Neurocomputing 73 (2010) 1570–1579

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A new kernelization framework for Mahalanobis distance learning algorithms

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

Department of Computer Engineering, Chulalongkorn University, Pathumwan, Bangkok 10330, Thailand

ARTICLE INFO

Available online 12 March 2010

Keywords:
Distance metric learning
Dimensionality reduction
Representer theorem
Kernel machines
Kernel alignment

ABSTRACT

This paper focuses on developing a new framework of kernelizing Mahalanobis distance learners. The new KPCA trick framework offers several practical advantages over the classical kernel trick framework, e.g. no mathematical formulas and no reprogramming are required for a kernel implementation, a way to speed up an algorithm is provided with no extra work, the framework avoids troublesome problems such as singularity. Rigorous representer theorems in countably infinite dimensional spaces are given to validate our framework. Furthermore, unlike previous works which always apply brute force methods to select a kernel, we derive a kernel alignment formula based on quadratic programming which can efficiently construct an appropriate kernel for a given dataset.

1. Introduction

Recently, the problem of learning an efficient Mahalanobis distance for *k-nearest neighbor* (kNN) classification has been one of the most active research topics [7,10,9,25–29,31,17,34–36]. Since learning a Mahalanobis distance is equivalent to learning a linear map, the inability to learn a non-linear transformation is one important limitation of the learners. As the research in Mahalanobis distance learning has just recently begun, several issues are left open such as (1) some efficient learners do not have non-linear extensions, (2) the *kernel trick* [24], a standard non-linearization method, is not fully automatic in the sense that new mathematical formulas have to be derived and new programming codes have to be implemented; this is not convenient to non-experts, and (3) the problem of how to select an efficient kernel function has been left untouched in previous works; in previous works, the best kernel function is achieved via a brute-force method such as cross validation. In this paper, we highlight the following key contributions:

- A *KPCA trick* framework is presented as an alternative choice to the kernel-trick framework. In contrast to the kernel trick, the KPCA trick does not require users to derive new mathematical formulas. Also, whenever an implementation of an original learner is available, users are not required to re-implement the kernel version of the original learner. Moreover, the new framework avoids problems such as singularity in eigen-decomposition and provides a convenient way to speed up a learner. Rigorous representer theorems in countably infinite dimensional spaces are given to validate our framework. Three recent learners which are difficult to kernelize are given as examples.
- The problem of efficient kernel selection is dealt with. We investigate the *kernel alignment* method proposed in previous works [14,37] to see whether it is appropriate for a kernelized Mahalanobis distance learner. A formula based on quadratic programming (QP) is derived to automatically construct an appropriate kernel function. Kernel construction based on the QP formula requires much shorter running time when compared to the standard cross validation approach.
- As kNN is already a non-linear classifier, there are some doubts about the usefulness of kernelizing Mahalanobis distance learners [27, p. 8]. We provide an explanation and conduct extensive experiments on real-world datasets to prove the usefulness of kernelization.

2. Background

Let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ denote a training set of n labeled examples with inputs $\mathbf{x}_i \in \mathbb{R}^D$ 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 $\mathbf{x}_i$ and $\mathbf{x}_j$, and a PSD matrix M, the Mahalanobis distance with respect to M between the two points is defined as $\|\mathbf{x}_i - \mathbf{x}_j\|_M = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j)}$. Our

* Corresponding author. Tel.: +66 86 536 9491.
E-mail addresses: ratthachat.c@gmail.com (R. Chatpatanasiri), g48tkr@cp.eng.chula.ac.th (T. Korsrilabutr), pasakorn.t@student.chula.ac.th (P. Tangchanachaianan), boonserm.k@chula.ac.th (B. Kijsirikul).

0925-2312/$ - see front matter Crown Copyright © 2010 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.11.037

goal is to find a PSD matrix M^* that minimizes a reasonable objective function $f(M)$:

$$M^* = \arg\min_{M \in \mathbb{S}_+^D} f(M). \quad (1)$$

Since the PSD matrix M can be decomposed to A^TA, we can equivalently restate our problem as learning the best matrix A minimizing an objective function $f(A)$[1]:

$$A^* = \arg\min_{A \in \mathbb{R}^{d \times D}} f(A). \quad (2)$$

Note that $d=D$ in the full-rank setting, but for the purpose of dimensionality reduction we can learn a low-rank projection by restricting $d < D$. After learning the best linear map A^*, it will be used by kNN to compute the distance between two points in the transformed space as $(\mathbf{x}_i-\mathbf{x}_j)^T M^*(\mathbf{x}_i-\mathbf{x}_j) = \|A^*\mathbf{x}_i - A^*\mathbf{x}_j\|^2$.

In the following subsections, three recent algorithms are presented. Despite their efficiency and popularity, the three algorithms do not have kernel versions, and thus we are primarily interested in kernelizing these three algorithms in order to improve their classification performances.

2.1. Neighborhood component analysis

The original goal of neighborhood component analysis (NCA) [10] 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. propose to minimize a stochastic variant of the LOO kNN score which is defined as follows:

$$f^{NCA}(A) = -\sum_i \sum_{y_j = c_i} p_{ij}, \quad (3)$$

where

$$p_{ij} = \frac{\exp(-\|A\mathbf{x}_i - A\mathbf{x}_j\|^2)}{\sum_{k \neq i} \exp(-\|A\mathbf{x}_i - A\mathbf{x}_k\|^2)}, \quad p_{ii} = 0.$$

Optimizing $f^{NCA}(\cdot)$ can be done by applying a gradient based method. One major disadvantage of NCA, however, is that $f^{NCA}(\cdot)$ is not convex, and the gradient based methods are thus prone to local optima.

2.2. Large margin nearest neighbor

In large margin nearest neighbor (LMNN) [27], 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 $\mathbf{x}_i$, we define its *k target neighbors* as the k other inputs with the same label y_i that are closest to $\mathbf{x}_i$ (with respect to the Euclidean distance in the input space). We use $w_{ij} \in \{0,1\}$ to indicate whether an input $\mathbf{x}_j$ is a target neighbor of an input $\mathbf{x}_i$. For convenience, we define $y_{ij} \in \{0,1\}$ to indicate whether or not the labels y_i and y_j match. The objective function of LMNN is as follows:

$$f^{LMNN}(M) = \sum_{i,j} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_M^2 + c\sum_{i,j,l}(w_{ij}(1-y_{il})[1 + \|\mathbf{x}_i - \mathbf{x}_j\|_M^2 - \|\mathbf{x}_i - \mathbf{x}_l\|_M^2]_+),$$

where $[\cdot]_+$ denotes the standard hinge loss: $[z]_+ = \max(z,0)$. The objective function above is convex[2] and 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.

2.3. Discriminant neighborhood embedding

Discriminant neighborhood embedding (DNE) [36] seeks a linear transformation such that, on average, neighborhood points in the same class are squeezed but those in different classes are separated as much as possible. Similar to LMNN, we define *two* sets of k target neighbors for each point $\mathbf{x}_i$. For each $\mathbf{x}_i$, let $Neig^I(i)$ be the set of k nearest neighbors having the same label y_i, and let $Neig^E(i)$ be the set of k nearest neighbors having different labels from y_i. We define w_{ij} as follows:

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

The objective function of DNE is

$$f^{DNE}(A) = \sum_{i,j} w_{ij}\|A\mathbf{x}_i - A\mathbf{x}_j\|^2,$$

which can be reformulated to be

$$f^{DNE}(A) = \mathrm{trace}(AX(D-W)X^TA^T),$$

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 $(\mathbf{x}_1,\ldots,\mathbf{x}_n)$. To solve the problem by eigen-decomposition, the constraint $AA^T = I$ is added so that we have the following optimization problem:

$$A^* = \arg\min_{AA^T = I} \mathrm{trace}(AX(D-W)X^TA^T). \quad (4)$$

2.4. Motivation of a new kernel framework

To understand the advantages of the KPCA trick presented in the next section over the kernel trick, it is best to derive a kernel trick formula for each algorithm and see difficulties of implementing a kernel trick. Note that their original papers do not show how to kernelize these algorithms. We denote KNCA, KLMNN and KDNE as the kernel versions of NCA, LMNN and DNE, respectively.

Let $k(\cdot,\cdot)$ be a PSD kernel function associated with a non-linear function $\phi(\cdot) : \mathbb{R}^D \mapsto \mathcal{H}$ such that $k(\mathbf{x},\mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ [24] where $\mathcal{H}$ is a Hilbert space. Denote ϕ_i for $\phi(\mathbf{x}_i)$ for $i = 1,\ldots,n$ and ϕ' for $\phi(\mathbf{x}')$. We define $\Phi = (\phi_1,\ldots,\phi_n)$. Before deriving kernel-trick formulas, we note that the essence of the kernel trick is to parameterize $A = U\Phi^T$ where U is a finite-dimensional matrix, and thus the quantity $A\phi_i$ can be computed as $A\phi_i = U\Phi^T\phi_i = U\mathbf{k}_i$ where $\mathbf{k}_i = (k(\mathbf{x}_1,\mathbf{x}_i),\ldots, k(\mathbf{x}_n,\mathbf{x}_i))$ [6].

2.4.1. KNCA

As noted in Section 2.1, in order to minimize the objective of NCA and KNCA, we need to derive gradient formulas, and the formula of $\partial f^{KNCA}/\partial A$ is [10]

$$-2A\sum_i \left(p_i \sum_k p_{ik}\phi_{ik}\phi_{ik}^T - \sum_{j \in c_i} p_{ij}\phi_{ij}\phi_{ij}^T \right), \quad (5)$$

where for brevity we denote $\phi_{ij} = \phi_i - \phi_j$. Nevertheless, since ϕ_i may lie in an infinite dimensional space, the above formula cannot be always implemented in practice. In order to implement the kernel-trick version of KNCA, users need to prove the following proposition which is not stated in the original work [10] (the proof is given in the Appendix B):

[1] To be precise, the objective function is $f(A^TA)$ but we just write $f(A)$ to simplify our notations in the paper.

[2] There is a variation on LMNN called "large margin component analysis" (LMCA) [26] which proposes to optimize A instead of M; however, LMCA does not preserve some desirable properties, such as convexity, of LMNN, and therefore the algorithm "kernel LMCA" presented there is different from "kernel LMNN" presented in this paper.

Proposition 1. *$\partial f^{KNCA}/\partial A$ can be formulated as $V\Phi^T$ where V depends on $\{\phi_i\}$ only in the form of $\langle \phi_i,\phi_j \rangle = k(\mathbf{x}_i,\mathbf{x}_j)$, and thus we can compute all elements of V.*

Therefore, at the i th iteration of an optimization step of a gradient optimizer, we needs to update the current best linear map as follows:

$$A^{(i)} = A^{(i-1)} + \varepsilon \frac{\partial f^{KNCA}}{\partial A} = (U^{(i-1)} + \varepsilon V^{(i-1)})\Phi^T = U^{(i)}\Phi^T, \tag{6}$$

where ε is a step size. The kernel-trick formulas of KNCA are thus finally achieved. However, we emphasize that the process of proving Proposition 1 and Eq. (6) is not trivial and may be tedious and difficult for non-experts and practitioners who focus their tasks on applications rather than theories. Moreover, since the formula of $\partial f^{KNCA}/\partial A$ is significantly different from $\partial f^{NCA}/\partial A$, users are required to re-implement KNCA (even they already possess an NCA implementation) which is again not at all convenient. In contrast, we note that all these difficulties disappear if the KPCA trick algorithm is applied instead of the kernel trick.

2.4.2. KLMNN

Similar to KNCA, the online-available code of LMNN[3] employs a gradient based optimization, and thus new gradient formulas in the feature space has to be derived and new implementation has to be done in order to apply the kernel trick. On the other hand, by applying the KPCA trick, the original LMNN code can be immediately used.

There are other advantages of the KPCA trick involving speeding up the running time and specifying w_{ij}, which will be explained later in Section 3.2 (Remarks 4 and 5).

2.4.3. KDNE

By applying $A = U\Phi^T$ and defining the Gram matrix $K = \Phi^T\Phi$, we have the following proposition.

Proposition 2. *The kernel-trick formula of KDNE is the following minimization problem*:

$$U^* = \underset{UKU^T = I}{\arg\min} \operatorname{trace}(UK(D-W)KU^T). \tag{7}$$

Note that this kernel-trick formula of KDNE involves a *generalized eigenvalue problem* instead of a plain eigenvalue problem involved in DNE. As a consequence, we face a *singularity* problem, i.e. if K is not full-rank, the constraint $UKU^T = I$ cannot be satisfied. Using elementary linear algebra, it can be shown that K is not full-rank if and only if $\{\phi_i\}$ is not linearly independent, and this condition is not highly improbable. Sugiyama [25], Yu and Yang [32], Yang et al. [30] suggest methods to cope with the singularity problem in the context of Fisher discriminant analysis which may be applicable to KDNE. Sugiyama [25] suggests to use the constraint $U(K+\varepsilon I)U^T = I$ instead of the original constraint; however, an appropriate value of ε has to be tuned by cross validation which is time-consuming. Alternatively, [32,30] propose more complicated methods of directly minimizing an objective function in the null space of the constraint matrix so that the singularity problem is explicitly avoided.

We note that a KPCA-trick implementation of KDNE does not have this singularity problem as only a plain eigenvalue problem has to be solved. Moreover, as in KLMNN, applying the KPCA trick instead of the kernel trick to KDNE avoid the tedious task of modifying the original code to appropriately specify w_{ij} in the feature space.

[3] http://www.weinbergerweb.net/Downloads/LMNN.html

3. The KPCA-trick framework

In this section, we develop a *kernel principal component analysis trick* or shortly *KPCA trick* framework which can be conveniently applied to kernelize the three learners described in the previous section. The main advantage of the KPCA-trick framework over the kernel-trick framework is that the new framework have no difficulties on kernelizing learners demonstrated in Section 2.4.

Historically, this name was first appeared in the paper of [4] who first applied this method to invariant support vector machines. Recently, Li et al. [16] invent similar trick in the context of data clustering. About the same time as our work, Zhang et al. [33] also propose the KPCA-trick in the context of finite-dimensionality reduction. Nevertheless, the applications of the KPCA-trick framework presented in Section 2.4 goes beyond what were shown in the previous works since these works constrain their method to the finite-dimensional cases. In contrast, learning a full Mahalanobis distance sometimes involves an infinite dimensional space. Thus, the new validation proof of the KPCA trick is needed in the context of Mahalanobis distance learning (see Theorem 1). Note that, parallel to our work, Jain et al. [12] propose another kernelization framework which is complimentary to ours.

In this section, we define $\{\phi_i\}$ as in Section 2.4. The central idea of the KPCA trick is to represent each ϕ_i and ϕ' in a new "finite"-dimensional space, without any loss of information. Within the framework, a new coordinate of each example is computed "explicitly", and each example in the new coordinate is then used as the input of any existing Mahalanobis distance learner.

To simplify the discussion, we assume that $\{\phi_i\}$ is linearly independent and has its center at the origin, i.e. $\sum_i \phi_i = 0$. Since we have n total examples, the span of $\{\phi_i\}$ has dimensionality n by our assumption. Here we claim that each example ϕ_i can be represented as $\varphi_i \in \mathbb{R}^n$ with respect to a new *orthonormal* basis $\{\psi_i\}_{i=1}^n$ such that $span(\{\psi_i\}_{i=1}^n)$ is the same as $span(\{\phi_i\}_{i=1}^n)$ without loss of any information (we will show that the claim is valid by stating representer theorems in the next section). More precisely, we define

$$\varphi_i = (\langle \phi_i,\psi_1 \rangle, \ldots, \langle \phi_i,\psi_n \rangle) = \Psi^T\phi_i, \tag{8}$$

where $\Psi = (\psi_1,\ldots,\psi_n)$. Note that although we may be unable to numerically represent each ψ_i, an inner-product of $\langle \phi_i,\psi_j \rangle$ can be conveniently computed by KPCA (and thus the name of the trick). Likewise, a new test point ϕ' can be mapped to $\varphi' = \Psi^T\phi'$. Consequently, the mapped data $\{\varphi_i\}$ and φ' are finite-dimensional and can be explicitly computed.

The KPCA-trick algorithm consisting of three simple steps is shown in Fig. 1. In the algorithm, we denote a Mahalanobis distance learner by maha performing the optimization process shown in Eq. (1) (or Eq. (2)) and outputs the best Mahalanobis distance M^* (or the best linear map A^*).

Input: 1. training examples: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$,
2. new example: $\mathbf{x}'$,
3. kernel function: $k(\cdot,\cdot)$
4. Mahalanobis distance learner: maha

Algorithm:
(1) Apply kpca$(k, \{\mathbf{x}_i\}, \mathbf{x}')$ such that $\{\mathbf{x}_i\} \mapsto \{\varphi_i\}$ and $\mathbf{x}' \mapsto \varphi'$.
(2) Apply maha with new inputs $\{(\varphi_i, y_i)\}$ to achieve M^* or A^*.
(3) Perform kNN based on the distance $||\varphi_i - \varphi'||_{M^*}$ or $||A^*\varphi_i - A^*\varphi'||$.

Fig. 1. The KPCA-trick algorithm.

Besides NCA, LMNN and DNE, most Mahalanobis distance learners we know to date can be kernelized by this simple algorithm. For examples, Zhang et al. [34,35] recently proposed a general manifold learning framework called *patch alignment* containing many existing Mahalanobis distance learners (each of which is in fact a linear version of a manifold learner), including new learners such as *local coordinates alignment* and *discriminative locality alignment*. Although these manifold learners are able to learn non-linear subspaces without using the kernel-based frameworks, they do not provide a coordinate in a manifold for a new test data point (the so-called *out-of-sample* problem). Therefore, the KPCA trick can be applied to the patch alignment framework to provide a non-linear manifold subspace together with an out-of-sample mapping. The application of the KPCA trick to the patch alignment framework has advantages similar to those of DNE stated in Section 2.4. Other learners whose kernel versions are previously undeveloped [31,28,17] can easily apply the KPCA trick to get their non-linear versions as well.

3.1. Representer theorems

Is it valid to represent an infinite-dimensional vector ϕ by a finite-dimensional vector φ? In the context of SVMs [4], this validity of the KPCA trick is easily achieved by straightforwardly extending a proof of a classical representer theorem [23,13]. In the context of Mahalanobis distance learning, however, we cannot directly extend a proof provided in previous works. Note that, in the SVM case, what is learned is a hyperplane, a linear functional outputting a one-dimensional value. In our case, what is learned is a linear map which, in general, outputs a *countably infinite dimensional* vector since, according to Mercer theorem [24], a feature space corresponding to, e.g. a gaussian kernel can be of countably infinite dimensionality. Hence, to prove the validity of the KPCA trick in our case, we need some mathematical tools which can handle a countably infinite dimensionality. Below we state our versions of representer theorems which prove the validity of the KPCA trick in the current context. Note that we write a function f with inputs $x_1,\ldots,x_n$ as $f(\{x_i\}_{i=1}^n)$.

By our representer theorems, it is the fact that, given an objective function $f(\cdot)$ (see Eq. (1)), the optimal value of $f(\cdot)$ based on the input $\{\phi_i\}$ is equal to the optimal value of $f(\cdot)$ based on the input $\{\varphi_i\}$. Hence, the representation of φ_i can be safely applied. We separate the problem of Mahalanobis distance learning into two different cases. The first theorem covers Mahalanobis distance learners (learning a full-rank linear transformation) while the second theorem covers dimensionality reduction algorithms (learning a low-rank linear transformation).

Theorem 1 (*Full-rank representer theorem*). *Let $\{\tilde{\psi}_i\}_{i=1}^n$ be a set of points in a feature space $\mathcal{X}$ such that $span(\{\tilde{\psi}_i\}_{i=1}^n) = span(\{\phi_i\}_{i=1}^n)$, and $\mathcal{X}$ and $\mathcal{Y}$ be separable Hilbert spaces. For arbitrary objective function f, the optimization*

$$\min_A f : (\{\langle A\phi_i, A\phi_j\rangle\}_{i,j=1}^n) \; s.t. A : \mathcal{X}\to\mathcal{Y} \quad \text{is a bounded linear map},$$

has the same optimal value as

$$\min_{A' \in \mathbb{R}^{n\times n}} f(\{\tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j\}_{i,j=1}^n),$$

where $\tilde{\varphi}_i = (\langle \phi_i, \tilde{\psi}_1\rangle, \ldots, \langle \phi_i, \tilde{\psi}_n\rangle)^T \in \mathbb{R}^n$.

Theorem 2 (*Low-rank representer theorem*). *Define $\{\tilde{\psi}_i\}_{i=1}^n$ and $\tilde{\varphi}_i$ as in Theorem 1. For arbitrary objective function f, the optimization*

$$\min_A f : (\{\langle A\phi_i, A\phi_j\rangle\}_{i,j=1}^n) \; s.t. A : \mathcal{X}\to\mathbb{R}^d \text{ is a bounded linear map},$$

has the same optimal value as

$$\min_{A' \in \mathbb{R}^{d\times n}} f(\{\tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j\}_{i,j=1}^n).$$

We note that Theorems 1 and 2 are more general than what we discuss above. They justify both the kernel trick (see [6]) and the KPCA trick (by substituting $\tilde{\psi}_i = \psi_i$ and hence $\tilde{\varphi}_i = \varphi_i$). Their proofs are given in Appendix A.

3.2. Remarks

1. Note that by Mercer theorem [24, p. 37], we can either think of each $\phi_i \in \ell_2$ or $\phi_i \in \mathbb{R}^N$ for some positive integer N, and thus the assumption of Theorem 1 that $\mathcal{X}$, as well as $\mathcal{Y}$, is separable Hilbert space is then valid. Also, both theorems require that the objective function of a learning algorithm must depend only on $\{\langle A\phi_i, A\phi_j\rangle\}_{i,j=1}^n$ or equivalently $\{\langle \phi_i, M\phi_j\rangle\}_{i,j=1}^n$. This condition is, actually, not a strict condition since learners in literatures have their objective functions in this form [7,10,9,25–27,29,31,20–22,17,34–36].
2. The two theorems stated in this section do not require $\{\tilde{\psi}_i\}$ to be an orthonormal set $\{\psi_i\}$. Thus, the representer theorems validate both the kernel and KPCA tricks. Restricting $\tilde{\psi}_i = \psi_i$ as in Eq. (8) results in the KPCA trick framework having many advantages as discussed earlier.
3. The statement of classical representer theorems deals with the representation of an optimal hyperplane [23]. In the same sense, our theorems also imply the representation of an optimal linear map A^* as shown in Appendix A.
4. Running times of learners such as NCA[4] and LMNN, which employ gradient-based implementation, strongly depend on the dimensionality of the input data. As recommended by [27], it can be helpful to first apply a dimensionality reduction algorithm such as PCA before performing a learning process: the learning process can be speed up by retaining only, say, the 100 largest-variance principal components of the input data. In the KPCA trick framework, dimensionality reduction can be performed without extra work as KPCA is already applied at the first place.
5. There is another advantage of the KPCA trick: LMNN and DNE require a specification of w_{ij} which is based on the quantity $\|\mathbf{x}_i - \mathbf{x}_j\|$. Thus, w_{ij} should be based on $\|\phi_i - \phi_j\| = \sqrt{k(\mathbf{x}_i,\mathbf{x}_i) + k(\mathbf{x}_j,\mathbf{x}_j) - 2k(\mathbf{x}_i,\mathbf{x}_j)}$ in the feature space of KLMNN, and, with the kernel trick, users have to modify the original code in order to appropriately specify w_{ij}. In contrast, by applying the KPCA trick which restricts $\{\psi_i\}$ to be an orthonormal set as in Eq. (8), we have the following proposition:

   Proposition 3. *Let $\{\psi_i\}_{i=1}^n$ be an orthonormal set such that $span(\{\psi_i\}_{i=1}^n) = span(\{\phi_i\}_{i=1}^n)$ and $\varphi_i = (\langle \phi_i, \psi_1\rangle, \ldots, \langle \phi_i, \psi_n\rangle)^T \in \mathbb{R}^n$, then $\|\varphi_i - \varphi_j\|^2 = \|\phi_i - \phi_j\|^2$ for each $1 \le i,j \le n$.*

   Proof. Since we work on a separable Hilbert space $\mathcal{X}$, we can extend the orthonormal set $\{\psi_i\}_{i=1}^n$ to $\{\psi_i\}_{i=1}^\infty$ such that $\overline{span(\{\psi_i\}_{i=1}^\infty)}$ is $\mathcal{X}$ and $\langle \phi_i, \psi_j\rangle = 0$ for each $i=1,\ldots,n$ and $j > n$. Then, by an application of the Parseval identity [15],

$$\|\phi_i - \phi_j\|^2 = \sum_{k=1}^\infty \langle \phi_i - \phi_j, \psi_k\rangle^2 = \sum_{k=1}^n \langle \phi_i - \phi_j, \psi_k\rangle^2 = \|\varphi_i - \varphi_j\|^2.$$

   The last equality comes from Eq. (8). □

[4] We slightly modify and apply the NCA code of Charless Fowlkes: http://www.cs.berkeley.edu/~fowlkes/software/nca/.

Therefore, with the KPCA trick, the target neighbors w_{ij} of each point is computed based on $\|\varphi_i-\varphi_j\|=\|\phi_i-\phi_j\|$ without any modification of the original code.

6. The stronger version of Theorem 2 which strictly limits the representation of A^* can be obtained by inserting a regularizer into the objective function of a (kernelized) Mahalanobis distance learner as stated in Theorem 3.

Theorem 3 (*Strong representer theorem*). *Define* $\{\psi_i\}_{i=1}^n$ *as in Eq.* (8). *For arbitrary objective function f. Consider the optimization problem*

$$\min_A \quad f(\{\langle A\phi_i,A\phi_j\rangle\}_{i,j=1}^n)+\|A\|_{HS}^2 s.t. \quad A:\mathcal{H}\to\mathbb{R}^d \quad \text{is a bounded linear map,}$$

where $\|\cdot\|_{HS}$ *denote the Hilbert–Schmidt norm. Then, an optimal solution* A^* *must admit the representation of* $A^*=U^*\Psi^T$ *for some* $U^*\in\mathbb{R}^{d\times n}$.

From the theorem, using the fact that $\Psi^T\Psi=I$, it can be shown [6] that $\|A\|_{HS}^2=\|U\|_F^2$ where $\|\cdot\|_F$ denote the Frobenius norm. By adding the regularizer, $\|U\|_F^2$, into existing objective functions, we have a new class of learners, namely, *regularized Mahalanobis distance learners* such as regularized KNCA (RKNCA), regularized KLMNN (RKLMNN) and regularized KDNE (RKDNE). We plan to investigate effects of using various types of regularizers in the near future.

4. Kernel alignment

The problem of selecting an efficient kernel function is central to all kernel machines. All previous works on Mahalanobis distance learners use exhaustive methods such as cross validation to select a kernel function. In this section, we investigate a possibility to automatically construct a kernel which is appropriate for a Mahalanobis distance learner. More specifically, we consider a method called *kernel alignment* [14,37] which is able to learn, from a training set, a kernel in the form of $k(\cdot,\cdot)=\sum_i\alpha_i k_i(\cdot,\cdot)$ where $k_1(\cdot,\cdot),\ldots,k_m(\cdot,\cdot)$ are pre-chosen base kernels.

Here, our kernel alignment formulation belongs to the class of quadratic programs (QPs) which can be solved more efficiently than the formulations proposed by [14,37] which belong to the class of semidefinite programs (SDPs) and quadratically constrained quadratic programs (QCQPs), respectively [3].

To use kernel alignment in classification problems, the following assumption is central: for each couple of examples $\mathbf{x}_i$, $\mathbf{x}_j$, the ideal kernel $k(\mathbf{x}_i,\mathbf{x}_j)$ is Y_{ij} [11] where

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

and 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:

$$\text{align}(K,Y)=\frac{\langle K,Y\rangle_F}{\|K\|_F\|Y\|_F}, \tag{9}$$

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

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. In this paper, the kernel function obtained from the alignment method is parameterized in the form of $k(\cdot,\cdot)=\sum_i\alpha_i k_i(\cdot,\cdot)$ 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:

$$\{\alpha_1,\ldots,\alpha_m\}=\arg\max_{\alpha_i\geq 0}\ \text{align}(K,Y), \tag{10}$$

where $K=\sum_i\alpha_i K_i$. 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 arbitrary large, we can simply fix the numerator to 1. We then reformulate the problem as

$$\arg\min_{\alpha_i\geq 0,\langle K,Y\rangle_F=1}\|K\|_F\|Y\|_F=\arg\min_{\alpha_i\geq 0,\langle K,Y\rangle_F=1}\|K\|_F^2$$
$$=\arg\min_{\alpha_i\geq 0,\sum_i\alpha_i\langle K_i,Y\rangle_F=1}\sum_{ij}\alpha_i\alpha_j\langle K_i,K_j\rangle_F.$$

Defining a PSD matrix S whose elements $S_{ij}=\langle K_i,K_j\rangle_F$, a vector $\mathbf{b}=(\langle K_1,Y\rangle_F,\ldots,\langle K_m,Y\rangle_F)^T$ and a vector $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_m)^T$, we then reformulate Eq. (10) as

$$\boldsymbol{\alpha}=\arg\min_{\alpha_i\geq 0,\boldsymbol{\alpha}^T\mathbf{b}=1}\ \boldsymbol{\alpha}^TS\boldsymbol{\alpha}. \tag{11}$$

This optimization problem is a QP and can be efficiently solved [3]; hence, we are able to learn the best kernel function $k(\cdot,\cdot)=\sum_i\alpha_i k_i(\cdot,\cdot)$ efficiently.

Since the magnitudes of the optimal α_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. (11). We define S' and $\mathbf{b}'$ similar to S and $\mathbf{b}$ except that they are based on K_i' instead of K_i. Let

$$\boldsymbol{\gamma}=\arg\min_{\gamma_i\geq 0,\boldsymbol{\gamma}^T\mathbf{b}'=1}\ \boldsymbol{\gamma}^TS'\boldsymbol{\gamma}. \tag{12}$$

It can be shown that the final kernel function $k(\cdot,\cdot)=\sum_i\gamma_i k_i'(\cdot,\cdot)$ achieved from Eq. (12) is not changed from the kernel achieved from Eq. (11).

5. Numerical experiments

In this section, we conduct extensive experiments to illustrate the performance of kernelized algorithms, especially for those who apply the kernel alignment method. Next, we give more insights of results obtained from the two kernelization frameworks. More specifically, we show that there are situations where *in practice* the two frameworks result in different Mahalanobis distance (while *in theory* they do not).

5.1. Accuracy performances

On page 8 of the LMNN paper [27], 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, before giving experimental results, we explain why "kernelizing" the algorithm can lead to significant improvements. The main intuition behind the kernelization of "Mahalanobis distance learners for the kNN classification algorithm" lies in the fact that non-linear boundaries produced by kNN (with or without Mahalanobis distance) are usually helpful for problems with multi-modalities, but they are sometimes not helpful when data of the same class stay on a low-dimensional non-linear manifold as shown in Fig. 2. Therefore, we answer the above KLMNN comment that 'KLMNN (and other kernelized Mahalanobis distance learners) does significantly improve the classification performance in some situations'.

To see the improvement in real-world applications, we conduct experiments on NCA, LMNN, DNE and their kernel versions on 10 real-world datasets to show that (1) it is really the case that the kernelized algorithms usually outperform their

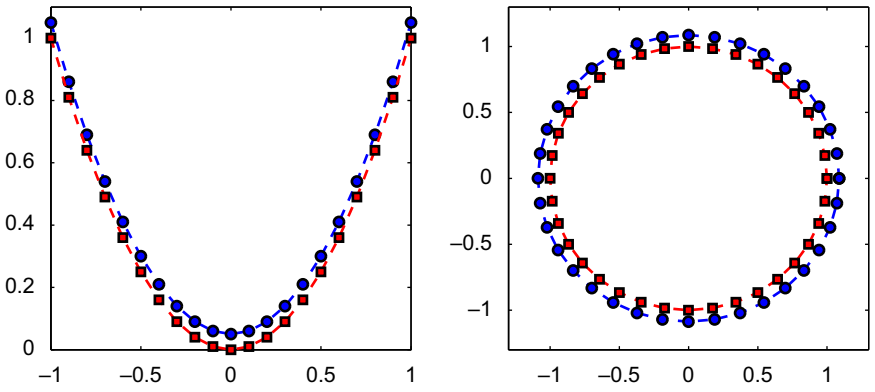

Fig. 2. Two synthetic examples where NCA, LMNN and DNE cannot learn any efficient Mahalanobis distances for kNN. Note that in each example, data in each class lie on a simple non-linear one-dimensional subspace (which, however, cannot be discovered by the three learners). In contrast, the kernel versions 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.

Table 1
The average accuracy with standard deviation of NCA and their kernel versions.

Name	NCA	KNCA	AKNCA
BALANCE	0.89 ± 0.03	**0.92 ± 0.01**	**0.92 ± 0.01**
BREAST	0.95 ± 0.01	**0.97 ± 0.01**	**0.96 ± 0.01**
GLASS	0.61 ± 0.05	**0.69 ± 0.02**	**0.66 ± 0.04**
IONOS	0.83 ± 0.04	**0.94 ± 0.03**	**0.92 ± 0.02**
IRIS	0.96 ± 0.03	0.96 ± 0.01	0.95 ± 0.03
MUSK2	0.87 ± 0.02	**0.90 ± 0.01**	**0.88 ± 0.02**
PIMA	0.68 ± 0.02	**0.71 ± 0.02**	0.67 ± 0.03
SATELLITE	0.82 ± 0.02	**0.84 ± 0.01**	**0.84 ± 0.01**
YEAST	0.47 ± 0.02	**0.50 ± 0.01**	**0.49 ± 0.02**
M-USPS	0.88 ± 0.02	**0.91 ± 0.02**	**0.91 ± 0.02**
W/D/L	–	9/1/0	8/0/2

On the bottom row, the win/draw/lose statistics of each kernelized algorithm comparing to its original version is drawn.

Table 2
The average accuracy with standard deviation of LMNN and their kernel versions.

Name	LMNN	KLMNN	AKLMNN
BALANCE	0.84 ± 0.04	**0.87 ± 0.01**	**0.88 ± 0.02**
BREAST	0.95 ± 0.01	**0.97 ± 0.01**	**0.97 ± 0.00**
GLASS	0.63 ± 0.05	**0.69 ± 0.04**	**0.66 ± 0.04**
IONOS	0.88 ± 0.02	**0.95 ± 0.02**	**0.94 ± 0.02**
IRIS	0.95 ± 0.02	**0.96 ± 0.02**	0.95 ± 0.02
MUSK2	0.80 ± 0.03	**0.93 ± 0.01**	**0.88 ± 0.02**
PIMA	0.68 ± 0.02	**0.71 ± 0.02**	**0.72 ± 0.02**
SATELLITE	0.81 ± 0.01	**0.85 ± 0.01**	**0.84 ± 0.01**
YEAST	0.47 ± 0.02	**0.48 ± 0.02**	**0.54 ± 0.02**
M-USPS	0.93 ± 0.01	**0.94 ± 0.01**	**0.95 ± 0.01**
W/D/L	–	10/0/0	9/1/0

Table 3
The average accuracy with standard deviation of DNE and their kernel versions.

Name	DNE	KDNE	AKDNE
BALANCE	0.79 ± 0.02	**0.90 ± 0.01**	**0.83 ± 0.02**
BREAST	0.96 ± 0.01	**0.97 ± 0.01**	0.96 ± 0.01
GLASS	0.65 ± 0.04	**0.70 ± 0.03**	**0.67 ± 0.04**
IONOS	0.87 ± 0.02	**0.95 ± 0.02**	**0.95 ± 0.02**
IRIS	0.95 ± 0.02	**0.97 ± 0.02**	**0.96 ± 0.02**
MUSK2	0.89 ± 0.02	**0.91 ± 0.01**	0.89 ± 0.02
PIMA	0.67 ± 0.02	**0.69 ± 0.02**	**0.70 ± 0.03**
SATELLITE	0.84 ± 0.01	**0.85 ± 0.01**	**0.85 ± 0.01**
YEAST	0.40 ± 0.05	**0.48 ± 0.01**	**0.47 ± 0.04**
M-USPS	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01
W/D/L	–	9/1/0	7/3/0

original versions on real-world datasets, and (2) the performance of the kernel alignment method is comparable to a kernel which is exhaustively selected, but the kernel alignment method requires much shorter running time.

To measure the generalization performance of each algorithm, we use the nine real-world datasets obtained from the UCI repository [1]: BALANCE, BREAST, GLASS, IONOS, IRIS, MUSK2, PIMA, SATELLITE and YEAST. We also experiment on a high-dimensional dataset called M-USPS which is a modified version of the famous USPS dataset [5, Chapter 21]. Following previous works, we use the 1NN classifier in all experiments, and we randomly divide each dataset into training and testing sets. By repeating the process 40 times, we have 40 training and testing sets for each dataset. The generalization performance of each algorithm is then measured by the average test accuracy over the 40 testing sets of 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.

To specify base kernels for kernel alignment, except for M-USPS, we consider scaled RBF base kernels [24, p. 216], $k(x,y)=\exp(-\|x-y\|^2/2D\sigma^2)$ where D is the dimensionality of input data. Twenty one base kernels specified by the following values of σ are considered: 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. For M-USPS, due to some knowledge on previous experiments e.g. [24, p. 216], we consider polynomial base kernels with degrees 1–5. The kernelized algorithms which apply kernel alignment and cross validation are implemented by the KPCA trick illustrated in Fig. 1.

The experimental results are shown in Tables 1–3. The three learners using aligned kernels are denoted by AKNCA, AKLMNN and AKDNE. From the results, it is clear that the kernelized algorithms usually improve the performance of their original algorithms. The kernelized algorithms applying cross validation obtain the best performance. They outperform the original methods in 28 out of 30 datasets. The three learners applying aligned kernels also achieve good performance as they outperform the original algorithms on 24 datasets and obtain an equal performance on 4 datasets. Only 2 out of 30 datasets where the original

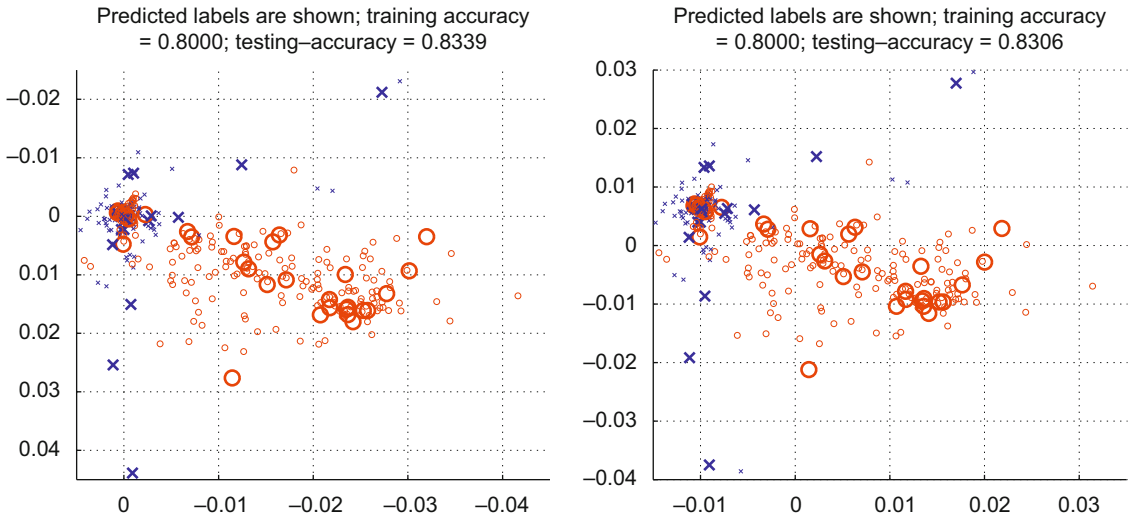

Fig. 3. An example of usual experimental results on Ionosphere where the KPCA trick (Left) and the kernel trick (Right) result in insignificantly different subspaces (only one test data is differently classified). Big points denote training data and little points denote testing data where their predicted labels are shown.

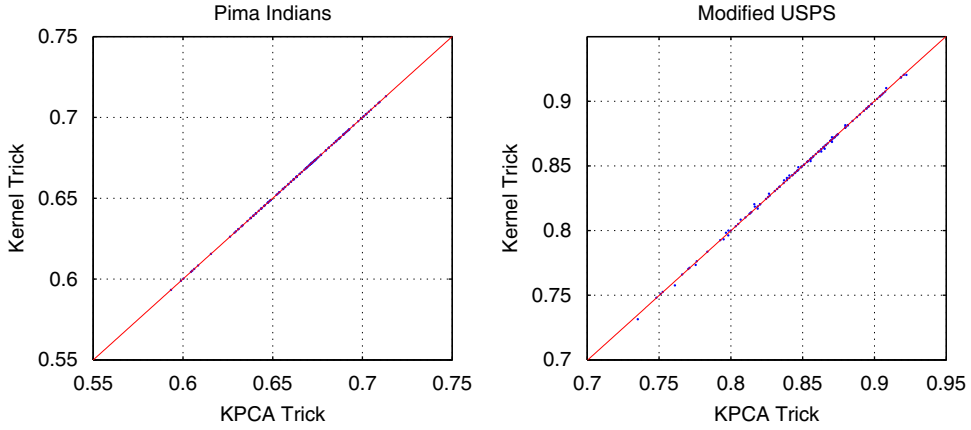

Fig. 4. Examples of datasets, Pima and M-USPS, which are not ill-conditioned with respect to KDNE. Therefore, the two frameworks result in usual cases of indifferent or insignificantly different Mahalanobis distances similar to that of Fig. 3. Points illustrate accuracies for the two frameworks on experiment settings varying from two different polynomial-kernel degrees, $d \in \{2,\ldots,5\}$, $n \in \{50,100,150\}$ and $k \in \{1,\ldots,5\}$ (notations are introduced in Section 2).

algorithms outperform the kernel algorithms applying kernel alignment.

We note that although the cross validation method usually provides slightly better performance than the kernel alignment method, the kernel alignment method provides comparable results in much shorter running time. For each dataset, a run-time overhead of the kernelized algorithms applying cross validation is of several hours (on Pentium IV 1.5 GHz, Ram 1 GB) while run-time overheads of the kernelized algorithms applying aligned kernels are about minutes for each dataset. Therefore, in a time-limited circumstance, it is attractive to apply an aligned kernel. More experiments can be found in [6].

Note that the kernel alignment method is not appropriate for a multi-modal dataset in which there may be several clusters of data points for each class since, from Eq. (9), the function align (K,Y) will attain the maximum value if and only if all points of the same class are collapsed into a single point. This may be one reason which explains why cross validated kernels give better results than results of aligned kernels in our experiments. Developing a new kernel alignment algorithm which suitable for multi-modality is currently an open problem.

5.2. Kernel trick and KPCA trick: practical investigation

In Section 3.1, the representer theorems, proving in Appendix A, state that, in each learning problem, optimal points of the two frameworks must have the same objective value. Nevertheless, for learners where their objective functions are not convex (e.g. KNCA) or not strictly convex (e.g. KLMNN[5]), it is not surprising that the two frameworks may not give an identical Mahalanobis distance. In the case of non-convexity a learner itself does not guarantee to find a global optimal solution, and in the case of non-strict-convexity there are plenty of global optimal solutions so that an obtained Mahalanobis distance does not depend on the KPCA trick or the kernel trick, but on an initial condition and on an optimizer's mechanism.

It is interesting to note that, in practice, even KDNE, which has a strictly convex (quadratic) objective function, can sometimes have different results obtained from the two kernelization frameworks. This is mainly because KDNE's involved matrices, K

[5] Because of the $[\cdot]_+$ function in its objective function.

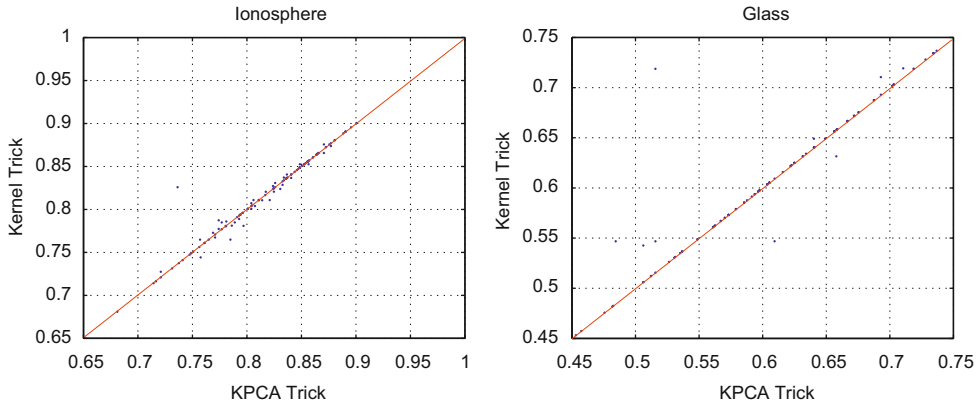

Fig. 5. Examples of datasets, IONOSPHERE and GLASS, which are, in a few cases, ill-conditioned with respect to KDNE (setting is the same as Fig. 4): the two frameworks occasionally give significantly different Mahalanobis distances.

and $K(D-W)K$, can be *ill-conditioned*, i.e. their *condition numbers* [8,18] are very high. See also Section 3.3 of [2].

Fig. 3 illustrates a situation where the two frameworks give very-slightly different subspaces where the condition number of $K(D-W)K$ is in the order of 10^{18}. The difference seems to be indistinguishable by human, and their accuracies on the test dataset are different by only one test data. This example shows that matrices having condition numbers in the order of 10^{18} are still not ill-conditioned with respect to the MATLAB'S EIG function. Fig. 4 illustrates more examples about this usual indistinguishable-difference situation. Nevertheless, when condition numbers are extremely high, differences become distinguishable. Fig. 5 illustrates examples about this distinguishable-difference situation. There are few points in Fig. 5 where the two frameworks provide totally different accuracies. Condition numbers of K and $K(D-W)K$ of these points are as high as 10^{32} for IONOSPHERE and 10^{35} for GLASS. Investigations of various numerical eigenvalue-decomposition approaches and of methods to improve the condition number of a matrix (e.g. [19]) are beyond the scope of this paper and are potential future works.

6. Summary

We have presented an alternative kernelization framework called "KPCA trick" which is applicable to Mahalanobis distance learners, as well as a kernel alignment method to efficiently select an appropriate kernel function. Advantages of our framework over the classical kernel-trick framework have been illustrated and evidence showing satisfiable performance of kernel alignment has been reported on three recent algorithms which previously did not have kernel versions. We believe that the methods presented in this paper can be extended to pattern recognition in other settings, and this possibility will be the main subject of our future work.

Acknowledgement

This work is supported by the Thailand Research Fund and the 90th Anniversary of Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment). We thank Wicharn Lewkeeratiyutkul who taught us the theory of Hilbert space. We also thank the editors and anonymous reviewers who gave constructive comments to improve this paper.

Appendix A. Proofs of Theorems 1 and 2

To start proving the two theorems, 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 $A:\mathcal{X}\to\mathcal{Y}$ can be uniquely decomposed as $\sum_{i=1}^{\infty}\langle\cdot,\tau_i\rangle_{\mathcal{X}}e_i$ for some $\{\tau_i\}_{i\in\mathbb{N}}\subseteq\mathcal{X}$.*

Proof. As A is bounded, the linear functional $\phi\mapsto\langle A\phi,e_i\rangle_{\mathcal{Y}}$ is bounded for every i since, by Cauchy–Schwarz inequality, $\|\langle A\phi,e_i\rangle_{\mathcal{Y}}\|\le\|A\phi\|\|e_i\|\le\|A\|\|\phi\|$. By Riesz representation theorem, the map $\langle A\cdot,e_i\rangle_{\mathcal{Y}}$ can be written as $\langle\cdot,\tau_i\rangle_{\mathcal{X}}$ for a unique $\tau_i\in\mathcal{X}$. Since $\{e_i\}_{i\in\mathbb{N}}$ is an orthonormal basis of $\mathcal{Y}$, for every $\phi\in\mathcal{X}$, $A\phi=\sum_{i=1}^{\infty}\langle A\phi,e_i\rangle_{\mathcal{Y}}e_i=\sum_{i=1}^{\infty}\langle\phi,\tau_i\rangle_{\mathcal{X}}e_i$. □

Proof of Theorem 1. To avoid complicated notations, we omit subscripts such as $\mathcal{X},\mathcal{Y}$ of inner products. The proof will consist of two steps. In the first step, we will prove the theorem by assuming that $\{\tilde{\psi}_i\}_{i=1}^n$ is an orthonormal set. In the second step, we prove the theorem in general cases where $\{\tilde{\psi}_i\}_{i=1}^n$ is not necessarily orthonormal. The proof of the first step requires an application of Fubini theorem [15].

Step 1: Assume that $\{\tilde{\psi}_i\}_{i=1}^n$ is an orthonormal set. Let $\{e_i\}_{i=1}^{\infty}$ be an orthonormal basis of $\mathcal{Y}$. For any $\phi'\in\mathcal{X}$, we have, by Lemma 1, $A\phi'=\sum_{k=1}^{\infty}\langle\phi',\tau_k\rangle e_k$. Hence, for each bounded linear map $A:\mathcal{X}\to\mathcal{Y}$, and $\phi,\phi'\in span(\{\tilde{\psi}_i\}_{i=1}^n)$, we have $\langle A\phi,A\phi'\rangle=\sum_{k=1}^{\infty}\langle\phi,\tau_k\rangle\langle\phi',\tau_k\rangle$.

Note that each τ_k can be decomposed as $\tau'_k+\tau_k^{\perp}$ such that τ'_k lies in $span(\{\tilde{\psi}_i\}_{i=1}^n)$ and $\tau_k^{\perp}$ is orthogonal to the span. These facts make $\langle\phi',\tau_k\rangle=\langle\phi',\tau'_k\rangle$ for every k. Moreover, $\tau'_k=\sum_{j=1}^n u_{kj}\tilde{\psi}_j$, for some $\{u_{k1},\ldots,u_{kn}\}\subset\mathbb{R}^n$. Hence, we have

$$\begin{aligned}
\langle A\phi,A\phi'\rangle &= \sum_{k=1}^{\infty}\langle\phi,\tau_k\rangle\langle\phi',\tau_k\rangle=\sum_{k=1}^{\infty}\langle\phi,\tau'_k\rangle\langle\phi',\tau'_k\rangle\\
&= \sum_{k=1}^{\infty}\left\langle\phi,\sum_{i=1}^n u_{ki}\tilde{\psi}_i\right\rangle\left\langle\phi',\sum_{i=1}^n u_{ki}\tilde{\psi}_i\right\rangle\\
&= \sum_{k=1}^{\infty}\sum_{i,j=1}^n u_{ki}u_{kj}\langle\phi,\tilde{\psi}_i\rangle\langle\phi',\tilde{\psi}_j\rangle\\
&= \sum_{i,j=1}^n\left(\sum_{k=1}^{\infty}u_{ki}u_{kj}\right)\langle\phi,\tilde{\psi}_i\rangle\langle\phi',\tilde{\psi}_j\rangle\ (\dagger)\\
&= \sum_{i,j=1}^n G_{ij}\langle\phi,\tilde{\psi}_i\rangle\langle\phi',\tilde{\psi}_j\rangle=\tilde{\varphi}^TG\tilde{\varphi}'=\tilde{\varphi}^TA'^TA'\tilde{\varphi}'.
\end{aligned}$$

At the fourth equality marked by (†), we apply Fubini theorem to swap the two summations. To see that Fubini theorem can be

applied at the fourth equality, we first note that $\sum_{k=1}^{\infty} u_{ki}^2$ is finite for each $i \in \{1 \dots n\}$ since

$$\sum_{k=1}^{\infty} u_{ki}^2 = \sum_{k=1}^{\infty} \left\langle \tilde{\psi}_i, \sum_{j=1}^{n} u_{kj}\tilde{\psi}_j \right\rangle \left\langle \tilde{\psi}_i, \sum_{j=1}^{n} u_{kj}\tilde{\psi}_j \right\rangle = \|A\tilde{\psi}_i\|^2 < \infty.$$

Applying the above result together with Cauchy–Schwarz inequality and Fubini theorem for non-negative summation, we have

$$\begin{aligned}
&\sum_{k=1}^{\infty}\sum_{i,j=1}^{n} |u_{ki}u_{kj}\langle \phi,\tilde{\psi}_i\rangle\langle \phi',\tilde{\psi}_j\rangle| \\
&= \sum_{i,j=1}^{n}\sum_{k=1}^{\infty} |u_{ki}u_{kj}\langle \phi,\tilde{\psi}_i\rangle\langle \phi',\tilde{\psi}_j\rangle| \\
&= \sum_{i,j=1}^{n} |\langle \phi,\tilde{\psi}_i\rangle\langle \phi',\tilde{\psi}_j\rangle| \left(\sum_{k=1}^{\infty} |u_{ki}u_{kj}|\right) \\
&\le \sum_{i,j=1}^{n} |\langle \phi,\tilde{\psi}_i\rangle\langle \phi',\tilde{\psi}_j\rangle| \sqrt{\left(\sum_{k=1}^{\infty} u_{ki}^2\right)\left(\sum_{k=1}^{\infty} u_{kj}^2\right)} < \infty.
\end{aligned}$$

Hence, the summation converges absolutely and thus Fubini theorem can be applied as claimed above. Again, using the fact that $\sum_{k=1}^{\infty} u_{ki}^2 < \infty$, we have that each element of G, $G_{ij} = \sum_{k=1}^{\infty} u_{ki}u_{kj}$, is finite. Furthermore, the matrix G is PSD since each of its elements can be regarded as an inner product of two vectors in ℓ_2.

Hence, we finally have that $\langle A\phi_i, A\phi_j\rangle = \tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j$, for each $1 \le i,j \le n$. Hence, whenever a map A is given, we can construct A' 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. The first step of the proof is finished.

Step 2: We now prove the theorem without assuming that $\{\tilde{\psi}_i\}_{i=1}^n$ is an orthonormal set. Let all notations be the same as in Step 1. Let Ψ' be the matrix $(\tilde{\psi}_1, \dots, \tilde{\psi}_n)$. Define $\{\psi_i\}_{i=1}^n$ as an orthonormal set such that $span(\{\psi_i\}_{i=1}^n) = span(\{\tilde{\psi}_i\}_{i=1}^n)$ and $\Psi = (\psi_1, \dots, \psi_n)$ and $\varphi_i = \Psi^T \phi_i$. Then, we have that $\tilde{\psi}_i = \Psi \mathbf{c}_i$ for some $\mathbf{c}_i \in \mathbb{R}^n$ and $\Psi' = \Psi C$ where $C = (\mathbf{c}_1, \dots, \mathbf{c}_n)$. Moreover, since C map from an independent set $\{\psi_i\}$ to another independent set $\{\tilde{\psi}_i\}$, C is invertible. We then have, for any A',

$$\tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j = \phi_i^T \Psi' A'^T A' \Psi'^T \phi_j = \phi_i^T \Psi C A'^T A' C^T \Psi^T \phi_j = \varphi_i^T C A'^T A' C^T \varphi_j = \varphi_i^T B^T B \varphi_j,$$

where $A'C^T = B$ and $A' = B(C^T)^{-1}$. Hence, for any B we have the matrix A' which gives $\tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j = \varphi_i^T B^T B \varphi_j$. Using the same arguments as in Step 1, we finish the proof of Step 2 and of Theorem 1. □

The proof of Theorem 2 is a generalization of the proofs in previous works [24, Chapter 4].

Proof of Theorem 2. Let $\{e_i\}_{i=1}^d$ be the canonical basis of $\mathbb{R}^d$, and let $\phi \in span\{\tilde{\psi}_1, \dots \tilde{\psi}_n\}$. By Lemma 1, $A\phi = \sum_{i=1}^d \langle \phi, \tau_i\rangle e_i$ for some $\tau_1, \dots, \tau_d \in \mathcal{X}$. Each τ_i can be decomposed as $\tau_i' + \tau_i^{\perp}$ such that τ_i' lies in $span\{\tilde{\psi}_1, \dots, \tilde{\psi}_n\}$ and $\tau_i^{\perp}$ is orthogonal to the span. These facts make $\langle \phi, \tau_i\rangle = \langle \phi, \tau_i'\rangle$ for every i. We then have, for some $u_{ij} \in \mathbb{R}$, $1 \le i \le d$, $1 \le j \le n$,

$$\begin{aligned}
A\phi &= \sum_{i=1}^{d} \left\langle \phi, \sum_{j=1}^{n} u_{ij}\tilde{\psi}_j \right\rangle e_i = \sum_{i=1}^{d} e_i \sum_{j=1}^{n} u_{ij}\langle \phi, \tilde{\psi}_j\rangle \\
&= \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{d1} & \cdots & u_{dn} \end{bmatrix} \begin{bmatrix} \langle \phi, \tilde{\psi}_1\rangle \\ \vdots \\ \langle \phi, \tilde{\psi}_n\rangle \end{bmatrix} = U\tilde{\varphi}.
\end{aligned}$$

Since every ϕ_i is in the span, we conclude that $A\phi_i = U\tilde{\varphi}_i$. Now, one can easily check that $\langle A\phi_i, A\phi_j\rangle = \tilde{\varphi}_i^T U^T U \tilde{\varphi}_j$. 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 (by renaming U to A'). □

Finally, notice that the statement of classical representer theorems deals with the representation of an optimal hyperplane. In the same sense, our theorems also identify the representation of an optimal linear map A^*. To see this, from the proofs of our theorems, we can write, for any A and ϕ,

$$A\phi = U\tilde{\varphi} = U\Psi'^T \phi$$

where $\Psi' = (\tilde{\psi}_1, \dots, \tilde{\psi}_n)$. Therefore, for an optimal linear map A^*, there exists a finite-dimensional matrix $U^* \in \mathbb{R}^{d \times n}$ such that $A^* = U^* \Psi'^T$, and since Ψ' is fixed, e.g. $\tilde{\psi}_i = \phi_i$ in the case of the kernel trick or $\tilde{\psi}_i = \psi_i$ in the case of KCPA trick, virtually, we can represent an (in general, infinite-dimensional) optimal map A^* by a *finite-dimensional* map U^*.

Appendix B. Proofs of Proposition 1

Proof. Define a matrix $B_i^{\phi} = (0,0,\dots,\phi,\dots,0,0)$ as a matrix with its i th column is ϕ and zero vectors otherwise. Substitute $A = U\Phi^T$ to Eq. (5) we have

$$\begin{aligned}
\frac{\partial f^{KNCA}}{\partial A} &= -2U\sum_i \left(p_i \sum_k p_{ik}\mathbf{k}_{ik}\phi_{ik}^T - \sum_{j \in c_i} p_{ij}\mathbf{k}_{ij}\phi_{ij}^T \right) \\
&= -2U\sum_i \left(p_i \sum_k p_{ik}(B_i^{\mathbf{k}_{ik}} - B_k^{\mathbf{k}_{ik}}) - \sum_{j \in c_i} p_{ij}(B_i^{\mathbf{k}_{ij}} - B_j^{\mathbf{k}_{ij}}) \right) \Phi^T = V\Phi^T,
\end{aligned}$$

which completes the proof. □

References

[1] A. Asuncion, D.J. Newman, UCI machine learning repository, 2007.
[2] F. Bach, M. Jordan, Learning spectral clustering, with application to speech separation, Journal of Machine Learning Research 7 (2006) 1963–2001.
[3] S. Boyd, L. Vandenberghe, Convex Optimization, 2004.
[4] O. Chapelle, B. Schölkopf, Incorporating invariances in nonlinear SVMs, in: NIPS, 2001.
[5] O. Chapelle, B. Schölkopf, A. Zien (Eds.), Semi-supervised Learning, The MIT Press, Cambridge, MA, 2006.
[6] R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, B. Kijsirikul, On kernelizing Mahalanobis distance learning algorithms, 2008. Arxiv preprint ⟨http://arxiv.org/abs/0804.1441⟩.
[7] H.-T. Chen, H.-W. Chang, T.-L. Liu, Local discriminant embedding and its variants, in: CVPR, vol. 2, 2005.
[8] J.W. Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.
[9] A. Globerson, S. Roweis, Metric learning by collapsing classes, in: NIPS, 2006.
[10] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Neighbourhood components analysis, in: NIPS, 2005.
[11] Y. Guermeur, A. Lifchitz, R. Vert, A kernel for protein secondary structure prediction, in: Kernel Methods in Computational Biology, 2004.
[12] P. Jain, B. Kulis, J. Davis, I. Dhillon, Metric and kernel learning using a linear transformation, 2009. Arxiv preprint: arXiv:0910.5932.
[13] G. Kimeldorf, G. Wahba, Some results on Tchebycheffian spline functions, Journal of Mathematical Analysis and Applications 33 (1971) 82–95.
[14] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, Journal of Machine Learning Research 5 (2004) 27–72.
[15] W. Lewkeeratiyutkul, Lecture Notes on Real Analysis I–II ⟨http://pioneer.netserv.chula.ac.th/∼lwicharn/2301622/⟩, 2006.
[16] J. Li, X. Li, D. Tao, KPCA for semantic object extraction in images, Pattern Recognition 41 (2008) 3244–3250.
[17] X. Li, S. Lin, S. Yan, D. Xu, Discriminant locally linear embedding with high-order tensor data, Systems Man and Cybernetics—Part B 38 (2008) 342–352.
[18] A. Ng, A. Zheng, M. Jordan, Link analysis, eigenvectors and stability, in: IJCAI, 2001, pp. 903–910.
[19] V. Pan, G. Qian, A. Zheng, Preconditioning, randomization, solving linear systems, eigen-solving, and root-finding, in: The 2009 Conference on Symbolic Numeric Computation, ACM, New York, NY, 2009.
[20] Y. Pang, Z. Liu, N. Yu, A new nonlinear feature extraction method for face recognition, Neurocomputing 69 (7–9) (2006) 949–953.

[21] Y. Pang, L. Wang, Y. Yuan, Generalized KPCA by Adaptive Rules in Feature Space, International Journal of Computer Mathematics (2009). DOI: 〈10.1080/00207160802044118〉.
[22] Y. Pang, Y. Yuan, X. Li, Effective feature extraction in high-dimensional space, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 38 (6) (2008) 1652–1656.
[23] B. Schölkopf, R. Herbrich, A.J. Smola, A generalized representer theorem, in: COLT, 2001.
[24] B. Schölkopf, A.J. Smola, Learning with kernels, 2001.
[25] M. Sugiyama, Local fisher discriminant analysis for supervised dimensionality reduction, in: ICML, 2006.
[26] L. Torresani, K. Lee, Large margin components analysis, in: NIPS, 2007.
[27] K. Weinberger, J. Blitzer, L. Saul, Distance metric learning for large margin nearest neighbor classification, in: NIPS, 2006.
[28] E.P. Xing, A.Y. Ng, M.I. Jordan, S. Russell, Distance metric learning with application to clustering with side-information, in: NIPS, 2003.
[29] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, Pattern Analysis and Machine Intelligence 29 (1) (2007).
[30] J. Yang, J.Y. Yang, Why can LDA be performed in PCA transformed space? Pattern Recognition 36 (2003) 563–566.
[31] L. Yang, R. Jin, R. Sukthankar, Y. Liu, An efficient algorithm for local distance metric learning, in: AAAI, 2006.
[32] H. Yu, J. Yang, A direct LDA algorithm for high-dimensional data—with application to face recognition, Pattern Recognition 34 (2001) 2067–2070.
[33] C. Zhang, F. Nie, S. Xiang, A general kernelization framework for learning algorithms based on kernel PCA, Neurocomputing 73 (2009) 959–967.
[34] T. Zhang, X. Li, D. Tao, J. Yang, Local coordinates alignment (LCA): a novel manifold learning approach, International Journal of Pattern Recognition and Artificial Intelligence 22 (4) (2008) 667–690.
[35] T. Zhang, X. Li, D. Tao, J. Yang, Patch alignment for dimensionality reduction, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1299–1313.
[36] W. Zhang, X. Xue, Z. Sun, Y.-F. Guo, H. Lu, Optimal dimensionality of metric space for classification, in: ICML, 2007.
[37] X. Zhu, J. Kandola, Z. Ghahramani, J. Lafferty, Nonparametric transforms of graph kernels for semi-supervised learning, in: NIPS, 2005.

Ratthachat Chatpatanasiri is a PhD student at the Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.

Teesid Korsrilabutr is a Master student at the Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.

Pasakorn Tangchanachaianan is a PhD student at the Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.

Boonserm Kijsirikul is a professor at the Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.