Mach Learn (2015) 99:353–372
DOI 10.1007/s10994-014-5456-x

Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (SGD)

Qi Qian · Rong Jin · Jinfeng Yi · Lijun Zhang · Shenghuo Zhu

Received: 25 May 2013 / Accepted: 2 June 2014 / Published online: 9 July 2014

Abstract Distance metric learning (DML) is an important task that has found applications in many domains. The high computational cost of DML arises from the large number of variables to be determined and the constraint that a distance metric has to be a positive semi-definite (PSD) matrix. Although stochastic gradient descent (SGD) has been successfully applied to improve the efficiency of DML, it can still be computationally expensive in order to ensure that the solution is a PSD matrix. It has to, at *every iteration*, project the updated distance metric onto the PSD cone, an expensive operation. We address this challenge by developing two strategies within SGD, i.e. mini-batch and adaptive sampling, to effectively reduce the number of updates (i.e. projections onto the PSD cone) in SGD. We also develop hybrid approaches that combine the strength of adaptive sampling with that of mini-batch online learning techniques to further improve the computational efficiency of SGD for DML. We prove the theoretical guarantees for both adaptive sampling and mini-batch based approaches for DML. We also conduct an extensive empirical study to verify the effectiveness of the proposed algorithms for DML.

Editor: Shai Shalev-Shwartz.

Q. Qian (✉) · R. Jin · J. Yi · L. Zhang
Department of Computer Science and Engineering, Michigan State University,
East Lansing, MI 48824, USA
e-mail: qianqi@cse.msu.edu

R. Jin
e-mail: rongjin@cse.msu.edu

J. Yi
e-mail: yijinfen@cse.msu.edu

L. Zhang
e-mail: zhanglij@cse.msu.edu

S. Zhu
NEC Laboratories America, Cupertino, CA 95014, USA
e-mail: zsh@nec-labs.com

1 Introduction

Distance metric learning (DML) is an important subject in machine learning, and has found applications in many domains, including information retrieval (He et al. 2004), supervised classification (Weinberger and Saul 2009), clustering (Xing et al. 2002), and semi-supervised clustering (Chang and Yeung 2004). The objective of DML is to learn a distance metric consistent with a given set of constraints, namely minimizing the distances between pairs of data points from the same class and maximizing the distances between pairs of data points from different classes. The constraints are often specified in the form of must-links, where data points belong to the same class, and cannot-links, where data points belong to different classes. The constraints can also be specified in the form of triplets $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ (Weinberger and Saul 2009), in which $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to a class different from that of $\mathbf{x}_k$ and therefore $\mathbf{x}_i$ and $\mathbf{x}_j$ should be separated by a distance smaller than that between $\mathbf{x}_i$ and $\mathbf{x}_k$. In this work, we focus on DML using triplet constraints due to its encouraging performance (Weinberger and Saul 2009; Chechik et al. 2010; Shaw et al. 2011).

The main computational challenge in DML arises from the restriction that the learned distance metric must be a positive semi-definite (PSD) matrix, which is often referred as the *PSD constraint*. Early approach (Xing et al. 2002) addressed the PSD constraint by exploring the technique of semi-definite programming (SDP) (Boyd and Vandenberghe 2004), which unfortunately does not scale to large and high dimensional datasets. More recent approaches (Jain et al. 2008; Shaw et al. 2011) addressed this challenge by exploiting the techniques of online learning and stochastic optimization, particularly stochastic gradient descent (SGD), that only needs to deal with one constraint at each iteration. Although these approaches are significantly more efficient than the early approach, they share one common drawback: in order to ensure that the learned distance metric is PSD, these approaches require, *at each iteration*, projecting the updated distance metric onto the PSD cone. The projection step requires performing the eigen-decomposition for a given matrix, and therefore is computationally expensive [1]. As a result, the key challenge in developing efficient SGD algorithms for DML is how to reduce the number of projections without affecting the performance of DML.

A common approach for reducing the number of updates and projections in DML is to use the non-smooth loss function. A popular choice of the non-smooth loss function is the hinge loss, whose derivative becomes zero when the input value exceeds a certain threshold. Many online learning algorithms for DML (Davis et al. 2007; Jain et al. 2008; Chechik et al. 2010) take advantage of the non-smooth loss function to reduce the number of updates and projections. In Shaw et al. (2011), the authors proposed a structure preserving metric learning algorithm (SPML) that combines a mini-batch strategy with the hinge loss to further reduce the number of updates for DML. It groups multiple constraints into a mini-batch and performs only one update of the distance metric for each mini-batch. But, according to our empirical study, although SPML reduces the running time of the standard SGD algorithm, it results in a significantly worse performance for several datasets, due to the deployment of the mini-batch strategy.

In this work, we first develop a new mini-batch based SGD algorithm for DML, termed **Mini-SGD**. Unlike SPML that relies on the hinge loss, the proposed Mini-SGD algorithm exploits a *smooth* loss function for DML. By using a smooth loss function, the proposed algorithm is able to effectively take advantage of the reduction in the variance of gradients

[1] The computational cost is $O(d^2)$ if we only need to compute the top eigenvectors of the distance metric and becomes $O(d^3)$ if all the eigenvalues and eigenvectors have to be computed for the projection step, where d is the dimensionality of the data.

achieved by the mini-batch, which in return leads to a better regret bound for online learning (Cotter 2011) and consequentially a more accurate prediction for the learned distance metric. We show theoretically that by using a smooth loss function, Mini-SGD is able to achieve similar convergence rate as the standard SGD algorithm but with significantly less number of updates. The second contribution of this work is to develop a new strategy, termed **adaptive sampling**, for reducing the number of projections in DML. The key idea of adaptive sampling is to first measure the "difficulty" in classifying a constraint using the learned distance metric, and then perform stochastic updating based on the difficulty measure. Finally, we develop two **hybrid approaches** that combine adaptive sampling with mini-batch to further improve the computational efficiency of SGD for DML. We conduct an extensive empirical study to verify the effectiveness and efficiency of the proposed algorithms for DML. We summarize the main contribution of this work as follows:

- To the best of our knowledge, this is the first work that exploits the combination of the mini-batch strategy with smooth loss function for DML. We verify, both theoretically and empirically, the efficiency and effectiveness of the mini-batch strategy with smooth loss for DML.
- We propose an adaptive sampling approach for efficient DML. We verify, both theoretically and empirically, the efficiency and effectiveness of the adaptive sampling approach for DML.
- We present two hybrid approaches that exploit the combination of the mini-batch strategy with adaptive sampling for DML. Our extensive empirical study verifies that the hybrid approaches are significantly more efficient than both the mini-batch strategy and the adaptive sampling approach.

The rest of the paper is organized as follows: Sect. 2 reviews the related work on distance metric learning and stochastic gradient descent with reduced number of projection steps. Section 3 describes the proposed SGD algorithms for DML based on mini-batch and adaptive sampling. Two hybrid approaches are presented that combine mini-batch and adaptive sampling for DML. The theoretical guarantees for both mini-batch based and adaptive sampling based SGD are also presented in Sect. 3. Section 4 summarizes the results of the empirical study, and Sect. 5 concludes this work with future directions.

2 Related work

Many algorithms have been developed to learn a linear distance metric from pairwise constraints, where must-links include pairs of data points from the same class and cannot-links include pairs of data points from different classes ((Yang and Jin 2006) and references therein). Besides pairwise constraints, an alternative strategy is to learn a distance metric from a set of triplet constraints $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$, $t = 1, \ldots, N$, where $\mathbf{x}_i^t$ is expected to be closer to $\mathbf{x}_j^t$ than to $\mathbf{x}_k^t$. Previous studies (Weinberger and Saul 2009; Chechik et al. 2010; Shaw et al. 2011) showed that triplet constraints could be more effective for DML than pairwise constraints.

Several online algorithms have been developed to reduce the computational cost of DML (Globerson and Roweis 2005; Davis et al. 2007; Jain et al. 2008). Most of these methods are based on stochastic gradient descent. At each iteration, they randomly sample *one* constraint, and update the distance metric based on the sampled constraint. The updated distance metric is further projected onto the PSD cone to ensure that it is PSD. Although these approaches are significantly more scalable than the batch learning algorithms for DML (Weinberger and Saul 2009), they suffer from the high computational cost in the projection step

that has to be performed at *every* iteration. One way to improve the efficiency of SGD for DML is to reduce the dimensionality d of the data as the computational cost of projection is at least $O(d^2)$ (Weinberger and Saul 2009). The main problem with these approaches is that they often result in a significantly worse performance than using the original data (Davis and Dhillon 2008).

An alternative approach for improving the efficiency of SGD for DML is to reduce the number of projections. A common approach for reducing the number of projections is to use a non-smooth loss function, such as the hinge loss. In addition, in Shaw et al. (2011), the authors proposed a structure preserving metric learning (SPML) that combines mini-batch with the hinge loss to further reduce the number of projections. The main problem with SPML is its relatively poor performance compared to the standard SGD algorithm. This is because in theory, the mini-batch strategy only works well with a smooth loss (Cotter 2011). Since the hinge loss is a non-smooth loss function, combining mini-batch with the hinge loss may result in a suboptimal performance. This is verified by our empirical study in which we observed that the distance metric learned by SPML performs significantly worse than that learned by the standard stochastic gradient descent method. We resolve this problem by presenting a new SGD algorithm for DML that combines mini-batch with a smooth loss, instead of the hinge loss.

Finally, it is worthwhile mentioning several recent studies proposed to avoid projections in SGD. In Hazan and Kale (2012), the authors developed a projection free SGD algorithm that replaces the projection step with a constrained linear programming problem. In Mahdavi et al. (2012), the authors proposed a SGD algorithm with only one projection that is performed at the end of the iterations. Unfortunately, the improvement of the two algorithms in computational efficiency is limited, because they require computing, *at each iteration*, the minimum eigenvalue and eigenvector of the updated distance metric, an operation with $O(d^2)$ cost, where d is the dimensionality of the data.

3 Improved SGD for DML by mini-batch and adaptive sampling

We first review the basic framework of DML with triplet constraints. We then present two strategies to improve the computational efficiency of SGD for DML, one by mini-batch and the other by adaptive sampling. We present the theoretical guarantees for both strategies, and defer more detailed analysis to the appendix. At the end of this section, we present two hybrid approaches that combine mini-batch with adaptive sampling for more efficient DML.

3.1 DML with triplet constraints

Let $\mathcal{X} \subset \mathbb{R}^d$ be the domain for input patterns, where d is the dimensionality. For the convenience of analysis, we assume all the input patterns with bounded norm, i.e. $\forall \mathbf{x} \in \mathcal{X}$, $|\mathbf{x}|_2 \leq r$. Given a distance metric $M \in \mathbb{R}^{d\times d}$, the distance square between $\mathbf{x}_a$ and $\mathbf{x}_b$, denoted by $|\mathbf{x}_a - \mathbf{x}_b|_M^2$, is measured by

$$|\mathbf{x}_a - \mathbf{x}_b|_M^2 = (\mathbf{x}_a - \mathbf{x}_b)^\top M(\mathbf{x}_a - \mathbf{x}_b)$$

Let $\Omega = \{M : M \succeq 0, \|M\|_F \leq R\}$ be the domain for distance metric M, where R specifies the domain size. Let $\mathcal{D} = \{(\mathbf{x}_i^1, \mathbf{x}_j^1, \mathbf{x}_k^1), \ldots, (\mathbf{x}_i^N, \mathbf{x}_j^N, \mathbf{x}_k^N)\}$ be the set of triplet constraints used for DML, where $\mathbf{x}_i^t$ is expected to be closer to $\mathbf{x}_j^t$ than to $\mathbf{x}_k^t$. Let $\ell(z)$ be the convex loss

function. Define $\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M)$ as

$$\begin{aligned}\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M) &= |\mathbf{x}_i^t - \mathbf{x}_k^t|_M^2 - |\mathbf{x}_i^t - \mathbf{x}_j^t|_M^2 \\ &= \left\langle M, (\mathbf{x}_i^t - \mathbf{x}_k^t)(\mathbf{x}_i^t - \mathbf{x}_k^t)^\top - (\mathbf{x}_i^t - \mathbf{x}_j^t)(\mathbf{x}_i^t - \mathbf{x}_j^t)^\top \right\rangle = \langle M, A_t \rangle\end{aligned}$$

where

$$A_t = (\mathbf{x}_i^t - \mathbf{x}_k^t)(\mathbf{x}_i^t - \mathbf{x}_k^t)^\top - (\mathbf{x}_i^t - \mathbf{x}_j^t)(\mathbf{x}_i^t - \mathbf{x}_j^t)^\top$$

Given the triplet constraints in $\mathcal{D}$ and the domain in Ω, we learn an optimal distance metric $M \in \mathbb{R}^{d\times d}$ by solving the following optimization problem

$$\min_{M\in\Omega} \widehat{\mathcal{L}}(M) = \frac{1}{N}\sum_{t=1}^N \ell\left(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M)\right) \tag{1}$$

We also define the expectation of the loss function as

$$\mathcal{L}(M) = \mathrm{E}\left[\ell(\Delta(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k; M))\right], \tag{2}$$

where the expectation is taken over $\mathbf{x}_i, \mathbf{x}_j$ and $\mathbf{x}_k$.

The key idea of online DML is to minimize the empirical loss $\widehat{\mathcal{L}}(M)$ by updating the distance metric based on one sampled constraint at each iteration. More specifically, at iteration t, it samples a triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$, and updates the distance metric M_t to M_{t+1} by

$$M_{t+1} = \Pi_\Omega\left(M_t - \eta\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))A_t\right),$$

where $\eta > 0$ is the step size, $\ell'(\cdot)$ is the derivative and $\Pi_\Omega(M)$ projects a matrix M onto the domain Ω. The following proposition shows $\Pi_\Omega(M)$ can be computed in two steps, i.e. first projecting M onto the PSD cone, and then scaling the projected M to fit in with the constraint $\|M\|_F \le R$.

Proposition 1 *Boyd and Vandenberghe (2004) We have*

$$\Pi_\Omega(M) = \frac{1}{\max(\|M'\|_F/R, 1)} P(M).$$

Here $P(M)$ projects matrix M onto the PSD cone and is computed as

$$P(M) = \sum_{i=1}^d \max(\lambda_i, 0)\mathbf{v}_i\mathbf{v}_i^\top,$$

where $(\lambda_i, \mathbf{v}_i), i = 1, \ldots, d$ are the eigenvalues and corresponding eigenvectors of M.

As indicated by Proposition 1, $\Pi_\Omega(M)$ requires projecting distance metric M onto the PSD cone, an expensive operation that requires eigen-decomposition of M.

Finally, we approximate the hinge loss by a smooth loss in our study

$$\ell(z) = \frac{1}{L}\log(1 + \exp(-L(z-1))) \tag{3}$$

where $L > 0$ is a parameter that controls the approximation error: the larger the L, the closer $\ell(z)$ is to the hinge loss. Note that the smooth approximation of the hinge loss was first suggested in Zhang and Oles (2001) for classification and was later verified by an empirical study in Zhang et al. (2003). The key properties of the loss function $\ell(z)$ in (3) are given in the following proposition.

Algorithm 1 Mini-batch Stochastic Gradient Descent (Mini-SGD) for DML

1: **Input:** triplet constraints $\{(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)\}_{t=1}^N$, step size η, mini-batch size b, and domain size R
2: Initialize $M_1 = I$ and $T = N/b$
3: **for** $t = 1, \ldots, T$ **do**
4: Sample b triplet constraints $\{(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s})\}_{s=1}^b$
5: Update the distance metric by

$$M_{t+1} = \Pi_\Omega \left(M_t - \eta \nabla \ell_t(M_t)\right)$$

6: **end for**
7: **return** $\bar{M} = \frac{1}{T}\sum_{t=1}^T M_t$

Proposition 2 *For the loss function defined in (3), we have*

$$\forall z \in \mathbb{R}, \quad |\ell'(z)| \leq 1, \ |\ell'(z)| \leq L\ell(z)$$

Compared to the hinge loss function, the main advantage of the loss function in (3) is that it is a smooth loss function. As will be revealed by our analysis, it is the smoothness of the loss function that allows us to effectively explore both the mini-batch and adaptive sampling strategies for more efficient DML without having to sacrifice the prediction performance.

3.2 Mini-batch SGD for DML (Mini-SGD)

Mini-batch SGD improves the computational efficiency of online DML by grouping multiple constraints into a mini-batch and only updating the distance metric once for each mini-batch. For brevity, we will refer to this algorithm as **Mini-SGD** for the rest of the paper.

Let b be the batch size. At iteration t, it samples b triplet constraints, denoted by

$$(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}), s = 1, \ldots, b,$$

and defines the mini-batch loss at iteration t as

$$\ell_t(M_t) = \frac{1}{b}\sum_{s=1}^b \ell\left(\Delta(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}; M_t)\right)$$

Mini-batch DML updates the distance metric M_t to M_{t+1} using the gradient of the mini-bach loss function $\ell_t(M)$, i.e.,

$$M_{t+1} = \Pi_\Omega \left(M_t - \eta \nabla \ell_t(M_t)\right)$$

Algorithm 1 gives the detailed steps of Mini-SGD for DML, where in step 5 Proposition 1 is used to efficiently compute the projection $\Pi_\Omega(\cdot)$.

The theorem below provides the theoretical guarantee for the Mini-SGD algorithm for DML using the smooth loss function defined in (3).

Theorem 1 *Let $\bar{M}$ be the solution output by Algorithm 1 that uses the loss function defined in (3). Let M_* be the optimal solution that minimizes $\mathcal{L}(M)$. Assuming $\|A_t\|_F \leq A$ for any triplet constraint and $\eta < 1/(3LA^2)$, we have*

$$\mathrm{E}[\mathcal{L}(\bar{M})] \leq \frac{\mathcal{L}(M_*)}{1 - 3\eta LA^2} + \frac{bR^2}{2(1 - 3\eta LA^2)\eta N} \tag{4}$$

where the expectation is taken over the sequence of triplet constraints.

Remark 1 First, we observe that the second term in the upper bound in (4), i.e., $bR^2/[2(1-3\eta LA^2)\eta N]$, has a linear dependence on mini-batch size b, implying that the larger the b, the less accurate the distance metric learned by Algorithm 1. Hence, by adjusting parameter b, the size of mini-batch, we are able to make appropriate tradeoff between the prediction accuracy and the computational efficiency: the smaller the b, the more accurate the distance metric but with more updates and consequentially higher computational cost. When $\mathcal{L}(M_*) = 0$, we have $\mathrm{E}[\mathcal{L}(\bar{M})] = O(1/N)$, i.e. the expected prediction error will be reduced at the rate of b/N, significantly faster than that of the mini-batch SGD algorithm (i.e. $O(1/\sqrt{N})$) given in Cotter (2011). Second, if we set η as:

$$\eta = \frac{\rho}{3LA^2(1+2\rho)} \quad where \quad \rho = \frac{3bLR^2A^2}{N\mathcal{L}(M_*)} \tag{5}$$

we have

$$\mathrm{E}[\mathcal{L}(\bar{M})] \leq 2\mathcal{L}(M_*) + \frac{6bLA^2R^2}{N} \tag{6}$$

Although the step size in (5) requires the knowledge of $\mathcal{L}(M_*)$ that is usually unavailable, as suggested in Jin (2013), $\mathcal{L}(M_*)$ can be estimated empirically using part of training examples. Third, the bound in (4) reveals the importance of using a smooth loss function as the last term in (4) is proportional to L that measures the smoothness of the loss function. As a result, using a non-smooth loss function (e.g. hinge loss) in DML will not be able to benefit the advantage of the mini-batch strategy. Finally, unlike the analysis in Shaw et al. (2011) (i.e. Theorem 2) that only consider the case when $b = 1$, Theorem 1 provide a general result for any mini-batch size b.

3.3 Adaptive sampling based SGD for DML (AS-SGD)

We now develop a new approach for reducing the number of updates in SGD in order to improve the computational efficiency of DML. Instead of updating the distance metric at each iteration, the proposed strategy introduces a random binary variable to decide if the distance metric M_t will be updated given a triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$. More specifically, it computes the derivative $\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))$, and samples a random variable Z_t with probability

$$\Pr(Z_t = 1) = |\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))|$$

The distance metric will be updated only when $Z_t = 1$. According to Proposition 2, we have $|\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))| \leq L\ell(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))$ for the smooth loss function given in (3), implying that a triplet constraint has a high chance to be used for updating the distance metric if it has a large loss. Therefore, the essential idea of the proposed adaptive sampling strategy is to give a large chance to update the distance metric when the triplet is difficult to be classified and a low chance when the triplet can be classified correctly with large margin. We note that an alternative strategy is to sample a triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$ base on its loss $\ell(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))$. We did not choose the loss as the basis for updating because it is the derivative, not the loss, that will be used by SGD for updating the distance metric. The detailed steps of adaptive sampling based SGD for DML is given in Algorithm 2. We refer to this algorithm as **AS-SGD** for short in the rest of this paper.

The theorem below provides the performance guarantee for AS-SGD. It also bounds the number of updates $\sum_{t=1}^{T} Z_t$ for AS-SGD.

Algorithm 2 Adaptive Sampling Stochastic Gradient Descent (AS-SGD) for DML

1: **Input:** triplet constraints $\{(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)\}_{t=1}^N$, step size η, and domain size R
2: Initialize $M_1 = I$
3: **for** $t = 1, \ldots, N$ **do**
4: Sample a binary random variable Z_t with

$$\Pr(Z_t = 1) = |\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t)|$$

5: **if** $Z_t = 1$ **then**
6: Update the distance metric by

$$\tau_t = \text{sign}(\ell'((\mathbf{x}_\mathbf{i}^\mathbf{t}, \mathbf{x}_\mathbf{j}^\mathbf{t}, \mathbf{x}_\mathbf{k}^\mathbf{t}; \mathrm{M_t})), \quad \mathrm{M_{t+1}} = \dot{} (\mathrm{M_t} - \eta\tau_\mathrm{t}\mathrm{A_t})$$

7: **end if**
8: **end for**
9: **return** $\bar{M} = \frac{1}{N}\sum_{t=1}^N M_t$

Theorem 2 *Let $\bar{M}$ be the solution output by Algorithm 2 that uses the loss function defined in (3). Let M_* be the optimal solution that minimizes $\mathcal{L}(M)$. Assuming $\|A_t\|_F \leq A$ for any triplet constraint and $\eta < 2/LA^2$, we have*

$$\mathrm{E}\left[\mathcal{L}(\bar{M})\right] \leq \frac{\mathcal{L}(M_*)}{1 - \eta LA^2/2} + \frac{R^2}{2\eta N(1 - \eta LA^2/2)} \tag{7}$$

and the number of updates bounded by

$$\mathrm{E}\left[\sum_{t=1}^N Z_t\right] \leq \frac{NL\mathcal{L}(M_*)}{1 - \eta LA^2/2} + \frac{LR^2}{2\eta(1 - \eta LA^2/2)}, \tag{8}$$

where the expectation is taken over both the binary random variables $\{Z_t\}_{t=1}^N$ and the sequence of triplet constraints.

Remark 2 If we set η as

$$\eta = \frac{2\rho}{LA^2(1 + 2\rho)} \quad where \quad \rho = \frac{LA^2R^2}{4N\mathcal{L}(M_*)}$$

we have

$$\mathrm{E}[\mathcal{L}(\bar{M})] \leq 2\mathcal{L}(M_*) + \frac{LR^2A^2}{N} \tag{9}$$

and

$$\mathrm{E}\left[\sum_{t=1}^N Z_t\right] \leq 2NL\mathcal{L}(M_*) + L^2A^2R^2 \tag{10}$$

The bounds given in (7) and (9) share similar structures as those given in (4) and (6) except that they do not have mini-batch size b that can be used to make tradeoff between the number of updates and the classification accuracy. The number of updates performed by Algorithm 2 is bounded by (10). The dominate term in (10) is $O(\mathcal{L}(M_*)N)$, implying that Algorithm 2 will have a small number of updates if the optimal distance metric only makes a small number of mistakes for the given set of training triplets. In the extreme case when $\mathcal{L}(M_*) \to 0$,

Algorithm 3 A Framework of Hybrid Stochastic Gradient Descent (Hybrid-SGD) for DML

1: **Input:** triplet constraints $\{(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)\}_{t=1}^N$, step size η, mini-batch size b, and domain size R
2: Initialize $M_1 = I$ and $T = N/b$
3: **for** $t = 1, \ldots, T$ **do**
4: Sample b triplets $\{\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}\}_{s=1}^b$
5: Compute sampling probability $\gamma_t(\{\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}\}_{s=1}^b; M_t)$ as in Eqn. 11 or 12
6: Sample a binary random variable Z_t with

$$\Pr(Z_t = 1) = \gamma_t$$

7: **if** $Z_t = 1$ **then**
8: Update the distance metric by

$$\begin{aligned} \tau_t &= 1/\gamma_t \\ M_{t+1} &= \Pi_\Omega(M_t - \eta\tau_t \nabla \ell_t(M_t)) \end{aligned}$$

9: **end if**
10: **end for**
11: **return** $\bar{M} = \frac{1}{T}\sum_{t=1}^T M_t$

the expected number of updates will be bounded by a constant $L^2A^2R^2$. We note that this is consistent with our intuition: it will be easy to learn a good distance metric when the optimal one only makes a few mistakes, and as a result, only a few updates are needed to find a distance metric that are consistent with most of the training triplets. Compared with the result of perceptron method (Shalev-Shwartz and Singer 2004), we do not assume that the dataset is separable, which makes our bound for the number of updates more practically useful.

3.4 Hybrid Approaches: combining mini-batch with adaptive sampling for DML

Since mini-batch and adaptive sampling improve the computational efficiency of SGD from different aspects, it is natural to combine them together for more efficient DML. Similar to the Mini-SGD algorithm, the hybrid approaches will group multiple triplet constraints into a mini-batch. But, unlike Mini-SGD that updates the distance metric for every mini-batch of constraints, the hybrid approaches follow the idea of adaptive sampling, and introduce a binary random variable to decide if the distance metric will be updated for every mini-batch of constraints. By combining the strength of mini-batch and adaptive sampling for SGD, the hybrid approaches are able to make further improvement in the computational efficiency of DML. Algorithm 3 highlights the key steps of the hybrid approaches.

One of the key steps in the hybrid approaches (step 5 in Algorithm 3) is to choose appropriate sampling probability γ_t for every mini-batch constraints $(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s})$, $s = 1, \ldots, b$. In this work, we study two different choices for sampling probability γ_t:

– The first approach chooses γ_t based on a triplet constraint randomly sampled from a mini-batch. More specifically, given a mini-batch of triplet constraints $\{\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}\}_{s=1}^b$, it randomly samples an index s' in the range $[1, b]$. It then sets the sampling probability γ_t to be the derivative for the randomly sampled triplet, i.e.,

$$\gamma_t = |\ell'(\Delta(\mathbf{x}_i^{t,s'}, \mathbf{x}_j^{t,s'}, \mathbf{x}_k^{t,s'}; M_t))| \tag{11}$$

Table 1 Statistics for the ten datasets used in our empirical study

	# class	# feature	# train	# test
semeion	10	256	1,115	478
dna	3	180	2,000	1,186
isolet	26	617	6,238	1,559
tdt30	30	200	6,575	2,819
letter	26	16	15,000	5,000
protein	3	357	17,766	6,621
connect4	3	42	47,289	20,268
sensit	3	100	78,823	19,705
rcv20	20	200	477,141	14,185
poker	10	10	1,000,000	25,010

We refer to this approach as **HR-SGD**.

– The second approach is based on the average case analysis. It sets the sampling probability as the average derivative measured by the norm of the gradient $\nabla \ell_t(M_t)$, i.e.

$$\gamma_t = \frac{1}{W} \|\nabla \ell_t(M_t)\|_F, \tag{12}$$

where $W = \max_t \|\nabla \ell_t(M_t)\|_F$ and is estimated by sampling. We refer to this approach as **HA-SGD**.

4 Experiments

Ten datasets are used to validate the effectiveness of the proposed algorithms. Table 1 summarizes the information of these datasets. Datasets *dna*, *letter* (Hsu and Lin 2002), *protein* and *sensit* (Duarte and Hu 2004) are downloaded from LIBSVM (Chang and Lin 2011). Datasets *tdt30* and *rcv20* are document corpora: *tdt30* is the subset of tdt2 data (Cai et al. 2009) comprised of the documents from the 30 most popular categories and *rcv20* is the subset of a large rcv1 dataset (Bekkerman and Scholz 2008) consisted of documents from the 20 most popular categories. Following Chechik et al. (2010), we reduce the dimensionality of these document datasets to 200 by principle components analysis (PCA). All the other datasets are downloaded directly from the UCI repository (Frank and Asuncion 2010). For all the datasets used in this study, we use the standard training/testing split provided by the original dataset, except for datasets *semeion*, *connect4* and *tdt30* where 70 % of data is randomly selected for training and the remaining 30 % is used for testing. All the experiments are repeated five times, and both the average results and their standard deviation are reported. All the experiments are run on a laptop with 8 GB memory and two 2.50 GHz Intel Core i5-2520M CPUs.

4.1 Parameter setting

The parameter L in the loss function (3) is set to be 3 according to the suggestion in Zhang and Oles (2001). The number of triplet constraints N is set to be 100, 000 for all the datasets except for two small datasets *semeion* and *dna* where $N = 20n$. To construct triplet constraints, we follow the active sampling strategy given in Weinberger and Saul (2009): at each iteration t, we

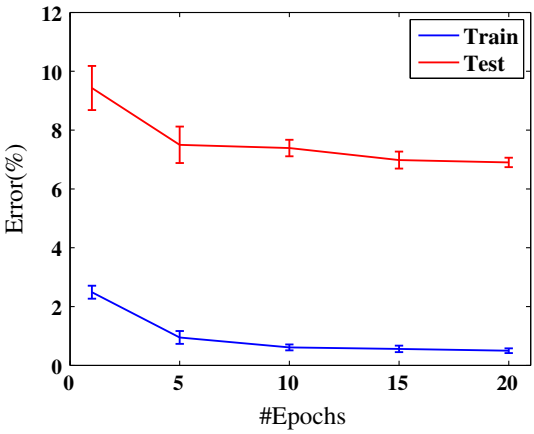

Fig. 1 The training and testing errors over epoches for dataset *dna*

first randomly pick a training example $\mathbf{x}_i^t$, and then $\mathbf{x}_j^t$ from the 3 positive nearest neighbors [2] of $\mathbf{x}_i^t$; we then randomly select a triplet constraint from the set of active constraints [3] involving $\mathbf{x}_i^t$ and $\mathbf{x}_j^t$. We note that this is different from Chechik et al. (2010), where triplet constraints are selected completely randomly. Our empirical study shows that the active sampling strategy is more effective than choosing triplet constraints completely randomly. This is because the actively sampled constraints are more informative to the learned distance metric than a completely random choice. Furthermore, to verify that the choice of N does not lead to the overfitting of training data, particularly for the two small datasets, in Fig. 1, we show the training and test errors for dataset *dna*. It is clear that both errors decline over epoches, suggesting that no overfitting is found even for the small dataset.

For Mini-SGD and the hybrid approaches, we set $b = 10$ for the size of mini-batch as in Shaw et al. (2011), leading to a total of $T = 10,000$ iterations for these approaches. We evaluate the learned distance metric by the classification error of a k-NN on the test data, where the number of nearest neighbors k is set to be 3 based on our experience.

Parameter R in the proposed algorithms determines the domain size for the distance metric to be learned. We observe that the classification error of k-NN remains almost unchanged when varying R in the range of $\{100, 1000, 10000\}$. We thus set $R = 1,000$ for all the experiments. Another important parameter used by the proposed algorithms is the step size η. We evaluate the impact of step size η by measuring the classification error of a k-NN algorithm that uses the distance metric learned by the Mini-SGD algorithm with $\eta = \{0.1, 1, 10\}$. We observe that $\eta = 1$ yields a low classification error for almost all datasets. We thus fix $\eta = 1$ for the proposed algorithms in all the experiments.

4.2 Experiment (I): effectiveness of the proposed SGD algorithms for DML

In this experiment, we compare the performance of the proposed SGD algorithms for DML, i.e. Mini-SGD, AS-SGD and two hybrid approaches (HR-SGD and HA-SGD), to the full version of SGD for DML (SGD). We also include Euclidean distance as the reference method in our comparison. Tables 2 shows the classification error of k-NN ($k = 3$) using the proposed DML algorithms and the baseline algorithms, respectively. First, it is not surprising to observe

[2] $\mathbf{x}_j$ is a positive nearest neighbor of $\mathbf{x}_i$ if $\mathbf{x}_j$ and $\mathbf{x}_i$ share the same class assignment.

[3] A constraint is active if its hinge loss based on the Euclidean distance is non-zero.

Table 2 Classification error (%) of k-NN ($k = 3$) using the distance metrics learned by the proposed SGD methods for DML

	Euclid	SGD	Mini-SGD	AS-SGD	HR-SGD	HA-SGD
semeion	8.79	4.39(0.30)	4.60(0.53)	4.23(0.60)	4.27(0.41)	4.18(0.26)
dna	20.71	6.90(0.16)	6.64(0.33)	7.15(0.42)	6.80(0.21)	6.86(0.15)
isolet	8.98	5.98(0.15)	4.23(0.19)	6.09(0.13)	4.59(0.30)	4.77(0.17)
tdt30	5.96	4.51(0.07)	3.52(0.08)	4.53(0.06)	3.70(0.20)	3.65(0.09)
letter	4.42	2.26(0.09)	2.54(0.06)	2.25(0.10)	2.31(0.08)	2.23(0.07)
protein	49.95	39.46(0.42)	38.16(0.24)	39.49(0.51)	40.76(0.20)	40.03(0.30)
connect4	29.48	20.16(0.08)	20.20(0.08)	20.22(0.12)	21.45(0.71)	20.41(0.14)
sensit	27.28	23.62(0.04)	22.95(0.07)	23.70(0.06)	23.39(0.20)	23.33(0.18)
rcv20	9.13	7.76(0.16)	8.42(0.04)	7.74(0.11)	8.40(0.04)	8.37(0.02)
poker	35.89	35.89(0.06)	35.22(0.18)	35.87(0.08)	35.74(0.41)	35.66(0.16)

Standard deviation computed from five trials is included in the parenthesis

that all the distance metric learning algorithms improve the classification performance of k-NN compared to the Euclidean distance. Second, for almost all datasets, we observe that all the proposed DML algorithms (i.e., Mini-SGD, AS-SGD, HR-SGD, and HA-SGD) yield similar classification performance as SGD, the full version of SGD algorithm for DML. This result confirms that the proposed SGD algorithms are effective for DML despite the modifications we made to the SGD algorithm.

4.3 Experiment (II): efficiency of the proposed SGD algorithms for DML

Figure 2 summarizes the running time for the proposed DML algorithms and the baseline SGD algorithm. We note that the running times in Fig. 2 do not take into account the time for constructing triplet constraints since it is shared by all the methods in comparison.

It is not surprising to observe that all the proposed SGD algorithms, including Mini-SGD, AS-SGD, HA-SGD and HR-SGD, significantly reduce the running time of SGD. For instance, for dataset *isolet*, it takes SGD more than 35, 000 seconds to learn a distance metric, while the running time is reduced to less than 4, 000 seconds when applying the proposed SGD algorithms, roughly a factor of 10 reduction in running time. Comparing the running time of AS-SGD to that of Mini-SGD, we observe that each method has its own advantage: AS-SGD is more efficient on dataset *semeion* while Mini-SGD is more efficient on the other datasets. This is because different mechanisms are employed by AS-SGD and Mini-SGD to reduce the computational cost: AS-SGD improves the computational efficiency of DML by skipping the constraints that are easy to be classified, while Mini-SGD improves the the computational efficiency of SGD by performing the updating of distance metric once for multiple triplet constraints. Finally, we observe that the two hybrid approaches that combine the strength of both adaptive sampling and mini-batch SGD, are computationally most efficient for almost all datasets. We also observe that HR-SGD appears to be more efficient than HA-SGD on six datasets and only loses it edge on datasets *protein* and *rcv20*. This is because HR-SGD computes the sampling probability γ_t based on one randomly sampled triplet while HA-SGD needs to compute the average derivative for each mini-batch of triplet constraints for the sampling probability.

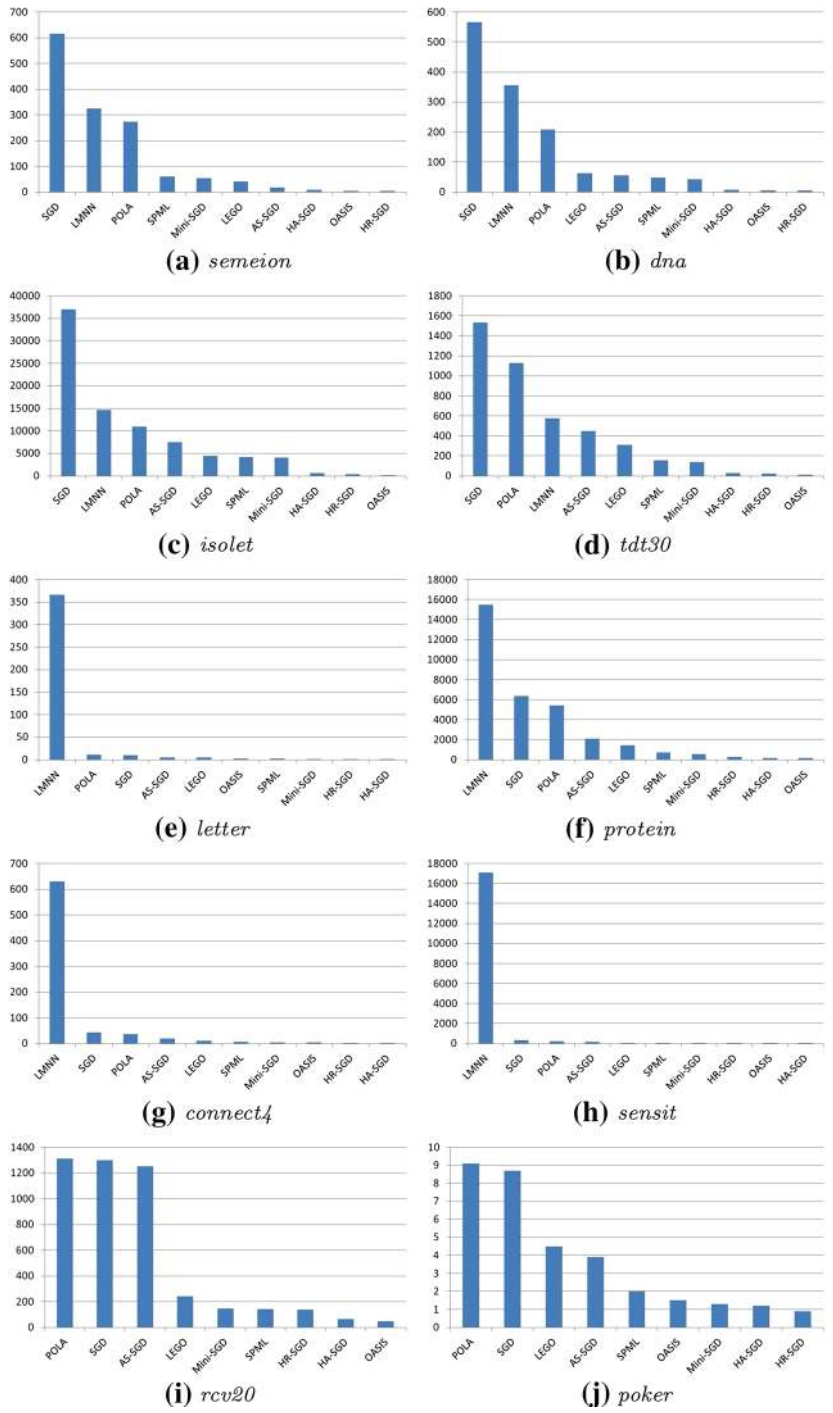

Fig. 2 The comparison of running time (seconds) for various SGD methods. Note that LMNN, a batch DML algorithm, is mainly implemented in C, which is computationally more efficient than our Matlab implementation. All the other methods are implemented in Matlab

To further examine the computational efficiency of proposed SGD algorithms for DML, we summarize in Fig. 3 the number of updates performed by the proposed SGD algorithms and the baseline SGD algorithm, respectively. We observe that all the proposed SGD algorithms for DML are able to reduce the number of updates significantly compared to SGD. Comparing Mini-SGD to AS-SGD, we observe that for *semeion*, the number of updates performed by AS-SGD is significantly less than Mini-SGD, while it is the other way around for the other datasets. This is again due to the fact that AS-SGD and Mini-SGD deploy different mechanisms for reducing computational costs. As we expect, the two hybrid approaches are able to further reduce the number of updates performed by AS-SGD and Mini-SGD, making them more efficient algorithms for DML.

By comparing the running time in Fig. 2 to the number of updates in Fig. 3, we observe that a small number of updates does NOT always guarantee a short running time. This is exhibited by the comparison between the two hybrid approaches: although HA-SGD performs the similar number of updates as HR-SGD on datasets *semeion* and *dna*, it takes HA-SGD significantly longer time to finish the computation than HR-SGD. This is also exhibited by comparing the results across different datasets for a fixed method. For example, for the HA-SGD method, the number of updates for the *protein* dataset is nearly the same as that for the *poker* dataset, but the running time for the *protein* dataset is about 100 times longer than that for the *poker* dataset. This result may sound counter intuitive at the first glance. But, a more careful analysis reveals that in addition to the number of updates, the running time of DML is also affected by the computational cost per iteration, which explains the consistency between Figs. 2 and 3. In the case of comparing the two hybrid approaches, we observe that HA-SGD is subjected to a higher computational cost per iteration than HR-SGD because HA-SGD has to compute the norm of the *average* gradient over each mini-batch while HR-SGD only needs to compute the derivative of *one* randomly sampled triplet constraint for each mini-batch. In the case of comparing the running time across different datasets, the *protein* dataset has a significantly higher dimensionality than the *poker* dataset, and therefore is subjected to a higher computational cost per iteration because the computational cost of projecting an updated distance metric onto the PSD cone increases at least quadratically in the dimensionality.

4.4 Experiment (III): comparison with state-of-the-art online DML methods

We compare the proposed SGD algorithms to three state-of-the-art online algorithms and one bath method for DML:

- **SPML** (Shaw et al. 2011): an online learning algorithm for DML that is based on mini-batch SGD and the hinge loss,
- **OASIS** (Chechik et al. 2010): a state-of-the-art online DML algorithm and symmetric version with only one projection is applied,
- **LEGO** (Jain et al. 2008): an online version of the information theoretic based DML algorithm Davis et al. (2007).
- **POLA** (Shalev-Shwartz and Singer 2004): a Perception based online DML algorithm.

Finally, for sanity checking, we also compare the proposed SGD algorithms to **LMNN** (Weinberger and Saul 2009), a state-of-the-art batch learning algorithm for DML.

Both SPML and OASIS use the same set of triplet constraints to learn a distance metric as the proposed SGD algorithms. Since LEGO and POLA are designed for pairwise constraints, for fair comparison, we generate pairwise constraints for LEGO and POLA by splitting each triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$ into two pairwise constraints: a must-link constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t)$

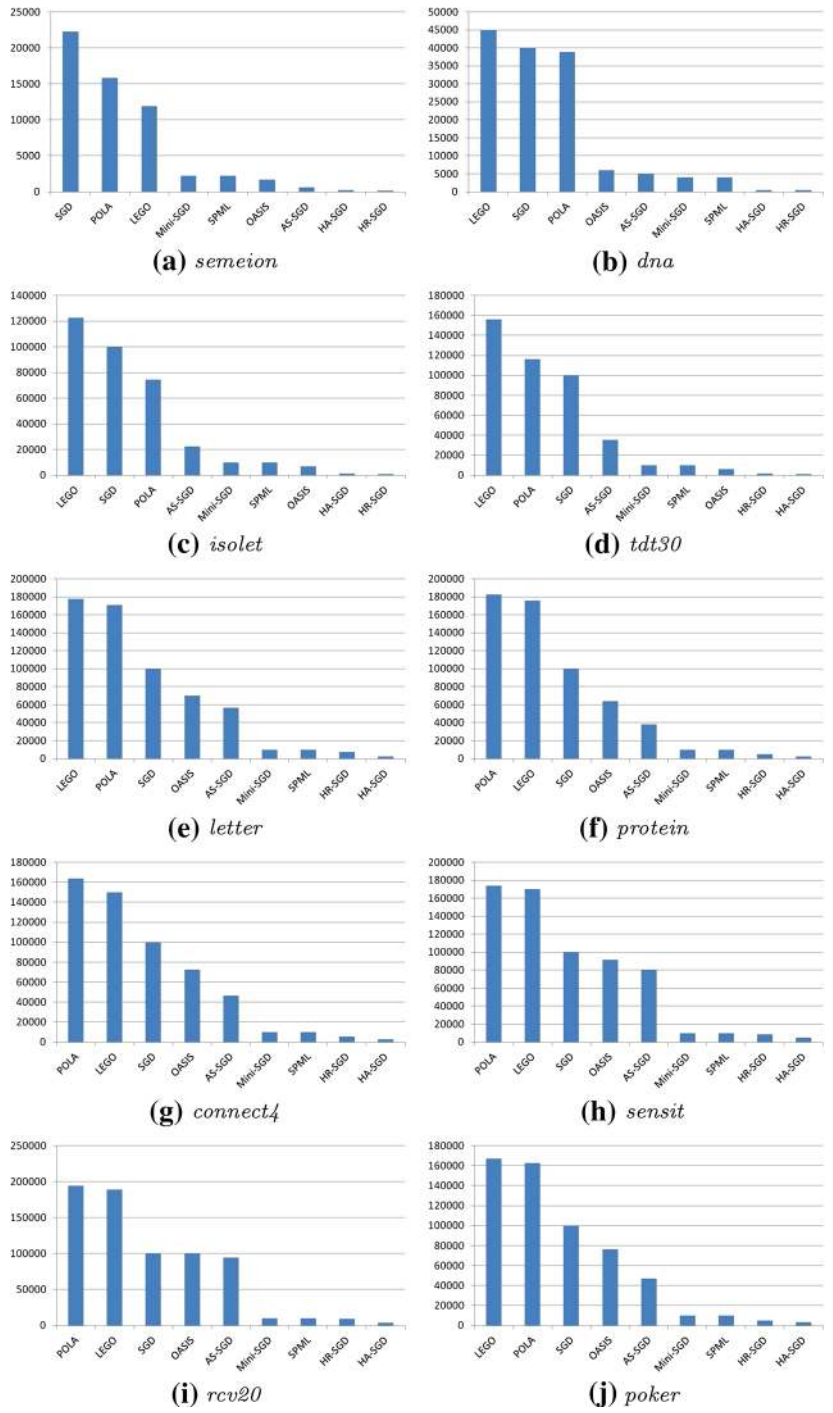

Fig. 3 The comparison of number of updates for various SGD methods. Note that since POLA and LEGO optimize pairwise constraints, we decompose each triplet constraint into two pairwise constraints for these two methods. As a result, the number of constraints is doubled for these two methods

Table 3 Classification error (%) of k-NN ($k = 3$) using the distance metrics learned by baseline SGD method, online learning algorithms and batch learning approach for DML

	Baseline	Batch	Online Learning			
	SGD*	LMNN	POLA	LEGO	OASIS	SPML
semeion	4.18	7.11(0.39)	19.25(1.95)	12.89(1.84)	6.74(0.34)	4.81(0.59)
dna	6.64	4.89(0.29)	7.32(0.55)	7.39(0.55)	11.75(0.43)	6.78(0.58)
isolet	4.23	4.11(0.08)	5.18(0.38)	18.08(6.98)	4.37(0.26)	4.36(0.18)
tdt30	3.52	2.80(0.0)	5.93(0.38)	21.11(3.68)	3.92(0.08)	3.47(0.13)
letter	2.23	3.20(0.00)	3.10(0.22)	5.24(0.45)	3.92(0.05)	3.98(0.53)
protein	38.16	39.86(0.16)	38.38(0.56)	42.60(1.13)	37.83(0.23)	40.12(0.53)
connect4	20.16	21.60(0.26)	25.67(0.85)	26.06(1.30)	22.37(0.63)	24.60(0.70)
sensit	22.95	24.45(0.02)	27.51(0.39)	26.50(1.37)	22.12(0.24)	23.48(0.25)
rcv20	7.74	N/A	7.96(0.08)	8.49(0.18)	8.08(0.06)	8.61(0.12)
poker	35.22	N/A	41.26(1.70)	40.58(1.23)	45.12(2.14)	39.42(0.71)

Standard deviation computed from five trials is included in the parenthesis

and a cannot-link constraint $(\mathbf{x}_i^t, \mathbf{x}_k^t)$. This splitting operation results in a total of $2N$ pairwise constraints for LEGO and POLA. Finally, we note that since LMNN is a batch learning method, it is allowed to utilize *any* triplet constraint derived from the data, and is not restricted to the set of triplet constraints we generate for the SGD methods. All the baseline DML algorithms are implemented by using the codes from the original authors except for SPML, for which we made appropriate changes to the original code in order to avoid large matrix multiplication and improve the computational efficiency. SPML, OASIS LEGO and POLA are implemented in Matlab, while the core parts of LMNN are implemented by C that is usually deemed to be more efficient than Matlab. The default parameters suggested by the original authors are used in the baseline algorithms. The step size of LEGO is set to be 1, as it was observed in Chechik et al. (2010) that the prediction performance of LEGO is in general insensitive to the step size. In all experiments, all the baseline methods set the initial solution for distance metric to be an identity matrix.

Tables 3 summarizes the classification results of k-NN ($k = 3$) using the distance metrics learned by the proposed method and by baseline algorithms, respectively. **SGD*** denotes the best result of propose methods in Table 2. First, we observe that LEGO and POLA perform significantly worse than the proposed DML algorithms for four datasets, including *semeion*, *connect4*, *sensit* and *poker*. LEGO also performs poorly on *isolet* and *tdt30*. This can be explained by the fact that LEGO and POLA use pairwise constraints for DML while the other methods in comparison use triplet constraints for DML. According to Chechik et al. (2010); Shaw et al. (2011); Weinberger and Saul (2009), triplet constraints are in general more effective than pairwise constraints. Second, although both SPML and Mini-SGD are based on the mini-batch strategy, SPML performs significantly worse than Mini-SGD on three datasets, i.e. *protein*, *connect4*, and *poker*. The performance difference between SPML and Mini-SGD can be explained by the fact that Mini-SGD uses a smooth loss function while a hinge loss is used by SPML. According to our analysis and the analysis in Cotter (2011), using a smooth loss function is critical for the success of the mini-batch strategy. Third, OASIS yields similar performance as the proposed algorithms for almost all datasets except for datasets *semeion*, *dna* and *poker*, for which OASIS performs significantly worse. Overall,

Table 4 The comparison of running time (seconds) for OASIS and the hybrid methods. Average results over five trials are reported

Methods	*semeion*	*dna*	*isolet*	*tdt30*	*letter*
OASIS	4.4	5.5	156.6	10.4	2.3
HR-SGD	3.6	5.1	363.2	21.7	1.4
HA-SGD	7.6	8.1	597.9	28.4	1.1
Methods	*protein*	*connect4*	*sensit*	*rcv20*	*poker*
OASIS	161.4	4.5	19.0	46.5	1.5
HR-SGD	275.7	3.0	23.5	139.2	0.9
HA-SGD	164.6	2.5	15.3	65.6	1.2

we conclude that the proposed DML algorithms yield similar, if not better, performance as the state-of-the-art online learning algorithms for DML.

Compared to LMNN, a state-of-the-art batch learning algorithm for DML, we observe that the proposed SGD algorithms yield similar performance on four datasets. They however perform significantly better than LMNN on datasets *semeion* and *letter*, and significantly worse on datasets *dna* and *tdt30*. We attribute the difference in classification error to the fact that the proposed DML algorithms are restricted to 100, 000 randomly sampled triplet constraints while LMNN is allowed to use *all* the triplet constraints that can be derived from the data. The restriction in triplet constraints could sometimes limit the classification performance but at the other time help avoid the overfitting problem. We also observe that LMNN is unable to run on the two large datasets *rcv20* and *poker*, indicating that LMNN does not scale well to the size of datasets.

The running times for the proposed algorithms and the baseline algorithms are summarized in Fig. 2. The number of updates for both groups of algorithms are provided in Fig. 3. It is not surprising to observe that two online DML algorithms (SPML, OASIS) are significantly more efficient than SGD in terms of both running time and the number of updates. We also observe that Mini-SGD and SPML share the same number of updates and similar running time for all datasets because they use the same mini-batch strategy. Furthermore, compared to three online DML algorithms (SPML, LEGO and POLA), the two hybrid approaches are significantly more efficient in both running time and the number of updates. Table 4 compares the detailed running time of OASIS and the hybrid methods. We also observe that the hybrid methods are more efficient than OASIS on six datasets (i.e. *semeion*, *dna*, *letter*, *connect4*, *sensit* and *poker*), even though OASIS only performs projection once at the end of the program. We attribute the efficiency of the hybrid approaches to the reduced number of updates they have to perform on the learned metric. Finally, since LMNN is implemented by C, it is not surprising to observe that LMNN shares similar running time as the other online DML algorithms for relatively small datasets. It is however significantly less efficient than the online learning algorithms for datasets of modest size (e.g. *letter*, *connect4* and *sensit*), and becomes computationally infeasible for the two large datasets *rcv20* and *poker*. Overall, we observe that the two hybrid approaches are significantly more efficient than the other DML algorithms in comparison.

5 Conclusion

In this paper, we propose two strategies to improve the computational efficiency of SGD for DML, i.e. mini-batch and adaptive sampling. The key idea of mini-batch is to group

multiple triplet constraints into a mini-batch, and only update the distance metric once for each mini-batch; the key idea of adaptive sampling is to perform stochastic updating by giving a difficult triplet constraint more chance to be used for updating the distance metric than an easy triplet constraint. We develop theoretical guarantees for both strategies. We also develop two variants of hybrid approaches that combine mini-batch with adaptive sampling for more efficient DML. Our empirical study confirms that the proposed algorithms yield similar, if not better, prediction performance as the state-of-the-art online learning algorithms for DML but with significantly less amount of running time. Since our empirical study is currently limited to datasets with relatively small number of features, we plan to examine the effectiveness of the proposed algorithms for DML with high dimensional data.

Acknowledgments Qi Qian and Rong Jin are supported in part by NSF (IIS-1251031) and ONR (N000141410631).

Appendix 1: Proof of Theorem 1

Using the standard analysis for online learning (Chapter 12, Cesa-Bianchi and Lugosi 2006), we have

$$\begin{aligned}
&\ell_t(M_t) - \ell_t(M_*) \leq \langle M_t - M_*, \nabla \ell_t(M_t)\rangle \\
&\leq \frac{\|M_t - M_*\|_F^2}{2\eta} - \frac{\|M_{t+1} - M_*\|_F^2}{2\eta} - \frac{\|M_t - M_{t+1}\|_F^2}{2\eta} + \langle M_t - M_{t+1}, \nabla \ell_t(M_t)\rangle \\
&\leq \frac{\|M_t - M_*\|_F^2}{2\eta} - \frac{\|M_{t+1} - M_*\|_F^2}{2\eta} - \frac{\|M_t - M_{t+1}\|_F^2}{2\eta} + \frac{1}{b}\sum_{s=1}^{b}\langle M_t - M_{t+1}, \nabla \ell_{t,s}(M_t)\rangle \\
&\leq \frac{\|M_t - M_*\|_F^2}{2\eta} - \frac{\|M_{t+1} - M_*\|_F^2}{2\eta} + \frac{\eta}{2b}\sum_{s=1}^{b}\|\nabla \ell_{t,s}(M_t)\|_F^2
\end{aligned}$$

By taking the expectation with respect to the t-th mini-batch of triplet constraint, we have

$$\mathcal{L}(M_t) - \mathcal{L}(M_*) \leq \frac{\|M_t - M_*\|_F^2}{2\eta} - \frac{\mathrm{E}_t[\|M_{t+1} - M_*\|_F^2]}{2\eta} + \frac{\eta}{2b}\sum_{s=1}^{b}\mathrm{E}_{t,s}[\|\nabla \ell_{t,s}(M_t)\|_F^2]$$

By adding the inequalities of all iterations and taking expectation over the sequence of triplet constraints, we have

$$\sum_{t=1}^{T}\mathrm{E}[\mathcal{L}(M_t)] - \mathcal{L}(M_*) \leq \frac{1}{2\eta}\|M_1 - M_*\|_F^2 + \frac{\eta}{2b}\underbrace{\sum_{t=1}^{T}\sum_{s=1}^{b}\mathrm{E}[\|\nabla \ell_{t,s}(M_t)\|_F^2]}_{:=C_T}$$

According to Proposition 2, we have $\ell'(z)^2 \leq |\ell'(z)| \leq L\ell(z)$. Using $A = \max_{1\leq t\leq N}\|A_t\|_F$, we have

$$C_T = \sum_{t=1}^{T}\sum_{s=1}^{b}\mathrm{E}_{t,s}[\|\nabla \ell_{t,s}(M_t)\|_F^2] \leq LA^2 b\sum_{t=1}^{T}\mathrm{E}[\mathcal{L}(M_t)]$$

Using the result for C_T, we have

$$\left(1 - 3\eta L A^2\right) \mathcal{L}(\bar{M}) \leq \mathcal{L}(M_*) + \frac{R^2}{2\eta T}$$

We complete the proof by dividing both sides with $1 - 3\eta L A^2$ and replacing T with N/b.

Appendix 2: Proof of Theorem 2

To bound the number of updates, we have

$$\mathrm{E}\left[\sum_{t=1}^{N} Z_t\right] = \mathrm{E}\left[\sum_{t=1}^{N} |\ell_t'(M_t)|\right] \leq L\mathrm{E}\left[\sum_{t=1}^{N} \mathcal{L}(M_t)\right],$$

where the last step follows from $|\ell_t'(M)| \leq L\ell_t(M)$.

Using the standard analysis for online learning (Chapter 12, Cesa-Bianchi and Lugosi 2006), we have:

$$\begin{aligned}
&\ell(M_t) - \ell(M_*) \leq \langle \ell'(M_t) A_t, M_t - M_* \rangle \\
&= \tau_t Z_t \langle A_t, M_t - M_* \rangle + (\ell'(M_t) - \tau_t Z_t) \langle A_t, M_t - M_* \rangle \\
&\leq \frac{\|M_t - M_*\|_F^2 - \|M_{t+1} - M_*\|_F^2}{2\eta} + \frac{\eta A^2 Z_t}{2} + \tau_t (|\ell'(M_t)| - Z_t) \langle A_t, M_t - M_* \rangle
\end{aligned}$$

Taking the sum from $t = 1$ to N and expectation over both binary variables $\{Z_t\}_{t=1}^N$ and the sequence of triplet constraints, we have:

$$\sum_{t=1}^{N} \mathrm{E}[\mathcal{L}(M_t)] - \mathcal{L}(M_*) \leq \frac{\|M_1 - M_*\|_F^2}{2\eta} + \frac{\eta A^2}{2} \mathrm{E}\left[\sum_{t=1}^{N} Z_t\right]$$

We complete the proof by reorganizing the above inequality.

References

Bekkerman, R.,& Scholz, M. (2008). Data weaving: Scaling up the state-of-the-art in data clustering. *In CIKM* (pp. 1083–1092).

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Cai, D., Wang, X., & He, X. (2009). Probabilistic dyadic data analysis with local and global consistency. In *ICML* (pp. 105–112).

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge: Cambridge University Press.

Chang, C.-C., & Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM TIST*, *2*(3), 27.

Chang, H., & Yeung, D.-Y. (2004). Locally linear metric adaptation for semi-supervised clustering. In *ICML* (pp. 153–160).

Chechik, G., Sharma, V., Shalit, U., & Bengio, S. (2010). Large scale online learning of image similarity through ranking. *JMLR*, *11*, 1109–1135.

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

Davis, J. V., & Dhillon, I. S. (2008). Structured metric learning for high dimensional problems. In *KDD* (pp. 195–203).

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

Duarte, M. F., & Hu, Y. H. (2004). Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, *64*(7), 826–838.
Frank, A., & Asuncion, A. (2010) UCI machine learning repository. http://archive.ics.uci.edu/ml
Globerson, A., & Roweis, S. T. (2005) Metric learning by collapsing classes. In *NIPS* (p. 451).
Hazan, E., & Kale, S. (2012). Projection-free online learning. In *ICML*.
He, X., Ma, W.-Y., & Zhang, H. (2004). Learning an image manifold for retrieval. In *ACM Multimedia* (pp. 17–23).
Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, *13*(2), 415–425.
Jain, P., Kulis, B., Dhillon, I. S., & Grauman, K. (2008). Online metric learning and fast similarity search. In *NIPS* (pp. 761–768).
Jin, R. (2013). Stochastic optimization of smooth loss. In arXiv:1312.0048.
Mahdavi, M., Yang, T., Jin, R., Zhu, S., & Yi, J. (2012). Stochastic gradient descent with only one projection. In *NIPS* (pp. 503–511).
Shalev-Shwartz, S., Singer, Y. A., & Ng. Y. (2004). In *ICML*. Online and batch learning of pseudo-metrics.
Shaw, B., Huang, B. C.,& Jebara, T. (2011). Learning a distance metric from a network. In *NIPS* (pp. 1899–1907).
Weinberger, K. Q., & 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., & Russell, S. J. (2002). Distance metric learning with application to clustering with side-information. In *NIPS* (pp. 505–512).
Yang, L., & Jin, R. (2006) Distance metric learning: A comprehensive survery. http://www.cs.cmu.edu/~liuy/frame_survey_v2.pdf.
Zhang, J., Jin, R., Yang, Y., & Hauptmann, A. G. (2003). Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. In *ICML* (pp. 888–895).
Zhang, T., & Oles, F. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, *4*(1), 5–31.