Equilibrium and non-Equilibrium regimes in the learning of Restricted Boltzmann Machines

Anonymous Author(s)
Affiliation
Address
email

Abstract

Training Restricted Boltzmann Machines (RBMs) has been challenging for a long time due to the difficulty of computing precisely the log-likelihood gradient. Over the past decades, many works have proposed more or less successful training recipes but without studying the crucial quantity of the problem: the mixing time, i.e. the number of Monte Carlo iterations needed to sample new configurations from a model. In this work, we show that this mixing time plays a crucial role in the dynamics and stability of the trained model, and that RBMs operate in two well-defined regimes, namely equilibrium and out-of-equilibrium, depending on the interplay between this mixing time of the model and the number of steps, k, used to approximate the gradient. We further show empirically that this mixing time increases with the learning, which often implies a transition from one regime to another as soon as k becomes smaller than this time. In particular, we show that using the popular k (persistent) contrastive divergence approaches, with k small, the dynamics of the learned model are extremely slow and often dominated by strong out-of-equilibrium effects. On the contrary, RBMs trained in equilibrium display faster dynamics, and a smooth convergence to dataset-like configurations during the sampling. Finally we discuss how to exploit in practice both regimes depending on the task one aims to fulfill: (i) short k can be used to generate convincing samples in short learning times, (ii) large k (or increasingly large) is needed to learn the correct equilibrium distribution of the RBM.

1 Introduction

Restricted Boltzmann Machines (RBM) are one of the generative stochastic neural networks that have been available for decades [1] and are renowned for their capacity to learn complex datasets [2] and draw similar new data. Furthermore, the hidden correlations found in the data are easily accessible [3, 4, 5], which is particularly interesting for a scientific use of Big Data. Despite these positive features, RBMs still remain hard to train and evaluate, and hence to use in practical problems. In comparison, Generative Adversarial Network (GAN) [6] (for which it is much harder to find a working architecture) are more commonly used, not only because of the advantages given by convolutional layers, but also because the generative process does not depend on a costly sampling procedure with an uncontrolled convergence time.

Both training and data generation of RBMs are tricky and unpredictable in inexperienced hands. One of the main difficulties is that it is hard to evaluate if the learning is progressing or not. Yet, many works in the past twenty years have proposed recipes for good practices in RBM training [7]. Unfortunately, the evaluation of these recipes relies only on the comparison of the values reached by the log-likelihood (LL) [8, 7, 9], a quantity that cannot be monitored in tractable times during the learning process. Studies properly characterizing the quality, independence, or stability of the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

generated samples using the different recipes are nearly absent in the Literature. In fact, many works just show a set of new samples (for eye evaluation) that were either obtained after a short Markov Chain Monte Carlo (MCMC) sampling initialized at the dataset, or borrowed from the permanent chain (negative particle) at the end of the training [10]. In other studies, a reconstruction error after one or several MCMC steps is used to determined if the machine was correctly trained [11]. However, none of these measures guarantees that the trained model can generate from scratch new samples nor that the dataset are typical samples of that model. At most these tests assure you that samples of the dataset are locally stable w.r.t few MCMC steps, unless a proper decorrelation from the initial conditions or a link to equilibrium properties are investigated. Finally, more recent works use a classification score on the activation of the latent variables [12, 13], which reflects that the relevant features were learned, but is silent about the generative power of the machine.

RBMs are receiving an increasing attention in life and pure sciences during the recent years [14, 15, 16, 17, 18, 5, 19, 20, 21] given their potential for interpretability of the patterns learned by the machine (a rare feature in machine learning). Indeed, the RBM formalism enables the extraction of the (unknown) probability distribution function of the dataset. Yet, such an approach is meaningful if the dataset is related to the equilibrium properties of the trained model. Given these new and exciting prospectives for the RBMs, it is more than ever important to establish reproducible protocols and evaluation tools to guarantee not only that the model generates good enough data, but also that the dataset are typical equilibrium samples of the model.

In this work we demonstrate that the classical training procedures can lead to two distinct regimes of the RBM: an equilibrium and an out-of-equilibrium (OOE) one. The former is observed if the number of MCMC steps, k, used to estimate the LL gradient is larger than the algorithm mixing time (MT). The latter emerges when k falls below this MT. In particular, we show that for a classical dataset such as MNIST, the MT is always rather large and grows with the learning time. This implies that learning an "equilibrium" model (i.e. a RBM that operates in the equilibrium regime) is costly, and that most of the works in the Literature operate without any doubt in the OOE regime. In this regime, the equilibrium distribution of the machine at the end of the learning is significantly different from that of the dataset. Yet, despite this handicap, we show the OOE regime can be exploited to to generate good samples at short training times. All our conclusions rely on the study of 3 datasets: MNIST [22], whose results are discussed in the main-text, a human genome dataset [23] (with similar dimensions as MNIST but more structured) and the high-quality CelebA [24] projected in black and white. The analysis of the last 2 datasets are discussed only in the SM and show a similar behavior to the one observed on MNIST.

Our paper is organized as follows. RBMs, their general principles and learning equations are defined in sec. 2. We review previous related works in sec. 3. In sec. 4 we define the different observables used to monitor the training and sampling. Finally, we discuss experimental results in sec. 5.

2 Definition of the model

An RBM is a Markov random field with pairwise interactions defined on a bipartite graph of two non-interacting layers of variables: the visible nodes, $\boldsymbol{v} = v_i, i = 1, ..., N_v$, represent the data, while the hidden ones, $\boldsymbol{h} = h_j, j = 1, ..., N_h$, are the latent representations of this data and are there to build arbitrary dependencies between the visible units (see sketch in fig. 1–A). Usually, the nodes are binary-valued in $\{0, 1\}$, yet Gaussian or more arbitrary distributions on real-valued bounded support are also used, ultimately making RBMs adaptable for more heterogeneous datasets. Here, we deal only with binary $\{0, 1\}$ variables for both the visible and hidden nodes. Other approaches using truncated-Gaussian hidden units [25], giving a ReLu type of activation functions for the hidden layer do work well, but in our experiments we observed qualitatively the similar dynamical behavior and therefore we will stick to binary hidden units for the rest of the article. The energy function of an RBM is taken as

$$E[\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}] = -\sum_{ia} v_i w_{ia} h_a - \sum_i b_i v_i - \sum_a c_a h_a, \tag{1}$$

with $\boldsymbol{w}$ the weight matrix and $\boldsymbol{b}$, $\boldsymbol{c}$ the visible and hidden biases, respectively. The Boltzmann distribution is then given by

$$p[\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}] = \frac{\exp(-E[\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}])}{Z} \quad \text{with } Z = \sum_{\{\boldsymbol{v}, \boldsymbol{h}\}} e^{-E[\boldsymbol{v}, \boldsymbol{h}]} \tag{2}$$

being the partition function of the system. RBMs are usually trained using gradient ascent of the LL function of the training dataset $\mathcal{D} = \{\boldsymbol{v}^{(1)}, \cdots, \boldsymbol{v}^{(M)}\}$, being the LL defined as

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}|\mathcal{D}) = M^{-1} \sum_{m=1}^{M} \ln p(\boldsymbol{v} = \boldsymbol{v}^{(m)}|\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}) = M^{-1} \sum_{m=1}^{M} \ln \sum_{\{\boldsymbol{h}\}} e^{-E[\boldsymbol{v}^{(m)}, \boldsymbol{h}; \boldsymbol{w}, \boldsymbol{b}, \boldsymbol{c}]} - \ln Z \quad (3)$$

The gradient is then composed of two terms: the first accounting for the interaction between the RBM's response and the training set, and same for the second, but using the samples drawn by the machine itself. The expression of the LL gradient w.r.t. all the parameters is given by

$$\frac{\partial \mathcal{L}}{\partial w_{ia}} = \langle v_i h_a \rangle_{\mathcal{D}} - \langle v_i h_a \rangle_{\mathcal{H}}, \quad \frac{\partial \mathcal{L}}{\partial b_i} = \langle v_i \rangle_{\mathcal{D}} - \langle v_i \rangle_{\mathcal{H}} \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial c_a} = \langle h_a \rangle_{\mathcal{D}} - \langle h_a \rangle_{\mathcal{H}}, \quad (4)$$

where $\langle f(\boldsymbol{v}, \boldsymbol{h}) \rangle_{\mathcal{D}} = M^{-1} \sum_m \sum_{\{\boldsymbol{h}\}} f(\boldsymbol{v}^{(m)}, \boldsymbol{h}) p(\boldsymbol{h}|\boldsymbol{v}^{(m)})$ denotes an average over the dataset, and $\langle f(\boldsymbol{v}, \boldsymbol{h}) \rangle_{\mathcal{H}}$, the average over the Boltzmann measure in Eq. 2.

A large part of the literature on RBMs focuses on schemes to approximate the r.h.s. of Eqs. 4. In theory, these terms can be computed up to an arbitrary precision using parallel MCMC simulations as long as the number of MC steps are large enough to ensure a proper sampling of the equilibrium configuration space. In other words, sampling times should be larger than the MCMC MT. A naive way to implement this idea is to initialize each of these parallel Markov chains on random conditions, perform k MCMC steps, and use the final configurations to estimate the gradient. We call this scheme **Rdm-k**. This scheme is not used in practice because it would require too many k steps if the MT is too large. Many recipes have been proposed to shorten this sampling and approximate the negative term of the gradient, but since the introduction of the so-called contrastive divergence (CD) by Hinton [26], only a limited number of schemes are used in practice. The first one, **CD-k**, initializes the MCMC simulation from the same data of the mini-batch used to compute the gradient, and performs $k \sim \mathcal{O}(1)$ sampling steps. This approximation relies on the idea that the dataset must be a good approximation of the equilibrium samples of a well trained RBM, something that is not true for a poorly trained machine, whose typical samples are quite distant from the dataset. In the second commonly used method, the last configurations of the Markov chains are saved from one parameter update to the other, and then used to initialize the subsequent chain. As in the other methods, k MCMC steps are used to estimate the gradient for each update. This method is known as persistent-CD: **PCD-k** [27], and takes advantage of the fact that the RBM's parameters change smoothly during the learning and the same is expected for the models' equilibrium distribution[1]. Finally, other approximations, such as mean-field or TAP equations [10], or more elaborate approaches, such as the parallel tempering technique [28, 29], can be used. A common feature of all these recipes is that they rely on very few simulation steps to sample the Boltzmann distribution, and the trained model often generates rather bad configurations.

To illustrate this last comment, let us discuss a provocative experiment shown on fig. 1–E where we try to sample new digits just by sampling the equilibrium measure of our trained RBMs using a MCMC initialized uniformly at random. Let us stress that this is not the standard set-up used in the field to generate new samples, but it is a strong test about the generation power of the trained model. We compare the outcome of the generated samples (as function of the sampling time) for three RBMs trained each following different recipes to compute the negative term of the gradient: CD-10, PCD-10 and Rdm-10. At this point, $k = 10$ might seem a very small number. It is however the typical order of magnitude used in the Literature. The details of the learning protocol are discussed later in the text and summarized in the SM. The first striking point on fig. 1–E, is the inability of the CD-10 trained RBM to properly sample in this set-up. This problem has been reported before [30, 8], but since CD is part of the standard methods we want to insist on this point. In addition, PCD-10 trained RBM generates proper digits but only after a very long sampling time, while with Rdm-10, quite surprisingly, digits are properly generated after only 10 MCMC steps. Yet, equilibrium samples obtained after numerous MCMC steps have unbalanced digits representations, like for instance the 1 being over-represented. In fact, we observe that poorly trained models generate samples that do not respect the correct balance between the 10 digits present in the original database (see fig. 1–D). We already mentioned the 1s in the Rdm-10 run, but it happens the same (to a lesser extent) with the 0 in the PCD-10 case. Alternatively to these generated samples, we show the evolution of the LL during

[1]This adiabatic scheme is thus valid as long as the relaxation time for the RBM parameters during the learning is much larger than the MCMC mixing times.

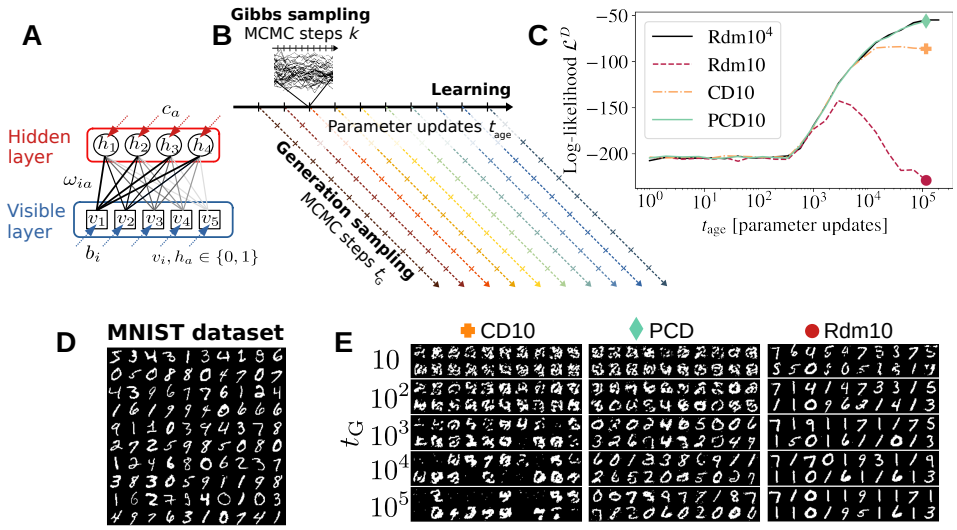

Figure 1: Sketch of an RBM in **A** and of the pipeline of our analysis in **B**. **C** Evolution of the LL of the dataset during the learning for 4 different RBMs trained using either a very long MCMC sampling to estimate the gradient ($k = 10^4$, Rdm-10^4) or short samplings $k = 10$ following different recipes for the initialization of the MC chains (CD-10, PCD-10 and Rdm-10). The large symbols mark the LL of the RBMs used to generate new samples in **E**. In **D** we show a subset of the MNIST database, and in **E** a subset of the samples drawn by the $k = 10$ RBMs marked in **C** through a MCMC sampling of the equilibrium Gibbs measure initialized at random and after 10,10^2,10^3,10^4 and 10^5 MCMC steps. Note that in our setup the MC chain are initialized at random, and we can observe that RBMs trained with CD-10 do not converge to any digit despite reaching higher value of the LL than Rdm10 which instead manage to create digits quite fast. Also note the lack of diversity of Rdm10 for large sampling time.

the learning of these same three RBMs in fig. 1–C. By definition of the loss function, the LL is intended to be maximized, so it constitutes a traditional measure of the quality of the RBM training. The LL of the models used for the generation sampling of 1–E are highlighted in large symbols. This plot illustrates the lack of reliability of the LL to monitor the quality of the learning. For instance, CD-10 reaches quite high values of LL and yet, it cannot generate a single digit. For the Rdm-10 is the other way around, it generates decent samples but with a very poor LL. Finally, we observe that the LL of PCD-10 reaches very high values comparable to RBMs trained using up to $k = 10^4$ MCMC steps to estimate the gradient. We will come back to this point. Our experiment highlights on the one hand, that the LL cannot be used all alone to quantify the generative power of an RBM, and on the other hand, that the quality of the generated samples may variate in a non-monotonic way during the generation sampling.

3 Related works

Recent years have witnessed a large number of studies trying to understand and control the learning of RBMs. In these works, meta-parameters are varied at the same time that an approximation of the LL [8, 31] of the train/test set is monitored throughout the learning process. The stability of the evolution of the LL and the maximum value achieved, are then used to compare the different recipes and parameters. For instance, in Ref. [8] the LL of different learning schemes are evaluated and the samples inspected visually. Unfortunately, no estimator is used to assess the quality of the samples, neither is clear the precise protocol used to generate these samples. In Ref. [9], the authors investigate the stability of the learning of RBMs with binary inputs[2] by comparing the learned features. They show (using mostly the LL and an eye inspection of the learned features) that a centered version of the gradient can be much stabler. In Ref. [7], the authors study, in a systematic way,

[2]It has been observed in MNIST that by flipping the input $0 \longleftrightarrow 1$, the usual gradient ascent was much slower in one case than in the other.

the convergence properties of CD, PCD and PT on several small toy models that can be analyzed exactly, that is, where the LL can be computed by brute-force. We can find more recent works [32, 33, 34, 35] improving the learning scheme for RBMs and yet, still not giving much information about the quality of the generated samples nor the equilibrium properties of the trained models. In our results below, we will show that without putting on the table this information, the comparison between methods or tuning of parameters, becomes extremely unstable.

Finally, we want to mention that it is particularly surprising that the CD recipe (with short k) is still used [36, 37, 38, 34]. As shown, RBMs trained with CD are not able to generate proper samples from scratch. And this behavior is true even if doing $k \approx O(100)$ MC steps at each update. The classical explanation for this is that CD is creating local energy well around the datapoint which are separated by large energy barriers [30], since the dynamics do not have time to relax when k is small. In the rest of the paper, we explain how, by characterizing the MT, we are able to explain the behavior of RBMs with respect to the different usual schemes.

4 Monitoring the learning performance of RBMs

In this work we wish to address the following questions: (i) How can one quantify that an RBM has correctly learned a dataset? (ii) has it learned the equilibrium model? (iii) when should one stop the learning and the sampling of new configurations? We already argued in sec. 2 that, alone, the LL of the dataset is not precise enough to address these questions. For this reason, we have considered several additional observables to quantify the generated samples' quality. We introduce shortly below the different metrics used in our analysis, the precise definitions of all of them can be found in the SM. All the observables are computed and averaged using sets of $N_{\rm s} = 1000$ true or generated samples.

Log-likelihood $\mathcal{L}$: On the contrary to other machine learning methods (such as classification), the LL cannot be exactly evaluated in an RBM without an exhaustive search of the configuration space. Thus we estimate the partition function using the Annealed Importance Sampling (AIS) method [39]. Details of our implementation can be found in SM. We call $\mathcal{L}^{\mathcal{D}}$ the LL of the (original) dataset with respect to the model, and $\mathcal{L}^{\rm RBM}$, the LL of the generated samples with respect to the model.

Error of the second moment $\mathcal{E}^{(2)}$: we compute the mean square error (MSE) between the covariance matrix computed with a set of samples generated by the RBM and one of the dataset.

Error in the power spectral density $\mathcal{E}^{\rm PSD}$: For images, we compute the log average power spectrum function of radial wave number distance for both a subset of the dataset and the generated set, and then compute the MSE between the two functions. In practice, we compute the error (between true and generated samples) on the logarithm of the squared module of the image's discrete 2D-Fourier transform elements A_{kl} at a fixed distance d in Fourier space

$$P(d) = \log\left(\langle \|A_{kl}\|^2 \rangle_{(k,l)|k^2+l^2=d^2}\right). \tag{5}$$

Error in the Adversarial Accuracy Indicator (AAI) $\mathcal{E}^{\rm AAI}$: The AAI was recently introduced in Ref. [40] with the goal of quantifying the resemblance and the "privacy" of data drawn by a generative model with respect to the training set. This metric is based on the idea of data nearest neighbors. We begin by defining two sets containing, each one, $N_{\rm s}$ samples: (i) a "target" set containing only generated samples $T \equiv \{\boldsymbol{v}_{RBM}^{(m)}\}_{m=1}^{N_s}$ and (ii) a "source" with samples from the dataset $S \equiv \{\boldsymbol{v}_{\mathcal{D}}^{(m)}\}_{m=1}^{N_s}$. Then, for each sample m in groups $\{T,\ S\}$, we compute the closest sample —using the euclidean distance– to the set groups $\{T,\ S\}$. For instance, the closest samples in $\{S\}$ for a sample m in $\{T\}$ would be $d_{TS}(m) = \min_{\rm n} \left\| {\rm v}_{\rm RBM}^{\rm (m)} - {\rm v}_{\mathcal{D}}^{\rm (n)} \right\|$. We can compute in the same way $d_{ST}(m)$, $d_{SS}(m)$ and $d_{TT}(m)$. Once we have all these 4 distances, we can estimate how often the "nearest neighbor" (the sample at the shortest distance) of each sample is contained in the same starting set, or in the other one. When the generated samples are statistically indistinguishable from real ones, both frequencies should converge to the value of a random guess 0.5 when $N_{\rm s}$ large. We therefore compute the MSE of these two frequencies with respect to the asymptotic value 0.5, which defines $\mathcal{E}^{\rm AAI}$. See SM for more details.

Relative entropy ΔS: We approximate the entropy of a given set of data by its byte size when compressed using gzip [41]. In particular, we compute the entropy of a group of $N_{\rm s}$ random samples

of the dataset, $S^{\mathcal{D}}$, and the entropy of another set of data of the same size, called $S^{\rm cross}$, composed in equal portions by samples of the train and generated set. Finally, we compute the relative size of this cross set with respect to the dataset's one, minus the expected value for identical sets, i.e. $\Delta S = S^{\rm cross}/S^{\mathcal{D}} - 1$. Then, a large ΔS indicates us that the generated set is less "ordered" than the dataset. On the contrary, a small ΔS reflects that the generated samples lack diversity.

As our goal is not to make a direct comparison between models, but rather to monitor the quality of the generated samples throughout the generating sampling stage. We have restricted our analysis to observables that can be computed online during the simulation. A more precise comparison between data distributions would require more complex measures such as the Inception score [42].

5 Results

Equipped with this set of quality measures, we now want to monitor and evaluate the learning process. To this end, we need to control precisely the learning dynamics and its interplay with the equilibration time. To do so, we try to isolate all the different factors affecting the learning, and for this reason, at a first stage, we avoid using the common tricks to speed up the estimation of the r.h.s. of the gradient of Eqs.4 during the learning. With this goal, we focus our first experiments on the Rdm-k schemes, and discuss the interplay with standard methods such as CD or PCD later on. The Rdm-k scheme is very practical to study the convergence to equilibrium for two reasons: (i) the MCMC chains are initialized from configurations that are uncorrelated with the dataset, and (ii) the sampling protocol used during the learning is identical to the one used to generate new samples from scratch with the learned RBM. The pipe-line of our experiments is summarized in Fig. 1–B. We split the analysis in two separated stages: learning and generation sampling. During the learning, we update the parameters for $\sim 2 \cdot 10^5$ times using the gradient estimated following the Rdm-k scheme (with $k = 10,\ 50,\ 100,\ 500,\ 10^3$ and 10^4). We save the RBM parameters at different numbers of updates equally spaced in logarithmic scale. The number of updates used to train a particular RBM will be referred as its *age*, $t_{\rm age}$. During the generation stage, we sample the equilibrium Gibbs measure of each RBM (of age $t_{\rm age}$) with a MCMC initialized at random. We evaluate the quality of the samples generated at Gibbs sampling times $t_{\rm G}$ equally spaced in logarithmic scale. Unless something else is mentioned, we will consider RBMs with $N_{\rm h} = 500$ hidden nodes and trained with a fixed learning rate of $\eta = 0.01$, hence the different RBMs will differ only by the value of k and $t_{\rm age}$. Further details on the learning parameters can be found in SM.

Evolution of the quality of the generated samples with the sampling time— We show in Fig. 2–A the evolution of the quality estimators of the generated samples throughout the generation sampling for RBMs of different ages (i.e. following the different color lines in Fig. 1–B), and trained with three different k. We recall that perfect samples, in the sense that they are indistinguishable from the dataset, would have a score 0 in all the quality observables with the exception of the LL, where $\mathcal{L}^{\rm RBM}$ should match $\mathcal{L}^{\mathcal{D}}$ ($\mathcal{L}^{\mathcal{D}}$ is marked as a horizontal line). For all k and early learning times (ages below $\lesssim 10^3$ updates in red-yellowish colors), all the metrics describe a constant behavior with $t_{\rm G}$ at roughly the same values obtained using a completely untrained RBM, and generally far away from 0. For an intermediate range of ages (pale blue colors), we observe an important improvement of the metrics with the RBM's age, up to a certain value (dark blue lines) above which time evolution curves start to collapse. Furthermore, for $k = 10$, only young ages converge to constant curves in $t_{\rm G}$, something that extends to intermediate ages for $k = 500$ and covers the majority of the ages for $k = 10^4$. In all the cases, the sampling of the oldest machine ($t_{\rm age} \sim$ 200k updates) describe a non-monotonous behavior in the quality measures, with a clear peak of best quality at $t_{\rm G} \sim k$. This is also the point where $\mathcal{L}^{\mathcal{D}}$ and $\mathcal{L}^{\rm RBM}$ intersect and where ΔS crosses 0. In Fig. 2–C, for each learning process using a different k, we plot the position of this $t_{\rm G}^{\rm best}$ (obtained for each quality metrics) against the value of k used to train the machine. We observe a perfect match between both times. At this point we want to stress that we obtain exactly the same qualitative behavior with the other considered datasets, as shown in the SM. Fig. 2–C reflects that these RBMs "memorize" the number of MCMC steps used to estimate the gradient during the learning phase. Memory effects are a clear signal of non-equilibrium effects [43] and in fact, this effect is not present as long as k is long enough to equilibrate the system during the learning. Indeed, we see that the larger k, the longer (in terms of number of updates) it takes to observe memory effects (for $k = 10^4$ we only appreciate out-of-equilibrium effects above 100k updates, while they appear around $t_{\rm age} \gtrsim 10^3$ updates in the case of $k = 10$). These two regimes (equilibrium and OOE) are also reproduced when using the CD

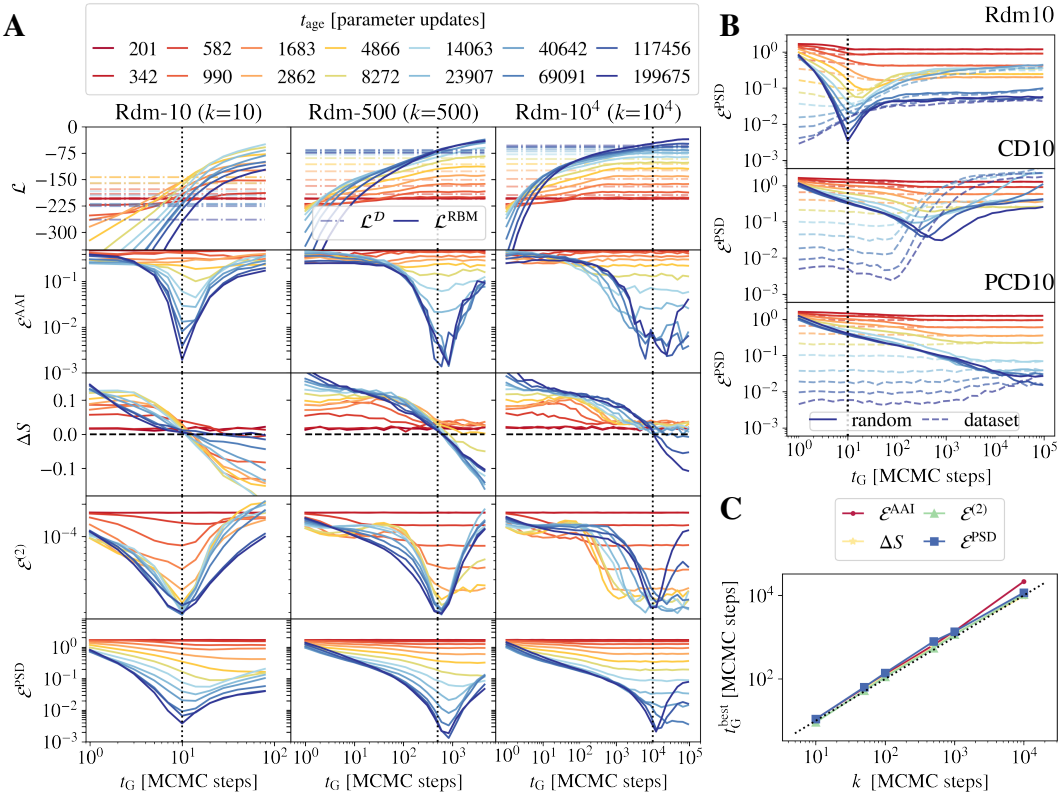

Figure 2: Evolution of the quality of the generated samples during the data generation sampling (along the color dashed lines of Fig. 1–B). In **A** we show the LL and the rest of quality estimators (introduced in Sect. 2) as function of the sampling time, $t_{\rm G}$, for chains initialized at random and using RBMs trained for an increasing number of updates $t_{\rm age}$ (age is coded by colors) using the Rdm-k scheme with $k = 10, 500, 10^4$. In general, the closer $\mathcal{E}$ and ΔS are to zero, the more similar the generated and original datasets are. For the LL, we show the value computed with the generated set ($\mathcal{L}^{\rm RBM}$, in continuous growing lines) and with the original dataset ($\mathcal{L}^{\mathcal{D}}$, in dashed horizontal lines). We observe that these two measures cross at roughly the same $t_{\rm G}$ at which all $\mathcal{E}$ reach a minimum and ΔS crosses zero. **B** We show the evolution of $\mathcal{E}^{\rm PSD}$ for 3 RBMs trained using the Rdm-10, CD-10 and PCD-10 recipes, for the same ages than in **A**. We consider two possible initializations of the Markov chain (random in full lines and dataset in dashed lines). **C** We plot the position of the best performance peak and the time at which $\Delta S = 0$ as function of the number of MCMC steps k used for estimating the negative term of the gradient.

and PCD schemes during the learning. We show in Fig. 2–B the evolution of $\mathcal{E}^{\rm PSD}$ of the generated samples as function of $t_{\rm G}$ for 3 RBMs trained with the 3 schemes and $k = 10$ steps (higher values of k are shown in the SM). We compare results with MCMC chains initialized either at random (continuous lines) or at the dataset (dashed lines). The sampling of the RBMs trained with CD-10 is also highly complex in time, and much slower than that of the Rdm-10 considering that the values of $\mathcal{E}^{\rm PSD}$ measured in runs from random initialization and from the dataset never converge to the same values for large $t_{\rm age}$. The situation is rather different for PCD-10, where the error sampling curves describe a very smooth behavior in time and a very slow convergence towards an equilibrium value. Yet According to the minimum values reached in $\mathcal{E}^{\rm PSD}$, the equilibrium samples of that model are significantly worse than the best samples created using the Rdm scheme (even working in the OOE regime). This difference is observed also in the rest of quality estimators and at much higher values of k, as shown in the SM.

Quantifying the mixing time: At this point, we try to quantify the MT or relaxation time of the MCMC dynamics of our RBM as a function of $t_{\rm age}$, to rationalize figs. 2. To do so, we compute the

time-autocorrelation function, ρ, of the averaged visible variables

$$\rho(t) = \frac{C_m(t)}{C_m(0)}, \text{ where } C_m(t) = \frac{1}{N_v}\sum_i (m_i(t) - \bar{m})(m_i(0) - \bar{m}) \tag{6}$$

where $m_i(t) = \text{sigmoid}(\sum_a w_{ia} h_a(t) + b_i)$ and $\bar{m}_i$ is the equilibrium magnetization.[3] At this point, let us stress that in a perfectly trained RBM, $\bar{m}_i$ would be equal to $1/N_v \sum_m^M v_i^{(m)}$. Yet, this is not the case in a poorly trained machine. For this reason, we estimate these equilibrium values from a long 10^5 MCMC steps simulation. The value of $\rho(t)$ will be close to 1 if the data at $t = 0$ and t are very similar, and to 0 if they are completely decorrelated. In order to measure properly the MT, we need to compute the autocorrelation function from equilibrium configurations. Therefore, we measure ρ with Eq. (6) but after discarding the first MCMC $\approx 10^4$ steps[4]. We show the time-correlation curves obtained for our best trained machine (using Rdm and $k = 10^4$) in Fig. 3–A for different values of t_{age}. Clearly, the older the machine, the slower the relaxation. Such a study is harder with the poorly trained RBMs because equilibration is achieved at times $t_{\text{G}} > 10^5$ for most of the t_{ages}, being particularly dramatic for Rdm-10 where relaxations are extremely slow, see SM. The decay being typically exponential, we can easily extract the MT (or the exponential relaxation time), t_α, from a fit of $\rho(t)$ to $A\exp(-t/t_\alpha)$. We show the dependence of t_α with t_{age} in Fig. 3–C. This t_α (for the slowest observable) controls the time-scale of the relaxations in the system, the time necessary to sample independent (uncorrelated) configurations, which means that the number of MCMC iterations necessary to reach equilibrium from the beginning of a MCMC simulation is expected to scale with it. Typically, the thermalization time, t_{therm}, is some few times this t_α [44]. This means that, in order to guarantee a sampling of equilibrium configurations, one needs Gibbs samplings longer than this t_{therm} (which scales with t_α). Note that t_{therm} (the distance to equilibrium) might depend on the MC starting point while t_α should not, yet t_{therm} will grow proportionally with t_α for any learning scheme. In fact, we can estimate independently the thermalisation time for the Rdm-k runs, t_{therm}, by comparing results from two independent simulations with different initializations: random and dataset, and wait until they converge to the same equilibrium expectation value. This means that one can estimate t_{therm} (for each t_{age}) by the number of MCMC steps needed to collapse the measures of the two trajectories to a constant value. See for instance Fig. 3–B, where we show the evolution of $\mathcal{E}^{\text{PSD}}$ with t_{G} for the two runs. The merging point for each t_{age} is highlighted with a square. We show the t_{therm} values we obtain with this procedure in Fig. 3–C for the different values of k.

As a result, these two times, t_α and t_{therm}, provide us with a lower and a safe upper bound (for instance, when initializing at random) for the minimum number of MCMC steps (for the learning's Gibbs sampling), k, needed to learn the equilibrium RBM model. As shown in Fig. 3–C, these times grow progressively with t_{age}, and in particular, falling always out of the range of iterations used to train the $k = 10$ runs (once the RBM starts to learn the dataset for $t_{\text{age}} \gtrsim 10^3$ updates), and falling out the range of scheme Rdm-500 for $t_{\text{age}} \gtrsim 10^4$ and for the last 2 t_{age} for the Rdm-10^4 scheme (i.e. for $t_{\text{age}} \gtrsim 10^5$ updates). In other words, at those ages, the learning enters in the OOE regime and the RBM starts to learn the dynamical process rather than the equilibrium properties. In fact, these ages mark the crossover between smooth and non-monotonic evolution of the observables in Fig. 2–A. This reinforces our previous claim that the apparition of the "best performance peak" in the error estimators is a purely non-equilibrium effect. A similar comment can be made for the LL. See for instance Fig. 2–A first row, for each t_{age}, we compare the LL measured on the generated $\mathcal{L}^{\text{RBM}}$ and original dataset $\mathcal{L}^{\mathcal{D}}$ as function of its generation sampling time t_{G}. We distinguish, again, two distinct behaviors associated to the equilibrium or OOE regime. In the first case, the LL of the generated samples converges to the value of the dataset and remain there even at large t_{G}. Whereas, in the OOE regime, $\mathcal{L}^{\text{RBM}}$ crosses $\mathcal{L}^{\mathcal{D}}$ at $t_{\text{G}} \approx k$ and continues growing afterwards. This highlights the problem of using the LL to evaluate the sample generation power. While the fact that the likelihood increases during the learning is in general a good sign, it is not necessary linked with the equilibrium properties of the model. In fact, the observation that $\mathcal{L}^{\text{RBM}}$ tends to reach much higher values than $\mathcal{L}^{\mathcal{D}}$ should be rather interpreted as a sign of poor learning, as we already showed in Fig. 1.

[3] The time-autocorrelation function of the visible, hidden or averaged hidden units are qualitatively the same and we will not show them here.

[4] In practice, we discard enough iteration steps so that we detect no dependence on the configuration used as $t = 0$ point.

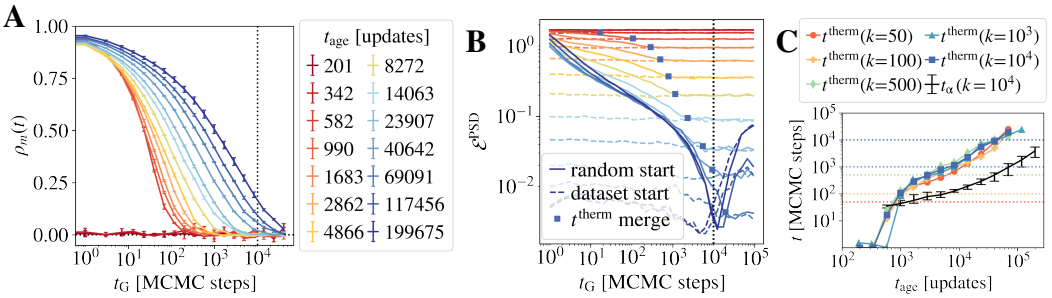

Figure 3: **A** The time-autocorrelation function of the visible mean magnetizations, ρ_m (see Eq. (6)), computed during the sampling of an RBM trained with Rdm-10^4 for different number of updates $t_{\rm age}$. This function carries the information about the number of MCMC steps that are needed (in equilibrium) to forget a initial configuration of the MC chain. Clearly, the time needed to decorrelate grows with the age of the RBM. In **B**, for the same RBMs, we show the evolution of the generated samples' quality through the $\mathcal{E}^{\rm PSD}$ observable, as function of the sampling time t_G, and for two different initializations: from the dataset and from random. We mark the times $t^{\rm therm}$ beyond which both runs converge to the same constant value. **C** We compare the mixing times, t_α, obtained from a fit of the curves $\rho_m(t)$ in **A** to the form $\exp(-t/t_\alpha)$, with the convergence-to-equilibrium times $t^{\rm therm}$ obtained in **B** for RBMs trained with different values of k. We see that the equilibration time $t^{\rm therm}$ follow the same trend for various values of k.

As a direct consequence of the results discussed above, we can see that even in the case where sampling is openly OOE, such as with the Rdm-10 scheme, one can generate fairly good samples as long as one limits the generation sampling to $t_{\rm G} \sim k = 10$, as shown in Fig. 1: digits are reproduced with uniform probability, and they look OK in visual inspection. For longer samplings we start to get clear unbalanced distributions. It is also quite remarkable that when matching $t_{\rm G}$ with k in the OOE regime of Rdm schemes, we get better quality samples than with the standard CD o PCD schemes even at very long sampling times, as we can see from the comparison of $\mathcal{E}^{\rm PSD}$ in Fig. 2–B. This is also true for larger values of k as shown in the SM.

6 Conclusions

In this work, we show that the training of an RBM is richer and more complex than what previously thought. We show that the choice of the number of MCMC steps, k, used to estimate the gradient during the learning has a dramatic impact on the learned model's dynamics and equilibrium states. In practice, we find two regimes: (i) when k is much lower than the MCMC mixing time, the trained RBM model encodes the dynamical process followed to train it, resulting in a highly non-linear, unstable and slow evolution of the quality of the generated samples with the sampling time, and a convergence towards poor and strongly biased samples. And (ii), when k overpasses largely the mixing time, the marginal equilibrium probability distribution of the trained model (over the visible layer) matches that of the dataset. In such case, the quality of the generated samples improves smoothly in time converging faster to the equilibrium values, converging much faster than in the other regime. While regime (ii) is needed to obtain good trained models, we show that regime (i) is particularly well suited for sample generation because the data sampling non-linearities can be controlled in the Rdm-k approach, and exploited to generate high quality data in short training times. We further show that the MC mixing time increases significantly as the learning progresses, being always much larger than the standard ks used in the Literature. Hence, our findings reveal a crucial competing mechanism in the training process of RBMs, which has all the reasons to be also present and influential in other models like Deep Boltzmann Machine (DBM). Interestingly, our results should simplify the training and the use of these deep models at least in the OOE regime. More generally, this work provides a set of key points to be checked when working with RBMs or DBMs, and a new fundamentally distinctive attribute to theses generative models: equilibrium or OOE. We leave for future works systematic comparisons between the models trained in each of these regimes, and in particular, the effect on the features. For instance, we observed that RBMs trained with low k suffer strong aging effects extremely similar to those observed on the spin glasses [45]. Yet, these effects do not concern RBMs trained in equilibrium.

References

[1] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[2] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649, 2008.

[3] R.D. Hjelm, V.D. Calhoun, R. Salakhutdinov, E.A. Allen, T. Adali, and S.M. Plis. Restricted Boltzmann machines for neuroimaging: an application in identifying intrinsic networks. *NeuroImage*, 96:245–260, 2014.

[4] X. Hu, H. Huang, B. Peng, J. Han, N. Liu, J. Lv, L. Guo, C. Guo, and T. Liu. Latent source mining in fmri via restricted Boltzmann machine. *Human brain mapping*, 39(6):2368–2380, 2018.

[5] Jérôme Tubiana, Simona Cocco, and Rémi Monasson. Learning protein constitutive motifs from sequence data. *Elife*, 8:e39397, 2019.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Asja Fischer and Christian Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47(1):25–39, 2014. Number: 1.

[8] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879, 2008.

[9] Jan Melchior, Asja Fischer, and Laurenz Wiskott. How to center deep boltzmann machines. *The Journal of Machine Learning Research*, 17(1):3387–3447, 2016.

[10] Marylou Gabrié, Eric W Tramel, and Florent Krzakala. Training restricted Boltzmann machine via the Thouless-Anderson-Palmer free energy. In *Advances in neural information processing systems*, pages 640–648, 2015.

[11] Peter de Boves Harrington. Feature expansion by a continuous restricted boltzmann machine for near-infrared spectrometric calibration. *Analytica chimica acta*, 1010:20–28, 2018.

[12] Huihui Shen and Hongwei Li. A gradient approximation algorithm based weight momentum for restricted boltzmann machine. *Neurocomputing*, 361:40–49, 2019.

[13] Ramasamy Savitha, ArulMurugan Ambikapathi, and Kanagasabai Rajaraman. Online rbm: Growing restricted boltzmann machine on the fly for unsupervised representation. *Applied Soft Computing*, 92:106278, 2020.

[14] Guido Montúfar. Restricted boltzmann machines: Introduction and review. In *Information Geometry and Its Applications IV*, pages 75–115. Springer, 2016.

[15] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. Equivalence of restricted boltzmann machines and tensor network states. *Physical Review B*, 97(8):085104, 2018.

[16] Aurélien Decelle, Giancarlo Fissore, and Cyril Furtlehner. Thermodynamics of restricted Boltzmann machines and related learning dynamics. *Journal of Statistical Physics*, 172(6):1576–1608, 2018.

[17] Kai Shimagaki and Martin Weigt. Selection of sequence motifs and generative hopfield-potts models for protein families. *Physical Review E*, 100(3):032128, 2019.

[18] Roger G Melko, Giuseppe Carleo, Juan Carrasquilla, and J Ignacio Cirac. Restricted boltzmann machines in quantum physics. *Nature Physics*, 15(9):887–892, 2019.

[19] Moshir Harsh, Jérôme Tubiana, Simona Cocco, and Remi Monasson. 'place-cell'emergence and learning of invariant data with restricted Boltzmann machines: breaking and dynamical restoration of continuous symmetries in the weight space. *Journal of Physics A: Mathematical and Theoretical*, 53(17):174002, 2020.

[20] Burak Yelmen, Aurélien Decelle, Linda Ongaro, Davide Marnetto, Corentin Tallec, Francesco Montinaro, Cyril Furtlehner, Luca Pagani, and Flora Jay. Creating artificial human genomes using generative neural networks. *PLoS genetics*, 17(2):e1009303, 2021.

[21] Barbara Bravi, Jérôme Tubiana, Simona Cocco, Rémi Monasson, Thierry Mora, and Aleksandra M Walczak. Rbm-mhc: A semi-supervised machine-learning method for sample-specific prediction of antigen presentation by hla-i alleles. *Cell systems*, 12(2):195–202, 2021.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Vincenza Colonna, Qasim Ayub, Yuan Chen, Luca Pagani, Pierre Luisi, Marc Pybus, Erik Garrison, Yali Xue, and Chris Tyler-Smith. Human genomic regions with exceptionally high levels of population differentiation identified from 911 whole-genome sequences. *Genome biology*, 15(6):1–14, 2014.

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[25] V. Nair and G.E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[26] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[27] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[28] Koji Hukushima and Koji Nemoto. Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, 1996.

[29] Russ R Salakhutdinov. Learning in markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[30] Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered markov chain monte carlo for training of restricted boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 145–152. JMLR Workshop and Conference Proceedings, 2010.

[31] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Artificial Intelligence and Statistics*, pages 102–110. PMLR, 2015.

[32] Roger Grosse and Ruslan Salakhudinov. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2015.

[33] Oswin Krause, Asja Fischer, and Christian Igel. Population-contrastive-divergence: Does consistency help with rbm training? *Pattern Recognition Letters*, 102:1–7, 2018.

[34] Enrique Romero, Ferran Mazzanti, Jordi Delgado, and David Buchaca. Weighted contrastive divergence. *Neural Networks*, 114:147–156, 2019.

[35] Vidyadhar Upadhya and PS Sastry. Efficient learning of restricted boltzmann machines using covariance estimates. In *Asian Conference on Machine Learning*, pages 836–851. PMLR, 2019.

[36] Ken-Ichi Aoki and Tamao Kobayashi. Restricted boltzmann machines for the long range ising models. *Modern Physics Letters B*, 30(34):1650401, 2016.

[37] Alan Morningstar and Roger G Melko. Deep learning the ising model near criticality. *arXiv preprint arXiv:1708.04622*, 2017.

[38] David Yevick and Roger Melko. The accuracy of restricted boltzmann machine models of ising systems. *Computer Physics Communications*, 258:107518, 2021.

[39] Oswin Krause, Asja Fischer, and Christian Igel. Algorithms for estimating the partition function of restricted boltzmann machines. *Artificial Intelligence*, 278:103195, 2020.

[40] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P Bennett. Generation and evaluation of privacy preserving synthetic health data. *Neurocomputing*, 416:244–255, 2020.

[41] Andrea Baronchelli, Emanuele Caglioti, and Vittorio Loreto. Measuring complexity with zippers. *European journal of physics*, 26(5):S69, 2005.

[42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[43] Nathan C Keim, Joseph D Paulsen, Zorana Zeravcic, Srikanth Sastry, and Sidney R Nagel. Memory formation in matter. *Reviews of Modern Physics*, 91(3):035002, 2019.

[44] Alan Sokal. Monte carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration*, pages 131–192. Springer, 1997.

[45] Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.

7 Paper checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] our claims have been checked with experimental results over three distinct datasets. We expect them to be valid in general.
   (b) Did you describe the limitations of your work? [Yes] Even though our main contribution is to explain a practical and general mechanism of the RBM's learning, we explicitly specify the practical impact it can have. In particular, the relaxation time of RBMs can be very large what limits strongly the expectations of learning good equilibrium models or very high-dimensional datasets. We alert this could have an impact of the reliability of the features extracted from a OOE machine.
   (c) Did you discuss any potential negative societal impacts of your work? [No] we do not discuss negative societal impact since we do not believe it applies for this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] we have read the ethics review guidelines and our paper conforms them or/and our work do not produce any specific risk on these lines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our work do not include theoretical results
   (b) Did you include complete proofs of all theoretical results? [N/A] Our work do not include theoretical results
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We released a version of the code on github with some scripts to reproduced part of the main experimental results. The dataset are public.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The training details are specified in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] and no... the curves that are of interest for their trend have been made without error bar. These curves have been however averaged over a 10^3 realization in order to make the curve smooth enough. The auto-correlation curves have error bar in order to demonstrate clearly our point on the relaxation time, and errors are also computed for the computation of the relaxation time.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We put in the SM estimate of the training time of our experiments and the used material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] we cite the original paper associate to each of the dataset.
   (b) the rest does not apply for us
   (c) Did you mention the license of the assets? [N/A]
   (d) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (e) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (f) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]