COMMUNICATIONS PHYSICS

ARTICLE

https://doi.org/10.1038/s42005-020-0373-8 OPEN

Mode-assisted unsupervised learning of restricted Boltzmann machines

Haik Manukian [1,2]✉, Yan Ru Pei[1,2]✉, Sean R. B. Bearden[1] & Massimiliano Di Ventra [1]✉

Restricted Boltzmann machines (RBMs) are a powerful class of generative models, but their training requires computing a gradient that, unlike supervised backpropagation on typical loss functions, is notoriously difficult even to approximate. Here, we show that properly combining standard gradient updates with an off-gradient direction, constructed from samples of the RBM ground state (mode), improves training dramatically over traditional gradient methods. This approach, which we call 'mode-assisted training', promotes faster training and stability, in addition to lower converged relative entropy (KL divergence). We demonstrate its efficacy on synthetic datasets where we can compute KL divergences exactly, as well as on a larger machine learning standard (MNIST). The proposed mode-assisted training can be applied in conjunction with any given gradient method, and is easily extended to more general energy-based neural network structures such as deep, convolutional and unrestricted Boltzmann machines.

[1] Department of Physics, University of California, San Diego, La Jolla, CA 92093, USA. [2] These authors contributed equally: Haik Manukian, Yan Ru Pei. ✉email: hmanukia@ucsd.edu; yrpei@ucsd.edu; diventra@physics.ucsd.edu

Boltzmann machines[1] and their restricted version (RBMs), are unsupervised generative models applied to a variety of machine learning problems[2]. They enjoy a universal approximation theorem for discrete probability distributions[3], are used as building blocks for deep-belief networks[4] and, in no small feat, can even represent correlated states in quantum many-body systems[5,6]. Despite these advantages, the limitations of the most popular training algorithms for RBMs, combined with rapid advances in supervised learning techniques, have led to the sideline of their unsupervised learning, known also as "pretraining", in favor of supervised backpropagation from random initial conditions[4].

However, supervised learning requires large datasets of labeled examples, and even then, state-of-the-art neural networks have been shown to be vulnerable to what are called adversarial examples[7], or slight perturbations of the input that 'fool' the network. On the other hand, unsupervised pretraining is possible in the absence of labels, and is known to be a strong regularizer[8], often resulting in better generalization for supervised models. Since adversarial vulnerability could be seen as a failure in generalization, an improvement in unsupervised training could lead to more robust performance in a downstream task. This motivates the search for better unsupervised training methods.

To set the stage for our training method, we provide a short introduction to RBMs and then describe standard training approaches and their limitations. RBMs are undirected weighted graphs with a bipartite structure that differentiates between n visible nodes, $\mathbf{v} \in \{0, 1\}^n$ and m latent, or 'hidden', nodes, $\mathbf{h} \in \{0, 1\}^m$, not directly constrained by the data[2]. (Note that an RBM does not allow connections within a layer.) These states are usually taken to be binary but can be generalized. Each state of the machine corresponds to an energy of the form

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{v}^T\mathbf{W}\mathbf{h}, \tag{1}$$

where the biases $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and weights $\mathbf{W} \in \mathbb{R}^{n\times m}$ are the learnable parameters. This induces a distribution over states given by a Boltzmann-Gibbs distribution,

$$p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\mathcal{Z}}. \tag{2}$$

The normalizing factor, $\mathcal{Z} = \sum_{\{\mathbf{v}\}}\sum_{\{\mathbf{h}\}} e^{-E(\mathbf{v},\mathbf{h})}$, is the formidable partition function that involves the sum over an exponentially scaling number of states, making the exact computation of its value infeasible. Additionally, the bipartite structure of the RBM makes the hidden nodes conditionally independent given the visible nodes (and vice versa), with a closed form conditional distribution given by[9] $p(h_j = 1|\mathbf{v}) = \sigma(\sum_i w_{ij}v_i + b_j)$, where $\sigma(x) = (1 + e^{-x})^{-1}$.

RBMs can learn to represent some unknown data distribution over the visible layer, $q(\mathbf{v})$, typically specified by a data set of samples. We indicate the unique elements of this data set as $\mathcal{D} = \{\mathbf{v}_1, \cdots, \mathbf{v}_{n_d}\} \subset \Omega$, where Ω is the space of all binary sequences of length n. Let us assume further that all data points have equal amplitude over the support, i.e., $p_i = 1/n_d$. Since most real world datasets consist of unique elements with no exact repeats, this class of distributions includes all relevant ones.

The RBM is tasked to match its marginal distribution over the visible layer, $p(\mathbf{v}) = \Sigma_{\{\mathbf{h}\}}p(\mathbf{v}, \mathbf{h})$, to this unknown data distribution. The former can be written as,

$$p(\mathbf{v}) = \frac{1}{\mathcal{Z}}\prod_{i=1}^{n} e^{a_i v_i}\prod_{j=1}^{m}\left(1 + e^{b_j + \sum_{i'=1}^{n} w_{i'j}v_{i'}}\right). \tag{3}$$

Training an RBM then amounts to a search for the appropriate weights and biases that will minimize the quantity known as the Kullback-Leibler (KL) divergence between the two distributions,

$$\mathrm{KL}\,(q||p) = \sum_{\{\mathbf{v}\}} q(\mathbf{v})\log\frac{q(\mathbf{v})}{p(\mathbf{v})}. \tag{4}$$

The optimization of Eq. (4) is typically done via stochastic gradient descent with respect to the RBM parameters, which leads to weight updates of the form[10],

$$\Delta w_{ij} \propto \left[\langle v_i h_j\rangle_{q(\mathbf{v})p(\mathbf{h}|\mathbf{v})} - \langle v_i h_j\rangle_{p(\mathbf{v},\mathbf{h})}\right]. \tag{5}$$

The first term on the rhs of Eq. (5) is an expectation with the hidden nodes driven by the data, and is referred to as the "data term". Since the conditional distributions across the hidden nodes are factorial and known in closed form, this inference problem is easy in the RBM case. The second term on the rhs of Eq. (5), instead, is an expectation over the model distribution with no nodes fixed, and called the "model term". The exact calculation of this term requires computing the partition function of the RBM, which is proved to be hard even to estimate[11].

This expectation value is popularly approximated by a Markov Chain Monte Carlo (MCMC) procedure dubbed 'contrastive divergence' (CD)[9]. Since CD initializes Markov chains from elements of the dataset, difficulties arise when the model distribution represented by the RBM contains modes where the data distribution has negligible probability, sometimes referred to as 'spurious modes'. The prohibitively slow mixing due to random-walk exploration, and typical high dimensionality of the problem render it difficult for CD to find and correct the probability of these modes.

In a recent work, a memcomputing-assisted training scheme for RBMs[12] was proposed to address this training difficulty. Memcomputing[13] is a novel computing paradigm in which memory (time non-locality) assists in the solution of hard decision and optimization problems[14,15]. In that previous work[12], the difficult model expectation in Eq. (5) was replaced by a sample of the RBM mode obtained by a memcomputing solver, which led to better quality solutions versus CD in a downstream classification task. However, despite demonstrating a significant reduction in the number of pretraining iterations necessary to achieve a given level of classification accuracy, as well as a total performance gain over CD, that algorithm[12] does not fully exploit samples of the mode. In particular, it does not give rise to training advantages over standard methods in the unsupervised setting.

In the present work, we overcome this limitation. We show that by appropriately combining the RBM's mode (ground state) samples and data initiated chains (as in CD) not only improves considerably the model quality over CD and other MCMC procedures, but also improves the stability of the pre-training routine. Specifically, we introduce (i) a principled schedule for incorporating samples of the RBM ground state into pre-training, (ii) an appropriate mode-driven learning rate, (iii) show comparisons to other state-of-the-art unsupervised pre-training approaches without the need of supervised fine-tuning, and (iv) provide proofs of advantageous properties of the method. To corroborate our method, we will show exact KL/log-likelihood achieved on small synthetic datasets and on the MNIST dataset. In all cases, we find that mode-assisted training is able to learn more accurate models than several other training methods such as CD, persistent contrastive divergence (PCD), parallel tempering (PT) used in tandem with enhanced gradient RBMs (E-RBMs), and centered RBMs (C-RBMs)[16].

Results

After these preliminaries we can now describe our mode-assisted training method. It consists in replacing the average in the model

term of Eq. (5) with the mode of $p(\mathbf{v})$ at appropriate steps of the training procedure. However, $p(\mathbf{v})$ is very cumbersome to compute (see Eq. (3)), thereby adding a considerable computational burden. Instead, we sample the *mode* of $p(\mathbf{v}, \mathbf{h})$, the model distribution of the RBM.

The rationale for replacing the mode, $\mathbf{v}^+$, of $p(\mathbf{v})$ with the visible configuration of the mode, $\mathbf{v}^*$, of $p(\mathbf{v}, \mathbf{h})$ is because the two modes are equivalent with high probability under scenarios typical for different stages of the RBM pre-training. We prove this in the Supplementary Note 1, while here we provide numerical evidence of this fact.

Mode correspondence of joint and marginal distributions. To illustrate the equivalence between the modes of $p(\mathbf{v})$ and $p(\mathbf{v}, \mathbf{h})$, let us begin by expressing the joint probability mass function (PMF) in terms of the product of the marginal PMF over the visible layer and the conditional PMF over the hidden layer $p(\mathbf{v}, \mathbf{h}) = p(\mathbf{v})p(\mathbf{h}|\mathbf{v})$. For any given visible configuration $\mathbf{v}$, we then have $\max_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) = p(\mathbf{v})[\max_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})]$. We can then define the hidden "activation" of $\mathbf{v}$ to be

$$r(\mathbf{v}) = \max_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}), \tag{6}$$

which allows us to write $\max_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) = p(\mathbf{v})r(\mathbf{v})$. Note that we can interpret $r(\mathbf{v})$ as a measure of the "certainty" that the hidden nodes acquire the value 0 or 1.

It is then clear that we can write the probability amplitude of the mode of the joint PMF as $\max_{\{\mathbf{v},\mathbf{h}\}} p(\mathbf{v}, \mathbf{h}) = \max_{\mathbf{v}}(\max_{\mathbf{h}} p(\mathbf{v}, \mathbf{h})) = \max_{\mathbf{v}}(p(\mathbf{v})r(\mathbf{v})) \leq \max_{\mathbf{v}}(p(\mathbf{v})) = p(\mathbf{v}^+)$, where we have used the fact that $r(\mathbf{v}) \leq 1$ and $\mathbf{v}^+$ is the mode of the marginal PMF, $p(\mathbf{v})$. If $r(\mathbf{v}^+) = 1$ then we have modal equivalence of the joint and marginal PMFs.

In Fig. 1, we plot the evolution of $r(\mathbf{v}^+)$ as a function of the number of CD-1 training iterations for a shifting bar synthetic dataset, which is small enough that we can compute the exact mode of $p(\mathbf{v})$ at any iteration. The figure indeed shows that $r(\mathbf{v}^+)$ approaches 1 rather quickly as pre-training proceeds. The activation of a random visible configuration is being used as comparison.

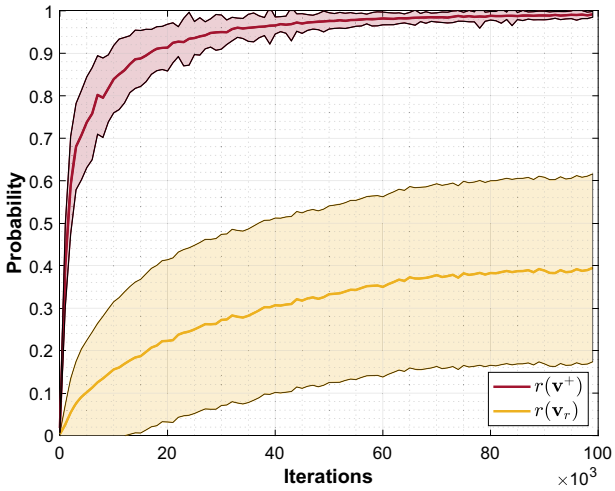

Fig. 1 Maximal conditional probability during training. We plot, in red, the maximal conditional probability of the hidden layer, $r(\mathbf{v}^+)$, where $\mathbf{v}^+$ is the mode of the marginal distribution, $p(\mathbf{v})$, as a function of contrastive divergence training iterations ($k = 1$). The results are averaged over an ensemble of 200 randomly initialized 15×10 restricted Boltzmann machines trained on a shifting bar dataset, with $\pm 1\sigma$ defining the shaded regions, where σ is the standard deviation. The same calculation conditioned on a random visible configuration, $\mathbf{v}_r$, is plot as a baseline for comparison in yellow.

In the Supplementary Note 1 we also prove that the condition of $r(\mathbf{v}^+)$ being close to 1 is not necessary for establishing modal equivalence. In fact, we prove that it is still possible for the two modes to be equal even when the weights are small (thus a smaller $r(\mathbf{v}^+)$ value). Additionally, we show in the Supplementary Note 1 that mode-assisted training is more effective in exploring the PMF of the model distribution for RBM instances of greater frustration. The latter is a measure of the degeneracy of the low-energy states of an RBM, and thus the difficulty of finding the ground state configuration. Since it was shown that the frustration of the RBM increases as pre-training proceeds[17], in order to effectively utilize the power of mode-assisted training, the frequency of mode updates should be higher at the later stages of the training than the earlier stages.

Optimal mode-training schedule. The results from the previous subsection then suggest a schedule for the mode-assisted training routine that performs mode updates more frequently the longer the pre-training routine has elapsed.

To realize this, we use a sigmoid, σ, to calculate the probability of replacing the data driven hidden CD term with a mode driven term at the iteration step n

$$P_{\text{mode}}(n) = P_{\max}\sigma(\alpha n + \beta). \tag{7}$$

Here, $0 < P_{\max} \leq 1$ is the maximum probability of employing a mode update, and α and β are parameters that control how the mode updates are introduced into the pre-training. They are chosen such that the frequency of mode updates becomes dominant only when both the conditions of large weights and frustration are met (see the "Methods" section for the value of these parameters). Initially, P_{mode} will be small, since the joint- and marginal-distribution modes are unequal, and gradually rises to its maximal value when the modes are of equal magnitude. Note that one may employ different functions to quantify the degree to which the joint- and marginal-distribution modes equalize during training. However, we have found that the sigmoid works well enough in practice.

Combining Markov chains with mode updates. We are now ready to outline the full procedure of mode-assisted training, that combines a MCMC method with the mode updates following the schedule (7). Although one may choose any variation of the MCMC method to train RBMs, for definiteness of discussion, we consider here the standard training method, CD[9]. In this case, weight updates follow the modified KL($q||p$) gradient. As discussed in the introduction, it evaluates to a difference of two expectations called the data term and model term which we can write as

$$\Delta w_{ij}^{\text{CD}} = \epsilon^{\text{CD}}\left[\langle v_i h_j\rangle_{q(\mathbf{v})p(\mathbf{h}|\mathbf{v})} - \langle v_i h_j\rangle_{p^k(\mathbf{v},\mathbf{h})}\right], \tag{8}$$

where ϵ^{CD} is the CD update learning rate, and the expectation in the second term is taken over the reconstructed distribution over a Markov chain initialized from the dataset after k Gibbs samples ($k = 1$ in most cases). When driving the weights with samples of the RBM ground state with the schedule (7), we use instead the following update,

$$\Delta w_{ij}^{\text{mode}} = \epsilon^{\text{mode}}\left[\langle v_i h_j\rangle_{q(\mathbf{v})p(\mathbf{h}|\mathbf{v})} - [v_i h_j]_{p(\mathbf{v},\mathbf{h})}\right], \tag{9}$$

where $[\ ]_p$ is the mode of the joint RBM model distribution. Note that the mode update learning rate, ϵ^{mode}, may be different from the CD learning rate, ϵ^{CD}.

We also stress that the updates in Eq. (9) are in an off-gradient direction. As we show now, this is the reason for the increased

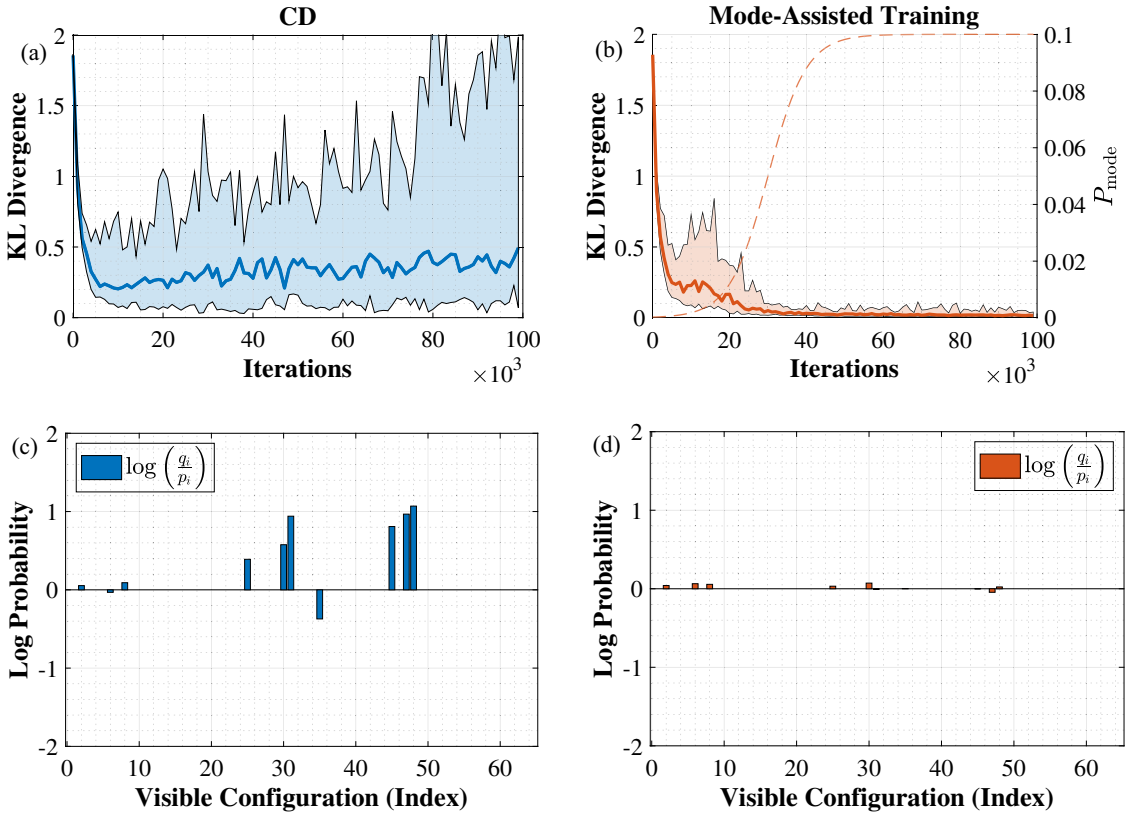

Fig. 2 Comparison of typical features in training. A training performance comparison between contrastive divergence (CD) with $k = 1$ steps of the Markov chain (in blue), and mode-assisted training (in orange) across 25 randomly generated 6 × 6 restricted Boltzmann machines with a random uniform data set of size $n_d = 10$. **a**, **b** Kullback-Leibler (KL) divergence as a function of training iterations of CD and mode-assisted training, respectively. Median KL divergence shown as the solid curves, with the shaded region defined by the maximum/minimum KL divergence at that point in training. The mode sampling probability in mode-assisted training, P_{mode}, is shown as the dotted line in **b**. **c**, **d** The median log-differences in probability between the data (q_i) and model (p_i) distributions for CD and mode-assisted training, respectively. In both cases, the learning rate was a constant $\epsilon^{\text{CD}} = 0.05$ for 100,000 iterations.

stability of the training over MCMC approaches, and its convergence to arbitrarily small KL divergences.

Stability and convergence. The data term, which is identical in both Eqs. (8) and (9), tends to increase the weights associated with the visible node configurations in the dataset, thereby increasing their relative probabilities compared with states not in the support set, $\mathbf{v} \in \Omega \backslash \mathcal{D}$. Instead, the model term decreases the weights/probability corresponding to highly probable model states, or equivalently, minimizes the partition function[2]. CD does this poorly and often diverges, while mode-assisted training achieves this with better stability and faster convergence (see Fig. 2). We provide here an intuitive explanation of this phenomenon, while a formal treatment on this topic will be provided in the Supplementary Note 2.

The pre-training routine can be broken down in two phases. In the first phase, the training procedure attempts to discover the support $\mathcal{D}$ of the data distribution $q(\mathbf{v})$. We call this phase the "discovery phase". To better see this, consider a randomly initialized RBM with small weights. These small and uncorrelated weights give rise to RBM energies close to zero for all nodal states, or $E(\mathbf{v}, \mathbf{h}) \approx 0$ for all $\mathbf{v}$ and $\mathbf{h}$, see Eq. (1). This results in the model distribution $p(\mathbf{v}, \mathbf{h})$ being almost uniform.

Therefore, we see that in the discovery phase of training, the model term plays little role in the training as it simply pushes down on the weights in a practically uniform manner, with $\langle v_i h_j \rangle_{\text{M}} \approx 0.25$. On the other hand, the data term drives the initial phase of the training by increasing the marginal probability of the visible states in the support, $\mathbf{v} \in \mathcal{D}$. We can then employ a large learning rate (say, $\epsilon^{\text{CD}} = 1$) in the beginning of the training, driving the visible layer configurations in the dataset, $\mathcal{D}$, to high probability versus configurations outside the support. Empirically, we find that CD training performs in the discovery phase reasonably well, and is quickly able to "find" the visible states in the support.

Now, having discovered the support, we arrive at the second phase of the training where we have to bring the model distribution as close to uniformity as possible over the support in order to minimize the KL-divergence. We call this phase the "matching phase" of the training, where we bring the model distribution as close to the data distribution as possible. CD usually performs poorly in this phase (see Fig. 2). To see this most directly, we simply have to consider a visible state with a slightly larger probability than the other states. It should then be necessary for the model term to locate and "push down" on this state to lower its probability, and in turn, increase the uniformity of the distribution over the support. However, for any CD approximation of the model term, this rarely happens in a timely manner as the mixing rate of the MCMC chain is far too slow to locate this state before the training diverges.

This is where samples of the mode are most effective, and can assist in the correction of the states' amplitudes. As we have anticipated, finding the modal state, $\mathbf{v}^*$, of the model distribution, $p(\mathbf{v}, \mathbf{h})$ allows us to immediately locate the mode, $\mathbf{v}^+$, of the marginal probability, $p(\mathbf{v})$, and decrease the weights

corresponding to that state, which in turn "pushes down" on the probability of this state through an iteration of weight updates. This "push" may result in another state "popping" up and becoming the new modal state. However, often times the probability amplitude of this new state will be less than that of the previous mode. This results in a training routine that "cycles" through any dominant state that emerges at each iteration, and the probability amplitude of the mode decreases as training proceeds until the probability amplitudes of all the states in the support become equal (see the formal demonstration of this in the Supplementary Note 2), which results in the desired uniform distribution over the support. This can be visualized as a "seesaw" between the dominant states, with the oscillation amplitude of this seesaw decaying to zero in time.

We outline the pseudo-code for mode-assisted training in Algorithm 1 and a visual depiction of the training side by side with CD-1 on a small data set is shown in Fig. 2. KL divergences for CD (Fig. 2a) and mode-assisted training (Fig. 2b) are presented in the top row. In Fig. 2c, the length of the bars above or below zero imply an over- or under-estimation, respectively, of probability to a given data vector by CD. In contrast, Fig. 2d shows that the mode-assisted training algorithm searches for the state with highest probability, and decreases the weights corresponding to that state, which in turn decreases the probability assigned to it by the model, aligning it closer to the data distribution. This prevents any probability from accumulating far away from the data distribution and eventually achieves a close to perfect model. The average height of the individual divergences is exactly the KL divergence plot in Fig. 2a, b.

As it should now be clearer, these mode-driven updates are deviations from the gradient direction, since in general the mode over the model distribution is different from the expected value. This makes the mode-training algorithm, which mixes mode driven samples and data-driven ones, distinct from gradient descent. This is also supported by the fact that our method tends toward a particular class of distributions (uniform), when gradient descent would settle in some local minima or saddle points in the KL landscape.

Algorithm 1. Unsupervised learning of a restricted Boltzmann machine with the mode-assisted training algorithm

procedure MT($P_{\max}, \alpha, \beta, \{\epsilon_n^{\mathrm{CD}}\}_{n=1}^{N}, N$)
  $\theta_0 \sim \mathcal{N}(0, 0.01)$
  for $i = 1; i \le N; i++$ **do**
    $p_{\mathrm{mode}} \leftarrow P_{\max}\sigma(\alpha i + \beta)$
    Sample $u \sim$ U[0, 1]
    if $u \le p_{\mathrm{mode}}$ **then**
      $\mathbf{v}^*, \mathbf{h}^*, E_0 \leftarrow \mathrm{argmin} E(\mathbf{v}, \mathbf{h})$
      $\gamma \leftarrow \frac{-E_0}{(n+1)(m+1)}$
      $\theta_i \leftarrow \theta_{i-1} + \gamma\epsilon_i^{\mathrm{CD}}\Delta\theta^{\mathrm{mode}}$ ▷ Eq. (9)
    else
      $\theta_i \leftarrow \theta_{i-1} + \epsilon_i^{\mathrm{CD}}\Delta\theta^{\mathrm{CD}}$ ▷ Eq. (8)
    end If
  end for
  return θ_N
end procedure

The free parameters in this method are the schedules of the mode sample using $P_{\mathrm{mode}}(n)$ (defined by $P_{\max}$, α and β in Eq. (7)) and the CD learning rate, ϵ^{CD}. With ϵ^{CD} fixed, we set $\epsilon^{\mathrm{mode}} = \gamma\epsilon^{\mathrm{CD}}$, where $\gamma = -E_0/[(n+1)(m+1)]$, with $E_0(<0)$ being the ground state of the corresponding RBM with nodal values $\{-1, 1\}^{n+m}$. This particular choice of γ is an upper bound to the learning rate which minimizes the RBM energy variance over all states (see the Supplementary Note 2 for the proof of this statement).

We find that the mode-assisted training method is not very sensitive to the parameters chosen. In fact, as long as the mode samples are incorporated after the joint and marginal mode equilibration, the training is stabilized and the learned distribution will tend to uniformity (see also the Supplementary Note 2). This result reinforces the intuitive notion that the pushes on the mode provide a stabilizing quality to the training over CD (or any other MCMC approach), which can otherwise diverge when mixing rates grow too large at later times during training.

To realize this method in practice, one must supplement standard gradient updates with updates constructed from samples of the ground state of the RBM. Finding this ground state is equivalent to a quadratic unconstrained binary optimization (QUBO) problem, known to be nondeterministic polynomial time (NP)-hard[18]. Therefore, although we can compute the ground state of RBMs exactly for small datasets, for efficient mode sampling in realistically sized cases, we employ a memcomputing solver that compares favorably to other state-of-the-art optimizers in efficiently sampling the ground states of similar non-convex landscapes[19,20]. Utilizing this approach and accounting for CPU time, we find that mode-assisted training leads to models with lower KL divergences than CD. The details of our implementation, including computational complexity and energy comparisons with MCMC, can be found in Supplementary Note 3. However, in principle, one could use other optimizers for mode-assisted training.

Importance of representability. Note that since mode-assisted training is driven to distributions of a particular form, instead of local minima as in the case of CD or other gradient approaches, the representability of the RBM becomes important. The ability of a RBM to represent a given data distribution is given by the amount of hidden nodes, where one is guaranteed universal representability with $n_d + 1$ hidden nodes[3]. In other words, one more hidden node than the number of visible configurations with non-zero probability is sufficient (but perhaps not necessary) for general representability. In practice, this bound is found to be very conservative and typically much fewer nodes are needed for a reasonable solution.

Representability can become an issue in mode-assisted training when the parameter space of the RBM does not include the uniform distribution over the support (or a reasonable approximation). Since mode-assisted training is generally in a non-gradient direction, this means that it may settle to a worse solution than a local optimum obtainable by CD. This is a signal that more hidden nodes are required for an optimal solution.

Since most natural datasets live on a very small dimensional manifold of the total visible phase space, $|n_d|/|\Omega| \ll 1$, the amount of hidden nodes required typically scales polynomially with the size of the problem, versus the exponential scaling of the visible phase space. This makes representability not an insurmountable problem for mode-assisted training, even in full size problems. In the Supplementary Note 2, we demonstrate that regardless of representability, a mode-assisted weight update reduces the variance of the energies across the RBM states. To this end, the examples of Figs. 2 and 3 show that mode-assisted training does not necessarily fail if the number of hidden nodes is less than that needed to guarantee representability.

Examples of mode-assisted training on datasets. As examples of our method, we have computed the log-likelihoods achieved with mode-assisted training across two synthetic and one realistic (MNIST) datasets, and compared the results against the best

achieved log-likelihoods with CD-1, PCD-1 and PT on standard RBMs, E-RBMs, and C-RBMs[16]. For the small synthetic datasets we could also compute the exact log-likelihoods, thus providing an even stronger comparison. An implementation of the mode-assisted training algorithm on these synthetic data sets is available in the Supplementary Code that accompanies this paper. For the larger MNIST case, mode sampling was done via simulation of a digital memcomputing machine based on ref. [21]. The specific details of our implementation can be found in Supplementary Note 3.

We plot an example of training progress in a moderately large synthetic problem in Fig. 3. Reported is the KL divergence (which differs from the log-likelihood by a constant factor independent of the RBM parameters[2]) of a slightly bigger 14×10 RBM as a function of number of parameter updates on a $L = 14$, $B = 7$ shifting bar set, for both CD-1 and mode-assisted training. We consider two learning rate schedules, constant ($\epsilon^{\mathrm{CD}} = 0.05$) and exponential decay ($\epsilon^{\mathrm{CD}}(n) = e^{-cn}$, $c = 4$, $n \in [0, 1]$, the fraction of completed training iterations). Additionally, every time a mode sample is taken, CD is allowed to run with $k = 720$, a number scaled to the equivalent computational cost of taking a mode sample. Thus, the x-axes in Fig. 3 could be interpreted as wall time. The details of the computational comparison between a mode sample using memcomputing and iterations of CD are discussed in greater detail in Supplementary Note 3. In both cases, even when computational cost is factored in, mode-assisted training leads to better solutions and proceeds in a much more stable way across runs (lower KL variance at convergence). Importantly, mode-assisted training never diverges while CD often times does. Following our intuition about mode-assisted training established in the "Mode Correspondence of Joint and Marginal Distributions" section, using larger learning rates in the CD-dominated phase accelerates the convergence of mode-assisted training.

It is known that using CD to train RBMs can result in poor models for the data distribution[22], for which PCD and PT are recommended. We note that for the mode-assisted training employed in this paper, CD-1 was used as the gradient approximation (except in the case for MNIST where PCD-1 was used). Impressively, in all cases tested, the mode samples were able to stabilize the CD algorithm sufficiently to overcome the other, more involved approximations (PT) and model enhancements (centering).

In addition, it is clear that mode-assisted training exhibits several desirable properties over CD (or other gradient approaches). Most significantly, it seems to perform better with larger learning rates during the gradient dominated phase, and smaller learning rates when using mode samples. CD and other gradient methods generally perform better with smaller learning rates, as their approximation to the exact gradient gets better. Irrespective, even in this regime, mode-assisted training eventually drives the system to the uniform solution compared with the local optimum of CD. The main advantage is that with mode-assisted training, one can (and often should) use larger learning rates, resulting in fewer required iterations.

For further comparison, we report in Table 1 results for the shifting and inverted bar, bars and stripes, and MNIST datasets obtained with mode-assisted training and those reported in past work[16]. The results show mode-assisted training with a standard RBM always converges to models with log-likelihoods higher than E-RBMs, and C-RBMs trained with CD-1, PCD-1, or PT. Furthermore, the mode-assisted training log-likelihood increases with an increasing number of hidden nodes (better representability). Empirically, we also find the incredible result that with sufficient representability and the proper learning rate, mode-assisted training can find solutions arbitrarily close to the exact distribution.

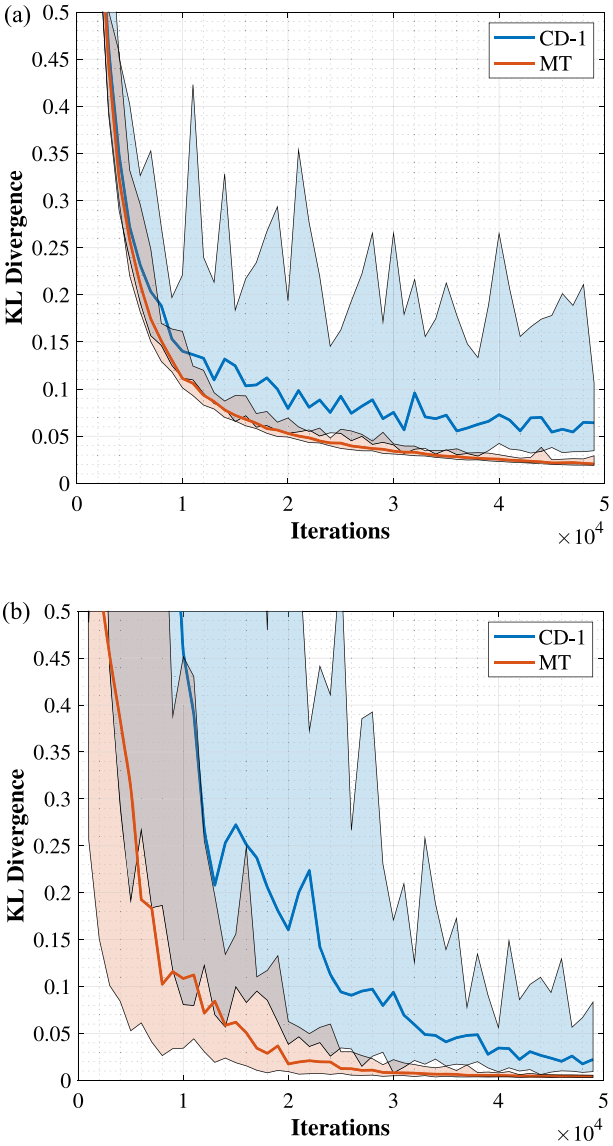

Fig. 3 Mode-Assisted training and contrastive divergence performance on the shifting bar dataset. Shown are the Kullback-Liebler (KL) divergences achieved on the binary shifting bar dataset across 25 randomly initialized 14×10 restricted Boltzmann machines for both contrastive divergence (CD) with $k = 1$ iteration and mode-assisted training (MT). In addition, every time a mode sample is taken, CD is allowed to run with $k = 720$ iterations, a number scaled to the equivalent computational cost of taking a mode sample. Thus, the x-axis can also be read as wall time. The bold line represents the median KL divergence across the runs, and the max/min KL divergences achieved at that training iteration define the shaded area. (**a**) is with a small CD learning rate, $\epsilon^{\mathrm{CD}} = 0.05$. (**b**) is with an exponentially decaying $\epsilon^{\mathrm{CD}}(n) = e^{-cn}$ with decay constant $c = 4$ and $n \in [0, 1]$ being the fraction of completed training iterations.

Discussion

In this paper we have introduced an unsupervised training method that stabilizes gradient based approaches by utilizing samples of the ground state of the RBM, and is empirically seen to get as close as desired to a given uniform data distribution. It relies on the realization that as training proceeds, the RBM becomes increasingly frustrated, leading to the modes of the visible layer distribution and joint model distribution becoming effectively equal. As a consequence, by using the mode (or ground state) of the RBM during training, our approach is able to

"flatten" all modes in the model distribution that do not correspond to modes in the data distribution, reaching a steady state only when all modes are of equal magnitude. In this sense, the ground state of the RBM can be thought of as 'supervising' the gradient approximation during training, preventing any pathological evolution of the model distribution.

Our results are valid if the representability of the RBM is enough to include good approximations of the data distribution. Once the representability is sufficient, a properly annealed learning-rate schedule will take the KL divergence as low as desired. Increasing the number of hidden nodes increases the non-convexity of the KL-divergence landscape, easily trapping standard algorithms in sub-optimal states. In practice, after some point, increasing the number of hidden nodes will not decrease the KL divergence that a pre-training procedure actually converges to, as the trade-off between effective gradient update and representation quality is reached. We here claim that this point of tradeoff for our mode-assisted procedure is reached at far greater number of nodes than standard procedures, thus allowing us to find representations with far smaller KL-divergence. The mode-assisted training we suggest then provides an extremely powerful tool for unsupervised learning, which (i) prevents a divergence of the model, (ii) promotes a more stable learning, and (iii) for data distributions uniform across some support, can lead to solutions with arbitrary high quality.

Superficially, this method resembles 'mode hopping' MCMC proposed in recent literature[23,24], where local maxima are either found with some optimization method or assumed to be known before hand (via a dataset). However, a crucial difference between our mode-assisted training for RBMs and mode hopping algorithms is that we do not use the modal configuration to initiate a new MCMC update to improve the mixing rate. Instead, the mode itself is used to inform the weight updates directly. The difference is substantial. In fact, since higher energy states are exponentially suppressed, exposing the Markov chain to the mode will most likely get it stuck there, which requires ad hoc constructions to recover detailed balance. Our mode-training method does not suffer from these drawbacks and is thus a more computationally efficient way to utilize the mode to train RBMs. Furthermore, we show that under a sufficiently large learning rate, sampling the global mode alone is capable of exploring efficiently a multi-modal energy landscape.

To scale our approach, one would need an efficient way to sample low-energy configurations of the RBM, a difficult optimization problem on its own. This is equivalent to a weighted MAX-SAT problem, for which there are several industrial-scale solvers available. Also, the recent successes of memcomputing on these kind of energy landscapes in large cases (million of variables) are fodder for optimism[19,20].

Finally, fitting general discrete distributions (with modes of different height) with this technique seems also within reach. In this respect, we can point to our results on the bars and stripes dataset (a non-uniform $q(\mathbf{v})$) for inspiration. We have found the best log-likelihood on that set with mode-assisted training with a lower frequency of the mode sampling, $P_{\max} = 0.1 \rightarrow 0.05$, compared with the shifting bar (a uniform set). This suggests that a general update, which properly weighs the mode sample in combination with the dataset samples, may extend this technique to general non-uniform probabilities, with the weight analogous to a tunable demand for uniformity.

Our method is useful from a number of perspectives. First, from the unsupervised learning point of view, it opens the door to the training of RBMs with unprecedented accuracy in a novel, non-gradient approach. Second, many unsupervised models are used as 'feature learners' in a downstream supervised training task (e.g., classification), where the unsupervised learning is referred to as pre-training. We suspect that starting backpropagation from an initial condition obtained through mode-assisted training would be highly advantageous. Third, the mode-assisted training we suggest can be done on models with any kind of pairwise connectivity, which include deep, convolutional, and fully connected Boltzmann machines. We leave the analysis of these types of networks for future work.

Table 1 Log-likelihood comparisons across data sets.

	S. Bar	Inv. S. Bar	Bars and stripes	MNIST
CD-1	−20.42	−20.73	−61.08	−152.42
PCD-1	−21.71	−21.64	−57.01	−140.43
PT	−20.57	−20.57	−51.99	−142.00
MT	**−19.85**	**−19.86**	**−50.79** (**−41.82**)	**−136.42**
Exact	−19.77	−19.77	−41.59	-

We report the highest achieved log-likelihoods over 50,000 gradient updates on a 9 × 4 restricted Boltzmann machine (RBM) across various RBM types (standard, enhanced-RBM, centered-RBM) and training techniques (contrastive divergence (CD), persistent-CD (PCD), parallel tempering (PT)) as reported in previous work[16] compared with mode-assisted training (MT) on a standard RBM. In the table, rows correspond to different training techniques and columns are different data sets. For each technique, the best achieved log-likelihood score across 25 runs is reported. In parenthesis are results for a 9 × 9 RBM. For these small datasets we can also compare with the exact result. For the MNIST dataset, the trained networks trained had 16 hidden nodes and PCD-1 was used as the gradient update, and average log-likelihood is reported.
The highest log-likelihood achieved on a given data set is shown in bold.

Methods

For synthetic data, we use the commonly employed binary shifting bar and bars and stripes datasets[25]. The former is defined by two parameters: the total length of the vector, L, and the amount of consecutive elements (with periodic boundary conditions), $B < L$, set to one, with the rest set to zero. This results in L unique elements in the dataset with uniform probability, giving a maximum likelihood of $L\log(1/L)$. The inverted shifting bar set is obtained by swapping ones and zeros. The bars and stripes dataset is constructed by setting each row of a $D \times D$ binary pattern to one with probability 1/2, and then rotating the resulting pattern 90° with probability 1/2. This produces 2^{D+1} elements, with the all-zero and all-one patterns being twice as likely as the others.

For a direct comparison to previous work, we followed the same training setup[16]. For the data in Table 1, a 9 × 4 RBM was tested on a shifting bar dataset with $L = 9$, $B = 1$ and a $D = 3$ bars and stripes dataset. Both synthetic sets were trained for 50,000 parameter updates, with no mini-batching, and a constant $\epsilon^{\mathrm{CD}} = 0.2$. For the MNIST dataset, a 784 × 16 sized model was trained for 100 epochs, with batch sizes of 100. The mode samples in both cases are slowly incorporated into training in a probabilistic way following Eq. (7), initially with $P_{\mathrm{mode}} = 0$ and driven to $P_{\max} = 0.1$ for the shifting bar and MNIST datasets, and $P_{\max} = 0.05$ for the bars and stripes dataset. In both cases, we chose $\alpha = 20/N$ and $\beta = -6$, where N is the total number of parameter updates.

Data availability

The MNIST data set used in this paper is publicly available (http://yann.lecun.com/exdb/mnist/). All other data that support the plots within this paper are available from the corresponding author upon request.

Code availability

The Matlab code for the training examples used in this work is provided in the Supplementary Code file which accompanies the paper.

Received: 9 April 2020; Accepted: 14 May 2020;
Published online: 05 June 2020

References

1. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. A learning algorithm for Boltzmann machines. *Cogn. Sci.* **9**, 147–169 (1985).
2. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. *Deep Learning* 1 (MIT Press, Cambridge, 2016).

3. LeRoux, N. & Bengio, Y. Representational power of restricted Boltzmann machines and deep belief networks. *Neural Comput.* **20**, 1631–1649 (2008).
4. Bengio, Y. et al. Learning deep architectures for ai. *Found. Trends® Mach. Learn.* **2**, 1–127 (2009).
5. Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial. *Neural Netw. Sci.* **355**, 602–606 (2017).
6. Gao, X. & Duan, L.-M. Efficient representation of quantum many-body states with deep neural networks. *Nat. Commun.* **8**, 662 (2017).
7. Szegedy, C. et al. Intriguing properties of neural networks. Preprint at https://arxiv.org/abs/1312.6199 (2013).
8. Erhan, D. et al. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* **11**, 625–660 (2010).
9. Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **14**, 1771–1800 (2002).
10. Fischer, A. & Igel, C. An introduction to restricted Boltzmann machines. In Alvarez, L., Mejail, M., Gomez, L. & Jacobo, J. (eds) *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 14–36 (Springer, Berlin, Heidelberg, 2012).
11. Long, P. M. & Servedio, R. A. Restricted Boltzmann machines are hard to approximately evaluate or simulate. *ICML* 703–710 (2010).
12. Manukian, H., Traversa, F. L. & Di Ventra, M. Accelerating deep learning with memcomputing. *Neural Netw.* **110**, 1–7 (2019).
13. Di Ventra, M. & Pershin, Y. V. The parallel approach. *Nat. Phys.* **9**, 200–202 (2013).
14. Traversa, F. L. & DiVentra, M. Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines. Chaos: an Interdisciplinary. *J. Nonlinear Sci.* **27**, 023107 (2017).
15. DiVentra, M. & Traversa, F. L. Memcomputing: Leveraging memory and physics to compute efficiently. *J. Appl. Phys.* **123**, 180901 (2018).
16. Melchior, J., Fischer, A. & Wiskott, L. How to center deep boltzmann machines. *J. Mach. Learn. Res.* **17**, 3387–3447 (2016).
17. Pei, Y. R., Manukian, H. & Di Ventra, M. Generating weighted max-2-sat instances of tunable difficulty with frustrated loops. Preprint at https://arxiv.org/abs/1905.05334 (2019).
18. Arora, S. & Barak, B. *Computational Complexity: A Modern Approach*. (Cambridge University Press, New York, 2009).
19. Traversa, F. L., Cicotti, P., Sheldon, F. & Di Ventra, M. Evidence of exponential speed-up in the solution of hard optimization problems. *Complexity* **2018**, 798285 (2018).
20. Sheldon, F., Traversa, F. L. & Di Ventra, M. Taming a non-convex landscape with dynamical long-range order: memcomputing the ising spin-glass. Preprint at https://arxiv.org/abs/1810.03712 (2018).
21. Bearden, S. R. B., Sheldon, F. & Di Ventra, M. Critical branching processes in digital memcomputing machines. *EPL (Europhys. Lett.)* **127**, 30005 (2019).
22. Hinton, G. A practical guide to training restricted Boltzmann machines. *Momentum* **9**, 926 (2010).
23. Sminchisescu, C. & Welling, M. Generalized darting monte carlo. In *Artificial Intelligence and Statistics*, 516–523 (2007).
24. Lan, S., Streets, J. & Shahbaba, B. Wormhole Hamiltonian Monte Carlo. *Proc Conf. AAAI Artif. Intell.* 1953–1959 (2014).
25. MacKay, D. J. *Information Theory, Inference and Learning Algorithms*. (Cambridge University Press, New York, 2003).

Acknowledgements

Work supported by DARPA under grant No. HR00111990069. H.M. acknowledges support from a DoD-SMART fellowship. M.D. and Y.R.P. acknowledge partial support from the Center for Memory and Recording Research at the University of California, San Diego. All memcomputing simulations reported in this paper have been done on a single core of an AMD EPYC server.

Author contributions

M.D. has supervised and suggested the project. H.M. and M.D. conceived the idea. Y.R.P. established the theoretical framework for analyzing the algorithm. H.M. has done all the simulations reported. S.R.B.B. has designed the digital memcomputing machine employed in this work. All authors have discussed the results and contributed to the writing of the paper.

Competing interests

M.D. is the co-founder of MemComputing, Inc. (https://memcpu.com/) that is attempting to commercialize the memcomputing technology. All other authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s42005-020-0373-8.

Correspondence and requests for materials should be addressed to H.M., Y.R.P. or M.D.V.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.