Stochastic Difference of Convex Algorithm and its Application to Training Deep Boltzmann Machines

Atsushi Nitanda[†‡]
nitanda@msi.co.jp

Taiji Suzuki[†⋆◇]
s-taiji@is.titech.ac.jp

[†] Tokyo Institute of Technology, Tokyo, Japan [‡] NTT DATA Mathematical Systems Inc., Tokyo, Japan
[⋆] PRESTO, Japan Science and Technology Agency, Japan
[◇] Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan

Abstract

Difference of convex functions (DC) programming is an important approach to nonconvex optimization problems because these structures can be encountered in several fields. Effective optimization methods, called DC algorithms, have been developed in deterministic optimization literature. In machine learning, a lot of important learning problems such as the Boltzmann machines (BMs) can be formulated as DC programming. However, there is no DC-like algorithm guaranteed by convergence rate analysis for stochastic problems that are more suitable settings for machine learning tasks. In this paper, we propose a stochastic variant of DC algorithm and give computational complexities to converge to a stationary point under several situations. Moreover, we show our method includes expectation-maximization (EM) and Monte Carlo EM (MCEM) algorithm as special cases on training BMs. In other words, we extend EM/MCEM algorithm to more effective methods from DC viewpoint with theoretical convergence guarantees. Experimental results indicate that our method performs well for training binary restricted Boltzmann machines and deep Boltzmann machines without pre-training.

1 Introduction

There is a strong need to develop better optimization methods for nonconvex problems because many scientific problems are nonconvex. Generally speaking, a nonconvex problem is hard to solve. However, several important problems possess a special structure (quadratic, finite sums, etc.), and it is expected that we can build effective algorithms by making full use of the special structure. In particular, a wide range of problems are reduced to *difference of convex functions* (DC) programming [1] which takes the the following form:

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \;\; f(x) \stackrel{\text{def}}{=} g(x) - h(x), \tag{1}$$

where g and h are differentiable convex functions from $\mathbb{R}^d$ to $\mathbb{R}$.

In fact, DC structures can be encountered in several fields, e.g., in economics, finance, operations research, and biology. In machine learning, multiple kernel learning [2] and feature selection in support vector machines [3] are formulated as DC programs. Moreover, it is shown that: (i) any continuous function over a compact set can be approximated by a DC function by Stone-Weierstrass theorem and DC decomposition of polynomials [4, 5, 6]; (ii) any C^2-function f whose eigenvalues of Hessian are lower bounded can be decomposed as a DC function; there exists a convex function h such that $f = g - h$ is DC, where $g = f + h$.

To solve optimization problem (1), practical methods are variants of DC algorithms (DCAs) [1] that generate a sequence by solving sub-problems that consist of the sum of the convex part g and the linear approximation of the concave part $-h$ at the current iterate. Due to their simplicity, efficiency, and robustness, DCAs have been widely applied to many fields.

Important applications of DC programming are *Boltzmann machines* (BMs) which are energy-based generative models over binary observations and binary hidden units. Restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs) [7] are special forms of BMs. These models are used for unsupervised learning, dimension reduction, feature extraction, and pre-training or initializa-

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

Table 1: Complexities of SPD

	General case	Smooth concave	Polyak-Łojasiewicz
Outer Iteration Complexity	$O(L_g/\epsilon)$	$O(\min\{L_g, L_h\}/\epsilon)$	$O(CL_g \log \frac{1}{\epsilon})$
Total Complexity (general)	$O(L_g/\epsilon^2)$	$O(L_g/\epsilon^2)$	$O\left(\frac{CL_g}{\epsilon} \log \frac{1}{\epsilon}\right)$
Total Complexity (variance growth condition)	$O\left(\frac{L_g(1+\beta)}{\epsilon} \log \frac{1}{\epsilon}\right)$	$O\left(\frac{L_g(1+\beta)}{\epsilon} \log \frac{L_g}{\epsilon L_h}\right)$	$O\left(CL_g(1+\beta)(\log \frac{1}{\epsilon})^2\right)$

tion of multi-layer perceptrons. RBMs and DBMs learning are easier than more general Boltzmann machine learning; however, it is still quite difficult. In fact, in recent years, several studies have exploited optimization methods [7, 8, 9]. The log-likelihood of a BM is the subtraction of two composite functions of linear mapping and a log-sum-exp function. That is, BM learning is DC programming. However, there is still no DC-like algorithm with any convergence rate analysis to solve stochastic problems that are more suitable settings for training BMs.

In this paper, we propose a *stochastic proximal DC algorithm (SPD)*. Our method works effectively not only under a deterministic setting but also under a stochastic setting, where only stochastic gradients are available for the convex part and sometimes for the concave part. Optimization methods built under this setting can be applied to a wider class of problems, including training BMs. Furthermore, we show that Expectation-maximization (EM) and Monte Carlo EM (MCEM) algorithms, which are heavily used for latent variable models, are recognized as special cases of our SPD algorithm, and our algorithm is even more effective than these algorithms.

In addition, we give convergence complexities: the number of iterations of SPD to obtain an ϵ-accurate solution in expectation (i.e., $\mathbb{E}\|\nabla f(x)\|_2^2 < \epsilon$) under several settings: Lipschitz smoothness of g, h and Polyak-Łojasiewicz condition on objective function f, whose definitions will be described in Section 4.

SPD requires only approximate solutions of sub-problems in expectation. To solve the sub-problems we can employ stochastic optimization methods for convex problems, which is a very active research area. Moreover, since a sub-problem becomes strongly convex, effective stochastic gradient-based methods [10, 11, 12, 13] can be used as underlying solvers to achieve fast convergence, and we give the total complexity analyses that include the complexity of such a method.

Table 1 shows the complexities of our method in general case ($g : L_g$-smooth), smooth concave case ($g, h : L_g, L_h$-smooth), and Polyak-Łojasiewicz case. The middle row of Table 1 lists the total complexities without additional structures for the sub-problem. RSG [14] is a stochastic optimization method for solving Lipschitz smooth non-convex problems, and in this case, we can obtain the same complexity $O(L_g/\epsilon^2)$ as SPD by slight modification of their proof. However, SPD has better practical performance than suggested by the theory because our analyses for the total complexities do not take into account the warm starting for sub-problems solved in SPD repeatedly. An intuition for the practical performance of SPD is described in Section 4.

Moreover, if the convex sub-problems have additional special structures such as 2nd-order derivative, noise condition, finite sums, it is possible to show much better convergence by utilizing this information for the convex optimization method used in the inner loop. Especially, we focus on a *variance growth condition* [13, 15], defined in Section 4 and we show that this condition strictly improves the total complexities as shown in the last row of Table 1.

Related Works

The stochastic majorization-minimization method and the online DCA were proposed in [16] for non-convex problems. Although SPD is also a type of majorization-minimization methods, it differs from their methods in several respects. The surrogate function used in [16] is more stochastic and is quadratically approximated, and can be solved exactly. On the other hand, the convex part is not approximated in SPD and it is relatively difficult to solve our surrogate function exactly; SPD only requires an approximation to the solution in expectation. Moreover, although they gave convergence analyses, convergence rates for the non-convex problem were not provided.

The method proposed in [17] for deterministic nonconvex problems is one of the methods which should be compared with our method. Applying their analysis to our problem, the complexity to obtain a *gradient mapping* of norm ϵ is $O(L_h/\epsilon)$. However, the norm of a gradient and the norm of a gradient mapping cannot be directly compared. Furthermore, while the coefficient of their order is always affected by L_h, our method is free from it when $L_h > L_g$.

2 DC Algorithm

DCAs are optimization algorithms for solving DC problems. To obtain the next iterate that linearly approximates the concave part $-h$ at the current iterate x_k and solves the resulting convex minimization problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}^d}\{g(x) - (h(x_k) + \langle \nabla h(x_k), x - x_k\rangle)\} \\ &\sim \min_{x\in\mathbb{R}^d}\{g(x) - \langle \nabla h(x_k), x\rangle\}, \end{aligned} \tag{2}$$

where $\langle , \rangle$ denotes the Euclidean inner product. Several studies have exploited the convergence properties and shown the efficiency of a DCA under the assumption that we can obtain an exact or deterministically approximate solution of the sub-problem (2). However, there is no algorithm with convergence rate analysis for stochastic problems frequently encountered in machine learning. Thus in the next section, we propose a more suitable DCA for such problems.

3 Stochastic Proximal DC Algorithm

In the remainder of this paper, we make the following stochastic assumption.

Assumption 1 (Stochastic Assumption)**.** *To solve DC problem (1), an optimization algorithm can use only the stochastic gradients of g and h.*

Note that although there are several problems such that a deterministic gradient of h can be computed and we can use ∇h, we also make stochastic assumption on h to handle some specific problems including general BMs. Here, we propose SPD, which is more suitable for this assumption. Let H_k denote the $d \times d$ positive definite matrix and $\| \cdot \|_{H_k}$ denote the Mahalanobis norm defined by H_k, i.e., for $v \in \mathbb{R}^d$, $\|v\|_{H_k} = \sqrt{\langle v, H_k v \rangle}$. Let $v_h(x)$ denote an unbiased estimator of $\nabla h(x)$ and σ_h^2 be an upper bound on the variance of v_h; $\mathbb{E}[v_h(x)] = \nabla h(x)$, $\mathbb{E}[\|v_h(x) - \nabla h(x)\|_2^2] \leq \sigma_h^2$. Let x_k be the current iterate. To obtain the next iterate x_{k+1}, SPD solves the following proximal sub-problem inexactly by a stochastic method:

$$SP(k): \min_{x \in \mathbb{R}^d} \{\phi_k(x) \stackrel{\mathrm{def}}{=} g(x) + \frac{1}{2}\|x - x_k\|_{H_k}^2 - (h(x_k) + \langle v_h(x_k), x - x_k \rangle)\}. \quad (3)$$

The difference between sub-problems (2) and (3) is that the latter problem is a stochastic approximation and includes the proximal term $\frac{1}{2}\|x - x_k\|_{H_k}^2$, which forces the solution to stay close to x_k with respect to the norm $\| \cdot \|_{H_k}$. Practical choices for the metric H_k and our motivations are described in the next subsection. Since it is impractical to obtain an exact or deterministic approximation to the solution, we employ the following condition on expectation of a solution of the sub-problem:

$$\mathbb{E}[\phi_k(x_{k+1}) | \mathcal{F}_k] \leq \phi_k^* + \delta, \quad (4)$$

where $\mathcal{F}_k$ is the filtration for all information up to iterate x_k, ϕ_k^* is the optimal value of *SP(k)*, and $\delta > 0$. For many stochastic algorithms, e.g., stochastic gradient descent, the global convergence property in expectation is shown for convex problems. Therefore, we can use such algorithms as an underlying solver of SPD. Note that we can warm start to solve sub-problems, i.e., by running a stochastic algorithm from a previous solution x_k with sufficiently small learning rates, it is not particularly difficult to satisfy the above condition empirically. In Section 4, we demonstrate how condition (4) guarantees the convergence of SPD with better convergence rates to obtain an ϵ-accurate solution in expectation. Here, we briefly give a connection between SPD and mirror descent method. Let $x_{k+1}^* = \arg\min \phi_k(x), \psi_k(x) \stackrel{\mathrm{def}}{=} g(x) + \frac{1}{2}\|x\|_{H_k}^2$, and assume $v_h(x_k) = \nabla h(x_k)$, then we have

$$\nabla f(x_k) = \nabla \psi_k(x_k) - \nabla \psi_k(x_{k+1}^*). \quad (5)$$

This equation means SPD can also be interpreted as an inexact variant of a stochastic mirror descent method using distance generating functions ψ_k for DC programming.

SPD runs for R iterations, where R is chosen uniformly at random from $\{1, 2, \ldots, M\}$ for $M \in \mathbb{Z}_+$. This is a standard technique for nonconvex analysis [14]. SPD is described in Algorithm 1.

Algorithm 1 SPD (Stochastic proximal DC algorithm)

Input: initial point x_1, the maximum number of iterations M, underlying solver $\mathcal{A}$ for solving $SP(k)$, the number of iterations T for $\mathcal{A}$

Randomly pick up $R \in \{1, 2, \ldots, M\}$
for $k = 1$ **to** $R - 1$ **do**
  Update the metric H_k
  Compute stochastic approximation $v_h(x_k)$ of $\nabla h(x_k)$
  $x_{k+1} \leftarrow$ Solve *SP(k)* by running $\mathcal{A}$ for T iterations
end for
Return x_R

3.1 Metrics

There are two aims for including the proximal term $\frac{1}{2}\|x - x_k\|_{H_k}^2$ of ϕ_k in sub-problems. The first is to keep the next iterate x_{k+1} in a neighborhood of the current x_k where the linear approximation of the concave part $-h$ is sufficiently accurate. In the gradient-based optimization literature, it is well studied theoretically and empirically that the proximity induced by an appropriate metric at each iteration improves the convergence behavior, e.g., Natural Gradient [18] and AdaGrad [19]. The second is to enhance the effect of strong convexity, which makes the sub-problem *SP(k)* better conditioned and easier to solve.

Next, we give practical choices for the metric H_k. The first choice is a scalar matrix, i.e., $H_k = \mu I_d$, $\mu > 0$. As will be discussed in Section 4, this choice with $\mu = L_g$ or L_h, where L_g, L_h are smoothness parameters, gives a better convergence complexity according to our analysis. Second, when the concave part $-h$ is twice differentiable, we propose a diagonal approximation to the Hessian of h [20]. In other words, we define H_k as follows:

$$H_k \leftarrow \left|\mathrm{diag}\left(\nabla^2 h(x_k)\right)\right| + \mu I_d, \quad (6)$$

where the absolute value operator $|\cdot|$ is applied elementwise to the diagonal of the Hessian and μ is a positive value

that guarantees sufficiently strong convexity to improve the conditioning of the curvature of h. This metric makes the update take large steps in the direction of low curvature compared to that of highly curved directions.

3.2 AdaSPD

Here, we derive a specific form of the SPD described by Algorithm 1. For a metric H_k, we use a scalar matrix or the diagonal Hessian (6) as in the previous subsection. For an underlying solver, due to the simplicity of implementation and better empirical performance, we adopt AdaGrad using the proximal term $\frac{1}{2}\|x - x_k\|^2_{H_k}$ as the regularization in its update. Let $y_{k,t}$ and $v_{k,t}$ $(t = 1, 2, \ldots)$ denote an inner iterate and a stochastic gradient of g at $y_{k,t}$, respectively, in outer iteration k. To adapt the step size to the geometry of the objective function, AdaGrad computes diagonal matrix $D_{k,t}$ as follows:

$$D_{k,t} \leftarrow \sqrt{\lambda I_d + \mathrm{diag}(\textstyle\sum_{i=1}^{t} s_{k,i} s_{k,i}^\top)},$$

where λ is a damping parameter for numerical stability and $s_{k,i}$ denotes $v_{k,i} - v_h(x_k)$. To obtain the next inner iterate $y_{k,t+1}$, we solve the following problem:

$$\arg\min_{y \in \mathbb{R}^d} \left\{ \langle s_{k,t}, y \rangle + \frac{1}{2}\|y - x_k\|^2_{H_k} + \frac{1}{2\eta}\|y - y_{k,t}\|^2_{D_{k,t}} \right\},$$

where η denotes the learning rate. Note that $D_{k,t}$ can be updated successively and $y_{k,t+1}$ can be computed in closed form. The algorithm *AdaSPD* is described in Algorithm 2.

Algorithm 2 AdaSPD

Input: initial point x_1, the maximum number of iterations M, (lower) scale μ of a metric H_k, the number of iterations T for the inner loop, damping parameter λ of $D_{k,t}$, learning rate $\eta > 0$, suffix averaging parameter $\alpha \in (0, 1)$ (assuming αT is an integer)

Randomly pick a $R \in \{1, 2, \ldots, M\}$
for $k = 1$ **to** $R - 1$ **do**
  scalar matrix option:
    $H_k \leftarrow \mu I_d$
  Diagonal Hessian option:
    $H_k \leftarrow \left|\mathrm{diag}\left(\nabla^2 h(x_k)\right)\right| + \mu I_d$
  $y_{k,1} \leftarrow x_k$
  $S_{k,0} \leftarrow O$
  for $t = 1$ **to** $T - 1$ **do**
    $v_{k,t} \leftarrow$ a stochastic gradient of g at $y_{k,t}$
    $s_{k,t} \leftarrow v_{k,t} - v_h(x_k)$
    $S_{k,t} \leftarrow S_{k,t-1} + \mathrm{diag}(s_{k,t} s_{k,t}^\top)$
    $D_{k,t} \leftarrow \sqrt{\lambda I_d + S_{k,t}}$
    $y_{k,t+1} \leftarrow (\eta H_k + D_{k,t})^{-1}(\eta H_k x_k + D_{k,t} y_{k,t} - \eta s_{k,t})$
  end for
  $x_{k+1} \leftarrow \dfrac{\sum_{t=(1-\alpha)T+1}^{T} y_{k,t}}{\alpha T}$
end for
Return x_R

4 Analysis

In this section, we give convergence analyses of SPD and complexities to obtain an ϵ-accurate solution in expectation under several situations. Note that all proofs can be found in the supplement. For simplicity, we only consider the scalar matrix $\mu_k I_d$ for H_k. We first give the definition of Lipschitz smoothness needed for analyses.

Definition 1. *A function ϕ is Lipschitz smooth if there exists $L_\phi > 0$ such that $\forall x, \forall y \in \mathbb{R}^d$,*

$$\|\nabla\phi(x) - \nabla\phi(y)\| \leq L_\phi \|x - y\|_2.$$

4.1 General Case

The following proposition shows the expected square norm of the gradient is upper-bounded by the expected reduction of the objective function per iteration up to δ and σ_h^2.

Proposition 1. *Consider Algorithm 1 under stochastic assumption 1. Suppose g is L_g-smooth and the expected condition (4) holds. Then, it follows that for $k = 1, 2, \ldots$*

$$\frac{\mu_k}{4}\mathbb{E}\left[\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k\right] + \frac{\|\nabla f(x_k)\|_2^2}{2(L_g + \mu_k)} \leq \delta + \frac{\sigma_h^2}{\mu_k} + \mathbb{E}[f(x_k) - f(x_{k+1}) | \mathcal{F}_k].$$

Using Proposition 1, we derive a convergence theorem.

Theorem 1. *Make the same assumption as Proposition 1 and assume the optimal value f_* of f is bounded from below. Let $\mu_k = O(L_g)$ and ($\mu_k = \Omega(L_g)$ or $\sigma_h = 0$). Then it follows that*

$$\mathbb{E}[\|\nabla f(x_R)\|_2^2] \leq O\left(L_g\delta + \sigma_h^2 + \frac{L_g(f(x_1) - f_*)}{M}\right).$$

We immediately obtain the following corollary.

Corollary 1. *Suppose the assumptions in Theorem 1 hold and $\sigma_h^2 = O(\epsilon)$. Set $\delta = O(\epsilon/L_g)$. Then, the complexity M to obtain an ϵ-accurate solution in expectation is $O(L_g/\epsilon)$.*

The readers might feel that the assumption $\sigma_h^2 = O(\epsilon)$ in the above corollary is unrealistic because the variance σ_h^2 of the stochastic gradient of h is assumed to be smaller than the solution accuracy ϵ. However, this is reasonable because the total complexity is unchanged even if we spend the same computational cost as that of solving $SP(k)$ to estimate ∇h and the variance σ_h^2 can be made sufficiently small by using a comparable number of samples in the mini-batch.

4.2 Smooth Concave Function

In this subsection, we give the convergence properties for problems having Lipschitz smooth h. To establish a com-

plexity analysis, we slightly modify the algorithm: we choose R, the number of iterations of SPD, uniformly at random from $\{2, 3, \ldots, M+1\}$ instead of $\{1, 2, \ldots, M\}$ as before for $M \in \mathbb{Z}_+$. Then, we have the following proposition.

Proposition 2. *Suppose that g, h are L_g, L_h-smooth, respectively. Then, it follows that*

$$\mathbb{E}\left[\|\nabla f(x_{k+1})\|_2^2 | \mathcal{F}_k\right] \leq 8(L_g + \mu_k)\delta + 4\sigma_h^2 + 4(\mu_k^2 + L_h^2)\mathbb{E}\left[\|x_{k+1} - x_k\|_2^2 | \mathcal{F}_k\right].$$

By combining Proposition 1 and 2, we have the following proposition.

Proposition 3. *Make the same assumption as Proposition 2. Let $\mu_k = O(L_h)$ and $\mu_k = \Omega(L_h)$. Then, it follows that*

$$\mathbb{E}\left[\|\nabla f(x_{k+1})\|_2^2 | \mathcal{F}_k\right] \leq O((L_g + L_h)\delta + \sigma_h^2 + L_h\mathbb{E}\left[f(x_k) - f(x_{k+1}) | \mathcal{F}_k\right]).$$

From Proposition 3, we can obtain the convergence theorem that indicates that as L_h decrease, SPD has better convergence.

Theorem 2. *Make the same assumption as Proposition 3. We assume $L_h = O(L_g)$ and the optimal value f_* of f is bounded from below. Then it follows that*

$$\mathbb{E}[\|\nabla f(x_R)\|_2^2] \leq O\left(L_g\delta + \sigma_h^2 + \frac{L_h(f(x_1) - f_*)}{M}\right).$$

Theorem 2 implies the following complexity result which is better than Corollary 1 because of $L_h = O(L_g)$.

Corollary 2. *Suppose the assumptions in Theorem 2 hold and $\sigma_h^2 = O(\epsilon)$.We set $\delta = O(\epsilon/L_g)$. Then, the complexity M to obtain an ϵ-accurate solution in expectation is $O(L_h/\epsilon)$.*

4.3 Polyak-Łojasiewicz Condition

Here, we show a fast convergence of *Double-loop SPD* described in Algorithm 3 under Polyak-Łojasiewicz condition:

Definition 2. *A function ϕ satisfies Polyak-Łojasiewicz condition, i.e., $\exists C > 0$ such that $\forall x \in \mathbb{R}^d$*

$$\phi(x) - \min \phi \leq C\|\nabla\phi(x)\|_2^2. \tag{7}$$

Note that Algorithm 1 and 3 are essentially the same up to the returned point. Therefore, algorithm remain unchanged in practical implementations.

Let $\delta = O(\epsilon/L_g), M = O(CL_g/2)$ and assume $\sigma_h^2 = O(\epsilon)$. Using Theorem 1 and (7), we can easily show

$$\mathbb{E}[\|\nabla f(y_{t+1})\|_2^2] \leq \epsilon + \frac{\mathbb{E}[\|\nabla f(y_t)\|_2^2]}{2}.$$

Algorithm 3 Double-loop SPD

Input: initial point y_1, the maximum number of outer-iterations N, the options for Algorithm 1 $M, \mathcal{A}, T$

for $t = 1$ **to** $N - 1$ **do**

$y_{t+1} \leftarrow$ Algorithm 1 $(y_t, M, \mathcal{A}, T)$

end for

Return y_N

This recurrence relation immediately implies $\mathbb{E}[\|\nabla f(y_{t+1})\|_2^2] \leq 2\epsilon + (\frac{1}{2})^t\|\nabla f(y_1)\|_2^2$. This mean that if we run Algorithm 3 for $N = O(\log 1/\epsilon)$ outer-iterations, we can obtain an ϵ-accurate solution. Thus, the following theorem holds.

Theorem 3. *Make the same assumption as Theorem 1 and assume Polyak-Łojasiewicz condition holds. Let δ, M and σ_h be as above. Then, the complexity including that of inner SPD to obtain a solution is $O(CL_g \log \frac{1}{\epsilon})$.*

4.4 Total Complexity

We consider the total complexity that includes the complexity of an underlying solver. In recent years, several stochastic optimization methods that can solve the sub-problem *SP(k)* have been developed. Note that the objective function of the sub-problem can be $\mu_k = O(L_g)$ or $O(L_h)$ strongly convex. Let us adopt SGD [10] as an underlying solver. Noting that, SGD can solve the sub-problem with a complexity of $O\left(\frac{1}{\mu_k\delta}\right)$, we obtain the total complexities as shown in the middle row of Table 1. Although the complexity $O(L_g/\epsilon^2)$ is the same as that of RSG method [14] for solving Lipschitz smooth non-convex problems, SPD has better practical performance for several reasons. Firstly, since we can warm start the underlying solver of SPD at the previous solution, it is enough to perform fewer iterations than suggested by the theory. By the strong convexity of the sub-problem, we get $\|\nabla f(x_k)\|_2^2 \geq 2\mu_k(\phi_k(x_k) - \phi_k^*)$, that is, as current iterate x_k is closer to a stationary point, initial objective gap of each sub-problem *SP(k)* also becomes small. Let us assume μ_k are uniformly upper and lower bounded by positive constants. Noting that smoothnesses $L_g + \mu_k$, strong convexities μ_k, and accuracy δ of sub-problems are the same order among all iterations, we find that as the initial objective gap of a sub-problem is smaller, we can easily solve it empirically. Secondly, although a performance of almost stochastic gradient based algorithms is affected by its variance, SPD reduces this effect by fixing an estimate of $\nabla h(x_k)$ in each inner loop.

Moreover, we can show improved convergence complexities by using an additional structure of the convex sub-problems such as a variance growth condition.

4.4.1 Variance Growth Condition

We first introduce the variance growth condition.

Definition 3. *A function ϕ satisfies the variance growth condition if there exist $\alpha, \beta > 0$ such that $\forall x \in \mathbb{R}^d$,*

$$\mathbb{V}[\Phi(x, \xi)] \leq \alpha + \beta \|\nabla \phi(x)\|_2^2,$$

where $\Phi(x, \xi)$ denotes a stochastic gradient of ϕ at x.

This condition can be found in [13, 15] and a stronger condition called gradient growth condition is used in [21, 22]. Note that the variance growth condition with $\alpha = 0$ is used in [15] to establish a convergence analysis of stochastic optimization method for the learning discrete graphical models including RBMs and this condition is controllable by mini-batching of gradient estimators.

Applying Theorem 4.6 in [13] to the sub-problem *SP(k)*, we immediately obtain a complexity to solve it as follows.

Proposition 4. *Let us assume that the objective function ϕ_k of SP(k) satisfies the variance growth condition with $\frac{\alpha}{(1+\beta)\mu_k} \leq \delta$. Then, if we run SGD, with a constant learning rate $\eta = O(\frac{1}{L_g(1+\beta)})$, a δ-accurate solution of SP(k) can be obtained with a complexity of*

$$O\left(\frac{L_g(1+\beta)}{\mu_k} \log \frac{\phi_k(x_k) - \phi_k^*}{\delta}\right).$$

Since $\phi_k(x_k) - \phi_k^* \leq \frac{1}{2\mu_k}\|\nabla f(x_k)\|_2^2$, if $\{\|\nabla f(x_k)\|_2\}_{k=1}^M$ are uniformly bounded and if we apply Proposition 4 with the same μ_k, δ as in Corollary 1, 2, or Theorem 3, we can find that the variance growth condition strictly improves the total complexities as shown in the last row of Table1.

5 Boltzmann Machines

Although we are mainly concerned with RBMs or DBMs rather than BMs, we describe an application to learn BMs because BMs are the general form of these models. The BM is a particular type of Markov random field with visible binary stochastic units $v \in \{0, 1\}^D$ and hidden binary stochastic units $h \in \{0, 1\}^M$. The negative energy of the state $\{v, h\}$ is

$$-E(v, h; \Theta) = v^\top b + h^\top c + v^\top U v + h^\top V h + v^\top W h,$$

where $\Theta = (b, c, U, V, W)$ are the model parameters, i.e., $b \in \mathbb{R}^D, c \in \mathbb{R}^M, U \in \mathbb{R}^{D\times D}, V \in \mathbb{R}^{M\times M}$, and $W \in \mathbb{R}^{D\times M}$. The diagonal elements of U and V are set to zeros. Note that special form of the Boltzmann machine with $U = 0$ and $V = 0$ is nothing else but RBMs. The joint distribution of v, h is defined as proportional to $\exp(-E(v, h; \Theta))$. Thus, the likelihood of BMs is

$$p(v|\Theta) = \frac{1}{Z(\Theta)} \sum_h \exp(-E(v, h; \Theta)),$$
$$Z(\Theta) = \sum_v \sum_h \exp(-E(v, h; \Theta)).$$

Learning the BM is achieved by minimizing the average negative log-likelihood, i.e., for i.i.d. samples $\{v_i\}_{i=1}^N$:

$$\underset{\Theta\in\mathbb{R}^d}{\text{minimize}}\ f(\Theta) = -\frac{1}{N}\sum_{i=1}^N \log p(v_i|\Theta) = g(\Theta) - h(\Theta),$$
$$\text{where } g(\Theta) = \log \sum_v \sum_h \exp(-E(v, h; \Theta)),$$
$$h(\Theta) = \frac{1}{N}\sum_{i=1}^N \log \sum_h \exp(-E(v_i, h; \Theta)).$$

Since a composite function of the convex *log-sum-exp* function and linear mapping is convex, training the BM is DC programming. The gradients of g and h are as follows: for the parameter $\theta \in \Theta$,

$$\nabla_\theta g(\Theta) = -\mathbb{E}_{p(v,h;\Theta)}\left[\nabla_\theta E(v, h; \Theta)\right],$$
$$\nabla_\theta h(\Theta) = -\mathbb{E}_{p(h|v;\Theta)p_0(v)}\left[\nabla_\theta E(v, h; \Theta)\right],$$

where $p_0(v)$ is the empirical distribution $\frac{1}{N}\sum_{i=1}^n \delta(v = v_i)$. To run SPD with a diagonal Hessian approximation, we give $\mathrm{diag}(\nabla_\theta^2 h(\Theta))$ based on the formulation:

$$\nabla_{\theta_i}^2 h(\Theta) = \nabla_{\theta_i} h(\Theta) - (\nabla_{\theta_i} h(\Theta))^2,$$

whose derivation can be found in the supplement.

Although for RBMs the second expectation $\nabla_\theta h(\Theta)$ is tractable, the first $\nabla_\theta g(\Theta)$ is not because the expectation is taken with respect to v and h. Practically, contrastive divergence (CD) [23] or persistent contrastive divergence (PCD) [24] is used to obtain a stochastic approximation of $\nabla_\theta g(\Theta)$. Therefore, we can apply SPD with $\sigma_h^2 = 0$ to training RBMs. Note that, in each iteration of SPD, $\nabla h(\Theta)$ is computed at the cost of N evaluations; however, in practice, this cost is relatively small compared to that of CD / PCD used in an underlying solver. In fact, previous work [25] has shown that to obtain a good approximation of the gradient, a sufficiently large number of Gibbs samples are required in the CD method.

It is intractable to compute both expectations $\nabla_\theta g(\Theta)$ and $\nabla_\theta h(\Theta)$ for general BMs. Therefore, we stochastically approximate these terms. We use persistent Gibbs sampling [7] to compute the model expectation term $\nabla_\theta g(\Theta)$. That is, we obtain new samples v, h in underlying solver by Gibbs sampling with few steps, initialized at the previous samples. This procedure is equivalent to PCD for RBMs. To estimate the data expectation $\nabla_\theta h(\Theta)$, we adopt self-normalized importance sampling using mean-field approximation: $q(h|\mu) = \prod_{j=1}^M q(h^j)$, with $q(h^j = 1) = \mu^j$, to the true distribution $p(h|v, \Theta)$. First, we perform following fixed-point iterations until convergence and obtain $q(h|\mu)$, where $\mu = (\mu^1, \ldots, \mu^M)$, as done in [7],

$$\mu^j \leftarrow \sigma\left(c^j + \sum_i W_{ij} v^i + \sum_k J_{kj}\mu^k\right),$$

where σ is the sigmoid function. Next we draw samples $\{h_s\}_{s=1}^P$ from $q(h|\mu)$ and approximate $\nabla_\theta h(\Theta)$ as follows:

$$\nabla_\theta h(\Theta) \sim \mathbb{E}_{p_0(v)}\left[\frac{\sum_{s=1}^P -\nabla_\theta E(v, h_s|\Theta)\cdot\omega_s}{\sum_{s=1}^P \omega_s}\right],$$

where $\omega_s = \frac{\exp(-E(v,h_s;\Theta))}{q(h_s|\mu)}$ is a ratio between joint distribution and $q(h)$, so that it is computable. This estimate is asymptotically consistent [26]. Using these approximations, we can run SPD for learning BMs. For simplicity of implementation, we may use $P = 1$ and the expectation μ instead of a sample h^1 to reduce the sampling variance, even though it may have a relatively large bias, and so the resulting approximation of ∇h is the same as the mean-field approximation.

5.1 SPD as The Extension of EM/MCEM Algorithms

In the following we describe the connection between EM, MCEM algorithms and SPD. Let Θ' be a current parameter of a BM, $V = \{v_i\}_{i=1}^N$, and $H = \{h_i\}_{i=1}^N$ be i.i.d data samples and corresponding hidden variables, respectively. At the E-step of the EM algorithm we computes the following expectation of the log-likelihood of the joint distribution:

$$\begin{aligned} Q(\Theta, \Theta') &= \frac{1}{N}\int p(H|V,\Theta')\log p(V,H|\Theta)dH \\ &= \mathbb{E}_{p_0(v)p(h|v,\Theta')}[-E(v,h;\Theta)] - \log Z. \end{aligned}$$

In MCEM, the first term of the right hand side is approximated by a Monte Carlo method. This term is a linear mapping with respect to $\forall\theta \in \Theta$ and its gradient is nothing else but $\nabla_\theta h(\Theta')$. Combining the fact $g(\Theta) = \log Z$, we conclude that $Q(\Theta, \Theta')$ is the objective function of the sub-problem of SPD with $\mu = 0$ and the M-step in EM/MCEM corresponds to solving this sub-problem. Therefore, SPD can be recognized as an extension of the EM/MCEM algorithm for training BMs with better convergence analyses. Note that the proximal term of SPD with L_g or L_h convexity does not affect the convergence rate by Theorem 1 and 2, while it facilitates the optimization of the sub-problems by its strong convexity. Thus, SPD may be the more efficient method than EM/MCEM algorithms.

6 Experiments

In this section, we demonstrate the effectiveness of AdaSPD on training RBMs and DBMs with the weight decay. Our implementation is done using Theano [27, 28]. We used the binarized MNIST [25] which has 60,000 training and 10,000 test images (28×28 pixels) of 10 handwritten digits (0-9) and used CalTech101 Silhouettes [29] which has 6,364 training and 2,307 test images (28×28 pixels) of 101 classes, representing object silhouettes.

Since computing the partition function of BMs is difficult (except for small RBMs), we used the annealed importance sampling (AIS) [25] to estimate it with the settings: (i) for RBM, 500 temperatures spaced uniformly from 0 to 0.5, 4,000 temperatures spaced uniformly from 0.5 to 0.9, 10,000 temperatures spaced uniformly from 0.9 to 1, and 100 particles, (ii) for DBM, 20,000 temperatures spaced uniformly, and 1,000 particles. Theoretically, AdaSPD uses a random iteration count R to establish complexity results to solve problems; however, we always evaluate the model at the current iteration. The number of underlying solver iterations T and the suffix averaging parameter α were set as follows: $T = \lceil N/b \rceil, \alpha T = \lceil T/2 \rceil$, where N is the number of data points and b is a mini-batch size. All parameter settings of AdaSPD used in experiments can be found in the supplement.

Restricted Boltzmann Machines

We compare AdaSPD to SGD and AdaGrad on RBMs with 15, 25, and 500 hidden units. For metric H_k of AdaSPD, we tested the diagonal Hessian approximation and scalar matrices with $\mu \in \{10^{-1}, 10^{-3}, 10^{-5}\}$.

The results are shown in Figure 1. The top row represents the result for binarized MNIST dataset and the bottom row represents the result for CalTech101 Silhouettes. The vertical axis is the average (estimated) log-likelihood on training dataset. As can be seen in the figure, AdaSPDs showed significantly fast convergence compared to the others, although it tends to over-fitting, especially for the 500-hidden RBM on CalTech101 Silhouettes dataset. For binarized MNIST, the best training log-likelihood of the 500-hidden RBM was -83.19 and test log-likelihood was -85.83 obtained by AdaSPD with the diagonal Hessian approximation. These results are comparable to those reported in [9, 25]. For CalTech101 Silhouettes, the best training log-likelihood of the 500-hidden RBM was -84.15 obtained by AdaSPD with the scalar matrix ($\mu = 10^{-3}$) and test log-likelihood was -109.95 obtained by AdaSPD with the scalar matrix ($\mu = 10^{-1}$).

Deep Boltzmann Machines

Next, we train two DBMs: one has three-hidden layers (500-500-1000 hidden units) and the other has four-hidden layers (500-500-500-1000 hidden units). The results are shown in Table 2. Stochastic approximation procedure (SAP) [7] is the standard method for training DBMs. We should point out that AdaSPD and SAP were run without any sophisticated pre-training such as [7, 8]. Although AdaSPD showed a little bit worse score than the method using the best pre-training strategy [8], it outperformed SAP and was comparable to or better than the other pre-training methods proposed in [7, 8]. Thus, our experiments showed the possibility that no pre-training methods can lead to a better BM model. Figure 2 shows the learning curves for AdaSPD.

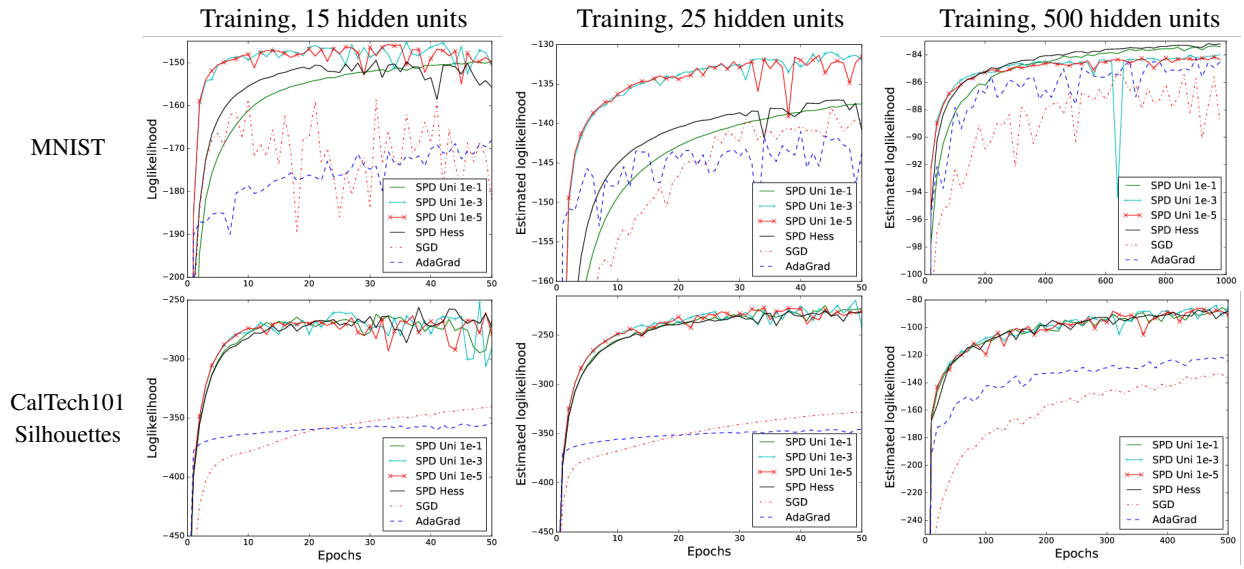

Figure 1: Comparison of algorithms on training RBMs with 15, 25, and 500 hidden units. The vertical axis is the average (estimated) log-likelihood on training dataset. Top row: MNIST, Bottom row: CalTech101 Silhouettes.

Table 2: Comparison of estimated variational lower bound on the log-likelihood of MNIST dataset.

Algorithms	3-hidden layers DBM		4-hidden layers DBM	
	Train	Test	Train	Test
AdaSPD	-82.28	-85.17	-82.85	-85.15
SAP [8]	-	-128.72	-	-128.70
Two-stage pre-training+SAP [8]	-	-81.84	-	-83.25
Pre-Training+SAP [7]	-84.49	-85.10	-	-

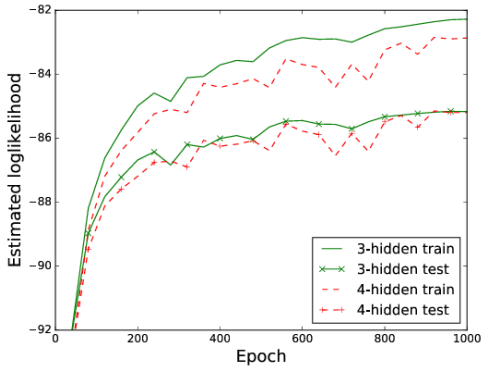

Figure 2: Learning curves for AdaSPD on DBMs.

7 Conclusion

We have proposed the SPD to solve DC programming under the stochastic setting and have given theoretical convergence analyses under several situations. Specifically, we have shown faster convergence than vanilla stochastic gradient methods when the variance growth condition is satisfied. Experiments have shown the effectiveness of AdaSPD for training RBMs and DBMs without pre-training.

Acknowledgements

This work was partially supported by MEXT kakenhi (25730013, 25120012, 26280009, 15H01678 and 15H05707), JST-PRESTO and JST-CREST.

References

[1] T. Pham Dinh and E. B. Souad. Algorithms for solving a class of nonconvex optimization problems: Methods of subgradient. In *Fermat Days 85: Mathematics for Optimization*, volume 129 of *North-Holland Mathematics Studies*, pages 249–271. Elsevier, 1986.

[2] A. Argyriou, R. Hauser, C. A. Micchelli, and M. Pontil. A DC-programming algorithm for kernel selection. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 41–48, 2006.

[3] H. A. Le Thi, L. H. Minh, N. V. Vinh, and T. Pham Dinh. A DC programming approach for feature selection in support vector machines learning. *Advances*

in Data Analysis and Classification, 2(3):259–278, 2008.

[4] A. Ferrer. Representation of a polynomial function as a difference of convex polynomials, with an application. *Lectures Notes in Economics and Mathematical Systems*, 502:189–207, 2001.

[5] S. Wang, A. Schwing, and R. Urtasun. Efficient inference of continuous markov random fields with polynomial potentials. In *Advances in Neural Information Processing Systems 25*, pages 936–944. 2014.

[6] A. Ahmadi and G. Hall. Dc decomposition of nonconvex polynomials with algebraic techniques. Technical report, arXiv:1510.01518, 2015.

[7] R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *Proceeding of The Twelfth International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

[8] K. Cho, T. Raiko, A Ilin, and J Karhunent. A two-stage pretraining algorithm for deep Boltzmann machines. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[9] D. Carlson, V. Cevher, and L. Carin. Stochastic spectral descent for restricted Boltzmann machines. In *Proceeding of The Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 111–119, 2015.

[10] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning*, pages 449–456, 2012.

[11] X. Chen, Q. Lin, and J. Pena. Optimal regularized dual averaging methods for stochastic optimization. In *Advances in Neural Information Processing Systems 25*, pages 395–403. 2012.

[12] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, ii: Shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089, 2013.

[13] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. Technical report, arXiv:1606.04838, 2016.

[14] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[15] D. Carlson, Y. Hsieh, E. Collins, L. Carin, and V. Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):296–311, 2016.

[16] J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems 26*, pages 2283–2291. 2013.

[17] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[18] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[19] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[20] S. Becker and Y. LeCun. Improving the convergence of back-propagation learning with second order methods. Technical report, Department of Computer Science, University of Toronto, 1989.

[21] M. Schmidt and N. Le Roux. Fast convergence of stochastic gradient descent under a strong growth condition. Technical report, arXiv:1308.6370, 2013.

[22] M. Gurbuzbalaban, A. Ozdaglar, and P. Parrilo. A globally convergent incremental Newton method. *Mathematical Programming*, 151(1):283–313, 2015.

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

[24] T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1033–1040, 2009.

[25] R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 872–879, 2008.

[26] A. B. Owen. *Monte Carlo Theory, Methods and Examples*. 2014.

[27] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

[28] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[29] B. M. Marlin, K. Swersky, B. Chen, and N de Freitas. Inductive principles for restricted Boltzmann machine learning. In *Proceeding of The Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 509–516, 2010.