Deep Tempering with Nested Restricted Boltzmann Machines

Clément Roussel, Jorge Fernandez-De-Cossio-Diaz, Simona Cocco, Rémi Monasson

Deep Tempering with Nested Restricted Boltzmann Machines

Clément Roussel, Jorge Fernandez-de-Cossio-Diaz, Simona Cocco, Rémi Monasson
Laboratory of Physics of the Ecole Normale Supérieure, CNRS UMR 8023 & PSL Research, Sorbonne Université, 24 rue Lhomond, 75005 Paris, France

Distributions of high-dimensional data can be learnt with unsupervised architectures, such as restricted Boltzmann machines (RBM). However, the resulting models are often uneasy to sample when the data distributions include multiple modes. We here consider deep tempering, a parallel-tempering-like Monte Carlo sampling algorithm based on a chain of several restricted Boltzmann machines (RBM), where hidden configuration of a machine can be exchanged with the visible configurations of the next one along the chain. Replica exchanges between the different RBM is facilitated by the increasingly clustered representations learnt by deeper RBMs along the chain, allowing for fast transitions between the different modes of the data distribution. We explain why deep tempering works on hierarchical data, and introduce a theoretical framework to understand how hyperparameters, such as the aspect ratios of the RBMs and the weight regularization should be chosen. Our findings are illustrated on two datasets: MNIST and in silico Lattice Proteins.

I. INTRODUCTION

Sampling complex energy landscapes is an important goal in statistical and computational physics. Following the introduction of Monte Carlo (MC) methods by Metropolis in the late 40's, several approaches have been considered to speed up sampling. Among them cluster-based algorithms are non-local moves in the configuration space that are able to update a large, possibly extensive number of microscopic variables at once [1]. Cluster algorithms, such as Wolff's algorithm [2] were shown to be extremely efficient to circumvent the so-called critical slowing down phenomenon accompanying second-order phase transitions, *e.g.* in the Ising model. However cluster algorithms implicitly require a deep physical knowledge of the systems, in particular of its ground states. In the case of disordered systems, where the low-energy states are very numerous, differ from each other by an extensive number of spins, and cannot be easily guessed from a direct inspection of the coupling matrix, this approach has not led to effective implementations.

Another improvement of standard MC methods, intensively used in particular in the context of disordered systems, is parallel tempering, also called replica exchange MC [3]. Parallel tempering consists in simulating more than one copy of the system, at higher temperatures than the target temperature of interest. These replicated systems are likely to be easier to sample, especially at very high temperatures for which the effective barriers in the energy landscape are low and easy to cross. The idea is then to allow for exchange of configurations between copies of the system thermalized at different temperatures. Hence, low-temperature systems will benefit from the capability of high-temperature systems to quickly explore the configuration space, instead of getting indefinitely stuck in the landscape valleys. The procedure requires that the exchange, which must satisfy detailed balance, has a reasonable probability of occurring, which implies that the two temperatures should not be too far away from one another. From a conceptual point of view, let us distinguish the idea of having a chain of different systems with slowly changing Hamiltonians, which is key to parallel tempering, with the standard implementation in which all these Hamiltonians are identical up to global rescalings encoding the temperatures of the systems. The idea of parallel tempering is more general than this standard and simple implementation, as we shall see below.

In this work we report on an application of the ideas of parallel tempering to the context of data-driven modeling. We use restricted Boltzmann machines (RBM), a paradigm of unsupervised architectures, to learn probability distributions from data. RBMs are graphical models, extracting representations from data, and in turn, able to generate new data from representations. They are known to be universal approximators, *i.e.* they can approximate any distribution over the visible variables when the size of the representation layer goes to infinity [4]. The distributions learnt by RBM may be complex and multimodal, and hence difficult to sample. Inspired by deep tempering, an approach introduced in the context of deep belief networks [5], we show how to learn stacks of nested RBMs, using the representations of a RBM as 'data' for the next one along the stack. Informally speaking, these RBMs learn more and more simplified versions of the true distributions, which become increasingly easier to sample with standard MC dynamics. We then couple the RBMs by allowing them to exchange configurations. These exchanges are made possible by the nested structure of the stack, *i.e.* the compatibility between the sizes of the layers of contiguous RBMs. We show numerically on two data sets, MNIST [6] and in silico proteins [7, 8], that the resulting procedure in much faster than standard MC for sampling the learnt distributions. We also study analytically the performance of our sampling algorithm on an analytically tractable distribution of hierarchically arranged data.

II. MODELS AND DATASETS

A. Restricted Boltzmann Machines

1. Definitions

Restricted Boltzmann Machines are undirected probabilistic models with two layers. A visible layer $\mathbf{v}$, which represents the data, is connected to a hidden layer $\mathbf{h}$ through a weight matrix W, see Fig. 1(a). There are no couplings between units within the same layer. The visible layer includes N visible units v_i and the hidden layer includes M hidden units h_μ. For simplicity we assume throughout this work that visible units take binary values, either 0,1 (Bernoulli) or ± 1 (similar to spins in statistical physics). For a given visible configuration $\mathbf{v} = \{v_i\}_{i=1\ldots N}$ and hidden configuration $\mathbf{h} = \{h_\mu\}_{\mu=1\ldots M}$, the joint probability distribution of the RBM is

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\, \exp\left(-E(\mathbf{v},\mathbf{h})\right) , \qquad (1)$$

where the energy E is defined as

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{N}\sum_{\mu=1}^{M} W_{i\mu} v_i h_\mu - \sum_{i=1}^{N} g_i\, v_i - \sum_{\mu=1}^{M} c_\mu\, h_\mu \ , \qquad (2)$$

and where the parameters c_μ and g_i represent biases acting on, respectively, units h_μ and v_i.

The probability distribution $P(\mathbf{v})$ of a visible configuration $\mathbf{v}$ can be computed by marginalizing over the hidden-unit configurations $\mathbf{h}$:

$$P(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v},\mathbf{h}) = \frac{1}{Z^v}\, \exp\left(-E^v(\mathbf{v})\right) , \qquad (3)$$

where

$$E^v(\mathbf{v}) \ = \ -\sum_{i=1}^{N} g_i\, v_i - \sum_{\mu=1}^{M} \Gamma_\mu(I_\mu(\mathbf{v})) \ . \qquad (4)$$

Here, $I_\mu(\mathbf{v}) = \sum_{i=1}^{N} W_{i\mu} v_i$ denotes the input received by hidden unit h_μ and $\Gamma_\mu(I) = \log \sum_h \exp\left[h\,(c_\mu + I_\mu(\mathbf{v}))\right]$ is simple to compute for binary unit h. A similar expression is obtained for the effective energy $E^h(\mathbf{h})$ corresponding to the log-probability of hidden configurations, obtained through marginalization over visible configurations.

2. Alternating Gibbs sampling

The bipartite nature of the RBM interaction graph suggests a simple sampling algorithm, called Alternating Gibbs Sampling (AGS) and depicted in Fig. 1(b). The basic observation underlying AGS is that the conditional distribution $P(\mathbf{h}|\mathbf{v})$ (respectively $P(\mathbf{v}|\mathbf{h})$) can be factorized over the hidden (respectively, visible) units. A step of AGS consists in

- Starting from a visible configuration $\mathbf{v}^t$ at step t, a hidden configuration $\mathbf{h}^{t+1}$ is drawn from $P(\mathbf{h}|\mathbf{v}^t) = \prod_{\mu=1}^{M} P(h_\mu|\mathbf{v}^t)$. Here $P(h_\mu|\mathbf{v}) \propto \exp\left[h_\mu(c_\mu + I_\mu(\mathbf{v}))\right]$. This step can be seen as a stochastic extraction of features from the configuration $\mathbf{v}^t$.

- A new visible configuration $\mathbf{v}^{t+1}$ is drawn from $P\left(\mathbf{v}|\mathbf{h}^{t+1}\right) = \prod_{i=1}^{N} P(v_i|\mathbf{h}^{t+1})$. Here $P(v_i|\mathbf{h}) \propto \exp\left[v_i(g_i + I_i(\mathbf{h}))\right]$, where $I_i(\mathbf{h})$ is the input of the visible unit v_i, i.e $I_i(\mathbf{h}) = \sum_{\mu=1}^{M} W_{i\mu} h_\mu$. This step can be seen as a stochastic reconstruction of $\mathbf{v}$ from the representation $\mathbf{h}^{t+1}$.

It is important to stress that the simplicity of AGS does not imply it is efficiently sampling the visible and hidden configuration spaces. In a recent work some of us have shown that AGS is generally not more efficient than standard Metropolis sampling of $P(\mathbf{v})$ in the visible space, and is unable to sample distributions with multiple modes separated by large energy barriers [9].

3. Training

The RBM is defined by its weights $W = \{w_{i\mu}\}$, as well as some variables parametrizing the classes of potentials $\mathcal{U}_\mu$ and $\mathcal{V}_i$ considered. All these parameters, generically denoted by Θ must be learned from the training data, consisting of a set of P samples $\{\mathbf{v}^p\}_{p=1\ldots P}$. To do so, one looks for the maximum of the log-likelihood of the data, $LL(\Theta) = \frac{1}{P}\sum_{p=1}^{P} \log P(\mathbf{v}^p) \equiv \langle \log P(\mathbf{v})\rangle_{\rm data}$. Maximization of LL over Θ is done through gradient ascent. The generic expression for the gradients is

$$\frac{\partial LL}{\partial \Theta} = -\left\langle \frac{\partial E(\mathbf{v})}{\partial \Theta} \right\rangle_{\rm data} + \left\langle \frac{\partial E(\mathbf{v})}{\partial \Theta} \right\rangle_{\rm model} , \qquad (5)$$

where $\langle . \rangle_{\rm data}$ denotes the expected value over the data $\{\mathbf{v}^p\}$ and $\langle . \rangle_{\rm model}$ over the model distribution $P(\mathbf{v})$. Notice it is easy to include regularization in the procedure above, e.g. through shifting $LL(\Theta) \to LL(\Theta) - \gamma||\Theta||^2$ in the case of L_2-based penalty over the model parameters.

In general, the exact evaluation of the gradient is intractable numerically because the expected value over the model is an average over an exponential number of terms. Markov chain Monte Carlo methods approximate the expected value over the model. Different algorithms, based on alternating Gibbs sampling between the visible and hidden layers, are used to generate samples from $P(\mathbf{v})$, such as Contrastive Divergence [10], or Persistent Contrastive Divergence [11].

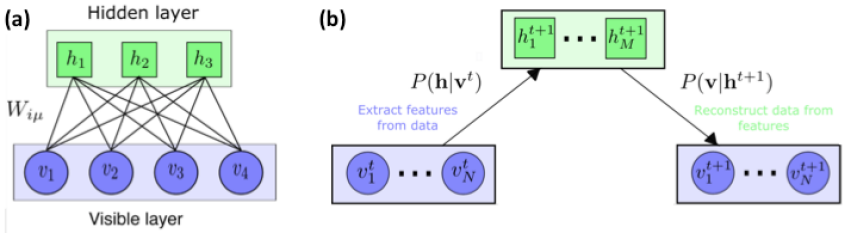

FIG. 1. Architecture (a) and sampling (b) of Restricted Boltzmann Machines.

B. Datasets

In this paper, we use different datasets to illustrate the performance of our sampling procedure.

1. *MNIST0/1*

MNIST dataset [6] is a large dataset of 28×28 pixel images of handwritten digits. We use the binarised version of MNIST (pixels are white or black), and limit ourselves to zero and one digits only, see Fig. 2(a). RBMs learnt on this restricted dataset, hereafter referred to as MNIST0/1, are empirically known to be harder to sample than the ones trained on full MNIST. The lack of similarities between 0 and 1 digits make transitions between the two classes very rare: a RBM initialized with either digit is likely to generate many variants of the same digit.

2. *Lattice Proteins*

The Lattice Protein models [7, 8] are artificial proteins used to investigate protein folding. Proteins are sequences of amino acids, and their 3d structures encode their functionalities. Predicting the 3d structure from the sequence of amino acids is a crucial challenge in biology. In this model, a structure is defined as a self-avoiding path of 27 amino-acid-long chains ($\mathbf{v}$ represents a protein) on the $3 \times 3 \times 3$ lattice cube. The lattice cube defines a set of $\mathcal{N} = 103,406$ distinct structures. For a given structure, there are 28 contacts between the amino acids. The probability of a sequence $\mathbf{v}$ to fold in a given structure S is expressed as

$$P_{\rm nat}(\mathbf{v}|S) = \frac{\exp\left(-E_{LP}(\mathbf{v},S)\right)}{\sum\limits_{S'} \exp\left(-E_{LP}(\mathbf{v},S')\right)} \ , \qquad (6)$$

where the sum runs over all N possible structures S', and the energy of the sequence $\mathbf{v}$ in a structure S is given by:

$$E_{LP}(\mathbf{v},S) = \sum_{i<j} c_{ij}^S \ \Delta E_{MJ}(v_i,v_j) \qquad (7)$$

In the previous formula, $c_{ij}^S = 1$ if the sites i and j are in contact in the fold S; Otherwise, $c_{ij}^S = 0$. The pairwise energy $\Delta E_{MJ}(v_i,v_j)$ is defined as the Miyazawa-Jernigan (MJ) potential, which is a proxy for the physico-chemical interactions between nearby amino acids in known structures [12].

Given a fold S we define the protein family associated to S as the set of sequences $\mathbf{v}$ such that $P_{\rm nat}(\mathbf{v}|S) > 0.99$. Here, we choose two structures, S_A and S_B shown in Fig. 2(b), and the two corresponding protein families, characterized in [13]. Due to the absence of similarity between the two folds, a RBM trained on a mixture of sequence data extracted from the two families samples diverse sequences with high $P_{\rm nat}$ in one class only, corresponding to the initial condition of the sampling dynamics. Transition from one to the other are extremely rare.

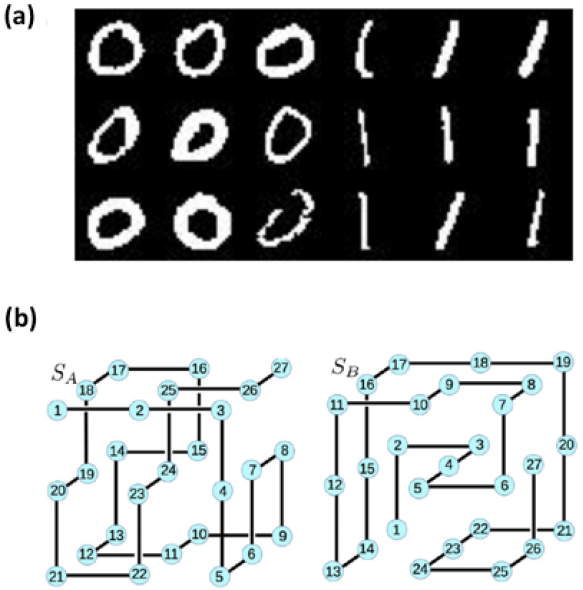

FIG. 2. (a) MNIST0/1: examples of 0 and 1 digits. Our dataset contains $P = 6000$ 0's and the same number of 1's. (b) Lattice Protein models: structures S_A and S_B defining the two families. Each family is represented by $P = 45,000$ sequences with folding probability > 0.99.

III. DEEP TEMPERING: PROCEDURE AND ILLUSTRATIONS

We now present the deep tempering procedure, based on the introduction of a nested set of RBMs recursively extracting relevant collective modes of their predecessors, and, in turn, easing their sampling. While this algorithm was first introduced in [5] to train deep belief networks [10, 14, 15] we use it here to sample a single RBM, which we refer to as bottom RBM. We then qualitatively illustrate the operation of deep tempering and the representations learned by nested RBMs when data are organized into a perfect hierarchical tree. Last of all we show how deep tempering efficiently samples (bottom) RBM trained on MNIST0/1 and Lattice Protein data, and largely outperforms standard alternating Gibbs sampling.

A. Architecture and training

The architecture supporting deep tempering is sketched in Fig. 3(a). It consists of a stack of RBMs, whose widths are such that the number M_n of hidden units of the n^{th} RBM equals the number N_n of visible units of the $(n+1)^{th}$ one. This constraint is essential to make communication between RBM possible, as explained below. In addition, we choose the sizes of visible layers to decrease with the index n, *i.e.* $N_n > N_{n+1}$, making deeper RBM 'simpler' than their predecessors in the stack.

The RBM of interest, which is trained on the data $\mathbf{v}^p$, $p = 1, ..., P$, and is sampled to generate new data is the bottom one, and corresponds to $n = 1$. We use standard maximum-likelihood learning procedure, see Section II A 3 for training. Note that there is no inconsistency between the capability of training a RBM, which heavily relies on training, and the hardness of sampling it at later times. Training extensively relies on the availablity of data to initialize the dynamical chains in constrative or persistent contrastive divergence, without consideration about mixing between the modes of the data distribution.

After the training of the bottom RBM, a set of hidden representations $\{\mathbf{h}_1^p\}$ of dimension M_1 are stochastically drawn from the conditional probabilities $P_1(\mathbf{h}|\mathbf{v}^p)$ for every $p = 1, ..., P$. These hidden representations are then considered as data configurations $\{\mathbf{v}_2^p\}$ for the next RBM, $n = 2$. This is possible since the dimension of the visible layer of the second RBM, N_2, is equal to M_1. The second RBM is then trained by maximizing the log-likelihood of the 'data' $\{\mathbf{v}_2^p\}_{p=1...P}$. After training of the second RBM a set of representations $\{\mathbf{h}_2^p\}$ of the 'data' $\{\mathbf{v}_2^p\}$ are drawn, which are then used for training the third RBM. The process can be iterated all the way up to the last RBM.

Once all RBM are trained the marginal distributions of their configurations are fully defined. We hereafter denote by, respectively, $P_n^v \propto \exp(-E_n^v)$ and $P_n^h \propto \exp(-E_n^h)$ the marginal distributions over the visible and hidden configurations of the n^{th} RBM.

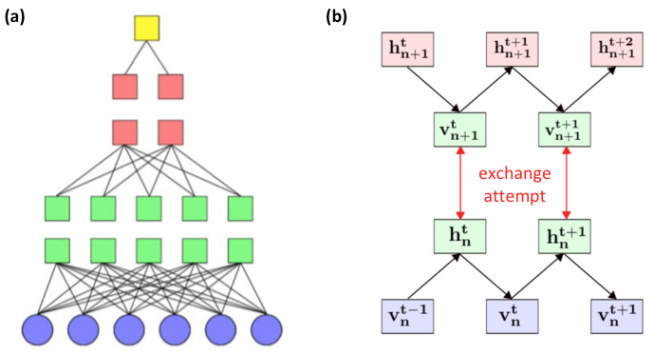

FIG. 3. Principle of deep tempering. (a) Example with three RBMs. The number of visible units of the $(n+1)^{th}$ RBM is equal to the number of hidden units of the n^{th} RBM. Data are learnt and generated by the bottom RBM. Deeper RBMs iteratively detect relevant collective modes of hidden units of the bottom RBM, hence facilitating its sampling. (b) Illustration of deep tempering (here, only the n^{th} and $(n+1)^{th}$ RBMs are shown). Alternating Gibbs Sampling is used to generate visible and hidden configurations in each RBM. At any step t the configurations $\mathbf{v}_{n+1}^t$ on the visible layer of the $(n+1)^{th}$ RBM and $\mathbf{h}_n^t$ on the hidden layer of the n^{th} RBM can be exchanged, with probability $A_n(\mathbf{h}_n^t, \mathbf{v}_{n+1}^t)$, see Eq. 8.

B. Sampling and exchanges

Each one of the RBMs can be sampled with Alternating Gibbs sampling, which consists of a stochastic chain $\mathbf{h}_n^t \sim P_n\left(\mathbf{h}|\mathbf{v}_n^{t-1}\right)$ and $\mathbf{v}_n^t \sim P_n\left(\mathbf{v}|\mathbf{h}_n^t\right)$. After each Gibbs step t one attemps to exchange the visible configuration $\mathbf{v}_{n+1}^t$ of the $(n+1)^{th}$ RBM and the hidden configuration $\mathbf{h}_n^t$ of the n^{th}, see Fig. 3(b). These two configurations are swapped with probability:

$$A_n(\mathbf{h}_n^t, \mathbf{v}_{n+1}^t) = \min\left(1, \frac{P_{n+1}^v(\mathbf{h}_n^t) P_n^h(\mathbf{v}_{n+1}^t)}{P_{n+1}^v(\mathbf{v}_{n+1}^t) P_n^h(\mathbf{h}_n^t)}\right) = \min\left(1, \frac{\exp\left(-E_{n+1}^v(\mathbf{h}_n^t) - E_n^h(\mathbf{v}_{n+1}^t)\right)}{\exp\left(-E_{n+1}^v(\mathbf{v}_{n+1}^t) - E_n^h(\mathbf{h}_n^t)\right)}\right) \quad (8)$$

The definition of the acceptance probability A_n ensures that detailed balance is satisfied. Crucially it does not depend on the intractable normalizations (Z factors) of P_{n+1}^v, P_n^h, but only on the effective energies E_{n+1}^v, E_n^h that are easy to compute.

The expression of A_n is reminiscent of the one used in the parallel tempering algorithm [16, 17], in which configurations sampled from an energy landscape at two (or more) temperatures can be exchanged. Here, exchange is possible between configurations $\mathbf{h}_n^t$ and $\mathbf{v}_{n+1}^t$ sampled from the energy landscapes $E_n^h(\mathbf{h})$ and $E_{n+1}^v(\mathbf{v})$. As the number of visible units is decreasing with the depth,

see Fig. 3(a), and the RBM parameters are regularized, the energy landscape E^v_{n+1} of the $(n+1)^{th}$ RBM is expected to be smoother than E^h_n. In accordance, as we shall see below, the diversity of hidden representations of the initial data progressively diminishes with the depth: the number of distinct configurations in $\{\mathbf{h}^p_n\}_{p=1\dots P}$ decreases with n.

Informally speaking, deeper RBM express simpler approximations of the landscape captured by the bottom RBM. They can therefore be easily sampled, and in turn help the bottom RBM to undergo non local moves in its complex landscape, without getting trapped in local minima or valleys. Contrary to the original version of deep tempering in [5] our objective is therefore not to sample accurate deep belief networks. By construction we here require our narrow and deeply nested RBMs to poorly represent the data at higher depths. Their role is to extract meaningful nested approximations of the data distribution.

C. Intuitive picture of deep tempering on hierarchically organized data

To provide support to the informal justification of deep tempering given above we consider a toy model of data $\mathbf{v}^k_1$, of dimension $N = 1000$, organized along a hierarchical tree. Data configurations define clusters, which can be divided into subclusters. This specific structure of configurations, called ultrametric, is reminiscent of spin glasses [18, 19]. Figure 4(a) depicts the correlation matrix of the data, with two main clusters divided into three subclusters.

We train several stacks of RBMs on these artificial data. Depending on the depth of the stacks and the number of hidden units of the machines, the hidden representations exhibit different behaviors, see Fig. 4(b,c,d). For the first RBM, the correlation matrix of $\mathbf{h}^k_1$ mimics the structure of the correlation matrix of the data $\mathbf{v}^k_1$, with a level of coarse graining depending on the number M_1 of hidden units:

- For M_1 of the same order as the number of clusters, all data points in a cluster are mapped onto the same hidden representation.
- For M_1 of the same order as the number of subclusters, all data points in a given subcluster have identical hidden representations, but this representation varies with the subcluster.
- For M_1 of the same order as the number of data points, each pattern has its own hidden representation.

Therefore, by tuning the number of hidden units, we can control the magnitude of the compression of the representations. If the compression is too important, *i.e.*, if the number of hidden units is too small, the RBM learns a poor representation of the data. $E^v_{n+1}(\mathbf{v})$ is a crude approximation of $E^h_n(\mathbf{h})$, the replica exchange rate is low and the dynamics of the RBMs are decoupled: in that case, Deep Tempering would be as efficient as Gibbs sampling. Fig. 4(d) shows an example where each RBM of the stack learns a different level of representation of the hierarchical tree: the bottom RBM learns one representation per pattern, the second RBM one representation per subcluster and the top RBM one representation per cluster. Fig. 4(e) exhibits schematic representations of the different landscapes learned by the RBMs in the stack. $E^v_1(\mathbf{v})$ has a local minimum per pattern, $E^v_2(\mathbf{v})$ per subcluster and $E^v_3(\mathbf{v})$ per cluster: $E^v_{n+1}(\mathbf{v})$ has to be a smooth approximation of $E^h_n(\mathbf{h})$ in order to have lower barriers while remaining a good approximation to keep the replica exchange rate high between the different RBMs.

D. Application to MNIST 0/1 and Lattice Proteins

We illustrate below these ideas on two different datasets: MNIST0/1 and Lattice Proteins. In these two examples, data are grouped into two distinct classes.

- For MNIST0/1, alternating Gibbs sampling is stuck in one the two digit classes, see Fig. 5(a). Conversely, deep tempering with a stack of four RBMs ($N_1 = 784$, $M_1 = N_2 = 200$, $M_2 = N_3 = 100$, $M_3 = N_4 = 25$, $M_4 = 10$) is able to jump from one class to the other in a very efficient way, see Fig. 5(b). The sampling algorithm with replica exchanges improves the mixing between the modes, as shown in Fig. 5(c,d).
- For Lattice Proteins, three RBMs are used ($N_1 = 27\times 20$, $M_1 = N_2 = 800$, $M_2 = N_3 = 50$, $M_3 = 25$). The sampling algorithm with replica exchanges improves the mixing between the two families and generates high-quality proteins, see Fig. 6(a). The sampled proteins also have high diversity: they are far away from the training data. Deep tempering efficicently mixes between the two families, while alternating Gibbs sampling cannot (Fig. 6(b)).

Notice that, contrary to the hierarchical model in Section III C), each class (0/1 or S_A/S_B) is not partitioned in well-defined clusters. Neverthless we remark that the different RBMs progressively compress the representations of the data, as the number of distinct hidden representations decreases with the depth. To evaluate the number of distinct hidden representations in the hidden layer, for each training sample $\mathbf{v}^p$ we can define its most probable configuration $\mathbf{h}^p_1 = \arg\max_{\mathbf{h}} P_1(\mathbf{h}|\mathbf{v}^p)$. We can also define the most probable hidden representation in the hidden layer of the n^{th} RBM as $\mathbf{h}^p_n = \arg\max_{\mathbf{h}} P_n(\mathbf{h}|\mathbf{h}^p_{n-1})$. We then compute the number of distinct representations in each hidden layer. Results are

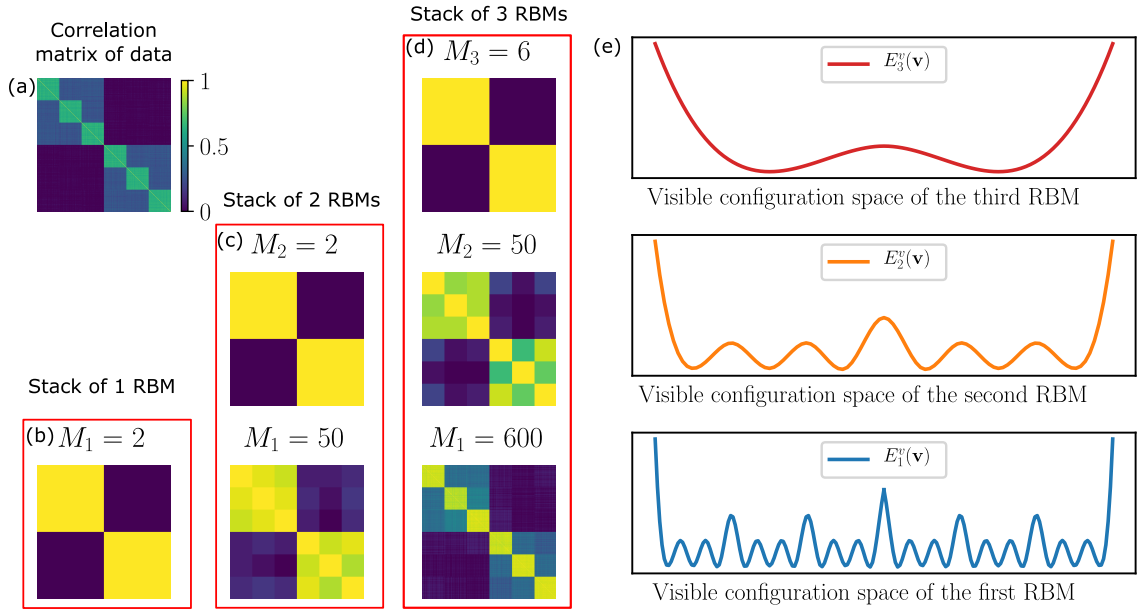

FIG. 4. The color map is the same for the different correlation matrices (from dark blue to yellow). (a) Correlation matrix of 600 visible patterns $\mathbf{v}_1^k$, $N = 1000$. The patterns are divided into two clusters. Each cluster has also three subclusters. (b) Correlation matrix of the hidden patterns $\mathbf{h}_1^k$. (c) From the bottom to the top: correlation matrices of the hidden patterns $\mathbf{h}_1^k$ and $\mathbf{h}_2^k$. (d) From the bottom to the top: correlation matrices of the hidden patterns $\mathbf{h}_1^k$, $\mathbf{h}_2^k$ and $\mathbf{h}_3^k$. (e) Schematic representation of the landscape $E_1^v(\mathbf{v})$, $E_2^v(\mathbf{v})$ and $E_3^v(\mathbf{v})$ learned by the different RBMs represented in the panel (d). The details of the landscape are progressively smoothing out but in the same time the free energy barriers between the different modes are decreasing.

shown in Table I for MNIST0/1 and in Table III D for Lattice Proteins S_A/S_B.

TABLE I. Number of distinct representations in the different hidden layers for MNIST0/1

	$H1$	$H2$	$H3$	$H4$
# configurations	12635	12560	3099	68

TABLE II. Number of distinct representations in the different hidden layers for Lattice Proteins S_A/S_B

	$H1$	$H2$	$H3$
# configurations	96234	39345	1273

IV. THEORETICAL ANALYSIS

We now seek to characterize the conditions on the data distribution and on the RBM hyperparameters allowing deep tempering to be efficient. To define an analytically tractable framework, we consider the case of strongly overparametrized RBM. Our analysis is organized along four steps:

A. the determination of the form of the coupling matrix after training a RBM in the overparametrized regime;

B. the derivation of the log-likelihood of the RBM in the case of structured data;

C. the identification of the pattern separation/pattern completion regime and phase transition depending on how much the RBM is regularized;

D. the estimation of the mixing and swapping times for a stack of two RBMs.

A. Learning K data points with a wide RBM: nature of the coupling matrix

We train a RBM with Contrastive Divergence on a dataset of K configurations $\mathbf{v}^k$, $k = 1...K$, whose components v_i^k, $i = 1...N$, are drawn independently and uniformly at random from $\{-1, +1\}$. Hence, these configurations are on average orthogonal. In addition, the weights $W_{i\mu}$ are initialized with small random values, drawn from a Gaussian distribution with zero mean and variance equal to $\frac{1}{N}$.

The outcome of the training procedure can be summarized as follows:

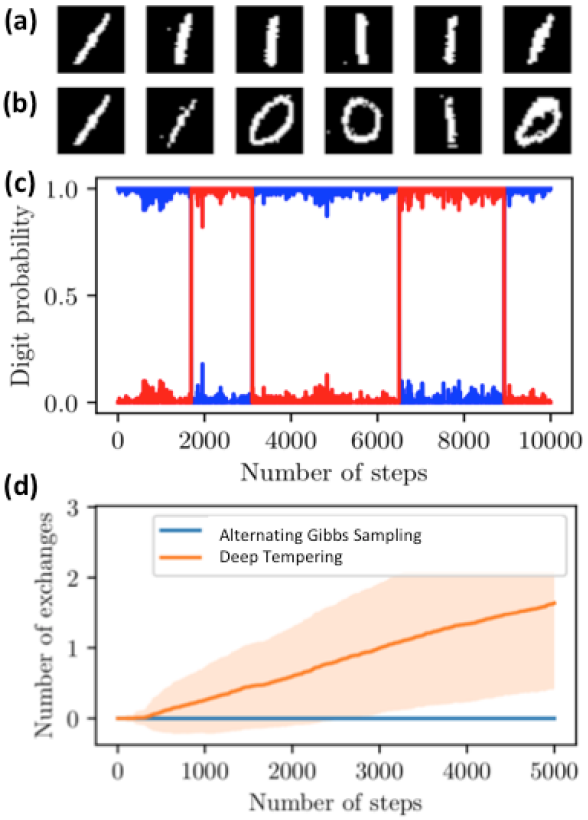

FIG. 5. Application of deep tempering to MNIST0/1. Example of sampled digits with (a) Alternating Gibbs Sampling and (b) Deep Tempering. Digits are displayed every 750 steps. The two dynamics start from the same initial configuration. (c) A random forest classifier is trained on MNIST0/1, and predicts the class of the visible configuration sampled by the RBM. Deep tempering generates high-quality digits and mixes well between the two classes. (d) Mean number of swaps between the two digits classes for the two sampling procedures. The initial configurations of the sampling dynamics are random digits of MNIST0/1. Averages are computed over 2,000 random initial configurations.

- The singular values σ_ℓ, with $\ell = 1 \ldots \min(N, M)$ of the coupling matrix W are clustered into two groups, see Fig. 7(a,b). All but (at most) K singular values decay with the learning time, down to a low level fixed by the intensity γ of the L_2 regularization.

- The left eigenvectors associated to the K relevant singular values span the same space as the K data points $\mathbf{v}^k$.

- The distribution of inputs

$$I_\mu(\mathbf{v}^k) \equiv \sum_i W_{i\mu}\, v_i^k \ , \tag{9}$$

is highly concentrated around two opposite values, which we refer to as $-w$ and w. The value of w is a decreasing function of γ.

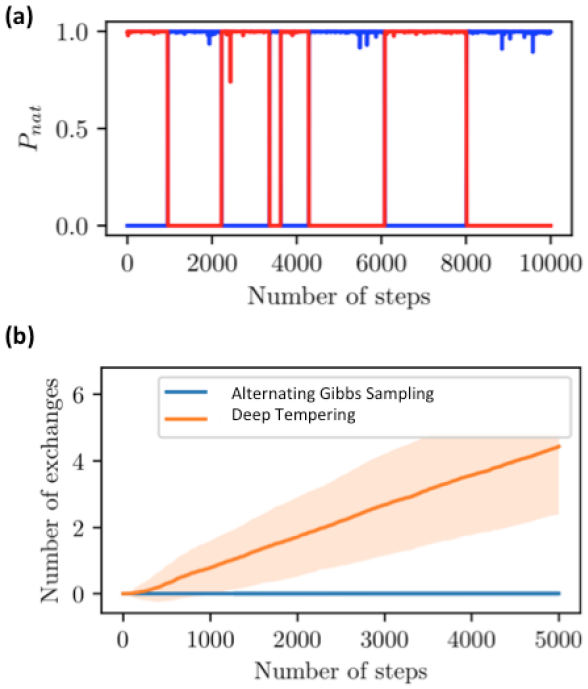

FIG. 6. Application of deep tempering to Lattice Proteins. (a) $P_{\rm nat}(\mathbf{v}|S)$ of sampled sequences for the two folds S_A (red) and S_B (blue). Deep Tempering algorithm generates high-quality sequences, folding either on S_A or S_B, and mixes well between the two families. (b) Mean number of swaps between the two families for the two dynamics. The initial configurations of the dynamics are random sequences in the training set. Averages are computed over 500 random initial configurations.

- In agreement with the statements above the coupling matrix can be approximately written as

$$W_{i\mu} = \frac{w}{N} \sum_{k=1}^{K} v_i^k h_\mu^k \ . \tag{10}$$

Due to the statistical orthogonality of the data points, $h_\mu^k \simeq I_\mu(\mathbf{v}^k)/w$ is equal to ± 1. This result is similar to the one considered by [20] although simpler, as we neglect noisy perturbation to the finite rank W in Eq. 10.

Notice that the direction of the vectors $\mathbf{h}^k$ is random and mostly determined by the initial coupling matrix, $W(t=0)$. Let us define the overlaps between the input vectors at times 0 and t,

$$C^{k,\ell}(t) = \frac{1}{M} \sum_\mu \text{sign}\left(\sum_i W_{i\mu}(0) v_i^k\right) \text{sign}\left(\sum_i W_{i\mu}(t) v_i^\ell\right) \tag{11}$$

and their 'diagonal' and 'off-diagonal' averages:

$$C_{\rm diag}(t) = \frac{1}{K} \sum_k C^{k,k}(t) \ , \ C_{\rm off}(t) = \frac{2}{K(K-1)} \sum_{k \neq \ell} C^{k,\ell}(t) \tag{12}$$

We show in Fig. 7(c) the behaviour of the diagonal and off-diagonal overlaps. We see that the inputs to the hidden units for a given data point remain close to their initial values, and essentially orthogonal to the one associated to another data configuration.

Last of all, the definition of the coupling matrix alone cannot break the symmetry between a data point, say, $\mathbf{v}^k$, and the opposite vector, $-\mathbf{v}^k$. This symmetry is broken during training through the learning of biases c^μ on the hidden units, see Eq. 2. These biases are strongly correlated with the sum of the $\mathbf{h}^k$ vectors, of the order of $N^{-1/2}$, see Fig. 7(d). While their presence is sufficient to ensure that, at the end of training, $P(\mathbf{v}^k) \gg P(-\mathbf{v}^k)$, but their magnitude is small enough to be neglected compared to the typical amplitude of the inputs I_μ.

B. Calculation of the log likelihood for structured data: a minimal setting

In the previous section, we have shown that K (statistically) orthogonal data points are learned by the RBM through a rank K coupling matrix, corresponding to K representations in one-to-one correspondence with the data configurations. However what happens if the data are structured, *i.e.* not statistically independent? Is the one-to-one mapping between data and representations maintained, or is there some form of clustering, in which similar data are mapped onto the same representation? To answer this question we consider the minimal case of $K = 2$ data configurations, $\mathbf{v}^1$ and $\mathbf{v}^2$, with an arbitrary overlap

$$x = \frac{1}{N}\sum_{i=1}^{N} v_i^1\, v_i^2 \ . \tag{13}$$

We learn these two data points with a RBM with $M = \alpha N$ hidden units (α is the aspect ratio) and L_2 regularization. According to the previous section we expect the weight matrix to be given by Eq. 10, where the two vectors $\mathbf{h}^1$ and $\mathbf{h}^2$ have dot product

$$y = \frac{1}{M}\sum_{\mu=1}^{M} h_\mu^1\, h_\mu^2 \ . \tag{14}$$

To determine the relationship between y and x, α, w we consider the log-likelihood of the data,

$$LL = \frac{1}{2}\big[\log P(\mathbf{v}^1) + \log P(\mathbf{v}^2)\big] \ , \tag{15}$$

where the probability of a visible configuration $\mathbf{v}$ is

$$\begin{aligned} P(\mathbf{v}) &= \frac{1}{Z^v}\sum_{\{h_\mu=\pm 1\}} \exp\left(\sum_{i,\mu} W_{i\mu}\, v_i\, h_\mu\right) \\ &= \frac{1}{Z^v}\prod_{\mu=1}^{M}\left(2\cosh\left(\sum_{i=1}^{N} W_{i\mu}\, v_i\right)\right) \end{aligned} \tag{16}$$

and Z^v is the partition function. Using expression in Eq. 10 for the coupling matrix W we readily obtain

$$\begin{aligned} LL &= \frac{M}{2}(1+y)\log\big(2\cosh\left(w(1+x)\right)\big) \\ &+ \frac{M}{2}(1-y)\log\big(2\cosh\left(w(1-x)\right)\big) - \log Z^v \end{aligned} \tag{17}$$

We are left with the computation of the partition function

$$\begin{aligned} Z^v &= \sum_{\{v_i,h_\mu=\pm 1\}} \exp\left(\sum_{i,\mu} W_{i\mu}\, v_i\, h_\mu\right) \\ &= \sum_{\{v_i,h_\mu=\pm 1\}} \exp\left[w\sum_{k=1,2}\left(\frac{1}{N}\sum_{i=1}^{N} v_i\, v_i^k\right)\left(\sum_{\mu=1}^{M} h_\mu\, h_\mu^k\right)\right] \ . \end{aligned} \tag{18}$$

According to the expression above the summation over the visible configurations $\mathbf{v}$ can be expressed as a four-dimensional integral over the overlaps

$$q^k = \frac{1}{N}\sum_{i=1}^{N} v_i\, v_i^k \ , \tag{19}$$

and their conjugated parameters, $\hat{q}^k$, for $k = 1, 2$. We write

$$\begin{aligned} Z^v &= \int \prod_{k=1,2}\frac{dq^k d\hat{q}^k}{2\pi/N}\sum_{\{h_\mu\}}\exp\left(w\sum_\mu h_\mu\sum_k q^h\, h_\mu^h\right)\times\sum_{\{v_i\}}\exp\left(\sum_k \hat{q}^k\big(\sum_i v_i^k\, v_i - N\, q^k\big)\right) \\ &= \int \prod_{k=1,2}\frac{dq^k d\hat{q}^k}{2\pi/N}\, e^{-N\sum_k q^k\hat{q}^k}\prod_{\sigma=\pm 1}\Big(2\cosh\big(w(q^1+\sigma\, q^2)\big)\Big)^{\frac{M}{2}(1+\sigma y)}\times\prod_{\tau=\pm 1}\Big(2\cosh\big(\hat{q}^1+\tau\,\hat{q}^2\big)\Big)^{\frac{N}{2}(1+\tau x)} \ . \end{aligned} \tag{20}$$

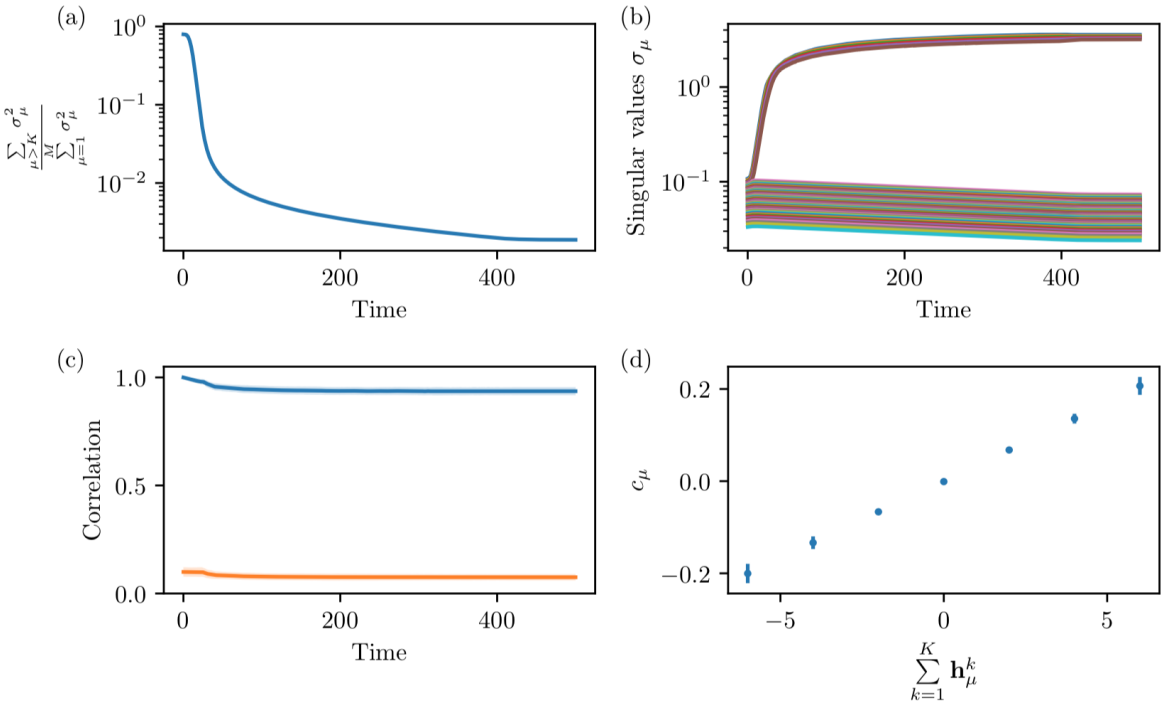

FIG. 7. Evolution of the coupling matrix during learning. Numerical experiments are realized with $N = 200$, $M = 60$, $K = 6$, and with L_2 regularization on the weights. Quantities of interests are averaged over 50 different initialization of K random data points and of the initial coupling matrix W_0. (a) Normalized sum of the $M - K$ smallest singular values of the coupling matrix W as a function of the learning time t. (b) Evolution of the singular values through time. K singular values emerge from the bulk. The other $M - K$ singular values decrease over time due to the L_2 regularization. (c) Blue line: diagonal overlap $C_{\text{diag}}(t)$, see Eq. 12. Orange line: off-diagonal overlap $C_{\text{off}}(t)$. (d) Mean value of the field $c_m u$ acting on hidden unit μ vs. average value of the data representations on this unit.

Defining $\Delta_\pm = q^1 \pm q^2$ and $\hat{\Delta}_\pm = \hat{q}^1 \pm \hat{q}^2$, we obtain, in the $N, M \to \infty$ limit (at fixed ratio α),

$$\frac{1}{N}\log Z^v = \frac{1}{2}\max_{\Delta_\pm,\hat{\Delta}_\pm}\Big[-\Delta_+\hat{\Delta}_+ - \Delta_-\hat{\Delta}_- + \alpha(1+y)\log\left(2\cosh\left(w\Delta_+\right)\right) + \alpha(1-y)\log\left(2\cosh\left(w\Delta_-\right)\right) + (1+x)\log\left(2\cosh\hat{\Delta}_+\right) + (1-x)\log\left(2\cosh\hat{\Delta}_-\right)\Big]. \tag{21}$$

Equations 17 and 21 give access to the log-likelihood LL of the data as a function of the control parameters. Optimization of LL over y yields the overlap between the $\mathbf{h}^k$ vectors as a function of the the overlap x between the data, the aspect ratio α and the amplitude w of the weight (which is, in practice, tuned through L_2 regularization).

C. Representational regimes and phase transition

The optimal overlap $y^*(x, \alpha, w)$ between the hidden representations of the data points is the one maximizing LL. Due to the parity of LL under the change of signs of x and y (at fixed α, w) we assume with no loss of generality that $x > 0$.

The log-likelihood LL given by Eqs. 17 and 21 can easily be optimized numerically with respect to y. We first find that $y^*(x, \alpha, w)$ is a decreasing function of the parameters α and w at fixed data overlap x, see Fig. 8(a,b). In other words, regularizing more the RBM, by decreasing either the amplitude w of the weights or the number of hidden units (equivalently, the aspect ratio α), produces more similar representations. For strong enough regularization, *i.e.* low enough α or w, we obtain $y^* = 1$, showing that the two representations of the data points become identical though $x < 1$.

We then plot y^* as a function of x in Fig. 8(c). We identify three different regimes:

- At low data overlaps, y^* is a slowly increasing function of x, with a slope smaller than unity. The RBM has a tendency to produce representations

less correlated than the data. This regime can be referred to as *pattern separation*, a vocable used in the context of neuroscience.

- At intermediate data overlaps, y^* is a quickly increasing function of x, with a slope larger than unity. The RBM has a tendency to produce representations more correlated than the data. This regime is reminiscent of *pattern completion*.

- At large data overlaps, for $x > x_c$, $y^* = 1$: the RBM has mapped similar but distinct data onto a unique representation. This regime can be referred to as *clustering* and is an extreme version of pattern completion. The value of x_c is computed in Appendix A.

As shown in Fig. 8 these theoretical predictions are in good agreement with direct estimation of the probabilities $P(\mathbf{v}^1)$ and $P(\mathbf{v}^2)$ (and thus of the log-likelihood LL), bypassing the calculation done in Section IV B. To estimate the partition function Z^v appearing in P we use Annealed Importance Sampling (AIS) [15, 21]. In practice, we sample random vectors $\mathbf{v}^1$, $\mathbf{v}^2$ with correlation x. For $y \in [0, 1]$, we then sample random vectors $\mathbf{h}^1$, $\mathbf{h}^2$ with a correlation y, and define the weight matrix according to Eq. 10. The partition function is then evaluated with AIS. The $[0, 1]$ range over y is discretized to locate the maximum of LL with sufficient accuracy, and the procedure is repeated 25 times for each x. Dots in Fig.8 are the mean value of the optimal y, and the shaded areas show the standard deviation of the optimal y.

The boundary between the pattern separation and completion regimes is a genuine phase transition, characterized by the onset of a non-zero order parameter, here $\Delta_- = q_2 - q_1$. In intermediate-to-large x regime, $\Delta_- = 0$, while $\Delta_- > 0$ at low x. This phase transition has a concrete interpretation in terms of the generative diversity of the RBM. Once the training is done we can use our RBM to generate 'new' data through Alternating Gibbs Sampling. In the $\Delta_- = 0$ regime, whatever the initial configuration $\mathbf{v} = \mathbf{v}^1$ or $\mathbf{v}^2$ on the visible layer, the RBM will generate the same data distribution. In the $\Delta_- > 0$ regime, there is ergodicity breaking between the two possible initial conditions, and the RBM will generate data similar to the data point initially present on the layer only.

As a final note let us emphasize that the representation regimes and phase transition identified here is not specific to the case $K = 2$. The analytical calculation of the log-likelihood LL done above can be extended to any finite $K > 2$ (while N, M are sent to infinity). Analysis of the extremization equations for K generic data points shows that, as α and w are progressively decreased, a sequence of clustering-like phase transitions takes place, with more and more similar representations. The number of distinct representations varies from K to 1 as the regularization is made stronger.

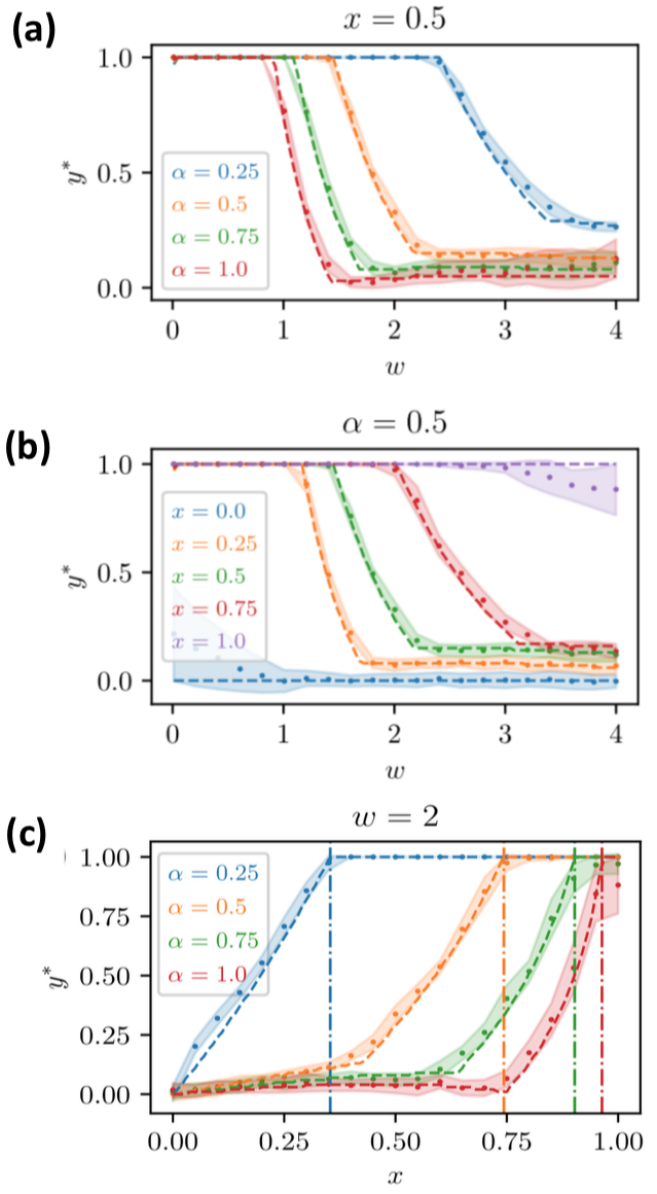

FIG. 8. Optimal overlap $y^*(w, x, \alpha)$. Dashed lines: theoretical results obtained by maximizing LL with respect to y. Dots: numerical estimate of y^* obtained with Annealed Importance Sampling ($N = 200$) (mean value over 25 realizations of $\mathbf{v}^1$ and $\mathbf{v}^2$ with overlap equal to x). Shaded areas correspond to the empirical error bars. (a) Behaviour of y^* vs. w at fixed $x = 0.5$ (b) Behaviour of y^* vs. w at fixed $\alpha = 0.5$. (c) Behaviour of y vs. x at fixed $w = 2$. The vertical bars locate x_c.

D. Mixing and swap time for a stack of two RBMs learning hierarchical data

We will illustrate the speed up offered by Deep Tempering with respect to conventional Gibbs sampling in a simple setting.

Consider a dataset of size K, made of K' orthogonal clusters $\mathcal{C}_k$ centered in $\mathbf{v}^k$, while each cluster includes

K/K' configurations, see Fig. 9(a). Cluster centers are orthogonal (and have thus zero overlap).

These data are learned by a RBM, with $M_1 = \alpha^{(1)} N$ hidden units (see Fig. 9(b)) and weight matrix

$$W_{i\mu}^{(1)} = \frac{w^{(1)}}{N} \sum_{k=1}^{K} v_i^k \, h_\mu^k \ . \tag{22}$$

This expression implicitly assumes that all the configurations in a cluster are mapped onto the same representation, see Section IV C and Fig. 9(a). In other words, the normalized overlap between the center of a cluster and any attached configuration, x, is larger than the clustering overlap x_c.

For this first RBM, we introduce the characteristic time scale $\tau_{\rm cross}^{(1)}$ to go from one cluster to another with Alternating Gibbs Sampling. This time scale is determined by the barrier height, in the effective free-energy landscape E^v, between the the different clusters. It can be determined with standard statistical mechanics calculations, with the result

$$\tau_{\rm cross}^{(1)} \propto \exp\left(N \, \mathcal{B}\left(\alpha^{(1)}, w^{(1)}, K \right) \right) \tag{23}$$

where the function $\mathcal{B}$ is defined in Appendix B, see Eq. B9. Hence, the time $\tau_{\rm cross}^{(1)}$ is exponential in N, showing that mixing is extremely slow, as expected for a multimodal data distribution.

A second RBM with M_2 hidden units ($M_2 \ll M_1$, $M_2 \gg K$) is trained on the K representations $\{\mathbf{h}_1^k\}_{k=1\ldots K}$ on the cluster centers produced by the first RBM, see Eq. 22. Its weight matrix can be written:

$$W_{i\mu}^{(2)} = \frac{w^{(2)}}{M_1} \sum_{k=1}^{K} h_i^k \, \tilde{h}_\mu^k \ , \tag{24}$$

where $\{\tilde{\mathbf{h}}^k\}_{k=1\ldots K}$ is a set of mutually orthogonal vectors, depending mainly on the (random) value of $W^{(2)}$ at the beginning of the training phase. $w^{(2)}$ is a free-parameter that can be tuned by adding a regularisation during the training.

For this second RBM, the characteristic time scale $\tau_{\rm cross}^{(2)}$ to sample, with alternating Gibbs sampling, multiple clusters can be computed again as in Appendix B. We thus find

$$\tau_{\rm cross}^{(2)} \propto \exp\left(M_1 \, \mathcal{B}\left(\alpha^{(2)}, w^{(2)}, K \right) \right) \ , \tag{25}$$

where $\alpha^{(2)} = M_2/M_1$. We stress that $\tau_{\rm cross}^{(2)}$ is exponential in M_1, and is therefore much smaller than the mixing time $\tau_{\rm cross}^{(1)}$ of the first RBM.

We now study how the two RBMs, which generate, respectively, configurations $\{\mathbf{v}_1^t, \mathbf{h}_1^t\}$ and $\{\mathbf{v}_2^t, \mathbf{h}_2^t\}$, can occasionally exchange their configurations:

$$\mathbf{v}_2^{t+1} = \mathbf{h}_1^t \ , \quad \mathbf{h}_1^{t+1} = \mathbf{v}_2^t \ , \tag{26}$$

see Figs. 3 and 9. The swap between $\mathbf{h}_1^t$ and $\mathbf{v}_2^t$ is accepted with probability $A_1(\mathbf{h}_1^t, \mathbf{v}_2^t)$, see Eq. 8. The mean value of this acceptance ratio can be computed, and its inverse is an estimate of the characteristic time $\tau_{\rm swap}$ between two replica exchanges, see Appendix C for a detailed calculation,

$$\tau_{\rm swap} \propto \exp\left(M_1 \, \mathcal{C}\left(\alpha^{(2)}, w^{(2)}, K \right) \right) \ , \tag{27}$$

which is exponentially large in M_1.

From the discussion, it appears that the time necessary for Deep Tempering to mix between the modes of the data distribution,

$$\tau_{\rm DT} = \max\left(\tau_{\rm cross}^{(2)}, \tau_{\rm swap} \right) \ , \tag{28}$$

scales exponentially with M_1 and is therefore much smaller than $\tau_{\rm cross}^{(1)}$.

The choice of the aspect ratio and of the amplitude of the weight can also help decrease $\tau_{\rm DT}$. As seen above the characteristic time $\tau_{\rm DT}$ is therefore the result of a trade-off implemented by the second RBM. On the one hand, it is better for this RBM to have low barriers and fast mixing, i.e. low $\tau_{\rm cross}^{(2)}$. On the other hand, the visible configurations of the second RBM must be similar to the representations produced by the first RBM, otherwise exchanges will be excessively rare and $\tau_{\rm swap}$ extremely long. In general, we find that $\tau_{\rm cross}^{(2)}$ is an increasing function of $w^{(2)}\alpha^{(2)}$, while the swap time $\tau_{\rm swap}$ decreases with this product, see Fig. 10.

There exists an optimal choice for $w^{(2)}$, see Fig. 10(c,f). For this optimal value, $\tau_{DT} \ll \tau_{\rm cross}^{(1)}$. Choosing the optimal w_2 is somewhat similar to choosing the optimal temperature in conventional parallel tempering.

TABLE III. Values of the products $w\alpha$ for the RBMs trained on MNIST0/1

	$w^{(1)}\alpha^{(1)}$	$w^{(2)}\alpha^{(2)}$	$w^{(3)}\alpha^{(3)}$	$w^{(4)}\alpha^{(4)}$
# Value	2.01	2.96	2.31	1.69

TABLE IV. Values of the products wM for the RBMs trained on Lattice Proteins S_A/S_B

	$w^{(1)}\alpha^{(1)}$	$w^{(2)}\alpha^{(2)}$	$w^{(3)}\alpha^{(3)}$
# Value	38.80	0.88	2.37

V. CONCLUSIONS

The idea of deep tempering was first proposed in the context of deep belief networks (DBN) [5, 22]. The DBN considered in [22] showed increasing numbers of hidden

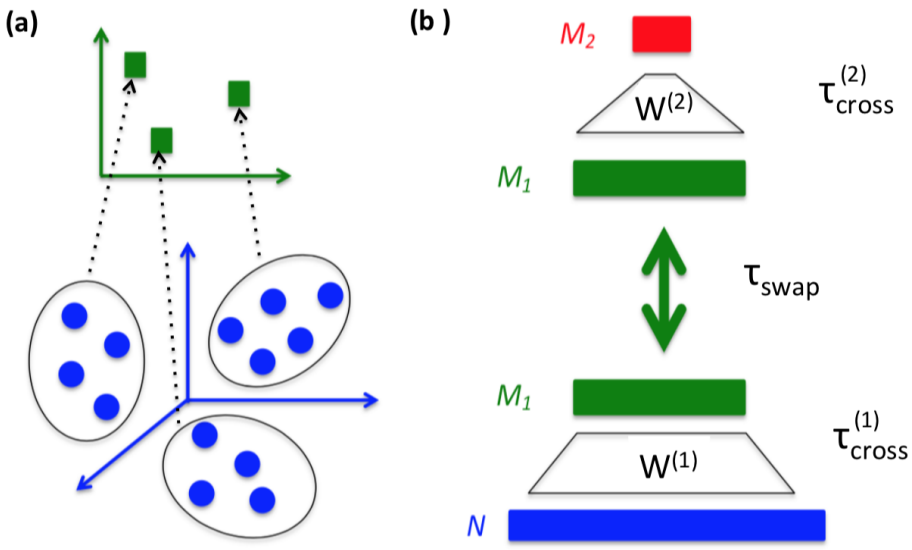

FIG. 9. Theoretical setting for deep tempering performance assessment. Data distribution and representations (a) and the stack of two RBMs (b). The times $\tau^{(1)}_{\text{cross}}$ and $\tau^{(2)}_{\text{cross}}$ refer to the mixing times of the two machines, and τ_{swap} to the average time between two exchanges of $\mathbf{h}^{(1)}$ and $\mathbf{v}^{(2)}$.

units with the depth, and deeper representations could better disentangle the underlying factors of variation of the data. DBN as a whole were therefore conceived to be a better model than a single RBM [15]. In [5], a new training algorithm for DBN, called Deep Tempering, was introduced. RBMs were trained jointly with replica exchanges between neighboring RBMs to exploit the progressive disentanglement along the stack. In these numerical experiments, the number of hidden units of the different RBMs were kept constant, while the regularization of their RBMs increased with the depth. DBNs obtained with this algorithm showed better performance than DBNs trained greedily.

In our work, the underlying mechanism for deep tempering is different: deeper restricted Boltzmann machines and representations are not meant to better disentangle the underlying factors of variation of the data, but rather to compress the bottom RBM's hidden representations. Each mode of the data has few distinct representations in the hidden space of the top RBM. By reducing the number of hidden units with the depth, the hidden representations are progressively simplified. Due to the compression, our DBN as a whole is a poorer model than the bottom RBM. However, we do not aim at sampling the visible landscape of the DBN. The Deep Tempering procedure is here used to sample the landscape of the bottom RBM only. In the present implementation, deep tempering is not a training algorithm for RBMs or DBNs: its main goal is to improve the sampling of the bottom RBM after its training. The other $(N-1)$ RBMs are meant to enhance the mixing between the modes of the bottom RBM through replica exchanges between them.

RBM can encode meaningful hidden representations of the data. Using these representations can be helpful to detect relevant collective modes of units. Adding Metropolis-Hastings steps in the hidden space can help the sampling [9]. We also introduce a stack of RBMs to detect and cluster the hidden representations of the data. In a simple example of a data distribution, where the data of interest are grouped into unrelated (orthogonal) clusters, we show that a dynamical scheme in which different RBMs are coupled together through configuration exchanges can decrease the characteristic time scale to go from one cluster to another: Deep Tempering is thus more efficient than Gibbs sampling. On real data, this algorithm with our deep architecture also enhances the mixing between different modes. In this context, beyond the gain in sampling time offered by deep tempering, it would be appealing to further study the properties of our architecture for real data, and better understand the ways a stack of RBMs can coarse-grain complex, multimodal distributions.

ACKNOWLEDGMENTS

C. Roussel acknowledges funding from DGA.

Appendix A: Expression of x_c

We first compute the derivative of the log-likelihood with respect to y

$$\frac{\partial\left(\frac{LL}{N}\right)}{\partial y} = \frac{1}{2}\log\left(\frac{\cosh\left(w(1+x)\right)\cosh\left(w\Delta_-\right)}{\cosh\left(w(1-x)\right)\cosh\left(w\Delta_+\right)}\right) \quad \text{(A1)}$$

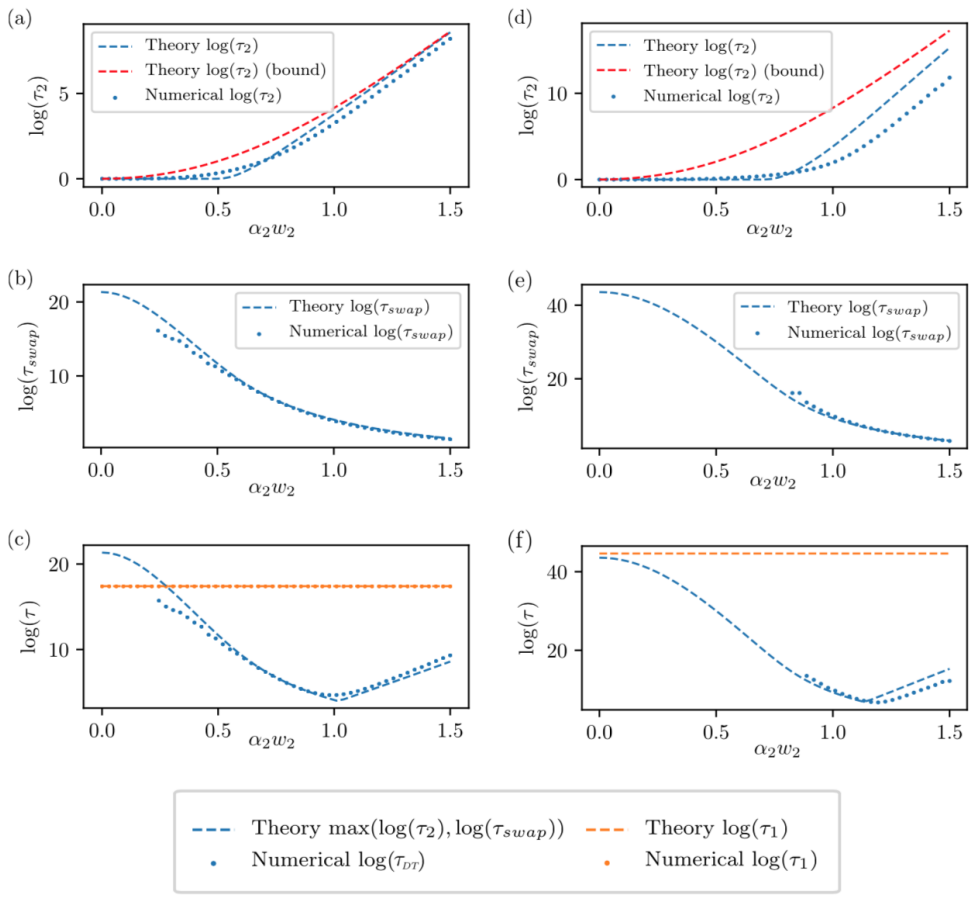

FIG. 10. Mixing and swap times for the hierarchical distribution of Fig. 9(a), with $K = 3, N = 128, M_1 = 32, M_2 = 8$ (left) and $K = 5, N = 256, M_1 = 64, M_2 = 32$ (right). Dashed lines: theoretical results. Dots: numerical estimates. (a) and (d) Characteristic time $\tau^{(2)}_{\rm cross}$ vs. $\alpha^{(2)}\, w^{(2)}$. The offset between the theory and the numerical results for large values $\alpha^{(2)}\, w^{(2)}$ is equal to $-(K-2)\log 2$. This term corresponds to the logarithm of the number of optimal distinct paths joining two global minima of the free energy. (b) and (e) Characteristic time $\tau_{\rm swap}$ vs $\alpha^{(2)}\, w^{(2)}$. (c) and (f) Characteristic time $\tau_{\rm DT}$ defined in Eq. 28 vs. $\alpha^{(2)}\, w^{(2)}$. For comparison we show $\tau^{(1)}_{\rm cross}$, the mixing time of the first RBM with Alternating Gibbs Sampling alone.

The clustering crossover point x_c is defined by the condition $\frac{\partial\left(\frac{LL}{N}\right)}{\partial y}\Big|_{y=1} = 0$. Then,

$$\frac{\cosh\left(w(1+x_c)\right)}{\cosh\left(w(1-x_c)\right)\cosh\left(w\Delta_+\right)} = 1 \tag{A2}$$

as $\Delta_- = 0$ in the pattern completion regime.

We now need to derive an expression for Δ_+. To do so we extremize Eq. 21 over $\hat{\Delta}_+, \Delta_+$. We obtain $\hat{\Delta}_+ = \alpha w(1+y)\tanh(w\,\Delta_+)$ and $\Delta_+ = (1+x)\ \tanh(\hat{\Delta}_+)$. Combining these equations and the one above we find that x_c is the root of the following implicit equation:

$$\frac{1}{w(1+x_c)}\cosh^{-1}\left(\frac{\cosh(w(1+x_c)}{\cosh(w(1-x_c)}\right) \tag{A3}$$
$$= \tanh\left(2\alpha w\tanh\left[\cosh^{-1}\left(\frac{\cosh(w(1+x_c)}{\cosh(w(1-x_c)}\right)\right]\right)$$

Appendix B: Computation of $\tau_{\rm cross}$

Let $f(\mathbf{m}^v, \mathbf{m}^h)$ be the free–energy density associated to states with magnetizations $\mathbf{m}^v = (m^v_1, ..., m^v_K)$ and $\mathbf{m}^h = (m^h_1, ..., m^h_K)$, where

$$m^v_k = \frac{1}{N}\sum_{i=1}^{N}\langle v_i\rangle v^k_i\ ,\ m^h_k = \frac{1}{M}\sum_{\mu=1}^{M}\langle h_\mu\rangle h^k_\mu \tag{B1}$$

and $\langle\cdot\rangle$ denotes the average over $P(\mathbf{v},\mathbf{h})$ in Eq. 1 and coupling matrix in Eq. 22. Following standard calculations [23] we obtain

$$F(\mathbf{m}^v,\mathbf{m}^h) \;=\; \alpha^{(1)} w^{(1)} \sum_{k=1}^{K} m_k^v\, m_k^h - \alpha^{(1)} \sum_{\boldsymbol{\sigma}} \varphi^v(\boldsymbol{\sigma}) \log\left(2\cosh\left(w^{(1)} \sum_{k=1}^{K} m_k^v\, \sigma_k\right)\right)$$
$$-\; \sum_{\boldsymbol{\tau}} \varphi^h(\boldsymbol{\tau}) \log\left(2\cosh\left(\alpha^{(1)} w^{(1)} \sum_{k=1}^{K} m_k^h\, \tau_k\right)\right)\;, \tag{B2}$$

where $\sum_{\boldsymbol{\sigma}}$ runs over the 2^K vectors $\boldsymbol{\sigma}$ of length K with binary components, $\sigma_k \pm 1$. The K data configurations $\mathbf{v}^k$ can be written in a matrix of size $N \times K$, and $\varphi^v(\boldsymbol{\sigma})$ is the frequency of $\boldsymbol{\sigma}$ among the N lines of this matrix. In the same way, we define $\varphi^h(\boldsymbol{\tau})$ based on the statistics the K vectors $\mathbf{h}^k$.

For statistically orthogonal and uniform vectors, $\varphi^v(\boldsymbol{\sigma}) = \varphi^h(\boldsymbol{\tau}) = \frac{1}{2^K}$, and we may look for saddle-points (barrier state) of the free energy f of the form

$$\mathbf{m}^v \;=\; m^v\, (\underbrace{1,1,\ldots 1}_{r},\underbrace{0,0,\ldots,0}_{K-r})\;,$$
$$\mathbf{m}^h \;=\; m^h\, (\overbrace{1,1,\ldots 1}^{r},\overbrace{0,0,\ldots,0}^{K-r})\;. \tag{B3}$$

Differentiating the free energy in Eq. (B2), we find two coupled equations for ξ^v, ξ^h:

$$m^h = \frac{1}{2^{r-1}} \sum_{\boldsymbol{\sigma}} \tanh\left(\alpha^{(1)} w^{(1)} m^v \left(1 + \sum_{\ell} \sigma_\ell\right)\right) \tag{B4}$$

where the sum runs over the 2^r vectors $\boldsymbol{\sigma}$ of length r with binary coefficients (± 1), and similarly

$$m^v = \frac{1}{2^{r-1}} \sum_{\boldsymbol{\sigma}} \tanh\left(w^{(1)} m^h \left(1 + \sum_{\ell} \sigma_\ell\right)\right) \tag{B5}$$

These equations admit non-zero solutions as soon as $\alpha^{(1)}\big(w^{(1)}\big)^2 > 1$, see Fig. 11(a,c). For $\alpha^{(1)} w^{(1)} m^h \gg 1$ and $w^{(1)}\xi^v \gg 1$, we obtain

$$m^v = m^h = \frac{1}{2^{2c}} \binom{2c}{c} \equiv m_r\;, \tag{B6}$$

where $r = 2c$ for even r and $r = 2c+1$ for odd r. This result is similar to the symmetric spurious memories of the Hopfield model at zero temperature [23].

The free-energies F_r of this symmetric saddle-point

$$F_r = \begin{cases} -\alpha^{(1)} w^{(1)} r\, m_r^2 & \text{if } r \text{ is odd} \\ -\alpha^{(1)} w^{(1)} r\, m_r^2 - (1+\alpha^{(1)}) \log 2 m_r & \text{if } r \text{ is even} \end{cases} \tag{B7}$$

can be ordered as follows

$$F_1 < F_3 < F_5 < \ldots < F_4 < F_2 \tag{B8}$$

For even r, the free energy F_r is a decreasing function of r, while, for odd r, it is increasing with r, see Fig. 11(b,d). Notice that saddle-point that are not symmetric under the permutation of the k indices exist, but their free energies are higher than the one of the symmetric saddle-point with $r = 3$.

We may now conclude:

- the lowest free energy solution corresponds to $r = 1$;
- the lowest excited state, on the transition path to state, say, $k = 1$ to $k = 2$, corresponds to $r = 3$ (and has therefore non-zero projection onto another state, here, $K = 3$);
- the lowest barrier to cross along the transition path is thus

$$\mathcal{B}(\alpha^{(1)}, w^{(1)}, K) = F_1 - F_3\;. \tag{B9}$$

Notice that the expression above is intensive, and must be multiplied by N to obtain the full barrier height.

Appendix C: Computation of τ_{swap}

We hereafter compute the characteristic time τ_{swap} between two replica exchanges between configurations $\mathbf{v}_2^t$ sampled by the top RBM and $\mathbf{h}_1^t$ sampled by the botton RBM in Fig. 9. The acceptance probability for the exchange is defined in Eq. 8, and equal to

$$A_1(\mathbf{h}_1^t, \mathbf{v}_2^t) = \min\left(1, \frac{P_2^v(\mathbf{h}_1^t) P_1^h(\mathbf{v}_2^t)}{P_2^v(\mathbf{v}_2^t) P_1^h(\mathbf{h}_1^t)}\right)\;. \tag{C1}$$

We want to estimate the mean value of this acceptance probability,

$$\langle A_1 \rangle = \sum_{\mathbf{h}_1, \mathbf{v}_2} P_1^h(\mathbf{h}_1)\, P_2^v(\mathbf{v}_2)\, A_1(\mathbf{h}_1, \mathbf{v}_2)\;. \tag{C2}$$

Notice that the expression above is exact if the two Monte Carlo chains sampling the two RBMs are independent from each other, otherwise the configurations $\mathbf{h}_1, \mathbf{v}_2$ are not independent. This approximation should be harmless if the exchanges are rare, *i.e.* when $\tau_{\text{swap}} \gg 1$.

We expect $P_1^h(\mathbf{h}_1)$ to be very peaked around the representations $\mathbf{h}_1^k$ of the data cluster centers. This statement implies that, in order to have a reasonable probability of an exchange, we should have both $\mathbf{h}_1$ and $\mathbf{v}_2$ close to one of the $\mathbf{h}_1^k$'s; if this condition is realized, we have $A_1 \simeq 1$.

Therefore, the mean value of the acceptance ratio can be bounded by

$$\langle A_1 \rangle \leq \sum_{k=1}^{K} P_1^h\big(\mathbf{h}_1^k\big) P_2^v\big(\mathbf{h}_1^k\big) \leq \frac{1}{K} \sum_{k=1}^{K} P_2^v\big(\mathbf{h}_1^k\big)\;. \tag{C3}$$

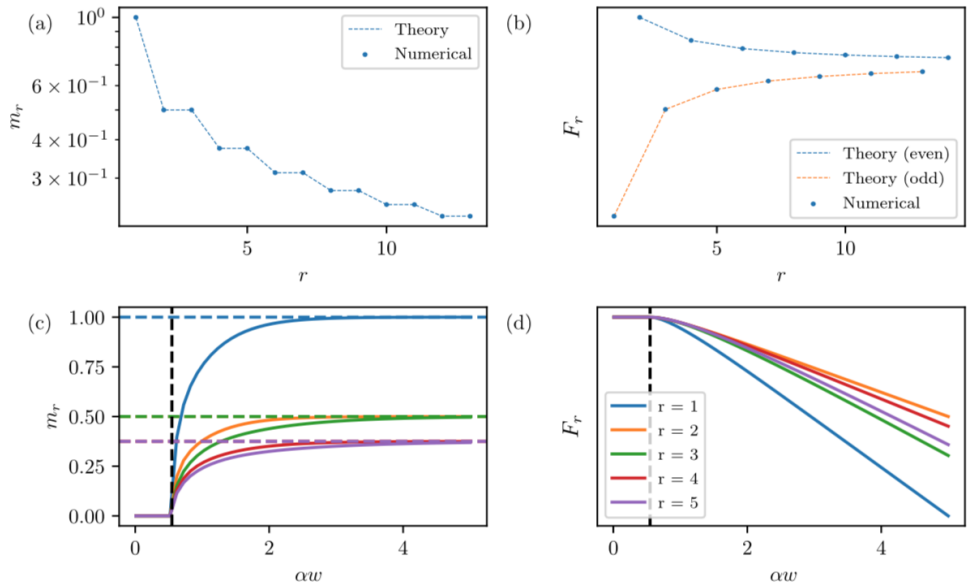

FIG. 11. (a-b) m_r and F_r under the assumptions $\alpha^{(1)} w^{(1)} m_r \gg 1$ and $w_r^{(1)} \gg 1$. Dashed lines: theoretical result in Eqs. B6 and B7. Dots: numerical results. (c-d) m_r and F_r against αw. Black dashed line: threshold where $\alpha w^2 = 1$.

We now need to estimate $P_2^v(\mathbf{v})$ for $\mathbf{v} \simeq \mathbf{h}_1^k$ for, say, $k = 1$. Suppose $\mathbf{v}$ differs from $\mathbf{h}_1^1$ on d sites (with $d \ll M_1/2$). Then, using Eq. 24 for the coupling matrix of the second RBM,

$$\begin{aligned} P_2^v(\mathbf{v}) &= \frac{1}{Z} \sum_{\{\tilde{h}_\mu = \pm 1\}} \exp\left(\sum_{i=1}^{M_1} \sum_{\mu=1}^{M_2} W_{i\mu}^{(2)} v_i \tilde{h}_\mu \right) \\ &= \frac{1}{Z} \prod_{\mu=1}^{M_2} 2\cosh\left(\frac{w^{(2)}}{M_1} (M_1 - 2d)\, \tilde{h}_\mu^1 \right) . \end{aligned} \quad \text{(C4)}$$

Therefore, the probability P_2^v decreases exponentially with d and the partition function Z can be exactly computed. We obtain

$$P_2^v(\mathbf{v}) = \frac{e^{-\eta\, d}}{K\left(1 + e^{-\eta}\right)^{M_1}} \, , \eta \equiv 2\alpha^{(2)} w^{(2)} \tanh w^{(2)} \quad \text{(C5)}$$

Keeping only the exponential in M_1 terms in the upper bound for $\langle A_1 \rangle$, we obtain

$$\tau_{\text{swap}} \simeq \frac{1}{\langle A_1 \rangle} \geq \exp\left(M_1\, \mathcal{C}\left(\alpha^{(2)}, w^{(2)}\right) \right) \, , \quad \text{(C6)}$$

where

$$\mathcal{C}\left(\alpha^{(2)}, w^{(2)}\right) = \log\left(1 + e^{-2\alpha^{(2)} w^{(2)} \tanh w^{(2)}} \right) . \quad \text{(C7)}$$

Notice that, to a good level of approximation, we have $\mathcal{C}\left(\alpha^{(2)}, w^{(2)}\right) \simeq \log\left(1 + e^{-2\alpha^{(2)} w^{(2)}}\right)$, which is a function of the product $\alpha^{(2)} w^{(2)}$ only.

[1] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987), URL https://link.aps.org/doi/10.1103/PhysRevLett.58.86.

[2] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989), URL https://link.aps.org/doi/10.1103/PhysRevLett.62.361.

[3] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **57**, 2607 (1986), URL https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

[4] N. L. Roux and Y. Bengio, Neural Computation **20**, 1631 (2008), ISSN 0899-7667, conference Name: Neural Computation.

[5] G. Desjardins, H. Luo, A. Courville, and Y. Bengio, arXiv:1410.0123 [cs, stat] (2014), arXiv: 1410.0123, URL http://arxiv.org/abs/1410.0123.

[6] Y. LeCun, http://yann. lecun. com/exdb/mnist/ (1998).

[7] E. Shakhnovich and A. Gutin, J. Chem. Phys. **93**, 5967 (1990), ISSN 0021-9606, publisher: American Institute of Physics, URL https://aip.scitation.org/doi/10.1063/1.459480.

[8] L. Mirny and E. Shakhnovich, Annu. Rev. Biophys. Biomol. Struct. **30**, 361 (2001), ISSN 1056-8700, publisher: Annual Reviews, URL https://www.annualreviews.org/doi/10.1146/annurev.biophys.30.1.361.

[9] C. Roussel, S. Cocco, and R. Monasson, Phys. Rev. E **104**, 034109 (2021), URL https://link.aps.org/doi/10.1103/PhysRevE.104.034109.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, Neural Comput **18**, 1527 (2006), ISSN 0899-7667.

[11] T. Tieleman, in *Proceedings of the 25th international conference on Machine learning* (Association for Computing Machinery, New York, NY, USA, 2008), ICML '08, pp. 1064–1071, ISBN 978-1-60558-205-4, URL https://doi.org/10.1145/1390156.1390290.

[12] S. Miyazawa and R. L. Jernigain, J Mol Biol **256**, 623 (1996), ISSN 0022-2836, 1089-8638, URL https://europepmc.org/article/med/8604144.

[13] H. Jacquin, A. Gilson, E. Shakhnovich, S. Cocco, and R. Monasson, PLOS Computational Biology **12**, 1 (2016), URL https://doi.org/10.1371/journal.pcbi.1004889.

[14] G. E. Hinton and R. R. Salakhutdinov, Science **313**, 504 (2006), ISSN 0036-8075, 1095-9203, publisher: American Association for the Advancement of Science Section: Report, URL https://science.sciencemag.org/content/313/5786/504.

[15] R. Salakhutdinov and I. Murray, in *Proceedings of the 25th international conference on Machine learning* (Association for Computing Machinery, New York, NY, USA, 2008), ICML '08, pp. 872–879, ISBN 978-1-60558-205-4, URL https://doi.org/10.1145/1390156.1390266.

[16] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **57**, 2607 (1986), publisher: American Physical Society, URL https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

[17] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (MIT Press Cambridge, MA, 2010), pp. 145–152.

[18] M. Mézard, G. Parisi, N. Sourlas, G. Toulouse, and M. Virasoro, Phys. Rev. Lett. **52**, 1156 (1984), publisher: American Physical Society, URL https://link.aps.org/doi/10.1103/PhysRevLett.52.1156.

[19] R. Rammal, G. Toulouse, and M. A. Virasoro, Rev. Mod. Phys. **58**, 765 (1986), publisher: American Physical Society, URL https://link.aps.org/doi/10.1103/RevModPhys.58.765.

[20] A. Decelle, G. Fissore, and C. Furtlehner, Europhysics Letters **119**, 60001 (2017), URL https://dx.doi.org/10.1209/0295-5075/119/60001.

[21] R. M. Neal, Statistics and Computing **11**, 125 (2001), ISSN 1573-1375, URL https://doi.org/10.1023/A:1008923215028.

[22] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, in *International Conference on Machine Learning* (PMLR, 2013), pp. 552–560, iSSN: 1938-7228, URL http://proceedings.mlr.press/v28/bengio13.html.

[23] D. J. Amit, H. Gutfreund, and H. Sompolinsky, Phys. Rev. A **32**, 1007 (1985), publisher: American Physical Society, URL https://link.aps.org/doi/10.1103/PhysRevA.32.1007.