Restricted Boltzmann Machines are Hard to Approximately Evaluate or Simulate

Philip M. Long PLONG@GOOGLE.COM
Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

Rocco A. Servedio ROCCO@CS.COLUMBIA.EDU
Computer Science Department, Columbia University, New York, NY 10027

Abstract

Restricted Boltzmann Machines (RBMs) are a type of probability model over the Boolean cube $\{-1,1\}^n$ that have recently received much attention. We establish the intractability of two basic computational tasks involving RBMs, even if only a coarse approximation to the correct output is required.

We first show that assuming P $\neq$ NP, for any fixed positive constant K (which may be arbitrarily large) there is no polynomial-time algorithm for the following problem: given an n-bit input string x and the parameters of a RBM M, output an estimate of the probability assigned to x by M that is accurate to within a multiplicative factor of e^{Kn}. This hardness result holds even if the parameters of M are constrained to be at most $\psi(n)$ for any function $\psi(n)$ that grows faster than linearly, and if the number of hidden nodes of M is at most n.

We then show that assuming RP $\neq$ NP, there is no polynomial-time randomized algorithm for the following problem: given the parameters of an RBM M, generate a random example from a probability distribution whose total variation distance from the distribution defined by M is at most 1/12.

1. Introduction

A *Restricted Boltzmann Machine* (Smolensky, 1987; Freund & Haussler, 1991; Hinton, 2002; Bengio, 2009) (henceforth simply denoted "RBM") with m hidden nodes is defined by an m by n real matrix A and two vectors $a \in \mathbb{R}^m$, $b \in \mathbb{R}^n$. These parameters $\theta = (A, a, b)$ define a probability distribution RBM_θ over $x \in \{-1,1\}^n$ in the following way:

$$\mathrm{RBM}_\theta(x) \overset{\text{def}}{=} \frac{\sum_{h \in \{-1,1\}^m} e^{h^T a + h^T A x + b^T x}}{Z_\theta}, \quad (1)$$

where Z_θ is a normalizing factor (sometimes referred to as the "partition function") so that RBM_θ is a probability distribution, i.e.

$$Z_\theta \overset{\text{def}}{=} \sum_{h \in \{-1,1\}^m, z \in \{-1,1\}^n} e^{h^T a + h^T A z + b^T z}.$$

While RBMs were first introduced more than two decades ago (Smolensky, 1987; Freund & Haussler, 1991), they have recently been used as constituents of "deep belief network" learning systems (Hinton et al., 2006; Bengio, 2009). An approach to training deep networks that has been growing in popularity involves unsupervised training of RBMs as a subroutine. The success of this approach (see (Hinton et al., 2006; Larochelle et al., 2007; Erhan et al.)) motivates the subject of this paper, which is to study the complexity of basic computational tasks related to learning with RBMs.

Since RBMs are a way of modeling probability distributions over $\{-1,1\}^n$, the two most natural computational tasks regarding RBMs would seem to be the following:

1. **Evaluating an RBM**: Given a parameter vector θ and an input vector $x \in \{-1,1\}^n$, the task is to output the probability value $p = \mathrm{RBM}_\theta(x)$ that the distribution assigns to x.

   A potentially easier task is to *approximately* evaluate an RBM to within some multiplicative factor:

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

given a parameter vector θ, a vector $x \in \{-1,1\}^n$, and an approximation parameter $c > 1$, the task is to output a value $\hat{p}$ such that

$$\frac{1}{c} \cdot p \leq \hat{p} \leq c \cdot p.$$

2. **Simulating an RBM distribution**: Given a parameter vector θ and an approximation parameter $0 < \eta < 1$, the task is to output an efficiently evaluatable representation[1] of a probability distribution P over $\{-1,1\}^n$ such that the total variation distance between P and RBM_θ is at most η.

This paper shows that each of these tasks is computationally hard in the worst case, even if only a coarse approximation to the correct output is required.

As our first main result, we show that if P $\neq$ NP then the approximate evaluation task cannot be solved in polynomial time even with approximation parameter $c = e^{Kn}$, where $K > 0$ may be any fixed constant. (A precise statement of our hardness result, which is somewhat technical, is given as Theorem 8 in Section 4.) As our second main result, we show that if RP $\neq$ NP then there is no polynomial-time algorithm for the simulation task with approximation parameter $\eta = 1/12$. (See Theorem 13 in Section 5 for a precise statement.)

These results show strong worst-case limitations on evaluating and simulating general RBMs, but in many cases one may only be dealing with RBMs that have moderate-sized weights and relatively few hidden nodes. Thus it is of special interest to understand the complexity of approximately evaluating and simulating RBMs of this sort. We consider the approximate evaluation and simulation tasks when the RBM is restricted to have at most n hidden nodes and each real parameter $A_{i,j}, a_i, b_j$ has magnitude at most $\psi(n)$. Our hardness results stated above hold for any bound $\psi(n)$ that grows faster than linearly, i.e. such that $\lim_{n\to\infty} \frac{\psi(n)}{n} = \infty$.

1.1. Related work.

Our results have the same high-level flavor as the work of several previous researchers who have studied the computational complexity of various computational tasks for different types of probability distribution models. In an early work of this sort, Abe and Warmuth (1992) showed that it is NP-hard to approximate the maximum likelihood model for probabilistic automata with a fixed number of states but a variable-sized alphabet. Kearns *et al.* (1994) showed that it is $\#P$-hard to exactly evaluate probability models over $\{0,1\}^n$ in which each observed bit is a disjunction of three unknown hidden variables, each of which is assigned an unknown independent uniform random bit. Roth (1996) showed that it is NP-hard to approximate the probability that a given node in a multiple-connected Bayesian belief network is true. Bogdanov, Mossel and Vadhan (2008) studied the complexity of various computational tasks related to Markov Random Fields. Yasuda and Tanaka (2008) claim that training RBMs is NP-hard, but such a claim does not seem to address the approximate generation and approximate simulation computational tasks that we consider.

To the best of our knowledge, ours is the first work establishing hardness of either evaluation or simulation (exact or approximate) for RBMs.

1.2. Our approach.

An important ingredient in our hardness results for both evaluation and simulation is a more technical result (Theorem 1 in Section 3) which shows that it is NP-hard to even coarsely approximate the value of the partition function Z_θ. We prove this by combining a binary search technique (Lemma 6) with a recent hardness result for approximating the pseudo-cut-norm of a matrix due to Alon and Naor (2004).

Our hardness result for approximate evaluation, Theorem 8, follows rather straightforwardly from Theorem 1. For our simulation hardness result, we combine Theorem 1 with ideas from Jerrum, Valiant and Vazirani's proof (1986) that approximate uniform sampling from a set S and approximate counting of the elements of S are polynomial-time equivalent.

2. Background and Preliminaries

We drop the subscript θ from Z_θ when there is no possibility of confusion. We sometimes concentrate on models whose parameters $\theta = (A, a, b)$ have $a = \vec{0}$ and $b = \vec{0}$. In this case, we sometimes replace θ with A, for example writing RBM_A and Z_A.

For A a matrix with real entries, we write $||A||_\infty$ to denote $\max_{i,j} |A_{ij}|$. Similarly we write $\|a\|_\infty$ to denote $\max_i |a_i|$ for a vector a.

[1]We formalize the notion of an "efficiently evaluatable representation of a distribution over $\{-1,1\}^n$" in the following standard way: Such a representation consists of a Boolean circuit of $\mathrm{poly}(n)$ size with $k = \mathrm{poly}(n)$ input bits and n output bits. A draw from the distribution is obtained by setting the input bits to an uniform random k-bit string and reading the output string.

For probability distributions P and Q over a finite set S, the total variation distance is $d_{TV}(P,Q) \overset{\text{def}}{=} \max_{E\subseteq S}|P(E) - Q(E)|$. Note that this is equivalent to $d_{TV}(P,Q) \overset{\text{def}}{=} \max_{E\subseteq S} P(E) - Q(E)$ since $P(S-E) - Q(S-E) = Q(E) - P(E)$.

A function ψ is $\omega(n)$ if it grows faster than linearly, i.e., if $\lim_{n\to\infty} \frac{\psi(n)}{n} = \infty$.

3. Approximating the partition function is hard

The main result of this section is the following theorem, which says that it is hard to approximate the partition function Z_θ:

Theorem 1 *There is a universal constant $\varepsilon > 0$ such that if $P \neq NP$, then there is no polynomial-time algorithm with the following property: Given as input an $n \times n$ matrix A satisfying $\|A\|_\infty \leq \psi(n)$ (where the function ψ satisfies $\psi(n) = \omega(n)$), the algorithm approximates the partition function Z_A to within a multiplicative factor of $e^{\epsilon\psi(n)}$.*

Our proof uses a reduction from the problem of approximating a norm defined by Alon and Naor (2004). We refer to this norm, which is denoted by $\|A\|_{\infty\mapsto 1}$ in (Alon & Naor, 2004), as the *pseudo-cut-norm* of A and denote it simply by $\|A\|$.

Definition 2 *The* pseudo-cut-norm *of an $m \times n$ real matrix A is defined by*

$$||A|| \overset{def}{=} \max_{h\in\{-1,1\}^m, x\in\{-1,1\}^n} h^T Ax.$$

Theorem 3 ((Alon & Naor, 2004)) *There is a universal constant $\varepsilon > 0$ such that, if $P \neq NP$, then there is no polynomial-time algorithm that approximates the pseudo-cut-norm to within a factor of $1+\varepsilon$.*

The reduction in the proof of Theorem 3 in (Alon & Naor, 2004) uses non-square matrices, but an easy corollary extends this hardness result to square matrices (we give the simple proof in Appendix A):

Corollary 4 *Theorem 3 holds even if the matrix is constrained to be square.*

We will need the following upper and lower bounds on the pseudo-cut-norm of A:

Lemma 5 *For an $m \times n$ matrix A, we have $||A||_\infty \leq ||A|| \leq mn||A||_\infty$.*

Proof: Alon and Naor (2004) note that the pseudo-cut-norm satisfies

$$||A|| \geq \max_{u,v\in\{0,1\}^n} \left|u^T Av\right|$$

(the RHS above is the actual "cut-norm" of A). Since A_{ij} equals $e_i^T Ae_j$ (where e_i has a 1 in the ith coordinate and 0's elsewhere), we get

$$||A|| \geq \max_{i,j} |A_{ij}| = \|A\|_\infty.$$

It remains only to observe that for any $h \in \{-1,1\}^m, x \in \{-1,1\}^n$, we have

$$h^T Ax = \sum_{ij} h_i x_j A_{ij} \leq mn||A_{ij}||_\infty. \qquad \square$$

3.1. Proof of Theorem 1

Throughout this section ψ denotes a function $\psi(n) = \omega(n)$ as in Theorem 1.

We first show that it is hard to distinguish between matrices with "large" versus "slightly less large" pseudo-cut-norm:

Lemma 6 *There is a universal constant $\alpha > 0$ such that if $P \neq NP$, then there is no polynomial-time algorithm to solve the following promise problem:*

Input: *An n-by-n matrix A such that $||A||_\infty \leq \psi(n)$ and either (i) $||A|| > \psi(n)$; or (ii) $||A|| \leq (1-\alpha)\psi(n)$.*

Output: *Answer whether (i) or (ii) holds.*

Proof: The proof is by contradiction; so suppose that for every $\alpha > 0$, ALG_α is a polynomial-time algorithm that solves the promise problem with parameter α. We will show that there is a polynomial-time algorithm ALG'_α (which performs a rough binary search using ALG_α as a subroutine) that can approximate the pseudo-cut-norm $\|B\|$ of any $n\times n$ input matrix B to within a multiplicative factor $\left(\frac{1}{1-\alpha}\right)^2$. This yields a contradiction with Corollary 4.

So let B be any $n \times n$ input matrix. Since $||\lambda B|| = \lambda||B||$, we may rescale B as a preprocessing step, so we assume without loss of generality that B has $||B||_\infty = 1$. Now fix any $c > 1$ and consider an execution of ALG_α on the matrix $A = (\psi(n)/c)B$. If ALG_α returns "(i)" then (ii) does not hold, so $||A|| > (1-\alpha)\psi(n)$, which implies that $\|B\| > (1-\alpha)c$. Similarly, if ALG_α returns "(ii)" then B must have $\|B\| \leq c$.

The algorithm ALG'_α maintains an interval $[\ell, u]$ of possible values for $\log ||B||$ which it successively prunes using ALG_α to do a rough binary search. Using $\|B\|_\infty = 1$ and Lemma 5, initially we may take $[\ell, u] = [0, 2\ln n]$ to be an interval of length $r_0 = 2\ln n$. After the tth stage of binary search using ALG_α, the new length r_t of the interval is related to the old length r_{t-1} by $r_t \leq r_{t-1}/2 + \log(1/(1-\alpha))$. As long as r_{t-1} is at least $4\log(1/(1-\alpha))$, this implies $r_t \leq 3r_{t-1}/4$.

So after $O(\log\log n)$ iterations of binary search, ALG'_α narrows the initial interval $[0, 2\ln n]$ to an interval $[\ell, u]$ of width at most $4\log(1/(1-\alpha))$. (We note that each execution of ALG_α in the binary search indeed uses a value c which is at least 1 as required.) Algorithm ALG'_α outputs $e^{(u+\ell)/2}$ as its estimate of $\|B\|$.

Since $u - \ell \leq 4\log(1/(1-\alpha))$ implies that $e^{u-\ell} \leq \left(\frac{1}{1-\alpha}\right)^4$, the estimate $e^{(u+\ell)/2}$ is accurate for $\|B\|$ to within a multiplicative approximation factor of $\left(\frac{1}{1-\alpha}\right)^2$. As noted at the start of the proof, since α could be any constant greater than 0, this contradicts hardness of approximating $\|B\|$ (Theorem 3). □

An easy consequence of Lemma 6 is that it is hard to distinguish between RBMs whose partition functions are "large" versus "much less large":

Lemma 7 *There is a universal constant $\alpha > 0$ such that if $P \neq NP$, then there is no polynomial-time algorithm to solve the following promise problem:*

> **Input:** *An n-by-n matrix A such that $||A||_\infty \leq \psi(n)$ and either (i) $Z_A > \exp(\psi(n))$; or (ii) $Z_A \leq 4^n \exp((1-\alpha)\psi(n))$.*
>
> **Output:** *Answer whether (i) or (ii) holds.*

Proof: By Lemma 6, if an n-by-n matrix A satisfies $||A||_\infty \leq \psi(n)$ and either (a) $\max_{h,x} h^T Ax > \psi(n)$ holds or (b) $\max_{h,x} h^T Ax \leq (1-\alpha)\psi(n)$ holds, it is hard to determine whether (a) or (b) holds.

It is clear that (a) implies (i) and that (b) implies (ii). Since $\psi(n) = \omega(n)$, for all but finitely many n we have that the two alternatives (i) and (ii) are mutually exclusive. So for all sufficiently large n, an algorithm to determine whether (i) or (ii) holds could directly be used to determine whether (a) or (b) holds. □

Proof of Theorem 1: Let $\alpha > 0$ be the constant from Lemma 7. Let $U = \exp(\psi(n))$ and $L = 4^n \exp((1-\alpha)\psi(n))$. An algorithm that can approximate Z to within a multiplicative factor of $\sqrt{U/L}$ can distinguish $Z \geq U$ from $Z < L$. We have

$$\sqrt{\frac{U}{L}} = \exp\left(\frac{\alpha\psi(n) - n\ln 4}{2}\right),$$

so an approximation factor better than this would contradict Lemma 7, and thus any $\epsilon < \alpha/2$ suffices in Theorem 1. □

4. Approximate Evaluation is Hard

In this section we show that it is hard to approximately evaluate a given RBM A on a given input string x:

Theorem 8 *There is a universal constant $\varepsilon > 0$ such that if $P \neq NP$, then there is no polynomial-time algorithm with the following property: Given as input an $n \times n$ matrix A satisfying $\|A\|_\infty \leq \psi(n)$ (where the function ψ satisfies $\psi(n) = \omega(n)$) and an input string $x \in \{-1, 1\}^n$, the algorithm approximates the probability $\mathrm{RBM}_A(x)$ to within a multiplicative factor of $e^{\varepsilon\psi(n)}$.*

Note that since $\psi(n) = \omega(n)$, the above result implies that approximating $\mathrm{RBM}_A(x)$ to the smaller multiplicative factor e^{Kn} is also hard, where K may be any positive constant.

We will use the fact that the numerator of the expression (1) for $\mathrm{RBM}_A(x)$ can be computed efficiently, which is known and not difficult to show. (See e.g. (5.12) of (Bengio, 2009) for an explicit proof.)

Lemma 9 *There is a* $\mathrm{poly}(n)$*-time algorithm that, given A and x, computes* $\sum_{h\in\{-1,1\}^n} \exp(h^T Ax)$.

Now we are ready for the proof that evaluation is hard.

Proof of Theorem 8: We actually show something stronger: if P ≠ NP then there is no polynomial-time algorithm which, given an RBM A as described, can output *any pair* $(x, \widetilde{\mathrm{RBM}}_A(x))$, where $\widetilde{\mathrm{RBM}}_A(x)$ is a multiplicative $e^{\varepsilon\psi(n)}$-approximation to $\mathrm{RBM}_A(x)$. (In other words, not only is it hard to approximate $\mathrm{RBM}_A(x)$ for a worst-case pair (A, x), but for a worst-case A it is hard to approximate $\mathrm{RBM}_A(x)$ for *any* x.)

To see this, note that since Lemma 9 implies that we can efficiently *exactly* evaluate the numerator of the expression (1) for $\mathrm{RBM}_A(x)$, approximating $\mathrm{RBM}_A(x)$ to within a given multiplicative factor is equivalent to approximating $1/Z_A$ to within the same factor.

But since $f(u) = 1/u$ is monotone in u, for an estimate $\widetilde{Z}$ of Z and a desired approximation factor c, we of course have that

$$\frac{1}{c} \times \frac{1}{Z} \leq \frac{1}{\hat{Z}} \leq c \times \frac{1}{Z}$$

if and only if

$$cZ \geq \hat{Z} \geq Z/c,$$

so we are done by Theorem 1. □.

5. Approximate Simulation is Hard

In this section we establish the hardness of constructing an efficiently evaluatable representation of any distribution P that is close to RBM_θ, where $\theta = (A, a, b)$ is a given set of RBM parameters. Our proof is inspired by Jerrum, Valiant and Vazirani's proof that approximate counting reduces to approximate uniform sampling (Jerrum et al., 1986). To aid in explaining our proof, in the following paragraph we briefly recall the idea of the (Jerrum et al., 1986) reduction from approximate counting to approximate uniform sampling, and then explain the connection to our scenario.

Let $S \subseteq \{0,1\}^n$ be a set whose elements we would like to approximately count, i.e. our goal is to approximate $|S|$. Suppose that ALG is an algorithm that can approximately sample a uniform element from S, i.e. each element of S is returned by ALG with some probability in the range $[\frac{1}{1+\tau} \cdot \frac{1}{|S|}, (1+\tau) \cdot \frac{1}{|S|}]$. The reduction proceeds for n stages. In the first stage, by drawing samples from S using ALG, it is possible to approximately estimate $|S_0|$ and $|S_1|$ where S_b is $\{x \in S : x_1 = b\}$, so that the larger one is estimated to within a multiplicative factor of $(1 + 2\tau)$. Let $b_1 \in \{0,1\}$ be the bit such that $|\hat{S}_{b_1}| \geq |S|/2$, where $|\hat{S}_{b_1}|$ is the estimated size of $|S_{b_1}|$, and let $\hat{p}_{1,b_1} \geq 1/2$ denote $|\hat{S}_{b_1}|/|S|$. In the second stage we repeat this process, using ALG on S_{b_1}, to obtain values b_2, $|\hat{S}_{b_1,b_2}|$ and $\hat{p}_{2,b_2}$, where $|\hat{S}_{b_1,b_2}|$ is the estimated size of $S_{b_1,b_2} = \{x \in S : x_1 = b_1$ and $x_2 = b_2\}$. By continuing in this way for n stages, we reach a set $S_{b_1,\ldots,b_n}$ of size 1. The final estimate of $|S|$ is $1/\prod_{i=1}^n \hat{p}_{i,b_i}$, and it can be shown that this is an accurate approximator of $|S|$ to within a multiplicative factor of $(1+2\tau)^n$.

In our setting the ability to approximately simulate a given RBM_θ distribution plays the role of the ability to approximately sample from sets like $S_{b_1,\ldots,b_i}$. Approximate counting of $|S|$ corresponds to approximately computing $\mathrm{RBM}_\theta(x)$ for a particular string x (which corresponds to the bitstring $b_1 \ldots b_n$). Since Theorem 1 implies that approximating $\mathrm{RBM}_\theta(x)$ is hard, it must be the case that approximately simulating a given RBM distribution is also hard.

5.1. Preliminaries

As the above proof sketch suggests, we will need to build RBM models for various distributions obtained by conditioning on fixed values of some of the observed variables.

Definition 10 *Let P be a probability distribution over $\{-1,1\}^n$, $i_1, \ldots, i_k$ be a list of distinct variable indices from $[n]$, and $x_{i_1}, \ldots, x_{i_k}$ be values for those variables. We write* $\mathrm{cond}(P; X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$ *to denote the distribution obtained by drawing a random variable $(X_1, ..., X_n)$ distributed according to P and conditioning on the event that $X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}$.*

It will be helpful to have notation for the numerator in the formula for the probability assigned by an RBM.

Definition 11 *For an RBM $\theta = (A, a, b)$ and a string $x \in \{-1,1\}^n$, define the* energy *of x* w.r.t. *θ, denoted by $f_\theta(x)$, to be $\sum_{h \in \{-1,1\}^n} \exp(h^T a + h^T A x + b^T x)$.*

5.2. Building RBM Models for Conditional Distributions

In the following lemmas, for parameters $\theta = (A, a, b)$ we write $\|\theta\|_\infty$ to denote $\max\{\|A\|_\infty, \|a\|_\infty, \|b\|_\infty\}$.

Lemma 12 *There is a poly(n,$1/\eta$, $\|\theta\|_\infty$)-time algorithm with the following properties: The algorithm is given any $\eta > 0$, any parameters $\theta = (A, a, b)$ where A is an $n \times n$ matrix (and all the parameters of A, a, b have poly(n) bits of precision), and any values $u_1, \ldots, u_k \in \{-1,1\}$ for observed variables $X_1, \ldots, X_k$. The algorithm outputs a parameterization $\theta' = (A', a', b')$ of an RBM such that $d_{TV}(\mathrm{RBM}_{\theta'}, \mathrm{cond}(\mathrm{RBM}_\theta; X_1 = u_1, ..., X_i = u_k)) \leq \eta$. Moreover, the matrix A is $(n+k) \times n$, the parameterization θ' satisfies $\|\theta'\|_\infty \leq \mathrm{poly}(\|\theta\|_\infty, n, \log 1/\eta)$, and all the parameters A', a', b' have poly(n) + $O(\log\log(1/\eta))$ bits of precision.*

Proof: We will start by describing how the algorithm handles the case in which $u_1 = ... = u_k = 1$, and then describe how to handle the general case at the end of the proof.

The RBM parameterized by θ' is obtained from θ by

- adding k extra hidden nodes;
- adding k rows to A, and making $A_{n+j,j}$ a large positive value M (we will show that an integer value that is not too large will suffice), and the other components of the $n+j$th row are all 0;
- adding k components to a that are the same large value M. The vector b' equals b.

Roughly, the idea is as follows. Each hidden node clamps the value of one variable. The role of a_{n+j} is to clamp the value of the jth extra hidden variable to 1. The role of $A_{n+j,j}$ is to force X_j to be equal to the value of the jth extra hidden variable, and therefore equal to 1.

Let A', a' and b' be these parameters. Breaking up the vector of hidden variables into its new components and old components, and applying the definitions of A' and h', we have

$$\begin{aligned} & f_{\theta'}(x_1, ..., x_n) \\ &= \sum_{h \in \{-1,1\}^{n+k}} \exp(h^T a' + h^T A' x + (b')^T x) \\ &= \sum_{h \in \{-1,1\}^{n}} \sum_{g \in \{-1,1\}^{k}} \exp(h^T a + h^T A x + b^T x \\ &\qquad + \sum_{j=1}^{k} M x_j g_j + M g_j) \end{aligned}$$

which immediately gives

$$\begin{aligned} & f_{\theta'}(x_1, ..., x_n) \\ &= \left(\sum_{g \in \{-1,1\}^{k}} \exp(\sum_{j=1}^{k} M x_j g_j + M g_j) \right) \\ &\qquad \times \sum_{h \in \{-1,1\}^{n}} \exp(h^T a + h^T A x + b^T x) \\ &= \left(\sum_{g \in \{-1,1\}^{k}} \exp(\sum_{j=1}^{k} M x_j g_j + M g_j) \right) \\ &\qquad \times f_\theta(x_1, ..., x_n). \end{aligned}$$

Let F be the event that $X_1 = 1, ..., X_k = 1$ and let

$$\psi(x) = \left(\sum_{g \in \{-1,1\}^{k}} \exp(\sum_{j=1}^{k} M x_j g_j + M g_j) \right).$$

Note that $\psi(x)$ is maximized using any $x \in F$. Let $\psi(F)$ be this value. Note that $\psi(F) \geq e^{2kM}$, and $\psi(x) \leq 2^k e^{(2k-2)M}$ otherwise, so that

$$\forall x \notin F,\ \psi(x) \leq 2^k e^{-2M} \psi(F). \tag{2}$$

Now fix any event E, and let $\mathcal{X}$ denote $\{-1,1\}^n$. We have

$$\mathrm{RBM}_{\theta'}(E) \leq \mathrm{RBM}_{\theta'}(E|F) + \mathrm{RBM}_{\theta'}(\neg F). \tag{3}$$

First,

$$\mathrm{RBM}_{\theta'}(E|F) = \frac{\sum_{x \in E \cap F} \psi(F) f_\theta(x)}{\sum_{x \in F} \psi(F) f_\theta(x)} = \mathrm{RBM}_{\theta}(E|F). \tag{4}$$

For any $x \notin F$, let the "correction" of x, called $c(x)$, be the member of F obtained by setting the first k components of x to 1. We have

$$\begin{aligned} & \mathrm{RBM}_{\theta'}(\neg F) \\ &= \frac{\sum_{x \notin F} \psi(x) f_\theta(x)}{\sum_{x \in \mathcal{X}} \psi(x) f_\theta(x)} \\ &\leq \frac{\sum_{x \notin F} \psi(x) f_\theta(x)}{\sum_{x \in F} \psi(x) f_\theta(x)} \\ &= \frac{(2^k - 1) \sum_{x \notin F} \psi(x) f_\theta(x)}{\sum_{x \in F} (2^k - 1) \psi(x) f_\theta(x)} \\ &= \frac{(2^k - 1) \sum_{x \notin F} \psi(x) f_\theta(x)}{\sum_{x \notin F} \psi(c(x)) f_\theta(c(x))} \\ &\qquad \text{(since, } \forall x \in F, |c^{-1}(\{x\})| = 2^k - 1) \\ &= \frac{(2^k - 1) \sum_{x \notin F} \psi(x) f_\theta(x)}{\sum_{x \notin F} \psi(F) f_\theta(c(x))} \quad \text{(since } c(x) \in F) \\ &\leq \frac{4^k e^{-2M} \sum_{x \notin F} f_\theta(x)}{\sum_{x \notin F} f_\theta(c(x))} \end{aligned} \tag{5}$$

by (2).

Finally, since $c(x)$ differs from x in k components, we have that for all $x \notin F$,

$$f_\theta(x) \leq e^{2k(n+1)||\theta||_\infty} f_\theta(c(x)). \tag{6}$$

Thus (5) implies that

$$\mathrm{RBM}_{\theta'}(\neg F) \leq 4^k e^{2k(n+1)||\theta||_\infty} e^{-2M}$$

and this, together with (3) and (4), implies

$$\mathrm{RBM}_{\theta'}(E) \leq \mathrm{cond}(\mathrm{RBM}_\theta; F)(E) + 4^k e^{2k(n+1)||\theta||_\infty} e^{-2M}.$$

This implies that $\mathrm{RBM}_{\theta'}(E) \leq \mathrm{cond}(\mathrm{RBM}_\theta; F)(E) + \eta$ provided that $M \geq (1/2)\ln(1/\eta) + k \ln 2 + k(n+1)||\theta||_\infty$. Since the event E was arbitrary, recalling the definition of total variation distance, this completes the proof in the case $u_1 = ... = u_k = 1$.

The general case can be treated with a nearly identical analysis, after setting $A_{n+j,j}$ to be $u_j M$ for each $j = 1, \ldots, k$. □

5.3. Approximate Simulation is Hard

Theorem 13 *If $RP \neq NP$, then there is no polynomial-time algorithm with the following property: Given as input $\theta = (A, a, b)$ such that A is an $n \times n$ matrix and $\|\theta\|_\infty \leq \psi(n)$ (where $\psi(n) = \omega(n)$), the algorithm outputs an efficiently evaluatable representation of a distribution whose total variation distance from* RBM_θ *is at most $\eta = 1/12$.*

Proof: The proof is by contradiction; here is a high-level outline. We will suppose that OUTPUT-DIST is a polynomial-time algorithm that, on input θ, constructs an η-close distribution to RBM_θ. We will show that there is a randomized algorithm which, using OUTPUT-DIST, with high probability (at least 9/10) can efficiently find a particular $x \in \{-1,1\}^n$ for which the value of $\mathrm{RBM}_\theta(x)$ can be estimated to within a multiplicative factor of 2^n. Since $\mathrm{RBM}_\theta(x) = f_\theta(x)/Z_\theta$, and we can efficiently exactly compute $f_\theta(x)$ (see Lemma 9), this will imply that we can approximate Z_θ to within a factor 2^n. Since 2^n is less than $e^{\varepsilon\psi(n)}$, where ε is the constant from Theorem 1, this contradicts Theorem 1 unless RP = NP.

The randomized algorithm creates $x = (x_1, ..., x_n)$ as follows. Fix $\delta = 1/10$. Set $\theta_1 = \theta$, and, for each $i \in [n]$ in turn,

- Run OUTPUT-DIST to obtain an efficiently evaluatable representation of a distribution P_i such that $d_{TV}(P_i, \mathrm{RBM}_{\theta_i}) \le \eta$;
- Sample $(4/\eta^2)\log(n/\delta)$ times from P_i (note that this can be done efficiently);
- Set x_i to be the bit that appears most often for the ith component X_i in the sampled strings;
- Use Lemma 12 to construct a set of RBM parameters θ_{i+1} such that $d_{TV}(\mathrm{RBM}_{\theta_{i+1}}, \mathrm{cond}(\mathrm{RBM}_\theta; X_1 = x_1, ..., X_i = x_i)) \le \eta$.

For each $i \in [n]$ let $\hat{p}_i$ denote the fraction of times that $X_i = x_i$ over the samples drawn during the ith stage. The Chernoff-Hoeffding bound implies that, with probability $1 - \delta$, for all $i \in [n]$ we have

$$|\hat{p}_i - P_i[X_i = x_i]| \le \eta$$

(where the notation $\mathcal{D}[E]$ denotes the probability of event E under distribution $\mathcal{D}$). Since $d_{TV}(P_i, \mathrm{RBM}_{\theta_i}) \le \eta$, this implies that

$$|\hat{p}_i - \mathrm{RBM}_{\theta_i}[X_i = x_i]| \le 2\eta,$$

and the fact that $d_{TV}(\mathrm{RBM}_{\theta_i}, \mathrm{cond}(\mathrm{RBM}_\theta; X_1 = x_1, ..., X_{i-1} = x_{i-1})) \le \eta$ implies that

$$|\hat{p}_i - \mathrm{RBM}_\theta[X_i = x_i \mid X_1 = x_1, ..., X_{i-1} = x_{i-1}]| \le 3\eta.$$

Since $\eta = 1/12$ and $\hat{p}_i \ge 1/2$, this implies that

$$\mathrm{RBM}_\theta[X_i = x_i \mid X_1 = x_1, ..., X_{i-1} = x_{i-1}] \ge 1/4$$

and therefore that

$$\frac{1}{2} \le \frac{\hat{p}_i}{\mathrm{RBM}_\theta[X_i = x_i \mid X_1 = x_1, ..., X_{i-1} = x_{i-1}]|} \le 2. \tag{7}$$

Let us compute an estimate $\hat{p}$ of $\mathrm{RBM}_\theta(x)$ by setting $\hat{p} = \prod_i \hat{p}_i$. We can use the probability chain rule to evaluate the accuracy of this estimate as follows:

$$\frac{\hat{p}}{\mathrm{RBM}_\theta(x)} = \frac{\prod_i \hat{p}_i}{\prod_i \mathrm{RBM}_\theta(x_i|x_1...x_{i-1})} \in \left[\frac{1}{2^n}, 2^n\right].$$

As mentioned above, since $\mathrm{RBM}_\theta(x) = f_\theta(x)/Z_\theta$ and we can exactly compute $f_\theta(x)$ in polynomial time, this implies that we can estimate $1/Z_\theta$ to within a 2^n factor, and, as noted in the proof of Theorem 8, this implies that we can estimate Z_θ to within a 2^n factor. This contradicts Theorem 1 and completes the proof. □

6. Discussion

We have established strong worst-case computational hardness results for the basic tasks of evaluating and simulating RBMs. We view these hardness results as providing additional motivation to obtain a comprehensive theoretical understanding of why and how RBMs perform well in practice. One possibility is that the parameters of real-world RBMs tend to be even smaller than the bounds satisfied by the constructions used to establish our results; the recent analysis of (Bengio & Delalleau, 2009) seems to conform with this possibility. Further study of the theoretical benefits of different properties of RBM models, and of algorithms that promote those properties, may lead to improvements in the practical state of the art of learning using these models.

References

Abe, N. and Warmuth, M. K. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9(2–3):205–260, 1992.

Alon, N. and Naor, A. Approximating the cut-norm via Grothendieck's inequality. In *STOC*, pp. 72–80, 2004.

Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Bengio, Y. and Delalleau, O. Justifying and generalizing contrastive divergence. *Neural Computation*, 21 (6):1601–1621, 2009.

Bogdanov, A., Mossel, E., and Vadhan, S. P. The complexity of distinguishing markov random fields. In *APPROX-RANDOM*, pp. 331–342, 2008.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*. To appear.

Freund, Y. and Haussler, D. Unsupervised learning of distributions of binary vectors using 2-layer networks. In *NIPS*, pp. 912–919, 1991.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Jerrum, M., Valiant, L. G., and Vazirani, V. V. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

Kearns, M., Mansour, Y., Ron, D., Rubinfeld, R., Schapire, R., and Sellie, L. On the learnability of discrete distributions. In *Proc. of Twenty-sixth ACM Symposium on Theory of Computing*, pp. 273–282, 1994.

Larochelle, H., Erhan, D., Courville, A. C., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pp. 473–480, 2007.

Roth, D. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

Smolensky, P. Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E., McClelland, J. L., et al. (eds.), *Parallel Distributed Processing: Volume 1: Foundations*, pp. 194–281. MIT Press, Cambridge, 1987.

Yasuda, M. and Tanaka, K. Approximate learning algorithm for restricted boltzmann machines. In *CIMCA/IAWTIC/ISE*, pp. 692–697, 2008.

A. Proof of Corollary 4

Let A be a non-square $m \times n$ matrix; we suppose $m > n$ (the other case is entirely similar). We may pad A with $m - n$ all-zero columns on the right, to form an $m \times m$ matrix B which is at most quadratically larger than A. We claim that the pseudo-cut-norm of B is the same as the pseudo-cut-norm of A: for any $h, u \in \{-1, 1\}^m$, if we form x out of the first n components of u, then, since the last $m - n$ columns of B are all zeroes, $h^T Bu$ does not depend on the last $m - n$ components of u, so $h^T Bu = h^T Ax$. Since any h and x may be formed this way, we have

$$||A|| = \max_{h,x} h^T Ax = \max_{h,u} h^T Bu = ||B||.$$

Thus, a polynomial-time algorithm for approximating the pseudo-cut-norm for square matrices (like B) to within an arbitrary constant factor would yield a corresponding polynomial-time algorithm for approximating the pseudo-cut-norm of matrices (like A) for which $m > n$.