Nonlinear Projection Trick in kernel methods: An alternative to the Kernel Trick

Nojun Kwak, *Member, IEEE,*

***Abstract*—In kernel methods such as kernel PCA and support vector machines, the so called *kernel trick* is used to avoid direct calculations in a high (virtually infinite) dimensional kernel space. In this paper, based on the fact that the effective dimensionality of a kernel space is less than the number of training samples, we propose an alternative to the kernel trick that explicitly maps the input data into a reduced dimensional kernel space. This is easily obtained by the eigenvalue decomposition of the kernel matrix. The proposed method is named as the *nonlinear projection trick* in contrast to the *kernel trick*. With this technique, the applicability of the kernel methods is widened to arbitrary algorithms that do not utilize the dot product. The equivalence between the kernel trick and the nonlinear projection trick is shown for several conventional kernel methods. In addition, we extend PCA-L1, which utilizes L_1-norm instead of L_2-norm (or dot product), into a kernel version and show the effectiveness of the proposed approach.**

***Index Terms*—Kernel methods, nonlinear projection trick, KPCA, Support vector machines, KPCA-L1, dimensionality reduction.**

I. Introduction

In pattern recognition and machine learning societies, kernel methods have started to gain attention since mid-1990s and have been successfully applied in pattern classification, dimensionality reduction, regression problems and so on [1] [2] [3] [4] [5] [6] [7] [8] [9] [10].

In kernel methods, the *kernel trick* is used to simplify the optimization in a high dimensional feature space[1] without an explicit mapping of input data into the feature space. This is done by introducing a kernel function of two elements in the input space and regarding it as a dot product of the corresponding two elements in the kernel space. If the optimization problem contains the elements in the feature space only in the form of the dot product, then the problem can be solved without knowing the explicit mapping. The dimension of the feature space, which is unmeasurable and can be infinite, depends on the specific kernel function used.

Because the kernel trick takes care of the dot product in the kernel space rather than the mapping itself, there has been little attention on the mapping of the input data to the feature space. However, there are many optimization problems that cannot be formulated by the dot product only, thus could not be extended to a kernel space by the kernel trick. In addition, without the knowledge of the mapped image in the feature space, it is difficult to solve data reconstruction problem by the kernel methods. Although, this *pre-image problem* in kernel PCA has been tackled in [11] and [12], a direct mapping of the input data to a vector in the feature space by kernel methods has not been studied yet.

It is well known that the mapped images of the finite number of training samples constitute a subspace of the possibly infinite dimensional feature space [1]. In addition, most operations in the kernel methods are performed in the training stage, which are done on this finite dimensional subspace of the possibly infinite dimensional feature space. Based on this idea, in this paper, we doubt and answer the question *"Can we find a direct mapping of the input data to a vector in the feature space by kernel methods?"*

This paper proposes an alternative method to the kernel trick which directly maps input samples into a reduced dimensional kernel space. The proposed approach is named as the *nonlinear projection trick* in contrast to the *kernel trick*. The proposed nonlinear kernel trick widens the applicability of kernel methods to any algorithms (even if their formulation does not rely only on the dot product) that are applicable in the input space. We also show the equivalence between the proposed nonlinear projection trick and the kernel trick by some examples. As an application of the nonlinear kernel trick, PCA-L1 [13], whose objective function is in L_1-norm rather than L_2-norm, is extended to a kernel version, which would not have been possible by the kernel trick.

The paper is organized as follows. In the next section, the structure of a kernel space such as the bases and the coordinates are studied. In section III, the proposed *nonlinear projection trick* is formulated. The proposed approach is applied to the conventional kernel methods such as KPCA [1], KSVM (kernel support vector machine) [2] and KFD (kernel fisher discriminant) [4] in Section IV to show the equivalence between the kernel trick and the nonlinear projection trick. In addition, a kernel version of PCA-L1 (KPCA-L1) which could not have been derived from the conventional kernel trick is derived by the nonlinear projection trick in this section. Finally, conclusions follow in Section V.

II. The structure of a kernel space

In this section, the structure, such as the bases and the corresponding coordinates, of a kernel space with finite number of training samples is studied. Before going on, in the following subsection, we briefly define the notations and variables that will be used in this paper.

N. Kwak is an associate professor at the Graduate School of Convergence Science and Technology, Seoul National University, Korea.

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2013006599)

[1]The terms *feature space* and *kernel space* are used interchangeably to denote the high dimensional space defined by a kernel function.

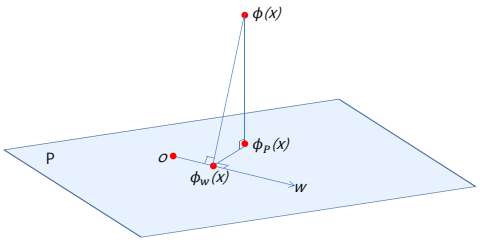

Fig. 1. Projections in the feature space.

A. Notations

- $X \triangleq [x_1, \cdots, x_n] \in \mathbb{R}^{d\times n}$: the training data matrix where d is the dimension of the input space and n is the number of training samples.
- $\Phi(X) \triangleq [\phi(x_1), \cdots, \phi(x_n)] \in \mathbb{R}^{f\times n}$: the mapped training data in the f-dimensional feature space after the mapping $\phi(\cdot)$. We consider $\phi(x_i)$ as a vector with respect to a countable orthornormal basis of the respective RKHS (reproducing kernel Hilbert space). $\Phi(X)$ is assumed to have zero mean, i.e., $\sum_{i=1}^n \phi(x_i) = 0$. Note that f can be infinite.
- $k(x,y) \triangleq \langle\phi(x), \phi(y)\rangle = \phi(x)^T\phi(y) \in \mathbb{R}$: a kernel function of any two inputs $x, y \in \mathbb{R}^d$.
- $K \triangleq \Phi(X)^T\Phi(X) = [k(x_i, x_j)] \in \mathbb{R}^{n\times n}$: a kernel matrix of the training data. The rank of K is r. It becomes $r \leq n-1$ because $\Phi(X)$ is assumed to be centered.
- $k(x) \triangleq \Phi(X)^T\phi(x) \in \mathbb{R}^n$: a kernel vector of any $x \in \mathbb{R}^d$.
- P: an r-dimensional subspace of the feature space formed by the mapped training samples $\Phi(X)$.
- $\phi_P(x)$: the projection of $\phi(x)$ onto P. If x lies on P (e.g., one of the training samples), $\phi_P(x) = \phi(x)$.
- $\phi_w(x)$: the projection of $\phi(x)$ onto a one-dimensional vector space formed by a vector $w \in \mathbb{R}^f$. In most cases, the vector w is restricted to reside in the subspace P, i.e., $w = \Phi(X)\alpha$ for some $\alpha \in \mathbb{R}^n$.

The relationship among $\phi(x)$, $\phi_P(x)$ and $\phi_w(x)$ is illustrated in Fig. 1. Note that if w resides in P as in the figure, then $w^T\phi(x) = w^T\phi_P(x)$.

B. The bases and the coordinates of a kernel space

It is well known that the effective dimensionality [2] of the feature space defined by the kernel function $k(\cdot,\cdot)$ given a finite number of training samples X is finite [1]. In this subsection, the bases of the subspace spanned by the mapped training data $\Phi(X)$, as well as the coordinate of the mapping $\phi(x)$ of an unseen data $x \in \mathbb{R}^d$ are specified.

Lemma 1: (Bases of P) Let r be the rank of K and $K = U\Lambda U^T$ be an eigenvalue decomposition of K composed of only the nonzero eigenvalues of K where $U = [u_1, \cdots, u_r] \in \mathbb{R}^{n\times r}$ whose columns are orthonormal and $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_r) \in \mathbb{R}^{r\times r}$. Then, the columns of $\Pi \triangleq \Phi(X)U\Lambda^{-\frac{1}{2}} = [\pi_1, \cdots, \pi_r] \in \mathbb{R}^{f\times r}$ constitute orthonormal bases of P.

Proof: Because K is the outer product of $\Phi(X)$ and itself, it is obvious that r, the rank of K, is the dimension of the subspace spanned by $\Phi(X)$. By utilizing the orthonormality of U ($U^TU = I_r$) and the definition of K ($K = \Phi(X)^T\Phi(X)$), it is easy to check that $\Pi^T\Pi = I_r$ and Π becomes orthonormal bases of P. ■

From the above lemma, $\pi_i = \Phi(X)u_i\lambda_i^{-\frac{1}{2}}$, $(1 \leq i \leq r)$ becomes the i-th basis vector of P, where u_i and λ_i are the i-th unit eigenvector and eigenvalue of K respectively.

Theorem 2: (Coordinate of $\phi_P(x)$) For any $x \in \mathbb{R}^d$, the projection $\phi_P(x)$ of $\phi(x)$ onto P is obtained by $\phi_P(x) = \Pi y$ where $y = \Lambda^{-\frac{1}{2}}U^Tk(x)$, which can be regarded as the coordinate of $\phi_P(x)$ in the cartesian coordinate system whose orthonormal bases are Π.

Proof: Because $\phi_P(x)$ is in P, $\phi_P(x) = \Pi y$ for some $y \in \mathbb{R}^r$. If $\phi_P(x)$ is a projection of $\phi(x)$ onto P whose bases are Π, it should minimize $||\phi(x) - \phi_P(x)||_2$. Therefore, the optimal y should satisfy the following.

$$\begin{aligned} y^\star &= \underset{y}{\mathrm{argmin}}\, ||\phi(x) - \Pi y||_2^2 \\ &= \underset{y}{\mathrm{argmin}}\, ||\phi(x)||_2^2 - 2\phi(x)^T\Pi y + y^T\Pi^T\Pi y \\ &= \underset{y}{\mathrm{argmin}}\, y^Ty - 2\phi(x)^T\Pi y \end{aligned} \tag{1}$$

The solution of the quadratic problem is $y = \Pi^T\phi(x) = \Lambda^{-\frac{1}{2}}U^Tk(x)$. ■

Note that $\phi_P(x)$ can be regarded as a nonlinear mapping of x onto the r-dimensional subspace spanned by Π. From the above theorem, this mapping can further be thought of as a direct mapping of x to $y = \Lambda^{-\frac{1}{2}}U^Tk(x)$.

Corollary 3: (Coordinates of the mapped training data) The coordinate of $\Phi(X)$ is $Y = \Lambda^{\frac{1}{2}}U^T$.

Proof: It is easy to show that the training data X is mapped to $Y = \Lambda^{-\frac{1}{2}}U^TK = \Lambda^{-\frac{1}{2}}U^TU\Lambda U^T = \Lambda^{\frac{1}{2}}U^T$. Note that $K = U\Lambda U^T = Y^TY$ [3]. ■

Lemma 4: (Mean of the mapped training data) The mapped training data $Y = \Lambda^{\frac{1}{2}}U^T$ are centered, i.e., $\sum_{i=1}^n y_i = 0$.

Proof: From *Theorem 2*, $y_i = \Lambda^{-\frac{1}{2}}U^Tk(x_i)$ and it becomes $\sum_{i=1}^n y_i = \Lambda^{-\frac{1}{2}}U^T\sum_{i=1}^n k(x_i) = \Lambda^{-\frac{1}{2}}U^T\Phi(X)^T\sum_{i=1}^n \phi(x_i) = 0$. ■

Lemma 5: (Coordinate of the residual) For any $x \in \mathbb{R}^d$, let $X' \triangleq [X, x] \in \mathbb{R}^{d\times(n+1)}$ and $\Phi(X') \triangleq [\Phi(X), \phi(x)] \in \mathbb{R}^{f\times(n+1)}$. Let $\delta\phi_P(x) \triangleq \phi(x) - \phi_P(x)$ be the residual vector of $\phi(x)$ with respect to P. If $\delta\phi_P(x) \neq 0$, $\Phi(X')$ lies in a $(r+1)$-dimensional subspace containing P. Furthermore, the coordinate of the mapped training data $\Phi(X)$ is $[Y^T, 0]^T \in \mathbb{R}^{(r+1)\times n}$ while the coordinate of $\phi(x)$ becomes $[y^T, y_{r+1}]^T \in \mathbb{R}^{r+1}$ where $y_{r+1} = \sqrt{k(x,x) - y^Ty}$.

Proof: It is obvious that $\delta\phi_P(x)$ is orthogonal to P because $\phi_P(x)$ is the projection of $\phi(x)$ onto P. Furthermore,

[2]The term *intrinsic dimensionality* is defined and widely used in the machine learning society [14]. Because projecting methods based on eigenvalue decomposition tends to overestimate the intrinsic dimensionality [15], the *effective dimensionality* can be interpreted as an upper bound on intrinsic dimensionality of a kernel space.

[3]Note that K can be directly decomposed into $K = Y'^TY'$ by Cholesky decomposition. In this case, Y' is a unique upper triangle matrix. The singular value decomposition of Y' is $Y' = V\Lambda^{\frac{1}{2}}U^T = VY$ for some unitary matrix V. Therefore, Y' can be interpreted as a rotation of Y.

if the residual vector $\delta\phi_P(x) \neq 0$, $\delta\phi_P(x)$ adds one dimension to $\phi(x)$, thus $\Phi(X')(= [\Phi(X), \phi(x)])$ lies in a $(r+1)$-dimensional subspace P' of the f-dimensional feature space which contains P.

If we retain the bases $\Pi(= [\pi_1, \cdots, \pi_r])$ of P as the r bases of P', then the new bases Π' of P' becomes $\Pi' = [\Pi, \pi_{r+1}]$ where $\pi_{r+1} = \frac{\delta\phi_P(x)}{||\delta\phi_P(x)||_2}$.

In the coordinate system of Π', the first r coordinates of the mapped training data $\Phi(X)$ are unchanged with the $(r+1)$-st coordinate being identically 0. Likewise, the first r coordinates of $\phi(x)$ are identical to the coordinate of $\phi_P(x)$, i.e., y, while the $(r+1)$-th coordinate becomes the length (norm) of the residual vector $\delta\phi_P(x)$.

Finally, because $\phi(x) = \phi_P(x) + \delta\phi_P(x)$ and $\phi_P(x) \perp \delta\phi_P(x)$, it becomes

$$\begin{aligned} ||\delta\phi_P(x)||_2^2 &= ||\phi(x)||_2^2 - ||\phi_P(x)||_2^2 \\ &= k(x,x) - y^T y \end{aligned} \tag{2}$$

by Pythagorean trigonometric identity. ■

Lemma 6: (Coordinate of a vector w) If a vector w in P can be written in the form of $w = \Phi(X)\alpha$, then it can also be written as $w = \Pi\beta$ where $\beta = Y\alpha$.

Proof: Because Π is an orthonormal bases of P, any vector w in P can be written as $w = \Pi\Pi^T w$. Therefore,

$$\begin{aligned} w &= \Pi\Lambda^{-\frac{1}{2}}U^T\Phi(X)^T\Phi(X)\alpha = \Pi\Lambda^{-\frac{1}{2}}U^T K\alpha \\ &= \Pi Y\alpha = \Pi\beta. \end{aligned} \tag{3}$$

■

Note that $\beta = Y\alpha$ is the coordinate of w in P.

Corollary 7: (Coordinate of $\phi_w(x)$) The projection $\phi_w(x)$ of $\phi(x)$ onto w can be obtained by $\phi_w(x) = \Pi\gamma$ where $\gamma = \frac{\beta\beta^T}{\beta^T\beta}y$.

Proof: Let $w' = \frac{w}{||w||_2}$ be a unit vector. Then

$$\begin{aligned} \phi_w(x) &= w'(w'^T\phi(x)) = w'w'^T(\phi_P(x) + \delta\phi_P(x)) \\ &= w'w'^T\phi_P(x) = w'w'^T\Pi\Lambda^{-\frac{1}{2}}U^T k(x) \\ &= \frac{1}{\beta^T\beta}\Pi\beta\beta^T\Pi^T\Pi\Lambda^{-\frac{1}{2}}U^T k(x) \\ &= \Pi\frac{\beta\beta^T}{\beta^T\beta}\Lambda^{-\frac{1}{2}}U^T k(x) = \Pi\frac{\beta\beta^T}{\beta^T\beta}y = \Pi\gamma. \end{aligned} \tag{4}$$

In the second equality, $\delta\phi_P(x)$ is orthogonal to P, thus also orthogonal to w which lies in P. Note that if $||w||_2 = 1$, it becomes $||\beta||_2 = 1$ and $\gamma = \beta\beta^T y$. ■

III. The proposed method

In this section, by using the results in the previous section, we give a new interpretation of kernel methods that any kernel method is equivalent to applying the corresponding conventional method applicable in the input space to the data with the new coordinates. Before doing that, we deal with the problem of centerization in the feature space.

A. Centerization

In the previous section, the training data $\Phi(X)$ are assumed to be centered in the feature space, i.e., $\sum_{i=1}^n \phi(x_i) = 0$. If this condition is not satisfied, P which is made up of $\{p : p = \sum_{i=1}^n \alpha_i\phi(x_i), \alpha_i \in \mathbb{R}\}$ is not a subspace but a manifold in the feature space, i.e., $0 \notin P$. Therefore, the centerization is a necessary step in practical application of the proposed idea on kernel methods.

Let $\Psi(X) \triangleq [\psi(x_1), \cdots, \psi(x_n)] \in \mathbb{R}^{f\times n}$ be the uncentered data in the feature space and $\bar{\psi} \triangleq \frac{1}{n}\sum_{i=1}^n \psi(x_i) = \frac{1}{n}\Psi(X)\mathbf{1}_n \in \mathbb{R}^f$ be the mean of $\Psi(X)$. Here, $\mathbf{1}_n \triangleq [1, \cdots, 1]^T \in \mathbb{R}^n$ is a vector whose components are identically 1. By subtracting the mean $\bar{\psi}$ from the uncentered data $\Psi(X)$, the centered data $\Phi(X)$ can be obtained as

$$\begin{aligned} \Phi(X) &= \Psi(X) - \bar{\psi}\mathbf{1}_n^T = \Psi(X)(I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T) \\ &= \Psi(X)(I_n - E_n). \end{aligned} \tag{5}$$

where I_n is the $n\times n$ identity matrix and $E_n \triangleq \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ is an $n\times n$ matrix whose components are identically $\frac{1}{n}$.

Consider an uncentered kernel function $\kappa(a,b) \triangleq \psi(a)^T\psi(b)$. Given this kernel function and the training data X, the uncentered kernel matrix $\mathcal{K}$ is defined as $\mathcal{K} \triangleq [\kappa(x_i, x_j)] \in \mathbb{R}^{n\times n}$. Writing $\mathcal{K}$ in matrix form, it becomes $\mathcal{K} = \Psi(X)^T\Psi(X)$. Then, from (5), the centered positive definite kernel matrix K can be obtained as

$$K = \Phi(X)^T\Phi(X) = (I_n - E_n)\mathcal{K}(I_n - E_n) \tag{6}$$

Once a test data $x \in \mathbb{R}^d$ is presented, an uncentered kernel vector $\kappa(x) \triangleq [\kappa(x_1, x), \cdots, \kappa(x_n, x)]^T \in \mathbb{R}^n$ can be computed. Because $\kappa(x_i, x) = \psi(x_i)^T\psi(x)$, if we subtract the mean $\bar{\psi}$ from both $\psi(x_i)$ and $\psi(x)$, we can get the centered kernel vector $k(x)$ from the uncentered kernel vector $\kappa(x)$ as follows:

$$\begin{aligned} k(x) &= \Phi(X)^T\phi(x) \\ &= [\Psi(X)(I_n - E_n)]^T(\psi(x) - \bar{\psi}) \\ &= (I_n - E_n)\Psi(X)^T(\psi(x) - \frac{1}{n}\Psi(X)\mathbf{1}_n) \\ &= (I_n - E_n)[\kappa(x) - \frac{1}{n}\mathcal{K}\mathbf{1}_n]. \end{aligned} \tag{7}$$

B. Kernel methods: nonlinear projection trick

Now, a new framework of kernel methods referred to as *nonlinear projection trick* is presented in this part. Below, a general kernel method is denoted as KM while its corresponding method in the input space is denoted as M. As inputs to KM, the training and test data X and x respectively, a specific (uncentered) kernel function $\kappa(\cdot,\cdot)$ and a method M are given. Here, M can be any feature extraction method such as PCA, LDA and ICA as well as any classifier/regressor such as SVM/SVR.

Algorithm: Kernel method KM (Input: X, x, $\kappa(\cdot,\cdot)$, method M)

- Training phase

1) Compute the uncentered kernel matrix $\mathcal{K}$ such that $\mathcal{K}_{ij} = \kappa(x_i, x_j)$.
2) Compute the centered kernel K by $K = (I_n - E_n)\mathcal{K}(I_n - E_n)$.
3) Obtain the eigenvalue decomposition of K such that $K = U\Lambda U^T$ where Λ is composed of only the nonzero eigenvalues of K and the columns of U are the corresponding unit eigenvectors of K.
4) Compute Y, the coordinates of $\Phi(X)$, by $Y = \Lambda^{\frac{1}{2}}U^T$.
5) Apply the method M to Y, then it is equivalent to applying the kernel method KM to X, i.e., $M(Y) \equiv KM(X)$.

- Test phase
1) Compute the uncentered kernel vector $\kappa(x)$.
2) Compute the centered kernel vector $k(x)$ by $k(x) = (I_n - E_n)[\kappa(x) - \frac{1}{n}\mathcal{K}\mathbf{1}_n]$.
3) Obtain y, the coordinate of $\phi(x)$, by $y = \Lambda^{-\frac{1}{2}}U^T k(x)$.
4) Apply the method M to y, then it is equivalent to applying the kernel method KM to x, i.e., $M(y) \equiv KM(x)$.

IV. Applications

In this section, the nonlinear kernel trick is applied to several conventional kernel methods such as KPCA, KFD, Kernel SVM and so on. In doing so, we show that different kernel methods can be interpreted in the unified framework of applying conventional linear methods to the reduced dimensional feature space. As a new contribution, we extend PCA-L1 [13] to a kernel version.

A. Kernel PCA (KPCA)

1) Conventional derivation: In the conventional KPCA, the objective is to find a projection vector w in the feature space that minimizes the sum of squared error or equivalently maximizes the variance along with the projection direction. The problem becomes

$$w^\star = \underset{w}{\operatorname{argmax}} ||w^T\Phi(X)||_2^2 = \underset{w}{\operatorname{argmax}}\, w^T\Phi(X)\Phi(X)^T w \quad \text{such that } ||w||_2 = 1. \tag{8}$$

The solution of (8) can be found by the eigenvalue decomposition of $S_f \triangleq \Phi(X)\Phi(X)^T$, the scatter matrix in the feature space, i.e.,

$$S_f w_i = \lambda_i w_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m. \tag{9}$$

Using the fact that w lies in P, w can be written as $w = \Phi(X)\alpha$, and (9) becomes

$$\Phi(X)K\alpha_i = \lambda_i\Phi(X)\alpha_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m. \tag{10}$$

Left multiplying both sides with $\Phi(X)^T$, the problem becomes finding $\{\alpha_i\}_{i=1}^m$ that satisfies $K^2\alpha_i = \lambda_i K\alpha_i$, which further reduces to finding $\{\alpha_i\}_{i=1}^m$ such that

$$K\alpha_i = \lambda_i\alpha_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m. \tag{11}$$

with the constraint $\alpha_i^T K\alpha_i = 1$. The solution $\{\alpha_i\}_{i=1}^m$ is the eigenvectors of K corresponding to the m largest eigenvalues. Now the i-th solution that satisfies $\alpha_i^T K\alpha_i = 1$ becomes

$$\alpha_i = u_i\lambda_i^{-\frac{1}{2}}. \tag{12}$$

Let $W \triangleq [w_1, \cdots, w_m] \in \mathbb{R}^{f\times m}$, then the projection of $\phi(x)$ onto W gives an m-dimensional vector $z = W^T\phi(x)$. Because $w_i = \Phi(X)\alpha_i$, it further reduces to $z = A^T k(x)$ where $A \triangleq [\alpha_1, \cdots, \alpha_m] \in \mathbb{R}^{n\times m}$. From (12), it becomes

$$A = [u_1\lambda_1^{-\frac{1}{2}}, \cdots, u_m\lambda_m^{-\frac{1}{2}}] = U_m\Lambda_m^{-\frac{1}{2}} \tag{13}$$

where $U_m \triangleq [u_1, \cdots, u_m]$ and $\Lambda_m \triangleq \operatorname{diag}(\lambda_1, \cdots, \lambda_m)$ and finally the nonlinear projection of x becomes

$$z = \Lambda_m^{-\frac{1}{2}}U_m^T k(x). \tag{14}$$

2) New derivation: In the new interpretation of kernel methods presented in the previous section, the coordinates of mapped training data $\Phi(X)$ are firstly computed as $Y = \Lambda^{\frac{1}{2}}U^T \in \mathbb{R}^{r\times n}$ from *Theorem 2*. Then, the problem becomes finding maximum variance direction vector in $\mathbb{R}^r$. It is easy to show that the scatter matrix of Y becomes $YY^T = \Lambda$ because the mapped training data Y are centered (*Lemma 2*) and e_i becomes the i-th eigenvector of the scatter matrix and therefore, the i-th principal vector. Here, $e_i \in \mathbb{R}^r$ is a column vector whose elements are identically zero except the i-th element which is one.

Once a new data x is presented, by *Theorem 2*, it is projected to $y = \Lambda^{\frac{1}{2}}U^T k(x)$ and the i-th projection of y becomes

$$z_i = e_i^T y = e_i^T\Lambda^{-\frac{1}{2}}U^T k(x) = \lambda_i^{-\frac{1}{2}}u_i^T k(x). \tag{15}$$

Now $z \triangleq [z_1, \cdots, z_m]^T$, the m-dimensional nonlinear projection of x, becomes

$$z = \Lambda_m^{-\frac{1}{2}}U_m^T k(x). \tag{16}$$

Note that (16) is identical to (14) and we can see that the new interpretation of KPCA is equivalent to the conventional derivation. However, by the new interpretation, we can obtain the exact coordinates of $\Phi(X)$ and the corresponding principal vectors, while conventional derivation cannot find those coordinates.

B. Kernel SVM (KSVM)

1) Conventional derivation: Given some training data $\{(x_i, c_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ and $c_i \in \{-1, 1\}$, the kernel SVM tries to solve the optimization problem:

$$(w^\star, b^\star) = \underset{(w,b)}{\operatorname{argmin}} \frac{1}{2}||w||_2^2 \quad \text{subject to} \quad c_i(w^T\phi(x_i) + b) \geq 1 \quad \forall i = 1, \cdots, n \tag{17}$$

By introducing Lagrange multipliers $\{\alpha_i\}_{i=1}^n$, the above constrained problem can be expressed as the dual form:

$$\alpha^\star = \underset{\alpha}{\operatorname{argmax}} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i\alpha_j c_i c_j k(x_i, x_j) \quad \text{subject to } \sum_{i=1}^n \alpha_i c_i = 0 \text{ and } \alpha_i \geq 0 \quad \forall i = 1, \cdots, n \tag{18}$$

Once we solve the above dual problem, w can be computed thanks to α terms as $w = \sum_{i=1}^{n} \alpha_i c_i \phi(x_i)$ and b can be computed so that it meets the KKT condition $\alpha_i[c_i(w^T\phi(x_i)+b)-1]=0$ for all $i=1,\cdots,n$.

Once a test sample x is presented, it can be classified as $\text{sgn}(w^T\phi(x)+b)$. Note that $w^T\phi(x) = \sum_{i=1}^{n} \alpha_i c_i k(x_i, x)$ and we do not have to find an explicit expression for w in both the training and the test phase.

2) New derivation: In the new derivation, given the training data $\{(x_i, c_i)\}_{i=1}^n$, $X = [x_1, \cdots, x_n]$ is firstly mapped to $Y = \Lambda^{\frac{1}{2}} U^T = [y_1, \cdots, y_n]$ and then a linear SVM is solved for $\{y_i, c_i\}_{i=1}^n$.

The primal problem becomes

$$\begin{aligned} (v^\star, d^\star) &= \underset{(v,d)}{\text{argmin}} \frac{1}{2}||v||_2^2 \\ \text{subject to} \quad & c_i(v^T y_i + d) \geq 1 \quad \forall i = 1, \cdots, n \end{aligned} \tag{19}$$

and the corresponding dual problem is

$$\begin{aligned} \beta^\star &= \underset{\beta}{\text{argmax}} \sum_{i=1}^{n} \beta_i - \frac{1}{2} \sum_{i,j=1}^{n} \beta_i \beta_j c_i c_j y_i^T y_j \\ \text{subject to} \quad & \sum_{i=1}^{n} \beta_i c_i = 0 \text{ and } \beta_i \geq 0 \quad \forall i = 1, \cdots, n \end{aligned} \tag{20}$$

Note that because $k(x_i, x_j) = y_i^T y_j$, (18) and (20) are exactly the same problem and it becomes $\beta = \alpha$ and $d = b$. In this case, v can be explicitly found as $v = \sum_{i=1}^{n} \beta_i c_i y_i$.

Once a test sample x is presented, it is firstly mapped to $y = \Lambda^{-\frac{1}{2}} U^T k(x)$ and then classified as $\text{sgn}(v^T y + d)$.

C. Kernel Fisher discriminant (KFD)

1) Conventional derivation: In the conventional KFD, which is a kernel extension of LDA (linear discriminant analysis), the objective is to find a projection vector w in the feature space such that the ratio of the between-class variance and the within-class variance after the projection is maximized. Given C class training data $\{(x_i, c_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ and $c_i \in \{1, \cdots, C\}$, the problem becomes

$$w^\star = \underset{w}{\text{argmax}} \frac{w^T S_B^\Phi w}{w^T S_W^\Phi w}. \tag{21}$$

Here

$$S_B^\Phi \triangleq \sum_{i=1}^{n} (\phi(x_i) - \bar{\phi})(\phi(x_i) - \bar{\phi})^T \in \mathbb{R}^{f \times f} \tag{22}$$

and

$$S_W^\Phi \triangleq \sum_{i=1}^{n} (\phi(x_i) - \bar{\phi}_{c_i})(\phi(x_i) - \bar{\phi}_{c_i})^T \in \mathbb{R}^{f \times f} \tag{23}$$

are the between scatter and the within scatter matrices in the feature space respectively, where $\bar{\phi} = \sum_{i=1}^{n} \phi(x_i)$ and $\bar{\phi}_c \triangleq \frac{1}{n_c} \sum_{i \in \{j | c_j = c\}} \phi(x_i)$ are the total mean and class mean in the feature space respectively. The term n_c denotes the number of training samples whose class identity is c. As in the previous section, without loss of generality, we assume that $\Phi(X)$ is centerized, i.e., $\bar{\phi} = 0$ and the between scatter matrix becomes $S_B^\Phi = \Phi(X)\Phi(X)^T$. Note that S_B^Φ in (22) can also be written as $S_B^\Phi = \sum_{c=1}^{C} n_c(\bar{\phi}_c - \bar{\phi})(\bar{\phi}_c - \bar{\phi})^T$.

Restricting w in P, w can be written as $w = \Phi(X)\alpha$ and it becomes $w^T\phi(x_i) = \alpha^T k(x_i)$. Using this, (21) becomes

$$\alpha^\star = \underset{\alpha}{\text{argmax}} \frac{\alpha^T S_B^K \alpha}{\alpha^T S_W^K \alpha} \tag{24}$$

where

$$S_B^K = K^2 \in \mathbb{R}^{n \times n} \tag{25}$$

and

$$S_W^K = \sum_{i=1}^{n} (k(x_i) - \bar{k}_{c_i})(k(x_i) - \bar{k}_{c_i})^T \in \mathbb{R}^{n \times n}. \tag{26}$$

Here, $\bar{k}_{c_i} \triangleq \frac{1}{n_c} \sum_{i \in \{j | c_j = c\}} k(x_i)$.

The solution of (24) can be obtained by solving the following generalized eigenvalue decomposition problem

$$S_B^K \alpha_i = \lambda S_W^K \alpha_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m. \tag{27}$$

Once obtaining $\{\alpha_i\}_{i=1}^m$, for a given sample x, an m-dimensional nonlinear feature vector $z \in \mathbb{R}^m$ is obtained by the dot product of $W \triangleq [w_1, \cdots, w_m] = \Phi(X)[\alpha_1, \cdots, \alpha_m]$ and $\phi(x)$, i.e., $z = W^T\phi(x) = A^T k(x)$, where $A \triangleq [\alpha_1, \cdots, \alpha_m] \in \mathbb{R}^{n \times m}$.

2) New derivation: In the new derivation of KFD, for given training data X, a (centerized) kernel matrix K is computed, then its eigenvalue decomposition $K = U\Lambda U^T$ is utilized to get the nonlinear mapping $Y = \Lambda^{\frac{1}{2}} U^T$. Then, its within scatter and between scatter matrices are computed as

$$S_W^Y = \sum_{i=1}^{n} (y_i - \bar{y}_{c_i})(y_i - \bar{y}_{c_i})^T \in \mathbb{R}^{r \times r} \tag{28}$$

and

$$S_B^Y = \sum_{c=1}^{C} n_c(\bar{y}_c - \bar{y})(\bar{y}_c - \bar{y})^T = \sum_{i=1}^{n} y_i y_i^T = YY^T \in \mathbb{R}^{r \times r} \tag{29}$$

respectively, where $\bar{y}$ and $\bar{y}_c$ are the total mean and the class mean of Y. Recall that $\bar{y} = 0$ from *Lemma 3*.

Applying LDA directly to Y, the objective becomes finding $\beta \in \mathbb{R}^r$ such that

$$\beta^\star = \underset{\beta}{\text{argmax}} \frac{\beta^T S_B^Y \beta}{\beta^T S_W^Y \beta}. \tag{30}$$

Noting that $y_i = \Lambda^{-\frac{1}{2}} U^T k(x_i)$ (from *Theorem 2*), the numerator becomes

$$\beta^T S_B^Y \beta = \beta^T \Lambda^{-\frac{1}{2}} U^T K K U \Lambda^{-\frac{1}{2}} \beta = \alpha^T K^2 \alpha \tag{31}$$

where $\alpha = U\Lambda^{-\frac{1}{2}}\beta$. Likewise the denominator becomes

$$\beta^T S_W^Y \beta = \alpha^T \left(\sum_{i=1}^{n} (k(x_i) - \bar{k}_{c_i})(k(x_i) - \bar{k}_{c_i})^T \right) \alpha \tag{32}$$

Replacing (31) and (32) into (30), we can see the problem becomes identical to (24). Furthermore, because $\alpha = U\Lambda^{-\frac{1}{2}}\beta$, if we multiply both sides with $\Lambda^{\frac{1}{2}} U^T$ to the left, $\beta = \Lambda^{\frac{1}{2}} U^T \alpha = Y\alpha$ holds, which was anticipated in *Lemma 5*.

After obtaining $\{\beta_i\}_{i=1}^m$, z, the nonlinear projection of x can be obtained by the dot product of $B \triangleq [\beta_1, \cdots, \beta_m]$ and $y = \Lambda^{-\frac{1}{2}} U^T k(x)$. Then $z = B^T \Lambda^{-\frac{1}{2}} U^T k(x) = A^T k(x)$.

D. Kernel PCA-L1 (KPCA-L1)

Until now, we presented examples of application of the new kernel interpretation to the existing kernel methods such as KPCA, KSVM and KFD. In this subsection, we show that this interpretation of kernel methods is applicable to algorithms which do not rely on the dot product, thus to which *kernel trick* is not applicable.

In [13], PCA-L1, a robust version of PCA which maximizes the dispersion of the projection was introduced. Given training data X, the objective of PCA-L1 is to find a set of projection vectors w's that solve the following optimization problem:

$$w^\star = \underset{w}{\operatorname{argmax}} ||w^T X||_1 = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} |w^T x_i| \quad \text{subject to } ||w||_2 = 1. \tag{33}$$

For information, the algorithm PCA-L1 is presented below.

Algorithm: PCA-L1 (Input: X)

1) Initialization: Pick any $w(0)$. Set $w(0) \leftarrow w(0)/||w(0)||_2$ and $t = 0$.
2) Polarity check: For all $i \in \{1, \cdots, n\}$, if $w^T(t)x_i < 0$, $p_i(t) = -1$, otherwise $p_i(t) = 1$.
3) Flipping and maximization: Set $t \leftarrow t + 1$ and $w(t) = \sum_{i=1}^{n} p_i(t-1)x_i$. Set $w(t) \leftarrow w(t)/||w(t)||_2$.
4) Convergence check:
   a. If $w(t) \neq w(t-1)$, go to Step 2.
   b. Else if there exists i such that $w^T(t)x_i = 0$, set $w(t) \leftarrow (w(t) + \Delta w)/||w(t) + \Delta w||_2$ and go to Step 2. Here, Δw is a small nonzero random vector.
   c. Otherwise, set $w^\star = w(t)$ and stop.

This problem formulation can be easily extended to a kernel version by replacing X with $\Phi(X)$ and the problem becomes

$$w^\star = \underset{w}{\operatorname{argmax}} ||w^T \Phi(X)||_1 = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} |w^T \phi(x_i)| \quad \text{subject to } ||w||_2 = 1. \tag{34}$$

Note that the kernel trick cannot be applicable to the above problem because the problem is not formulated based on $L2$-norm.

By *Lemma 1*, w can be expressed using the bases Π as $w = \Pi\beta$ and from *Theorem 2* $w^T\Phi(X)$ can be expressed as $\beta^T Y$. Furthermore, the constraint $||w||_2 = 1$ becomes $||\beta||_2 = 1$ because the columns of Π are orthonormal. Therefore, (34) becomes

$$\beta^\star = \underset{\beta}{\operatorname{argmax}} ||\beta^T Y||_1 \quad \text{subject to} \quad ||\beta||_2 = 1. \tag{35}$$

Because the form of (35) is exactly the same as (33), the conventional PCA-L1 algorithm can directly be applied to Y to obtain the KPCA-L1 algorithm.

Note that this is a simple example of making a kernel version of an algorithm by using the nonlinear projection trick. The point is that this trick can be applied to an arbitrary method as shown in Section III.

V. Conclusions

This paper proposed a kernel method that directly maps the input space to the feature space by deriving an exact coordinates of the mapped input data. In the conventional kernel methods, the *kernel trick* is used to avoid direct nonlinear mapping of input data onto a high (could be infinite) dimensional feature space. Instead, the dot products of the mapped data are converted to kernel functions in the problems to be solved, making the problems solvable without ever having to explicitly map the original input data into a kernel space. However, the applicability of the kernel trick is restricted to problems where the mapped data appear in the problem only in the form of dot products. Therefore, arbitrary methods that do not utilize the dot product (or L_2-norm) could not be extended to kernel methods.

In this paper, based on the fact that the effective dimensionality of a kernel space is less than the number of training samples, an explicit mapping of the input data into a reduced kernel space is proposed. This is obtained by eigenvalue decomposition of the kernel matrix. With the proposed *nonlinear projection trick*, the applicability of the kernel methods is widened to arbitrary methods that can be applied in the input space. In other words, not only the conventional kernel methods based on L_2-norm optimization such as KPCA and kernel SVM, but the methods utilizing other objective functions can also be extended to kernel versions. The equivalence between the kernel trick and the nonlinear projection trick is shown in several conventional kernel methods such as KPCA, KSVM and KFD. In addition, we extend PCA-L1 that utilize L_1-norm into a kernel version and show the effectiveness of the proposed approach.

References

[1] B. Schölkopf, A.J. Somla, and K.R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[2] B. Schölkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, 2002.

[3] R. Rosipal, M. Girolami, and L.J. Trejo, "Kernel PCA for feature extraction and de-noising in non-linear regression," *Neural Computing & Applications*, vol. 10, pp. 231–243, 2001.

[4] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K.R. Müller, "Fisher discriminant analysis with kernels," in *Proceedings of the 1999 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing IX*, 1999, pp. 41–48.

[5] J. Yang, A.F. Frangi, J.-Y. Yang, D. Zhang, and Z. Jin, "KPCA plus LDA: A complete kernel fisher discriminant framework for feature extraction and recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 230–244, Feb. 2005.

[6] G. Baudat and R. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[7] M. H. Nguyen and F. De la Torre, "Robust kernel principal component analysis," Tech. Rep. Paper 144, Robotics Institute, 2008, http://repository.cmu.edu/robitics/144.

[8] N. Kwak, "Kernel discriminant analysis for regression problems," *Pattern Recognition*, vol. 45, no. 5, pp. 2019–2031, May 2012.

[9] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, March 2007.

[10] T. Hofmann, B. Schölkopf, and A.J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[11] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Ratsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*. 1999, pp. 536–542, MIT Press.

[12] T.-Y. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Trans. on Neural Networks*, vol. 15, no. 6, pp. 1517–1525, Nov. 2004.
[13] N. Kwak, "Principal component analysis based on l1 norm maximization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, Sep. 2008.
[14] K. Fukunaga, "Intrinsic dimensionality extraction," *Handbook of Statistics*, vol. 2, pp. 347 – 360, 1982.
[15] F. Camastra, "Data dimensionality estimation methods: a survey," *Pattern Recognition*, vol. 36, pp. 2945–2954, 2003.