1-1-2010

A scalable algorithm for learning a Mahalanobis distance metric

junae kim
National ICT Australia

Chunhua Shen
University of Adelaide

Lei Wang
Australian National University, leiw@uow.edu.au

A scalable algorithm for learning a Mahalanobis distance metric

Abstract

A distance metric that can accurately re°ect the intrinsic characteristics of data is critical for visual recognition tasks. An e®ective solution to de¯ning such a metric is to learn it from a set of training samples. In this work, we propose a fast and scalable algorithm to learn a Ma- halanobis distance. By employing the principle of margin maximization to secure better generalization performances, this algorithm formulates the metric learning as a convex optimization problem with a positive semide¯nite (psd) matrix variable. Based on an important theorem that a psd matrix with trace of one can always be represented as a convex combination of multiple rank-one matrices, our algorithm employs a differentiable loss function and solves the above convex optimization with gradient descent methods. This algorithm not only naturally maintains the psd requirement of the matrix variable that is essential for metric learning, but also signi¯cantly cuts down computational overhead, making it much more e±cient with the increasing dimensions of fea- ture vectors. Experimental study on benchmark data sets indicates that, compared with the existing metric learning algorithms, our algorithm can achieve higher classi¯cation accuracy with much less computational load.

Keywords

scalable, metric, algorithm, learning, distance, mahalanobis

Disciplines

Engineering | Science and Technology Studies

Publication Details

kim, j., Shen, C. & Wang, L. (2010). A scalable algorithm for learning a Mahalanobis distance metric. Computer Science, Springer, the 9th Asian Conference on Computer Vision (ACCV) (pp. 1-12). Springer.

A Scalable Algorithm for Learning a Mahalanobis Distance Metric

Junae Kim[1,2], Chunhua Shen[2,1], and Lei Wang[1]

[1] The Australian National University, Canberra, ACT, Australia
{junae.kim, lei.wang}@anu.edu.au
[2] NICTA, Canberra Research Laboratory, Canberra, ACT, Australia**
chunhua.shen@nicta.com.au

Abstract. A distance metric that can accurately reflect the intrinsic characteristics of data is critical for visual recognition tasks. An effective solution to defining such a metric is to learn it from a set of training samples. In this work, we propose a fast and scalable algorithm to learn a Mahalanobis distance. By employing the principle of margin maximization to secure better generalization performances, this algorithm formulates the metric learning as a convex optimization problem with a positive semidefinite (psd) matrix variable. Based on an important theorem that a psd matrix with trace of one can always be represented as a convex combination of multiple rank-one matrices, our algorithm employs a differentiable loss function and solves the above convex optimization with gradient descent methods. This algorithm not only naturally maintains the psd requirement of the matrix variable that is essential for metric learning, but also significantly cuts down computational overhead, making it much more efficient with the increasing dimensions of feature vectors. Experimental study on benchmark data sets indicates that, compared with the existing metric learning algorithms, our algorithm can achieve higher classification accuracy with much less computational load.

1 Introduction

Many visual recognition tasks can be regarded as inferring a distance metric that is able to measure the similarity of visual data in a way consistent with human perception. Typical examples include visual object categorization [1] and content-based image retrieval [2], in which a similarity metric is needed to discriminate different object classes or the relevant and irrelevant images for a given query. Classifiers, from the simple k-Nearest Neighbor (k-NN) [3] to the advanced Support Vector Machines (SVMs) [4], can be explicitly or implicitly related to a distance metric. As one of the representative classifiers, k-NN has been

** NICTA is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through *Backing Australia's Ability initiative* and the ICT Research Center of Excellence programs.

applied to a wide range of visual recognition tasks and it is the classifier that directly depends on a predefined distance metric. To make this classifier work well, an appropriate distance metric has to be applied. Previous work (*e.g.*, [5, 6]) has shown that compared to using the standard Euclidean distance, employing an well-designed distance metric to measure the (dis)similarity of data can significantly boost the classification accuracy of k-NN.

In this work, we propose a scalable and fast algorithm to learn a Mahalanobis distance metric. The key issue in this task is to learn an optimal Mahalanobis matrix in this distance metric. It has been shown in the statistical learning theory [7] that increasing the margin between different classes helps to reduce the generalization error. Hence, our algorithm formulates the Mahalanobis matrix as a variable of the margin and optimizes it via margin maximization. By doing so, the learned Mahalanobis distance metric can achieve sufficient separation at the boundaries between different classes. More importantly, we address the scalability problem of learning a Mahalanobis distance in the presence of high-dimensional feature vectors, which is a critical issue of distance metric learning. As indicated in a theorem in [8], a positive semidefinite matrix (psd) with trace of one can always be represented as a convex combination of a set of rank-one matrices with trace being one. This 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 largest principal eigenvector of a gradient matrix and to update the current Mahalanobis matrix. This process incurs much less computational overhead than the metric learning algorithms in the literature [9]. Moreover, thanks to the above theorem, this process automatically preserves the psd property of the Mahalanobis matrix. To verify its efficiency, the proposed algorithm is tested on a set of 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 higher classification accuracy. Meanwhile, in terms of the optimization time, our algorithm is much less affected by the increased dimensionality of feature vectors.

2 Related work

For a given classification task, 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. There has been some work on distance metric learning in the literature. Xing et al. [5] propose an approach to learn a Mahalanobis distance for supervised clustering. It minimizes the sum of the distances 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 improves clustering performance significantly. However, to maintain the psd property, they use 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 handling high-dimensional data. Goldberger et al. [10]

develop the algorithm called Neighborhood Component Analysis, which learns a Mahalanobis distance by minimizing the leave-one-out cross-validation error of the k-NN classifier on a training set. However, their algorithm leads to a non-convex optimization problem and expensive computational load. Although the work in [10] learns a Mahalanobis distance metric as ours does, it does not study and make use of the psd property of the Mahalanobis matrix. The work closest to ours is [9] 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 created and solved by using the semidefinite programming (SDP) technique. Note that they have adopted an alternating projection algorithm for solving the resulted SDP because standard interior-point methods do not scale well. At each step of iteration, similar to [5], also a full eigen-decomposition is needed. Our approach is largely inspired by their work. However, we take a different way to achieving the margin maximization and lead to a different objective function. To develop our fast algorithm, we adopt a differentiable loss function rather than the discontinuous hinge loss function in [9]. More importantly, our algorithm has a clear advantage on computational efficiency (we only need to compute the leading eigenvector) and achieves better classification performance.

3 Formulation of our optimization problem

Let $\mathbf{a}_i \in \mathbb{R}^D (i = 1, 2, \cdots, n)$ 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 $\mathcal{S}$ which 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 a different class. A Mahalanobis distance can be defined as follows. Let $\mathbf{P} \in \mathbb{R}^{D \times D}$ denote a linear transformation and $\mathbf{dist}$ be the squared Euclidean distance in the transformed space. The squared distance between the projections of $\mathbf{a}_i$ and $\mathbf{a}_j$ is

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

According to the class membership of $\mathbf{a}_i$, $\mathbf{a}_j$ and $\mathbf{a}_k$, we require that $\mathbf{dist}_{ik} \geq \mathbf{dist}_{ij}$ and it can be obtained that

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

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 $\mathbf{X} = \mathbf{P}\mathbf{P}^T$ is introduced and used through out the whole learning process. Learning a Mahalanobis distance is essentially to learn the Mahalanobis matrix $\mathbf{X}$. (2) becomes linear in $\mathbf{X}$.

3.1 Maximization of a soft margin

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

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

It is maximized to identify the optimal Mahalanobis matrix $\mathbf{X}$. In the meantime, to deal with non-separable data sets and avoid over-fitting training samples, we must allow some training errors while maximizing the margin. Considering these factors, we define the objective function for learning $\mathbf{X}$ as

$$\begin{aligned} &\max_{\rho, \mathbf{X}, \boldsymbol{\xi}} \ \rho - C \textstyle\sum_{r=1}^{|\mathcal{S}|} \xi_r \\ &\text{s.t.} \ \ \mathbf{X} \succcurlyeq 0, \mathbf{Tr}(\mathbf{X}) = 1, \quad \xi_r \geq 0, \quad r = 1, 2, \cdots, |\mathcal{S}|, \\ &\quad (\mathbf{a}_i - \mathbf{a}_k)^T \mathbf{X} (\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{X} (\mathbf{a}_i - \mathbf{a}_j) \geq \rho - \xi_r, \ \forall (\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{S}, \end{aligned} \tag{4}$$

where $\mathbf{X} \succcurlyeq 0$ constrains $\mathbf{X}$ to be a psd matrix and $\mathbf{Tr}(\mathbf{X})$ denotes of 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 training error and the margin. $\xi \geq 0$ is the slack variable similar to that used in the SVMs and it corresponds to the soft-margin hinge loss. Imposing $\mathbf{Tr}(\mathbf{X}) = 1$ removes the scale ambiguity because the inequality constrains are scale invariant. To simplify exposition, we define

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

By doing so, the last constraint in (4) can be written as

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

Note that this is a linear constrain on $\mathbf{X}$. Problem (4) is thus a typical SDP problem since it has a linear objective function and linear constraints plus a psd conic constraint. It can be solved using off-the-shelf SDP solvers. However, directly solving problem (4) using standard interior-point SDP solvers would quickly become computationally intractable with the increasing dimensionality of feature vectors. The following shows our way of developing a fast algorithm for (4).

3.2 Employing a differentiable loss function

It is proven in [8] that *a psd matrix can always be decomposed as a linear convex combination of a set of rank-one matrices.* In the context of our problem, this means that $\mathbf{X} = \sum \theta_i \mathbf{Z}_i$, where $\mathbf{Z}_i$ is a rank-one matrix and $\mathbf{Tr}(\mathbf{Z}_i) = 1$. This important result inspires us to develop a gradient descent based optimization algorithm. In each iteration $\mathbf{X}$ is updated as

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

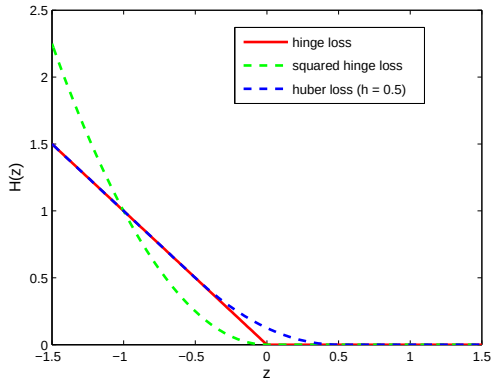

Fig. 1. The hinge loss, squared hinge loss and huber loss used in our work

where $\triangle\mathbf{X}$ is a rank-one and trace-one matrix. $\mathbf{p}_i$ is the search direction and will be introduced in Table 2.

To make a gradient descent method applicable, we need to ensure that the object function to be differentiable with respect to the variables ρ and $\mathbf{X}$.

Let f denote the objective function and ℓ be a loss function. Our optimization problem can be written as

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

The above problem (4) adopts the hinge loss function which is defined as $\ell(z) = \max(0, -z)$. However, the hinge loss is not differentiable at the point of $z = 0$, and gradient-based optimization will encounter problems. In order to remove this problem, we propose to use differentiable loss functions, for example, the squared hinge loss or huber loss functions discussed below.

The squared hinge loss function can be represented as

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

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:

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

Table 1. The proposed optimization algorithm.

1. Randomly initialize $\mathbf{X}_0$ such that $\mathbf{Tr}(\mathbf{X}_0) = 1, \mathbf{rank}(\mathbf{X}_0) = 1$;
ε is a pre-set small value.
2. For $k = 1, 2, \ldots$
    2.1 Compute ρ_k by solving the subproblem
$$\rho_k = \arg\max_{\rho>0} f(\mathbf{X}_{k-1}, \rho).$$
    2.2 Compute $\mathbf{X}_k$ by solving the problem
$$\mathbf{X}_k = \arg\max_{\mathbf{X}\succcurlyeq 0, \mathbf{Tr}(\mathbf{X})=1} f(\mathbf{X}, \rho_k).$$
    2.3 If $|f(\mathbf{X}_k, \rho_k) - f(\mathbf{X}_{k-1}, \rho_k)| < \varepsilon$ and $|f(\mathbf{X}_{k-1}, \rho_k) - f(\mathbf{X}_{k-1}, \rho_{k-1})| < \varepsilon$ $(k > 1)$,
     break.
3. End for.

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 of z. This loss function approaches the hinge loss curve when $h \to 0$. Although the huber loss is a bit 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 contains outliers or sample heavily contaminated by noise, the huber loss can often give a more reasonable (milder) penalty than the squared hinge loss does. We discuss both loss functions in our experimental study. Note 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 $\mathbf{X}$ and ρ.

4 A 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 optimizing in this alternative way will not prevent the global optimum from being obtained because $f(\mathbf{X}, \rho)$ is a convex function in both variables $(\mathbf{X}, \rho)$ and $(\mathbf{X}, \rho)$ are not coupled together. The pseudo-code of the proposed algorithm is given in Table 1. Note that ρ_k is a scalar and the step 2.1 in Table 1 can be solved directly by a simple one-dimensional maximization process. However, $\mathbf{X}$ is a psd 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.

4.1 Compute 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 Step 2.2 in Table 1, we need

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

1. Given ρ_k and initial approximation $\mathbf{X}_k$, calculate $\nabla f(\mathbf{X}_k, \rho_k)$.
2. For $i = 1, 2, \ldots$
2.1 Compute v_i corresponds to the largest eigen value of $\nabla f(\mathbf{X}_i, \rho_k)$.
2.2 If the largest eigen value is less than ε, break.
2.3 Let the search direction $\mathbf{p}_i = \mathbf{v}_i \mathbf{v}_i^T - \mathbf{X}_i$.
2.4 Set $\mathbf{X}_{i+1} = \mathbf{X}_i + \alpha \mathbf{p}_i$. α is found by line search.
3. End for.
4. Set $\mathbf{X}_{k+1} = \mathbf{X}_i$.

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 Step 2.1. Our algorithm offers a simple and efficient way for solving this problem by automatically maintaining the positive semidefinite property of the matrix $\mathbf{X}$. It needs only compute the principal eigenvalue computation whereas the previous approaches such as the method of [9] require to carry out a full eigen-decomposition of $\mathbf{X}$.

Let $\nabla f(\mathbf{X}, \rho_k)$ be the gradient matrix of f with respect to $\mathbf{X}$ and α be a step size for updating $\mathbf{X}$. Recall that we update $\mathbf{X}$ in such a way that $\mathbf{X}_{i+1} = (1-\alpha)\mathbf{X}_{i+1} + \alpha \triangle \mathbf{X}$, where $\mathbf{rank}(\triangle \mathbf{X}) = 1$ and $\mathbf{Tr}(\triangle \mathbf{X}) = 1$. To find the $\triangle \mathbf{X}$ 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:

$$\begin{aligned} \max \quad & \left\langle \nabla f(\mathbf{X}, \rho_k), \triangle \mathbf{X} \right\rangle \\ \text{s.t. } & \mathbf{rank}(\triangle \mathbf{X}) = 1, \quad \mathbf{Tr}(\triangle \mathbf{X}) = 1. \end{aligned} \tag{12}$$

Clearly *the optimal* $\triangle \mathbf{X}^\star$ *is exactly* $\mathbf{v}_i \mathbf{v}_i^T$ *where* $\mathbf{v}_i$ *is the eigenvector of* $\nabla f(\mathbf{X}, \rho_k)$ *that corresponds to the largest eigenvalue*. Hence, to solve the above optimization, we only need to compute the principal eigenvector of the matrix $\nabla f(\mathbf{X}, \rho_k)$. The step 2.2 is elaborated in Table 2. Note that $\mathbf{X}$ still retains the properties of $\mathbf{X} \succcurlyeq 0, \mathbf{Tr}(\mathbf{X}) = 1$ after 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. This is discussed in the following part.

4.2 Compute the step size α

We employ the backtracking line search algorithm in [11] to identify a suitable α. It reduces the value of α until the Wolfe conditions are satisfied. As shown in

Table 2, the search direction is $\mathbf{p}_i = \mathbf{v}_i\mathbf{v}_i^T - \mathbf{X}_i$. The Wolfe conditions that we use are

$$f(\mathbf{X}_i + \alpha\mathbf{p}_i, \rho_i) \leq f(\mathbf{X}_i, \rho_i) + c_1\alpha\mathbf{p}_i^T\nabla f(\mathbf{X}_i, \rho_i),$$
$$\left|\mathbf{p}_i^T\nabla f(\mathbf{X}_i + \alpha\mathbf{p}_i, \rho_i)\right| \leq c_2\left|\mathbf{p}_i^T\nabla f(\mathbf{X}_i, \rho_i)\right|. \quad (13)$$

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. It will shown in the experimental study that with this setting our optimization algorithm can achieve higher computational efficiency that the existing solvers.

5 Experimental result

The goal of this experiment is to verify the efficiency of our algorithm in achieving better classification performances with less computational cost. We perform experiments on 10 data sets described in Table 3. Here, N_{train}, N_{vali}, and N_{test} denote the sizes of the training sets, validation sets, and the test sets, respectively. N_{class} is the number of class and N_{fea} shows the dimensionality of the feature vectors and "N_{fea} after PCA" is the number of features that are preserved after the Principal Component Analysis. The Wine, Balance, Vehicle, Breast-Cancer and Diabetes sets are obtained from UCI Machine Learning Repository[3], and USPS, MNIST and Letter are from libSVM[4]. For MNIST, we only use its test data in our experiment. The ORLface data is from ATT research[5] and Twin-Peaks is downloaded from Laurens van der Maaten's website[6]. The Face and Background classes (435 and 520 images respectively) in the image retrieval experiment are obtained from the Caltech-101 object database [12]. To accumulate statistics, 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 set are repeated 10 times with each pairs.

The k-NN classifier with the Mahalanobis distance learned by our algorithm (called SDPMetric in short) is compared with the k-NN classifiers using a simple Euclidean distance ("Euclidean" in short) and that learned by the large margin nearest neighbor in [9] (LMNN[7] in short). Since Weinberger et al. [9] has shown that LMNN obtains the classification performance comparable to SVMs, we focus on the comparison between our algorithm and LMNN. To prepare the

[3] Asuncion, A. Newman, D.J. (2007) UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[4] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001. The software is freely available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] Available at http://www.uk.research.att.com/facedatabase.html

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

[7] Note that to be consistent with the setting in [9], 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 more slower due to full eigen-decomposition at each iterations.

Table 3. The ten benchmark data sets used in our experiment

	N_{trn}	N_{val}	N_{tst}	N_{fea}	N_{fea} after PCA	N_{class}	N_{runs}
USPS	5,833	1,458	2,007	256	60	10	1
MNIST	7,000	1,500	1,500	784	60	10	1
Letter	10,500	4,500	5,000	16	16	26	1
ORLface	280	60	60	2,576	42	40	10
Twin-Peaks	14,000	3,000	3,000	3	3	11	10
Wine	126	26	26	13	13	3	10
Balance	439	93	93	4	4	3	10
Vehicle	593	127	126	18	18	4	10
Breast-Cancer	479	102	102	10	10	2	10
Diabetes	538	115	115	8	8	2	10
Face-Background	472	101	382	100	100	2	10

training set $\mathcal{S}$, we apply the 3-Nearest Neighbor method to these data sets to generate the training triplets for our algorithm and LMNN, 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 (SDPMetric-S in short) and the huber loss(SDPMetric-H in short), respectively. We split each data set into 70/15/15% randomly and refer to those split sets as training, validating and testing sets except pre-separated data sets(Letter and USPS) and Face-Background which was made for image retrieval. Following [9], LMNN uses 85/15% data for training and testing. The training data is also split into 70/15% inside LMNN to be consistent with our SDPMetric. Since USPS data set has been split into training/test already, only the training data are divided into 70/15% as training and validation sets. The Letter data set is separated according to Hsu et al. [13]. As in [9], the Principal Component Analysis (PCA) is applied to USPS, MNIST and ORLface to reduce the dimensions of feature vectors.

The following experimental study demonstrates that our algorithm achieves higher classification accuracy with much less computational cost than LMNN on most of the tested data sets. The detailed test error rates and timing results are reported in Table 4 and 5. As shown, the test error rates of SDPMetric-S are comparable to those of LMNN and SDPMetric-H achieves lower misclassification error rates than LMNN and the Euclidean distance on most of data sets except Face-Background data which made as a image retrieval problem and MNIST on which SDPMetric-S achieves low error rate.

Before reporting the timing result on these benchmark data sets, we compare our algorithm with SeDuMi[8] and SDPT3[9] which are used as solvers in CVX[10]. We randomly generate 1,000 training triplets and gradually increase the dimensionality of feature vectors from 20 to 100. Fig. 2 illustrates computational time

[8] A software package to solve optimization problems which is from http://sedumi.ie.lehigh.edu/.

[9] A software to solve conic optimization problems involving semidefinite, second-order and linear cone constraints. http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

[10] Matlab Software for Disciplined Convex Programming. The CVX package is available from http://www.stanford.edu/boyd/cvx/index.html

of ours, CVX/SeDuMi and CVX/SDPT3. As shown, the computational load of our algorithm almost keeps constant as the dimensionality increases. In contrast, the computational load of CVX/SeDuMi and CVX/SDPT3 rise rapidly in this course. In the case of the dimension of 100, the difference on optimization time can be as large as 800–1000 seconds. The computational time of LMNN, SDPMetric-S and SDPMetric-H are compared in Table 5. 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.

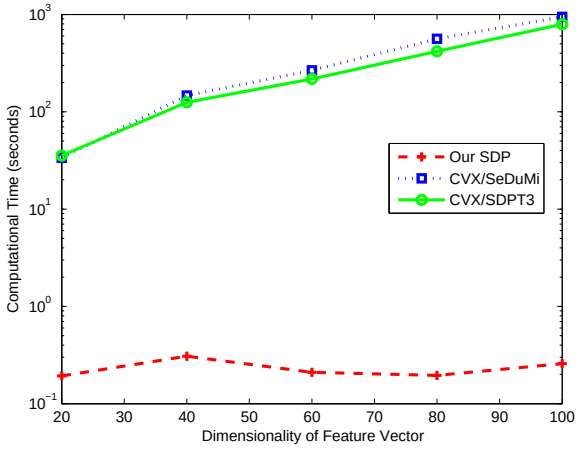

Fig. 2. Computational time vs. the dimensionality of feature vector.

Table 4. 3-Nearest Neighbor misclassification error rates. The standard deviation values are in brackets.

	Euclidean	LMNN	SDPMetric-S	SDPMetric-H
USPS	5.63	**5.18**	5.28	**5.18**
MNIST	3.15	3.15	**3.00**	3.35
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)

Face-Background data set consists of the two object classes, Face_easy and BACKGROUND_Google in [12], as a retrieval problem. The images in the class

Table 5. Computational time per each run(seconds).

	LMNN	SDPMetric-S	SDPMetric-H
USPS	256s	111s	258s
MNIST	219s	111s	99s
Letter	1036s	6s	136s
ORLface	13s	4s	3s
Twin-peakes	595s	less than 1s	less than 1s
Wine	9s	2s	2s
Bal	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

of BACKGROUND_Google are randomly collected from the Internet and they are used to represent the non-target class. For each image, a number of interest regions are identified by the Harris-Affine detector [14] and the visual content in each region is characterized by the SIFT descriptor [15]. A codebook of size 100 is created by using k-means clustering. Each image is then represented by a 100-dimensional histogram 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. we clearly observe that SDPMetric-H and SDPMetric-S consistently give the higher retrieval accuracy values, which again verifies their advantages over the LMNN and Euclidean distance.

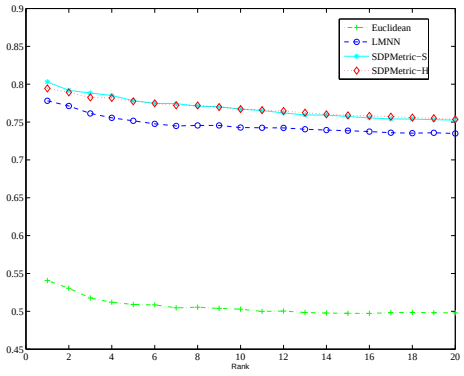

Fig. 3. Retrieval performance of SDPMetric, LMNN and the Euclidean distance

6 Conclusion

We have proposed an algorithm to show how to efficiently learn the Mahalanobis distance metric with maximal margin. Enlightened by that important theorem on psd matrix decomposition, we design a gradient descent approach to update the Mahalanobis matrix with light computational load and well maintain its psd property in the whole optimization process. Experimental study on benchmark data sets and the retrieval problem verify the superior classification performance and computational efficiency of the proposed distance metric learning algorithm.

References

1. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: Proc. IEEE International Conference on Computer Vision, Beijing, China (2005) 1800–1807
2. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(12) (December 2000) 1349–1380
3. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13** (1967) 21–27
4. Scholkpf, B., Smola, A.: Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2002)
5. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Proc. Advances in Neural Information Processing Systems, MIT Press (2003) 505–512
6. Yang, L., Sukthankar, R., Hoi, S.C.: A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence **10** (November 2008)
7. Vapnik, V.: Statistical learning theory. John Wiley and Sons Inc., New York (1998)
8. Shen, C., Welsh, A., Wang, L.: PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning. In: Proc. Advances in Neural Information Processing Systems, MIT Press (December 2008) 1473–1480
9. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Proc. Advances in Neural Information Processing Systems. (2006) 1475–1482
10. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Proc. Advances in Neural Information Processing Systems. (2005)
11. Nocedal, J., Wright, S.J.: Numerical optimization. Springer Verlag, New York (1999)
12. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In: Workshop on Generative-Model Based Vision, IEEE Conference on Computer Vision and Pattern Recognition. (2004)
13. Hsu, C.W., Lin, C.J.: A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks **13**(2) (January 2002) 415–425
14. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision **60**(1) (2004) 63–86
15. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proc. IEEE International Conference on Computer Vision. (1999) 1150–1157