Learning with Idealized Kernels

James T. Kwok JAMESK@CS.UST.HK
Ivor W. Tsang IVOR@CS.UST.HK
Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Abstract

The kernel function plays a central role in kernel methods. Existing methods typically fix the functional form of the kernel in advance and then only adapt the associated kernel parameters based on empirical data. In this paper, we consider the problem of adapting the kernel so that it becomes more similar to the so-called ideal kernel. We formulate this as a distance metric learning problem that searches for a suitable linear transform (feature weighting) in the kernel-induced feature space. This formulation is applicable even when the training set can only provide examples of similar and dissimilar pairs, but not explicit class label information. Computationally, this leads to a local-optima-free quadratic programming problem, with the number of variables independent of the number of features. Performance of this method is evaluated on classification and clustering tasks on both toy and real-world data sets.

1. Introduction

In recent years, there has been a lot of interest on using kernels in various aspects of machine learning, such as classification, regression, clustering, ranking and principal component analysis (Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002; Vapnik, 1998). The basic idea of kernel methods is to map the data from an input space to a feature space $\mathcal{F}$ via some map φ, and then apply a linear procedure there. It is now well-known that the computations do not involve φ explicitly, but depend only on the inner product defined in $\mathcal{F}$, which in turn can be obtained efficiently from a suitable *kernel* function (the "kernel trick").

Because of the central role of the kernel, a poor kernel choice can lead to significantly impaired performance. Typically, the practitioner has to select the kernel before learning starts, with common choices being the polynomial kernel and radial basis function (RBF) kernel. The associated kernel parameters (such as the order in polynomial kernel, or the width in RBF kernel) can then be determined by optimizing a quality functional of the kernel (Ong et al., 2003). Examples of quality functionals include kernel target alignment (Cristianini et al., 2002b), generalization error bounds (Chapelle et al., 2002; Vapnik, 1998), Bayesian probabilities (Sollich, 2002) and cross-validation error.

Instead of adapting only the kernel parameters, a recent development is on adapting also the form of the kernel itself. As all information on the feature space is encoded in the kernel matrix (also called the *Gram matrix*), one can bypass the learning of the kernel function by just learning the kernel matrix instead, using techniques such as semi-definite programming (Lanckriet et al., 2002) or alignment maximization (Cristianini et al., 2002b). However, these usually work better in transduction problems. For the induction setting, a novel approach that selects the kernel function directly is by using the superkernel (Ong et al., 2003), which is a linear combination of kernels in a reproducing kernel Hilbert space on the space of kernels itself. A related approach is to obtain this kernel combination by boosting (Crammer et al., 2003). An earlier kernel adaptation work based on information geometry has also been proposed in (Amari & Wu, 1999).

On the other hand, as a kernel defines an inner product (and, consequently, a distance metric) in the feature space $\mathcal{F}$, the kernel design problem can also be regarded as the problem of finding a good distance metric or a set of good feature weights in $\mathcal{F}$. In the past decades, a large number of feature weighting methods have been proposed (Wettschereck et al., 1997). However, standard feature weighting methods

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

operate in the input space and the number of parameters (weights) increases with the number of features. Hence, they cannot be easily kernelized, as the dimensionality of $\mathcal{F}$ is usually very large (sometimes even infinite). Note that some feature weighting methods have recently been proposed specifically for support vector machines (SVMs) (e.g., (Grandvalet & Canu, 2003)). However, they again can only select features in the input space, but not in the feature space.

Another limitation of existing kernel design methods and most feature weighting methods is that they are designed for classification problems, and thus assume the availability of class label information in the training set. However, this is sometimes difficult to obtain and we may only know that certain pairs of patterns are similar (or dissimilar). Recently, (Xing et al., 2003) proposed a distance metric learning method that utilizes such similarity information using convex programming. Experimentally, this leads to significantly improved clustering performance. However, it involves an iterative procedure with projection and eigen decomposition, and becomes very computationally expensive when the number of features is large.

More informally, the kernel can be regarded as a function defining the pairwise similarity between patterns. For perfect classification (clustering), two patterns should be considered "similar" if and only if they belong to the same class (cluster). The resultant kernel constructed from such an "oracle" is called the *ideal kernel* (Cristianini et al., 2002b). This, of course, implies perfect knowledge of the problem and such an "oracle" does not exist in practice.

While obtaining the ideal kernel function is infeasible, in a classification problem, one can at least define the ideal kernel matrix on the training patterns based on the provided label information. As the ideal kernel is optimal, we can *idealize* a given kernel by making it more similar to the ideal kernel matrix. The crucial question, however, is on how to generalize this to patterns outside the training set. In this paper, we constrain the kernel adaptation by a linear transform on $\mathcal{F}$, which can also be regarded as learning the feature weights or distance metric in $\mathcal{F}$. We then show that this can be further extended from classification problems to situations where only similarity information is available. Computationally, we obtain a quadratic programming problem, as is commonly encountered in the kernel literature.

The rest of this paper is organized as follows. Section 2 gives a short review on the ideal kernel and alignment as detailed in (Cristianini et al., 2002a; Cristianini et al., 2002b). Section 3 describes the proposed kernel adaptation method. Experimental results on both toy and real-world data sets are presented in Section 4, and the last section gives some concluding remarks.

2. Ideal Kernel and Alignment

Consider a two-class classification problem with training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y \in \{-1, +1\}$. The kernel function defines an inner product or, more generally, a pairwise similarity measure, in the feature space. In the ideal case where the class label $y(\mathbf{x})$ for any pattern $\mathbf{x}$ can be obtained from an "oracle", two patterns $\mathbf{x}_i, \mathbf{x}_j$ should be considered "similar" if and only if they belong to the same class. Thus, the optimal kernel, or the so-called *ideal kernel*, K^* is given by (Cristianini et al., 2002b)[1]:

$$K^*_{ij} = K^*(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & y(\mathbf{x}_i) = y(\mathbf{x}_j), \\ 0 & y(\mathbf{x}_i) \neq y(\mathbf{x}_j). \end{cases} \quad (1)$$

Of course, this requires perfect knowledge of the problem and such an "oracle" does not exist in practice. Another extreme in kernel design is to avoid the use of domain knowledge completely. However, this will lead to the *trivial kernel* and is impossible to learn (Cristianini et al., 2002b). Hence, some prior knowledge or constraint is always needed in designing the kernel. A commonly used constraint is that the desired kernel is a weighted combination of some *base kernels* (Cristianini et al., 2002a; Lanckriet et al., 2002).

To assess the quality of a kernel, we will focus on one particular quality functional called the *alignment* (Cristianini et al., 2002b). The (empirical) alignment of a kernel k_1 with a kernel k_2 w.r.t. to the sample is defined as: $A(k_1, k_2) = \frac{\langle K_1, K_2 \rangle}{\sqrt{\langle K_1, K_1 \rangle}\sqrt{\langle K_2, K_2 \rangle}}$, where K_i is the kernel matrix for the sample using kernel k_i, and $\langle \cdot, \cdot \rangle$ is the Frobenius product[2]. When K_2 is the ideal kernel, this (kernel target) alignment can be used to measure the similarity between the kernel and target. Moreover, the estimate of alignment is concentrated[3], meaning that if we obtain a high alignment on the training set, we can also expect a high alignment on the test set. Finally, high alignment also implies good generalization performance of the resulting classifier.

[1] In (Cristianini et al., 2002b), $K^*_{ij} = -1$ if $y(\mathbf{x}_i) \neq y(\mathbf{x}_j)$.

[2] The Frobenius product between two matrices $M = [m_{ij}]$ and $N = [n_{ij}]$ is given by $\langle M, N \rangle = \sum_{ij} m_{ij} n_{ij}$.

[3] An empirical estimate is concentrated if the probability that it deviates from its mean can be bounded as an exponentially decaying function of that deviation.

3. Adapting the Kernel

3.1. The Idealized Kernel

In this Section, we consider *idealizing* a given kernel K such that it becomes more similar to the ideal kernel K^*. A simple way is to modify K to

$$\tilde{K} = K + \frac{\gamma}{2}K^*, \tag{2}$$

where $\gamma \geq 0$ is a parameter to be determined (Section 3.2.3). For a two-class problem, this means

$$\tilde{K}_{ij} = \begin{cases} K_{ij} + \frac{\gamma}{2} & y_i = y_j, \\ K_{ij} & y_i \neq y_j. \end{cases}$$

As both K and K^* are valid kernels, so is the *idealized kernel* $\tilde{K}$. On restricting the value of γ to 2, $\tilde{K}$ reduces to the kernel proposed in (Zhang et al., 2002).

As we would expect, $\tilde{K}$ will have a higher alignment.

Proposition 1 *Assuming that $\gamma > 0$, then the alignment of the idealized kernel $\tilde{K}$ will be greater than that of the original kernel K if $\gamma > -\frac{\langle K,K^*\rangle}{n_+^2+n_-^2}$, where n_+, n_- are the numbers of positive and negative examples in the training set respectively.*

Proof. The alignments for K and $\tilde{K}$ are $A(K,K^*) = \frac{\langle K,K^*\rangle}{\sqrt{\langle K^*,K^*\rangle}\sqrt{\langle K,K\rangle}}$ and $\frac{\langle \tilde{K},K^*\rangle}{\sqrt{\langle K^*,K^*\rangle}\sqrt{\langle \tilde{K},\tilde{K}\rangle}}$ respectively. Now, $\langle K^*,K^*\rangle = n_+^2 + n_-^2$, $\langle \tilde{K},K^*\rangle = \langle K + \frac{\gamma}{2}K^*,K^*\rangle = \langle K,K^*\rangle + \frac{\gamma}{2}(n_+^2+n_-^2)$ and $\langle \tilde{K},\tilde{K}\rangle = \langle K,K\rangle + \gamma\langle K,K^*\rangle + \frac{\gamma^2}{4}(n_+^2+n_-^2)$. Using the facts that $\langle K,K\rangle > 0, |A(K,K^*)| \leq 1$ and the assumption that $\gamma > 0$, result then follows after some simplification. □

Remark. As $\gamma > 0$, so as long as $\langle K,K^*\rangle \geq 0$ (i.e., K is aligned in the "right" direction), then the idealized kernel will have an increased alignment.

Extending to $C > 2$ classes is straightforward. Analogous to (1), the ideal kernel matrix K^* will be block diagonal (after some row/column permutations) of the form:

$$K^* = \begin{bmatrix} \mathbf{1}\mathbf{1}'_{n_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}\mathbf{1}'_{n_2} & \cdots & \mathbf{0} \\ \vdots & \cdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1}\mathbf{1}'_{n_C} \end{bmatrix},$$

where n_i is the number of patterns belonging to the ith class, and $\mathbf{1}\mathbf{1}'_{n_i}$ is a $n_i \times n_i$ matrix with all ones. Obviously, both K^* and $\tilde{K}$ are again valid kernels, and $\tilde{K}$ will have a higher alignment if $\gamma > -\frac{\langle K,K^*\rangle}{\sum_{i=1}^{C} n_i^2}$. In general, we may assign different weights to different classes (as when the classes are highly unbalanced), and similar properties on $\tilde{K}$ still apply.

3.2. Formulation as Distance Metric Learning

Recall that the ideal kernel can only be defined on the training patterns in practice, and so consequently is the idealized kernel in (2). An important question is then on how to generalize this for unseen patterns. In this Section, we first address the linear kernel (and the feature space is thus identical to the input space), and the nonlinear case will be postponed to Section 3.3.

First, assume that the original inner product for any two patterns $\mathbf{x}_i, \mathbf{x}_j$ in input space $\mathbb{R}^p$ is defined as $s_{ij} = K_{ij} = \mathbf{x}_i'\mathbf{M}\mathbf{x}_j$, where $\mathbf{M}_{p\times p}$ is a positive semi-definite matrix. The corresponding squared distance is then $d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)'\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$. On changing K to $\tilde{K}$, this squared distance will be changed to

$$\tilde{K}_{ii} + \tilde{K}_{jj} - 2\tilde{K}_{ij} = \begin{cases} d_{ij}^2 & y_i = y_j, \\ d_{ij}^2 + \gamma & y_i \neq y_j. \end{cases} \tag{3}$$

Now, consider incorporating input feature weights as in (Wettschereck et al., 1997), and modify the inner product to

$$\tilde{s}_{ij} = \mathbf{x}_i'\mathbf{A}\mathbf{A}'\mathbf{x}_j. \tag{4}$$

Here, $\mathbf{A}_{p\times p} = [\mathbf{a}_1, \ldots, \mathbf{a}_p]$ where the $\mathbf{a}_i$'s are a set of "useful" directions in the input space. The corresponding distance metric becomes:

$$\tilde{d}_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)'\mathbf{A}\mathbf{A}'(\mathbf{x}_i - \mathbf{x}_j). \tag{5}$$

Note that as $\mathbf{A}\mathbf{A}'$ is positive semi-definite, $\tilde{d}$ is always a valid distance metric.

We now search for an $\mathbf{A}$ such that the distance metric (5) obtained from feature weighting approximates the desired (3) obtained from the idealized kernel:

$$\tilde{d}_{ij}^2 \begin{cases} \leq d_{ij}^2 & y_i = y_j, \\ \geq d_{ij}^2 + \gamma & y_i \neq y_j. \end{cases} \tag{6}$$

In other words, patterns in different classes will be pulled apart (by an amount at least equal to γ) under the modified distance metric, while those in the same class may get closer together. This is again in line with the traditional wisdom of reducing intra-class variability and increasing inter-class variability.

The above formulation can be easily extended to the case where only similarity information is available. Denote the sets containing similar and dissimilar pairs by $\mathcal{S}$ and $\mathcal{D}$ respectively. (6) can then be modified to:

$$\tilde{d}_{ij}^2 - d_{ij}^2 \begin{cases} \leq 0 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \\ \geq \gamma & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}. \end{cases} \tag{7}$$

This setting will be our main focus in the sequel.

3.2.1. Primal and Dual

In general, we may not be able to perfectly enforce (7) for all pairs in $\mathcal{D}$ and $\mathcal{S}$. Hence, as for SVMs, slack variables will be introduced in the optimization problem below. Moreover, recall that $\mathbf{A}$ performs a projection onto a set of useful features. This set should ideally be small, meaning that a small rank for $\mathbf{A}$ (or $\mathbf{AA}'$, as $\text{rank}(\mathbf{AA}') = \text{rank}(\mathbf{A})$) will be desirable. Assume that the eigen decomposition of $\mathbf{AA}'$ is $\mathbf{U\Sigma U}'$. Then, $\text{rank}(\mathbf{AA}') = \text{rank}(\mathbf{\Sigma}) = \|\mathbf{\Sigma}\|_0$. However, a direct minimization of the zero norm is difficult and hence we will approximate it by the Euclidean norm $\|\mathbf{\Sigma}\|_2 = \|\mathbf{AA}'\|_2$. The above discussion thus leads to the following primal problem[4], which is similar to that of the ν-SVM (Schölkopf et al., 2000):

$$\begin{aligned}\min_{\mathbf{A},\gamma,\xi_{ij}} \quad & \frac{1}{2}\|\mathbf{AA}'\|_2^2 + \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \xi_{ij} \\ & + C_{\mathcal{D}} \left(-\nu\gamma + \frac{1}{N_{\mathcal{D}}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \xi_{ij} \right),\end{aligned}$$

subject to

$$\begin{aligned} d_{ij}^2 &\geq \tilde{d}_{ij}^2 - \xi_{ij}, \qquad (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}, \qquad (8)\\ \tilde{d}_{ij}^2 - d_{ij}^2 &\geq \gamma - \xi_{ij}, \qquad (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}, \qquad (9)\\ \xi_{ij} &\geq 0,\\ \gamma &\geq 0.\end{aligned}$$

Here, $N_{\mathcal{S}}$ and $N_{\mathcal{D}}$ are the numbers of pairs in $\mathcal{S}$ and $\mathcal{D}$ respectively, and $C_{\mathcal{S}}, C_{\mathcal{D}}, \nu$ are non-negative adjustable parameters.

To solve this constrained optimization problem, we use the standard method of Lagrange multipliers. The Lagrangian function is

$$\begin{aligned}\mathcal{L} = & \frac{1}{2}\|\mathbf{AA}'\|_2^2 + \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \xi_{ij} \\ & + C_{\mathcal{D}} \left(-\nu\gamma + \frac{1}{N_{\mathcal{D}}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \xi_{ij} \right) \\ & - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}((\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{M}-\mathbf{AA}')(\mathbf{x}_i-\mathbf{x}_j) \\ & + \xi_{ij}) \\ & - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}\left((\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{AA}'-\mathbf{M})(\mathbf{x}_i-\mathbf{x}_j)\right. \\ & \left. -\gamma+\xi_{ij}\right) - \sum_{i>j} \eta_{ij}\xi_{ij} - \mu\gamma. \qquad (10)\end{aligned}$$

Setting the derivatives w.r.t. the primal variables to zero, we obtain (on assuming that $\mathbf{A}$ is non-singular[5]),

$$\begin{aligned}\mathbf{AA}' = & -\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)' \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)', \qquad (11)\\ \nu = & \frac{1}{C_{\mathcal{D}}}\left(\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij} - \mu\right), \qquad (12)\\ \alpha_{ij} = & \begin{cases} \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}} - \eta_{ij} & (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S},\\ \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}} - \eta_{ij} & (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}.\end{cases} \qquad (13)\end{aligned}$$

Substitute into (10), the dual problem becomes maximizing (w.r.t. the α_{ij}'s)

$$\begin{aligned}\Psi = & -\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\mathbf{x}_i-\mathbf{x}_j)'\mathbf{M}(\mathbf{x}_i-\mathbf{x}_j) \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(\mathbf{x}_i-\mathbf{x}_j)'\mathbf{M}(\mathbf{x}_i-\mathbf{x}_j) \\ & - \frac{1}{2}\sum_{(\mathbf{x}_i,\mathbf{x}_j),(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{S}} \alpha_{ij}\alpha_{kl}((\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{x}_k-\mathbf{x}_l))^2 \\ & - \frac{1}{2}\sum_{(\mathbf{x}_i,\mathbf{x}_j),(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{D}} \alpha_{ij}\alpha_{kl}((\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{x}_k-\mathbf{x}_l))^2 \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}\sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{D}} \alpha_{ij}\alpha_{kl}((\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{x}_k-\mathbf{x}_l))^2, \qquad (14)\end{aligned}$$

subject to

$$\frac{1}{C_{\mathcal{D}}}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij} \geq \nu, \qquad (15)$$

and

$$0 \leq \alpha_{ij} \leq \begin{cases} \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}} & (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S},\\ \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}} & (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}.\end{cases} \qquad (16)$$

This is a standard quadratic programming (QP) problem with $N_{\mathcal{S}}+N_{\mathcal{D}}$ variables (which is independent of p, the dimensionality of $\mathbf{x}$). Being a QP problem, it thus does not suffer from the problem of local optimum. Recently, (Xing et al., 2003) also proposed a distance metric learning method based on similarity information by using a full (or diagonal) distance metric matrix. However, this leads to a convex programming problem with p^2 (or p) variables when a full (diagonal) matrix is used, and involves an iterative procedure comprising projection and eigen decomposition. It is thus more costly, especially when p is high.

[4] Note that $d_{ii} = \tilde{d}_{ii} = 0$ and hence (7) is always satisfied when $i = j$ (with $\xi_{ii} = 0$). Thus, in the optimization problem, we only need to consider those ξ_{ij}'s with $i \neq j$.

[5] This is always the case in the experiments. Geometrically, this means that the neighborhood at any point will not extend to infinity.

Finally, using (11) and the α_{ij}'s obtained from the dual, we can compute the modified inner product from (4) and the corresponding distance metric from (5).

3.2.2. "Support Vectors" and "Error Pairs"

In general, the Lagrangian multipliers in a constrained optimization problem measure the rates of change in the objective function, consequent upon changes in the corresponding constraints. Hence, α_{ij} reflects how difficult it is for the modified distance metric $\tilde{d}$ to satisfy the requirements in (7). By studying the Karush-Kuhn-Tucker (KKT) conditions of the primal problem, we obtain the following proposition:

Proposition 2 *For* $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$,

$$\tilde{d}_{ij}^2 - d_{ij}^2 \begin{cases} = \gamma & 0 < \alpha_{ij} < \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}, \\ \geq \gamma & \alpha_{ij} = 0, \\ \leq \gamma & \alpha_{ij} = \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}, \end{cases}$$

whereas for $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$,

$$d_{ij}^2 - \tilde{d}_{ij}^2 \begin{cases} = 0 & 0 < \alpha_{ij} < \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}}, \\ \geq 0 & \alpha_{ij} = 0, \\ \leq 0 & \alpha_{ij} = \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}}. \end{cases}$$

Proof. The KKT conditions of the primal are:

$$\begin{aligned} \alpha_{ij}(d_{ij}^2 - \tilde{d}_{ij}^2 + \xi_{ij}) &= 0, \quad (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, && (17) \\ \alpha_{ij}(\tilde{d}_{ij}^2 - d_{ij}^2 - \gamma + \xi_{ij}) &= 0, \quad (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, && (18) \\ \eta_{ij}\xi_{ij} &= 0, && (19) \\ \mu\gamma &= 0. && (20) \end{aligned}$$

When $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$, there are three possible cases. First, $0 < \alpha_{ij} < \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}$. (18) gives

$$\tilde{d}_{ij}^2 - d_{ij}^2 - \gamma + \xi_{ij} = 0, \qquad (21)$$

while (13) leads to $\eta_{ij} > 0$, which in turn gives $\xi_{ij} = 0$ on using (19). Putting this back into (21), we obtain $\tilde{d}_{ij}^2 - d_{ij}^2 = \gamma$. For the second case, consider $\alpha_{ij} = 0$. Again, from (13), we have $\eta_{ij} > 0$ and thus $\xi_{ij} = 0$. Substituting back into the constraint in (9), we obtain $\tilde{d}_{ij}^2 - d_{ij}^2 \geq \gamma$. For the last case, consider $\alpha_{ij} = \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}$. Again, $\alpha_{ij} > 0$ leads to (21). Moreover, from (13), $\eta_{ij} = 0$ which means ξ_{ij} may be nonzero (from (19)). Thus $\tilde{d}_{ij}^2 - d_{ij}^2 \leq \gamma$.

Now, consider $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$. When $0 < \alpha_{ij} < \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}}$, (17) gives

$$d_{ij}^2 - \tilde{d}_{ij}^2 + \xi_{ij} = 0, \qquad (22)$$

while (13) leads to $\eta_{ij} > 0$, which in turn gives $\xi_{ij} = 0$ on using (19). Putting this back into (22), we obtain $\tilde{d}_{ij}^2 - d_{ij}^2 = 0$. For the second case, consider $\alpha_{ij} = 0$. Again, from (13), we have $\eta_{ij} > 0$ and thus $\xi_{ij} = 0$. Substituting back into the constraint in (8), we obtain $d_{ij}^2 - \tilde{d}_{ij}^2 \geq 0$. For the last case, consider $\alpha_{ij} = \frac{C_{\mathcal{S}}}{N_{\mathcal{S}}}$. Again, $\alpha_{ij} > 0$ leads to (22). Moreover, from (13), $\eta_{ij} = 0$ which means ξ_{ij} may be nonzero (from (19)). Thus $d_{ij}^2 - \tilde{d}_{ij}^2 \leq 0$. □

Remark. In other words, when the Lagrange multipliers are nonzero but less than the upper bound, the constraints in (7) will be exactly met. When the Lagrange multipliers are zero, the constraints in (7) will be met with a larger "margin" and the corresponding pairs will not appear in the final solution in (11) (i.e., they are not "support vectors"). Finally, when the Lagrange multipliers are at the upper bound, the constraints in (7) may be violated ("error") and the corresponding ξ_{ij}'s may be nonzero. The situation is thus analogous to that of SVM.

3.2.3. Value of γ

Consider taking a pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$ such that $0 < \alpha_{ij} < \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}$. Using Proposition 2, γ can then be obtained as $\gamma = \tilde{d}_{ij}^2 - d_{ij}^2$ (which is also intuitive from (7)). In the implementation, we obtain an average value of γ over all $(\mathbf{x}_i, \mathbf{x}_j)$ pairs that satisfy the above criteria.

3.2.4. Interpretation of ν

Analogous to ν-SVM (Schölkopf et al., 2000), we have:

Proposition 3 *ν is a lower bound on the fraction of support vectors in $\mathcal{D}$. Moreover, if $\gamma > 0$, then ν is also an upper bound on the fraction of error pairs in $\mathcal{D}$.*

Proof. From (15), (16), we have

$$\begin{aligned} \nu &\leq \frac{1}{C_{\mathcal{D}}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \alpha_{ij} > 0} \alpha_{ij} \leq \frac{1}{C_{\mathcal{D}}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \alpha_{ij} > 0} \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}} \\ &= \frac{\#((\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \alpha_{ij} > 0)}{N_{\mathcal{D}}}. \end{aligned}$$

Recall that there are $N_{\mathcal{D}}$ α_{ij}'s in $\mathcal{D}$ and all support vectors have $\alpha_{ij} > 0$. Hence, ν is a lower bound on the fraction of support vectors in $\mathcal{D}$. Moreover, as $\gamma > 0$, then $\mu = 0$ from (20), and (12) yields

$$\begin{aligned} \nu &= \frac{1}{C_{\mathcal{D}}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij} \geq \frac{1}{C_{\mathcal{D}}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \alpha_{ij} = \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}} \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}} \\ &= \frac{\#((\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \alpha_{ij} = \frac{C_{\mathcal{D}}}{N_{\mathcal{D}}})}{N_{\mathcal{D}}}. \end{aligned}$$

The inequality holds as the second summation includes only a subset of the nonzero α_{ij}'s. Recall that for the error pairs in $\mathcal{D}$, α_{ij} is at the upper bound $\frac{C_{\mathcal{D}}}{N_{\mathcal{D}}}$. Hence, ν is also an upper bound on the fraction of error pairs in $\mathcal{D}$. □

3.2.5. Heuristic for Computational Speedup

Recall that our QP problem has $N_{\mathcal{S}} + N_{\mathcal{D}}$ variables. When similarity information is abundant, $N_{\mathcal{S}}+N_{\mathcal{D}}$ can be of $O(n^2)$ for a data set with n patterns, and hence computationally expensive for large data sets. In this paper, we use a simple heuristic inspired from locally linear embedding (Roweis & Saul, 2000). The basic idea is that for each pattern $\mathbf{x}$, its local neighborhood will be the most influential. Hence, we only select the m closest $(\mathbf{x}, \mathbf{x}_j)$ pairs in $\mathcal{S}$ and $\mathcal{D}$ such that each of these $\mathbf{x}_j$'s is also within a radius of R from $\mathbf{x}$[6]. Thus, $N_{\mathcal{S}} + N_{\mathcal{D}}$ will at most be of $O(n)$.

3.3. Kernel Version

The previous results can be easily kernelized by replacing all the $\mathbf{x}$ by $\varphi(\mathbf{x})$, where φ is the feature map corresponding to the original kernel K. The idealized kernel $\tilde{K}$ is then given by:

$$\begin{aligned}\tilde{K}(\mathbf{x}_a, \mathbf{x}_b) &= -\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(K_{ai} - K_{aj})(K_{ib} - K_{jb}) \\ &\quad + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(K_{ai} - K_{aj})(K_{ib} - K_{jb}).\end{aligned}$$

and the distance metric is:

$$\begin{aligned}\tilde{d}^2(\mathbf{x}_a, \mathbf{x}_b) &= -\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(K_{ai} - K_{aj} - K_{bi} + K_{bj})^2 \\ &\quad + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(K_{ai} - K_{aj} - K_{bi} + K_{bj})^2.\end{aligned}$$

4. Experiments

In this Section, we perform experiments on one toy data set and four real-world data sets (Table 1)[7]. The toy data has one relevant feature, for which the first class is normally distributed as $N(3, 1)$ and the second class as $N(-3, 1)$, and ten other irrelevant features, each of them is distributed as $N(0, 25)$.

Experiments in Section 4.1 assume only the availability of similarity information, while Section 4.2 uses a standard classification setting. Our results are compared with those of the original kernels (linear kernel and RBF kernel) and, wherever possible[8], those of (Xing et al., 2003) with a full distance metric matrix. To reduce statistical variability, results here are based on averages over 50 random repetitions.

Table 1. Datasets used in the experiments.

data set	dim	class	# patterns
toy	11	1	50
		2	50
colon	2000	1	22
		2	40
lymphoma	4026	1	61
		2	35
soybean	35	1	10
		2	10
		3	10
		4	17
wine	12	1	59
		2	71
		3	48

4.1. With Similarity Information Only

The experimental setup is similar to that in (Xing et al., 2003). $\mathcal{S}$ is generated as a random subset of all pairs of patterns that belong to the same class. Its size is chosen such that the number of resulting connected components is roughly 70% of the size of the original data set. The learned distance metric is then used for both classification (using the 1-nearest neighbor classifier) and clustering (using the k-means clustering algorithm[9]).

As can be seen from Tables 2 and 3, the learned metric leads to better classification and clustering results. Figure 1 shows the kernel matrices for the toy data on a

[6] In the experiments, m is set to 5 and R is set to the median of distances for all pairs in $\mathcal{S}$ and $\mathcal{D}$.

[7] soybean and wine are from the UCI machine learning repository, while colon and lymphoma are from http://www.kyb.tuebingen.mpg.de/bs/people/weston/l0.

[8] As mentioned in Section 3.2.1, the number of parameters in (Xing et al., 2003) scales quadratically with the number of features. Hence, we cannot apply this method on the colon and lymphoma data sets, nor for adapting the RBF kernel (which induces an infinitely-dimensional feature space).

[9] Here, k is set to the number of classes in each data set. Moreover, clustering accuracy is defined as (Xing et al., 2003):

$$\text{accuracy} = \sum_{i>j} \frac{1\{1\{c_i = c_j\} = 1\{\hat{c}_i = \hat{c}_j\}\}}{0.5n(n-1)},$$

where $1\{\cdot\}$ is the indicator function, n is the number of patterns in the data set, c_i is the true cluster label for pattern $\mathbf{x}_i$, and $\hat{c}_i$ is the corresponding label returned by the clustering algorithm.

Table 2. Classification errors with kernel adaptation using similarity information.

data set	kernel	Euclidean metric	learned metric	Xing *et al.*
toy	linear	28.25%	9.83%	9.83%
	rbf	27.75%	16.50%	-
colon	linear	28.75%	17.08%	-
	rbf	27.83%	22.67%	-
lymphoma	linear	14.17%	8.50%	-
	rbf	11.00%	9.94%	-
soybean	linear	2.82%	0.11%	0.71%
	rbf	1.76%	0.94%	-
wine	linear	28.03%	12.00%	13.00%
	rbf	27.36%	26.28%	-

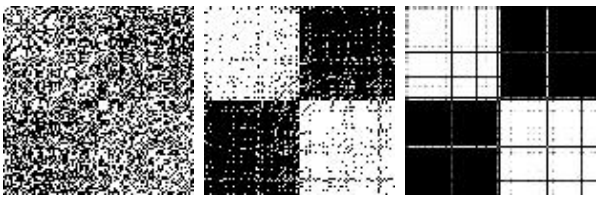
typical run of the clustering experiment. The original kernel matrix corresponding to the Euclidean metric (Figure 1(a)) is very noisy, due to the presence of irrelevant features. In contrast, those obtained from the learned metrics exhibit clear block structures, though the one produced by (Xing et al., 2003) still has some undesirable lines (Figure 1(c)).

Table 3. Clustering errors with kernel adaptation using similarity information.

data set	kernel	Euclidean metric	learned metric	Xing *et al.*
toy	linear	44.67%	0.00%	1.89%
	rbf	49.84%	0.00%	-
colon	linear	22.85%	17.72%	-
	rbf	17.77%	14.87%	-
lymphoma	linear	20.50%	11.14%	-
	rbf	20.50%	15.08%	-
soybean	linear	16.37%	0.00%	0.00%
	rbf	15.55%	0.00%	-
wine	linear	28.13%	22.37%	26.54%
	rbf	27.96%	26.88%	-

4.2. With Class Label Information

In this Section, we employ a standard classification setting. A subset of patterns (60 for toy, 50 for colon, 60 for lymphoma, and about two-third of the whole data set for soybean and wine) are randomly selected to form the training set, while the remaining patterns are used for testing. The sets $\mathcal{S}$ and $\mathcal{D}$ are constructed by defining two patterns as similar if they belong to the same class, and dissimilar otherwise.



(a) Before. (b) After. (c) Xing *et al.*.

Figure 1. Kernel matrices on the toy data (with linear kernel).

Table 4 shows the classification results and the changes in alignments. Again, our method compares favorably with the original kernel and (Xing et al., 2003). Moreover, the alignments on both the training and test sets typically improve after adaptation.

5. Conclusion

In this paper, we propose a kernel *idealization* scheme that aims at adapting a given kernel to be more similar to the ideal kernel. Because in practice the ideal kernel can only be defined on the training patterns, this cannot be achieved in a straightforward manner. However, by noting that a kernel effectively defines a distance metric on the corresponding feature space, we formulate this idealization task as a distance metric learning problem that looks for a suitable linear transform (feature weighting) in the feature space. Moreover, this formulation only requires a training set with examples of similar and dissimilar pairs, but not explicit class label information. Computationally, it leads to a local-optima-free quadratic programming problem, with the number of variables is independent of the number of features. Experiments on toy and real-world data sets demonstrate that our proposed method yields improved performance on both classification and clustering tasks, with the presence of either similarity or class label information.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grants HKUST2033/00E and HKUST6195/02E. The authors would like to thank Eric Xing for providing his code used in (Xing et al., 2003).

Table 4. Classification errors with kernel adaptation using class label information.

data set	kernel	Euclidean metric	learned metric	Xing *et al.*	train align (before)	train align (after)	test align (before)	test align (after)
toy	linear	28.50%	3.08%	3.33%	0.17	0.62	0.15	0.53
	rbf	27.75%	7.92%	-	0.70	0.77	0.69	0.73
colon	linear	29.83%	14.67%	-	0.15	0.28	0.31	0.33
	rbf	27.83%	16.83%	-	0.74	0.70	0.76	0.68
lymphoma	linear	14.17%	8.11%	-	0.19	0.30	0.18	0.20
	rbf	13.67%	11.17%	-	0.68	0.69	0.70	0.75
soybean	linear	2.82%	0.12%	0.59%	0.60	0.70	0.61	0.68
	rbf	2.82%	1.17%	-	0.77	0.86	0.79	0.84
wine	linear	28.03%	10.13%	22.58%	0.53	0.54	0.55	0.56
	rbf	27.62%	26.82%	-	0.71	0.58	0.66	0.50

References

Amari, S., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, *12*, 783–789.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Crammer, K., Keshet, J., & Singer, Y. (2003). Kernel design using boosting. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Cristianini, N., Kandola, J., Elisseeff, A., & Shawe-Taylor, J. (2002a). On kernel target alignment. Submitted to Journal of Machine Learning Research.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. (2002b). On kernel-target alignment. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Grandvalet, Y., & Canu, S. (2003). Adaptive scaling for feature selection in SVMs. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. (2002). Learning the kernel matrix with semi-definite programming. *Proceedings of the International Conference on Machine Learning* (pp. 323–330).

Ong, C., Smola, A., & Williamson, R. (2003). Superkernels. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT.

Schölkopf, B., Smola, A., Williamson, R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, *12*, 1207–1245.

Sollich, P. (2002). Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, *46*, 21–52.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Wettschereck, D., Aha, D., & Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, *11*, 273–314.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Zhang, Z., Kwok, J., Yeung, D., & Wang, W. (2002). *A novel distance-based classifier using convolution kernels and Euclidean embeddings* (Technical Report HKUST-CS02-28). Department of Computer Science, Hong Kong University of Science and Technology.