Efficient Stochastic Optimization for Low-Rank Distance Metric Learning

Jie Zhang, Lijun Zhang
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
zhangj@lamda.nju.edu.cn, zhanglj@lamda.nju.edu.cn

Abstract

Although distance metric learning has been successfully applied to many real-world applications, learning a distance metric from large-scale and high-dimensional data remains a challenging problem. Due to the PSD constraint, the computational complexity of previous algorithms per iteration is at least $O(d^2)$ where d is the dimensionality of the data. In this paper, we develop an efficient stochastic algorithm for a class of distance metric learning problems with nuclear norm regularization, referred to as low-rank DML. By utilizing the low-rank structure of the intermediate solutions and stochastic gradients, the complexity of our algorithm has a linear dependence on the dimensionality d. The key idea is to maintain all the iterates in factorized representations and construct stochastic gradients that are low-rank. In this way, the projection onto the PSD cone can be implemented efficiently by incremental SVD. Experimental results on several data sets validate the effectiveness and efficiency of our method.

Introduction

Distance metric learning (DML) aims to learn a distance metric for the input space of data from a given set of training examples (Yang and Jin 2006). A well-learned distance metric can reflect domain-specific connections and relationships, and significantly improve the performance of distance-based learning algorithms such as K-means clustering (Xing et al. 2002) and nearest neighbor classification (Weinberger and Saul 2009). In the past decades, a large number of DML methods have been proposed like Pseudo-metric Online Learning Algorithm (POLA) (Shalev-Shwartz, Singer, and Ng 2004), Maximally Collapsing Metric Learning (MCML) (Globerson and Roweis 2005), Information-Theoretic Metric Learning (ITML) (Davis et al. 2007), Large Margin Nearest Neighbor (LMNN) (Weinberger and Saul 2009) and Riemannian Similarity Learning (RSL) (Cheng 2013). These methods have been successfully applied to many real-world applications, such as image retrieval and classification (Kulis 2013; Frome et al. 2007), bioinformatics (Kato and Nagano 2010), text analysis (Davis and Dhillon 2008), and Automated Program Debugging (Ha et al. 2007).

Although DML has achieved great success in practice, learning a distance metric from large-scale and high-dimensional data remains a big challenge, due to the high computational complexity. Generally speaking, the optimization problem of DML is formulated as a convex optimization problem over the positive semi-definite (PSD) cone, which can be solved by gradient-based methods (Nesterov 2004). The computational challenge is mainly caused by two factors: (i) the evaluation of the gradient is expensive when there is a large number of training examples, and (ii) the projection onto the PSD cone is also costly when the dimensionality is high. While the first issue can be easily addressed by stochastic methods (Shalev-Shwartz et al. 2009), such as stochastic gradient descent (SGD) which only samples a subset of training examples to calculate the gradient, the second one is much more tricky. Let d be the dimensionality of the data. The projection onto the PSD cone needs to eigen decompose a matrix of size $d \times d$, and thus has $O(d^3)$ time complexity.

To reduce the computational cost of projections, the most straightforward way is to learn the distance metric after performing dimensionality reduction (Weinberger and Saul 2009; Tsagkatakis and Savakis 2010; Qian et al. 2015b). However, due to the subspace removed by dimension reduction methods, this kind of methods always lead to sub-optimal solutions. Another way to reduce the computational cost is to utilize more advanced optimization techniques that avoid the projection step, such as the Frank-Wolfe technique (Hazan and Kale 2012), and the convex-concave formulation (Mahdavi et al. 2012). Although the computational cost per iteration can be reduced from $O(d^3)$ to $O(d^2)$, it is still prohibitive for high dimensional problems. A recent work has shown that by constructing a low-rank stochastic gradient, it is also possible to reduce the computational cost of SGD per iteration to $O(d^2)$ (Chen, Yang, and Zhu 2014).

In this paper, we focus on the problem of low-rank distance metric learning (Davis and Dhillon 2008; Liu et al. 2015), in which a nuclear norm regularizer is introduced to promote low-rankness of the distance metric. The advantage of low-rank DML is that it can reduce effects of noise and prevent model overfitting as well. Similar to previous studies (Chen, Yang, and Zhu 2014), we construct a low-rank stochastic gradient in each round and apply SGD to update the intermediate solution. However, our algorithm goes one step further by exploiting the fact that both the intermediate solution and the stochastic gradient are low-rank to accelerate the projection operation. Specifically, we maintain all the iterates and stochastic gradients in factorized representations and the projection operation can be implemented in $O(dr^2)$ time by incremental SVD (Brand 2006), where r is the rank of the intermediate solution. The main advantages of the proposed algorithm are summarized as follows.

- By utilizing the low-rank structure of intermediate solutions and stochastic gradients, the time complexity per iteration is $O(dr^2)$, which depends on d linearly.
- The proposed algorithm always maintains low-rank factorizations of iterates and gradients, and the space complexity is $O(dr)$, in contrast to the $O(d^2)$ space complexity of previous methods.
- Since the proposed algorithm is a SGD-based approach, existing theoretical guarantees for SGD also hold for our algorithm. In particular, the convergence rate of our algorithm is $O(\log T/\sqrt{T})$ and $O(\log T/T)$ for general convex functions and strongly convex functions, respectively.

Related Work

Distance metric learning (DML) is a main research field in machine learning and has been extensively studied in the past decades. In this section, we mainly describe techniques for reducing the computational cost of DML. For other aspects of DML, please refer to (Yang and Jin 2006; Bellet, Habrard, and Sebban 2013; Kulis 2013).

A majority of DML methods concentrated on learning a Mahalanobis distance

$$d_W(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^\top W (\mathbf{x} - \mathbf{y})$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ are two data points, and $W \in \mathbb{R}^{d\times d}$ is a symmetric PSD matrix. The optimization problem of DML is generally formulated as

$$\begin{aligned} \min_{W\in\mathbb{R}^{d\times d}} \quad & f(W) \\ \text{s. t.} \quad & W \in \mathbb{S}_+ \end{aligned} \tag{1}$$

where $\mathbb{S}_+$ is the set of symmetric positive semi-definite matrices, and referred to as the PSD cone in optimization (Boyd and Vandenberghe 2004). Here, $f(\cdot)$ is the loss function that is algorithm-dependent. The standard algorithm for optimizing (1) is the gradient descent (GD) (Nesterov 2004)

$$W_{t+1} = \Pi_{\mathbb{S}_+}\left[W_t - \eta_t \nabla f(W_t)\right]$$

where W_t is the solution in the t-th round, $\eta_t > 0$ is the step size, and $\Pi_{\mathbb{S}_+}[\cdot]$ denotes the projection onto the PSD cone $\mathbb{S}_+$. The complexity of calculating the gradient $\nabla f(W_t)$ depends on the number of training examples and the dimensionality d, and the complexity of the projection is $O(d^3)$. To make DML methods practical for large-scale and high-dimensional problems, three main techniques are applied.

The first strategy performs certain preprocessing to the data before learning the distance metric. In the literatures, many methods choose to employ dimensionality reduction methods like Principal Component Analysis (PCA) (Weinberger and Saul 2009), and random projection (RP) (Tsagkatakis and Savakis 2010). However, because of the subspace removed by dimensionality reduction, these methods always result in suboptimal solutions. To alleviate this limitation, Qi et al. present a dual random projection framework for DML (Qian et al. 2015b). Though they can recover the optimal solution with a small error, their formulation essentially drops the PSD constraint. For low-rank DML, Liu et al. propose to apply Singular Value Decomposition (SVD) first and then use the linearized modification of ADMM to learn a low-dimensional metric (Liu et al. 2015). Their method works fine when the number of data is small but is not feasible when the number of data is very large.

The second strategy resorts to making assumptions on the structure of the distance metric to be learned. For example, the authors of (Schultz and Joachims 2003) suppose the distance metric is diagonal, and the authors of (Lim, Lanckriet, and McFee 2013; Qi et al. 2009) enforce the distance metric to be sparse. However, strong assumptions of the distance metric may limit their applications. In (Weinberger and Saul 2009) and (Kedem et al. 2012), the authors assume that $W = LL^\top$ where L is low-rank. Though these methods allow us to remove the PSD constraint on W and the computation of gradients is also easier, they can only find local optima because of their non-convex formulations.

The third strategy is based on more advanced optimization methods. Perhaps the most well-known one is the stochastic gradient descent (SGD)

$$W_{t+1} = \Pi_{\mathbb{S}_+}\left[W_t - \eta_t g(W_t)\right]$$

where $g(W_t)$ is a stochastic gradient such that $\mathrm{E}[g(W_t)] = \nabla f(W_t)$. Because the stochastic gradient is easy to calculate, the complexity of SGD is cheaper than GD per iteration. However, the projection step remains there, and may become the bottleneck when the dimensionality is high. Although mini-batches can be incorporated into SGD to reduce the number of projections to some extent (Qian et al. 2015a), the number of projections is still very large if we want to find a high-precision solution (Cotter et al. 2011). In (Chen, Yang, and Zhu 2014), the authors describe an efficient SGD algorithm that takes advantage of the low-rank structure of the stochastic gradient.

There are also some optimization techniques that do not need the projection step. For example, in (Hazan and Kale 2012), the authors develop a projection-free SGD algorithm based on the Frank-Wolfe technique, which involves solving constrained linear programming problems. In (Mahdavi et al. 2012), the authors propose a convex-concave formulation that moves the domain constraint into the objective function, and thus avoid the projection. The algorithms in (Hazan and Kale 2012; Mahdavi et al. 2012; Chen, Yang, and Zhu 2014) are efficient in the sense that the computational cost per iteration is reduced from $O(d^3)$ to $O(d^2)$, but it is still a heavy burden for large d. Furthermore, their space complexity is also $O(d^2)$ since they need to store a $d \times d$ matrix in memory.

Compared to (Hazan and Kale 2012; Mahdavi et al. 2012; Chen, Yang, and Zhu 2014), the time complexity of the algorithm in this paper only has a linear dependence on d instead

Algorithm 1 Efficient SGD for low-rank DML

```
Input: The number of trials $T$
1: Initialize $W_1 = 0$
2: for $t = 1, 2, \ldots, T$ do
3:     Construct a low-rank stochastic gradient $g(W_t) = A_t A_t^\top$ of $f(\cdot)$ at $W_t$
4:     Calculate $W_{t+1}$ according to (3) {The implementation is based on Incremental SVD}
5: end for
6: return $W_{T+1}$
```

of a quadratic dependence. Furthermore, the space complexity of our algorithm is also linear in d.

Algorithm

In this section, we present an efficient SGD method for low-rank DML (Davis and Dhillon 2008; Liu et al. 2015).

The Overall Procedure

To promote low-rank distance metrics, we introduce a nuclear norm regularizer into (1). Since W is a PSD matrix, we have

$$\|W\|_* = \operatorname{tr}(W)$$

where $\|\cdot\|_*$ is the nuclear norm and $\operatorname{tr}(\cdot)$ is the trace operation. As a result, the optimization problem of low-rank DML can be written as

$$\begin{aligned} \min_{W\in\mathbb{R}^{d\times d}} \quad & f(W) + \lambda \operatorname{tr}(W) \\ \text{s.t.} \quad & W \in \mathbb{S}_+ \end{aligned} \tag{2}$$

where $\lambda > 0$ is a trade-off parameter. As can be seen, due to the PSD constraint, the non-smooth nuclear norm $\|\cdot\|_*$ is replaced with a linear function $\operatorname{tr}(\cdot)$, which simplifies the development of optimization algorithms. Based on SGD, we will update W_t according to

$$W_{t+1} = \Pi_{\mathbb{S}_+}\left[W_t - \eta_t g(W_t) - \eta_t \lambda I\right] \tag{3}$$

where $g(W_t)$ is a stochastic gradient of $f(\cdot)$ evaluated at W_t. A remarkable property of (3) is that if W_t is low-rank, W_{t+1} also tends to be low-rank due to the minus of $\eta_t\lambda I$ and the projection operation.

Following (Avron et al. 2012; Zhang et al. 2016), we will enforce the following two requirements:

1. W_t is symmetric and represented in the form of eigen decomposition. [1] Let the rank of W_t be r_t. We have $W_t = U_t \Sigma_t U_t^\top$, where $U_t \in \mathbb{R}^{d\times r_t}$ and $\Sigma_t \in \mathbb{R}^{r_t\times r_t}$.
2. $g(W_t)$ is a low-rank and symmetric matrix, and is represented as $g(W_t) = A_t A_t^\top$, where $A_t \in \mathbb{R}^{d\times c_t}$.

When the above two conditions are satisfied, the updating rule in (2) can be solved efficiently by the incremental SVD (Brand 2006). The initial solution W_1 is set to the zero matrix, and we return the last iterate W_{T+1} as the final solution. The overall procedure is described in Algorithm 1.

[1]By default, we will use "compact" eigen decomposition, which only keeps the nonzero eigenvalues.

The application of incremental SVD to (2) is inspired by the previous stochastic algorithms for nuclear norm regularization (Avron et al. 2012; Zhang et al. 2016), so the overall frameworks are similar. However, there also exists some differences, as stated below.

- Both our algorithm and that in (Avron et al. 2012) are based on SGD, but we study the problem of distance metric learning and have an additional PSD constraint. Furthermore, in our algorithm, we simply take the last iterate of SGD as the final solution. In contrast, the algorithm in (Avron et al. 2012) needs to find the iterate that minimizes the objective function, which incurs additional computations.
- While our algorithm is a SGD-type algorithm, the algorithm in (Zhang et al. 2016) is based on stochastic proximal gradient descent (SPGD). Moreover, the authors in (Zhang et al. 2016) only consider the square loss and there is no PSD constraint.

In the following, we will discuss how to construct a low-rank stochastic gradient and how to solve (2) efficiently.

Constructing a Low-rank Stochastic Gradient

In the literature, we can find various ways to construct a low-rank stochastic gradient of $f(\cdot)$ (Avron et al. 2012; Chen, Yang, and Zhu 2014). In the following, we will provide one approach by taking the loss of Large Margin Nearest Neighbor (LMNN) (Weinberger and Saul 2009) as an example.

In LMNN, the loss function is given by

$$\begin{aligned} & f(W) \\ =& \frac{c}{n}\sum_{i,j,k} \rho_{ij}(1-y_{ij})\ell\big(1 + ||\mathbf{x}_i - \mathbf{x}_j||_W^2 - ||\mathbf{x}_i - \mathbf{x}_k||_W^2, 0\big) \\ & + \frac{1}{m}\sum_{i,j} \rho_{ij} ||\mathbf{x}_i - \mathbf{x}_j||_W^2 \end{aligned}$$

where $\rho_{ij} \in \{1, 0\}$ indicates if $\mathbf{x}_j$ is a target neighbor of $\mathbf{x}_i$, $y_{ij} \in \{1, 0\}$ represents whether $\mathbf{x}_i$ and $\mathbf{x}_j$ shares the same class label, $\ell(x, 0) = \max(x, 0)$ is the hinge loss, and n, m, c respectively denotes the number of active triplets, the number of pairs and a balancing parameter of the two terms.

The low-rank stochastic gradient of $f(\cdot)$ can be constructed as follows. We randomly sample a triplet (i_1, j_1, k) and a pair (i_2, j_2). Here, $(\mathbf{x}_{i_1}, \mathbf{x}_{j_1})$ and $(\mathbf{x}_{i_2}, \mathbf{x}_{j_2})$ respectively share the same label while $\mathbf{x}_k$ has a different label to the pair $(\mathbf{x}_{i_1}, \mathbf{x}_{j_1})$. Then, the stochastic gradient is given by

$$\begin{aligned} & g(W_t) \\ =& c(\mathbf{x}_{i_1} - \mathbf{x}_{j_1})(\mathbf{x}_{i_1} - \mathbf{x}_{j_1})^\top - c(\mathbf{x}_{i_1} - \mathbf{x}_k)(\mathbf{x}_{i_1} - \mathbf{x}_k)^\top \\ & + (\mathbf{x}_{i_2} - \mathbf{x}_{j_2})(\mathbf{x}_{i_2} - \mathbf{x}_{j_2})^\top , \end{aligned}$$

when $1 + ||\mathbf{x}_{i_1} - \mathbf{x}_{j_1}||_W^2 - ||\mathbf{x}_{i_1} - \mathbf{x}_k||_W^2 > 0$, and

$$g(W_t) = (\mathbf{x}_{i_2} - \mathbf{x}_{j_2})(\mathbf{x}_{i_2} - \mathbf{x}_{j_2})^\top ,$$

otherwise. It is obvious that $g(W_t)$ is a stochastic gradient of $g(\cdot)$. Furthermore, it is symmetric and low-rank, and thus can be represented as $A_t A_t^\top$. Finally, note that we can control the rank of $g(W_t)$ by sampling specific number of triplets and pairs.

Updating W_t by Incremental SVD

Next, we show that when W_t is represented in the form of eigen decomposition (i.e, $W_t = U_t\Sigma_t U_t^\top$) and $g(W_t)$ is represented in factorized form (i.e., $g(W_t) = A_t A_t^\top$), the updating rule in (3) can be implemented efficiently.

The key idea is to make use of the incremental SVD (Brand 2006) to calculate the eigen decomposition of $W_t - \eta_t g(W_t)$, which is described below.

Incremental SVD Let $W \in \mathbb{R}^{d\times d}$ be a rank-r matrix with eigen decomposition $W = U\Sigma U^\top$, where $U \in \mathbb{R}^{d\times r}$ and $\Sigma \in \mathbb{R}^{r\times r}$. Let $A \in \mathbb{R}^{d\times c}$. Then, the eigen decomposition of $W - AA^\top$ can be calculated in $O(d(r+c)^2 + (r+c)^3)$ time with $O(d(r+c))$ memory.

Let P be an orthogonal basis of the column space of $(I - UU^\top)A$, and set $R_A = P^\top(I-UU^\top)A$. Note that $\text{cols}(P) = \text{rows}(R_A) = \text{rank}((I-UU^\top)A) \le c$, and may be zero if A lies in the column space of U. Then, we have

$$[\,U \quad A\,] = [\,U \quad P\,]\begin{bmatrix} I & U^\top A \\ 0 & R_A \end{bmatrix}.$$

Then, we can easily get

$$W - AA^\top = [\,U \quad P\,]\,K\,[\,U \quad P\,]^\top$$

where

$$K = \begin{bmatrix} I & U^\top A \\ 0 & R_A \end{bmatrix}\begin{bmatrix} \Sigma & 0 \\ 0 & -I \end{bmatrix}\begin{bmatrix} I & U^\top A \\ 0 & R_A \end{bmatrix}^\top$$
$$= \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} U^\top A \\ R_A \end{bmatrix}\begin{bmatrix} U^\top A \\ R_A \end{bmatrix}^\top \in \mathbb{R}^{(r+p)\times(r+p)}$$

and $p = \text{cols}(P)$. Let the eigen decomposition of K be $K = \widehat{U}\widehat{\Sigma}\widehat{U}^\top$. Then, the eigen decomposition of $W - AA^\top$ is given by

$$W - AA^\top = \left([\,U \quad P\,]\widehat{U}\right)\widehat{\Sigma}\left([\,U \quad P\,]\widehat{U}\right)^\top.$$

Efficient Updating

We first recall some basic facts of projection onto the PSD cone.

Theorem 1. *Let $X \in \mathbb{R}^{d\times d}$ be a symmetric matrix with eigen decomposition*

$$X = \sum_i \lambda_i \mathbf{u}_i\mathbf{u}_i^\top$$

where $(\lambda_i \in \mathbb{R}, \mathbf{u}_i \in \mathbb{R}^d)$ is the i-th pair of eigenvalue and eigenvector. Then, the projection of X onto the PSD cone $\mathbb{S}_+$ is given by (Boyd and Vandenberghe 2004)

$$\Pi_{\mathbb{S}_+}[X] = \sum_{i:\lambda_i>0} \lambda_i \mathbf{u}_i\mathbf{u}_i^\top.$$

Following the above theorem, the updating rule in (3) can be implemented in two steps:

1. Given $W_t = U_t\Sigma_t U_t^\top$ and $g(W_t) = A_tA_t^\top$, we use the incremental SVD to obtain the eigen decomposition of $\widehat{W}_{t+1} = W_t - \eta_t g(W_t)$. Let r_t and c_t be the rank of W_t and $g(W_t)$, respectively. The time and space complexities of this step are $O(d(r_t+c_t)^2 + (r_t+c_t)^3)$ and $O(d(r_t+c_t))$, respectively.
2. Based on the eigen decomposition of $\widehat{W}_{t+1} = \sum_i \mu_i \mathbf{v}_i\mathbf{v}_i^\top$, $W_{t+1} = \Pi_{\mathbb{S}_+}[\widehat{W}_{t+1} - \eta_t\lambda I]$ can be calculated directly according to Theorem 1. Specifically, we have
$$W_{t+1} = \sum_{i:\mu_i>\eta_t\lambda}(\mu_i - \eta_t\lambda)\mathbf{v}_i\mathbf{v}_i^\top.$$
As can be seen, W_{t+1} is also represented in the form of eigen decomposition. The time complexity of this step is $O(r_t + c_t)$, and there is no additional space requirement.

Note that the eigenvalues of $\widehat{W}_{t+1}$ that is smaller than $\eta_t\lambda$ will be removed, and thus W_{t+1} tends to be a low-rank matrix. By choosing $c_t = O(r_t)$, the time complexity per iteration is $O(dr^2)$ and the space complexity is $O(dr)$, where $r = \max_t r_t$.

Discussions

In this section, we discuss some extensions of low-rank DML in (2), and then provide the convergence rate of the proposed algorithm.

Low-rank DML with Additional Constraints

To avoid overfitting, we may want to add some constraints to control the size of the distance metric.

We first consider imposing a spectral norm constraint. Then, the optimization problem becomes

$$\min_{W\in\mathbb{R}^{d\times d}} \quad f(W) + \lambda\,\text{tr}(W)$$
$$\text{s.t.} \quad W \in \mathbb{S}_+,\ \|W\|_2 \le \tau$$

where $\|\cdot\|_2$ is the spectral norm of matrices. Then, the updating rule of SGD becomes

$$W_{t+1} = \Pi_{\mathbb{S}_+\cap\{W|\|W\|_2\le\tau\}}[W_t - \eta_t g(W_t) - \eta_t\lambda I]. \quad (4)$$

To calculate W_{t+1}, we need the following theorem.

Theorem 2. *Let $X \in \mathbb{R}^{d\times d}$ be a symmetric matrix with eigen decomposition*

$$X = \sum_i \lambda_i \mathbf{u}_i\mathbf{u}_i^\top$$

where $(\lambda_i \in \mathbb{R}, \mathbf{u}_i \in \mathbb{R}^d)$ is the i-th pair of eigenvalue and eigenvector. Then, the projection of X onto the intersection of $\mathbb{S}_+$ and $\{W|\|W\|_2 \le \tau\}$ is given by

$$\Pi_{\mathbb{S}_+\cap\{W|\|W\|_2\le\tau\}}[X] = \sum_{i:\lambda_i>\tau}\tau\mathbf{u}_i\mathbf{u}_i^\top + \sum_{i:0<\lambda_i\le\tau}\lambda_i\mathbf{u}_i\mathbf{u}_i^\top$$

According to Theorem 2, the updating rule in (4) can be implemented as:

1. We use the incremental SVD to obtain the eigen decomposition of $\widehat{W}_{t+1} = W_t - \eta_t g(W_t)$.
2. Based on the eigen decomposition of $\widehat{W}_{t+1} = \sum_i \mu_i\mathbf{v}_i\mathbf{v}_i^\top$, W_{t+1} is calculated as
$$W_{t+1} = \sum_{i:\mu_i-\eta_t\lambda>\tau}\tau\mathbf{u}_i\mathbf{u}_i^\top + \sum_{i:0<\mu_i-\eta_t\lambda\le\tau}(\mu_i-\eta_t\lambda)\mathbf{u}_i\mathbf{u}_i^\top$$

We then consider adding a Frobenius norm constraint to (2). In this case, the updating rule of SGD becomes

$$W_{t+1} = \Pi_{\mathbb{S}_+ \cap \{W | \|W\|_F \le \tau\}} \left[W_t - \eta_t g(W_t) - \eta_t \lambda I \right] . \quad (5)$$

To calculate W_{t+1}, we need the following theorem.

Theorem 3. *Let $X \in \mathbb{R}^{d \times d}$ be a symmetric matrix with eigen decomposition*

$$X = \sum_i \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$$

where $(\lambda_i \in \mathbb{R}, \mathbf{u}_i \in \mathbb{R}^d)$ is the i-th pair of eigenvalue and eigenvector. Then, the projection of X onto the intersection of $\mathbb{S}_+$ and $\{W | \|W\|_F \le \tau\}$ is given by

$$\Pi_{\mathbb{S}_+ \cap \{W | \|W\|_F \le \tau\}} [X] = \min \left(1, \frac{\tau}{\|\Pi_{\mathbb{S}_+}[X]\|_F} \right) \Pi_{\mathbb{S}_+} [X]$$

According to Theorem 3, the updating rule in (5) can be implemented as:

1. We use the incremental SVD to obtain the eigen decomposition of $\widehat{W}_{t+1} = W_t - \eta_t g(W_t)$.
2. Based on the eigen decomposition of $\widehat{W}_{t+1} = \sum_i \mu_i \mathbf{v}_i \mathbf{v}_i^\top$, W_{t+1} is calculated as
$$W_{t+1} = \min \left(1, \frac{\tau}{F} \right) \sum_{i: \mu_i > \eta_t \lambda} (\mu_i - \eta_t \lambda) \mathbf{v}_i \mathbf{v}_i^\top$$
where $F = \sqrt{\sum_{i: \mu_i > \eta_t \lambda} (\mu_i - \eta_t \lambda)^2}$.

Convergence Rate

Since our method is based on SGD, the recent theoretical guarantees of SGD can be directly applied. Specifically, we need the convergence rate of the last iterate of SGD (Rakhlin, Shamir, and Sridharan 2012; Shamir and Zhang 2013).

We first consider the case that $f(\cdot)$ in (2) is a general convex function. Define $F(\cdot) = f(\cdot) + \lambda \operatorname{tr}(\cdot)$, and let $\mathcal{W}$ be

$$\mathbb{S}_+ \cap \{W | \|W\|_2 \le \tau\} \text{ or } \mathbb{S}_+ \cap \{W | \|W\|_F \le \tau\} .$$

According to Theorem 2 of (Shamir and Zhang 2013), we have the following theorem.

Theorem 4. *Suppose $f(\cdot)$ is convex and that for some constants D, G, it holds that $\mathrm{E}[\|g(W_t) + \lambda I\|_F^2] \le G^2$ for all t, and $\sup_{W, W' \in \mathcal{W}} \|W - W'\|_F \le D$. Consider Algorithm 1 with step size $\eta_t = c/\sqrt{t}$ where c > 0 is a constant. Then for any $T > 1$, it holds that*

$$\mathrm{E}[F(W_T) - F(W^*)] \le \left(\frac{D^2}{c} + cG^2 \right) \frac{2 + \log(T)}{\sqrt{T}}$$

where $W^* = \operatorname{argmin}_{W \in \mathcal{W}} F(W)$.

The above theorem indicates that the last iterate converges at an $O(\log T / \sqrt{T})$ rate.

When the loss $f(\cdot)$ is strongly convex, which could be true if $\|W\|_F^2$ is a part of $f(\cdot)$, we have a faster convergence rate. Based on Theorem 1 of (Shamir and Zhang 2013), we obtain the theorem below.

Theorem 5. *Suppose $f(\cdot)$ is μ-strongly convex, and $\mathrm{E}[\|g(W_t) + \lambda I\|_F^2] \le G^2$ for all t. Consider Algorithm 1 with step size $\eta_t = 1/(\mu t)$. Then for any $T > 1$, it holds that*

$$\mathrm{E}[F(W_T) - F(W^*)] \le \frac{17 G^2 (1 + \log T)}{\mu T} .$$

Thus, for strongly convex losses, our algorithm has an $O(\log T / T)$ rate of convergence. Finally, we note that when the loss is both smooth and strongly convex, the rate can be further improved to $O(1/T)$ (Rakhlin, Shamir, and Sridharan 2012).

Experiments

In this section, we present empirical studies of the proposed method. The main purpose is to illustrate two characteristics of our method:

- Memory-efficient: The proposed algorithm has an $O(dr)$ space complexity, where r is an upper bound of the rank of the intermediate solution. To verify our algorithm is memory-efficient, we need to show that all the intermediate solutions are indeed low-rank.
- Computation-efficient: By taking advantage of the low-rank structure of intermediate solutions and stochastic gradients, the time complexity is $O(dr^2)$ per iteration. We will examine the convergence behavior of our method.

We will choose the following formulation of low-rank DML (Qian et al. 2015a):

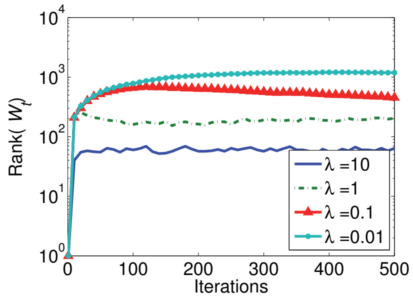
$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times d}} \quad & \frac{1}{T} \sum_{i,j,k} \max(1 + \|\mathbf{x}_i - \mathbf{x}_j\|_W^2 - \|\mathbf{x}_i - \mathbf{x}_k\|_W^2, 0) \\ & + \lambda \operatorname{tr}(W) \\ \mathrm{s.\,t.} \quad & W \in \mathbb{S}_+, \ \|W\|_F \le 1 \end{aligned}$$

where T is the total number of triplets. We will use three real-world data sets in our experiments: Gisette (Chang and Lin 2011), Dexter (Guyon et al. 2004) and News20 (Lang 1995). The dimensionality of them is 5000, 20000, and 62061, respectively. The generation of the training triplets is similar to (Qian et al. 2015a). In each iteration, we randomly sample 100 triplets to construct the low-rank stochastic gradient. All algorithms are tested on a computer with 3.1GHz CPU and 8GB RAM.

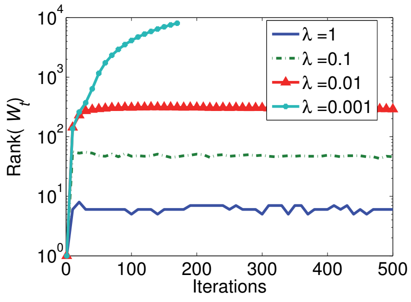
The Rank of the Intermediate Distance Metrics

We set the step size $\eta_t = c/\sqrt{t}$ where c is searched in $\{1e{-7}, 1e{-6}, \ldots, 1, 10\}$, and we choose the one that leads to the largest decrement of the objective value. We run our algorithm 500 iterations and record the rank of the intermediate solution W_t in each round. We terminate the program if the running time exceeds 3 hours.

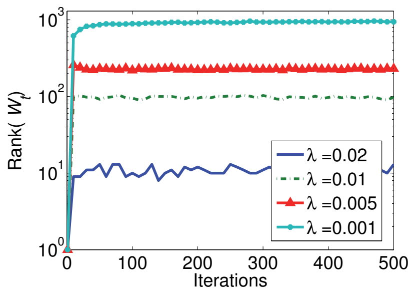
We have tried different settings of the regularization parameter λ, and the rank of W_t of all the data sets is plotted in Fig. 1. As can be seen, the rank could be much smaller than the dimensionality with a suitable choice of λ, leading to a low memory requirement. For example, on the Dexter data set, the rank is around 100 if we set $\lambda = 0.1$, which is much smaller than its dimensionality 20000. The experiment validates our method's memory-efficient property.



(a) Gisette with $c = 0.001$

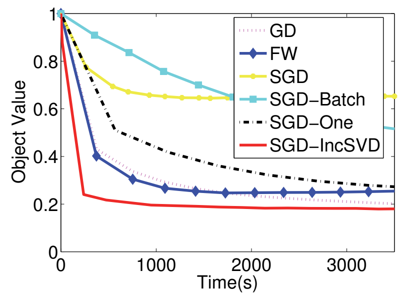


(b) Dexter with $c = 0.01$

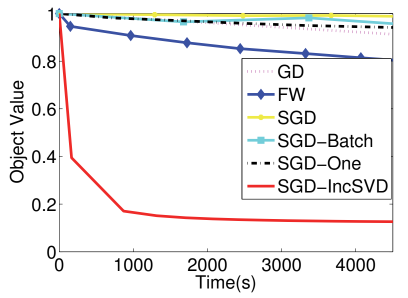


(c) News20 with $c = 10$

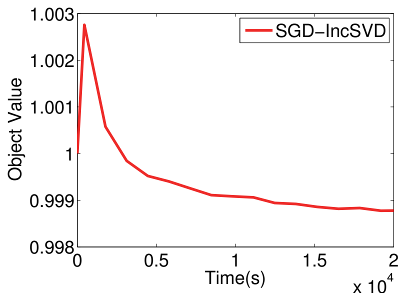
Figure 1: Rank of the intermediate solution versus the number of iterations



(a) Gisette



(b) Dexter



(c) News20

Figure 2: Objective value versus the running time

The Convergence Behavior

To show the efficiency of our method, we compare it with the following optimization algorithms:

- Gradient descent (GD), which calculates the full gradient and performs the projection by eigen decomposition;
- Stochastic gradient descent (SGD), which replaces gradients in GD with stochastic gradients;
- SGD with mini-batch (SGD-Batch), which uses mini-batch to reduce the number of projections of SGD (Shaw, Huang, and Jebara 2011; Qian et al. 2015a);
- Frank-Wolfe method (FW), which solves constrained linear programming problems instead of projections (Ying and Li 2012; Cao, Ying, and Li 2012; Jaggi 2013);
- Stochastic gradient descent with only one projection (SGD-One) (Mahdavi et al. 2012): which only requires one projection computation at the final iteration.

For convenience, we denote our algorithm by SGD-IncSVD to emphasize the fact it is built upon the incremental SVD.

Parameters of all the methods are determined in the following way: For GD, SGD-Batch and SGD-One, we search the step size in $\{1e{-}7, 1e{-}6, \ldots, 1, 10\}$; For SGD and SGD-IncSVD, we set the step size $\eta_t = c/\sqrt{t}$, where c is searched in $\{1e{-}7, 1e{-}6, \ldots, 1, 10\}$; For the FW method, we set $\eta_t = c/(t+2)$ where c is searched in the same range; For SGD-Batch, we set the batch size to 10, according to the suggestion in (Qian et al. 2015a).

In Fig. 2, we plot the value of the objective function versus the running time of different methods, where the regularization parameter λ is set to be 1, 0.1 and 0.01 for Gisette, Dexter and News20, respectively. As can be seen from Fig. 2, our algorithm converges much faster than all the compared algorithms and when the dimension of the data is much higher, the advantage of SGD-IncSVD is more obvious. Thus, we may claim that reducing the cost of projection is essential for improving the efficiency. On the News20 data set, we only provide the result of SGD-IncSVD because all the other methods suffer memory overflow issues, which again validates our method is memory-efficient.

Conclusions

We present an efficient stochastic algorithm for solving low-rank DML. By taking advantage of the low-rank structure, we use incremental SVD to obtain $O(dr^2)$ time complexity per iteration. Since all the iterates and stochastic gradients are represented in factorized forms, the space complexity is also linear in d. Experimental results validate the memory-efficient and computation-efficient features of our method.

Acknowledgments

This work was partially supported by the NSFC (61603177, 61333014), JiangsuSF (BK20160658), and the Collaborative Innovation Center of Novel Software Technology and Industrialization of Nanjing University.

References

Avron, H.; Kale, S.; Kasiviswanathan, S.; and Sindhwani, V. 2012. Efficient and practical stochastic subgradient descent for nuclear norm regularization. In *ICML*, 1231–1238.

Bellet, A.; Habrard, A.; and Sebban, M. 2013. A survey on metric learning for feature vectors and structured data. *CoRR* abs/1306.6709.

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Brand, M. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra Appl.* 415(1):20–30.

Cao, Q.; Ying, Y.; and Li, P. 2012. Distance metric learning revisited. In *ECML PKDD*, 283–298.

Chang, C., and Lin, C. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2(3):27.

Chen, J.; Yang, T.; and Zhu, S. 2014. Efficient low-rank stochastic gradient descent methods for solving semidefinite programs. In *AISTATS*, 122–130.

Cheng, L. 2013. Riemannian similarity learning. In *ICML*, 540–548.

Cotter, A.; Shamir, O.; Srebro, N.; and Sridharan, K. 2011. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, 1647–1655.

Davis, J. V., and Dhillon, I. S. 2008. Structured metric learning for high dimensional problems. In *SIGKDD*, 195–203.

Davis, J. V.; Kulis, B.; Jain, P.; Sra, S.; and Dhillon, I. S. 2007. Information-theoretic metric learning. In *ICML*, 209–216.

Frome, A.; Singer, Y.; Sha, F.; and Malik, J. 2007. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 1–8.

Globerson, A., and Roweis, S. T. 2005. Metric learning by collapsing classes. In *NIPS*, 451–458.

Guyon, I.; Gunn, S. R.; Ben-Hur, A.; and Dror, G. 2004. Result analysis of the NIPS 2003 feature selection challenge. In *NIPS*, 545–552.

Ha, J.; Rossbach, C. J.; Davis, J. V.; Roy, I.; Ramadan, H. E.; Porter, D. E.; Chen, D. L.; and Witchel, E. 2007. Improved error reporting for software that uses black-box components. In *PLDI*, 101–111.

Hazan, E., and Kale, S. 2012. Projection-free online learning. In *ICML*.

Jaggi, M. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*, 427–435.

Kato, T., and Nagano, N. 2010. Metric learning for enzyme active-site search. *Bioinformatics* 26(21):2698–2704.

Kedem, D.; Tyree, S.; Weinberger, K. Q.; Sha, F.; and Lanckriet, G. R. G. 2012. Non-linear metric learning. In *NIPS*, 2582–2590.

Kulis, B. 2013. Metric learning: A survey. *Foundations and Trends in Machine Learning* 5(4):287–364.

Lang, K. 1995. Newsweeder: Learning to filter netnews. In *ICML*, 331–339.

Lim, D.; Lanckriet, G. R. G.; and McFee, B. 2013. Robust structural metric learning. In *ICML*, 615–623.

Liu, W.; Mu, C.; Ji, R.; Ma, S.; Smith, J. R.; and Chang, S. 2015. Low-rank similarity metric learning in high dimensions. In *AAAI*, 2792–2799.

Mahdavi, M.; Jin, R.; Zhu, S.; and Yi, J. 2012. Stochastic gradient descent with only one projection. In *NIPS*, 503–511.

Nesterov, Y. 2004. *Introductory lectures on convex optimization: a basic course*, volume 87 of *Applied optimization*. Kluwer Academic Publishers.

Qi, G.; Tang, J.; Zha, Z.; Chua, T.; and Zhang, H. 2009. An efficient sparse metric learning in high-dimensional space via l_1-penalized log-determinant regularization. In *ICML*, 841–848.

Qian, Q.; Jin, R.; Yi, J.; Zhang, L.; and Zhu, S. 2015a. Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (SGD). *MLJ* 99(3):353–372.

Qian, Q.; Jin, R.; Zhang, L.; and Zhu, S. 2015b. Towards making high dimensional distance metric learning practical. *CoRR* abs/1509.04355.

Rakhlin, A.; Shamir, O.; and Sridharan, K. 2012. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*.

Schultz, M., and Joachims, T. 2003. Learning a distance metric from relative comparisons. In *NIPS*, 41–48.

Shalev-Shwartz, S.; Shamir, O.; Srebro, N.; and Sridharan, K. 2009. Stochastic convex optimization. In *COLT*.

Shalev-Shwartz, S.; Singer, Y.; and Ng, A. Y. 2004. Online and batch learning of pseudo-metrics. In *ICML*, 94–101.

Shamir, O., and Zhang, T. 2013. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 71–79.

Shaw, B.; Huang, B. C.; and Jebara, T. 2011. Learning a distance metric from a network. In *NIPS*, 1899–1907.

Tsagkatakis, G., and Savakis, A. E. 2010. Manifold modeling with learned distance in random projection space for face recognition. In *ICPR*, 653–656.

Weinberger, K. Q., and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *JMLR* 10:207–244.

Xing, E. P.; Ng, A. Y.; Jordan, M. I.; and Russell, S. J. 2002. Distance metric learning with application to clustering with side-information. In *NIPS*, 505–512.

Yang, L., and Jin, R. 2006. Distance metric learning: A comprehensive survey. Technical report, Michigan State University.

Ying, Y., and Li, P. 2012. Distance metric learning with eigenvalue optimization. *JMLR* 13:1–26.

Zhang, L.; Yang, T.; Yi, J.; Jin, R.; and Zhou, Z.-H. 2016. Stochastic optimization for kernel pca. In *AAAI*, 2315–2322.