Stochastic Spectral Descent for Restricted Boltzmann Machines

David Carlson[1] **Volkan Cevher**[2] **Lawrence Carin**[1]
[1] Department of Electrical and Computer Engineering, Duke University
[2]Laboratory for Information and Inference Systems (LIONS), EPFL

Abstract

Restricted Boltzmann Machines (RBMs) are widely used as building blocks for deep learning models. Learning typically proceeds by using stochastic gradient descent, and the gradients are estimated with sampling methods. However, the gradient estimation is a computational bottleneck, so better use of the gradients will speed up the descent algorithm. To this end, we first derive upper bounds on the RBM cost function, then show that descent methods can have natural advantages by operating in the ℓ_∞ and Shatten-∞ norm. We introduce a new method called "Stochastic Spectral Descent" that updates parameters in the normed space. Empirical results show dramatic improvements over stochastic gradient descent, and have only have a fractional increase on the per-iteration cost.

1 Introduction

Deep learning methods are becoming increasingly popular for feature extraction applications, having produced state-of-the-art results for many classification problems [Bengio, 2012]. The impressiveness of the results are unfortunately matched by the (often) extraordinary amount of computation that goes into training them.

Instead of optimizing the cost function generated by a deep network model directly, we can use a divide and conquer strategy via Restricted Boltzmann Machines (RBMs). An RBM is a probabilistic generative model over binary observations and binary hidden nodes, with connections only between the observed visible nodes and unobserved hidden nodes. Indeed, one can exploit RBMs as a building block for learning Deep Restricted Boltzmann Machines (DRBMs) [Salakhutdinov and Hinton, 2009], Deep Belief Networks (DBN) [Hinton et al., 2006] and Deep Sigmoid Belief Nets [Neal, 1992, Mnih and Gregor, 2014]. *To this end, this paper introduces a new method, motivated by convex optimization principles, to improve the efficiency of training the RBM models.*

To explain the novelty in our approach, we first elaborate on the specific computational challenge we address in this paper. The training objective of the RBM model is expressed as a minimization of a non-convex composite objective over a matrix, that connects the hidden and visible units as well as bias terms for both the visible and hidden units. The first term in the objective is convex and captures the partition function of the RBM. The second term is concave and encodes the influence of the observations. Since calculating the gradient of the individual terms imposes a significant computational burden, we leverage Monte Carlo integration via Contrastive Divergence (CD) [Hinton, 2002, Tieleman and Hinton, 2009] to obtain statistical estimates.

Because the cost of estimating the gradients is a major bottleneck, our key contention is that exploiting the information in the gradients more effectively can greatly speed up the overall inference. To achieve this desideratum, we change the space in which our gradient method operates to better match the geometry of the RBM problem. Indeed, we treat the matrix variable explicitly as a matrix by choosing a normed space based on the Shatten-∞ norm (i.e., the spectral norm). We use a similar idea for the bias variables and operate in the ℓ_∞-space. Changing the normed space impacts the optimization radius, which can provably improve optimization efficiency in the deterministic setting.

Our algorithm, termed *stochastic spectral descent* (SSD) operates as follows: we obtain the stochastic gradient estimates using CD. We then use an appropriate *sharp*-operator, which applies a nonlinear transformation on the gradient. We update the putative

Appearing in Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38. Copyright 2015 by the authors.

solution using the *gradient-sharp* and a constant step-size. For the matrix variables, the sharp operator takes the singular value decomposition of the gradient and sets all the nontrivial singular values to the trace-norm of the gradient. For the bias terms, the sharp-operator replaces the vector coefficients simply with their sign (i.e., ± 1) and multiplies the overall vector with the ℓ_1-norm of the original gradient.

From a deterministic convex optimization perspective, the proposed algorithm is almost classic [Nesterov, 2012]. In fact, we can derive the algorithm as a majorization-minimization method by minimizing an upper bound of the objective equation. However, to the best of our knowledge, its application to learning deep networks is novel, where the deterministic gradient estimates are replaced by their stochastic estimates. While we do not provide a rigorous convergence proof of our algorithm in this paper, we provide enough empirical evidence to justify its usage.

Recent work on improving the efficiency of training RBMs largely focus on stochastic gradient descent and its variations. These variations include autotuning of the step-size in SGD [Schaul et al., 2012], the Enhanced Gradient method [Cho et al., 2013], and using a variable metric [Duchi et al., 2010]. For training RBMs, we can also change the cost function to improve training quality. Dropout RBM [Srivastava et al., 2014], for instance, emprically reduces the number of dead units. Here, the focus is on using the geometry for theory-based algorithms instead of improving step-size selection or gradient estimates. However, like the Dropout procedure, the SSD empirically learns an RBM that has higher usages of hidden nodes with fewer "dead units."

2 Preliminaries and Model Definitions

2.1 Notation

Bold lower-case letters represent vectors, and bold upper-case letters represent matrices. $\langle \cdot, \cdot \rangle$ denotes an inner product, and $\boldsymbol{x} \odot \boldsymbol{y}$ denotes element-wise multiplication. The ℓ_p norm for a vector $\boldsymbol{x}$ is defined $||\boldsymbol{x}||_p = (\sum_n |x_m|^p)^{1/p}$. Letting $\boldsymbol{\lambda}$ be the vector of singular values of a matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$, then the Shatten p-norm is defined as $||\mathbf{X}||_{S_p} = (\sum_{n=1}^{\min(M,N)} \lambda_n^p)^{1/p}$ and $||\mathbf{X}||_{S_\infty} = \max(\lambda)$.

The gradient of a function f is Lipschitz continuous with parameter $L > 0$ if $||\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})||_p \leq L||\boldsymbol{x} - \boldsymbol{y}||_q$, where q is the dual norm to p. Functions that are Lipschitz gradient have an upper bound $f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), (\boldsymbol{y} - \boldsymbol{x}) \rangle + \frac{L}{2}||\boldsymbol{x} - \boldsymbol{y}||_q^2$. When this is the ℓ_2 norm, minimizing the upper surrogate leads to the gradient method, which the stochastic gradient method mirrors. The upper bounds for the RBM are given in Section 3.

2.2 Restricted Boltzmann Machines

The RBM is a two-layer binary Markov Random Field, where the observed binary stochastic visible units $\boldsymbol{v} \in \{0,1\}^M$ have pairwise connections to the binary stochastic hidden units $\boldsymbol{h} \in \{0,1\}^J$. There are no pairwise connections within the visible units, nor within hidden units. The negative energy for a state $\{\boldsymbol{v}, \boldsymbol{h}\}$ is

$$-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \boldsymbol{v}^T \boldsymbol{c} + \boldsymbol{h}^T \boldsymbol{b} + \boldsymbol{v}^T \mathbf{W} \boldsymbol{h} \qquad (1)$$

with $\boldsymbol{\theta} = \{\boldsymbol{c}, \boldsymbol{b}, \mathbf{W}\}$, $\boldsymbol{c} \in \mathbb{R}^M$, $\boldsymbol{b} \in \mathbb{R}^J$, and $\mathbf{W} \in \mathbb{R}^{M \times J}$. The joint probability of the model is defined as $p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h}) = \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}))/Z_{\boldsymbol{\theta}}$,where $Z_{\boldsymbol{\theta}}$ is the partition function. The likelihood for an observation $\boldsymbol{v}$ is:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{v}) &= \frac{1}{Z_{\boldsymbol{\theta}}} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})) && (2) \\ Z_{\boldsymbol{\theta}} &= \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})) && (3) \end{aligned}$$

The sum over $\boldsymbol{h}$ denotes the sum over all 2^J binary vectors, and likewise for $\boldsymbol{v}$. The optimization goal is to minimize the negative log-likelihood of the model, defined for N data samples $\{\boldsymbol{v}_n\}_{n=1,\ldots,N}$ as:

$$\arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = \frac{-1}{N} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{v}_n) = f(\boldsymbol{\theta}) - g(\boldsymbol{\theta}) \qquad (4)$$

$$\begin{aligned} f(\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp(\boldsymbol{v}^T \boldsymbol{c} + \boldsymbol{h}^T \boldsymbol{b} + \boldsymbol{v}^T \mathbf{W} \boldsymbol{h}) && (5) \\ g(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} \exp(\boldsymbol{v}_n^T \boldsymbol{c} + \boldsymbol{h}^T \boldsymbol{b} + \boldsymbol{v}_n^T \mathbf{W} \boldsymbol{h}) \end{aligned}$$

The gradients of the objective function are dependent on expectations of the model:

$$\begin{aligned} \nabla_{\mathbf{W}} F(\theta) &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})}[\boldsymbol{v}\boldsymbol{h}^T] - \mathbb{E}_{p_0}[\boldsymbol{v}\boldsymbol{h}^T] && (6) \\ \nabla_{\boldsymbol{b}} F(\theta) &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})}[\boldsymbol{h}] - \mathbb{E}_{p_0}[\boldsymbol{h}] && (7) \\ \nabla_{\boldsymbol{c}} F(\theta) &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})}[\boldsymbol{v}] - \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{v}_n && (8) \end{aligned}$$

where p_0 denotes $\frac{1}{N} \sum_{n=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{v}_n)$. The gradients cannot be directly calculated in all but the smallest problems, so Monte Carlo integration is used to estimate the gradients. Sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})$ is intractable, so Persistent Contrastive Divergence (PCD) sampling schemes [Hinton, 2002, Tieleman and Hinton, 2009] are used to generate approximate samples.

In general, explicit objective function evaluations are intractable, but Annealed Importance Sampling [Salakhutdinov and Murray, 2008] allows tight estimates of the log partition function. Given the estimate on $\log Z_{\boldsymbol{\theta}}$, the objective function is analytic.

2.3 Deep Belief Nets

A Deep Belief Net (DBN) [Hinton et al., 2006] is an $(L+1)$-layer deep model of binary nodes. The bottom layer consists of the visible units $\boldsymbol{v} \in \{0,1\}^M$, and L hidden layers are stacked on top. Each hidden layer is a binary vector, $\boldsymbol{h}^\ell \in \{0,1\}^{J^\ell}$. The top two hidden layers are jointly drawn from a RBM:

$$\begin{aligned} p(\boldsymbol{h}^L, \boldsymbol{h}^{L-1}) &\propto \exp(-E(\boldsymbol{h}^L, \boldsymbol{h}^{L-1})) && (9)\\ -E(\boldsymbol{h}^L, \boldsymbol{h}^{L-1}) &= (\boldsymbol{h}^L)^T \boldsymbol{b}^L + (\boldsymbol{h}^{L-1})^T \boldsymbol{b}^{L-1} \\ &\quad +(\boldsymbol{h}^{L-1})\mathbf{W}^L \boldsymbol{h}^L && (10)\end{aligned}$$

Given $\boldsymbol{h}^{L-1}$, the DBN is a directed generative model with the same form of the Sigmoid Belief Net [Neal, 1992]. The hidden layers, $\boldsymbol{h}^1, \ldots, \boldsymbol{h}^{L-2}$, and the visible units are generated by:

$$\begin{aligned} p(h_j^\ell | \boldsymbol{h}^{\ell+1}) &= \text{Bern}(\sigma(b_j^\ell + \sum_{k=1}^{J^{\ell+1}} W_{jk}^{\ell+1} h_k^{\ell+1})) && (11)\\ p(v_m | \boldsymbol{h}^1) &= \text{Bern}(\sigma(c_m^\ell + \sum_{j=1}^{J^1} W_{mj}^1 h_j^{\ell+1})) && (12)\end{aligned}$$

Learning a Sigmoid Belief Network or a DBN is a challenging problem, and much recent work has explored approximate learning using recognition models [Hinton et al., 1995, Mnih and Gregor, 2014, Gregor et al., 2014]. In [Hinton et al., 2006], it was shown that greedy layer-wise training of RBMs would provide an effective "pre-training" initialization of the DBN.

The pre-training method starts by training $\boldsymbol{v}$ and $\boldsymbol{h}^1$ as an RBM. Then, using samples from the first-layer RBM for $\boldsymbol{h}^1$, $\boldsymbol{h}^1$ and $\boldsymbol{h}^2$ are trained as an RBM. This procedure continues until the last layer is trained. "Fine-tuning" updates using the cost function of the DBN show only minor performance gains [Hinton et al., 2006].

The model is evaluated by using variational evidence lower bounds. Considering a 3-layer DBN, the model likelihood is lower bounded by:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\boldsymbol{v}) \geq\ & \mathbb{E}_q[\log p(\boldsymbol{v}|\boldsymbol{h}^1; \mathbf{W}^1, \boldsymbol{c})] - \mathbb{E}_q[\log q(\boldsymbol{h}^1)] \\ & +\mathbb{E}_q[\log p(\boldsymbol{h}^1; \mathbf{W}^2, \boldsymbol{b}^1, \boldsymbol{b}^2)] \quad (13)\end{aligned}$$

The second hidden layer, $\boldsymbol{h}^2$, is analytically integrated out, and the variational distributions q are set during the greedy layer-wise training. Recent methods have explored methods to learn better variational distributions [Mnih and Gregor, 2014, Gregor et al., 2014] in these models. To evaluate the term $\mathbb{E}_q[p(\boldsymbol{h}^1; \mathbf{W}^2, \boldsymbol{b}^1, \boldsymbol{b}^2)]$, first the log partition function is estimated by Annealed Importance Sampling (AIS) and then $\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{v}))$ is estimated through Monte-Carlo integration. Further details can be found in [Salakhutdinov and Murray, 2008].

3 Learning Restricted Boltzmann Machines

To the best of our knowledge, virtually all RBM training approaches operate in the Euclidean space, which leads to stochastic gradient descent and its variations. Because the gradients are expensive to estimate, exploiting the information in the gradients can dramatically speed up inference.

The non-convex objective function of the RBM is

$$F(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) - g(\boldsymbol{\theta}) \quad (14)$$

where $f(\boldsymbol{\theta})$ comes from the log partition function and $g(\boldsymbol{\theta})$ depends on the data. Both $f(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ are convex functions, so the objective function can be analyzed as the sum of a convex and concave function. The key algorithmic idea is the use of majorization-minimization to update the parameters. We first find an upper bound on the function $F(\boldsymbol{\theta})$, and then steps are taken to minimize this upper bound. The function $f(\boldsymbol{\theta})$ is examined first.

3.1 Upper bounds on the log partition function, $f(\boldsymbol{\theta})$

Lipschitz-gradient functions have a quadratic upper bound with a function-dependent parameter L:

$$f(\boldsymbol{u}) \leq f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{u} - \boldsymbol{v} \rangle + \frac{L}{2}||\boldsymbol{u} - \boldsymbol{v}||^2 \quad (15)$$

If the norm is ℓ_2, this bound is minimized by taking a step in the negative direction of the gradient, with $\boldsymbol{u} = \boldsymbol{v} - L^{-1}\nabla f(\boldsymbol{v})$. Our function $f(\boldsymbol{\theta})$ is not dependent on the ℓ_2 and Frobenius norms, but instead is bound by the ℓ_∞ and the S_∞ norms.

The analysis in this paper hinges on the form of this inequality for functions of the form $\log \sum_j \omega_j \exp(u_j)$:

Theorem 1. *Define a function* $lse_{\boldsymbol{\omega}}(\boldsymbol{u}) = \log \sum_{j=1}^J \omega_j \exp(u_j)$, *then this function has an upper bound independent of* $\boldsymbol{\omega}$*:*

$$lse_{\boldsymbol{\omega}}(\boldsymbol{v}) \leq lse_\omega(\boldsymbol{u}) + \langle \nabla lse_\omega(\boldsymbol{u}), \boldsymbol{v} - \boldsymbol{u} \rangle + \frac{1}{2}||\boldsymbol{v} - \boldsymbol{u}||_\infty^2 \quad (16)$$

Proof. See Supplemental Section A. □

Critically, this inequality is independent of the constant vector $\boldsymbol{\omega}$.

The bound given in Theorem 1 is different than the ℓ_2 bound on the *lse* function in the literature [Böhning, 1992], which in the RBM case is

$$lse_{\boldsymbol{\omega}}(\boldsymbol{v}) \leq lse_\omega(\boldsymbol{u}) + \langle \nabla lse_\omega(\boldsymbol{u}), \boldsymbol{v} - \boldsymbol{u} \rangle + \frac{1}{4}||\boldsymbol{v} - \boldsymbol{u}||_2^2 \quad (17)$$

However, in the RBM the *lse* function has a dimensionality of the number of possible states, which is 2^{M+J}. Using the established ℓ_2 bound instead of the ℓ_∞ bound would lead to exponentially looser estimates in realistic problem sizes, as well as not correctly capturing the geometry. See Supplemental Section C for further details.

Theorem 1 is directly utilized to bound the function $f(\boldsymbol{\theta})$. Let $\boldsymbol{\theta}_k = \{\boldsymbol{b}^k, \boldsymbol{c}^k, \mathbf{W}^k\}$, then:

Theorem 2. *The bias terms have an upper bound*

$$\begin{aligned} f(\{\boldsymbol{b}, \boldsymbol{c}^k, \mathbf{W}^k\}) &\leq f(\boldsymbol{\theta}^k) + \langle \nabla_{\boldsymbol{b}} f(\boldsymbol{\theta}^k), \boldsymbol{b} - \boldsymbol{b}^k \rangle \\ &\quad + \frac{J}{2} ||\boldsymbol{b} - \boldsymbol{b}^k||_\infty^2 \quad (18) \\ f(\{\boldsymbol{b}^k, \boldsymbol{c}, \mathbf{W}^k\}) &\leq f(\boldsymbol{\theta}^k) + \langle \nabla_{\boldsymbol{c}} f(\boldsymbol{\theta}^k), \boldsymbol{c} - \boldsymbol{c}^k \rangle \\ &\quad + \frac{M}{2} ||\boldsymbol{c} - \boldsymbol{c}^k||_\infty^2 \quad (19) \end{aligned}$$

Proof. See Supplemental Section A. □

Given the gradient with respect to $\boldsymbol{b}$, Equation 18 is bound independently of $\boldsymbol{c}^k$ and $\mathbf{W}^k$, but the gradient is dependent on the current values of these parameters.

Next, the majorization function on $\mathbf{W}$ for f is bound by the Shatten-∞ norm:

Theorem 3. $f(\{\boldsymbol{b}^k, \boldsymbol{c}^k, \mathbf{W}\})$ *has an upper bound:*

$$\begin{aligned} f(\{\boldsymbol{b}^k, \boldsymbol{c}^k, \mathbf{W}\}) \leq \quad & f(\boldsymbol{\theta}^k) + tr((\mathbf{W} - \mathbf{W}^k)^T \nabla_{\mathbf{W}} f(\boldsymbol{\theta}^k)) \\ & + \tfrac{MJ}{2} ||\mathbf{W} - \mathbf{W}^k||_{S^\infty}^2 \quad (20) \end{aligned}$$

Proof. See Supplemental Section A. □

3.2 Majorization bounds on $F(\boldsymbol{\theta})$

Because of the convexity of $g(\boldsymbol{\theta})$, the bounds on $g(\boldsymbol{\theta})$ are:

$$\begin{aligned} g(\{\boldsymbol{b}, \boldsymbol{c}^k, \mathbf{W}^k\}) &\geq g(\boldsymbol{\theta}^k) + \langle \nabla_{\boldsymbol{b}} g(\boldsymbol{\theta}^k), \boldsymbol{b} - \boldsymbol{b}^k \rangle \quad (21) \\ g(\{\boldsymbol{b}^k, \boldsymbol{c}, \mathbf{W}^k\}) &\geq g(\boldsymbol{\theta}^k) + \langle \nabla_{\boldsymbol{c}} g(\boldsymbol{\theta}^k), \boldsymbol{c} - \boldsymbol{c}^k \rangle \quad (22) \\ g(\{\boldsymbol{b}^k, \boldsymbol{c}^k, \mathbf{W}\}) &\geq g(\boldsymbol{\theta}^k) + \langle \nabla_{\mathbf{W}} g(\boldsymbol{\theta}^k), \mathbf{W} - \mathbf{W}^k \rangle \end{aligned}$$

Combining these inequalities with the inequalities on $f(\boldsymbol{\theta})$ for the bias terms $\boldsymbol{b}$ and $\boldsymbol{c}$ gives

$$\begin{aligned} F(\{\boldsymbol{b}, \boldsymbol{c}^k, \mathbf{W}^k\}) &\leq F(\boldsymbol{\theta}^k) + \langle \nabla_{\boldsymbol{b}} F(\boldsymbol{\theta}^k), \boldsymbol{b} - \boldsymbol{b}^k \rangle \\ &\quad + \frac{J}{2} ||\boldsymbol{b} - \boldsymbol{b}^k||_\infty^2 \quad (23) \\ F(\{\boldsymbol{b}^k, \boldsymbol{c}, \mathbf{W}^k\}) &\leq F(\boldsymbol{\theta}^k) + \langle \nabla_{\boldsymbol{c}} F(\boldsymbol{\theta}^k), \boldsymbol{c} - \boldsymbol{c}^k \rangle \\ &\quad + \frac{M}{2} ||\boldsymbol{c} - \boldsymbol{c}^k||_\infty^2 \quad (24) \end{aligned}$$

These majorization bounds for the bias terms on f are *not* minimized in the direction of the gradient. Instead the minimizers are

$$\begin{aligned} \boldsymbol{b}^* = \quad & \boldsymbol{b}^k - \tfrac{1}{J} ||\nabla_{\boldsymbol{b}} f(\boldsymbol{\theta}^k)||_1 \times \text{sign}(\nabla_{\boldsymbol{b}} F(\boldsymbol{\theta}^k)) \quad (25) \\ \boldsymbol{c}^* = \quad & \boldsymbol{c}^k - \tfrac{1}{M} ||\nabla_{\boldsymbol{c}} f(\boldsymbol{\theta}^k)||_1 \times \text{sign}(\nabla_{\boldsymbol{c}} F(\boldsymbol{\theta}^k)) \quad (26) \end{aligned}$$

Derivations for Equations 25 and 26 can be found in Supplemental Section B.

The bound on the objective function for $\mathbf{W}$ is:

$$\begin{aligned} F(\{\boldsymbol{b}^k, \boldsymbol{c}^k, \mathbf{W}\}) \quad \leq \quad & F(\boldsymbol{\theta}^k) + \text{tr}(\nabla_{\mathbf{W}} F(\boldsymbol{\theta}^k)(\mathbf{W} - \mathbf{W}^k)) \\ & + \frac{MJ}{2} ||\mathbf{W} - \mathbf{W}^k||_{S^\infty}^2 \quad (27) \end{aligned}$$

Because the term $||\mathbf{W} - \mathbf{W}^k||_{S^\infty}^2$ is dependent on the largest eigenvalue, the minimization of this majorization function is not maximized in the direction of the gradient. Representing the gradient by its singular value decomposition $\nabla_{\mathbf{W}} F(\boldsymbol{\theta}) = \sum_{k=1}^{\min(M,J)} \lambda_k \boldsymbol{a}_k \boldsymbol{b}_k^T$, this bound is minimized at:

$$\mathbf{W}^* = \mathbf{W} - \frac{||\boldsymbol{\lambda}||_1}{MJ} \sum_{k=1}^{\text{rank}(\nabla_{\mathbf{W}} F(\boldsymbol{\theta}^k))} \boldsymbol{a}_k \boldsymbol{b}_k^T \quad (28)$$

The derivation for Equation 28 can be found in Supplemental Section B.

3.3 Stochastic Spectral Descent

In Section 3.2 the majorization bounds on the objective function are given for each set of parameters $\boldsymbol{b}$, $\boldsymbol{c}$, and $\mathbf{W}$. In a gradient descent scheme, the gradient would be exactly calculated between each of the updates. In an RBM, the cost of estimating the gradient is high and mini-batches are necessary for large data to reasonably estimate the gradients. To set the step-size on $\mathbf{W}$ on the SSD, we can simply use $1/MJ$, which can exhibit slow convergence. However, the presence of local strong convexity in $g(\boldsymbol{\theta})$ suggests that we can choose a higher-step-size, which might potentially depend on the minibatch size. Overall, with some experimentation, we found the following setting robust in many of the scenarios we tested, the step-size on $\mathbf{W}$ is set to $\frac{1}{\sqrt{MJ}}$, and $\boldsymbol{b}$ and $\boldsymbol{c}$ are set to $\frac{1}{M}$ and $\frac{1}{J}$ respectively. This procedure is summarized in Algorithm 1.

Algorithm 1 RBM Stochastic Spectral Descent

```
1: Inputs: $\boldsymbol{v}_{1,...,N}$, $J$, $\alpha$
2: Initialize: $\boldsymbol{b} = \mathbf{0}$, $\boldsymbol{c} = \mathbf{0}$, $W_{mj} \sim \mathcal{N}(0, .1)$
3: for i do=1,...
4:     Sample a minibatch $v_{1,...,B}$
5:     $[\mathbf{dW}, \mathbf{db}, \mathbf{dc}]$=CDGradEstimate($\boldsymbol{v}_{1,...,B}, \boldsymbol{\theta}$)
6:     $\boldsymbol{b} = \boldsymbol{b} - (\text{sum}(\text{abs}(\mathbf{db}))/J)\text{sign}(\mathbf{db})$
7:     $\boldsymbol{c} = \boldsymbol{c} - (\text{sum}(\text{abs}(\mathbf{dc}))/M)\text{sign}(\mathbf{dc})$
8:     $[\mathbf{A}, \boldsymbol{\lambda}, \mathbf{B}] = \text{svd}(\mathbf{dW})$
9:     $\mathbf{W} = \mathbf{W} - \alpha(||\boldsymbol{\lambda}||_1/MJ)\mathbf{A}\mathbf{B}^T$
10: end for
```

3.4 Discussion of potential issues

The proposed algorithm exploits the natural geometry of the logsumexp function and how it relates to the RBM cost function. There are a few concerns about using these projections instead of using the gradients directly, including the additional computational cost of the projection, the inexact nature of the estimating the gradient, and the effect of small mini-batches on the ability to estimate the eigenvectors of the gradient. These concerns are addressed below.

Computational Cost of the Spectral Norm
The cost of computing the singular value decomposition is in general not trivial, and requires $\mathcal{O}(MJ\min(M,J))$ time to calculate. However, in the RBM, the cost of estimating the gradient is expensive. The cost of a single Gibbs sample from $\boldsymbol{v}^{(k-1)} \rightarrow \boldsymbol{v}^{k}$, the key operation of contrastive divergence, costs $\mathcal{O}(MJ)$. A mini-batch of size N_{batch} and Contrastive Divergence order C (the number of Gibbs iterations) costs $\mathcal{O}(MJN_{batch}C)$. Previous work has shown that a large Contrastive Divergence order is necessary to fit the generative model, with $C \geq 25$ [Salakhutdinov and Murray, 2008]. In our experiments, the computational cost of calculating the SVD is essentially free compared to the computational cost of estimating the gradient.

Issues with Noisy Gradient Estimation
The method for estimating the gradient is to take a mini-batch of data and estimate the gradient on $g(\boldsymbol{\theta})$ and estimate the gradient on $f(\boldsymbol{\theta})$ with samples via persistent Contrastive Divergence. From SVD theory, if the small batch sizes cause additive white Gaussian noise, the expectation of the eigenvalues is biased upwards. However, the expectations of the subspaces does not change. In practice, these issues don't seem to affect the performance.

Effect of mini-batches smaller than $\min(M,J)$
In many cases, it's of interest to use very small mini-batches of size $r << \min(M,J)$. In this case $\text{rank}(\nabla_{\mathbf{W}} F(\theta)) \leq 2r$, which gives a low rank matrix. Learning is noisier but still provides competitive results, as shown in Section 5.1.

4 Related Work

Other optimization procedures have attempted to adapt to geometry. ADAgrad [Duchi et al., 2010] provides a element-wise step size scheme that dynamically adapts to the geometry of the data with theoretical guarantees on the regret bound. In machine learning problems with very noisy gradients, ADAgrad may reduce the step-size too quickly, so ADAdelta [Zeiler, 2012] and RMSprop [Tieleman and LeCun, 2012] were introduced to provide schemes that adapt to the geometry of the problem with alternative step-size reduction. These methods have shown empirically good performance in autoencoder models, and here we will show results on both ADAgrad and RMSprop in RBMs. These methods all attempt to learn the geometry, where the proposed algorithm is given the geometry of the problem.

Recent work has also focused on the SGD step-size. [Schulz et al., 2010] explored the convergence of RBM when a constant step size is used, as well as model divergence when using smaller numbers of Contrastive Divergence steps. [Schaul et al., 2012] explored automatic step-size selection for deep models, but did not specifically consider RBMs. [Yuille, 2004] explored when the CD based descent would converge.

Dropout [Srivastava et al., 2014] exploits model averaging to improve the gradient descent algorithms by preventing co-adaptation. Empirically, using a dropout RBM models decreases the number of dead units as compared to traditional RBMs. However, Dropout RBMs differ from traditional RBMs due to the penalization scheme on the model, and does not give maximum likelihood estimates of the model, so is not compared to here.

5 Experiments

Experiments were performed on simulated data and the MNIST dataset. Estimation of the model likelihood $\frac{1}{N}\sum_n \log p(\boldsymbol{v}_n) = -F(\boldsymbol{\theta})$ is calculated through AIS [Salakhutdinov and Murray, 2008] with 10,000 temperature scales evenly spaced from 0 to 1 and 100 particles. The base distribution in AIS was set to independent binary draws at the mean of the observations. The performance of SSD was compared to stochastic gradient descent (SGD), ADAgrad [Duchi et al., 2010], and RMSprop [Tieleman and LeCun, 2012]. The step-size in ADAgrad was set to 0.1, and the step size in RMSprop was set to 0.1 with a decay parameter of 0.999, which were optimal at both $J = 25$ and $J = 100$ in the MNIST data set. We compared to SGD with constant step sizes set at 0.1 and 0.01. The algorithms all shared the same initialization for each problem with the bias terms set to zero vectors, and the pairwise weights $\mathbf{W}$ initialized to random Gaussian draws with variance 0.01. The CD order was set to 25; this was chosen based on the results of [Salakhutdinov and Murray, 2008], that empirically showed a higher number was necessary for good model estimates. The batch size was set to $2J$ unless otherwise stated.

5.1 Synthetic Data

Experiments were run on synthetic data sets with $M = 100$ and $J = 25$. $N = 5000$ data samples were used. The observations were approximately generated from the model by first drawing random binary vectors, and

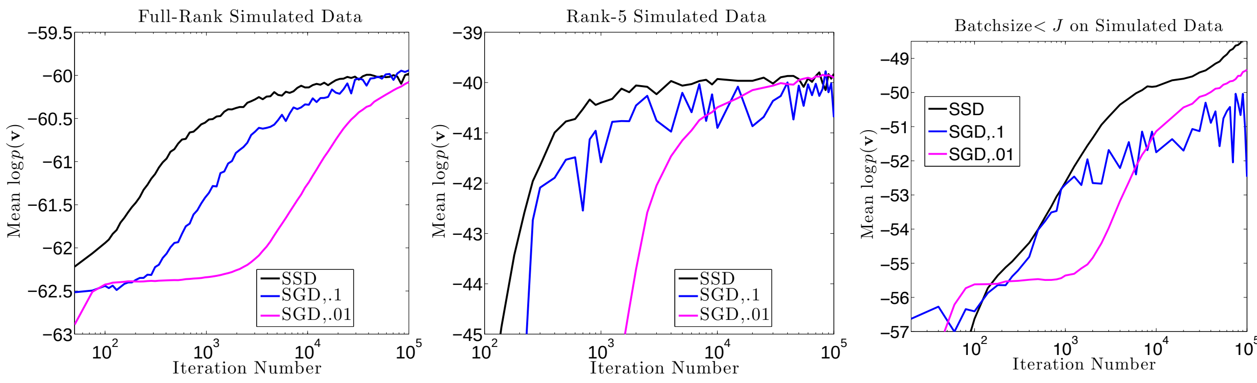

Figure 1: Results on simulated $M = 100$, $J = 25$ datasets, all plots shown in the *log* scale on iterations (Left) $\mathbf{W}$ set to a full matrix. (Middle) $\mathbf{W}$ set to a matrix with rank 5. All algorithms converge to the same approximate objective value, but SSD gets to that level in fewer iterations. (Right) $\mathbf{W}$ set to a full matrix, but the batch size is set to 10.

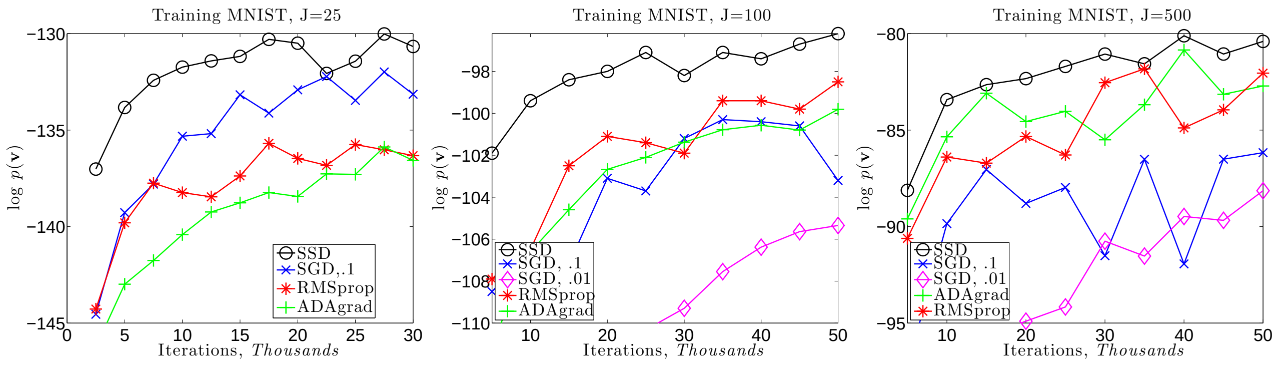

Figure 2: Learning curves for a single-layer RBM on MNIST showing the mean $\log p(\boldsymbol{v})$ versus the number of iterations. (Left) J=25 (Middle) J=100 (Right) J=500. SSD shows improved performance over competing algorithms across all levels of J for this dataset.

then running 10,000 Gibbs iterations using the true parameters.

In the first synthetic data example, $\mathbf{W}$ was set to random Gaussian weights with variance 0.5 to mimic a full rank matrix, and the biases were set to zeros. Figure 1 (left) shows the mean $\log p(\boldsymbol{v})$ for all of the data sample versus the log of the number of iterations. Here, SSD reaches the saturating value with a full order of magnitude fewer iterations. Larger step-sizes on SGD were unstable.

The next simulated dataset was generated from a low-rank matrix on $\mathbf{W}$. This is meant to address concerns on the ability of the algorithm to fit low-rank data when the steps on $\mathbf{W}$ are full-rank. The pairwise matrix $\mathbf{W}$ was set to $\mathbf{AB}^T$ with $\mathbf{A} \in \mathbb{R}^{M\times R}$ and $\mathbf{B} \in \mathbb{R}^{J\times R}$, with $R = 5$. $\mathbf{A}$ was generated from random Gaussians with variance 1, and $\mathbf{B}$ was generated from random Gaussian with variance 0.5. The biases were set to zero. Here, the performance from SSD is still better than SGD, although the performance margin is smaller.

Finally, to examine the effects of small batch sizes, a random full-weight matrix was generated as in the first example. In this example, SSD still performs quite well. The improvement in performance is less than the cases where the gradient is better estimated, however.

5.2 MNIST

The MNIST digit dataset contains 60,000 training and 10,000 test images of handwritten digits (0 to 9) of size 28 × 28 pixels. Each image was vectorized and binarized as in [Salakhutdinov and Murray, 2008].

5.2.1 RBM

The learning curves for different levels of J are first examined. In Figure 2(left), the learning curves for $J = 25$ are shown. Here, SSD gives dominant performance and has a fairly smooth curve. ADAgrad and RMSprop are surprisingly uncompetetive on this problem size; further tuning parameters on these algortihms did not result in improved performance. After 30,000 iterations, SGD has yet to approach the per-

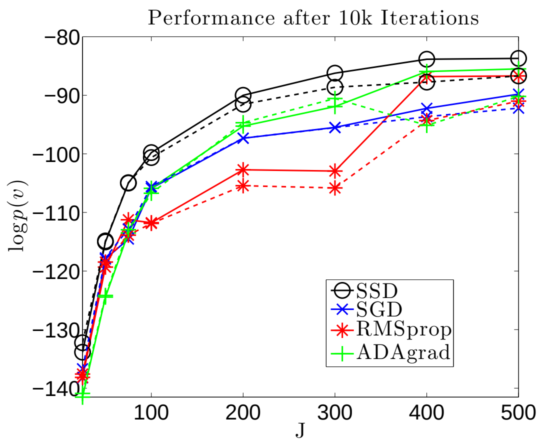

Figure 3: Train (solid line) and test (dashed line) model likelihood after 10,000 iterations for different values of J with different optimization methods.

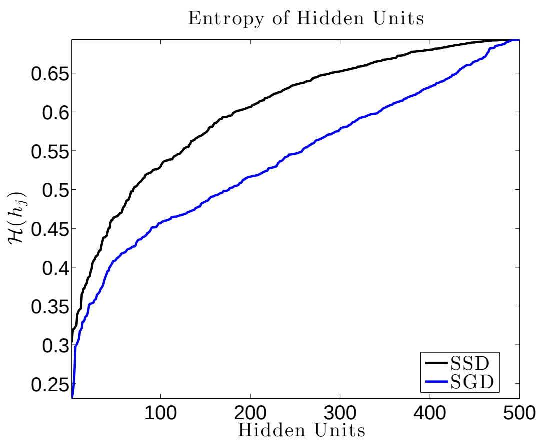

Figure 4: Per-unit empirical entropy for a $J = 500$ RBM training on the MNIST data set after 50,000 iterations. SSD shows higher entropies

formance given by SSD, and SSD is able to appoach this level of performance quickly. While the focus of this paper is on RBM training, it is interesting to examine the testing performance as well. The mean test log likelihoods were -129.76 for SSD, -131.73 for SGD, -137.02 for RMSprop, and -135.97 for ADAgrad after 30,000 iterations. The MNIST dataset is unusual in this case because the test set has a bigger likelihood than the training dataset. This is consistent with the literature on networks of this size [Salakhutdinov and Murray, 2008]. The learning curves on these algorithms are quite noisy, but this is consistent with observations of long runs of SGD in RBMs [Schulz et al., 2010].

In Figure 2(middle), the learning curves for $J = 100$ are shown. SSD once again shows dominant performance. After 30,000 iterations, the test performance is -97.11 for SSD, -101.78 for SGD, -99.82 for RMSprop, and -100.10 for ADAgrad.

In Figure 2(right), the learning curves for $J = 500$ are shown. Here, both ADAgrad and RMSprop show significant improvement over SGD, but still lag behind the performance of SSD. The test performance is -86.77 for SSD, -89.61 for RMSprop, -90.29 for SGD with parameter 0.1, and -90.36 for SGD with parameter 0.01. In fact, after 50,000 iterations, SGD is at roughly the same level from SSD at 5,000. Note that this is marginally below the value reported for the test set in [Salakhutdinov and Murray, 2008], but there a penalized likelihood scheme was used instead of maximum likelihood, giving less overfitting. Matching the penalized likelihood scheme with SSD gives a test performance of -85.65 with a training likelihood of -82.60 after 50,000 iterations.

To investigate the effect of using this algorithm on a given number of iterations, the number of iterations was set to 10,000 and run for J values varying from 25 to 500. The results of these simulations are shown in Figure 3. Because the computational costs of running the algorithm are similar for all algorithms (in our code, a single iteration took 3% longer per iteration in SSD than SGD for J=100), this provides a metric of how well each algorithm will do in a given amount of time.

Figure 3 shows the training mean log probability after 10,000 iterations for all algorithms considered. Its interesting to note that both RMSprop and ADAgrad seem to be improving in comparative performance as the problem size increases, but SSD is provides the best performance over the experimental range. In fact, after 10,000 iterations, SSD has already reached the best reported training likelihood in the literature.

The usage of units is also considered. It was shown in (Srivastava et al. [2014]) that using a Dropout-RBM results in fewer "dead" units, which are units that are nearly completely on or nearly completely off. Here, the per-unit binary entropy of the empirical distribution of each unit is used to measure how much a unit is effectively used. This result for a $J = 500$ RBM is shown in Figure 4, which shows that SSD hidden units have higher entropies than SGD hidden units. Thus, each unit seems to be utilized to a higher extent by training with SSD instead of SGD.

5.2.2 DBN

Here a 3-layer Deep Belief Net is considered. To examine how the different learning speeds of optimization procedures propagate on the deep model, the first layer was trained with the different algorithms for 1000, 5000, 10000, 20000, and 30000 gradient steps for $J = 100$ hidden nodes. A DBN with 200 nodes in the

L1 Iters	SSD	ADAgrad	RMSprop	SGD
1k, Train	-118.3	-125.5	-124.9	-133.9
2k, Train	-106.2	-111.7	-110.0	-116.8
10k, Train	-104.7	-107.7	-106.2	-113.9
20k, Train	-101.6	-103.2	-102.0	-109.3
30k, Train	-101.0	-102.3	-101.6	-108.0
1k, Test	-117.5	-124.4	-124.1	-132.4
5k, Test	-105.6	-110.7	-109.8	-115.7
10k, Test	-103.5	-106.4	-105.2	-113.2
20k, Test	-102.3	-104.6	-103.0	-110.2
30k, Test	-101.7	-103.4	-102.7	-108.7

Table 1: A DBN with 100 hidden nodes in the first layer and 200 hidden nodes in the second layer. The left column denotes the number of training samples on the first layer, and then the corresponding lower bounds on the log likelihood given for both the training and the testing set.

second hidden layer was learn based on this output, and the lower bound was evaluated. 10,000 iterations were used in the second layer. Because the second layer network was much smaller, this was an effective number of samples to converge and was much quicker. The lower bounds are shown in Table 1. SSD shows a performance improvement at every number of gradient steps, often by several nats. All algorithms that use geometry outperform SGD here.

6 Discussion

We have introduced a novel descent method for RBMs, Stochastic Spectral Descent, which utilizes the natural geometry of the RBM cost function by operating in the ℓ_∞ and S_∞ norms. Empirical results suggest that this is a good general purpose algorithm for RBMs. To the best of our knowledge, the bounds presented in this paper are novel, and the algorithm gives state-of-the-art learning performance. Further improvements may be obtained by using better gradient estimators, either by using tempering schemes [Salakhutdinov, 2010, Cho et al., 2010] or the Enhanced Gradient [Cho et al., 2013]. Future work will including extending this analysis directly to deep models, as well as related models, including the Sigmoid Belief Net.

Acknowledgements

David Carlson and Lawrence Carin are supported in part by ARO, DARPA, DOE, NGA and ONR. Volkan Cevher's work is supported in part by the European Commission under grants MIRG-268398 and ERC Future Proof and by the Swiss Science Foundation under grants SNF 200021-132548, SNF 200021- 146750, and SNF CRSII2-147633.

References

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *arXiv 1206.5533*, June 2012.

D. Böhning. Multinomial logistic regression algorithm. *Ann. Inst. Statist. Math*, 1992.

K. Cho, T. Raiko, and A. Ilin. Parallel tempering is efficient for learning restricted Boltzmann machines. *IJCNN*, July 2010.

K. Cho, T. Raiko, and A. Ilin. Enhanced Gradient for Training Restricted Botlzmann Machines. *Neural Computation*, 2013.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 2010.

K. Gregor, C. Blundell, A. Mnih, and D. Wierstra. Deep AutoRegressive Networks. *ICML*, 2014.

G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

G. E. Hinton, P. Dayan, D. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 1995.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comp.*, 2006.

A. Mnih and K. Gregor. Neural Variational Inference and Learning in Belief Networks. *ICML*, 2014.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, July 1992. ISSN 00043702.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 2012.

R. Salakhutdinov. Learning in Markov Random Fields using Tempered Transitions. *NIPS*, 2010.

R. Salakhutdinov and G. Hinton. Deep Boltzmann Machines. *AISTATS*, 2009. ISSN 15324435.

R. Salakhutdinov and I. Murray. On the Quantitative Analysis of Deep Belief Networks. *ICML*, 2008.

T. Schaul, S. Zhang, and Y. LeCun. No More Pesky Learning Rates. *arXiv 1206.1106*, June 2012.

H. Schulz, A. Müller, and S. Behnke. Investigating Convergence of Restricted Boltzmann Machine Learning. *NIPS Workshop on Deep Learning*, 2010.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 2014.

T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. *ICML*, 2009.

T. Tieleman and Y. LeCun. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

A. Yuille. The convergence of contrastive divergences. *NIPS*, 2004.

M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv 1212.5701*, page 6, Dec. 2012.