Positive Semidefinite Metric Learning with Boosting

Chunhua Shen[†‡], **Junae Kim**[†‡], **Lei Wang**[‡], **Anton van den Hengel**[¶]
[†] NICTA Canberra Research Lab, Canberra, ACT 2601, Australia[*]
[‡] Australian National University, Canberra, ACT 0200, Australia
[¶] The University of Adelaide, Adelaide, SA 5005, Australia

Abstract

The learning of appropriate distance metrics is a critical problem in image classification and retrieval. In this work, we propose a boosting-based technique, termed BoostMetric, for learning a Mahalanobis distance metric. One of the primary difficulties in learning such a metric is to ensure that the Mahalanobis matrix remains positive semidefinite. Semidefinite programming is sometimes used to enforce this constraint, but does not scale well. BoostMetric is instead based on a key observation that any positive semidefinite matrix can be decomposed into a linear positive combination of trace-one rank-one matrices. BoostMetric thus uses rank-one positive semidefinite matrices as weak learners within an efficient and scalable boosting-based learning process. The resulting method is easy to implement, does not require tuning, and can accommodate various types of constraints. Experiments on various datasets show that the proposed algorithm compares favorably to those state-of-the-art methods in terms of classification accuracy and running time.

1 Introduction

It has been an extensively sought-after goal to learn an appropriate distance metric in image classification and retrieval problems using *simple and efficient* algorithms [1–5]. Such distance metrics are essential to the effectiveness of many critical algorithms such as k-nearest neighbor (kNN), k-means clustering, and kernel regression, for example. We show in this work how a Mahalanobis metric is learned from proximity comparisons among triples of training data. Mahalanobis distance, *a.k.a.* Gaussian quadratic distance, is parameterized by a positive semidefinite (p.s.d.) matrix. Therefore, typically methods for learning a Mahalanobis distance result in constrained semidefinite programs. We discuss the problem setting as well as the difficulties for learning such a p.s.d. matrix. If we let $\mathbf{a}_i, i = 1, 2 \cdots$, represent a set of points in $\mathbb{R}^D$, the training data consist of a set of constraints upon the relative distances between these points, $\mathcal{S} = \{(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) | \mathbf{dist}_{ij} < \mathbf{dist}_{ik}\}$, where $\mathbf{dist}_{ij}$ measures the distance between $\mathbf{a}_i$ and $\mathbf{a}_j$. We are interested in the case that $\mathbf{dist}$ computes the Mahalanobis distance. The Mahalanobis distance between two vectors takes the form: $\|\mathbf{a}_i - \mathbf{a}_j\|_{\mathbf{X}} = \sqrt{(\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X} (\mathbf{a}_i - \mathbf{a}_j)}$, with $\mathbf{X} \succcurlyeq 0$, a p.s.d. matrix. It is equivalent to learn a projection matrix $\mathbf{L}$ and $\mathbf{X} = \mathbf{L}\mathbf{L}^\top$. Constraints such as those above often arise when it is known that $\mathbf{a}_i$ and $\mathbf{a}_j$ belong to the same class of data points while $\mathbf{a}_i, \mathbf{a}_k$ belong to different classes. In some cases, these comparison constraints are much easier to obtain than either the class labels or distances between data elements. For example, in video content retrieval, faces extracted from successive frames at close locations can be safely assumed to belong to the same person, without requiring the individual to be identified. In web search, the results returned by a search engine are ranked according to the relevance, an ordering which allows a natural conversion into a set of constraints.

[*]NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

The requirement of $\mathbf{X}$ being p.s.d. has led to the development of a number of methods for learning a Mahalanobis distance which rely upon constrained semidefinite programing. This approach has a number of limitations, however, which we now discuss with reference to the problem of learning a p.s.d. matrix from a set of constraints upon pairwise-distance comparisons. Relevant work on this topic includes [3–8] amongst others.

Xing *et al* [4] firstly proposed to learn a Mahalanobis metric for clustering using convex optimization. The inputs are two sets: a similarity set and a dis-similarity set. The algorithm maximizes the distance between points in the dis-similarity set under the constraint that the distance between points in the similarity set is upper-bounded. Neighborhood component analysis (NCA) [6] and large margin nearest neighbor (LMNN) [7] learn a metric by maintaining consistency in data's neighborhood and keeping a large margin at the boundaries of different classes. It has been shown in [7] that LMNN delivers the state-of-the-art performance among most distance metric learning algorithms.

The work of LMNN [7] and PSDBoost [9] has directly inspired our work. Instead of using hinge loss in LMNN and PSDBoost, we use the exponential loss function in order to derive an AdaBoost-like optimization procedure. Hence, despite similar purposes, our algorithm differs essentially in the optimization. While the formulation of LMNN looks more similar to support vector machines (SVM's) and PSDBoost to LPBoost, our algorithm, termed BOOSTMETRIC, largely draws upon AdaBoost [10].

In many cases, it is difficult to find a global optimum in the projection matrix $\mathbf{L}$ [6]. Reformulation-linearization is a typical technique in convex optimization to relax and convexify the problem [11]. In metric learning, much existing work instead learns $\mathbf{X} = \mathbf{L}\mathbf{L}^\top$ for seeking a global optimum, *e.g.*, [4, 7, 12, 8]. The price is heavy computation and poor scalability: it is not trivial to preserve the semidefiniteness of $\mathbf{X}$ during the course of learning. Standard approaches like interior point Newton methods require the Hessian, which usually requires $O(D^4)$ resources (where D is the input dimension). It could be prohibitive for many real-world problems. Alternative projected (sub-)gradient is adopted in [7, 4, 8]. The disadvantages of this algorithm are: (1) not easy to implement; (2) many parameters involved; (3) slow convergence. PSDBoost [9] converts the particular semidefinite program in metric learning into a sequence of linear programs (LP's). At each iteration of PSDBoost, an LP needs to be solved as in LPBoost, which scales around $O(J^{3.5})$ with J the number of iterations (and therefore variables). As J increases, the scale of the LP becomes larger. Another problem is that PSDBoost needs to store all the weak learners (the rank-one matrices) during the optimization. When the input dimension D is large, the memory required is proportional to JD^2, which can be prohibitively huge at a late iteration J. Our proposed algorithm solves both of these problems.

Based on the observation from [9] that any positive semidefinite matrix can be decomposed into a linear positive combination of trace-one rank-one matrices, we propose BOOSTMETRIC for learning a p.s.d. matrix. The weak learner of BOOSTMETRIC is a rank-one p.s.d. matrix as in PSDBoost. The proposed BOOSTMETRIC algorithm has the following desirable properties: (1) BOOSTMETRIC is efficient and scalable. Unlike most existing methods, no semidefinite programming is required. At each iteration, only the largest eigenvalue and its corresponding eigenvector are needed. (2) BOOSTMETRIC can accommodate various types of constraints. We demonstrate learning a Mahalanobis metric by proximity comparison constraints. (3) Like AdaBoost, BOOSTMETRIC does not have any parameter to tune. The user only needs to know when to stop. In contrast, both LMNN and PSDBoost have parameters to cross validate. Also like AdaBoost it is easy to implement. No sophisticated optimization techniques such as LP solvers are involved. Unlike PSDBoost, we do not need to store all the weak learners. The efficacy and efficiency of the proposed BOOSTMETRIC is demonstrated on various datasets.

Throughout this paper, a matrix is denoted by a bold upper-case letter ($\mathbf{X}$); a column vector is denoted by a bold lower-case letter ($\boldsymbol{x}$). The ith row of $\mathbf{X}$ is denoted by $\mathbf{X}_{i:}$ and the ith column $\mathbf{X}_{:i}$. $\mathbf{Tr}(\cdot)$ is the trace of a symmetric matrix and $\langle \mathbf{X}, \mathbf{Z} \rangle = \mathbf{Tr}(\mathbf{X}\mathbf{Z}^\top) = \sum_{ij} \mathbf{X}_{ij}\mathbf{Z}_{ij}$ calculates the inner product of two matrices. An element-wise inequality between two vectors like $\boldsymbol{u} \leq \boldsymbol{v}$ means $u_i \leq v_i$ for all i. We use $\mathbf{X} \succcurlyeq 0$ to indicate that matrix $\mathbf{X}$ is positive semidefinite.

2 Algorithms

2.1 Distance Metric Learning

As discussed, the Mahalanobis metric is equivalent to linearly transform the data by a projection matrix $\mathbf{L} \in \mathbb{R}^{D \times d}$ (usually $D \geq d$) before calculating the standard Euclidean distance:

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

Although one can learn $\mathbf{L}$ directly as many conventional approaches do, in this setting, non-convex constraints are involved, which make the problem difficult to solve. As we will show, in order to *convexify* these conditions, a new variable $\mathbf{X} = \mathbf{L}\mathbf{L}^\top$ is introduced instead. This technique has been used widely in convex optimization and machine learning such as [12]. If $\mathbf{X} = \mathbf{I}$, it reduces to the Euclidean distance. If $\mathbf{X}$ is diagonal, the problem corresponds to learning a metric in which the different features are given different weights, *a.k.a.* feature weighting.

In the framework of large-margin learning, we want to maximize the distance between $\mathbf{dist}_{ij}$ and $\mathbf{dist}_{ik}$. That is, we wish to make $\mathbf{dist}_{ij}^2 - \mathbf{dist}_{ik}^2 = (\mathbf{a}_i - \mathbf{a}_k)^\top \mathbf{X} (\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X} (\mathbf{a}_i - \mathbf{a}_j)$ as large as possible under some regularization. To simplify notation, we rewrite the distance between $\mathbf{dist}_{ij}^2$ and $\mathbf{dist}_{ik}^2$ as $\mathbf{dist}_{ij}^2 - \mathbf{dist}_{ik}^2 = \langle \mathbf{A}_r, \mathbf{X} \rangle$,

$$\mathbf{A}_r = (\mathbf{a}_i - \mathbf{a}_k)(\mathbf{a}_i - \mathbf{a}_k)^\top - (\mathbf{a}_i - \mathbf{a}_j)(\mathbf{a}_i - \mathbf{a}_j)^\top, \tag{2}$$

$r = 1, \cdots, |\mathcal{S}|$. $|\mathcal{S}|$ is the size of the set $\mathcal{S}$.

2.2 Learning with Exponential Loss

We derive a general algorithm for p.s.d. matrix learning with exponential loss. Assume that we want to find a p.s.d. matrix $\mathbf{X} \succcurlyeq 0$ such that a bunch of constraints

$$\langle \mathbf{A}_r, \mathbf{X} \rangle > 0, r = 1, 2, \cdots,$$

are satisfied as *well* as possible. These constraints need not be all strictly satisfied. We can define the margin $\rho_r = \langle \mathbf{A}_r, \mathbf{X} \rangle, \forall r$. By employing exponential loss, we want to optimize

$$\min \log\bigl(\textstyle\sum_{r=1}^{|\mathcal{S}|} \exp -\rho_r\bigr) + v\, \mathbf{Tr}(\mathbf{X}) \quad \text{s.t. } \rho_r = \langle \mathbf{A}_r, \mathbf{X} \rangle, r = 1, \cdots, |\mathcal{S}|,\ \mathbf{X} \succcurlyeq 0. \tag{P0}$$

Note that: (1) We have worked on the logarithmic version of the sum of exponential loss. This transform does not change the original optimization problem of sum of exponential loss because the logarithmic function is strictly monotonically decreasing. (2) A regularization term $\mathbf{Tr}(\mathbf{X})$ has been applied. Without this regularization, one can always multiply an arbitrarily large factor to $\mathbf{X}$ to make the exponential loss approach zero in the case of all constraints being satisfied. This trace-norm regularization may also lead to low-rank solutions. (3) An auxiliary variable $\rho_r, r = 1, \ldots$ must be introduced for deriving a meaningful dual problem, as we show later.

We can decompose $\mathbf{X}$ into: $\mathbf{X} = \sum_{j=1}^J w_j \mathbf{Z}_j$, with $w_j \geq 0$, $\mathbf{rank}(\mathbf{Z}_j) = 1$ and $\mathbf{Tr}(\mathbf{Z}_j) = 1, \forall j$. So

$$\rho_r = \langle \mathbf{A}_r, \mathbf{X} \rangle = \Bigl\langle \mathbf{A}_r, \textstyle\sum_{j=1}^J w_j \mathbf{Z}_j \Bigr\rangle = \textstyle\sum_{j=1}^J w_j \langle \mathbf{A}_r, \mathbf{Z}_j \rangle = \sum_{j=1}^J w_j \mathbf{H}_{rj} = \mathbf{H}_{r:} \boldsymbol{w}, \forall r. \tag{3}$$

Here $\mathbf{H}_{rj}$ is a shorthand for $\mathbf{H}_{rj} = \langle \mathbf{A}_r, \mathbf{Z}_j \rangle$. Clearly $\mathbf{Tr}(\mathbf{X}) = \sum_{j=1}^J w_j\, \mathbf{Tr}(\mathbf{Z}_j) = \mathbf{1}^\top \boldsymbol{w}$.

2.3 The Lagrange Dual Problem

We now derive the Lagrange dual of the problem we are interested in. The original problem (P0) now becomes

$$\min \log\bigl(\textstyle\sum_{r=1}^{|\mathcal{S}|} \exp -\rho_r\bigr) + v \mathbf{1}^\top \boldsymbol{w}, \text{ s.t. } \rho_r = \mathbf{H}_{r:} \boldsymbol{w}, r = 1, \cdots, |\mathcal{S}|,\ \boldsymbol{w} \geq \mathbf{0}. \tag{P1}$$

In order to derive its dual, we write its Lagrangian

$$L(\boldsymbol{w}, \boldsymbol{\rho}, \boldsymbol{u}, \boldsymbol{p}) = \log\bigl(\textstyle\sum_{r=1}^{|\mathcal{S}|} \exp -\rho_r\bigr) + v \mathbf{1}^\top \boldsymbol{w} + \sum_{r=1}^{|\mathcal{S}|} u_r (\rho_r - \mathbf{H}_{r:} \boldsymbol{w}) - \boldsymbol{p}^\top \boldsymbol{w}, \tag{4}$$

with $\boldsymbol{p} \geq 0$. Here $\boldsymbol{u}$ and $\boldsymbol{p}$ are Lagrange multipliers. The dual problem is obtained by finding the saddle point of L; *i.e.*, $\sup_{\boldsymbol{u},\boldsymbol{p}} \inf_{\boldsymbol{w},\boldsymbol{\rho}} L$.

$$\inf_{\boldsymbol{w},\boldsymbol{\rho}} L = \inf_{\boldsymbol{\rho}} \overbrace{\log\left(\textstyle\sum_{r=1}^{|\boldsymbol{S}|} \exp -\rho_r\right) + \boldsymbol{u}^\top \boldsymbol{\rho}}^{L_1} + \inf_{\boldsymbol{w}} \overbrace{(v\mathbf{1}^\top - \textstyle\sum_{r=1}^{|\boldsymbol{S}|} u_r \mathbf{H}_{r:} - \boldsymbol{p}^\top)\boldsymbol{w}}^{L_2} = -\textstyle\sum_{r=1}^{|\boldsymbol{S}|} u_r \log u_r.$$

The infimum of L_1 is found by setting its first derivative to zero and we have:

$$\inf_{\boldsymbol{\rho}} L_1 = \begin{cases} -\sum_r u_r \log u_r & \text{if } \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1, \\ -\infty & \text{otherwise.} \end{cases}$$

The infimum is Shannon entropy. L_2 is linear in $\boldsymbol{w}$, hence L_2 must be $\mathbf{0}$. It leads to

$$\textstyle\sum_{r=1}^{|\boldsymbol{S}|} u_r \mathbf{H}_{r:} \leq v\mathbf{1}^\top. \tag{5}$$

The Lagrange dual problem of (P1) is an entropy maximization problem, which writes

$$\max_{\boldsymbol{u}} \; - \textstyle\sum_{r=1}^{|\boldsymbol{S}|} u_r \log u_r, \text{ s.t. } \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1, \text{and (5).} \tag{D1}$$

Weak and strong duality hold under mild conditions [11]. That means, one can usually solve one problem from the other. The KKT conditions link the optimal between these two problems. In our case, it is

$$u_r^\star = \frac{\exp -\rho_r^\star}{\sum_{k=1}^{|\boldsymbol{S}|} \exp -\rho_k^\star}, \forall r. \tag{6}$$

While it is possible to devise a totally-corrective column generation based optimization procedure for solving our problem as the case of LPBoost [13], we are more interested in considering *one-at-a-time* coordinate-wise descent algorithms, as the case of AdaBoost [10], which has the advantages: (1) computationally efficient and (2) parameter free. Let us start from some basic knowledge of column generation because our coordinate descent strategy is inspired by column generation.

If we knew all the bases $\mathbf{Z}_j (j = 1 \ldots J)$ and hence the entire matrix $\mathbf{H}$ is known, then either the primal (P1) or the dual (D1) could be trivially solved (at least in theory) because both are convex optimization problems. We can solve them in polynomial time. Especially the primal problem is convex minimization with simple nonnegativeness constraints. Off-the-shelf software like LBFGS-B [14] can be used for this purpose. Unfortunately, in practice, we do not access all the bases: the number of possible $\mathbf{Z}$'s is infinite. In convex optimization, column generation is a technique that is designed for solving this difficulty.

Instead of directly solving the primal problem (P1), we find the most violated constraint in the dual (D1) iteratively for the current solution and add this constraint to the optimization problem. For this purpose, we need to solve

$$\hat{\mathbf{Z}} = \operatorname{argmax}_{\mathbf{Z}} \left\{ \textstyle\sum_{r=1}^{|\boldsymbol{S}|} u_r \langle \mathbf{A}_r, \mathbf{Z} \rangle, \text{ s.t. } \mathbf{Z} \in \Omega_1 \right\}. \tag{7}$$

Here Ω_1 is the set of trace-one rank-one matrices. We discuss how to efficiently solve (7) later. Now we move on to derive a coordinate descent optimization procedure.

2.4 Coordinate Descent Optimization

We show how an AdaBoost-like optimization procedure can be derived for our metric learning problem. As in AdaBoost, we need to solve for the primal variables w_j given all the weak learners up to iteration j.

Optimizing for w_j Since we are interested in the *one-at-a-time* coordinate-wise optimization, we keep $w_1, w_2, \ldots, w_{j-1}$ fixed when solving for w_j. The cost function of the primal problem is (in the following derivation, we drop those terms irrelevant to the variable w_j)

$$C_p(w_j) = \log\left[\textstyle\sum_{r=1}^{|\boldsymbol{S}|} \exp(-\rho_r^{j-1}) \cdot \exp(-\mathbf{H}_{rj} w_j)\right] + v w_j.$$

Clearly, C_p is convex in w_j and hence there is only one minimum that is also globally optimal. The first derivative of C_p w.r.t. w_j vanishes at optimality, which results in

$$\textstyle\sum_{r=1}^{|\boldsymbol{S}|} (\mathbf{H}_{rj} - v) u_r^{j-1} \exp(-w_j \mathbf{H}_{rj}) = 0. \tag{8}$$

Algorithm 1 Bisection search for w_j.

```
Input: An interval $[w_l, w_u]$ known to contain the optimal value of $w_j$ and convergence tolerance $\varepsilon > 0$.
1 repeat
2     $w_j = 0.5(w_l + w_u)$;
3     if l.h.s. of (8) $> 0$ then
4         $w_l = w_j$;
5     else
6         $w_u = w_j$.
7 until $w_u - w_l < \varepsilon$ ;
Output: $w_j$.
```

If $\mathbf{H}_{rj}$ is discrete, such as $\{+1, -1\}$ in standard AdaBoost, we can obtain a close-form solution similar to AdaBoost. Unfortunately in our case, $\mathbf{H}_{rj}$ can be any real value. We instead use bisection to search for the optimal w_j. The bisection method is one of the root-finding algorithms. It repeatedly divides an interval in half and then selects the subinterval in which a root exists. Bisection is a simple and robust, although it is not the fastest algorithm for root-finding. Newton-type algorithms are also applicable here. Algorithm 1 gives the bisection procedure. We have utilized the fact that the l.h.s. of (8) must be positive at w_l. Otherwise no solution can be found. When $w_j = 0$, clearly the l.h.s. of (8) is positive.

Updating $\boldsymbol{u}$ The rule for updating the dual variable $\boldsymbol{u}$ can be easily obtained from (6). At iteration j, we have

$$u_r^j \propto \exp -\rho_r^j \propto u_r^{j-1} \exp(-\mathbf{H}_{rj} w_j), \text{ and } \textstyle\sum_{r=1}^{|S|} u_r^j = 1,$$

derived from (6). So once w_j is calculated, we can update $\boldsymbol{u}$ as

$$u_r^j = \frac{u_r^{j-1} \exp(-\mathbf{H}_{rj} w_j)}{z}, r = 1, \dots, |\mathcal{S}|, \tag{9}$$

where z is a normalization factor so that $\sum_{r=1}^{|S|} u_r^j = 1$. This is exactly the same as AdaBoost.

2.5 Base Learning Algorithm

In this section, we show that the optimization problem (7) can be exactly and efficiently solved using eigenvalue-decomposition (EVD). From $\mathbf{Z} \succcurlyeq 0$ and $\mathbf{rank}(\mathbf{Z}) = 1$, we know that $\mathbf{Z}$ has the format: $\mathbf{Z} = \boldsymbol{\xi}\boldsymbol{\xi}^\top$, $\boldsymbol{\xi} \in \mathbb{R}^D$; and $\mathbf{Tr}(\mathbf{Z}) = 1$ means $\|\boldsymbol{\xi}\|_2 = 1$. We have

$$\textstyle\sum_{r=1}^{|S|} u_r \langle \mathbf{A}_r, \mathbf{Z} \rangle = \langle \sum_{r=1}^{|S|} u_r \mathbf{A}_r, \mathbf{Z} \rangle = \boldsymbol{\xi}^\top \big(\sum_{r=1}^{|S|} u_r \mathbf{A}_r \big) \boldsymbol{\xi}.$$

By denoting

$$\hat{\mathbf{A}} = \textstyle\sum_{r=1}^{|S|} u_r \mathbf{A}_r, \tag{10}$$

the base learning optimization equals: $\max_{\boldsymbol{\xi}} \ \boldsymbol{\xi}^\top \hat{\mathbf{A}} \boldsymbol{\xi}, \ \text{s.t.} \|\boldsymbol{\xi}\|_2 = 1$. It is clear that the largest eigenvalue of $\hat{\mathbf{A}}$, $\lambda_{\max}(\hat{\mathbf{A}})$, and its corresponding eigenvector $\boldsymbol{\xi}_1$ gives the solution to the above problem. Note that $\hat{\mathbf{A}}$ is symmetric. Also see [9] for details.

$\lambda_{\max}(\hat{\mathbf{A}})$ is also used as one of the stopping criteria of the algorithm. Form the condition (5), $\lambda_{\max}(\hat{\mathbf{A}}) < v$ means that we are not able to find a new base matrix $\hat{\mathbf{Z}}$ that violates (5)—the algorithm converges. We summarize our main algorithmic results in Algorithm 2.

3 Experiments

3.1 Classification on Benchmark Datasets

We evaluate BoostMetric on 15 datasets of different sizes. Some of the datasets have very high dimensional inputs. We use PCA to decrease the dimensionality before training on these datasets (datasets 2-6). PCA pre-processing helps to eliminate noises and speed up computation. We have

Algorithm 2 Positive semidefinite matrix learning with boosting.

```
Input:
    • Training set triplets $(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{S}$; Compute $\mathbf{A}_r, r = 1, 2, \cdots$, using (2).
    • $J$: maximum number of iterations;
    • (optional) regularization parameter $v$; We may simply set $v$ to a very small value, e.g., $10^{-7}$.
1 Initialize: $u_r^0 = \frac{1}{|S|}, r = 1 \cdots |\mathcal{S}|$;
2 for $j = 1, 2, \cdots, J$ do
3     · Find a new base $\mathbf{Z}_j$ by finding the largest eigenvalue ($\lambda_{\max}(\hat{\mathbf{A}})$) and its eigenvector of $\hat{\mathbf{A}}$ in (10);
4     · if $\lambda_{\max}(\hat{\mathbf{A}}) < v$ then
5         break (converged);
6     · Compute $w_j$ using Algorithm 1;
7     · Update $\boldsymbol{u}$ to obtain $u_r^j, r = 1, \cdots |\mathcal{S}|$ using (9);
Output: The final p.s.d. matrix $\mathbf{X} \in \mathbb{R}^{D \times D}$, $\mathbf{X} = \sum_{j=1}^{J} w_j \mathbf{Z}_j$.
```

used USPS and MNIST handwritten digits, ORL face recognition datasets, Columbia University Image Library (COIL20)[1], and UCI machine learning datasets[2] (datasets 7-13), Twin Peaks and Helix. The last two are artificial datasets[3].

Experimental results are obtained by averaging over 10 runs (except USPS-1). We randomly split the datasets for each run. We have used the same mechanism to generate training triplets as described in [7]. Briefly, for each training point $\mathbf{a}_i$, k nearest neighbors that have same labels as y_i (targets), as well as k nearest neighbors that have different labels from y_i (imposers) are found. We then construct triplets from $\mathbf{a}_i$ and its corresponding targets and imposers. For all the datasets, we have set $k = 3$ except that $k = 1$ for datasets USPS-1, ORLFace-1 and ORLFace-2 due to their large size. We have compared our method against a few methods: Xing *et al* [4], RCA [5], NCA [6] and LMNN [7]. LMNN is one of the state-of-the-art according to recent studies such as [15]. Also in Table 1, "Euclidean" is the baseline algorithm that uses the standard Euclidean distance. The codes for these compared algorithms are downloaded from the corresponding authors' websites. We have released our codes for BOOSTMETRIC at [16]. Experiment setting for LMNN follows [7]. For BOOSTMETRIC, we have set $v = 10^{-7}$, the maximum number of iterations $J = 500$. As we can see from Table 1, we can conclude: (1) BOOSTMETRIC consistently improves kNN classification using Euclidean distance on most datasets. So learning a Mahalanobis metric based upon the large margin concept does lead to improvements in kNN classification. (2) BOOSTMETRIC outperforms other algorithms in most cases (on 11 out of 15 datasets). LMNN is the second best algorithm on these 15 datasets statistically. LMNN's results are consistent with those given in [7]. (3) Xing *et al* [4] and NCA can only handle a few small datasets. In general they do not perform very well. A good initialization is important for NCA because NCA's cost function is non-convex and can only find a local optimum.

Influence of v Previously, we claim that our algorithm is parameter-free like AdaBoost. However, we do have a parameter v in BOOSTMETRIC. Actually, AdaBoost simply set $v = 0$. The coordinate-wise gradient descent optimization strategy of AdaBoost leads to an ℓ_1-norm regularized maximum margin classifier [17]. It is shown that AdaBoost minimizes its loss criterion with an ℓ_1 constraint on the coefficient vector. Given the similarity of the optimization of BOOSTMETRIC with AdaBoost, we conjecture that BOOSTMETRIC has the same property. Here we empirically prove that *as long as v is sufficiently small, the final performance is not affected by the value of v*. We have set v from 10^{-8} to 10^{-4} and run BOOSTMETRIC on 3 UCI datasets. Table 2 reports the final 3NN classification error with different v. The results are nearly identical.

Computational time As we discussed, one major issue in learning a Mahalanobis distance is heavy computational cost because of the semidefiniteness constraint.

[1]`http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php`
[2]`http://archive.ics.uci.edu/ml/`
[3]`http://boosting.googlecode.com/files/dataset1.tar.bz2`

Table 1: Test classification error rates (%) of a 3-nearest neighbor classifier on benchmark datasets. Results of NCA and Xing *et al* [4] on large datasets are not available either because the algorithm does not converge or due to the out-of-memory problem.

	dataset	Euclidean	Xing *et al* [4]	RCA	NCA	LMNN	BoostMetric
1	USPS-1	5.18		32.71		7.51	**2.96**
2	USPS-2	3.56 (0.28)		5.57 (0.33)		2.18 (0.27)	**1.99 (0.24)**
3	ORLFace-1	3.33 (1.47)		5.75 (2.85)	3.92 (2.01)	6.67 (2.94)	**2.00 (1.05)**
4	ORLFace-2	5.33 (2.70)		4.42 (2.08)	3.75 (1.63)	**2.83 (1.77)**	3.00 (1.31)
5	MNIST	4.11 (0.43)		4.31 (0.42)		4.19 (0.49)	**4.09 (0.31)**
6	COIL20	0.19 (0.21)		0.32 (0.29)		2.41 (1.80)	**0.02 (0.07)**
7	Letters	5.74 (0.24)		5.06 (0.26)		4.34 (0.36)	**3.54 (0.18)**
8	Wine	26.23 (5.52)	10.38 (4.81)	**2.26 (1.95)**	27.36 (6.31)	5.47 (3.01)	2.64 (1.59)
9	Bal	18.13 (1.79)	11.12 (2.12)	19.47 (2.39)	**4.81 (1.80)**	11.87 (2.14)	8.93 (2.28)
10	Iris	2.22 (2.10)	**2.22 (2.10)**	3.11 (2.15)	2.89 (2.58)	2.89 (2.58)	2.89 (2.78)
11	Vehicle	30.47 (2.41)	28.66 (2.49)	21.42 (2.46)	22.61 (3.26)	22.57 (2.16)	**19.17 (2.10)**
12	Breast-Cancer	3.28 (1.06)	3.63 (0.93)	3.82 (1.15)	4.31 (1.10)	3.19 (1.43)	**2.45 (0.95)**
13	Diabetes	27.43 (2.93)	27.87 (2.71)	26.48 (1.61)	27.61 (1.55)	26.78 (2.42)	**25.04 (2.25)**
14	Twin Peaks	1.13 (0.09)		1.02 (0.09)		0.98 (0.11)	**0.14 (0.08)**
15	Helix	0.60 (0.12)		0.61 (0.11)		0.61 (0.13)	**0.58 (0.12)**

Table 2: Test error (%) of a 3-nearest neighbor classifier with different values of the parameter v. Each experiment is run 10 times. We report the mean and variance. As expected, as long as v is sufficiently small, in a wide range it almost does not affect the final classification performance.

v	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Bal	8.98 (2.59)	8.88 (2.52)	8.88 (2.52)	8.88 (2.52)	8.93 (2.52)
B-Cancer	2.11 (0.69)	2.11 (0.69)	2.11 (0.69)	2.11 (0.69)	2.11 (0.69)
Diabetes	26.0 (1.33)	26.0 (1.33)	26.0 (1.33)	26.0 (1.34)	26.0 (1.46)

Our algorithm is generally fast. It involves matrix operations and an EVD for finding its largest eigenvalue and its corresponding eigenvector. The time complexity of this EVD is $O(D^2)$ with D the input dimensions. We compare our algorithm's running time with LMNN in Fig. 1 on the artificial dataset (concentric circles). We vary the input dimensions from 50 to 1000 and keep the number of triplets fixed to 250. Instead of using standard interior-point SDP solvers that do not scale well, LMNN heuristically combines sub-gradient descent in both the matrices $\mathbf{L}$ and $\mathbf{X}$. At each iteration, $\mathbf{X}$ is projected back onto the p.s.d. cone using EVD. So a full EVD with time complexity $O(D^3)$ is needed. Note that LMNN is much faster than SDP solvers like CSDP [18]. As seen from Fig. 1, when the input dimensions are low, BoostMetric is comparable to LMNN. As expected, when the input dimensions become high, BoostMetric is significantly faster than LMNN. Note that our implementation is in Matlab. Improvements are expected if implemented in C/C++.

3.2 Visual Object Categorization and Detection

The proposed BoostMetric and the LMNN are further compared on four classes of the Caltech-101 object recognition database [19], including Motorbikes (798 images), Airplanes (800), Faces (435), and Background-Google (520). For each image, a number of interest regions are identified by the Harris-affine detector [20] and the visual content in each region is characterized by the SIFT descriptor [21]. The total number of local descriptors extracted from the images of the four classes

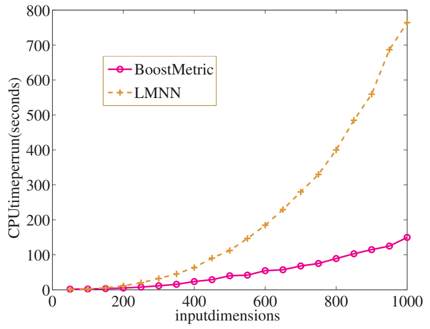

Figure 1: Computation time of the proposed BoostMetric and the LMNN method versus the input data's dimensions on an artificial dataset. BoostMetric is faster than LMNN with large input dimensions because at each iteration BoostMetric only needs to calculate the largest eigenvector and LMNN needs a full eigen-decomposition.

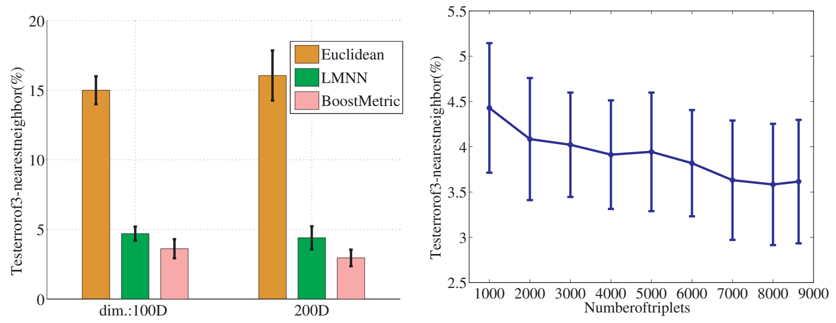

Figure 2: Test error (3-nearest neighbor) of BOOSTMETRIC on the Motorbikes *vs.* Airplanes datasets. The second figure shows the test error against the number of training triplets with a 100-word codebook. Test error of LMNN is $4.7\% \pm 0.5\%$ with 8631 triplets for training, which is worse than BOOSTMETRIC. For Euclidean distance, the error is much larger: $15\% \pm 1\%$.

are about $134,000$, $84,000$, $57,000$, and $293,000$, respectively. This experiment includes both object categorization (Motorbikes *vs.* Airplanes) and object detection (Faces *vs.* Background-Google) problems. To accumulate statistics, the images of two involved object classes are randomly split as 10 pairs of training/test subsets. Restricted to the images in a training subset (those in a test subset are only used for test), their local descriptors are clustered to form visual words by using k-means clustering. Each image is then represented by a histogram containing the number of occurrences of each visual word.

Motorbikes *vs.* Airplanes This experiment discriminates the images of a motorbike from those of an airplane. In each of the 10 pairs of training/test subsets, there are 959 training images and 639 test images. Two visual codebooks of size 100 and 200 are used, respectively. With the resulting histograms, the proposed BOOSTMETRIC and the LMNN are learned on a training subset and evaluated on the corresponding test subset. Their averaged classification error rates are compared in Fig. 2 (left). For both visual codebooks, the proposed BOOSTMETRIC achieves lower error rates than the LMNN and the Euclidean distance, demonstrating its superior performance. We also apply a linear SVM classifier with its regularization parameter carefully tuned by 5-fold cross-validation. Its error rates are $3.87\% \pm 0.69\%$ and $3.00\% \pm 0.72\%$ on the two visual codebooks, respectively. In contrast, a 3NN with BOOSTMETRIC has error rates $3.63\% \pm 0.68\%$ and $2.96\% \pm 0.59\%$. Hence, the performance of the proposed BOOSTMETRIC is comparable to or even slightly better than the SVM classifier. Also, Fig. 2 (right) plots the test error of the BOOSTMETRIC against the number of triplets for training. The general trend is that more triplets lead to smaller errors.

Faces *vs.* Background-Google This experiment uses the two object classes as a retrieval problem. The target of retrieval is the face images. The images in the class of Background-Google are randomly collected from the Internet and they are used to represent the non-target class. BOOSTMETRIC is first learned from a training subset and retrieval is conducted on the corresponding test subset. In each of the 10 training/test subsets, there are 573 training images and 382 test images. Again, two visual codebooks of size 100 and 200 are used. Each face image in a test subset is used as a query, and its distances from other test images are calculated by BOOSTMETRIC, LMNN and the Euclidean distance. For each metric, the *precision* of the retrieved top 5, 10, 15 and 20 images are computed. The retrieval precision for each query are averaged on this test subset and then averaged over the whole 10 test subsets. BOOSTMETRIC consistently attains the highest values, which again verifies its advantages over LMNN and the Euclidean distance. With a codebook size 200, very similar results are obtained. See [16] for the experiment results.

4 Conclusion

We have presented a new algorithm, BOOSTMETRIC, to learn a positive semidefinite metric using boosting techniques. We have generalized AdaBoost in the sense that the weak learner of BOOSTMETRIC is a matrix, rather than a classifier. Our algorithm is simple and efficient. Experiments show its better performance over a few state-of-the-art existing metric learning methods. We are currently combining the idea of on-line learning into BOOSTMETRIC to make it handle even larger datasets.

References

[1] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):607–616, 1996.

[2] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian. Distance learning for similarity estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(3):451–462, 2008.

[3] B. Jian and B. C. Vemuri. Metric learning using Iwasawa decomposition. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 1–6, Rio de Janeiro, Brazil, 2007. IEEE.

[4] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Proc. Adv. Neural Inf. Process. Syst.* MIT Press, 2002.

[5] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *J. Mach. Learn. Res.*, 6:937–965, 2005.

[6] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *Proc. Adv. Neural Inf. Process. Syst.* MIT Press, 2004.

[7] 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.*, pages 1473–1480, 2005.

[8] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Proc. Adv. Neural Inf. Process. Syst.*, 2005.

[9] C. Shen, A. Welsh, and L. Wang. PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Proc. Adv. Neural Inf. Process. Syst.*, pages 1473–1480, Vancouver, Canada, 2008.

[10] R. E. Schapire. Theoretical views of boosting and applications. In *Proc. Int. Conf. Algorithmic Learn. Theory*, pages 13–25, London, UK, 1999. Springer-Verlag.

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

[12] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comp. Vis.*, 70(1):77–90, 2006.

[13] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Mach. Learn.*, 46(1-3):225–254, 2002.

[14] C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, 1997.

[15] L. Yang, R. Jin, L. Mummert, R. Sukthankar, A. Goode, B. Zheng, S. Hoi, and M. Satyanarayanan. A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* IEEE computer Society Digital Library, November 2008, `http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.273`.

[16] `http://code.google.com/p/boosting/`.

[17] S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *J. Mach. Learn. Res.*, 5:941–973, 2004.

[18] B. Borchers. CSDP, a C library for semidefinite programming. *Optim. Methods and Softw.*, 11(1):613–623, 1999.

[19] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, April 2006.

[20] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *Int. J. Comp. Vis.*, 60(1):63–86, 2004.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, 60(2):91–110, 2004.