An Analysis of Transformation on Non-Positive Semidefinite Similarity Matrix for Kernel Machines

Gang Wu GWU@ECE.UCSB.EDU
Edward Y. Chang ECHANG@ECE.UCSB.EDU
Zhihua Zhang ZHZHANG@ECE.UCSB.EDU
Department of Electrical & Computer Engineering, University of California, Santa Barbara, CA 93106 USA

Abstract

Many emerging applications formulate non-positive semidefinite similarity matrices, and hence cannot fit into the framework of kernel machines. A popular approach to this problem is to transform the spectrum of the similarity matrix so as to generate a positive semidefinite kernel matrix. In this paper, we present an analytical framework to explore four representative transformation methods: denoise, flip, diffusion, and shift. Theoretically, we interpret each transformation and analyze its influence on classification using kernel machines. Moreover, when situations arise where the test data are not available during transformation, we propose an efficient algorithm to address the problem of updating the cross-similarity matrix between test and training data. Extensive experiments have been conducted to evaluate the performance of these methods on several real-world (dis)similarity matrices with semantic meanings.

1. Introduction

Kernel methods work by embedding data instances into a feature space $\mathcal{F}$, and searching for linear relations in such a space. This embedding is defined implicitly through a kernel function $K(\mathbf{x}, \mathbf{x}')$. Evaluating the kernel function on all pairs of data instances produces a symmetric, positive semidefinite (*psd*) *kernel matrix* $\mathbf{K}$. The *psd* property of a kernel matrix (the Mercer's condition (Vapnik, 1995)) is required to ensure that there exits a Reproducing Kernel Hilbert Space (RKKS) where a convex optimization formulation can be derived so as to yield an optimal solution.

Technical Report. Department of Electrical & Computer Engineering, University of California, Santa Barbara. June, 2005

In practice, however, similarity matrices generated by many emerging applications for characterizing (dis)similarity between data instances can violate the *psd* property. For instance, some popular functions in bioinformatics for measuring pair-wise similarity between DNA and protein sequences produce non-psd (or indefinite) kernel matrices. Example functions are dynamic time warping (DTW) (Shimodaira et al., 2001), the Smith-Waterman algorithm, or BLAST (Altschul et al., 1990). In image/video retrieval, the earth-mover's distance function (Rubner et al., 2000) and dynamic partial function (Qamra et al., 2005), which have proven to be effective for quantifying perceptual similarity, are both non-metric. In handwritten-digit recognition, the distance of two characters can effectively be calculated using a tangent distance function to overcome the phenomenon of translation and rotation (Simard et al., 1993). But, the kernel formulated from tangent distance might not be psd (Hassdonk & Keysers, 2002). Another example is the sigmoid kernel $K(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^T\mathbf{x}' + r)$ of Neural Networks, which violates Mercer's condition for various values of its parameters (Vapnik, 1995). Many other examples exist.

Several methods have been proposed to address the problem of learning with a non-*psd* similarity matrix in kernel machines. Let us use Support Vector Machines (SVMs) as an example. These methods can be divided into two approaches: *algorithmic* and *spectrum-transformation*. The algorithmic approach still uses the non-psd similarity matrix as a kernel, but it changes the formulation of SVMs or employs heuristics to find local solutions. For example, Lin and Lin (2003) propose an SMO-type decomposition method to solve the non-convex dual formulation in SVMs, which can converge to some stationary points for the sigmoid kernel. This method implies that there still exits a corresponding RKHS so that the formulation of SVMs and its derivation are valid. However, if the kernel does not satisfy the Mercer's condition, such an RKHS does not exist, and the associated Representer theorem (Kimeldorf & Wahba, 1970) might collapse. Hence, in the non-RKHS, we cannot even formulate SVMs and derive its dual formulation using the traditional mathematical tools, which are only valid in an RKHS. Haasdonk (2004) argues that learning with indefinite kernel is equivalent to minimizing the distance of two convex hulls in a pseudo-Euclidean space formed by the kernel. However, it still implies that the Representer theorem is satisfied in the pseudo-Euclidean space, which might not be the case. Therefore, Ong et al. (2004) propose to associate the non-psd kernels with a Reproducing Kernel Kreǐn Space (RKKS), where a general Representer theorem exists and a regularized risk functional can be defined. Since the kernel is indefinite, some solutions could be found by stabilizing the risk function, instead of minimizing it. In an RKKS space, SVMs can be re-formulated. However, computationally intensive tools will have to be employed to find solutions.

Instead of inventing new algorithms, the *spectrum-transformation* approach embarks on changing the spectrum of an indefinite kernel matrix so as to generate a *psd* one. We believe this approach to be more attractive because it keeps SVMs formulated in an RKHS and preserves the problem as a convex optimization. In this paper, we first briefly propose an analytical framework for interpreting different spectrum-transformation methods. We analyze four representative methods including denoise, flip, diffusion, and shift. Our analytical framework points out that a successful transformation should consider semantics preserving, computational cost, and generalization capability. We show that the shift method can be tied to the regularization term in SVMs, and therefore its computation time on spectral decomposition could be somehow replaced by the cross-validation procedure on tuning hyperparameter in SVMs. We also show that the first three transformation methods, denoise, flip and diffusion, suffer from the generalization problem when situations arise where the test data are not available during transformation. We then propose an efficient incremental algorithm to address the problem of updating the cross-similarity matrix between test and training data. Empirical results on several datasets are provided to compare transformation and generalization methods in Section 4.

2. Analytical Framework

Let us consider a binary-class supervised classification problem with training data $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$. Suppose we are given pairwise similarity of the instances in kernel matrix $\mathbf{K}$, which can be indefinite. We analyze in this section four transformation methods that change the spectrum of $\mathbf{K}$ to form a new positive semidefinite $\tilde{\mathbf{K}}$.

We first write matrix $\mathbf{K}$ in terms of its $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$ is the diagonal matrix of the eigenvalues λ_i's, and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]$ the orthogonal matrix of corresponding eigenvectors. For convenience, we assume that λ_i's are sorted in a descending order such that $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_N$. The methods belonging to the *spectrum-transformation* approach can be regarded as applying a mapping function $f(\cdot)$ to transform the eigenvalues λ_i, or $\tilde{\lambda}_i = f(\lambda_i)$, so that all $\tilde{\lambda}_i$'s are non-negative. The induced matrix $\tilde{\mathbf{K}} = \mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{U}^T$ is thus a *psd* kernel matrix. Using function $f(\cdot)$, we can interpret four representative spectrum-transformation methods: denoise, flip, diffusion, and shift.

2.1. Denoise: $f(\lambda) = \max(0, \lambda)$

The works of (Graepel et al., 1999) and (Pekalska et al., 2002) treat all negative eigenvalues as noise and replace them with zero. This method seems to be arbitrary, but it has a solid theoretical backing. Let us consider the problem of approximating the non-*psd* kernel $\mathbf{K}$ with a *psd* one $\tilde{\mathbf{K}}$ in Frobenius norm (Golub & Loan, 1996). The problem can be formulated as a convex optimization problem as follows:

$$\begin{aligned} \min \quad & e(\tilde{\mathbf{K}}) = \|\mathbf{K} - \tilde{\mathbf{K}}\|_F^2 \\ s.t. \quad & \tilde{\mathbf{K}} \succeq 0. \end{aligned} \qquad (1)$$

According to the Wielandt-Hoffman theorem (Golub & Loan, 1996), the solution to (1) can be calculated as $\tilde{\mathbf{K}} = \mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{U}^T$, where $\tilde{\mathbf{\Lambda}} = \mathrm{diag}(\max(0, \lambda_1), \max(0, \lambda_2), \ldots, \max(0, \lambda_N))$. In other words, the best approximation of a *psd* matrix to a non-*psd* matrix is achieved by neglecting all its negative eigenvalues.

2.2. Flip: $f(\lambda) = |\lambda|$

This transformation (Graepel et al., 1999) flips the sign of negative eigenvalues in $\mathbf{K}$ so as to form a *psd* kernel matrix $\tilde{\mathbf{K}}$. In (Graepel et al., 1999), the authors explain this transformation by perceiving that the data are embedded in a pseudo-Euclidean space $\Re^{(N^+, N^-)}$, where N^+ and N^- are the number of positive and negative λ_i's of $\mathbf{K}$, and $N^+ + N^- = N$. In the space

$\Re^{(N^+,N^-)}$, the distance between $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ is calculated as $\sqrt{(\Phi(\mathbf{x}_i)-\Phi(\mathbf{x}_j))^T\mathbf{M}(\Phi(\mathbf{x}_i)-\Phi(\mathbf{x}_j))}$, where $\mathbf{M} = \text{diag}(\mathbf{I}_{N^+}, -\mathbf{I}_{N^-})$. Therefore, the $\tilde{\mathbf{K}}$ can be considered as an inner product matrix in the pseudo-Euclidean space.

We can explain the flip method from the perspective of SVD. Let the SVD decomposition of $\mathbf{K}$ be

$$\mathbf{U}\text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_N)\mathbf{V}^T,$$

where $\mathbf{U}^T\mathbf{U} = \mathbf{I}_N$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_N$, and σ_i's are permuted so that the $\mathbf{U}$ above is same with $\mathbf{U}$ in $\mathbf{K}$'s spectral decomposition. In the other hand, from $\mathbf{K}$'s spectral decomposition, we have $\mathbf{K}\mathbf{K}^T = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^T$, where $\mathbf{\Lambda}^2 = \text{diag}(\lambda_1^2, \lambda_2^2, \cdots, \lambda_N^2)$ is the eigenvalue matrix of $\mathbf{K}\mathbf{K}^T$. Since the singular value σ_i is the square root of the corresponding eigenvalue of $\mathbf{K}\mathbf{K}^T$ (Horn & Johnson, 1985), $\sigma_i = \sqrt{\lambda_i^2} = |\lambda_i|$. The fact means the flip method actually approximates the eigenvalues of $\mathbf{K}$ by its corresponding singular values. The induced $\tilde{\mathbf{K}} = \mathbf{U}\text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_N)\mathbf{U}^T$ is then used as kernel matrix in kernel methods.

2.3. Diffusion: $f(\lambda) = e^{\beta\lambda}$

The main idea of a diffusion kernel (Kondor & Lafferty, 2002) is to consider data distribution when computing pairwise similarity. Let the mapping function $f(\lambda)$ be $e^{\beta\lambda}$ with $\beta > 0$, we can rewrite $\tilde{\mathbf{K}} = \mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{U}^T$ as

$$\begin{aligned}\tilde{\mathbf{K}} &= \mathbf{U}\text{diag}(e^{\beta\lambda_1}, e^{\beta\lambda_2}, \cdots, e^{\beta\lambda_N})\mathbf{U}^T \\ &= \mathbf{U}\left[\sum_{n=0}^{\infty}\beta^n \text{diag}(\frac{\lambda_1^n}{n!}, \frac{\lambda_2^n}{n!}, \cdots, \frac{\lambda_N^n}{n!})\right]\mathbf{U}^T \\ &= \sum_{n=0}^{\infty}\frac{\beta^n}{n!}\mathbf{U}\mathbf{\Lambda}^n\mathbf{U}^T = \sum_{n=0}^{\infty}\frac{\beta^n}{n!}\mathbf{K}^n = e^{\beta\mathbf{K}}.\end{aligned} \tag{2}$$

This is exactly the diffusion kernel proposed in (Kondor & Lafferty, 2002). Let us consider the training dataset $\mathcal{X}$ in an undirected graph Γ with each instance $\mathbf{x}_i$ as a vertex (V) and k_{ij} as the weight of an edge (E) connecting $\mathbf{x}_i$ and $\mathbf{x}_j$. The i,j-th component of the matrix $\mathbf{K}^n$ is

$$(\mathbf{K}^n)_{ij} = \sum_{r_1, r_2, \cdots, r_{n-1}=0}^{N} k_{ir_1}k_{r_1r_2}\cdots k_{r_{n-1}j}.$$

When we interpret the weight of an edge in Γ as probability, $(\mathbf{K}^n)_{ij}$ is the sum of all probabilities with which the information transfers from $\mathbf{x}_i$ to $\mathbf{x}_j$ along an n-step random walk (Kondor & Lafferty, 2002), as illustrated in Figure 1. Thus, the mapping of $f(\lambda) = e^{\beta\lambda}$ is equivalent to considering the data distribution in $\mathcal{X}$.

This transformation can be helpful in an unsupervised learning setting because clustering is based on the assumption that data instances in the same neighborhood tend to belong to the same cluster (Chapelle et al., 2003). However, in a supervised setting when data instances in the neighborhood might belong to different classes, the diffusion method is inappropriate. Figure 1 illustrates how the diffusion method works and why it can be counter-productive.

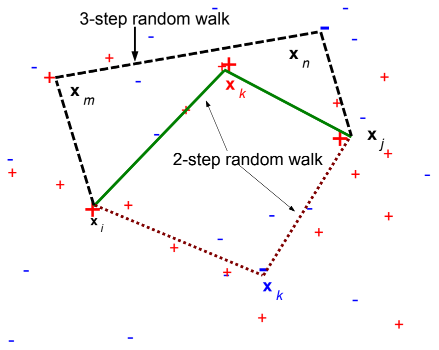


Figure 1. Illustration of the Diffusing Method. This simple example illustrates how other data instances ($\mathbf{x}_k$ in 2-step random walk, and $\mathbf{x}_m$ and $\mathbf{x}_n$ in 3-step random walk) affects the calculation of similarity score k_{ij}. In this example, $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same class label, and $\mathbf{x}_k$ (the one with a different label) is located in a 2-step random walk from $\mathbf{x}_i$ to $\mathbf{x}_j$. Diffusion kernel considers $\mathbf{x}_k$ when computing the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. But when $\mathbf{x}_k$ belongs to a different class, this diffusion process is not helpful. A similar case is found in 3-step random walk.

2.4. Shift: $f(\lambda) = \lambda + \eta$

Shift adds a constant to all eigenvalues and has been used in clustering to transform a non-metric distance matrix (Roth et al., 2003). Suppose we use SVMs with penalizing 1-norm training error $C_1|\boldsymbol{\xi}|$ (denoted as 1-norm SVMs). Its dual formulation is as follows:

$$\begin{aligned}\min \quad & \mathcal{D}_1(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ s.t. \quad & \mathbf{0} \le \boldsymbol{\alpha} \le C_1\mathbf{1}, \quad \mathbf{y}^T\boldsymbol{\alpha} = 0,\end{aligned} \tag{3}$$

where $\boldsymbol{\alpha}$ is the vector of Lagrangian multipliers, $\mathbf{y}$ is the label vector, C_1 is the hyperparameter, and $\mathbf{Q} = [y_iy_jk_{ij}]$ implies a kernel matrix.

Let us start from a general case that $\lambda \ge 0$. Suppose we use mapping-function $f(\lambda) = \lambda + \eta$. We obtain the following equation:

$$\tilde{\mathbf{K}} = \mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{U}^T = \mathbf{U}(\mathbf{\Lambda} + \eta\mathbf{I})\mathbf{U}^T = \mathbf{K} + \eta\mathbf{I}.$$

Since $\mathbf{Q} = \mathbf{y}\mathbf{y}^T \odot \mathbf{K}$, where $\odot$ denotes the Hadamard product (Horn & Johnson, 1985), we have

$$\tilde{\mathbf{Q}} = \mathbf{y}\mathbf{y}^T \odot (\mathbf{K} + \eta\mathbf{I}) = \mathbf{Q} + \eta\mathbf{I}. \tag{4}$$

Substituting Eqn. 4 into the dual formulation (3), we achieve a new dual problem of SVMs as follows:

$$\begin{aligned}\tilde{\mathcal{D}}_1(\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{\alpha}^T(\mathbf{Q}+\eta\mathbf{I})\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \qquad (5)\\ &= \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} + \frac{\eta}{2}\boldsymbol{\alpha}^T\mathbf{I}\boldsymbol{\alpha}\\ &= \mathcal{D}_1(\boldsymbol{\alpha}) + \frac{\eta}{2}\|\boldsymbol{\alpha}\|^2, \text{ where } \mathbf{0} \le \boldsymbol{\alpha} \le C_1\mathbf{1}.\end{aligned}$$

Using the same procedure above, we can derive the dual formulation of SVMs with penalizing 2-norm training error $C_2\|\boldsymbol{\xi}\|^2$ (denoted as 2-norm SVMs) as

$$\begin{aligned}\tilde{\mathcal{D}}_2(\boldsymbol{\alpha}) &= \mathcal{D}_2(\boldsymbol{\alpha}) + \frac{\eta}{2}\|\boldsymbol{\alpha}\|^2 \qquad (6)\\ &= \mathcal{D}_1(\boldsymbol{\alpha}) + \left(C_2 + \frac{\eta}{2}\right)\|\boldsymbol{\alpha}\|^2, \text{ where } \mathbf{0} \le \boldsymbol{\alpha} \le C_2\mathbf{1}.\end{aligned}$$

From Equations 5 and 6, we can see that minimizing the dual formulation after shift is equivalent to minimizing both the dual formulation before shift and the 2-norm of the multiplier vector $\boldsymbol{\alpha}$. The shift constant η can be considered as a regularizer that penalizes the length of $\boldsymbol{\alpha}$. Recall that in SVMs, the non-support vectors have α_i as zero, support vectors have $\alpha_i \in (0, C]$. Assigning a large regularizer η tends to decrease the α_i of each training instance, such that more instances would be treated as non-support vectors. We therefore could expect to achieve a large margin in SVMs.

The smallest η to make $\tilde{\mathbf{K}}$ to be psd is $|\lambda_N|$. The work of (Roth et al., 2003) shows that it is not necessary to choose an η larger than $|\lambda_N|$ for clustering applications. However, this conclusion may not be appropriate for SVMs. The above analysis about formulation (5) and (6) tells us that once we shift λ by $|\lambda_N|$, further increasing η attempts to increase the margin of SVMs. In other words, η plays a role similar to the hyper-parameter C_1 or C_2 in regularizing SVMs.

The following lemma proves a relation on the roles played by η and the hyperparameter in SVMs.

Lemma 1 *If $C_1 = C_2$ and $\eta = 2C_2$, then optimizing 1-norm SVMs after applying* shift *with η becomes optimizing 2-norm SVMs without applying* shift.

Proof. It can be directly derived from (5) and (6) ■

The shift transformation preserves semantic in the kernel matrix since it does not change the off-diagonal similarity scores. The η parameter provides a principled way for trading off bias and variance in SVMs.

2.5. Discussion

When choosing a method to transform a kernel, three issues should be considered: semantics preservation, computational cost, and generalization capability. Shift seems to be a better choice compared to denoise, flip, and diffusion when considering these three factors.

Regarding semantics, denoise, flip, and diffusion can potentially change the semantics of similarity for the target application. For instance, denoise regards negative eigenvalues as noise, but the fact that a non-metric distance function has been proven to work better means that the negative eigenvalues are not merely noise, and they can carry valuable information. Therefore, although an optimal solution can be obtained after the transformation procedure, we cannot be sure valuable information has not been lost in the transformation process. Contrary to these three transformations, shift provides a principled way for trading semantic preservation and classification performance through cross validation.

Regarding computational cost, the first three methods suffer from high computational intensity. This high cost derives from the fact that all these three methods based on spectral decomposition have to deal with an N-by-N kernel matrix $\mathbf{K}$. Therefore, they require $O(N^3)$ computational cost. The diffusion method incurs an even higher cost due to the exponential operation. In contrast, the shift method needs only to estimate an η which is at least larger than $|\lambda_N|$. Recall the eigenvalue λ_i has the following properties (Golub & Loan, 1996):

$$\|\mathbf{K}\|_F^2 = \sum_{i=1}^N \lambda_i^2, \mathrm{tr}(\mathbf{K}) = \sum_{i=1}^N \lambda_i, \|\mathbf{K}\|_2 \ge \max(\frac{\|\mathbf{K}\|_{col}}{\sqrt{N}}, \frac{\|\mathbf{K}\|_{row}}{\sqrt{N}}),$$

where $\|\mathbf{K}\|_{col} = \max_{1\le j\le N}\sum_{i=1}^N |k_{ij}|$, and $\|\mathbf{K}\|_{row} = \max_{1\le i\le N}\sum_{j=1}^N |k_{ij}|$. We can estimate the smallest eigenvalue λ_N in the following:

$$-\sqrt{\|\mathbf{K}\|_F^2 - \frac{\max(\|\mathbf{K}\|_{col}^2, \|\mathbf{K}\|_{row}^2)}{N}} \le \lambda_N \le \max(0, \frac{\mathrm{tr}(\mathbf{K})}{N}),$$

where we assume $\lambda_N \le 0$.

In shift, we actually do not need an exact approximation to $|\lambda_N|$ since η can be regarded as a regularizer like C in SVMs, as what we have discussed in Section 2.4. Even we calculate an exact $|\lambda_N|$ for η and train SVMs with new $\tilde{\mathbf{K}}$, tuning the hyperparameter C in SVMs indirectly changes η. Therefore, one who is used to performing cross validation on C can fix C, and perform that on η. Since cross validation is a procedure that must be performed to tune C, regardless of whether a spectrum transformation is performed, shift does not incur any additional computational cost.

Finally, regarding generalization, the first three methods cannot effectively deal with classifying unseen test instances (the test instances must be present at the training time). Given a set of test instances in

$\mathcal{X}^{(t)} = \{\mathbf{x}_1^{(t)}, \cdots, \mathbf{x}_M^{(t)}\}$ and a prior kernel $\mathbf{K}$, a matrix with $\mathcal{X}$ and $\mathcal{X}^{(t)}$ is formed as follows:

$$\mathbf{K}_c = \begin{pmatrix} \tilde{\mathbf{K}}_{11} & \mathbf{K}_{21}^T \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{U}\tilde{\Lambda}\mathbf{U}^T & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix} \quad (7)$$

where $\tilde{\mathbf{K}}_{11} = \mathbf{U}\tilde{\Lambda}\mathbf{U}^T$ is an $N \times N$ kernel matrix after spectrum transformation, formed by training set $\mathcal{X}$. After training, a set of test instances in $\mathcal{X}_t$ comes with a self-similarity matrix $\mathbf{K}_{22}$ and a cross-similarity matrix $\mathbf{K}_{21}$ with $\mathcal{X}$. Since only $\mathbf{K}_{11}$ has been transformed to $\tilde{\mathbf{K}}_{11}$ and the rest have not, $\mathbf{K}_c$ might not be psd. We hope that the transformation on $\mathbf{K}_{11}$ can be "generalized" to the cross-similarity matrix $\mathbf{K}_{21}$ for making class predictions, i.e., sgn $\left(\tilde{\mathbf{K}}_{21}\mathbf{Y}\boldsymbol{\alpha}\right)$, where $\mathbf{Y} = \text{diag}(y_1, y_2, \cdots, y_N)$.

One method for achieving this goal is to recompute the spectral decomposition of $\mathbf{K}_c$. Obviously, the computation cost is very high. An alternative is to project the unseen test instances $\mathbf{X}^{(t)}$ onto the eigenspace spanned by $\mathbf{U}$ of $\tilde{\mathbf{K}}_{11}$. According to KPCA (Schölkopf et al., 1998) and MDS (Cox & Cox, 2001), the training data can be considered as being embedded in a spanned eigenvector space $\tilde{\mathbf{X}} = \mathbf{U}\tilde{\Lambda}^{\frac{1}{2}}$, where $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_N]^T$. Given a set of unseen instances $\mathbf{X}^{(t)}$ with a prior cross-similarity matrix $\mathbf{K}_{21}$, we first map $\mathbf{X}^{(t)}$ into $\{\mathbf{U}\}$ as $\tilde{\mathbf{X}}^{(t)}$, whose coordinates in the space $\{\mathbf{U}\}$ is $\tilde{\mathbf{X}}^{(t)} = \mathbf{K}_{21}\mathbf{U}\tilde{\Lambda}^{-\frac{1}{2}}$. Therefore, the new cross-similarity matrix $\tilde{\mathbf{K}}_{21}$ between the embedded $\tilde{\mathbf{X}}^{(t)}$ and $\tilde{\mathbf{X}}$ is calculated as

$$\tilde{\mathbf{K}}_{21} = \tilde{\mathbf{X}}^{(t)}\tilde{\mathbf{X}}^T = \mathbf{K}_{21}\mathbf{U}\tilde{\Lambda}^{-\frac{1}{2}}(\tilde{\Lambda}^{\frac{1}{2}})^T\mathbf{U}^T = \mathbf{K}_{21}.$$

Surprisingly, we can see that the generalization method based on embedding actually cannot adapt to the unseen data. This problem can be explained by MDS: MDS finds a set of lower-dimensional data in an Euclidean space whose pairwise (dis)similarity is preserved (Cox & Cox, 2001) ($\tilde{\mathbf{K}}_{21} = \mathbf{K}_{21}$).

The shift method does not change the off-diagonal items of kernel matrix , $\tilde{\mathbf{K}}_{21}$ thus equals $\mathbf{K}_{21}$. One minor problem is that $\mathbf{K}_c$ might become non-psd when $\mathbf{K}_{22}$ is introduced. Therefore, $\tilde{\mathbf{K}}_{11}$ needs a new η to adapt to such a change. However, unless we need to retrain $\mathcal{X}$ with the updated $\mathbf{K}_{11}$, we can use the old $\boldsymbol{\alpha}$ for test as an approximation.

3. Updating Cross-Similarity Matrix

In this section, we propose an incremental way to efficiently update the cross-similarity matrix $\mathbf{K}_{21}$. We first discuss the case that all test instances are present before training, followed by the case that the test instances arrive individually after training.

Suppose the test data $\mathbf{x}_j^{(t)}$'s $(j = 1, \cdots, M)$ are available before training. We can rewrite the big matrix in (7) as follows:

$$\mathbf{K}_j = \begin{pmatrix} \mathbf{K}_{j-1} & \mathbf{d}_j \\ \mathbf{d}_j^T & k_j \end{pmatrix} = \begin{pmatrix} \mathbf{U}_{j-1}\Lambda_{j-1}\mathbf{U}_{j-1}^T & \mathbf{d}_j \\ \mathbf{d}_j^T & k_j \end{pmatrix}, \quad (8)$$

where $\mathbf{K}_{j-1}$ is an $(N + j - 1) \times (N + j - 1)$ matrix before adding $\mathbf{x}_j^{(t)}$, k_j is the self-similarity score of $\mathbf{x}_j^{(t)}$, and $\mathbf{d}_j$ is an $(N + j - 1) \times 1$ cross-similarity vector. Based on the last spectral decomposition $\mathbf{K}_{j-1} = \mathbf{U}_{j-1}\Lambda_{j-1}\mathbf{U}_{j-1}^T$, we attempt to calculate $\mathbf{K}_j = \mathbf{U}_j\Lambda_j\mathbf{U}_j^T$ without directly decomposing it.

Decompose $\mathbf{K}_j$ in Eqn. 8 into the multiplication of several blocks matrices as follows:

$$\begin{pmatrix} \mathbf{U}_{j-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \Lambda_{j-1} & \mathbf{U}_{j-1}^T\mathbf{d}_j \\ \mathbf{d}_j^T\mathbf{U}_{j-1} & k_j \end{pmatrix} \begin{pmatrix} \mathbf{U}_{j-1}^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}, \quad (9)$$

where the middle block matrix is a *bordered diagonal* (or *arrowhead*) matrix (Wilkinson, 1988), whose spectral decomposition can be quickly calculated in $O(N + j)$ using the *fast multipole method* (Carrier et al., 1988; Gu & Eisenstat, 1995). Denote its decomposition as $\mathbf{U}_a\Lambda_a\mathbf{U}_a^T$ and substitute it into (9),

$$\mathbf{K}_j = \begin{pmatrix} \mathbf{U}_{j-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \mathbf{U}_a\Lambda_a\mathbf{U}_a^T \begin{pmatrix} \mathbf{U}_{j-1}^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}. \quad (10)$$

It is easy to verify that $\begin{pmatrix} \mathbf{U}_{j-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \mathbf{U}_a$ is an orthogonal matrix. Therefore,

$$\mathbf{U}_j = \begin{pmatrix} \mathbf{U}_{j-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \mathbf{U}_a \quad \text{and} \quad \boldsymbol{\Lambda}_j = \boldsymbol{\Lambda}_a.$$

Once $\mathbf{K}_j$ is calculated, substitute it into (8) and repeat the iteration until last test data $\mathbf{x}_M^{(t)}$ is done. Once $\mathbf{x}_M^{(t)}$ arrives, $\mathbf{K}_M$ becomes $\mathbf{K}_c$ in Eqn. 7 before applying sepectrum transformation. We can then transform $\boldsymbol{\Lambda}_M$ by using the methods discussed in Section 2 to generate a new cross-similarity matrix $\tilde{\mathbf{K}}_{21}$. The whole computational cost is less than $O(M(N+M))$, but the calculated $\mathbf{U}_M$ and $\boldsymbol{\Lambda}_M$ are exactly same as those of the full decomposition on $\mathbf{K}_c$. According to (Carrier et al., 1988), the computational gain of using such an incremental updating method becomes more obvious when the sizes of training and test dataset N and M become larger, especially when N and M are larger than 800.

Suppose we have transformed $\mathbf{K}_{11}$ to $\tilde{\mathbf{K}}_{11}$ using training data, and then test data $\mathbf{x}_j^{(t)}$ arrives one at a time. We can update $\mathbf{d}_j$ without retraining as follows:

- Calculate $\mathbf{K}_j = \begin{pmatrix} \tilde{\mathbf{K}}_{11} & \mathbf{d}_j \\ \mathbf{d}_j^T & k_j \end{pmatrix} = \mathbf{U}_j\boldsymbol{\Lambda}_j\mathbf{U}_j^T$.

dataset	#num(#class)	measure
Cat-cortex	65(4)	connection strength
Proteins	226(4)	evolutionary distance
Music-emd	50(2)	earth mover's distance
Music-ptd	50(2)	transportation distance
Kimia-hau	72(6)	hausdorff distance
Glass-sig	214(6)	sigmoid kernel
Unipen-dtw	250(5)	dynamic time warping
Usps-td	250(2)	tangent distance

Table 1. Datasets Description.

- Transform $\tilde{\mathbf{\Lambda}}_j = f(\mathbf{\Lambda}_j)$ to get a psd. $\tilde{\mathbf{K}}_j$.
- Use the associated partition $\tilde{\mathbf{d}}_j^T$ for test in the SVM's decision function $\operatorname{sgn}\left(f'(\mathbf{x}_j^{(t)}) = \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_j^{(t)}, \mathbf{x}_i) = \tilde{d}^T \mathbf{Y}\boldsymbol{\alpha}\right)$.

This incremental method uses the old $\boldsymbol{\alpha}$ from the training process, and thus can be regarded as a fast approximation to the full decomposition with retraining.

4. Experimental Evaluation

We conducted extensive experiments to compare four spectrum-transformation methods. Specifically, we attempt to empirically demonstrate the following:

1. Illustrate the impact of shift η on SVMs.

2. Compare four methods on transforming the similarity matrices generated using different measures.

3. Compare different generalization methods on updating the cross-similarity matrix $\mathbf{K}_{21}$.

In the experiments we used eight real-world datasets, seven of which are given in terms of a proximity matrix $\mathbf{K}$ and one of which uses sigmoid kernel ($\tanh(s\mathbf{x}^T\mathbf{x}' + d)$) to form a similarity matrix $\mathbf{K}$. When the proximity matrix is a dissimilarity matrix $\mathbf{D}$, we use the technique in MDS (Cox & Cox, 2001) of doubly centering it in $-\frac{1}{2}\mathbf{HDH}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, so as to form an inner product matrix as $\mathbf{K}$.

Table 1 lists a brief[1] description of these eight datasets and the corresponding proximity measures. Cat-cortex consists of a matrix of connection strengths between 65 cortical areas in 4 different regions. Proteins consists of a proximity matrix from the structural comparison of 226 protein sequences in 4 classes of globins. Music-emd and Music-ptd include two dissimilarity matrices of 50 and 57 music pieces in a binary setting, where the former uses the earth-mover's distance (EMD) as a similarity measure and the latter uses the proportional transportation distance (PTD). Kimia has 72 binary images in 6 classes, and pairwise similarity is measured by a modified Hausdorff distance. Glass-sig comes from the UCI repository and consists of 214 instances in 4 classes. We use a sigmoid kernel to form a proximity matrix $\mathbf{K}$. Unipen-dtw includes a dynamic-time-warping dissimilarity matrix of 250 handwritten sequences in 5 classes. Usps-td uses a tangent-distance matrix of 250 USPS handwritten-digit data in a binary class.

All experiments were performed using Spider machine learning library in Matlab. For each dataset, we used 10 cross validation and reported both the mean and standard error of validation error rates. For a multi-class dataset, we used one-per-class (OPC) ensemble strategy in SVMs with the formulation of 1-norm error. Its hyperparameter C was tuned to be optimal for each dataset-transformation pair. The β in diffusion was empirically[2] chosen as $\frac{1}{\max|\lambda_i|}$.

4.1. Role of η in shift

Figure 2 uses all datasets to illustrate how the length of vector $\boldsymbol{\alpha}$ and classification-error rate in SVMs change when the η used in shift is varied. Because of page limits, we only show four of them. In each sub-figure, x-axis represents how many times η is shifted to transform the kernel matrix. The darker curve (red, lower curve) represents the distribution of $\|\boldsymbol{\alpha}\|^2$, and the lighter curve (green, upper curve) represents the distribution of error rate. We can see that with the scaling factor increasing, $\|\boldsymbol{\alpha}\|^2$ keeps decreasing. The error-rate first goes down, but then goes up. This bears out our prediction in Section 2.4 that the shift η penalizes the length of $\boldsymbol{\alpha}$. It also tells us that we can not keep increasing η; otherwise it will overfit the dataset.

4.2. Comparison of classification performance

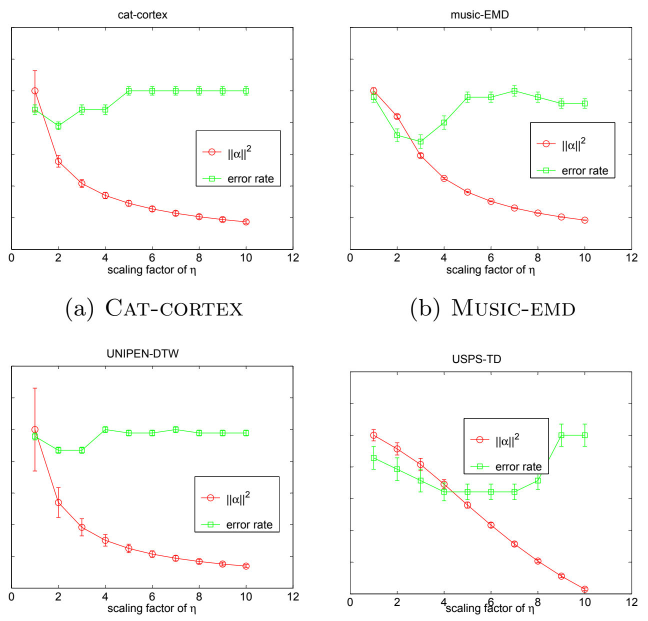
We used all datasets to compare the classification performance. For each dataset, we ran four spectrum-transformation techniques (denoise, flip, diffusion, and shift), and reported their optimal results after cross-validation. Table 2 reports the results in error rate with the mean and standard error. Each row of the table presents the results on a dataset. The second column lists the smallest eigenvalue λ_N. The third column lists the normalized λ_N, which is defined as the absolute value of itself divided by the largest eigenvalue. The third and fourth columns list the mean and standard error of the negative and positive λ_i's, respectively. The best results are marked with a bold font for clear illustration.

[1]The detailed definition of each dataset can be found in http://mmdb.ece.ucsb.edu/~gangwu/datasets/distances.htm.

[2]Diffsusion method involves a parameter β which needs a lot of cross-validation work to tune. Our empirical results showed that choosing $\frac{1}{\beta} = \max|\lambda_i|$ works better than other strategies, such as mean$|\lambda_i|$ or min $|\lambda_i|$.

dataset	λ_N	$\lvert\frac{\lambda_N}{\lambda_1}\rvert$	negative λ_i's	positive λ_i's	denoise	flip	diffusion	shift
CAT-CORTEX	−2.47	0.20	−0.91(0.76)	2.62(2.69)	12.38(2.83)	13.81(2.52)	11.20(2.30)	**10.95**(**3.29**)
PROTEINS	−0.01	0.00	−0.01(0.00)	5.20(14.32)	**4.01**(**1.19**)	**4.01**(**1.19**)	8.38(1.29)	**4.01**(**1.19**)
MUSIC-EMD	−0.58	0.32	−0.15(0.17)	0.26(0.33)	42.00(7.72)	52.00(5.80)	36.00(1.84)	**36.00**(**5.51**)
MUSIC-PTD	−0.21	0.13	−0.08(0.08)	0.24(0.33)	40.00(6.93)	**36.00**(**5.51**)	44.00(2.27)	44.00(4.73)
GLASS-SIG	−83.97	6.10	−2.28(6.89)	1.02(5.23)	38.25(2.67)	**36.36**(**3.60**)	44.48(1.37)	49.42(4.18)
KIMIA-HAU	−0.01	0.00	−0.01(0.01)	0.04(0.14)	**6.79**(**2.95**)	**6.79**(**2.95**)	17.86(5.83)	**6.79**(**2.95**)
UNIPEN-DTW	−4.33	0.12	−0.63(0.83)	1.35(3.98)	8.00(2.04)	**7.60**(**2.00**)	18.00(0.90)	10.00(2.61)
USPS-TD	−0.15	0.03	−0.04(0.04)	0.23(0.60)	5.20(1.60)	5.60(1.41)	13.20(0.82)	**5.20**(**1.52**)

Table 2. Classification-Error Rate Using Different Transformation Methods. For each dataset, the result is reported by transforming the full matrix $\mathbf{K}_c$ consisted of both training and test data.



(a) CAT-CORTEX (b) MUSIC-EMD

(c) UNIPEN-DTW (d) USPS-TD

Figure 2. $\|\boldsymbol{\alpha}\|^2$ and error rate with different η. In each plot, $\|\boldsymbol{\alpha}\|^2$ and error rate are scaled differently for clear illustration.

Some observations were found from Table 2. First, the diffusion method does not work well for all datasets in this classification setting. It also involved two tedious cross-validation works on both β and the hyperparameter C. Second, denoise works well only when normalized-λ_N is close to zero. In fact, when normalized-λ_N becomes larger, as in CAT-CORTEX, GLASS-SIG, and UNIPEN-DTW, denoise performs worse than flip. Third, flip works well when normalized-λ_N is very large. This means that the negative eigenvalues with large magnitude might not be considered as noise. Fourth, shift works best in five out of eight datasets, especially when normalized-λ_N has a smaller magnitude. However, when normalized-λ_N has a larger magnitude, it does not work better than flip. The reason could be that true noisy eigenvalues around zero are magnified by shifting with a sizable η, which could deteriorate the classification performance.

4.3. Comparison of generalization performance

We used all datasets to compare several methods in both effectiveness and efficiency for updating the cross-similarity matrix $\mathbf{K}_{21}$. Table 4 reports the classification-error rate with mean and standard error and the CPU time in seconds. Table 3 reports the p-value using two-way ANOVA, which is a matched statistical significance test. For each dataset, we first chose a spectrum-transformation method (denoise, flip, or shift), and then ran the three generalization methods for updating the $\mathbf{K}_{21}$, which were discussed in Sections 2.5 and 3. In Table 4, the three generalization methods, sorted from top to bottom for each dataset, is the embedding method, the full decomposition method, and the incremental decomposition method without retraining.

When denoise and flip were employed, the embedding method performed the worst on almost all datasets. This result was expected since the embedding method was proven to have no generalization ability in Section 2.5. For shift, since it does not update the off-diagonal items, using the embedding method works exactly the same as using the incremental method, but worse than the full decomposition. Compared to the full decomposition method, the incremental method achieved slightly worse results. However, considering that the incremental method does not require the test data to be available during training and it is faster than the full decomposition method (as reported in Table 4), a little sacrifice on performance may be acceptable. More computational benefits can be achieved while using a larger-size proximity matrix.

5. Conclusion

In this paper, we have reported our research on a critical problem in many emerging applications where data can only formulate a non-*psd* similarity matrix and cannot train with kernel methods. We gave an analytical framework on evaluating several representa-

dataset	effectiveness (in error-rate)			efficiency (in cputime)		
	denoise	flip	shift	denoise	flip	shift
Cat-cortex†	21.43(4.91)	**9.29(2.42)**	10.95(3.29)	0.017	0.022	–
Cat-cortex‡	12.38(2.83)	13.81(2.52)	**10.95(3.29)**	0.058	0.048	–
Cat-cortex§	14.05(2.53)	**10.71(2.24)**	10.95(3.29)	0.058	0.049	–
Proteins†	4.01(1.19)	4.01(1.19)	4.01(1.19)	0.356	0.366	–
Proteins‡	4.01(1.19)	4.01(1.19)	4.01(1.19)	0.889	0.903	–
Proteins§	4.01(1.19)	4.01(1.19)	4.01(1.19)	0.729	0.742	–
Music-emd†	44.00(6.81)	42.00(6.60)	**38.00(5.25)**	0.003	0.017	–
Music-emd‡	42.00(7.72)	52.00(5.80)	**36.00(5.51)**	0.034	0.033	–
Music-emd§	44.00(7.38)	50.00(6.48)	**38.00(5.25)**	0.034	0.034	–
Music-ptd†	48.00(5.06)	46.00(6.96)	**44.00(4.73)**	0.002	0.016	–
Music-ptd‡	40.00(6.93)	**36.00(5.51)**	44.00(4.73)	0.033	0.028	–
Music-ptd§	44.00(6.20)	**40.00(4.90)**	44.00(4.73)	0.034	0.028	–
Glass-sig†	81.80(2.33)	57.53(1.86)	**48.48(4.28)**	0.066	0.064	–
Glass-sig‡	38.25(2.67)	**36.36(3.60)**	49.42(4.18)	0.372	0.400	–
Glass-sig§	69.65(1.98)	**36.36(3.54)**	48.48(4.28)	0.280	0.299	–
Kimia-hau†	6.79(2.95)	6.79(2.95)	6.79(2.95)	0.017	0.030	–
Kimia-hau‡	6.79(2.95)	6.79(2.95)	6.79(2.95)	0.059	0.061	–
Kimia-hau§	6.79(2.95)	6.79(2.95)	6.79(2.95)	0.050	0.051	–
Unipen-dtw†	32.00(3.15)	14.00(2.55)	**10.00(2.61)**	0.456	0.461	–
Unipen-dtw‡	8.00(2.04)	**7.60(2.00)**	10.00(2.61)	1.128	1.197	–
Unipen-dtw§	10.40(2.13)	**7.20(1.86)**	10.00(2.61)	0.926	0.976	–
Usps-td†	**4.80(1.24)**	4.80(1.36)	5.20(1.52)	0.480	0.472	–
Usps-td‡	5.20(1.60)	5.60(1.41)	**5.20(1.52)**	1.170	1.161	–
Usps-td§	4.80(1.36)	5.60(1.62)	**5.20(1.52)**	0.961	0.954	–

Table 4. Performance Comparison in Effectiveness and Efficiency Using Different Generalization Methods. For each dataset, we report the error rate with the mean and standard error and the cputime in second using three generalization methods. The first row, denoted with †, gives the result using the embedding generalization method. The second row, with ‡, gives the result by decomposing the full matrix consisting of both training and test data. The third row, with §, gives the result of using incremental decomposition without retraining. Since we estimate the λ_N using the method discussed in Section 2.5, the shift method does not need an eigendecompostion on the kernel matrix and thus its CPU time on transformation is negligible.

dataset	transformation	generalization	interaction
Cat-cortex	0.1218	0.7447	0.3332
Proteins	1.0000	1.0000	1.0000
Music-emd	0.1558	0.8799	0.8927
Music-ptd	0.7277	0.4624	0.9354
Glass-sig	0.0000	0.0000	0.0000
Kimia-hau	1.0000	1.0000	1.0000
Unipen-dtw	0.0010	0.0000	0.0001
Usps-td	0.9513	0.9513	0.9984

Table 3. A nonparametric Statistical Testing using Two-Way Analysis of Variance (ANOVA). Results are reported using p-value. For each dataset, two factors, generalization and transformation, affects the classification performance. In the table, the second column records the p-value of the first factor, generalization. The third column represents the second factor, transformation. The fourth column records the p-value of the effects due to the interaction of both generalization and transformation factors.

tive spectrum-transformation methods. For all methods, we gave both theoretical and experimental analyses on their relationship with kernel machines, including semantics preservation, computational cost, and generalization capability. We also proposed an incremental generalization method to efficiently update the cross-similarity matrix when the test instances are not present during training.

One intriguing insight that we obtain from this study is the intricate balance between semantics preservation and noise removal when spectrum-transformation is performed. Our future work will embark on discerning useful semantics from noise, and performing spectrum mapping in an adaptive way.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). A basic local alignment search tool. *Journal of Molecular Biology*, *215*, 403–410.

Carrier, J., Greengard, L., & Rokhlin, V. (1988). A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Statist. Comput.*, *9*, 669–686.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In *NIPS 15*.

Cox, T. F., & Cox, M. A. A. (2001). *Multidimensional scaling.* Chapman & Hall/CRC.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations*. Baltimore: The Johns Hopkins University Press.

Graepel, T., Herbrich, R., Bollmann-Sdorra, P., & Obermayer, K. (1999). Classification on pairwise proximity data. In *NIPS 11*.

Gu, M., & Eisenstat, S. C. (1995). A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Anal. Appl.*, *16*, 172–191.

Haasdonk, B. (2004). Feature space interpretation of svms with indefinite kernels. *To appear in IEEE Trans. on PAMI*.

Hassdonk, B., & Keysers, D. (2002). Tangent distance kernels for support vector machines. In *ICPR'02*.

Horn, R. A., & Johnson, C. R. (1985). *Matrix analysis*. Cambridge, UK: Cambridge University Press.

Kimeldorf, G. S., & Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, *41*, 495–502.

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. In *ICML'02*.

Lin, H.-T., & Lin, C.-J. (2003). *A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods* (Technical Report). National Taiwan Univ.

Ong, C. S., Mary, X., Canu, S., & Smola, A. J. (2004). Learning with non-positive kernels. In *ICML'04*.

Pekalska, E., Paclik, P., & Duin, R. P. W. (2002). A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, *2*, 175–211.

Qamra, A., Meng, Y., & Chang, E. Y. (2005). Enhanced perceptual distance functions and indexing for image replica recognition. *IEEE Trans. on PAMI*, *27*, 379–391.

Roth, V., Laub, J., Kawanabe, M., & Buhmann, J. M. (2003). Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Trans. on PAMI*, *25*.

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, *40*, 99–121.

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Shimodaira, H., ichi Noma, K., Nakai, M., & Sagayama, S. (2001). Dynamic time-alignment kernel in support vector machine. In *NIPS 14*.

Simard, P., Cun, Y. L., & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In *NIPS 5*.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Wilkinson, J. H. (1988). *The algebraic eigenvalue problem*. Oxford University Press.