IPSJ Transactions on Computer Vision and Applications Vol.5 85–89 (July 2013)

[DOI: 10.2197/ipsjtcva.5.85]

Express Paper

A Discriminative Metric Learning Algorithm for Face Recognition

Tsuyoshi Kato[1,a)] Wataru Takei[1,b)] Shinichiro Omachi[2,c)]

Received: March 11, 2013, Accepted: April 24, 2013, Released: July 29, 2013

Abstract: Face recognition is a multi-class classification problem that has long attracted many researchers in the community of image analysis. We consider using the Mahalanobis distance for the task. Classically, the inverse of a covariance matrix has been chosen as the Mahalanobis matrix, a parameter of the Mahalanobis distance. Modern studies often employ machine learning algorithms called metric learning to determine the Mahalanobis matrix so that the distance is more discriminative, although they resort to eigen-decomposition requiring heavy computation. This paper presents a new metric learning algorithm that finds discriminative Mahalanobis matrices efficiently without eigen-decomposition, and shows promising experimental results on real-world face-image datasets.

Keywords: face recognition, metric learning, Mahalanobis distance, optimization, nearest neighbor classifier

1. Introduction

The problem of face recognition has continued to be tackled by many researchers in the field of image analysis, pattern recognition, and psychology, supported by a variety of applications such as biometric verification, surveillance, and database investigation [1], [5]. A face recognition system is an integration of various technologies including sensing devices, image processing, and pattern recognition. Among those technologies, pattern recognition is the most necessary technique and controls the performance of the overall face recognition system. This paper contributes to the stage of pattern recognition that identifies an unknown person from a still cropped image of the frontal face.

Many face recognition methods linearize a face image into a vector, to pose a statistical multiclass classification problem. Several attempts including PCA [1], [5], FDA [1], Mahalanobis distance [7] and their variants have been performed for the face recognition task. Our work focuses on the nearest neighbor approach which not only performs classification, but also provides useful information about how the input image is classified by showing its nearest neighbors. This property is suitable for interactive systems such as image search. This study employs Mahalanobis distance to improve classification accuracy (See **Fig. 1**). The Mahalanobis distance is expressed as $D_{\mathrm{maha}}(\boldsymbol{x}, \boldsymbol{m}; \boldsymbol{W}) = (\boldsymbol{x}-\boldsymbol{m})^\top \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{m})$ that computes a deviation of an input vector $\boldsymbol{x}$ from a specified point $\boldsymbol{m}$, with a *Mahalanobis matrix* $\boldsymbol{W}$.

Classically, the parameter $\boldsymbol{m}$ is set to the mean vector in the class, and $\boldsymbol{W}$ is to the inverse of the covariance matrix modified to address the small sample size problem. A single Mahalanobis matrix common to all classes is sometimes used by taking the inverse of the covariance matrix averaged over classes. In both the ways, the parameters are determined only with positive data, wit negative data, or data in the other classes discarded. Such approaches are called *generative learning*, which is a contrasting manner of so-called *discriminative learning* exploiting negative data to improve the classification boundaries. A classical statistical analysis, Fisher discriminant analysis, is a well-known example of discriminative learning, that yields different prediction results except for the binary classification case. Due to the enormous number of classes and the limitation of computational resources, generative learning approaches were often employed in the 1990s.

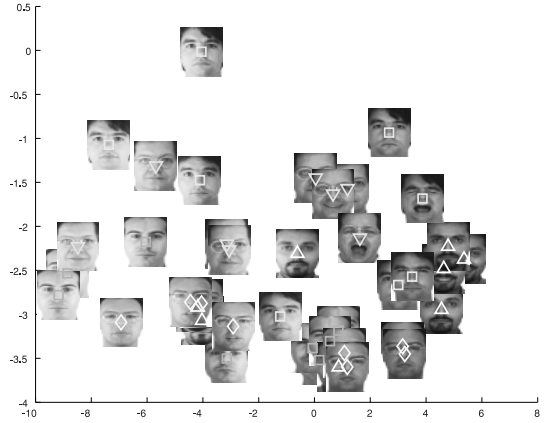
The advent of binary classifiers, such as the support vector machine (SVM), renewed the understanding of the importance of exploiting negative data, and bore a sequence of studies that developed learning machines to find the Mahalanobis matrices in a discriminative learning fashion [2], [3], [6], [7]. The learning of Mahalanobis matrices became *metric learning*. The learned Mahalanobis matrices are often presumed to be used subsequently in the nearest neighbor classifier. In this setting, metric learning methods determine $\boldsymbol{W}$ with many triplets, $\mathcal{R}_1, \ldots, \mathcal{R}_K$, each of which is given by $\mathcal{R}_k = (\boldsymbol{x}_{i_k}, \boldsymbol{x}_{j_k}, \boldsymbol{x}_{l_k}) \in \mathbb{R}^{n\times 3}$, $\forall k = 1, \ldots, K$, and $\boldsymbol{x}_{i_k}$ and $\boldsymbol{x}_{j_k}$ belong to different classes, but $\boldsymbol{x}_{l_k}$ belongs to the same class as that of $\boldsymbol{x}_{i_k}$, so that

$$D_{\mathrm{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{j_k}; \boldsymbol{W}) \geq D_{\mathrm{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{l_k}; \boldsymbol{W}) + \epsilon.$$

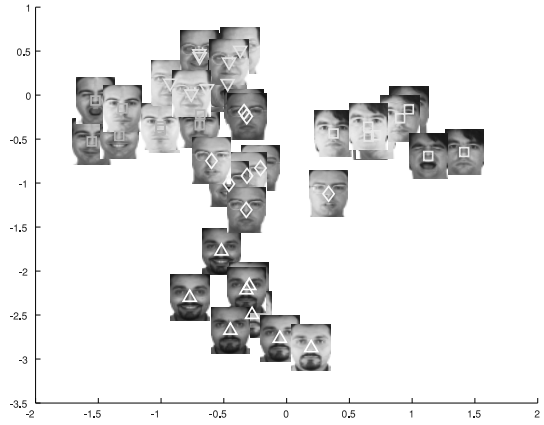
$\forall k = 1, \ldots, K$, where ϵ is a small positive constant. In other words, differences below ϵ between two distances, $D_{\mathrm{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{j_k}; \boldsymbol{W})$ and $D_{\mathrm{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{l_k}; \boldsymbol{W})$, are penalized as a *loss function* in their learning algorithms. This approach is called the *relative distance penalization*.

In this paper, we present a new metric learning algorithm aimed at tuning a nearest neighbor classifier for face recognition. Our algorithm is based on an implementation of the popular software `liblinear` developed by the `libsvm` team. The `liblinear`

[1] Graduate School of Engineering, Gunma University, Kiryu, Gunma 376–8515, Japan
[2] Graduate School of Engineering, Tohoku University, Sendai, Miyagi 980–8579, Japan
[a)] katotsu@cs.gunma-u.ac.jp
[b)] takei-wataru@kato-lab.cs.gunma-u.ac.jp
[c)] machi@ecei.tohoku.ac.jp



(a) Euclidean metric



(b) Malalanobis metric

Fig. 1 Projections onto two-dimensional principal subspace with different metrics. Plots in (a) are of the Euclidean metric, and plots in (b) are obtained using the metric aquired with our metric learning algorithm, BDRM. Overlaps among classes are reduced dramatically by metric learning.

software solves the linear L2-SVM problem minimizing

$$J(\boldsymbol{w}) = r(\boldsymbol{w}) + C\sum_{k=1}^{K}(1 - \langle \boldsymbol{w}, y_k \boldsymbol{x}_k\rangle)_+^2$$

as described in Hsieh et al.'s paper [4]. Therein, $C > 0$ is a constant called the regularization parameter, and $(\boldsymbol{x}_k, y_k) \in \mathbb{R}^n \times \{\pm 1\}$ is the k-th training data. The first term $r(\boldsymbol{w})$ is called the regularization function, and, in the case of SVM, it is defined as the square of the $L2$-norm: $r(\boldsymbol{w}) \equiv \frac{1}{2}\|\boldsymbol{w}\|^2$. We generalize the regularization function to Bregman divergences, and extend their optimization algorithm for the generalized regularization function. We refer to the generalized algorithm as a *Bregman divergence regularized machine* (*BDRM*). We developed the new metric learning algorithm as an instance of BDRM. The framework of BDRM inherits the property of the linear convergence of Hsieh et al.'s algorithm [4], and so does our metric learning algorithm.

Remarkably, for obtaining the optimal Mahalanobis matrix, our metric learning algorithm is exempted from eigen-decomposition, which is computationally expensive but necessary for most of existing metric learning methods [2], [6], [7]. That property theoretically guarantees to cut down, to $O(n^2 K \log(1/\varepsilon))$, the total computational cost for obtaining an ε-accurate solution [*1] of the optimal Mahalanobis matrix.

[*1] The two variables, ϵ and ε, are distinct in this paper.

2. Related Work

Importantly, Mahalanobis matrices must be positive definite. Metric learning algorithms typically follow the modern machine learning techniques by finding a Mahalanobis matrix that minimizes the sum of a *regularization function* and a loss function over the positive definite cone [2], [6], [7]. To ensure this constraint, many methods [6], [7] project, in every iteration, the $n \times n$ symmetric matrix onto the positive definite cone, which requires the eigen-decomposition to be computationally intensive, $O(n^3)$, where n is the number of features.

Davis et al. developed a break-through algorithm, ITML [2], that addresses the issue of the expensive computational cost for keeping the positive definiteness. ITML employs the Bregman divergences for a regularization function as well as a loss function. They discovered a surprising fact that each iteration of the successive projection algorithm can be performed in $O(n^2)$ computation, if LogDet divergence is chosen as a Bregman divergence, yet the positive definiteness of $\boldsymbol{W}$ and the linear convergence of the iterative algorithm are still guaranteed.

A major difference of ITML from the other metric learning methods [6], [7] is that users have to give, in advance, two constant parameters that represent the lower bound b_ℓ for distances between examples in different classes, and the upper bound b_u for distances between examples in same classes. The loss function is formulated with the LogDet divergence to evaluate how the pre-defined two bounds are violated, instead of using the relative distance penalization.

A contribution of this paper is to demonstrate that a metric learning algorithm using the relative distance penalization can be constructed, yet possessing theoretical guarantee of keeping the computational cost of each iteration $O(n^2)$ and still ensuring the positive definiteness of the Mahalanobis matrix $\boldsymbol{W}$ without requiring two user-defined bounds, b_ℓ and b_u.

3. Bregman Divergence Regularized Machine

In this section, we present a new framework for learning machines named BDRM. Bregman divergence is a class of a large number of functions including the squared Euclidean distance, Itakura-Saito distance, KL-divergence etc. Bregman divergence is defined with a seed function φ which is assumed to be continuously-differentiable, real-valued and strictly convex. A *Bregman divergence* $D_\varphi : \mathrm{dom}\varphi \times \mathrm{ri}(\mathrm{dom}\varphi) \to [0, +\infty)$ is constructed with φ as

$$D_\varphi(\boldsymbol{x}; \boldsymbol{y}) \equiv \varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y}) - \langle \boldsymbol{x} - \boldsymbol{y}, \nabla\varphi(\boldsymbol{y})\rangle$$

where $\mathrm{dom}\varphi$ is φ's domain, and $\mathrm{ri}(\mathrm{dom}\varphi)$ is the relative interior of $\mathrm{dom}\varphi$. We consider the following class of learning machines:

$$\begin{aligned} &\min \quad D_\varphi(\boldsymbol{w}; \boldsymbol{w}_0) + \frac{1}{2}\sum_{k=1}^{K} c_k \xi_k^2, \quad \boldsymbol{w} \in \mathrm{dom}\varphi \\ &\text{subject to} \quad \forall k \ \langle \boldsymbol{a}_k, \boldsymbol{w}\rangle \geq \epsilon_k - \xi_k, \quad \xi_k \geq 0, \\ &\text{where} \quad \boldsymbol{w}_0 \in \mathrm{ri}(\mathrm{dom}\varphi), \quad \boldsymbol{c} = [c_1, \ldots, c_K]^\top > \boldsymbol{0}_K, \\ &\qquad \boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_K]^\top \geq \boldsymbol{0}_K, \qquad \forall k, \ \boldsymbol{a}_k \in \mathbb{R}^n \setminus \{\boldsymbol{0}_n\}. \end{aligned} \tag{1}$$

L2-SVM is shown to be an instance of these learning machines by

Algorithm 1 General BDRM.

1: $\boldsymbol{v} := \nabla\varphi(\boldsymbol{w}_0)$; $\boldsymbol{\alpha} := \mathbf{0}_K$;
2: **for** $t = 1, 2, \ldots$ **do**
3: **for** $k = 1, \ldots, K$ **do**
4: $\boldsymbol{u}_k = \boldsymbol{v} - \alpha_k \boldsymbol{a}_k$;
5: **if** $\langle \boldsymbol{a}_k, \nabla\varphi_*(\boldsymbol{u}_k)\rangle \geq \epsilon_k$ **then**
6: $\alpha_k = 0$;
7: **else**
8: Find positive α_k satisfying a nonlinear equation

$$\langle \boldsymbol{a}_k, \nabla\varphi_*(\boldsymbol{u}_k - \alpha_k \boldsymbol{a}_k)\rangle = \epsilon_k - \alpha_k / c_k. \quad (2)$$

9: **end if**
10: $\boldsymbol{v} := \boldsymbol{u}_k + \alpha_k \boldsymbol{a}_k$;
11: **end for**
12: **end for**

setting $\varphi(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$, $\boldsymbol{w}_0 = \mathbf{0}_n$, $\boldsymbol{a}_k = y_k \boldsymbol{x}_k$, $\boldsymbol{c} = C\mathbf{1}_K$, and $\boldsymbol{\epsilon} = \mathbf{1}_K$. Hsieh et al. [4] dualize the optimization problem to solve the primal problem by maximizing the dual objective function with respect to dual variables. Introducing dual variables, $\boldsymbol{\alpha} \in \mathbb{R}_+^K$, the dual function of Eq. (1) is given by:

$$g(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w},\boldsymbol{\xi}} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})$$

where $L(\boldsymbol{w}, \boldsymbol{\alpha})$ is the Lagrangean function written as:

$$L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = D_\varphi(\boldsymbol{w}; \boldsymbol{w}_0) + \frac{1}{2}\sum_{k=1}^K c_k \xi_k^2 + \sum_{k=1}^K \alpha_k(\epsilon_k - \langle \boldsymbol{a}_k, \boldsymbol{w}\rangle - \xi_k).$$

Note that the non-negativeness of slack variables ξ_k is ensured even without Lagrangean multipliers for the constraints. Following their approach, we employ the coordinate ascent method for maximization of the dual problem, and we obtain Algorithm 1. Here, $\varphi_*(\cdot)$ denotes the convex conjugate of $\varphi(\cdot)$.

This algorithm inherits a favorable property of Hsieh et al.'s algorithm, linear convergence, because Algorithm 1 still exactly performs a linearly converged coordinate ascent method exactly. If $g(\boldsymbol{\alpha}) + \varepsilon \geq g(\boldsymbol{\alpha}^*)$, where $\boldsymbol{\alpha}^*$ is the optimal solution, the solution $\boldsymbol{\alpha}$ is said to be ε-accurate. Thanks to the linear convergence, an ε-accurate solution is obtained in $O(\log(1/\varepsilon))$ iterations. Another point that determines the total time complexity is how fast each iteration works. Since Step 4 and Step 10 need only $O(n)$ computation, the time complexity of each iteration depends on the computation of $\nabla\varphi_*$ for Step 5 and Step 8. Hence, if the Bregman divergence is chosen so $\nabla\varphi_*$ can be computed quickly, the algorithm works efficiently even in a large scale scenario.

4. Metric Learning as a BDRM

This section employs the framework of BDRM to devise a new metric learning algorithm without a time-consuming step of projection onto the positive definite cone. We here consider a multiclass classification setting. First of all, we pick many triplets $\mathcal{R}_1, \ldots, \mathcal{R}_K$ from training data where each triplet contains three examples $\mathcal{R}_k = (\boldsymbol{x}_{i_k}, \boldsymbol{x}_{j_k}, \boldsymbol{x}_{l_k})$ where $\boldsymbol{x}_{i_k}$ and $\boldsymbol{x}_{j_k}$ are in different classes, but $\boldsymbol{x}_{l_k}$ is in the same class as that of $\boldsymbol{x}_{i_k}$. We have denoted the number of triplets by K. The acquired Mahalanobis distance is used in the prediction stage using a nearest neighbor classifier. Ideally, we wish to obtain a Mahalanobis matrix $\boldsymbol{W}$ such that $D_{\text{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{j_k}; \boldsymbol{W}) \geq D_{\text{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{l_k}; \boldsymbol{W}) + \epsilon$, where ϵ is a small positive constant. In our experiments described later, we form the $\mathcal{R}_k$'s as follows: For each class, take every pair of examples, i and l, in the class; then, for each pair (i, l), find five nearest neighbors of i according to the Euclidean distance; finally, form a triplet as $\mathcal{R}_k = (i, j, l)$ for each pair (i, l) and each of its nearest neighbors j.

We use a regularization function to avoid over-fitting. Following ITML, we employ LogDet divergence defined by

$$\boldsymbol{D}_{\text{ld}}(\boldsymbol{W}; \boldsymbol{W}_0) = \text{tr}\left(\boldsymbol{W}\boldsymbol{W}_0^{-1}\right) - \log\det \boldsymbol{W}\boldsymbol{W}_0^{-1} - n$$

as a regularization function. The LogDet divergence is constructed by defining the seed function as $\varphi_{\text{ld}}(\boldsymbol{W}) = -\log\det \boldsymbol{W}$ whose derivative is given by $\nabla\varphi_{\text{ld}}(\boldsymbol{W}) = -\boldsymbol{W}^{-1}$. Usually, we set $\boldsymbol{W}_0 = \boldsymbol{I}_n$ to restrain a Mahalanobis metric far from the Euclidean metric, but different choices of $\boldsymbol{W}_0$ are also useful for some special settings such as transfer learning. Furthermore, to ensure the feasibility of the learning problem, we introduce slack variables ξ_k.

Then, we obtain the following optimization problem:

$$\begin{aligned} &\text{min} \quad D_{\text{ld}}(\boldsymbol{W}; \boldsymbol{W}_0) + \frac{C}{2}\sum_{k=1}^K \xi_k^2, \\ &\text{wrt} \quad \boldsymbol{W} \in \mathbb{S}_{++}^n, \text{and } \boldsymbol{\xi} \in \mathbb{R}^K \\ &\text{subject to} \quad \forall k,\ D_{\text{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{j_k}; \boldsymbol{W}) \\ &\qquad\qquad \geq D_{\text{maha}}(\boldsymbol{x}_{i_k}, \boldsymbol{x}_{l_k}; \boldsymbol{W}) + \epsilon - \xi_k \end{aligned} \quad (3)$$

where $\boldsymbol{W}_0 \succeq \boldsymbol{O}_n$, $C > 0$, and $\epsilon > 0$ are constant, and $\mathbb{S}_{++}^n$ denotes a set of $n \times n$ strictly positive definite matrices. It can be verified that this formulation is a BDRM with

$$\begin{aligned} &\boldsymbol{w} = \text{vec}(\boldsymbol{W}), \quad \boldsymbol{w}_0 = \text{vec}(\boldsymbol{W}_0), \quad \boldsymbol{c} = C\mathbf{1}_K, \\ &\boldsymbol{\epsilon} = \epsilon\mathbf{1}_K, \qquad \text{and} \quad \boldsymbol{a}_k = \text{vec}(\boldsymbol{A}_k) \end{aligned} \quad (4)$$

where $\boldsymbol{A}_k \equiv (\boldsymbol{x}_{j_k} - \boldsymbol{x}_{i_k})(\boldsymbol{x}_{j_k} - \boldsymbol{x}_{i_k})^\top - (\boldsymbol{x}_{l_k} - \boldsymbol{x}_{i_k})(\boldsymbol{x}_{l_k} - \boldsymbol{x}_{i_k})^\top$.

Letting $\boldsymbol{Z}_k \equiv \left[\boldsymbol{x}_{j_k}, \boldsymbol{x}_{l_k}\right] - \boldsymbol{x}_{i_k}\mathbf{1}_2^\top$, Algorithm 1 for this setting is re-written in Algorithm 2. Derivation is given in Appendix A.1. The single-variable nonlinear system in Step 10 can be solved by the Newton-Raphson method, and a sufficiently precise solution is obtained with around ten iterations.

In case of $\alpha_k = 0$, Step 4 and Step 12 in Algorithm 2 can be done in time $O(n^2)$ just by copying matrices: $\boldsymbol{Y}_k = \boldsymbol{W}$ and $\boldsymbol{W} = \boldsymbol{Y}_k$, respectively. Even if $\alpha_k > 0$, only $O(n^2)$ computations accomplish the two steps as

$$\begin{aligned} \boldsymbol{Y}_k &= \boldsymbol{W} - \boldsymbol{W}\boldsymbol{Z}_k(\boldsymbol{Q} + \alpha_k^{-1}\boldsymbol{E})^{-1}\boldsymbol{Z}_k^\top\boldsymbol{W}, \\ \boldsymbol{W} &= \boldsymbol{Y}_k - \boldsymbol{Y}_k\boldsymbol{Z}_k(\boldsymbol{P} - \alpha_k^{-1}\boldsymbol{E})^{-1}\boldsymbol{Z}_k^\top\boldsymbol{Y}_k, \end{aligned}$$

which are derived with the well-known matrix inversion lemma, where we have defined $\boldsymbol{Q} \equiv \boldsymbol{Z}_k^\top\boldsymbol{W}\boldsymbol{Z}_k$ and $\boldsymbol{E} \equiv \text{diag}(\{+1, -1\})$.

As a consequence, we are ready to show the main result:

Theorem 1. Algorithm 2 achieves an ε-accurate solution for problem Eq. (3) in computational time $O(n^2 K \log(1/\varepsilon))$. □

5. Experiments

We tested the proposed method, BDRM, on two face databases,

Algorithm 2 BDRM for Metric Learning.

$\boldsymbol{W} = \boldsymbol{W}_0$; $\boldsymbol{\alpha} := \boldsymbol{0}_K$;
for $t = 1, 2, \ldots$ **do**
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{Y}_k := (\boldsymbol{W}^{-1} + \alpha_k \boldsymbol{A}_k)^{-1}$;
    $\boldsymbol{P} := \boldsymbol{Z}_k^\top \boldsymbol{Y}_k \boldsymbol{Z}_k$;
    if $P_{1,1} - P_{2,2} \geq \epsilon$ **then**
      $\alpha_k := 0$;
    else
      Inverse $\boldsymbol{P}$ to get $p^{i,j} = [\boldsymbol{P}^{-1}]_{i,j}$, $\forall i, \forall j$.
      Solve the following system to update α_k:

$$(\epsilon - \alpha_k/C)((p^{1,1} - \alpha_k)(p^{2,2} + \alpha_k) - p^{1,2}p^{2,1}) = p^{2,2} - p^{1,1} + 2\alpha_k \quad \text{and } \alpha_k > 0; \tag{5}$$

    end if
    $\boldsymbol{W} := (\boldsymbol{Y}_k^{-1} + \alpha_k \boldsymbol{A}_k)^{-1}$;
  end for
end for

Table 1 Classification accuracies of different metrics.

	Euclid	Cov	FLD	BDRM	ITML	LMNN
Olivetti	0.902	0.942	0.957	**0.967**	0.952	0.910
AR	0.719	0.893	0.972	**0.979**	0.973	0.959

Olivetti and AR. The database, Olivetti, contains face images for 40 individuals posing a multiclass classification problem with 40 classes. Ten images are given for each person. The database AR contains face images for 100 individuals, and 13 images are included for each person. For both databases, we picked five images at random for performance evaluation using 1-nearest neighbor classifier according to the acquired Mahalanobis distance. The remaining images are used for training. We performed dimension reduction by principal component analysis (PCA) to convert each gray-scale image to a 100-dimensional vector. We use BDRM with $C = 100$ and $\epsilon = 0.01$ to obtain the Mahalanobis matrix $\boldsymbol{W}$, where the values of the two parameters were determined through preliminary experiments. We repeat this procedure three times to report the average accuracies.

Table 1 shows the accuracies of the multiclass classification on the two databases. BDRM is the newly proposed method. We compared BDRM with five existing methods based on the nearest neighbor classifier: 'Euclid', 'FLD', 'Cov', 'ITML', and 'LMNN'. 'Euclid' signifies the method which uses the Euclidean distance. This method is equivalent to the 'Eigenface' method [1], because all input vectors are projections onto the principal subspace. 'FLD' is the so-called Fisherface method [1] that uses the Fisher discriminant analysis to find a discriminative subspace of the principal subspace. 'Cov' takes the average of covariance matrices for all classes, adds a small value to diagonal entries, and sets, to its inverse, a Mahalanobis matrix. 'ITML' [2] and 'LMNN' [7] are existing state-of-the-art metric learning algorithms to obtain discriminative Mahalanobis matrices. In each of six methods, the final prediction is done with the nearest neighbor classifier. The hyper-parameters of those methods are determined in preliminary experiments. As shown in Table 1, BDRM achieves much better classification performances compared to Euclidean distance and covariance-based Mahalanobis distance, and slightly improves the accuracies from existing metric learning algorithms. We further compared our method with SVM. The accuracies of SVMs are summarized in **Table 2**, arguing that our method is superior both to SVM with a linear kernel and to SVM with RBF kernel.

Table 2 Classification accuracies of SVM.

	Linear kernel	RBF kernel
Olivetti	0.927	0.929
AR	0.956	0.946

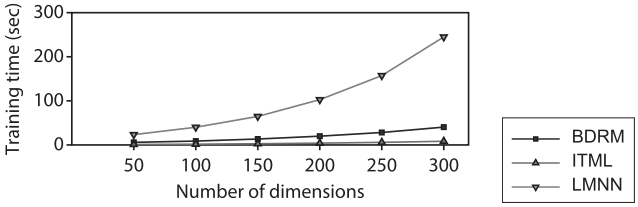


Fig. 2 Computational time for metric learning.

Figure 1 depicting the distributions of projections on the two-dimensional principal subspace somewhat testifies the high classification accuracies of our metric learning algorithm. Kernel PCA with the inner-product defined as $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x}'$, where $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$, produces a principal subspace, and visualizes the distributions of points from each class in the high-dimensional space $\mathbb{R}^n$ induced by the acquired metric $\boldsymbol{W}$. We picked data in five classes of the database AR, and performed Kernel PCA with Euclidean metric and the learned metric $\boldsymbol{W}$ to project the data points onto two-dimensional principal subspaces, respectively. Comparison of two plots in Fig. 1 demonstrates that, whereas the distributions of each class are heavily overlapped in the Euclidean metric-induced principal subspace (Fig. 1 (a)), the overlaps among class distributions are reduced dramatically in the $\boldsymbol{W}$-metric-induced principal subspace (Fig. 1 (b)).

Figure 2 shows the computational time for learning $\boldsymbol{W}$ using the face image database AR, varying the number of dimensions of the subspace obtained by PCA in pre-processing. ITML and our method takes much smaller time for learning, compared to the most popular metric learning method, LMNN. ITML was faster than BDRM because the constraints used for ITML is fewer than those of BDRM; constraints for ITML are formed from doublets, whereas constraints for BDRM are from triplets. It suggested that BDRM bought better classification performances with more constraints.

6. Concluding Remarks

In this paper, we presented a new metric learning algorithm that finds discriminative positive definite Mahalanobis matrices efficiently without eigen-decomposition, and showed promising experimental results on real-world face-image datasets.

ITML [2] introduces a loss function based on LogDet divergence that has seldom been employed in other studies in order to update $\boldsymbol{W}$ in $O(n^2)$ computation. Another important fact utilized by ITML to achieve $O(n^2)$ computational time for an update is the $n \times n$ coefficient matrix in a constraint — which corresponds to $\boldsymbol{A}_k$ in BDRM — derived from a doublet is one-rank. Our study employs triplets $\mathcal{R}_k$ to form constraints that make $\boldsymbol{A}_k$ two-rank, which disables use of the ITML's approach. We tackled this is-

sue by employing Newton method instead of deriving a closed-form update rule, and found out that the computational time is still $O(n^2)$ even by using Newton method. Thus, this study disclosed that the Mahalanobis matrix $\boldsymbol{W}$ can be updated in $O(n^2)$ and achieves an excellent generalization performance, even when using ℓ_2-loss and relative distance penalization, both of which are accepted widely in many machine learning studies (e.g., Ref. [7]).

Acknowledgments TK was supported by Grant-in-Aid for Scientific Research (C) 23500373.

References

[1] Belhumeur, P.N., Hespanha, J.a.P. and Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.19, No.7, pp.711–720 (1997).

[2] Davis, J.V., Kulis, B., Jain, P., Sra, S. and Dhillon, I.S.: Information-theoretic metric learning, *Proc. 24th International Conference on Machine Learning*, *ICML '07*, pp.209–216, New York, NY, USA, ACM (2007).

[3] Globerson, A. and Roweis, S.: Metric Learning by Collapsing Classes, *Advances in Neural Information Processing Systems 18*, Weiss, Y., Schölkopf, B. and Platt, J. (Eds.), pp.451–458, MIT Press, Cambridge, MA (2006).

[4] Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S. and Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM, *Proc. 25th International Conference on Machine Learning*, *ICML '08*, pp.408–415, New York, NY, USA, ACM (2008).

[5] Li, S.Z. and Jain, A.K.: *Handbook of Face Recognition*, Springer (2011).

[6] Torresani, L. and Lee, K.: Large Margin Component Analysis, *Advances in Neural Information Processing Systems 19*, Schölkopf, B., Platt, J. and Hoffman, T. (Eds.), pp.1385–1392, MIT Press, Cambridge, MA (2007).

[7] Weinberger, K.Q. and Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification, *J. Mach. Learn. Res.*, Vol.10, pp.207–244 (2009).

(Communicated by *Koichi Shinoda*)

Appendix

A.1 Derivation of Algorithm 2

Here we derive Algorithm 2 from Algorithm 1. To this end, we first introduce two symmetric matrices $\boldsymbol{U}_k$ and $\boldsymbol{V}$ such that $\mathrm{vec}(\boldsymbol{U}_k) = \boldsymbol{u}_k$ and $\mathrm{vec}(\boldsymbol{V}) = \boldsymbol{v}$, respectively. We then show the following claims by induction:

- $\boldsymbol{Y}_k = \nabla\varphi_*(\boldsymbol{U}_k)$,
- $P_{1,1} - P_{2,2} = \langle \boldsymbol{A}_k, \nabla\varphi_*(\boldsymbol{U}_k)\rangle$,
- Equivalence between Eq. (2) and Eq. (5), and
- $\boldsymbol{W} = \nabla\varphi_*(\boldsymbol{V})$.

We assume the four claims have been maintained until $(t, k-1)$-th iteration. Then, we have

$$\begin{aligned}\boldsymbol{Y}_k &= (\boldsymbol{W}^{-1} + \alpha_k \boldsymbol{A}_k)^{-1} = (-\boldsymbol{V} + \alpha_k \boldsymbol{A}_k)^{-1} = (-\boldsymbol{U}_k)^{-1} \\ &= \nabla\varphi_*(\boldsymbol{U}_k).\end{aligned}$$

Since $\boldsymbol{A}_k = \boldsymbol{Z}_k \boldsymbol{E} \boldsymbol{Z}_k^\top$, we get

$$\begin{aligned}\langle \boldsymbol{A}_k, \nabla\varphi_*(\boldsymbol{U}_k)\rangle &= \left\langle \boldsymbol{Z}_k \boldsymbol{E} \boldsymbol{Z}_k^\top, \boldsymbol{Y}_k \right\rangle = \left\langle \boldsymbol{E}, \boldsymbol{Z}_k^\top \boldsymbol{Y}_k \boldsymbol{Z}_k \right\rangle = \langle \boldsymbol{E}, \boldsymbol{P}\rangle \\ &= P_{1,1} - P_{2,2}.\end{aligned}$$

Equivalence between Eq. (2) and Eq. (5) can be shown by the equality between the lefthand sides of the two equations as

$$\langle \boldsymbol{A}_k, \nabla\varphi_*(\boldsymbol{U}_k - \alpha_k \boldsymbol{A}_k)\rangle = \frac{p^{2,2} - p^{1,1} + 2\alpha_k}{(p^{1,1} - \alpha_k)(p^{2,2} + \alpha_k) - p^{1,2} p^{2,1}}.$$

The last claim is derived as

$$\begin{aligned}\boldsymbol{W} &= (\boldsymbol{Y}_k^{-1} + \alpha_k \boldsymbol{A}_k)^{-1} = (-\boldsymbol{U}_k + \alpha_k \boldsymbol{A}_k)^{-1} = (-\boldsymbol{V})^{-1} \\ &= \nabla\varphi_*(\boldsymbol{V}).\end{aligned}$$

Thus, all the four claims have been shown, establishing the equivalence between Algorithm 1 and Algorithm 2 in the setting of Eq. (4).