On Kernelization of Supervised Mahalanobis Distance Learners

Ratthachat Chatpatanasiri, Teesid Korsrilabutr,
Pasakorn Tangchanachaianan, and Boonserm Kijsirikul

Department of Computer Engineering, Chulalongkorn University,
Pathumwan, Bangkok, 10330, Thailand.
{ratthachat.c,teesid.k,pasakorn.t}@student.chula.ac.th
and boonserm.k@chula.ac.th

Abstract. This paper contains three contributions to the problem of learning a Mahalanobis distance. First, a general framework for kernelizing Mahalanobis distance learners is presented. The framework allows existing algorithms to learn a Mahalanobis distance in a feature space associated with a pre-specified kernel function. The framework is then used for kernelizing three well-known learners, namely, “neighborhood component analysis”, “large margin nearest neighbors” and “discriminant neighborhood embedding”; open problems of recent works are thus solved. Second, while the truths of representer theorems are just assumptions in previous papers related to ours, here representer theorems in the context of kernelized Mahalanobis distance learners are formally proven. Third, unlike previous works which demand cross validation to select a kernel, an inductive kernel alignment method based on quadratic programming is derived in this paper and is used to automatically select an efficient kernel function. Numerical results on various real-world datasets are presented.

Keywords: Mahalanobis distance learning, kernel machines, inductive kernel alignment, representer theorem

1 Introduction

Recently, the problem of learning an efficient Mahalanobis distance for \(k\)-nearest-neighbor (kNN) classification is one of the most active research topics (Goldberger et al., 2005; Chen et al., 2005; Globerson & Roweis, 2006; Weinberger et al., 2006; Sugiyama, 2006; Yang et al., 2006; Yan et al., 2007; Zhang et al., 2007). A Mahalanobis distance can be thought of as the Euclidean distance on a linearly transformed space, and Mahalanobis distance learners can be viewed as “modern” methods for learning a linear map. When the modern learners are compared with classical learners such as principal component analysis and Fisher discriminant analysis, recent experiments with kNN in the transformed spaces show that the modern methods usually outperform the classical methods.

As previously mentioned, the problem of learning a Mahalanobis distance is equivalent to the problem of learning a linear transformation. Within this viewpoint, one important limitation of Mahalanobis distance learners is the fact that
they can learn only a linear transformation. Weinberger et al. (2006); Zhang et al. (2007) consider the problem of learning a non-linear transformation via the kernel trick as an open problem. Chen et al. (2005); Globerson and Roweis (2006); Yan et al. (2007) all propose the same way to kernelize their learners; however, their works do not consider that the kernelization process is, mathematically speaking, possible or not. In other words, the representer theorem, in the context of kernelized Mahalanobis distance learners, has not been proven. In addition, even if one can kernelize a learner, another important problem remains; that is, the problem of selecting an appropriate kernel function.

The key contributions of this paper are three folds. First, we present a general framework for kernelizing Mahalanobis distance learners. By using our framework, we are able to kernelize three recent algorithms, namely, neighborhood component analysis (NCA) (Goldberger et al., 2005), large margin nearest neighbors (LMNN) (Weinberger et al., 2006) and discriminant neighborhood embedding (DNE) (Zhang et al., 2007), and thus we solve open problems listed in their original papers. Second, we prove the representer theorem in the context of Mahalanobis distance learners. The theorem makes both our framework and existing kernelized learners theoretically justified. Third, unlike existing works where cross validation is used to select a kernel function, we derive an inductive kernel alignment algorithm which can be efficiently used for learning a good kernel function from training data.

The rest of the paper is arranged as follows. We provide backgrounds on NCA, LMNN and DNE in Section 2. The kernelization framework and the representer theorem are given in Section 3 where we also apply our results to the three learners mentioned above. In Section 4, we present our inductive kernel alignment algorithm. Numerical results with 9 datasets are presented in Section 5. Finally we summarize our work and our future work in Section 6.

2 Background

Let \( \{x_i, y_i\}_{i=1}^n \) denote a training set of \( n \) labeled examples with inputs \( x_i \in \mathbb{R}^D \) and corresponding class labels \( y_i \in \{c_1, \ldots, c_p\} \). Any Mahalanobis distance can be represented by a symmetric positive semi-definite (PSD) matrix \( M \in \mathbb{S}_+^D \). Here, we denote \( \mathbb{S}_+^D \) as a space of \( D \times D \) PSD matrices. Given two points \( x_i \) and \( x_j \), and a PSD matrix \( M \), the Mahalanobis distance with respect to \( M \) between the two points is defined as \( \|x_i - x_j\|_M = \sqrt{(x_i - x_j)^T M (x_i - x_j)} \). Our goal is to find a PSD matrix \( M^* \) that minimizes a reasonable objective function \( f(\cdot) \):

\[
M^* = \arg\min_{M \in \mathbb{S}_+^D} f(M).
\]

Since the matrix \( M \) can be decomposed to \( A^T A \), we can equivalently restate our problem as learning the best matrix \( A \) which minimizes a pre-defined objective function:

\[
A^* = \arg\min_{A \in \mathbb{R}^{d \times D}} f(A).
\]
Note that $d = D$ in the standard setting, but we can learn a low-rank projection by restricting $d < D$. After learning the best linear transformation $A^*$, it will be used by kNN to compute the distance between two points in the transformed space as $(x_i - x_j)^T M^* (x_i - x_j) = \| A^* x_i - A^* x_j \|^2$, where $\| \cdot \|$ is the standard Euclidean norm in the new space. Hence, the original problem of learning a Mahalanobis distance for kNN now turns to the problem of learning a linear transformation $A$ of the input space such that in the transformed space, kNN performs well.

### 2.1 Neighborhood Component Analysis (NCA) Algorithm

The original goal of NCA is to optimize the leave-one-out (LOO) performance on training data. However, as the actual LOO classification error of kNN is a non-smooth function of the matrix $A$, Goldberger et al. (2005) propose to optimize a stochastic variant of the LOO kNN score which is defined as follows:

$$-f^{NCA}(A) = \sum_i \sum_{j \in c_i} p_{ij},$$

where

$$p_{ij} = \frac{\exp(-\|A x_i - A x_j\|^2)}{\sum_{k \neq i} \exp(-\|A x_i - A x_k\|^2)}, \quad p_{ii} = 0.$$

In eq. (2), there is the minus sign in front of $f^{NCA}(\cdot)$ because we want to maximize this function instead of minimizing it (see eq. (1)). This objective function of NCA can be interpreted as the expected number of points correctly classified under “soft”-kNN.

Optimizing $f^{NCA}(\cdot)$ is quite straightforward as the authors propose to use a gradient based method such as delta-bar-delta or conjugate gradients. Following Goldberger et al. (2005), the formula of $\partial f^{NCA}/\partial A$ can be obtained as follows:

$$\frac{\partial f^{NCA}}{\partial A} = 2A \sum_i \left( p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in c_i} p_{ij} x_{ij} x_{ij}^T \right).$$

One major disadvantage of NCA, however, is that $f^{NCA}(\cdot)$ is not convex, and a gradient based method is thus prone to local optima.

### 2.2 Large Margin Nearest Neighbor (LMNN) Algorithm

In LMNN (Weinberger et al., 2006), the output Mahalanobis distance is optimized with the goal that for each point, its k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin.

For each point $x_i$, we define its k target neighbors as the k other inputs with the same label $y_i$ that are closest to $x_i$ (with respect to the Euclidean distance in the input space). We use $w_{ij} \in \{0, 1\}$ to indicate whether an input $x_j$ is a
Minimize \( \sum_{i,j} w_{ij} (x_i - x_j)^T M (x_i - x_i) + c \sum_{i,j,l} w_{ij} (1 - y_{il}) \xi_{ijl} \)

Subject to:
(1) \( (x_i - x_i)^T M (x_i - x_i) - (x_i - x_j)^T M (x_i - x_j) \geq 1 - \xi_{ijl} \).
(2) \( \xi_{ijl} \geq 0. \)
(3) \( M \in \mathbb{S}_+^D. \)

Fig. 1. SDP formulation for the LMNN algorithm.

The objective function of LMNN is as follows:

\[
  f_{\text{LMNN}}(M) = \sum_{i,j} w_{ij} ||x_i - x_j||_M^2 + c \sum_{i,j,l} w_{ij} (1 - y_{il}) \left[ 1 + ||x_i - x_j||_M^2 - ||x_i - x_l||_M^2 \right]_+, 
\]

where \([\cdot]_+\) denotes the standard hinge loss: \([z]_+ = \max(z, 0)\). The term \( c > 0 \) is some positive constant typically set by cross validation. The objective function above has two competing terms. The first term penalizes large distances between each input and its target neighbors, while the second term penalizes small distances between each input and all other inputs that do not share the same label.

The objective function above can be reformulated as an instance of semidefinite programming (SDP) (Boyd & Vandenberghe, 2004) as shown in Figure 1. Since SDP is an instance of convex programming, in contrast to \( f^{NCA}() \), the global optimum of \( f_{\text{LMNN}}() \) can be efficiently computed. Although Weinberger et al. (2006) do not mention how to get a low-rank transformation \( A \), there is a standard way to achieve this goal. First, \( M \) can be decomposed to \( Q \Lambda Q^T \) where \( \Lambda \) is a diagonal matrix with only non-negative real-valued entries. Finally, we may use \( A = [Q \Lambda^{1/2}]_d \) where \([\cdot]_d\) denotes the truncated matrix which is abandoning the first \( D - d \) columns (assuming the eigenvalues are sorted such that \( \lambda_1 \leq ... \leq \lambda_D \)).

### 2.3 Discriminant Neighborhood Embedding (DNE) Algorithm

The main idea of DNE (Zhang et al., 2007) is quite similar to LMNN. DNE seeks a linear transformation such that neighborhood points in the same class are squeezed but those in different classes are separated as much as possible. However, DNE does not care about the notion of margin; in the case of LMNN, we want every point to stay far from points of other classes, but for DNE, we want the average distance between two neighborhood points of different classes to be large. Another difference is that LMNN can learn a full Mahalanobis distance, i.e. a general weighted linear projection, while DNE can learn only an unweighted linear projection.

Similar to LMNN, we define two sets of \( k \) target neighbors for each point \( x_i \) based on the Euclidean distance in the input space. For each \( x_i \), let \( \text{Neig}^I(i) \) be the set of \( k \) nearest neighbors having the same label \( y_i \), and let \( \text{Neig}^E(i) \) be
the set of \( k \) nearest neighbors having different labels from \( y_i \). We define \( w_{ij} \) as follows

\[
w_{ij} = \begin{cases} 
+1, & \text{if } j \in \text{Neig}^I(i) \lor i \in \text{Neig}^I(j), \\
-1, & \text{if } j \in \text{Neig}^E(i) \lor i \in \text{Neig}^E(j), \\
0, & \text{otherwise}. 
\end{cases}
\]

The objective function of DNE is:

\[
f_{\text{DNE}}(A) = \sum_{i,j} w_{ij} \|Ax_i - Ax_j\|^2. 
\]

With little algebra, we can reformulate the objective function (up to a constant factor) to be

\[
f_{\text{DNE}}(A) = \text{trace}(AX(D - W)X^T A^T), 
\]

where \( W \) is a symmetric matrix with elements \( w_{ij} \), \( D \) is a diagonal matrix with \( D_{ii} = \sum_j w_{ij} \) and \( X \) is the matrix of input points \( (x_1, ..., x_n) \). It is easy to see that \( D - W \) is symmetric but is not necessarily PSD.

Recall that for the purpose of finding low-dimensional projection, \( A \in \mathbb{R}^{d \times D} (d \leq D) \). From now on, we will work with \( A^T \in \mathbb{R}^{D \times d} \) instead of \( A \) for notational convenience. Let \( A^T = (a_1, ..., a_d) \) where \( a_i \in \mathbb{R}^D \). By adding further constraints that,

\[
a_i^T a_i = 1 \quad \text{and} \quad a_i^T a_j = 0 \quad (i \neq j), \tag{3}
\]

the optimal vectors \( a_1, ..., a_d \) can be found by solving the following eigenvalue problem (Fukunaga, 1990):

\[
X(D - W)X^T a_i = \lambda_i a_i. 
\]

By rearranging the eigenvalues such that \( \lambda_1 \leq ... \leq \lambda_D \), the optimal matrix \( (a_1, ..., a_d) \) can be constructed from the unit eigenvectors corresponding to the first \( d \) eigenvalues.

Note that, as LMNN, the global optimum of \( f_{\text{DNE}}(\cdot) \) with constraints specified by eq. (3) can be found efficiently, but instead of using SDP, we now just use a simpler eigen-decomposition algorithm. One advantage of DNE over LMNN and NCA is that, for DNE, we have a deterministic rule to select the optimal dimensionality \( d \) of the transformed space: \( d \) will be the number of negative eigenvalues obtained from the above eigenvalue problem (Zhang et al., 2007).

### 3 Kernelization

In this section, we show a general framework for kernelizing Mahalanobis distance learners such as NCA, LMNN, and DNE. In fact, due to the representer theorem shown in Subsection 3.3 other learners, e.g. ones proposed by Xing et al. (2003) and by Yang et al. (2006), can also be kernelized by using our framework as well. Due to lack of spaces, we omit the details here.
3.1 Framework

Suppose we have a non-linear map from the input space to the feature space: \( x \mapsto \phi(x) \). A squared Mahalanobis distance under a matrix \( M \) in the feature space is

\[
(\phi(x_i) - \phi(x_j))^T M (\phi(x_i) - \phi(x_j)) = (\phi(x_i) - \phi(x_j))^T A^T A (\phi(x_i) - \phi(x_j)).
\] (4)

As in Subsection 2.3, it is more convenient to work with \( A^T = (a_1, \ldots, a_d) \) instead of \( A \). Denote a (possibly infinite-dimensional) matrix \( \Phi = (\phi(x_1), \ldots, \phi(x_n)) \). By the representer theorem (see below), it is the case that each \( a_i \) can be represented by

\[
a_i = \Phi u_i = \sum_j u_{ij} \phi(x_j),
\]

for some \( u_i = (u_{i1}, \ldots, u_{in})^T \in \mathbb{R}^n \). Hence, \( A^T \) is represented by

\[
A^T = \Phi U^T,
\] (5)

where \( U^T = (u_1, \ldots, u_d) \). Insert the result from eq. (5) to eq. (4), we have

\[
(\phi(x_i) - \phi(x_j))^T M (\phi(x_i) - \phi(x_j)) = (k_i - k_j)^T U^T U (k_i - k_j),
\]

where

\[
k_i = \Phi^T \phi(x_i) = (\langle \phi(x_1), \phi(x_i) \rangle, \ldots, \langle \phi(x_n), \phi(x_i) \rangle)^T.
\] (6)

Now our formula depends only on an inner-product \( \langle \phi(x_i), \phi(x_j) \rangle \), and thus the kernel trick can be now applied by using the fact that \( k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \) for a PSD kernel function \( k(\cdot, \cdot) \) (Schölkopf & Smola, 2001). Therefore, the problem of finding the best Mahalanobis distance in the feature space is now reduced to finding the best linear transformation \( U \) of size \( d \times n \). The process of learning the matrix \( U \) is as easy as the original process of learning the best matrix \( A \) in the input space as demonstrated by three examples below.

Once we find the matrix \( U \), the Mahalanobis distance from a new test point \( x' \) to any input point \( x_i \) in the feature space can be calculated as follows:

\[
||\phi(x') - \phi(x_i)||_M^2 = (k' - k_i)^T U^T U (k' - k_i),
\] (7)

where \( k' = (k(x', x_1), \ldots, k(x', x_n))^T \). Consequently, we can easily perform kNN classification in the feature space under the obtained Mahalanobis distance.

**Example 1: Kernel NCA (KNCA)** The KNCA algorithm is essentially the same as the original algorithm. The objective function is the following:

\[
-f^{KNCA}(U) = \sum_i \sum_{j \in c_i} p_{ij},
\]
Minimize $\sum_{i,j} w_{ij} (k_i - k_j)^T S (k_i - k_j) + c \sum_{i,j,l} w_{ij} (1 - y_{il}) \xi_{ijl}$
Subject to:
1. $(k_i - k_l)^T S (k_i - k_l) - (k_i - k_j)^T S (k_i - k_j) \geq 1 - \xi_{ijl}$.
2. $\xi_{ijl} \geq 0$.
3. $S \in S^n_+$.

Fig. 2. SDP formulation for the KLMNN algorithm.

where
$$p_{ij} = \frac{\exp(-\|Uk_i - Uk_j\|^2)}{\sum_{k \neq i} \exp(-\|Uk_i - Uk_k\|^2)}, \quad p_{ii} = 0,$$

and $k_i$ are defined as in eq. (6). The gradient of $f^{KCA}(U)$ is the following:
$$\frac{\partial f^{KCA}}{\partial U} = 2U \sum_i \left( p_i \sum_k p_{ik} k_{ik} k_{ik}^T - \sum_{j \in c_i} p_{ij} k_{ij} k_{ij}^T \right).$$

In fact, all we need to do is just to transform
$$(x_i \in \mathbb{R}^D) \mapsto (k_i \in \mathbb{R}^n),$$

and run the original algorithm with the new input $k_i$ (but remember to convert the test point as in eq. (7)).

Example 2: Kernel LMNN (KLMNN) Almost the same as KNCA, KLMNN is very similar to the original LMNN. To obtain $U$, we first solve the SDP problem for the PSD matrix $S$ as shown in Figure 2. Then we decompose $S = U^T U$ by using the eigen-decomposition method.

One subtle point here is how to construct $w_{ij}$. Now, we have several choices for completing this process. In fact, the specification of $w_{ij}$ should be based on prior knowledge (Weinberger et al., 2006), in the absence of prior knowledge, the Euclidean distance on the input space is usually the practical choice. However, the Euclidean distance on the feature space
$$\|\phi(x_i) - \phi(x_j)\| = \sqrt{k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)}$$
is reasonable as well.

Remark (Practical Implementation). In the experiments shown in Section 5, we modified the LMNN implementation of Weinberger et al. (2006) for our KLMNN. We found that applying kernel PCA (KPCA) (Schölkopf & Smola, 2001) before KLMNN (and also before KNCA) provides several advantages. First, the stability of KLMNN algorithm is noticeably improved (without KPCA, KLMNN often terminated but does not converge). Second, as recommended by Weinberger et al. (2006), we can speed-up the learning process by retaining only, says, the 200 largest-variance principal components. Third, let
\{\psi_i\}_{i=1}^n$ be the set of principal components in the feature space corresponding to a choice of kernel function and let $\hat{k}_j = (\langle \phi(x_j), \psi_1 \rangle, \ldots, \langle \phi(x_j), \psi_n \rangle)^T$. We have

$$||\hat{k}_i - \hat{k}_j||^2 = \sum_{k=1}^n \langle \phi(x_i) - \phi(x_j), \psi_k \rangle^2 = ||\phi(x_i) - \phi(x_j)||^2$$

by using Parseval identity. Hence, based on $||\hat{k}_i - \hat{k}_j||$, $w_{ij}$ can be constructed, and, similar to KNCA (see eq. (8)), KLMNN can be easily implemented.

**Example 3: Kernel DNE (KDNE)** As in KLMNN, we have several choices to specify $w_{ij}$ for KDNE (which we will not repeat here). After specifying $w_{ij}$, the objective function of KDNE is

$$f_{KDNE}(U) = \text{trace}(UK(D - W)KU^T),$$

where we use the same algebra as presented in Subsection 3.1 and $K = (k_1, \ldots, k_n) = K^T$ (since a PSD kernel function is always symmetric). Using the fact that $a_i = \Phi u_i$, the constraints are now reformulated as

$$u_i^T Ku_i = 1 \text{ and } u_i^T Ku_j = 0 \ (i \neq j).$$

Hence, we need to solve the following generalized eigenvalue problem to obtain $u_i$:

$$K(D - W)Ku_i = \lambda_i Ku_i.$$ 

Similar to DNE, we solve the generalized eigenvalue problem above for the first $d$ generalized eigenvectors where the optimal choice of $d$ corresponds to the number of negative generalized eigenvalues.

### 3.2 Do we really need a kernel?

On page 8 of the LMNN paper, Weinberger et al. gave a comment about KLMNN: ‘as LMNN already yields highly nonlinear decision boundaries in the original input space, however, it is not obvious that “kernelizing” the algorithm will lead to significant further improvement’. Here, we note that “kernelizing” the algorithm certainly leads to significant improvements even in simple problems shown in Figure 3.

The main intuition behind the kernelization of “Mahalanobis distance learners for kNN classifier” lies in the fact that non-linear boundaries produced by kNN (with or without Mahalanobis distance) is usually helpful for problems with multi-modalities; however, the non-linear boundaries of kNN is sometimes not useful when data of the same class stay on a low-dimensional non-linear manifold as shown in Figure 3.
Fig. 3. Two synthetic examples where NCA, LMNN and DNE cannot learn any efficient Mahalanobis distances for kNN. Note that in each example, data in each class lie on a simple non-linear 1-dimensional subspace (which cannot be discovered by the three learners, however). In contrast, the kernelized versions of the three algorithms (using the 2nd-order polynomial kernel) can learn very efficient distances, i.e., the non-linear subspaces are discovered by the kernelized algorithms.

3.3 Representer Theorem

Intuitively, the representer theorem says that, in a feature space, the optimal subspace (with respect to a certain objective function) must be representable by a subspace of the span of the training data. This fact justifies all formulations in our framework. In fact, besides KNCA, KLMNN and KDNE, existing kernelized learners usually assume that the representer theorem is true without formally proving the theorem (Chen et al., 2005; Globerson & Roweis, 2006; Yan et al., 2007). Hence, our result here justifies these learners as well.

To start proving our result, the following lemma is useful.

Lemma 1 Let $\mathcal{X}, \mathcal{Y}$ be two Hilbert spaces and $\mathcal{Y}$ is separable, i.e. $\mathcal{Y}$ has a countable orthonormal basis $\{e_i\}_{i \in \mathbb{N}}$. Any bounded linear map $A : \mathcal{X} \rightarrow \mathcal{Y}$ can be uniquely decomposed as $\sum_{i=1}^{\infty} \langle \cdot, \varphi_i \rangle_{\mathcal{X}} e_i$ for some $\{\varphi_i\}_{i \in \mathbb{N}} \subseteq \mathcal{X}$.

Proof. Since $A$ is bounded, the linear functional $\phi \mapsto \langle A\phi, e_i \rangle_{\mathcal{Y}}$ is bounded for every $i$ since

$$|\langle A\phi, e_i \rangle_{\mathcal{Y}}| \leq ||A\phi|| ||e_i|| \leq ||A|| ||\phi||.$$

By Riesz representation theorem, the map $\langle A, e_i \rangle_{\mathcal{Y}}$ can be written as $\langle \cdot, \varphi_i \rangle_{\mathcal{X}}$ for a unique $\varphi_i \in \mathcal{X}$. Since $\{e_i\}_{i \in \mathbb{N}}$ is an orthonormal basis of $\mathcal{Y}$, for every $\phi \in \mathcal{X}$,

$$A\phi = \sum_{i=1}^{\infty} \langle A\phi, e_i \rangle_{\mathcal{Y}} e_i = \sum_{i=1}^{\infty} \langle \phi, \varphi_i \rangle_{\mathcal{X}} e_i.$$

Remember that a PSD kernel function $k(\cdot, \cdot)$ over the set $\mathbb{R}^D$ is always associated with some “feature map” $\phi(\cdot)$ from $\mathbb{R}^D$ to a Hilbert space $\mathcal{X}$ such that
$$\langle \phi(x), \phi(x') \rangle = k(x, x')$$ (Schölkopf & Smola, 2001). To avoid complicated notations, in this section we write $\phi_i$ to denote $\phi(x_i)$, and we omit subscripts such as $\mathcal{X}, \mathcal{Y}$ of inner products.

**Theorem 1 (Weak Representer Theorem)** For arbitrary objective function $f$, the optimization,

$$
\min_A f \left( \langle A\phi_1, A\phi_1 \rangle, \ldots, \langle A\phi_i, A\phi_j \rangle, \ldots, \langle A\phi_n, A\phi_n \rangle \right)
$$

s.t. $A : \mathcal{X} \rightarrow \mathbb{R}^d$ is a bounded linear map,

has the same optimal value as,

$$
\min_{U \in \mathbb{R}^{d \times n}} f (k_1^T U \phi_1, \ldots, k_i^T U \phi_j, \ldots, k_n^T U \phi_n).
$$

**Proof.** Let $\{e_i\}_{i=1}^d$ be the canonical basis of $\mathbb{R}^d$, and let $\phi' \in \text{span}\{\phi_1, \ldots, \phi_n\}$. By Lemma 1,

$$
A\phi' = \sum_{i=1}^d \langle \phi', \phi_i \rangle e_i,
$$

for some $\varphi_1, \ldots, \varphi_d \in \mathcal{X}$. Each $\varphi_i$ can be decomposed as $\varphi_i = \varphi_i' + \varphi_i^\perp$ such that $\varphi_i'$ lies in $\text{span}\{\phi_1, \ldots, \phi_n\}$ and $\varphi_i^\perp$ is orthogonal to the span. These facts make $\langle \phi', \varphi_i \rangle = \langle \phi', \varphi_i' \rangle$ for every $i$. We then have, for some $u_{ij} \in \mathbb{R}$, $1 \leq i \leq d$, $1 \leq j \leq n,$

$$
A\phi' = \sum_{i=1}^d \sum_{j=1}^n u_{ij} \langle \phi', \phi_j \rangle e_i
$$

$$
= \sum_{i=1}^d e_i \sum_{j=1}^n u_{ij} \langle \phi', \phi_j \rangle
$$

$$
= \begin{bmatrix}
    u_{11} & \cdots & u_{1n} \\
    \vdots & \ddots & \vdots \\
    u_{d1} & \cdots & u_{dn}
\end{bmatrix}
\begin{bmatrix}
    \langle \phi', \phi_1 \rangle \\
    \vdots \\
    \langle \phi', \phi_n \rangle
\end{bmatrix}
= Uk'.
$$

Since every $\phi_i$ is in the span, we conclude that $A\phi_i = Uk_i$. Now, one can easily check that $\langle A\phi_i, A\phi_j \rangle = k_i^T U^T U k_j$. Hence, whenever a map $A$ is given, we can construct $U$ such that it results in the same objective function value. By reversing the proof, it is easy to see that the converse is also true, and thus the theorem is proven.

**Remark.** Since $A\phi' = Uk' = U\Phi^T \phi'$, one immediate corollary which is already used in eq. (5) is that $A^T$ can be represented by $\Phi U^T$.

The above theorem is called a *weak* representer theorem since it does not say that an optimal subspace, i.e. the span of the obtained $\varphi_i$s, must lie in the span of the training data; it says that at least one optimal subspace must lie in
the span. Nevertheless, by adding a regularizer into an objective function, we can strengthen the above result so that all optimal subspaces must lie in the span of the training data. In the following theorem, we use the fact that \( A \) is representable by \( \{ \phi_i \} \) as shown in Lemma 1.

**Theorem 2** (Strong Representer Theorem) For arbitrary objective function \( f \) and monotonically increasing functions \( g_i \), let

\[
h(\varphi_1, ..., \varphi_d, \phi_1, ..., \phi_n) = f\left( (\phi_1, \varphi_1), ..., (\phi_i, \varphi_j), ..., (\phi_n, \varphi_d) \right) + \sum_{i=1}^{d} g_i(||\varphi_i||).
\]

Any optimal set of linear functionals

\[
\arg\min_{\{\varphi_i\}} h(\varphi_1, ..., \varphi_d, \phi_1, ..., \phi_n)
\]

s.t. \( \forall \ i \ \varphi_i : X \to \mathbb{R} \) is a bounded linear functional

must admit the representation of

\[
\varphi_i = \sum_{j=1}^{n} u_{ij} \phi_j \quad (i = 1, ..., d).
\]

**Proof.** We prove by contrapositive. Consider a set of linear functionals \( \{ \varphi_i \} \) such that it is not in the span of \( \phi_j \)s, i.e. \( \varphi_i = \varphi'_i + \varphi^\perp_i \) where \( \varphi'_i = \sum_{j=1}^{n} u_{ij} \phi_j \) and \( \langle \varphi^\perp_i, \phi_j \rangle = 0 \) for all \( j = 1, ..., n \) and \( \exists \ i \ \varphi_i^\perp \neq 0 \). Then \( \langle \varphi_i, \phi_j \rangle = \langle \varphi'_i, \phi_j \rangle \). Thus

\[
f\left( (\phi_1, \varphi_1), ..., (\phi_i, \varphi_j), ..., (\phi_n, \varphi_d) \right) = f\left( (\phi_1, \varphi'_1), ..., (\phi'_i, \varphi'_j), ..., (\phi_n, \varphi'_d) \right).
\]

However,

\[
\sum_{i=1}^{d} g_i(||\varphi_i||) = \sum_{i=1}^{d} g_i(\sqrt{||\varphi'_i||^2 + ||\varphi^\perp_i||^2}) > \sum_{i=1}^{d} g_i(||\varphi'_i||),
\]

using the fact that \( \exists \ i \ \varphi^\perp_i \neq 0 \). Hence,

\[
h(\varphi_1, ..., \varphi_d, \phi_1, ..., \phi_n) > h(\varphi'_1, ..., \varphi'_d, \phi_1, ..., \phi_n),
\]

and thus \( \{ \varphi_i \} \) is not an optimal solution. The proof is complete.

**Remark.** To apply the above theorem to our framework, we can simply view \( A = (\varphi_1, ..., \varphi_d)^T \). If each \( g_i \) is the square function, then it is easy to see that the regularizer becomes \( \sum_{i=1}^{d} ||\varphi_i||^2 = ||A||_{HS} \) where \( || \cdot ||_{HS} \) is Hilbert-Schmidt (HS) norm of an operator. If each \( \varphi_i \) is finite-dimensional, HS norm is reduced to Frobenius norm. Here, we allow HS norm of a bounded linear operator to take a value of \( \infty \). The result above states that any optimal \( \{ \varphi_i \} \) must be represented by \( \{ \Phi u_i \} \). Therefore, using the same notation as Subsection 3.1, we have

\[
\sum_{i=1}^{d} ||\varphi_i||^2 = \sum_{i=1}^{d} u_i^T \Phi^T \Phi u_i = \sum_{i=1}^{d} u_i^T K u_i = \text{trace}(UK^T).
\]
Hence, adding trace($UKU^T$) into existing objective functions creates a new class of learners, namely, regularized Mahalanobis distance learners such as regularized KNCA (RKNCA), regularized KLMNN (RKLMNN) and regularized KDNE (RKDNE) which we plan to investigate them in the near future.

4 Inductive Kernel Alignment

The problem of selecting an efficient kernel function is central to all kernel machines. All existing Mahalanobis distance learners use exhaustive methods such as cross validation to select a kernel function. In this section, we introduce an inductive kernel alignment method which can be used to efficiently learn a good kernel function. To our knowledge, in literatures of supervised Mahalanobis distance learners, this is the first attempt to investigate the possibility of learning a kernel function from data.

To use the kernel alignment method in classification problems, the following assumption is central: for each couple of example $x_i, x_j$, the ideal kernel $k(x_i, x_j)$ is $Y_{ij}$ where $Y_{ij}$ is defined as (Guermeur et al., 2004)

$$Y_{ij} = \begin{cases} +1, & \text{if } y_i = y_j, \\ \frac{1}{p-1}, & \text{otherwise}, \end{cases}$$

where $p$ is the number of classes in the training data. Denoting $Y$ as the matrix having elements of $Y_{ij}$, we then define the alignment between the kernel matrix $K$ and the ideal kernel matrix $Y$ as follows:

$$\text{align}(K, Y) = \frac{\langle K, Y \rangle_F}{||K||_F ||Y||_F},$$

where $\langle \cdot, \cdot \rangle_F$ denotes Frobenius inner-product such that $\langle K, Y \rangle_F = \text{trace}(K^T Y)$ and $|| \cdot ||_F$ is Frobenius norm induced by Frobenius inner-product.

The method presented in this section is inspired from the work of Kandola et al. (2002), but our formulation and our inductive algorithm are new and simpler than one proposed in Kandola et al.’s work. Other kernel alignment algorithms such as one proposed by Lanckriet et al. (2004) can be used only in transductive setting, and thus cannot be directly employed to learn a kernel function from training data. Assume that we have $m$ kernel functions, $k_1(\cdot, \cdot), \ldots, k_m(\cdot, \cdot)$ and $K_1, \ldots, K_m$ are their corresponding Gram matrices with respect to the training data. Here, the kernel function obtained from the alignment method is parameterized in the form of $k(\cdot, \cdot) = \sum \alpha_i k_i(\cdot, \cdot)$ where $\alpha_i \geq 0$. Note that the obtained kernel function is guaranteed to be positive semidefinite. In order to learn the best coefficients $\alpha_1, \ldots, \alpha_m$, we solve the following optimization problem:

$$\{\alpha_1, \ldots, \alpha_m\} = \arg \max_{\alpha_i \geq 0} \text{align}(K, Y),$$

where $K = \sum \alpha_i K_i$. Note that as $K$ and $Y$ are PSD, $\langle K, Y \rangle_F \geq 0$. Since both the numerator and denominator terms in the alignment equation can be
On Kernelization of Supervised Mahalanobis Distance Learners

arbitrary large, we can simply fix the numerator to 1. We then reformulate the problem as follows:

\[
\begin{align*}
\text{arg max } \alpha_i & \geq 0, \langle K, Y \rangle_F = 1 \\
& = \text{arg min } \alpha_i \geq 0, \langle K \rangle_F^2 \\
& = \text{arg min } \alpha_i \geq 0, \langle K \rangle_F^2 = \sum_{i,j} \alpha_i \alpha_j \langle K_i, K_j \rangle_F.
\end{align*}
\]

Defining a PSD matrix \( S \) whose elements \( S_{ij} = \langle K_i, K_j \rangle_F \), a vector \( b = (\langle K_1, Y \rangle_F, \ldots, \langle K_m, Y \rangle_F)^T \) and a vector \( \alpha = (\alpha_1, \ldots, \alpha_m)^T \), we then reformulate eq. (9) as follows:

\[
\alpha = \text{arg min } \alpha_i \geq 0, \alpha^T b = 1, \alpha^T S \alpha. \tag{10}
\]

This optimization problem can be efficiently solved using quadratic programming (Boyd & Vandenberghe, 2004) so that we can learn the best kernel function \( k(\cdot, \cdot) = \sum_i \alpha_i k_i(\cdot, \cdot) \) efficiently.

Since the magnitudes of the optimal \( \alpha_i \) are varied due to \( \|K_i\|_F \), it is convenient to use \( k'_i(\cdot, \cdot) = k_i(\cdot, \cdot)/\|K_i\|_F \) and hence \( K'_i = K_i/\|K_i\|_F \) in the derivation of eq. (10). We define \( S' \) and \( b' \) similar to \( S \) and \( b \) except that they are based on \( K'_i \) instead of \( K_i \). Let

\[
\gamma = \text{arg min } \gamma_i \geq 0, \gamma^T b' = 1, \gamma^T S' \gamma. \tag{11}
\]

It is easy to see that the final kernel function \( k(\cdot, \cdot) = \sum_i \gamma_i k'_i(\cdot, \cdot) \) achieved from eq. (11) is not changed from the kernel achieved from eq. (10).

5 Numerical Results

In this section, we conduct experiments of NCA, LMNN, DNE and their kernelized versions on 9 real-world datasets. In all experiment, we set \( k=1 \) for the kNN classification algorithm. In order to use the inductive kernel alignment method presented in Section 4, we need to specify base kernel functions. In the experiments, we use scaled RBF-kernel functions (Schölkopf & Smola, 2001, p. 216),

\[
k(x, y) = \exp(-\frac{|x-y|^2}{2D\sigma^2})
\]

where \( D \) is the dimensionality of the input data, as base kernel functions specified by the following different 21 values of \( \sigma \): 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1, 2.5, 5, 7.5, 10, 25, 50, 75, 100, 250, 500, 750, 1000.

As we remarked in Subsections 3.1, there are several reasonable choices to specify \( w_{ij} \) for KLMNN and KDNE. Therefore, in all experiments, we take our convenience as the first priority. As a result, we base the specification of \( w_{ij} \) by \( \|\phi(x_i) - \phi(x_j)\| \) in the case of KLMNN and by \( \|x_i - x_j\| \) in the case of KDNE.
Table 1. Average Accuracy with Standard Deviation: NCA versus KNCA.

| Name         | NCA        | KNCA       |
|--------------|------------|------------|
| Balance      | 0.89 ± 0.03| 0.92 ± 0.01|
| Breast Cancer| 0.95 ± 0.01| 0.96 ± 0.01|
| Glass        | 0.61 ± 0.05| 0.66 ± 0.04|
| Ionosphere   | 0.83 ± 0.04| 0.92 ± 0.02|
| Iris         | 0.96 ± 0.03| 0.95 ± 0.03|
| Musk2        | 0.87 ± 0.02| 0.88 ± 0.02|
| Pima         | 0.68 ± 0.02| 0.67 ± 0.03|
| Satellite    | 0.82 ± 0.02| 0.84 ± 0.01|
| Yeast        | 0.47 ± 0.02| 0.49 ± 0.02|

Table 2. Average Accuracy with Standard Deviation: LMNN versus KLMNN.

| Name         | LMNN       | KLMNN      |
|--------------|------------|------------|
| Balance      | 0.84 ± 0.04| 0.88 ± 0.02|
| Breast Cancer| 0.95 ± 0.01| 0.97 ± 0.00|
| Glass        | 0.63 ± 0.05| 0.66 ± 0.04|
| Ionosphere   | 0.88 ± 0.02| 0.94 ± 0.02|
| Iris         | 0.95 ± 0.02| 0.95 ± 0.02|
| Musk2        | 0.80 ± 0.03| 0.88 ± 0.02|
| Pima         | 0.68 ± 0.02| 0.72 ± 0.02|
| Satellite    | 0.81 ± 0.01| 0.84 ± 0.01|
| Yeast        | 0.47 ± 0.02| 0.54 ± 0.02|

The 9 real-world datasets obtained from the UCI repository are Balance, Breast Cancer, Glass, Ionosphere, Iris, Musk2, Pima, Satellite and Yeast. The generalization performance of each algorithm is measured by the average test accuracy over 40 realizations of randomly splitting each dataset into training and testing subsets. For each dataset, the number of training data is 200 except for Glass and Iris where we use 100 examples because these two datasets contain only 214 and 150 total examples, respectively. The results are shown in Tables 1, 2 and 3. It is clear from the results that the kernelized algorithms applying the inductive kernel alignment method often improve the performance of their original algorithms.

6 Summary and Future Work

We have presented a general framework to kernelize Mahalanobis distance learners. Although we have focused only on the supervised settings, the framework is clearly applicable to learners in other settings as well, e.g. a semi-supervised learner presented by Xing et al. (2003). We proved two versions of representer theorems which justify both our framework and those in previous works. The theorems proven in the paper can also be applied to Mahalanobis distance learners in unsupervised and semi-supervised settings. We derived an inductive kernel
Table 3. Average Accuracy with Standard Deviation: DNE versus KDNE.

| Name       | DNE     | KDNE    |
|------------|---------|---------|
| Balance    | 0.79 ± 0.02 | 0.83 ± 0.02 |
| Breast Cancer | 0.96 ± 0.01 | 0.95 ± 0.02 |
| Glass      | 0.65 ± 0.04 | 0.63 ± 0.04 |
| Ionosphere | 0.87 ± 0.02 | 0.95 ± 0.02 |
| Iris       | 0.95 ± 0.02 | 0.96 ± 0.02 |
| Musk2      | 0.89 ± 0.02 | 0.87 ± 0.02 |
| Pima       | 0.67 ± 0.02 | 0.70 ± 0.03 |
| Satellite  | 0.84 ± 0.01 | 0.85 ± 0.01 |
| Yeast      | 0.40 ± 0.05 | 0.47 ± 0.04 |

alignment algorithm which can be efficiently used for learning a good kernel function from training data. Although we have concentrated only on Mahalanobis distance learners, our kernel alignment algorithm can be indeed applied to all kernel classifiers. Numerical results over various real-world datasets showed consistent improvements of kernelized learners over their original versions.

In the future, we plan to extend the current work to other settings such as semi-supervised, transductive and online settings. We also plan to investigate in details various types of regularizers appeared in the regularized versions of kernelized Mahalanobis distance learners proposed in Subsection 3.3.
Bibliography

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

Chen, H.-T., Chang, H.-W., & Liu, T.-L. (2005). Local discriminant embedding and its variants. CVPR.

Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition. Academic Press.

Globerson, A., & Roweis, S. (2006). Metric learning by collapsing classes. NIPS, 18, 451–458.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2005). Neighbourhood components analysis. NIPS, 513–520.

Guermeur, Y., Lifchitz, A., & Vert, R. (2004). A kernel for protein secondary structure prediction. Kernel Methods in Computational Biology.

Kandola, J., Taylor, S. J., & Cristianini, N. (2002). Optimizing kernel alignment over combinations of kernels. Technical Report 2002-121.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. JMLR, 5, 27–72.

Sugiyama, M. (2006). Local fisher discriminant analysis for supervised dimensionality reduction. ICML.

Weinberger, K., Blitzer, J., & Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. NIPS, 18, 1473–1480.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning with application to clustering with side-information. NIPS, 505–512.

Yang, L., Jin, R., Sukthankar, R., & Liu, Y. (2006). An efficient algorithm for local distance metric learning. AAAI.

Zhang, W., Xue, X., Sun, Z., Guo, Y.-F., & Lu, H. (2007). Optimal dimensionality of metric space for classification. ICML (pp. 1135–1142).