scientific reports

Check for updates

OPEN

Mode-assisted joint training of deep Boltzmann machines

Haik Manukian✉ & Massimiliano Di Ventra

The deep extension of the restricted Boltzmann machine (RBM), known as the deep Boltzmann machine (DBM), is an expressive family of machine learning models which can serve as compact representations of complex probability distributions. However, *jointly* training DBMs in the *unsupervised* setting has proven to be a formidable task. A recent technique we have proposed, called mode-assisted training, has shown great success in improving the unsupervised training of RBMs. Here, we show that the performance gains of the mode-assisted training are even more dramatic for DBMs. In fact, DBMs jointly trained with the mode-assisted algorithm can represent the same data set with *orders of magnitude* lower number of total parameters compared to state-of-the-art training procedures and even with respect to RBMs, provided a *fan-in* network topology is also introduced. This substantial saving in number of parameters makes this training method very appealing also for hardware implementations.

One of the most influential models in Artificial Intelligence (AI) and deep learning in particular is the Boltzmann machine (BM). It was constructed[1,2] as a powerful stochastic generalization of Hopfield networks[3] that possess a simple expression for their log-likelihood gradient. However, they remained impractical to train due to their reliance on high-dimensional sampling to calculate that gradient. A relatively efficient learning algorithm, called contrastive divergence (CD), was discovered for BMs with a simplified topology called restricted Boltzmann machines (RBMs)[4], which have since gone on to see success in various domains[5]. However, the extension of RBMs to deep Boltzmann machines (DBMs) has been difficult[6], and as such, DBMs are now mostly overshadowed by their deep feedforward cousins[7] in generative applications.

This is not due to DBM's lack of ability, but rather the absence of effective means to train these models. There remain quite a few reasons to search for better learning algorithms for DBMs, as they are a versatile computational medium. A principle use is as compact generative models for complex probability distributions in unsupervised settings, considered a critical component of the forthcoming "third-wave" of AI[8]. Trained DBMs can also serve as an informed prior for feedforward networks, leading to better generalization in supervised tasks[9]. In the physical sciences, DBMs serve as powerful variational representations of many body wavefunctions, more efficiently[10] than RBMs[11,12], and have potential applications in condensed matter physics and quantum computing[13].

Various attempts have been made to climb the summit of DBM training[14–18]. Most approaches rely on pre-training by breaking up layers into RBMs and training them sequentially, after which the DBM is fine-tuned jointly. Correlations between layers are ignored during pre-training, minimizing the potential advantages of the deep architecture which can be disruptive to joint training[18]. Recently, the authors introduced mode-assisted training[19,20], which combines CD with samples of the model distribution mode. This stabilizes training, allows the learning of very accurate densities, and strikes a better tradeoff between accuracy and computational cost compared to CD. As the method is agnostic to the connectivity of the network, we can apply mode-assistance to DBMs with the hope of capturing the model capacity missed by other approaches.

In this work, we find that the benefits of mode-assisted training are even more dramatic in the case of DBMs. In fact, it produces more accurate models without requiring pre-training while also utilizing *orders of magnitude* less parameters, compared to pre-trained[14] or centered DBMs[18]. The role of the network topology is also discussed, where we discover that DBMs are easier to train if the size of the layers decreases with depth. We evaluate the density modeling performance of mode-assisted DBMs by computing exact log-likelihoods achieved on small data sets and approximating the likelihood on the MNIST data set. The approach we propose can be extended to other types of neural networks, and is relevant also for the hardware implementation of these models, where a much smaller number of parameters directly translates into components and energy savings.

To see how mode-assisted training can be extended to deep architectures, we give a quick overview of DBMs and the basic approach to their training. DBMs are undirected weighted graphs that differentiate between n_v visible nodes, and ℓ layers of n_ℓ latent, or 'hidden', nodes, not directly constrained by the data[14]. We assume that

Department of Physics, University of California, San Diego, La Jolla, CA 92093, USA. ✉email: hmanukia@ucsd.edu

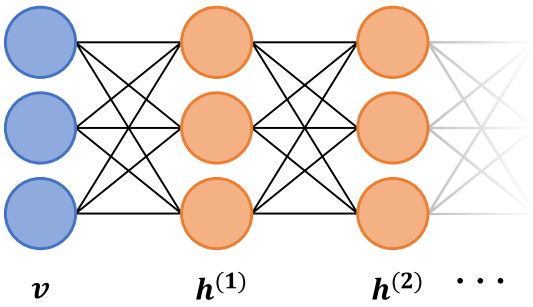

Figure 1. Schematic of a deep Boltzmann Machine with a visible layer, v, and hidden layers, $h^{(j)}$, with $j = 1, 2, \ldots$. Connections between nodes are symmetric and undirected, in contrast to typical directed feedforward networks.

$\ell > 1$ as $\ell = 1$ recovers the RBM, and like an RBM, there are no connections within a layer. A graphical example of a two layer DBM is shown in Fig. 1. Each state of the machine corresponds to an energy of the form

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(\ell)}) = -\mathbf{a}^T\mathbf{v} - \mathbf{v}^T\mathbf{W}^{(1)}\mathbf{h}^{(1)} - \sum_{i=1}^{\ell}\mathbf{b}^{(i)T}\mathbf{h}^{(i)} - \sum_{i=2}^{\ell}\mathbf{h}^{(i-1)T}\mathbf{W}^{(i-1)}\mathbf{h}^{(i)}, \tag{1}$$

where the biases $\mathbf{a} \in \mathbb{R}^{n_0}$, $\mathbf{b}^{\ell} \in \mathbb{R}^{n_\ell}$, and weights $\mathbf{W}^{\ell} \in \mathbb{R}^{n_{\ell-1}\times n_\ell}$ are the learnable parameters. The energy function in Eq. (1) induces a Boltzmann-Gibbs distribution over states,

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

where $\mathbf{x} = (\mathbf{v}, \mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(\ell)})$. The partition function, $\mathcal{Z} = \sum_{\{\mathbf{x}\}} e^{-E(\mathbf{x})}$, involves the sum of an exponentially growing number of states, making the exact computation of its value infeasible for most data sets.

During learning, a DBM is tasked to match its marginal distribution over the visible layer, $p(\mathbf{v}) = \sum_{\{\mathbf{h}\}} p(\mathbf{v}, \mathbf{h})$, to an unknown data distribution, $q(\mathbf{v})$, represented by a data set, $\mathcal{D}$. Training a DBM 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,

$$\mathrm{KL}(q||p) = \sum_{\{\mathbf{v}\}} q(\mathbf{v}) \log \frac{q(\mathbf{v})}{p(\mathbf{v})}, \tag{3}$$

or, equivalently, maximizing the log-likelihood of the dataset, $\mathrm{LL}(p) = \sum_{\mathbf{v}\in\mathcal{D}} \log p(\mathbf{v})$. The optimization of the non-linear, and typically high dimensional Eq. (3) (or log-likelihood), is often done via stochastic gradient descent with respect to the DBM parameters, which leads to weight updates of the form[21],

$$\Delta w_{ij} \propto \langle x_i x_j \rangle_D - \langle x_i x_j \rangle_M. \tag{4}$$

For every gradient update in Eq. (4), nodal statistics must be computed under two different distributions. The first one on the RHS of Eq. (4) is called the "data term", and is an expectation over the data induced distribution, $q(\mathbf{v})p(\mathbf{h}|\mathbf{v})$, with the network's visible layer fixed to the data. The second term on the RHS of Eq. (4) is called the "model term" which is an expectation over the entire model distribution in Eq. (2). In the case of RBMs, the data term can be sampled from exactly, but the model term must be approximated. With DBMs, the data term must also be approximated, most popularly with an iterative mean field procedure (see Methods).

In both cases, model statistics are collected via a Markov Chain Monte Carlo (MCMC) procedure dubbed 'contrastive divergence' (CD)[4]. CD-k is a form of Gibbs sampling that initializes chains of length k from elements of the dataset. Trouble arises when the model distribution contains 'spurious' modes where the data distribution has negligible probability. In these cases, ergodicity breaks down, and mixing times become prohibitively long, frequently resulting in CD becoming biased enough to cause training to diverge[22]. Training a DBM *jointly* with CD has proven to be a formidable task. Even a two-layer DBM on MNIST has not seen success without some kind of modification[6,14,18,23].

Here, instead, we use mode-assisted training in the *joint* and *unsupervised* learning of DBMs. The essence of mode-assisted training is the replacement of gradient updates in Eq. (4) with ones of the form,

$$\Delta w_{ij} \propto \begin{cases} [x_i x_j]_D - [x_i x_j]_M, & \text{for } u \le P_{\text{mode}} \\ \langle x_i x_j \rangle_D - \langle x_i x_j \rangle_M, & \text{else} \end{cases} \tag{5}$$

Here, u is a random variable sampled from the uniform distribution over the unit interval, $U[0, 1]$. The notation $[f(\mathbf{x})]_q$ represents $f(\mathbf{x}_{\text{mode}})$, evaluated at $\mathbf{x}_{\text{mode}}$, the mode of some distribution, $q(\mathbf{x})$. One may employ any

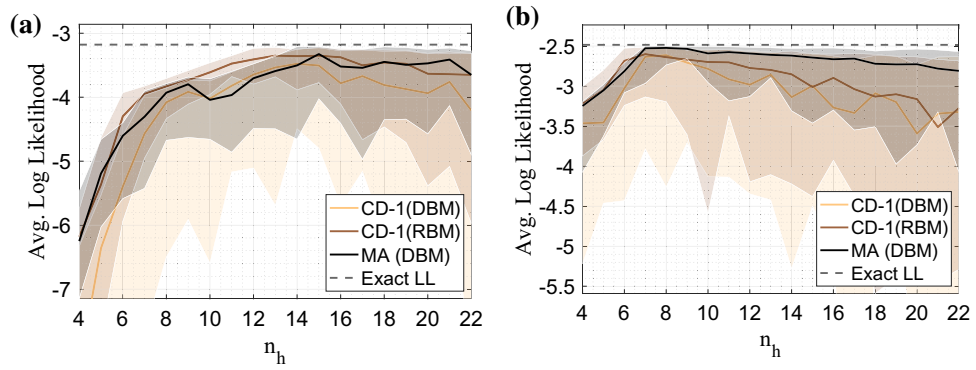

Figure 2. Average converged log-likelihood performance (lower bound) between RBMs trained with CD-1, mode-assisted DBMs (MA), and unassisted DBMs with CD-1. The DBMs have two hidden layers. The networks were trained on the shifting bar data set with $n_v = 24$ (left plot) and $n_v = 12$ (right plot) for 200,000 and 100,000 gradient updates respectively, following a linearly decaying learning rate schedule from $\epsilon = 1 \rightarrow 0.001$. For the DBMs the hidden layer ratio was fixed at $\alpha = n_{h^{(2)}}/n_{h^{(1)}} = 0.2$. Performance is shown as a function of total number of hidden nodes, $n_h = n_{h^{(1)}} + n_{h^{(2)}}$. The solid lines are the median obtained across an ensemble of 50 networks, and the shaded regions enclose the 95th and 5th percentiles.

optimization solver to sample the mode of the above distributions. Due to its proven efficiency, here we employ a memcomputing one as reported in our previous work[20].

The weight updates driven by the mode are incorporated in a probabilistic way, with the probability of a mode driven update, P_{mode}, following a sigmoid, starting low in the initial phases of the training and reaching a maximal value at the end of training:

$$P_{\text{mode}}(n) = P_{\text{max}}\sigma(\alpha n + \beta). \tag{6}$$

Here, n is the current epoch, and α, β, P_{max} control the shape of the sigmoid. Throughout the work we set, $\alpha = 20/N$ (N is the total number of epochs), $\beta = -6$ and $P_{\text{max}} = 0.1$. This schedule was introduced in Ref.[20], based on the empirical observation that the mode samples work best when the support has been 'discovered' by CD.

The mode-assisted update can be thought of as a saddle-point approximation of an expectation[24]. This technique, also known as Laplace's method, is commonly employed to approximate integrals (expectations) of the form,

$$\langle f(\mathbf{x})\rangle = \frac{\int e^{-E(\mathbf{x})} f(\mathbf{x}) d\mathbf{x}}{\int e^{-E(\mathbf{x})} d\mathbf{x}} \approx f(\mathbf{x}_{\text{mode}}). \tag{7}$$

The key insight of mode assisted training is *not* to explicitly improve the mixing time of CD, but rather directly prevent spurious modes from appearing in the model distribution. This allows the inexact but efficient approximation of CD-k to minimize the KL divergence (or maximize the log-likelihood) without diverging. Searching for the mode with a highly specialized optimizer results in a better trade-off between log-likelihood performance and computation time.

Results

To illuminate the effectiveness of mode assistance on DBMs, we first evaluate performance on two differently sized synthetic shifting bar data sets. In Fig. 2, the mode-assisted algorithm on a two-layer DBM is compared to two baselines, CD on the same DBM and to CD on an RBM with the same number of hidden nodes. The converged log-likelihoods are reported, and results are shown as a function of the total number of hidden nodes. Overall, mode-assisted training almost always converges to more accurate densities than CD alone on a DBM, and in most cases also does better than the corresponding RBM with CD. Mode assistance is also seen to prevent the divergence sometimes seen with Gibbs sampling, as well as reducing the variance of the converged models, resulting in smaller error bars. Past a certain point, all methods expectedly incur a loss in performance as the number of hidden nodes increases for a fixed number of training iterations. However, mode-assisted training suffers the least in this regard.

Although the DBMs in Fig. 2 possess the same total number of hidden nodes as the RBMs, they contain fewer total parameters. This means that a better trained DBM takes advantage of abstract features composition afforded by the depth of the network, mirroring similar gains found in deep feedforward neural networks compared to single layer perceptrons[5].

Note, however, that this parameter efficiency in DBMs is *not* present when training jointly with CD. In fact, they perform systematically worse than an equivalently sized RBM in terms of log-likelihood. Mode assisted DBMs on the other hand perform as well as RBMs or better, all the while maintaining parameter efficiency.

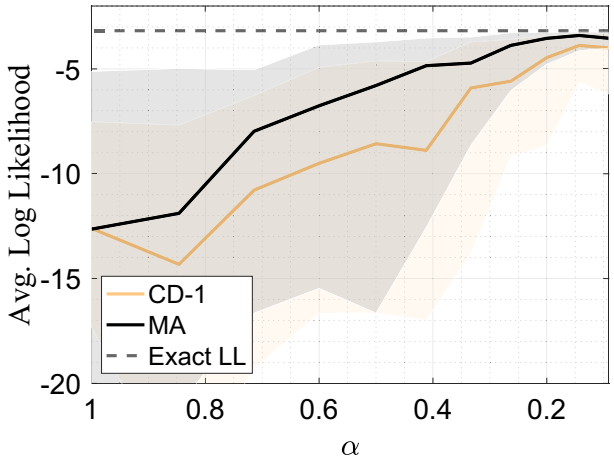

Figure 3. Average Log-Likelihood achieved on an $n_v = 24$ dimensional shifting bar data set as a function of DBM shape, α. The total number of hidden nodes are kept fixed at $n_{h^{(1)}} + n_{h^{(2)}} = 22$. An ensemble of 50 DBMs were trained with CD-1 and MA, where the solid line shows the median average log-likelihood achieved after 10^5 gradient iterations with a linearly decaying learning rate $\epsilon = 1 \to 0.01$, and shaded regions delineate the 95th and 5th percentiles.

To quantify this efficiency gain, let us consider the case of a DBM with two hidden layers. If the total number of hidden nodes is fixed, $n_h = n_{h^{(1)}} + n_{h^{(2)}}$, then a measure of parameter efficiency compared to an RBM with the same number of nodes is captured by the parameter $\alpha = n_{h^{(2)}}/n_{h^{(1)}}$. The total number of parameters (weights and biases) in an RBM with n_v visible nodes and n_h hidden nodes is,

$$n_{\text{RBM}} = n_v n_h + n_v + n_h.$$

A DBM with the same total number of hidden nodes would have the following number of parameters

$$n_{\text{DBM}} = n_v n_{h^{(1)}} + n_{h^{(1)}} n_{h^{(2)}} + n_v + n_{h^{(1)}} + n_{h^{(2)}}.$$

Considering typical training scenarios of a large visible layer, $n_v \gg n_h \gg 1$, the fraction of total parameters used in the DBM compared to the RBM (the *parameter efficiency*) can be simplified as,

$$e = \frac{n_{\text{DBM}}}{n_{\text{RBM}}} \approx \frac{1}{1+\alpha}. \tag{8}$$

The smaller e, the fewer parameters a DBM possesses with respect to an RBM with the same number of nodes. We then see that a naïve interpretation of Eq. (8) would suggest that the DBM parameter efficiency becomes more significant with increasing α, meaning the resulting DBM has more neurons in the deeper layers (fans out). However, this assumes equal log-likelihood (performance) of the two models, which is not obvious. In fact, for DBMs we expect that a *redistribution* of the relative number of nodes in the different hidden layers plays a significant role in their performance. This tradeoff between parameter efficiency and performance merits further investigation.

This is shown in Fig. 3, where DBMs trained with and without mode assistance are compared as a function of α. For all cases, the mode-assisted algorithm performs better than its unassisted counterpart, which is nothing other than the confirmation of the results of Fig. 2. Importantly, however, we also observe that the log-likelihood performance *increases* as the network becomes less parameter efficient compared to an RBM ($\alpha \to 0$), namely when it has a *fan-in* topology. We find that a balance is reached around $\alpha \sim 0.15$, namely the second hidden layer has only about 15% of nodes than the first hidden layer. The message here is that for a *fixed* number of hidden nodes, it is best to organize the network as a fan-in DBM rather than an RBM, if one has a method to train the DBM close to capacity in the first place.

Finally, to show the substantial improvements provided by the mode-assisted training of DBMs, we compare it to the training of centered DBMs[18] and pre-trained DBMs[14] using CD on the MNIST data set. In Fig. 4, we report similar trends we observed in the smaller data sets, but scaled to more dramatic results. As the number of hidden nodes increases, small mode-assisted DBMs surpass the performance of significantly larger standard DBMs, centered DBMs and eclipse the performance of much larger pre-trained DBMs. In fact, with a DBM of only 120×18 hidden nodes (and no pre-training), trained with our mode-assisted approach, we reach the same log-likelihood of a DBM with 500×1000 hidden nodes *and* pre-training—a parameter savings of two orders of magnitude.

For a fair comparison, the total training iterations were set to 100 epochs in each case, and the log-likelihood results with our approach are shown as a function of total number of hidden nodes at a fixed ratio, $\alpha = 0.15$. The partition function was approximated using Annealed Importance Sampling (AIS) (see Methods), identical

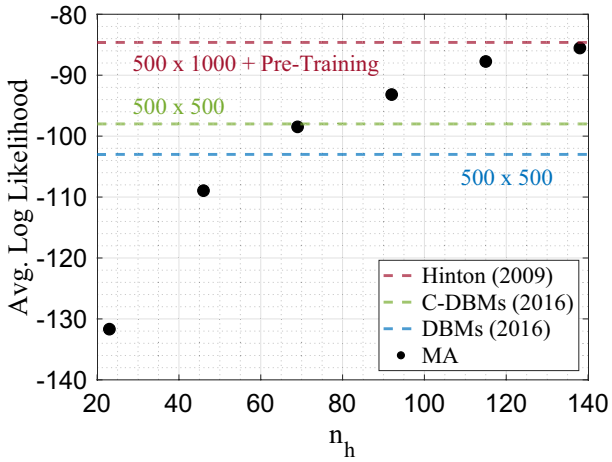

Figure 4. Average log-likelihood on the MNIST data set achieved after 100 epochs with mode-assisted training (MA) compared to the recent best achieved results on DBMs using CD as well as pre-training. DBMs trained with MA used PCD-1 for all non-modal samples, matching Ref.[18]. Network topology was fixed to $\alpha = 0.15$ with an increasing number of hidden nodes, n_h. The learning rate was chosen to follow a linear decay, $\epsilon = 0.05 \rightarrow 0.0005$ and no pre-training was employed. The best log-likelihood was reported out of 10 randomly initialized runs.

to the other referenced results[18]. It's worth noting that AIS tends to *under*-approximate the partition function, leading to an over-estimate of the log-likelihood. This effect is exaggerated in larger models. Even with this negative impact, a mode-assisted DBM with a total of 138 nodes without pre-training achieves about the same performance as one with 1500 that was pre-trained.

Discussion

We conclude by providing an intuitive understanding of the dramatic performance improvement of the mode-assisted training over Gibbs sampling (CD). The standard method of pretraining DBMs initializes weights at a greedy starting point but critically ignores correlations within the system. The end result is a dramatic loss of parameter efficiency compared to mode-assisted joint training of DBMs. Even training methods like centering, with more advanced sampling techniques like parallel tempering[25], fail to train the DBM near its capacity. The issue is the reliance on Gibbs-like sampling to explore the states of the DBM, and using only these statistics to compute the likelihood gradient. The breakdown occurs when Gibbs chains fail to explore adequately the states of the system.

Because of its random-walk nature, the local dynamics of Gibbs sampling suffers long auto-correlation times when equally likely states are separated by an extensive number of variables flips. As a consequence of long mixing times between such states, statistics become heavily biased. In the initial phase of DBM training, when weights are small and randomly distributed, these issues are minimized. Interactions between nodes in this 'high temperature' regime are weak, and are well approximated by a mean-field theory[26]. In this phase, CD works well without much bias, and is the reason mode-assistance is really not necessary early on in the training.

However, as training continues, the DBM is effectively 'cooled' as weights learn from the data and grow larger in magnitude. Correlations between nodes become significant, and can no longer be ignored or averaged over: mean-field theory is inadequate to describe this phase. Gibbs sampling can then dramatically fail in these conditions, that is why mode-assistance is primarily applied in this phase. During mode updates, information is propagated from the ground state, the most 'collective' or 'coordinated' state of the DBM, to all the weights. This prevents probability mass density from accumulating far away from the current state of the chain, keeping Gibbs sampling in its effective regime.

Aside from mode-assistance, we have discovered that the topology of a DBM plays a major role in its performance as well as its parameter efficiency. Too many hidden nodes act as a noise source during training, slowing down learning. Too few hidden nodes reduce the capacity of the network. For the network sizes considered, a 'sweet spot' was found where these two effects are in balance. At this stage, it is not clear if this is particular to DBMs, or there are deeper reasons why this is the case, and further work in this direction would be interesting.

Finally, mode training relies on an efficient search for the mode of a DBM, which is an NP-hard computational problem on its own. On this front, we employ a novel dynamical systems based optimization technique called memcomputing, which has shown great promise in efficient optimization of non-convex energy landscapes like the DBM[20,27]. Since the DBM energy landscape can be represented within the optimizing dynamics, a path exists to extend these systems to (learning) samplers. In doing so, the need for CD would be entirely eliminated. We leave these avenues of research to be explored in future work.

Methods

DBMs are powerful models in machine learning, but bring with them considerable computational burdens during evaluation. To deal with them, we follow the procedure outlined in Ref.[14]. First, due to additional complexity, the log-likelihood is not directly maximized as in the case of RBMs. Instead, a variational lower bound to the log-likelihood is optimized. Second, the most common variational form chosen leads to mean-field updates for the data term. Finally, for evaluation of log-likelihood lower bounds, an approximation of the partition function is necessary. We provide a short description of all these approximations which are frequently encountered in DBM training scenarios.

Likelihood Lower Bound—Since the data expectation in Eq. (4) is no longer in closed form for the DBM, data-dependent statistics must be approximated with a sampling technique over the conditional distribution, $p(\mathbf{h}|\mathbf{v})$, where $\mathbf{h} = \{\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(\ell)}\}$. In practice, a variational lower bound to the log-likelihood is maximized instead, which is tractable and is found to work well (as in the model term)[14,28,29].

The variational approximation replaces the original posterior distribution, $p(\mathbf{h}|\mathbf{v})$ with an approximate distribution, $r(\mathbf{h}|\mathbf{v})$, where the parameters of r follow the gradient of the resulting *lower bound* on the original log-likelihood[28],

$$\begin{aligned} \log p(\mathbf{v}) &\geq \sum_{\mathbf{h}} r(\mathbf{h}|\mathbf{v}) \log p(\mathbf{v}, \mathbf{h}) + H_r(\mathbf{v}) \\ &= \log p(\mathbf{v}) - \mathrm{KL}(r(\mathbf{h}|\mathbf{v})||p(\mathbf{h}|\mathbf{v})), \end{aligned} \tag{9}$$

where $H_r(\mathbf{v}) = -\sum_{\mathbf{h}} r(\mathbf{h}|\mathbf{v}) \log r(\mathbf{h}|\mathbf{v})$ is the Shannon entropy of r. This variational loss simultaneously attempts to maximize the log-likelihood of the data set and minimize the KL divergence between the true conditional distribution, p and its approximation, r.

Data Term—A fully factorial mean field ansatz is often used in the variational approach, $r(\mathbf{h}|\mathbf{v}) = \prod_i r(h_i|\mathbf{v})$ with $r(h_i = 1|\mathbf{v}) = \mu_i$, with $\mu_i \in [0, 1]$ randomly initialized from a uniform distribution. Maximizing the lower bound in Eq. (9) results in updates of the form,

$$\mu_i \leftarrow \sigma\left(\sum_j W_{ij}^{(1)} v_j + \sum_{i \neq j} J_{ij}\mu_i + b_i\right). \tag{10}$$

Here J is a block matrix containing the weights of the hidden nodes and b_i is the bias of the i-th hidden node. Convergence is typically fast, (in our experiments less than 30 iterations are enough) and these states are then used to calculate the data expectation in Eq. (4) during the Gibbs phase of the training.

Partition Function—Computing the partition function in Eq. (2) exactly is infeasible for large DBMs. Its value is only required to evaluate the performance of the networks and appears in the log-likelihood as a normalizing constant. Annealed Importance Sampling (AIS)[30] is the procedure often used to approximate the partition function of large RBMs and DBMs. AIS estimates the ratio of partition functions, Z_N/Z_0, using a sequence of probability distributions between a chosen initial distribution and the desired one. The initial distribution, p_0, is chosen to have an exactly known partition function and to be simple to sample from (e.g. uniform) and p_N is the desired distribution whose partition function one wants to compute.

The sequence in the case of DBMs is parameterized by β_k (inverse temperatures), giving $p_k(\mathbf{x}) = e^{-\beta_k E(\mathbf{x})}/Z_k$. Markov chains are initialized uniformly according to p_0 and $0 \leq \beta_k \leq 1$ is slowly annealed to unity according to a desired schedule, all the while allowing the chains to run. The ratio is then approximated by the product of ratios of the unnormalized intermediate distributions,

$$\frac{Z_N}{Z_0} = \frac{p_1^*(\mathbf{x})}{p_0^*(\mathbf{x})} \cdots \frac{p_k^*(\mathbf{x})}{p_{k-1}^*(\mathbf{x})}. \tag{11}$$

When dealing with bipartite graphs like DBMs, either all the even or all the odd layers can be analytically traced out, resulting in a tighter approximation. For a direct comparison with previous work[14,18], we average over an identical number of 100 AIS runs with 29,000 linearly spaced intermediate distributions.

Sampling the Mode—The optimization technique used to sample the mode of the data and model distributions of DBMs is based on the digital memcomputing (DMM) approach[31], a computing paradigm based on computing with and in memory. There are both hardware designs[32] as well as software algorithms[27,33] that have seen success on difficult optimization problems compared to standard approaches. The application to machine learning is based on Refs.[19,20], which we briefly outline here.

Finding the mode of the DBM is equivalent to finding the ground state (or lowest energy state) of the DBM energy in Eq. (1). This can be written as the quadratic form, $E = -\mathbf{x}^T Q \mathbf{x}$, where $\mathbf{x}$ is the vector collection of all visible and hidden variables, and the block matrix Q can be read from Eq. (1). This optimization problem can be mapped to a weighted maximum satisfiability problem, or MAX-SAT problem, which is a logical expression between clauses of Boolean variables with associated weights. A system of ordinary differential equations representing a DMM is then constructed to find the configuration satisfying the largest weighted sum of satisfied clauses (which is also the ground state of the DBM). We have used the same differential equations reported in Ref.[20]. This system of equations is numerically integrated up to a fixed maximum number of time steps, and the state which achieved the lowest energy during the dynamics is read out. The modal update in Eq. (5) is then computed from this mode sample.

To account for the additional computational complexity introduced by mode sampling with DMMs, when a mode-sample is taken, the CD chain it is compared against is allowed to run for $k = 720$ iterations. This factor,

derived in Ref.[20], allows for a more fair comparison between the mode-assisted and its corresponding unassisted algorithm.

Data availibility

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

Received: 11 August 2021; Accepted: 6 September 2021
Published online: 24 September 2021

References

1. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. A learning algorithm for Boltzmann machines. *Cognit. Sci.* **9**, 147–169 (1985).
2. Smolensky, P. Information processing in dynamical systems: Foundations of harmony theory (1986).
3. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* **79**, 2554–2558 (1982).
4. Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **14**, 1771–1800 (2002).
5. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. *Deep learning* Vol. 1 (MIT Press, Cambridge, 2016).
6. Goodfellow, I. J., Courville, A. & Bengio, Y. Scaling up spike-and-slab models for unsupervised feature learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1902–1914 (2012).
7. Goodfellow, I. J. *et al.* Generative adversarial networks. arXiv:1406.2661 (2014).
8. Launchbury, J. A darpa perspective on artificial intelligence. *Retrieved November* **11**, 2019 (2017).
9. Erhan, D. *et al.* Why does unsupervised pre-training help deep learning?. *J. Mach. Learn. Res.* **11**, 625–660 (2010).
10. Gao, X. & Duan, L.-M. Efficient representation of quantum many-body states with deep neural networks. *Nat. Commun.* **8**, 1–6 (2017).
11. Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).
12. Melko, R. G., Carleo, G., Carrasquilla, J. & Cirac, J. I. Restricted boltzmann machines in quantum physics. *Nat. Phys.* **15**, 887–892 (2019).
13. Carleo, G. *et al.* Machine learning and the physical sciences. *Rev. Mod. Phys.* **91**, 045002 (2019).
14. Salakhutdinov, R. & Hinton, G. Deep Boltzmann machines. *Artif. Intell. Stat.* **448–455**, (2009).
15. Salakhutdinov, R., & Larochelle, H. Efficient learning of deep Boltzmann machines. *Proceedings of the thirteenth international conference on artificial intelligence and statistics* **693–700**, (2010).
16. Hinton, G. E., & Salakhutdinov, R. R. A better way to pretrain deep Boltzmann machines. *Adv. Neural Inf. Process. Syst.* **2447–2455**, (2012).
17. Goodfellow, I. J., Courville, A. & Bengio, Y. Joint training deep Boltzmann machines for classification. arXiv:1301.3568 (2013).
18. Melchior, J., Fischer, A. & Wiskott, L. How to center deep Boltzmann machines. *J. Mach. Learn. Res.* **17**, 3387–3447 (2016).
19. Manukian, H., Traversa, F. L. & Di Ventra, M. Accelerating deep learning with memcomputing. *Neural Netw.* **110**, 1–7 (2019).
20. Manukian, H., Pei, Y. R., Bearden, S. R. & Di Ventra, M. Mode-assisted unsupervised learning of restricted Boltzmann machines. *Commun. Phys.* **3**, 1–8 (2020).
21. Fischer, A. & Igel, C. An introduction to restricted boltzmann machines. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 14–36 (Springer* (eds Alvarez, L. *et al.*) (Berlin Heidelberg, Berlin, Heidelberg, 2012).
22. Fischer, A. & Igel, C. Empirical analysis of the divergence of gibbs sampling based learning algorithms for restricted boltzmann machines. In *International conference on artificial neural networks*, 208–217 (Springer, 2010).
23. Desjardins, G., Courville, A. & Bengio, Y. On training deep boltzmann machines. arXiv:1203.4416 (2012).
24. Bender, C. M., & Orszag, S. A. *Advanced mathematical methods for scientists and engineers I: Asymptotic methods and perturbation theory* (Springer, 2013).
25. Swendsen, R. H. & Wang, J.-S. Replica Monte Carlo simulation of spin-glasses. *Phys. Rev. Lett.* **57**, 2607 (1986).
26. Sherrington, D. & Kirkpatrick, S. Solvable model of a spin-glass. *Phys. Rev. Lett.* **35**, 1792 (1975).
27. Sheldon, F., Traversa, F. L. & Di Ventra, M. Taming a nonconvex landscape with dynamical long-range order: Memcomputing ising benchmarks. *Phys. Rev. E* **100**, 053311 (2019).
28. Neal, R. M. & Hinton, G. E. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, 355–368 (Springer, 1998).
29. Salakhutdinov, R. & Hinton, G. An efficient learning procedure for deep boltzmann machines. *Neural Comput.* **24**, 1967–2006 (2012).
30. Salakhutdinov, R. Learning and evaluating Boltzmann machines (2008).
31. Di Ventra, M. & Traversa, F. L. Memcomputing: Leveraging memory and physics to compute efficiently. *J. Appl. Phys.* **123**, 180901 (2018).
32. Traversa, F. L. & Di Ventra, M. Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines. *Chaos: Interdiscipl. J. Nonlinear Sci.* **27**, 023107 (2017).
33. Bearden, S. R., Pei, Y. R. & Di Ventra, M. Efficient solution of boolean satisfiability problems with digital memcomputing. *Sci. Rep.* **10**, 1–8 (2020).

Acknowledgements

Work supported by DARPA under grant No. HR00111990069. H.M. acknowledges support from a DoD-SMART fellowship. 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 the project. H.M. has done all the simulations reported and produced the figures in the paper. 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. (www.memcpu.com) that is attempting to commercialize the memcomputing technology. All other authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to H.M.

Reprints and permissions information is available at www.nature.com/reprints.

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