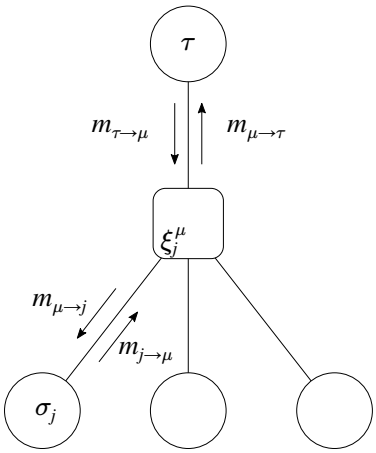
A. Belief Propagation implementation notes

Belief Propagation (BP) is an iterative message passing algorithm that can be used to derive marginal probabilities on a system within the Bethe-Peierls approximation [8–10]. The messages $P_{j\to\mu}(\sigma_j)$ (from variable node j to factor node μ) and $P_{\mu\to j}(\sigma_j)$ (from factor node a to variable node j) represent cavity probability distributions (called messages) over a single variable σ_j. In the case of Ising systems of binary ± 1 variables like the ones we are using in the network models considered in this work, the messages can be represented as a single number, usually a magnetization $m_{i\to\mu} = P_{i\to\mu}(+1) - P_{\mu\to i}(-1)$ (and analogous for the other case).

Our implementation of BP on binary networks follows very closely that of [11], since we only consider the zero temperature case and we are interested in the "satisfiable" phase, thus considering only configurations of zero energy. However, in order to avoid some numerical precision issues that affected the computations at high values of α, y and γ, we lifted some of the approximations used in that paper. Here therefore we recapitulate the BP equations used



(a) BP factor graph scheme. This scheme exemplifies a factor graph for a committee machine with $N = 15$ variables, $K = 3$ units in the second layer, trained on 2 patterns. The two patterns are distinguished by different colors. The graph can represent a fully-connected committee machine if the patterns are the same for all first-layer units, or a tree-like one if they are different. The variable nodes are represented as circles, the interaction by other geometrical figures. The hexagons at the bottom represent pseudo-self-interaction nodes (see main text, figure 4), the large squares with rounded corners represent perceptron-like nodes, the small squares at the top represent external fields enforcing the desired output of the machine. The synaptic variables W_j^k are at the bottom (black circles), while the rest of the variables are auxiliary and represent the output of each unit for a given pattern.



(b) BP messages naming scheme used in section S4 A for a perceptron-like factor node. The node μ is represented by the central square. Input variables, denoted by σ_j, are at the bottom. The output variable is called τ. The couplings ξ_j^μ parametrize the factor node (one parameter per input edge) and can either represent an input pattern (for the first layer of the network) or be 1 (for the second layer of the network).

Figure S3

and highlight the differences with the previous work. The factor graph scheme for a committee machine is shown for reference in figure S3a. The BP equations for the messages from a variable node j to a factor node μ can be written in general as:

$$m_{j\to\mu}^t = \tanh\left(\sum_{\nu\in\partial j\setminus\mu} \tanh^{-1}\left(m_{\nu\to j}^t\right) + \tanh^{-1}\left(m_{\star\to j}^t\right)\right) \tag{S36}$$

where ∂j represent the set of all factor nodes in which variable j is involved. This expression includes the Focusing BP (fBP) extra message $m_{\star\to j}$ described in the main text. The general expression for perceptron-like factor nodes is considerably more complicated. For the sake of generality, here we will use the symbol σ to denote input variables of the node (with subscript j indicating the variable), and τ for the output variable. To each perceptron-like factor μ is associated a vector of couplings ξ^μ: in a committee-machine, these represent the patterns for the first layer nodes, and are simply vectors of ones in the second layer. See figure S3b.

Let us define the auxiliary functions:

$$\mathrm{f}_j^\mu\left(\{m_{i\to\mu}\}_{i\in\partial\mu\setminus j}, m_{\tau\to\mu}, \sigma_j\right) = \sum_{\tau,\sigma_{\partial\mu\setminus j}} \left(\frac{1+\tau\, m_{\tau\to\mu}}{2}\right) \Theta\left(\tau\left(\sum_{i\in\partial\mu\setminus j} \xi_i^\mu\sigma_i + \xi_j^\mu\sigma_j\right)\right) \prod_{i\in\partial\mu\setminus j}\left(\frac{1+\sigma_i m_{i\to\mu}}{2}\right) \tag{S37}$$

$$\mathrm{f}^\mu\left(\{m_{i\to\mu}\}_{i\in\partial\mu}, \tau\right) = \sum_{\sigma_{\partial\mu\setminus j}} \Theta\left(\tau\left(\sum_{i\in\partial\mu} \xi_i^\mu\sigma_i\right)\right) \prod_{i\in\partial\mu}\left(\frac{1+\sigma_i m_{i\to\mu}}{2}\right) \tag{S38}$$

where $\partial\mu$ represents the set of all input variables involved in node μ, $\sigma_{\partial\mu} = \{\sigma_i\}_{i\in\partial\mu}$ the configuration of input variables involved in node μ, $m_{\tau\to\mu}$ the message from the output variable τ to the node μ (see figure **S3b** for reference). With

these, the messages from factor node μ to the output variable node τ can be expressed as:

$$m_{\mu\to\tau}^{t+1} = \frac{\mathrm{f}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu}, +1\right) - \mathrm{f}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu}, -1\right)}{\mathrm{f}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu}, +1\right) + \mathrm{f}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu}, -1\right)} \tag{S39}$$

while the message from factor node μ to input variable node j is:

$$m_{\mu\to j}^{t+1} = \frac{\mathrm{f}_{j}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu\setminus j}, m_{\tau\to\mu}^{t}, +1\right) - \mathrm{f}_{j}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu\setminus j}, m_{\tau\to\mu}^{t}, -1\right)}{\mathrm{f}_{j}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu\setminus j}, m_{\tau\to\mu}^{t}, +1\right) + \mathrm{f}_{j}^{\mu}\left(\left\{m_{i\to\mu}^{t}\right\}_{i\in\partial\mu\setminus j}, m_{\tau\to\mu}^{t}, -1\right)} \tag{S40}$$

These functions can be computed exactly in $\mathcal{O}\left(N^3\right)$ operations, where N is the size of the input, using either a partial convolution scheme or discrete Fourier transforms. When N is sufficiently large, it is also possible to approximate them in $O\left(N\right)$ operations using the central limit theorem, as explained in [11]. In our tests on the committee machine, due to our choice of the parameters, we used the approximated fast version on the first layer and the exact version on the much smaller second layer.

In the fast approximated version, eqs. (S39) and (S40) become:

$$m_{\mu\to\tau}^{t+1} = \mathrm{erf}\left(\frac{a_{\mu}^{t}}{\sqrt{2b_{\mu}^{t}}}\right) \tag{S41}$$

$$m_{\mu\to j}^{t+1} = m_{\tau\to\mu}^{t}\frac{\mathrm{g}_{\mu\to j}^{t}\left(+1\right) - \mathrm{g}_{\mu\to j}^{t}\left(-1\right)}{2 + m_{\tau\to\mu}^{t}\left(\mathrm{g}_{\mu\to j}^{t}\left(+1\right) + \mathrm{g}_{\mu\to j}^{t}\left(-1\right)\right)} \tag{S42}$$

where we have defined the following quantities:

$$a_{\mu}^{t} = \sum_{i\in\partial\mu}\xi_{i}^{\mu}m_{i\to\mu}^{t} \tag{S43}$$

$$b_{\mu}^{t} = \sum_{i\in\partial\mu}\left(1 - \left(m_{i\to\mu}^{t}\right)^{2}\right) \tag{S44}$$

$$\mathrm{g}_{\mu\to j}^{t}\left(\sigma\right) = \mathrm{erf}\left(\frac{a_{\mu\to j}^{t} + \sigma\xi_{j}^{\mu}}{\sqrt{2\left(b_{\mu\to j}^{t}\right)}}\right) \tag{S45}$$

$$a_{\mu\to j}^{t} = a_{\mu}^{t} - \xi_{j}^{\mu}m_{j\to\mu}^{t} \tag{S46}$$

$$b_{\mu\to j}^{t} = b_{\mu}^{t} - \left(1 - \left(m_{j\to\mu}^{t}\right)^{2}\right) \tag{S47}$$

In [11], eq. (S42) was approximated with a more computationally efficient expression in the limit of large N. We found that this approximation leads to numerical issues with the type of architectures which we used in our simulation at large values of α, y and γ. For the same reason, it is convenient to represent all messages internally in "field representation" as was done in [11], i.e. using $h_{\mu\to j} = \tanh^{-1}\left(m_{\mu\to j}\right)$ (and analogous expressions for all messages); furthermore, some expressions need to be treated specially to avoid numerical precision loss. For example, computing $h_{\mu\to\tau}$ according to eq. (S41) requires the computation of an expression of the type $\tanh^{-1}\left(\mathrm{erf}\left(x\right)\right)$, which, when computed naïvely with standard 64-bit IEEE floating point machine numbers and using standard library functions, rapidly loses precision at moderate-to-large values of the argument, thus requiring us to write a custom function to avoid this effect. The same kind of treatment is necessary throughout the code, particularly when computing the thermodynamic functions.

The code for our implementation is available at [12].

After convergence, the single-site magnetizations can be computed as:

$$m_{j} = \tanh\left(\sum_{\nu\in\partial j}\tanh^{-1}\left(m_{\nu\to j}\right) + \tanh^{-1}\left(m_{\star\to j}\right)\right) \tag{S48}$$

and the average overlap between replicas (plotted in Fig. 5 of the main text) as:

$$q = \frac{1}{N}\sum_{j}m_{j}^{2} \tag{S49}$$

The local entropy is computed from the entropy of the whole replicated system from the BP messages at their fixed point, as usually done within the Bethe-Peierls approximation, minus the entropy of the reference variables. The result is then divided by the number of variables N and of replicas y. (This procedure is equivalent to taking the partial derivative of the free energy expression with respect to y.) Finally, we take a Legendre transform by subtracting the interaction term γS, where S is the estimated overlap between each replica's weights and the reference:

$$S = \frac{1}{N} \sum_j \frac{m_{j\to\star} m_{\star\to j} + \tanh(\gamma)}{1 + m_{j\to\star} m_{\star\to j} \tanh(\gamma)} \tag{S50}$$

From S, the distance between the replicas and the reference is simply computed as $(1+S)/2$.

B. Focusing BP vs Reinforced BP

As mentioned in the main text, the equation for the pseudo-self-interaction of the replicated Belief Propagation algorithm (which we called "Focusing BP", fBP) is (eq. (6) in the main text):

$$m^{t+1}_{\star\to j} = \tanh\left((y-1)\tanh^{-1}\left(m^t_{j\to\star}\tanh\gamma\right)\right)\tanh\gamma \tag{S51}$$

See also figure 4 in the main text for a graphical description. The analogous equation for the reinforcement term which has been used in several previous works is (eq. (7) in the main text):

$$m^{t+1}_{\star\to j} = \tanh\left(\rho\tanh^{-1}\left(m^t_j\right)\right) \tag{S52}$$

The reinforced BP has traditionally been used as follows: the reinforcement parameter ρ is changed dynamically, starting from 0 and increasing it up to 1 in parallel with an ongoing BP message-passing iteration scheme. Therefore, in this approach, the BP messages can only converge (when $\rho = 1$) to a completely polarized configuration, i.e. one where $m_j \in \{-1, +1\}$ for all j.

The same approach can be applied with the fBP scheme, except that eq. (S51) involves two parameters, γ and y, rather than one, and both need to diverge in order to ensure that the marginals m_j become completely polarized as well.

In this scheme, however, it is unclear how to compare directly the two equations, since in eq. (S51) the self-reinforcing message $m_{\star\to j}$ is a function of a cavity marginal $m_{j\to\star}$, while in eq. (S52) it is a function of a non-cavity marginal m_j. In order to understand the relationship between the two, we take a different approach: we assume that the parameters involved in the two update schemes (γ and y on one side, ρ on the other) are fixed until convergence of the BP messages. In that case, one can then remove the time index t from eqs. (S51),(S52) and obtain a self-consistent condition between the quantities $m_{\star\to j}$, $m_{j\to\star}$ and m_j at the fixed point:

$$m_j = \tanh\left(\tanh^{-1}(m_{\star\to j}) + \tanh^{-1}(m_{j\to\star})\right) \tag{S53}$$

Therefore eq. (S52) in this case becomes equivalent to:

$$m_j = \tanh\left(\frac{1}{1-\rho}\tanh^{-1}(m_{j\to\star})\right) \tag{S54}$$

to be compared with the analogous expression for the fBP case:

$$m_j = \tanh\left(\tanh^{-1}(m_{j\to\star}) + \tanh^{-1}\left(\tanh\left((y-1)\tanh^{-1}(m_{j\to\star}\tanh\gamma)\right)\tanh\gamma\right)\right) \tag{S55}$$

This latter expression is clearly much more complicated, but by letting $\gamma \to \infty$ and setting $y = \frac{1}{1-\rho}$ it simplifies to eq. (S54). Therefore, we have an exact mapping between fBP and the reinforced BP. The interpretation of this mapping in terms of the reweighted entropic measure (eq. (3) of the main text) is not straightforward, because of the requirement $\gamma = \infty$. The $\gamma = \infty$ case at finite y is an extreme case: as mentioned in the main text, the BP equations applied to the replicated factor graph (figure 4 of the main text) neglect the correlations between the messages $m_{\star\to j}$ targeting different replicas. Since as $\gamma \to \infty$ these messages become completely correlated, the approximation fails. As demonstrated by our results, though (figures 5 and 6 in the main text, and figure S5 below), this failure in the approximation only happens at very large values of γ, and its onset is shifted to larger values of γ as y is increased, so that even keeping y fixed (but sufficiently large) and gradually increasing γ gives very good results (figure 6 in the

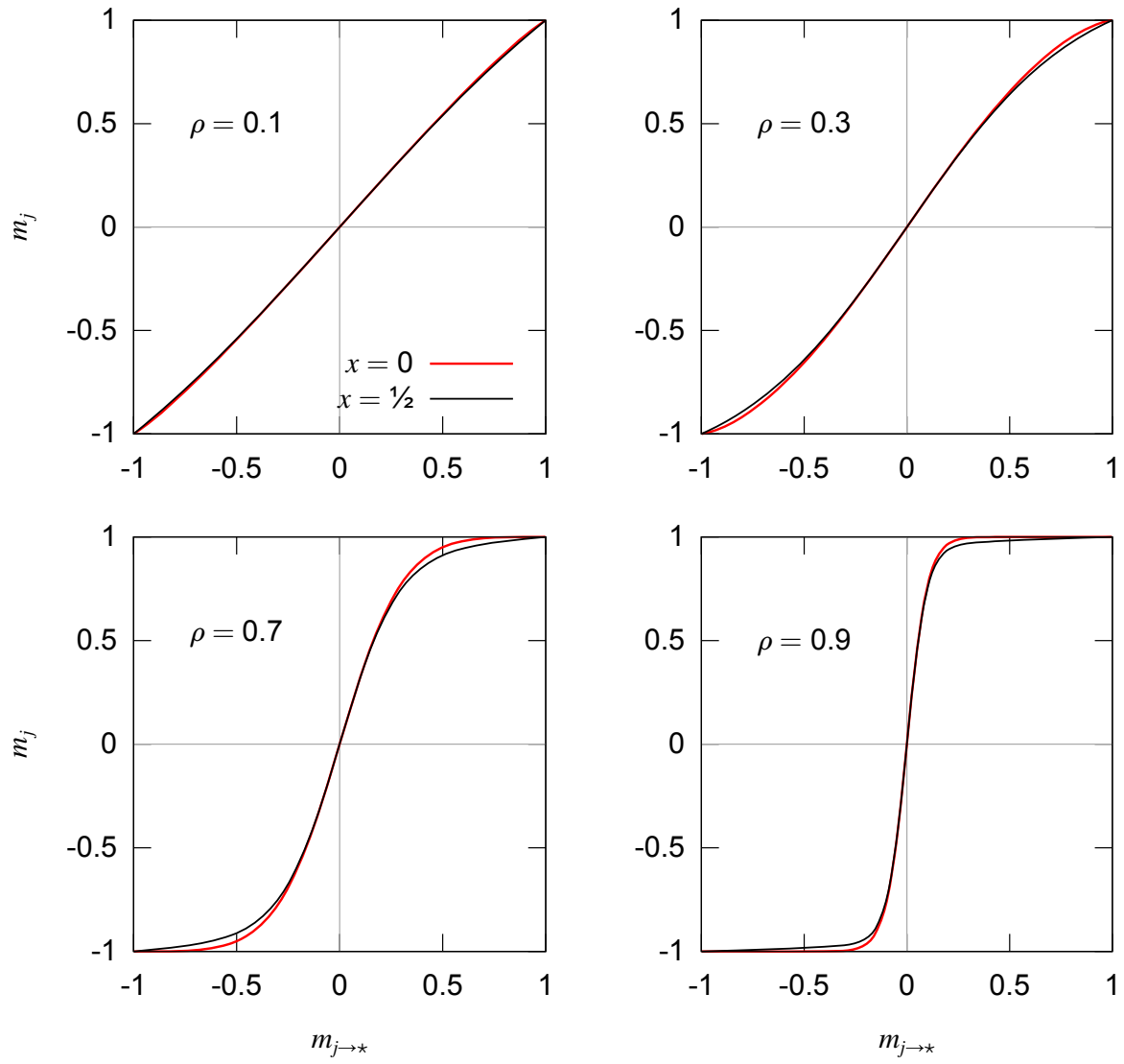


Figure S4: Plots of eq. (S55), comparison of protocols defined by eqs. (S57) and (S56) with two different values of the parameter x. The $x = 0$ case (thick red lines) corresponds to standard reinforcement. The curves are in fact very similar across the whole range of $\rho \in [0, 1]$ and $x \in [0, 1]$, and consequently display similar performance properties in practice.

main text). A different approach to the interpretation of the reinforcement protocol is instead to consider that it is only one among several possible protocols. One possible generalization, in which both γ and y start from low values and are progressively increased, is:

$$\gamma = \tanh^{-1}(\rho^x) \tag{S56}$$

$$y = 1 + \frac{\rho^{1-2x}}{(1-\rho)} \tag{S57}$$

The second expression was obtained by assuming the first one and matching the derivative of the curves of eqs. (S54) and (S55) in the point $m_{j\to\star} = 0$. Note that with this choice, both $\gamma \to \infty$ and $y \to \infty$ in the limit $\rho \to 1$, thus ensuring that, in that limit, the only fixed points of the iterative message passing procedure are completely polarized, and consistently with the notion that we are looking regions of maximal density ($y \to \infty$) at small distances ($\gamma \to \infty$). When setting $x = 0$, this reproduces the standard reinforcement relations. However, other values of x produce the same qualitative behavior, and are quantitatively very similar: figure (S4) shows the comparison with the case $x = 0.5$. In practice, in our tests these protocols have proved to be equally effective in finding solutions of the learning problem.

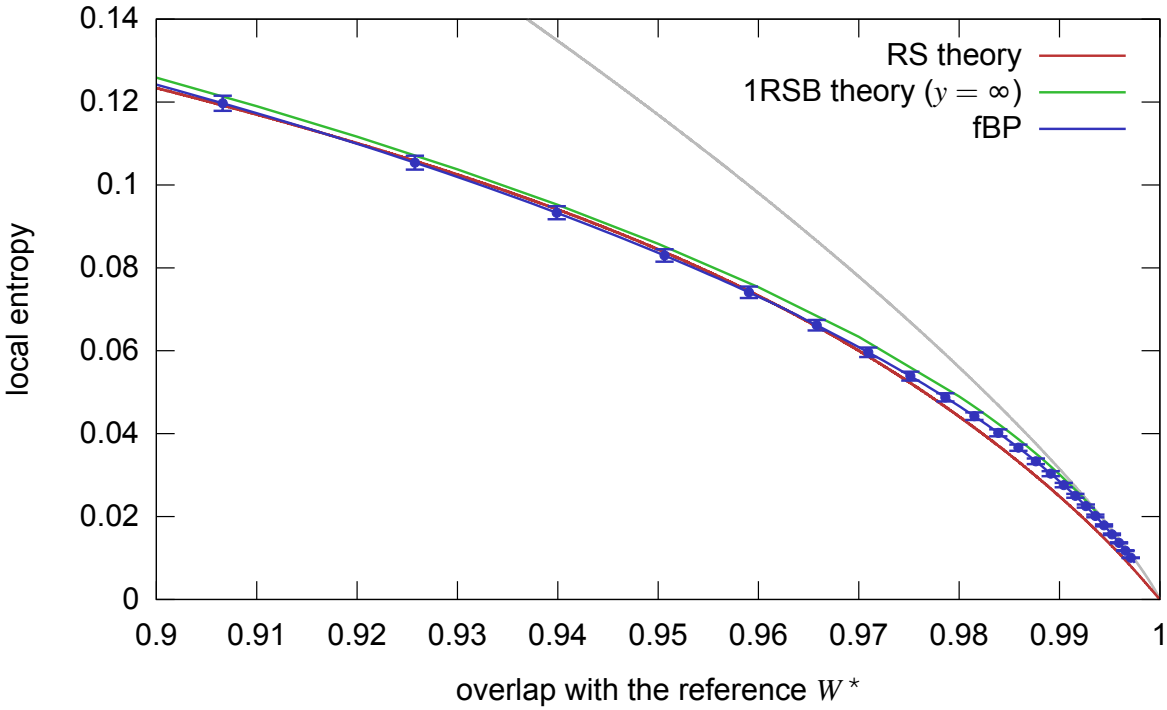


Figure S5: Comparison of local entropy curves between the fBP results and the analytical predictions, for the case of the perceptron with $\alpha = 0.6$. The algorithmic results (blue curve) were obtained with $N = 1001$ at $y = 21$, averaging over 50 samples. Error bars indicate the estimated standard deviation of the mean. The RS results (red curve) were also obtained with $y = 21$. The 1RSB results, however, are for the $y = \infty$ case, and it is therefore to be expected that the corresponding curve is slightly higher.

C. Focusing BP vs analytical results

We compared the local entropy curves produced with the fBP algorithm on perceptron problems with the RS and 1RSB results obtained analytically in [4, 13]. We produced curves at fixed y and α, while varying γ. However, we only have 1RSB results for the $y = \infty$ case. Figure S5 shows the results for $\alpha = 0.6$ and $y = 21$, demonstrating that the fBP curve deviates from the RS prediction and is very close to the 1RSB case. Our tests show that the fBP curve get closer to the 1RSB curve as y grows. This analysis confirms a scenario in which the fBP algorithm spontaneously choses a high density state, breaking the symmetry in a way which seems to approximate well the 1RSB description. Numerical precision issues limited the range of parameters that we could explore in a reasonable time.

D. Focusing BP on random K-SAT

In this section, we show how to apply the Focusing BP (fBP) scheme on a prototypical constraint statisfaction problem, the random K-satisfiability (K-SAT) problem [10], and present the results of some preliminary experiments in which it is used as a solver algorithm.

An instance of K-SAT is defined by N variables and M clauses, each clause μ involving K variables indexed in $\partial\mu \subseteq \{1, \dots, N\}$, $|\partial\mu| = K$. We call $\alpha = M/N$ the clause density. In the notation of this section, variables are denoted by σ_i, $i = 1, \dots, N$, and take values in $\{-1, +1\}$. To each clause μ is associated a set of couplings, one for each variable in $\partial\mu$. The coupling $J_{\mu i}$ takes the value -1 if the variable i is negated in clause μ, and $+1$ otherwise. A configuration $\{\sigma_i\}_{i=1}^N$ satisfies the clause μ if $\exists i \in \partial\mu$ such that $J_{\mu i} = \sigma_i$. An instance of the problem is said to be satisfiable if there exists a configuration which satisfies all clauses. The Hamiltonian for this problem, which counts the number of violated clauses, is then given by:

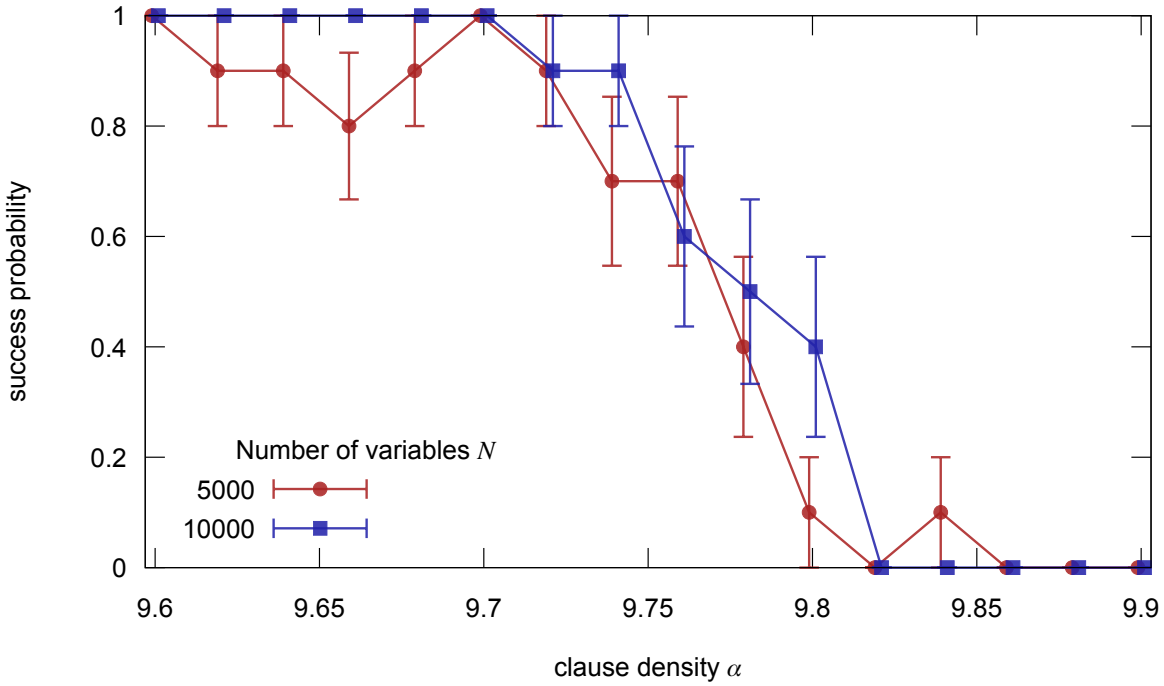


Figure S6: Probability of finding a solution in random instances of 4-SAT as a function of clause density α using the fBP algorithm described in the text. The results are shown for two values of N, slightly shifted relative to each other to improve readability. The points and error bars represent averages and standard deviations over 10 samples.

$$H(\sigma) = \sum_{\mu=1}^{M} \prod_{i \in \partial\mu} \frac{1 - J_{\mu i}\sigma_i}{2}. \tag{S58}$$

Random instances of the problem are generated by sampling uniformly and independently at random the subsets $\partial\mu$ and choosing the couplings $J_{\mu i}$ independently in $\{-1, +1\}$ with equal probability. Random instances of K-SAT are locally tree-like in the large N limit, therefore the problem is suitable to investigation through the tools of statistical physics of disordered systems, namely the cavity method [10]. Such investigation in fact led in the past decades to outstanding theoretical advances in understanding the structure of the problem [10, 14] and to devise state-of-the-art solvers [15].

In order to apply the cavity method to K-SAT one identifies each clause with a factor node and each variable with a variable node; it is then convenient to parametrize clauses-to-variables cavity messages with $P_{\mu\to i}(\sigma_i = J_{\mu i}) \equiv \eta_{\mu\to i}$, and variables-to-clauses ones with $P_{i\to\mu}(\sigma_i \neq J_{\mu i}) \equiv \zeta_{i\to\mu}$. We also define the variables' subsets $\partial^+ i$ and $\partial^- i$ as $\partial^\pm i = \{\mu \in \partial i \ : \ J_{\mu i} = \pm 1\}$. With these definitions, the zero temperature BP update rules take the form:

$$\pi^t_{\pm,i\to\mu} = \tilde{\pi}^t_{\pm,i} \prod_{\nu \in \partial^\pm i \setminus \mu} \eta^t_{\nu\to i}, \tag{S59}$$

$$\zeta^t_{i\to\mu} = \frac{\pi^t_{J_{\mu i},i\to\mu}}{\pi^t_{+,i\to\mu} + \pi^t_{-,i\to\mu}}, \tag{S60}$$

$$\eta^{t+1}_{\mu\to i} = 1 - \prod_{j \in \partial\mu \setminus i} \zeta^t_{j\to\mu}. \tag{S61}$$

Magnetizations at time t are given by:

$$m^t_i = \frac{\tilde{\pi}^t_{+,i} \prod_{\nu\in\partial^+ i} \eta^t_{\nu\to i} - \tilde{\pi}^t_{-,i} \prod_{\nu\in\partial^+ i} \eta^t_{\nu\to i}}{\tilde{\pi}^t_{+,i} \prod_{\nu\in\partial^+ i} \eta^t_{\nu\to i} + \tilde{\pi}^t_{-,i} \prod_{\nu\in\partial^+ i} \eta^t_{\nu\to i}}. \tag{S62}$$

The update rules of the coefficients $\tilde{\pi}^t_{\pm,i}$ depend on the algorithm under consideration. In standard BP, the coefficients $\tilde{\pi}^t_{\pm,i}$ are set to:

$$\tilde{\pi}^t_{\pm,i} \equiv 1. \tag{S63}$$

A common procedure for finding solutions for the K-SAT problem, called BP guided decimation (BPGD), consists in iterating Eqs. (S59-S61) until convergence; computing marginals according to Eq. (S62); fixing a certain fraction of variables, the ones with most polarized marginals, to $\sigma_i = \mathrm{sgn}\,(m_i)$; repeating the procedure on the reduced problem, until a satisfying configuration is found or a contradiction is produced. (A contradiction occurs when all variables involved in a clause are fixed to values that violate the clause.)

We can avoid to irrevocably fix the variables at intermediate steps of the algorithm, as in BPGD, using instead the reinforced BP (rBP) heuristic, which acts as a "soft" decimation. For rBP the new update rule reads:

$$\tilde{\pi}^{t+1}_{\pm,i} = \left(\pi^t_{\pm,i}\right)^\rho. \tag{S64}$$

In rBP, Eqs. (S59-S61) and (S64) are iterated, while the coefficient ρ is increased up to one. The algorithm stops when clamping the magnetizations gives a solution, or a contradiction is produced.

For fBP, instead, the update rules are derived from the robust ensemble and are similar to the ones for neural networks:

$$m^t_{i\to\star} = \frac{\prod_{\nu\in\partial^+ i}\eta^t_{\nu\to i} \;-\; \prod_{\nu\in\partial^+ i}\eta^t_{\nu\to i}}{\prod_{\nu\in\partial^+ i}\eta^t_{\nu\to i} \;+\; \prod_{\nu\in\partial^+ i}\eta^t_{\nu\to i}}, \tag{S65}$$

$$m^{t+1}_{\star\to i} = \tanh\left((y-1)\tanh^{-1}\left(m^t_{i\to\star}\tanh\gamma\right)\right)\tanh\gamma, \tag{S66}$$

$$\tilde{\pi}^{t+1}_{\pm,i} = 1 \pm m^{t+1}_{\star\to i}. \tag{S67}$$

We performed some preliminary tests of the effectiveness of fBP as a solver in K-SAT (see Fig. S6), to be expanded in future investigations. We used the following protocol: we iterated Eqs. (S59-S61,S65-S67) at fixed $y = 6$ and increasing γ in steps of 0.01 every 2000 iterations, starting from $\gamma = 0.01$. The algorithm is stopped when a solution is found (by clamping to ±1 the cavity marginals) or when $\gamma = 0.6$ is reached without finding any solution. The BP update equations often converged to a fixed point before reaching 2000 iterations, in which case we skipped to the next value of γ; in order to aid convergence, we added some damping to the iterative procedure.

We didn't test extensively the possible heuristic schemes to turn fBP into an efficient solver, therefore there is surely room for improvement over the one adopted here. Nonetheless, our procedure—when successful—found solutions quite rapidly, even though it is not as fast as rBP. More importantly, it finds solutions up to high values of α. Defining $\alpha_d = 9.38$, $\alpha_c = 9.547$ and $\alpha_s = 9.93$ the dynamic, static and SAT/UNSAT transitions respectively in 4-SAT [14], from Fig. S6 it appears that the algorithmic threshold for fBP is $\alpha_c < \alpha_{fBP} < \alpha_s$. In comparison, BPGD starts to fail much earlier, $\alpha_{BPGD} \approx \alpha_d$ [10]. The fBP performance seems in fact to be rather close to that of the "Survey Propagation with Backtracking" algorithm of [16] (in particular, the curves in Fig. 1 in that paper can be directly compared and appear to be similar to those in Fig. S6 in the present paper), although a much more extensive and detailed analysis (like the one performed in [16]) would be required in order to obtain a good estimate of the algorithmic threshold of fBP in the large N limit.

[1] Carlo Baldassi. A method to reduce the rejection rate in Monte Carlo Markov Chains on Ising spin models. *http://arxiv.org/abs/1608.05899*, 2016.

[2] Carlo Baldassi, Alfredo Braunstein, Nicolas Brunel, and Riccardo Zecchina. Efficient supervised learning in networks with binary synapses. *Proceedings of the National Academy of Sciences*, 104:11079–11084, 2007.

[3] Carlo Baldassi. Generalization learning in a perceptron with binary synapses. *J. Stat. Phys.*, 136:902–916, 2009.

[4] Carlo Baldassi, Alessandro Ingrosso, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Subdominant Dense Clusters Allow for Simple Learning and High Computational Performance in Neural Networks with Discrete Synapses. *Physical Review Letters*, 115(12):128101, September 2015.

[5] GJ Mitchison and RM Durbin. Bounds on the learning capacity of some multi-layer networks. *Biological Cybernetics*, 60(5):345–365, 1989.

[6] Implementation of the Replicated Stochastic Gradient Descent algorithm for binary committee machines. https://github.com/carlobaldassi/BinaryCommitteeMachineRSGD.jl.

[7] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 685–693. Curran Associates, Inc., 2015.

[8] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[9] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.

[10] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, January 2009.

[11] Alfredo Braunstein and Riccardo Zecchina. Learning by message-passing in neural networks with material synapses. *Phys. Rev. Lett.*, 96:030201, 2006.

[12] Implementation of the Focusing Belief Propagation algorithm for binary committee machines. https://github.com/carlobaldassi/BinaryCommitteeMachineFBP.jl.

[13] Carlo Baldassi, Federica Gerace, Carlo Lucibello, Luca Saglietti, and Riccardo Zecchina. Learning may need only a few bits of synaptic precision. *Physical Review E*, 93(5):052313, May 2016.

[14] Florent Krzakala, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborova. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007.

[15] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[16] Raffaele Marino, Giorgio Parisi, and Federico Ricci-Tersenghi. The backtracking survey propagation algorithm for solving random K-SAT problems. *http://arxiv.org/abs/1508.05117*, 2015.