Scalable Large-Margin Mahalanobis Distance Metric Learning

Chunhua Shen, Junae Kim, and Lei Wang

***Abstract*—For many machine learning algorithms such as k-nearest neighbor (k-NN) classifiers and k-means clustering, often their success heavily depends on the metric used to calculate distances between different data points. An effective solution for defining such a metric is to learn it from a set of labeled training samples. In this work, we propose a fast and scalable algorithm to learn a Mahalanobis distance metric. The Mahalanobis metric can be viewed as the Euclidean distance metric on the input data that have been linearly transformed. By employing the principle of margin maximization to achieve better generalization performances, this algorithm formulates the metric learning as a convex optimization problem and a positive semidefinite (p.s.d.) matrix is the unknown variable. Based on an important theorem that a p.s.d.trace-one matrix can always be represented as a convex combination of multiple rank-one matrices, our algorithm accommodates any differentiable loss function and solves the resulting optimization problem using a specialized gradient descent procedure. During the course of optimization, the proposed algorithm maintains the positive semidefiniteness of the matrix variable that is essential for a Mahalanobis metric. Compared with conventional methods like standard interior-point algorithms [2] or the special solver used in large margin nearest neighbor [24], our algorithm is much more efficient and has a better performance in scalability. Experiments on benchmark data sets suggest that, compared with state-of-the-art metric learning algorithms, our algorithm can achieve a comparable classification accuracy with reduced computational complexity.**

***Index Terms*—Distance metric learning, large-margin nearest neighbor, Mahalanobis distance, semidefinite optimization.**

I. Introduction

In many machine learning problems, the distance metric used over the input data has critical impact on the success of a learning algorithm. For instance, k-nearest neighbor (k-NN) classification [4], and clustering algorithms such as k-means rely on if an appropriate distance metric is used to faithfully model the underlying relationships between the input data points. A more concrete example is visual object recognition. Many visual recognition tasks can be viewed as inferring a distance metric that is able to measure the (dis)similarity of the input visual data, ideally being consistent with human perception. Typical examples include object categorization [25] and content-based image retrieval [20], in which a similarity metric is needed to discriminate different object classes or relevant and irrelevant images against a given query. As one of the most classic and simplest classifiers, k-NN has been applied to a wide range of vision tasks and it is the classifier that directly depends on a predefined distance metric. An appropriate distance metric is usually needed for achieving a promising accuracy. Previous works (e.g., [26] and [27]) have shown that compared to using the standard Euclidean distance, applying a well-designed distance often can significantly boost the classification accuracy of a k-NN classifier. In this work, we propose a scalable and fast algorithm to learn a Mahalanobis distance metric. Mahalanobis metric removes the main limitation of the Euclidean metric in that it corrects for correlation between the different features.

Recently, much research effort has been spent on learning a Mahalanobis distance metric from labeled data [5], [24], [26], [27]. Typically, a convex cost function is defined such that a global optimum can be achieved in polynomial time. It has been shown in the statistical learning theory [23] that increasing the margin between different classes helps to reduce the generalization error. Inspired by the work of [24], we directly learn the Mahalanobis matrix from a set of *distance comparisons*, and optimize it via margin maximization. The intuition is that such a learned Mahalanobis distance metric may achieve sufficient separation at the boundaries between different classes. More importantly, we address the scalability problem of learning the Mahalanobis distance matrix in the presence of high-dimensional feature vectors, which is a critical issue of distance metric learning. As indicated in a theorem in [19], a positive semidefinite trace-one matrix can always be decomposed as a convex combination of a set of rank-one matrices. This theorem has inspired us to develop a fast optimization algorithm that works in the style of gradient descent. At each iteration, it only needs to find the principal eigenvector of a matrix of size $D \times D$ (D is the dimensionality of the input data) and a simple matrix update. This process incurs much less computational overhead than the metric learning algorithms in the literature [2], [24]. Moreover, thanks to the above theorem, this process automatically preserves the p.s.d.property of the Mahalanobis matrix. To verify its effectiveness and efficiency, the proposed algorithm is tested on a few benchmark data sets and is compared with the state-of-the-art distance metric learning algorithms. As experimentally demonstrated, k-NN with the Mahalanobis distance learned by our algorithms attains comparable (sometimes slightly better) classification accuracy. Meanwhile, in terms of the computation time, the proposed algorithm has much better scalability in terms of the dimensionality of input feature vectors.

We briefly review some related work before we present our method. Given a classification task, some previous work on learning a distance metric aims to find a metric that makes the data in the same class close and separates those in different classes from each other as far as possible. Xing *et al.* [26] proposed an approach to learn a Mahalanobis distance for supervised clustering. It minimizes the sum of the distances

Manuscript received July 21, 2009; revised March 14, 2010; accepted June 2, 2010. Date of publication August 13, 2010; date of current version September 1, 2010. This work was supported in part by the National ICT Australia, funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy, and the Australian Research Council through the Information and Communications Technology Center of Excellence program.

C. Shen is with NICTA, Canberra Research Laboratory, Canberra, ACT 2601, Australia, and also with the Australian National University, Canberra, ACT 0200, Australia (e-mail: chunhua.shen@nicta.com.au).

J. Kim and L. Wang are with the Australian National University, Canberra, ACT 0200, Australia (e-mail: junae.kim@anu.edu.au; lei.wang@anu.edu.au).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2052630

among data in the same class while maximizing the sum of the distances among data in different classes. Their work shows that the learned metric could improve clustering performance significantly. However, to maintain the p.s.d.property, they have used projected gradient descent and their approach has to perform a *full* eigen-decomposition of the Mahalanobis matrix at each iteration. Its computational cost rises rapidly when the number of features increases, and this makes it less efficient in coping with high-dimensional data. Goldberger *et al.* [7] developed an algorithm termed neighborhood component analysis (NCA), which learns a Mahalanobis distance by minimizing the leave-one-out cross-validation error of the k-NN classifier on the training set. NCA needs to solve a nonconvex optimization problem, which might have many local optima. Thus, it is critically important to start the search from a reasonable initial point. Goldberger *et al.*have used the result of linear discriminant analysis as the initial point. In NCA, the variable to optimize is the projection matrix.

The work closest to ours is large margin nearest neighbor (LMNN) [24] in the sense that it also learns a Mahalanobis distance in the large margin framework. In their approach, the distances between each sample and its "target neighbors" are minimized while the distances among the data with different labels are maximized. A convex objective function is obtained and the resulting problem is a semidefinite program (SDP). Since, conventional interior-point based SDP solvers can only solve problems of up to a few thousand variables, LMNN has adopted an alternating projection algorithm for solving the SDP problem. At each iteration, similar to [26], also a full eigen-decomposition is needed. Our approach is largely inspired by their work. Our method differs from LMNN [24] in the following ways. 1) LMNN learns the metric from the pairwise distance information. In contrast, our algorithm uses examples of proximity comparisons among triples of objects (e.g., example i is closer to example j than example k). In some applications like image retrieval, this type of information could be easier to obtain than to tag the actual class label of each training image. Rosales and Fung [18] have used similar information for metric learning. 2) More importantly, we design an optimization method that has a clear advantage on computational efficiency (we only need to compute the leading eigenvector at each iteration). The optimization problems of [18] and [24] are both SDPs, which are computationally heavy. Linear programs are used in [18] to approximate the SDP problem. It remains unclear how well this approximation could achieve. Preliminary results of this work has appeared in [10].

The problem of learning a kernel from a set of labeled data shares similarities with metric learning because the optimization involved has similar formulations. Lanckriet *et al.* [12] and Kulis *et al.* [11] considered learning p.s.d.kernels subject to some pre-defined constraints. An appropriate kernel can often offer algorithmic improvements. It is possible to apply the proposed gradient descent optimization technique to solve the kernel learning problems. We leave this topic for future work.

The rest of this work is organized as follows. Section II presents the convex formulation of learning a Mahalanobis metric. In Section III, we show how to efficiently solve the optimization problem by a specialized gradient descent procedure, which is the main contribution of this work. The performance of our approach is experimentally demonstrated in Section IV. Finally, we conclude this work in Section V.

II. Large-Margin Mahalanobis Metric Learning

In this section, we propose our distance metric learning approach as follows. The intuition is to find a particular distance metric for which the margin of separation between the classes is maximized. In particular, we are interested in learning a quadratic Mahalanobis metric.

Let $\mathbf{a}_i \in \mathbb{R}^D (i = 1, 2, \ldots, n)$ denote a training sample where n is the number of training samples and D is the number of features. To learn a Mahalanobis distance, we create a set $\mathcal{S}$ that contains a group of training triplets as $\mathcal{S} = \{(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k)\}$, where $\mathbf{a}_i$ and $\mathbf{a}_j$ come from the same class and $\mathbf{a}_k$ belongs to different classes. A Mahalanobis distance is defined as follows. Let $\mathbf{P} \in \mathbb{R}^{D\times d}$ denote a linear transformation and **dist** be the squared Euclidean distance in the transformed space. The squared distance between the projections of $\mathbf{a}_i$ and $\mathbf{a}_j$ writes

$$\mathbf{dist}_{ij} = \|\mathbf{P}^\top \mathbf{a}_i - \mathbf{P}^\top \mathbf{a}_j\|_2^2 = (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{P}\mathbf{P}^\top (\mathbf{a}_i - \mathbf{a}_j). \quad (1)$$

According to the class memberships of $\mathbf{a}_i$, $\mathbf{a}_j$, and $\mathbf{a}_k$, we wish to achieve $\mathbf{dist}_{ik} \geq \mathbf{dist}_{ij}$ and it can be obtained as

$$(\mathbf{a}_i - \mathbf{a}_k)^\top \mathbf{P}\mathbf{P}^\top (\mathbf{a}_i - \mathbf{a}_k) \geq (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{P}\mathbf{P}^\top (\mathbf{a}_i - \mathbf{a}_j). \quad (2)$$

It is not difficult to see that this inequality is generally not a convex constrain in $\mathbf{P}$ because the difference of quadratic terms in $\mathbf{P}$ is involved. In order to make this inequality constrain convex, a new variable $\mathbf{X} = \mathbf{P}\mathbf{P}^\top$ is introduced and used throughout the whole learning process. Learning a Mahalanobis distance is essentially learning the Mahalanobis matrix $\mathbf{X}$. (2) becomes linear in $\mathbf{X}$. This is a typical technique to *convexify* a problem in convex optimization [2].

A. Maximization of Soft Margins

In our algorithm, a *margin* is defined as the difference between $\mathbf{dist}_{ik}$ and $\mathbf{dist}_{ij}$, that is

$$\begin{aligned} &\rho_r = (\mathbf{a}_i - \mathbf{a}_k)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_j) \\ &\forall(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{S}, \quad r = 1, 2, \ldots, |\mathcal{S}|. \end{aligned} \quad (3)$$

Similar to the large margin principle that has been widely used in machine learning algorithms such as support vector machines and boosting, here we maximize this margin (3) to obtain the optimal Mahalanobis matrix $\mathbf{X}$. Clearly, the larger is the margin ρ_r, the better metric might be achieved. To enable some flexibility, *i.e.*, to allow some inequalities of (2) not to be satisfied, a soft-margin criterion is needed. Considering these factors, we could define the objective function for learning $\mathbf{X}$ as

$$\begin{aligned} &\max_{\rho,\mathbf{X},\boldsymbol{\xi}} \quad \rho - C\sum_{r=1}^{|\mathcal{S}|} \xi_r, \text{ subject to} \\ &\mathbf{X} \succcurlyeq 0 \quad \mathbf{Tr}(\mathbf{X}) = 1 \\ &\xi_r \geq 0 \quad r = 1, 2, \ldots, |\mathcal{S}| \\ &(\mathbf{a}_i - \mathbf{a}_k)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_j) \geq \rho - \xi_r \\ &\forall(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{S} \end{aligned} \quad (4)$$

where $\mathbf{X} \succcurlyeq 0$ constrains $\mathbf{X}$ to be a p.s.d.matrix and $\mathbf{Tr}(\mathbf{X})$ denotes the trace of $\mathbf{X}$. r indexes the training set $\mathcal{S}$ and $|\mathcal{S}|$ denotes the size of $\mathcal{S}$. C is an algorithmic parameter that balances the violation of (2) and the margin maximization. $\boldsymbol{\xi} \geq 0$ is the slack variable similar to that used in support vector machines and it corresponds to the soft-margin hinge loss. Enforcing $\mathbf{Tr}(\mathbf{X}) = 1$ removes the scale ambiguity because the inequality constrains are scale invariant. To simplify exposition, we define

$$\mathbf{A}^r = (\mathbf{a}_i - \mathbf{a}_k)(\mathbf{a}_i - \mathbf{a}_k)^\top - (\mathbf{a}_i - \mathbf{a}_j)(\mathbf{a}_i - \mathbf{a}_j)^\top. \tag{5}$$

Therefore, the last constraint in (4) can be written as

$$\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle \geq \rho - \xi_r \quad r = 1, \ldots, |S|. \tag{6}$$

Note that this is a linear constraint on $\mathbf{X}$. Problem (4) is thus a typical SDP problem since it has a linear objective function and linear constraints plus a p.s.d.conic constraint. One may solve it using off-the-shelf SDP solvers like CSDP [1]. However, directly solving (4) using those standard interior-point SDP solvers would quickly become computationally intractable with the increasing dimensionality of feature vectors. We show how to efficiently solve (4) in a fashion of first-order gradient descent.

B. Employment of a Differentiable Loss Function

It is proved in [19] that a trace-one p.s.d.matrix can always be decomposed as a linear convex combination of a set of rank-one trace-one matrices. In the context of our problem, this means that $\mathbf{X} = \sum_i \theta_i \mathbf{Z}_i$, where $\mathbf{rank}(\mathbf{Z}_i) = 1$, $\mathbf{Tr}(\mathbf{Z}_i) = 1$, $\theta_i > 0 \quad \forall \theta_i$, and $\sum_i \theta_i = 1$. This important result inspires us to develop a gradient descent based optimization algorithm. In each iteration, $\mathbf{X}$ can be updated as

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \alpha(\delta\mathbf{X} - \mathbf{X}_i) = \mathbf{X}_i + \alpha\mathbf{p}_i \quad 0 \leq \alpha \leq 1 \tag{7}$$

where $\delta\mathbf{X}$ is a rank-one and trace-one matrix. $\mathbf{p}_i$ is the search direction. It is straightforward to verify that $\mathbf{Tr}(\mathbf{X}_{i+1}) = 1$ and $\mathbf{X}_{i+1} \succcurlyeq 0$ hold. This is the starting point of our gradient descent algorithm. With this update strategy, the trace-one and positive semidefinteness of $\mathbf{X}$ is always retained. We show how to calculate this search direction in Algorithm 2. Although it is possible to use subgradient methods to optimize nonsmooth objective functions, we use a differentiable objective function instead so that the optimization procedure is simplified (standard gradient descent can be applied). So, we need to ensure that the objective function is differentiable with respect to the variables ρ and $\mathbf{X}$.

Let $f(\cdot)$ denote the objective function and $\lambda(\cdot)$ be a loss function. Our objective function can be rewritten as

$$f(\mathbf{X}, \rho) = \rho - C \cdot \sum_{r=1}^{|\mathcal{S}|} \lambda\left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right). \tag{8}$$

The above cost function is equivalent to the cost function in (4), which adopts the hinge loss function defined as $\lambda(z) = \max(0, -z)$. However, the hinge loss is not differentiable at the point of $z = 0$, and standard gradient-based optimization cannot be applied directly. In order to make standard gradient descent methods applicable, we propose to use differentiable loss functions, for example, the squared hinge loss or Huber loss functions as discussed below.

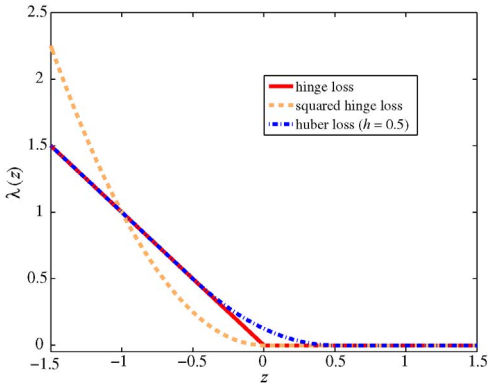

Fig. 1. Hinge loss, squared hinge loss, and Huber loss.

The squared hinge loss function can be represented as

$$\lambda\left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) = \begin{cases} 0, & \text{if } \left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) \geq 0, \\ \left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right)^2, & \text{if } \left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) < 0. \end{cases} \tag{9}$$

As shown in Fig. 1, this function connects the positive and zero segments smoothly and it is differentiable everywhere including the point $z = 0$. We also consider the Huber loss function in this work

$$\lambda\left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) = \begin{cases} 0, & \text{if } \left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) \geq h \\ \frac{(h - (\langle \mathbf{A}^r, \mathbf{X} \rangle - \rho))^2}{4h}, & \text{if } -h < \left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) < h \\ -(\langle \mathbf{A}^r, \mathbf{X} \rangle - \rho), & \text{if } \left(\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho\right) \leq -h \end{cases} \tag{10}$$

where h is a parameter whose value is usually between 0.01 and 0.5. A Huber loss function with $h = 0.5$ is plotted in Fig. 1. There are three different parts in the Huber loss function, and they together form a continuous and differentiable function. This loss function approaches the hinge loss curve when $h \to 0$. Although the Huber loss is more complicated than the squared hinge loss, its function value increases linearly with the value of $\left\langle \mathbf{A}^r, \mathbf{X} \right\rangle - \rho$. Hence, when a training set containing outliers or samples heavily contaminated by noise, the Huber loss might give a more reasonable (milder) penalty than the squared hinge loss does. We discuss both loss functions in our experimental study. Again, we highlight that by using these two loss functions, the cost function $f(\mathbf{X}, \rho)$ that we are going to optimization becomes differentiable with respect to both $\mathbf{X}$

Algorithm 1 The proposed optimization algorithm.

Input:
The maximum number of iterations K;
A pre-set tolerance value ε (e.g., 10^{-5}).
1 **Initialize**: $\mathbf{X}_0$ such that $\mathbf{Tr}(\mathbf{X}_0) = 1, \mathbf{rank}(\mathbf{X}_0) = 1$;
2 **for** $k = 1, 2, \ldots, K$ **do**
3 ∘ Compute ρ_k by solving the subproblem $\rho_k = \arg\max_{\rho>0} f(\mathbf{X}_{k-1}, \rho)$;
4 ∘ Compute $\mathbf{X}_k$ by solving the problem $\mathbf{X}_k = \arg\max_{\mathbf{X}\succcurlyeq 0, \mathbf{Tr}(\mathbf{X})=1} f(\mathbf{X}, \rho_k)$;
5 ∘ **if** $k > 1$ and $|f(\mathbf{X}_k, \rho_k) - f(\mathbf{X}_{k-1}, \rho_k)| < \varepsilon$ and $|f(\mathbf{X}_{k-1}, \rho_k) - f(\mathbf{X}_{k-1}, \rho_{k-1})| < \varepsilon$ **then**
6 break (converged);
Output: The final p.s.d.matrix $\mathbf{X}_k$.

Algorithm 2 Compute $\mathbf{X}_k$ in the proposed algorithm.

Input:
ρ_k and $\mathbf{X}_1$ which is an initial approximation of $\mathbf{X}_k$;
The maximum number of iterations J.
1 **for** $i = 1, 2, \ldots, J$ **do**
2 ∘ Compute $\mathbf{v}_i$ that corresponds to the largest eigenvalue l_i of the matrix $\nabla f(\mathbf{X}_i, \rho_k)$;
3 ∘ **if** $l_i < \varepsilon$ **then**
4 break (converged);
5 ∘ Let the search direction be $\mathbf{p}_i = \mathbf{v}_i\mathbf{v}_i^\top - \mathbf{X}_i$;
6 ∘ Set $\mathbf{X}_{i+1} = \mathbf{X}_i + \alpha\mathbf{p}_i$. Here, α is found by line search;
Output: Set $\mathbf{X}_k = \mathbf{X}_i$.

and ρ.

III. Scalable and Fast Optimization Algorithm

The proposed algorithm maximizes the objective function iteratively, and in each iteration the two variables $\mathbf{X}$ and ρ are optimized alternatively. Note that the optimization in this alternative strategy retains the global optimum because $f(\mathbf{X}, \rho)$ is a convex function in both variables $(\mathbf{X}, \rho)$ and $(\mathbf{X}, \rho)$ are not coupled together. We summarize the proposed algorithm in Algorithm 1. Note that ρ_k is a scalar and line 3 in Algorithm 1 can be solved directly by a simple 1-D maximization process. However, $\mathbf{X}$ is a p.s.d.matrix with size of $D \times D$. Recall that D is the dimensionality of feature vectors. The following section presents how $\mathbf{X}$ is efficiently optimized in our algorithm.

A. Optimizing for the Mahalanobis Matrix $\mathbf{X}_k$

Let $\mathcal{P} = \{\mathbf{X} \in \mathbb{R}^{D\times D} : \mathbf{X} \succcurlyeq 0, \mathbf{Tr}(\mathbf{X}) = 1\}$ be the domain in which a feasible $\mathbf{X}$ lies. Note that $\mathcal{P}$ is a convex set of $\mathbf{X}$. As shown in line 4 in Algorithm 1, we need to solve the following maximization problem:

$$\max_{\mathbf{X}\in\mathcal{P}} \; f(\mathbf{X}, \rho_k) \tag{11}$$

where ρ_k is the output of line 3 in Algorithm 1. Our algorithm offers a simple and efficient way for solving this problem by explicitly maintaining the positive semidefiniteness property of the matrix $\mathbf{X}$. It needs only compute the largest eigenvalue and the corresponding eigenvector whereas most previous approaches such as the method of [24] require a full eigen-decomposition of $\mathbf{X}$. Their computational complexities are $O(D^2)$ and $O(D^3)$, respectively. When D is large, this computational complexity difference could be significant.

Let $\nabla f(\mathbf{X}, \rho_k)$ be the gradient matrix of $f(\cdot)$ with respect to $\mathbf{X}$ and α be the step size for updating $\mathbf{X}$. Recall that we update $\mathbf{X}$ in such a way that $\mathbf{X}_{i+1} = (1-\alpha)\mathbf{X}_{i+1} + \alpha\delta\mathbf{X}$, where $\mathbf{rank}(\delta\mathbf{X}) = 1$ and $\mathbf{Tr}(\delta\mathbf{X}) = 1$. To find the $\delta\mathbf{X}$ that satisfies these constraints and in the meantime can best approximate the gradient matrix $\nabla f(\mathbf{X}, \rho_k)$, we need to solve the following optimization problem:

$$\begin{aligned} \max_{\delta\mathbf{X}} \quad & \left\langle \nabla f(\mathbf{X}, \rho_k), \delta\mathbf{X} \right\rangle \\ \text{subject to} \quad & \mathbf{rank}(\delta\mathbf{X}) = 1 \;\; \mathbf{Tr}(\delta\mathbf{X}) = 1. \end{aligned} \tag{12}$$

The optimal $\delta\mathbf{X}^\star$ is exactly $\mathbf{v}\mathbf{v}^\top$ where $\mathbf{v}$ is the eigenvector of $\nabla f(\mathbf{X}, \rho_k)$ that corresponds to the largest eigenvalue. The constraints says that $\delta\mathbf{X}$ is an outer product of a unit vector: $\delta\mathbf{X} = \mathbf{v}\mathbf{v}^\top$ with $||\mathbf{v}||_2 = 1$. Here, $||\cdot||_2$ is the Euclidean norm. Problem (12) can then be written as: $\max_{\mathbf{v}} \mathbf{v}^\top[\nabla f(\mathbf{X}, \rho_k)]\mathbf{v}$, subject to $||\mathbf{v}||_2 = 1$. It is clear now that an eigen-decomposition gives the solution to the above problem.

Hence, to solve the above optimization, we only need to compute the leading eigenvector of the matrix $\nabla f(\mathbf{X}, \rho_k)$. Note that $\mathbf{X}$ still retains the properties of $\mathbf{X} \succcurlyeq 0, \mathbf{Tr}(\mathbf{X}) = 1$ after applying this process.

Clearly, a key parameter of this optimization process is α which implicitly decides the total number of iterations. The computational overhead of our algorithm is proportional to the number of iterations. Hence, to achieve a fast optimization process, we need to ensure that in each iteration the α can lead to a sufficient reduction on the value of $f(\cdot)$. This is discussed in the following part.

B. Finding the Optimal Step Size α

We employ the backtracking line search algorithm in [17] to identify a suitable α. It reduces the value of α until the Wolfe conditions are satisfied. As shown in Algorithm 2, the search direction is $\mathbf{p}_i = \mathbf{v}_i\mathbf{v}_i^\top - \mathbf{X}_i$. The Wolfe conditions that we use are

$$\begin{aligned} & f(\mathbf{X}_i + \alpha\mathbf{p}_i, \rho_i) \le f(\mathbf{X}_i, \rho_i) + c_1\alpha\mathbf{p}_i^\top\nabla f(\mathbf{X}_i, \rho_i) \\ & \left|\mathbf{p}_i^\top\nabla f(\mathbf{X}_i + \alpha\mathbf{p}_i, \rho_i)\right| \le c_2\left|\mathbf{p}_i^\top\nabla f(\mathbf{X}_i, \rho_i)\right| \end{aligned} \tag{13}$$

where $0 < c_1 < c_2 < 1$. The result of backtracking line search is an acceptable α which can give rise to sufficient reduction on the function value of $f(\cdot)$. We show in the experiments that with this setting our optimization algorithm can achieve better computational efficiency than some of the existing solvers.

IV. Experiments

The goal of these experiments is to verify the efficiency of our algorithm in achieving comparable (or sometimes even better) classification performances with a reduced computational cost. We perform experiments on 10 data

sets described in Table I. For some data sets, principal component analysis (PCA) is performed to remove noises and reduce the dimensionality. The metric learning algorithms are then run on the data sets preprocessed by PCA. The Wine, Balance, Vehicle, Breast-Cancer and Diabetes data sets are obtained from University of California, Irvine, Machine Learning Repository [16]; USPS is from S. Roweis' website;[1] MNIST and Letter are from Libsvm [3]. For MNIST, we only use its test data in our experiment. The ORLface data is from AT&T research[2] and Twin-Peaks is downloaded from L. van der Maaten's website.[3] The Face and Background classes (435 and 520 images, respectively) in the image retrieval experiment are obtained from the Caltech-101 object database [6]. In order to perform statistics analysis, the ORLface, Twin-Peaks, Wine, Balance, Vehicle, Diabetes, and Face-Background data sets are randomly split as 10 pairs of train/validation/test subsets and experiments on those data sets are repeated 10 times on each split.

The k-NN classifier with the Mahalanobis distance learned by our algorithm (termed SDPMetric in short) is compared with the k-NN classifiers using a simple Euclidean distance ("Euclidean" in short) and that learned by LMNN[4] in [24]. Since, Weinberger *et al.* [24] have shown that LMNN obtains the classification performance comparable to support vector machines on some data sets, we focus on the comparison between our algorithm and LMNN, which is considered as the state-of-the-art. To prepare the training triplet set $\mathcal{S}$, we apply the 3-NN method to these data sets and generate the training triplets for our algorithm. The training data sets for LMNN are also generated using 3-NN, except that the Twin-peaks and ORLface are applied with the 1-NN method. Also, the experiment compares the two variants of our proposed SDPMetric, which use the squared hinge loss (denoted as SDPMetric-S) and the Huber loss(SDPMetric-H), respectively. We split each data set into 70/15/15% randomly and refer to those split sets as training, cross validation and test sets except preseparated data sets (Letter) and Face-Background which was made for image retrieval. Following [24], LMNN uses 85/15% data for training and testing. The training data are also split into 70/15% in LMNN for cross validation to be consistent with our SDPMetric. The Letter data set is separated according to Hsu and Lin [9]. Same as in [24], PCA is applied to USPS, MNIST, and ORLface to reduce the dimensionality of feature vectors.

The following experimental study demonstrates that our algorithm achieves slightly better classification accuracy rates with a much less computational cost than LMNN on most of the tested data sets. The detailed test error rates and timing results are reported in Tables II and III. As we can see, the test error rates of SDPMetric-S are comparable to those of LMNN. SDPMetric-H achieves lower misclassification error rates than LMNN and the Euclidean distance on most of data sets except Face-Background data (which is treated as an image retrieval problem) and MNIST, on which SDPMetric-S achieves a lower error rate. Overall, we can conclude that the proposed SDPMetric either with squared hinge loss or Huber loss is at least comparable to (or sometimes slightly better than) the state-of-the-art LMNN method in terms of classification performance.

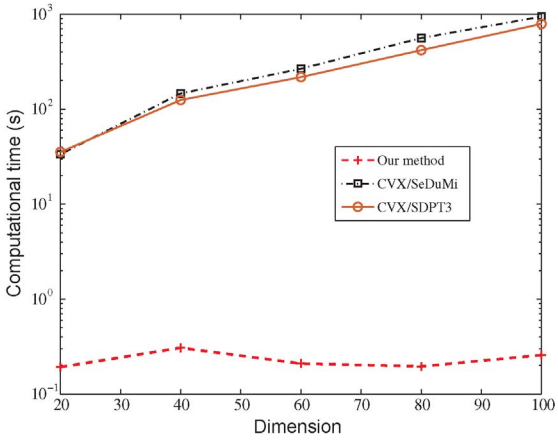

Fig. 2. Computational time vs. the dimensionality of feature vectors.

Before reporting the timing result on these benchmark data sets, we compared our algorithm (SDPMetric-H) with two convex optimization solvers, namely, SeDuMi [21] and SDPT3 [22] which are used as internal solvers in the disciplined convex programming software CVX [8]. Both SeDuMi and SDPT3 use interior-point based methods. To perform eigen-decomposition, our SDPMetric uses ARPACK [13], which is designed to solve large scale eigenvalue problems. Our SDPMetric is implemented in standard C/C++. Experiments have been conducted on a standard desktop. We randomly generated 1000 training triplets and gradually increase the dimensionality of feature vectors from 20 to 100. Fig. 2 illustrates computational time of ours, CVX/SeDuMi and CVX/SDPT3. As shown, the computational load of our algorithm almost keeps constant as the dimensionality increases. This might be because the proportion of eigen-decomposition's CPU time does not dominate with dimensions varying from 20 to 100 in SDPMetric on this data set. In contrast, the computational loads of CVX/SeDuMi and CVX/SDPT3 increase quickly in this course. In the case of the dimension of 100, the difference on CPU time can be as large as $800 \sim 1000$ s. This shows the inefficiency and poor scalability of standard interior-point methods. Secondly, the computational time of LMNN, SDPMetric-S, and SDPMetric-H on these benchmark data sets are compared in Table III. As shown, LMNN is always slower than the proposed SDPMetric which converges very fast on these data sets. Especially, on the Letter and Twin-Peaks data sets, SDPMetric shows significantly improved computational efficiency. Note that for a fair comparison, same k-NN strategy is used to generate triplets for SDPMetric and pairwise constraints for LMNN in this experiments.

Face-Background data set consists of the two object classes, Face-easy and Background-Google in [6], as a retrieval prob-

[1] http://www.cs.nyu.edu/~roweis/data.html

[2] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[3] http://ticc.uvt.nl/lvdrmaaten/

[4] In our experiment, we have used the implementation of LMNN's authors. Note that to be consistent with the setting in [24], LMNN here also uses the "obj = 1" option and updates the projection matrix to speed up its computation. If we update the distance matrix directly to get global optimum, LMNN would be much slower due to full eigen-decomposition at each iteration.

TABLE I

TEN BENCHMARK DATA SETS USED IN THE EXPERIMENT. MISSING ENTRIES IN ''DIMENSION AFTER PCA'' INDICATE NO PCA PROCESSING

	# Training	# Validation	# Test	Dimension	Dimension After PCA	# Classes	# Runs	# Triplets for SDPMetric
$USPS_{PCA}$	7700	1650	1650	256	60	10	1	69 300
USPS	7700	1650	1650	256		10	1	7700
$MNIST_{PCA}$	7000	1500	1500	784	60	10	1	63 000
MNIST	7000	1500	1500	784		10	1	7000
Letter	10 500	4500	5000	16		26	1	94 500
ORLface	280	60	60	2576	42	40	10	280
Twin-Peaks	14 000	3000	3000	3		11	10	14 000
Wine	126	26	26	13		3	10	1134
Balance	439	93	93	4		3	10	3951
Vehicle	593	127	126	18		4	10	5337
Breast-Cancer	479	102	102	10		2	10	4311
Diabetes	538	115	115	8		2	10	4842
Face-Background	472	101	382	100		2	10	4428

TABLE II

3-NEAREST NEIGHBOR MISCLASSIFICATION ERROR RATES (%). THE STANDARD DEVIATION VALUES ARE IN BRACKETS. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

	Euclidean	LMNN	SDPMetric-S	SDPMetric-H
$USPS_{PCA}$	2.85	2.00	2.30	**1.70**
USPS	4.42	3.58	4.36	**3.21**
$MNIST_{PCA}$	3.15	3.15	**3.00**	3.35
MNIST	4.80	4.60	**4.13**	4.53
Letter	5.38	4.04	3.60	**3.46**
ORLface	6.00 (3.46)	5.00 (2.36)	4.75 (2.36)	**4.25 (2.97)**
Twin-Peaks	1.03 (0.21)	0.90 (0.19)	1.17 (0.20)	**0.79 (0.19)**
Wine	24.62 (5.83)	3.85 (2.72)	3.46 (2.69)	**3.08 (2.31)**
Bal	19.14 (1.59)	14.19 (4.12)	**9.78 (3.17)**	10.32 (3.44)
Vehicle	28.41 (2.41)	21.59 (2.71)	21.67 (4.00)	**20.87 (2.97)**
Breast-Cancer	4.51 (1.49)	4.71 (1.61)	3.33 (1.40)	**2.94 (0.88)**
Diabetes	28.00 (2.84)	27.65 (3.45)	28.70 (3.67)	**27.64 (3.71)**
Face-Background	26.41 (2.72)	**14.71 (1.33)**	16.75 (1.72)	15.86 (1.37)

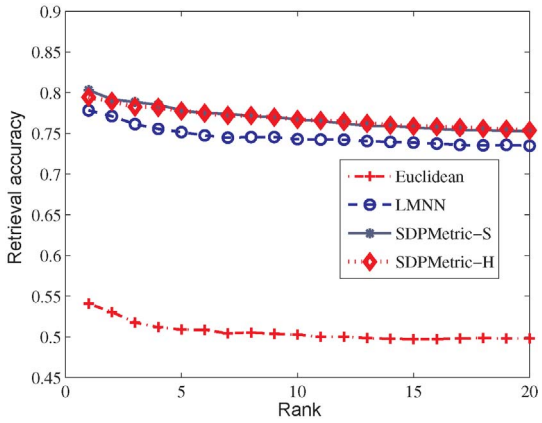

Fig. 3. Retrieval performances of SDPMetric-S, SDPMetric-H, LMNN, and the Euclidean distance. The curves of SDPMetric-S and SDPMetric-H are very close.

TABLE III

COMPUTATIONAL TIME FOR EACH RUN

	LMNN	SDPMetric-S	SDPMetric-H
$USPS_{PCA}$	256s	111s	258s
USPS	1.6h	16m	20m
$MNIST_{PCA}$	219s	111s	99s
MNIST	9.7h	1.4h	37m
Letter	1036s	6s	136s
ORLface	13s	4s	3s
Twin-Peakes	595s	less than 1s	less than 1s
Wine	9s	2s	2s
Balance	7s	less than 1s	2s
Vehicle	19s	2s	7s
Breast-Cancer	4s	2s	3s
Diabetes	10s	less than 1s	2s
Face-Background	92s	5s	5s

lem. The images in the class of Background-Google are randomly collected from the Internet and they are used to represent the nontarget class. For each image, a number of interest regions are identified by the Harris-Affine detector [15] and the visual content in each region is characterized by the SIFT descriptor [14]. A codebook of size 100 is created by using k-means clustering. Each image is then represented by a 100-dimensional histogram vector containing the number of occurrences of each visual word. We evaluate retrieval accuracy using each facial image in a test subset as a query. For each compared metric, the **accuracy** of the retrieved top 1 to 20 images are computed, which is defined as the ratio of the number of facial images to the total number of retrieved images. We calculate the average accuracy of each test subset and then average over the whole 10 test subsets. Fig. 3 shows the retrieval accuracies of the Mahalanobis distances learned by Euclidean, LMNN, and SDPMetric. Clearly we can observe that SDPMetric-H and SDPMetric-S consistently present higher retrieval accuracy values, which again verifies their advantages over the LMNN method and Euclidean distance.

We have also evaluated SDPMetric-H on USPS with different settings. First, we experiment with varying number of triplets on $USPS_{PCA}$ (the feature dimension is 60 after PCA).

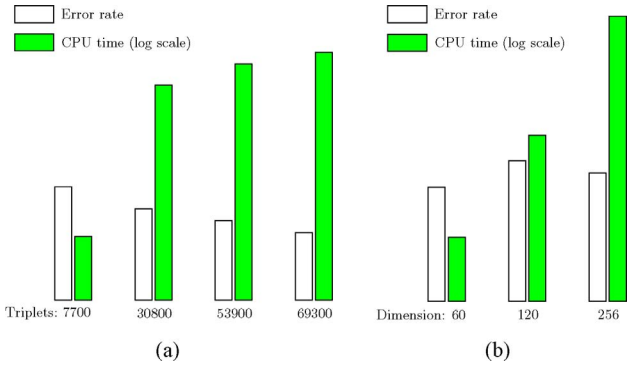

Fig. 4. Classification error rates of 3-nearest neighbor and running time of SDPMetric-H on the USPS data set. (a) Varying the number of training triplets. Error rates are 2.85%, 2.30%, 2.00%, and 1.70%; computational time for each run is 5, 217, 368, and 491 s. (b) Varying the PCA dimension. Error rates are 2.85%, 3.51%, and 3.21%; computational time for each run is 5, 63, and 1205 s, respectively.

The results are reported in Fig. 4. Second, we keep the number of triplets fixed to 7700 and vary the feature dimension after PCA. For the first setting, as expected, when more triplets are used, the test error decreases from 2.85% to 1.70%. The computational time increases with more triplets, from 5s to 491s. For the second setting, we see that with the dimension being 60, 120, and 256, the computational time for each case is 5, 53, and 1205 s, respectively. The error rates in this setting show that PCA does help improve the classification performance on USPS, if the dimension is properly chosen, probably due to noise reduction. The PCA dimension might need to be tuned by cross validation.

V. Conclusion

We have proposed a new algorithm to demonstrate how to efficiently learn a Mahalanobis distance metric with the principle of margin maximization. Enlightened by the important theorem on p.s.d.matrix decomposition in [19], we have designed a gradient descent method to update the Mahalanobis matrix with cheap computational loads and at the same time, the p.s.d.property of the learned matrix is maintained during the whole optimization process. Experiments on benchmark data sets and the retrieval problem verify the superior classification performance and computational efficiency of the proposed distance metric learning algorithm.

The proposed algorithm may be used to solve more general SDP problems in machine learning. To look for other applications is one of the future research directions.

References

[1] B. Borchers, "CSDP, a C library for semidefinite programming," *Optim. Methods Softw.*, vol. 11, no. 1, pp. 613–623, 1999.

[2] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[3] C.-C. Chang and C.-J. Lin. (2001). *LIBSVM: A Library for Support Vector Machines* [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

[4] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2007, pp. 209–216.

[6] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," in *Proc. Workshop Generative-Model Based Vision, Conjunction IEEE Conf. Comp. Vis. Patt. Recogn.*, Jul. 2004, pp. 178–186.

[7] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 17. Dec. 2005, pp. 513–520.

[8] M. Grant, S. Boyd, and Y. Ye, "CVX user's guide: For CVX version 1.1," Stanford University, Stanford, CA, Tech. Rep., 2007 [Online]. Available: http://www.stanford.edu/∼boyd/cvx/

[9] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Feb. 2002.

[10] J. Kim, C. Shen, and L. Wang, "A scalable algorithm for learning a Mahalanobis distance metric," in *Proc. Asian Conf. Comput. Vis.*, LNCS. 5996/2010. Sep. 2009, pp. 299–310.

[11] B. Kulis, M. A. Sustik, and I. S. Dhillon, "Low-rank kernel learning with Bregman matrix divergences," *J. Mach. Learn. Res.*, vol. 10, pp. 341–376, Dec. 2009.

[12] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, no. 1, pp. 27–72, Dec. 2004.

[13] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods*. Phildelphia, PA: SIAM, 1998.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.

[15] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int. J. Comp. Vis.*, vol. 60, no. 1, pp. 63–86, 2004.

[16] D. Newman, S. Hettich, C. Blake, and C. Merz. (1998). *UCI Repository of Machine Learning Databases* [Online]. Available: http://archive.ics.uci.edu/ml/

[17] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer Verlag, 1999.

[18] R. Rosales and G. Fung, "Learning sparse metrics via linear programming," in *Proc. ACM Int. Conf. Knowledge Discovery Data Mining*, Aug. 2006, pp. 367–373.

[19] C. Shen, A. Welsh, and L. Wang, "PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning," in *Proc. Adv. Neural Inf. Process. Syst.* Vancouver, Canada: MIT Press, Dec. 2008, pp. 1473–1480.

[20] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.

[21] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones (updated for version 1.05)," *Optim. Methods Softw.*, vols. 11–12, no. 1, pp. 625–653, 1999.

[22] R. H. Tütüncü, K. C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," *Math. Program.*, vol. 95, no. 2, pp. 189–217, 2003.

[23] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[24] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2006, pp. 1475–1482.

[25] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2. Oct. 2005, pp. 1800–1807.

[26] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2003, pp. 505–512.

[27] L. Yang, R. Sukthankar, and S. C. H. Hoi, "A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 30–44, Jan. 2010.