Kernel Relevant Component Analysis for Distance Metric Learning

Ivor W. Tsang Pak-Ming Cheung James T. Kwok
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay
Hong Kong
Email: {ivor,pakming,jamesk}@cs.ust.hk

Abstract— **Defining a good distance measure between patterns is of crucial importance in many classification and clustering algorithms. Recently, relevant component analysis (RCA) is proposed which offers a simple yet powerful method to learn this distance metric. However, it is confined to linear transforms in the input space. In this paper, we show that RCA can also be kernelized, which then results in significant improvements when nonlinearities are needed. Moreover, it becomes applicable to distance metric learning for structured objects that have no natural vectorial representation. Besides, it can be used in an incremental setting. Performance of this kernel method is evaluated on both toy and real-world data sets with encouraging results.**

I. Introduction

Many classification and clustering algorithms rely on the use of an inner product or distance measure between patterns. Examples include the nearest neighbor classifiers, radial basis function networks, kernel methods and k-means clustering. While measuring distances appears to be a simple problem when the data can be described by a handful of meaningful features (attributes), many real-world applications involve the use of thousands of features (or even more). Given this large pool of features, it is apparent that many of them may be highly correlated with each other or even irrelevant to the task being considered. The commonly-used Euclidean distance assumes that all features are of equal importance, which is thus often violated in practice.

As a remedy, a large number of feature weighting methods have been proposed [1], [19]. However, the number of parameters involved typically increases with the number of features, and they are thus prone to the curse of dimensionality problem. Moreover, another limitation is that they are usually designed for classification problems, and thus require the availability of class label information. However, label information is often too strong, and may not be readily available in some applications.

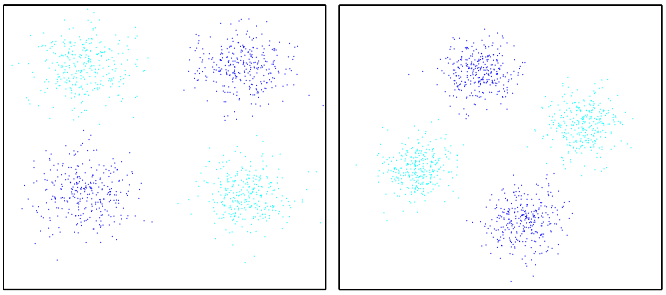
Recently, there has been a lot of interest on learning with side information (e.g., [7]). One type of side information that is often easier to obtain is similarity (dissimilarity) information [20], i.e., we may only know that certain pairs of patterns are similar (dissimilar). An example in surveillance application is suggested in [14]. Here, faces from successive frames of the surveillance video in roughly the same location often come from the same person. Thus, we can know that these successive faces are similar, although we are not given the exact identities (labels) of these faces. Unlike class label information, such similarity information can be obtained at virtually no cost in this case. The use of similarity information has led to significant improvements in clustering [3], [16], [17]. Recently, a number of researchers have also proposed distance metric methods that can utilize such similarity information [8], [13], [20].

However, these distance metric learning methods rely heavily on convex programming, which can be computational expensive in some applications (e.g., information retrieval), and a faster method is thus called for. Recently, [14] proposed the method of relevant component analysis (RCA). It performs a linear transform on the input space such that the Euclidean distance in the transformed space is less affected by irrelevant variabilities. The basic idea is to assign large weights to the relevant features and small weights to the irrelevant features. Moreover, it can also be shown that the RCA transform is optimal in an information theoretic sense [2]. Computationally, it involves only one matrix inversion. Results obtained in [2] are comparable or even better than those in [20].

Though simple yet powerful, RCA still suffers from several limitations. First, RCA is restricted to the use of linear transforms in the input space, and can fail even in simple nonlinear problems such as the XOR problem. As can be seen in Figure 1, the two classes cannot be separated even after performing the RCA transform. Second, as RCA operates in the input space, its number of parameters is dependent on the dimensionality of the feature vectors. Hence, it suffers from the same curse of dimensionality problem that plagues many traditional feature weighting methods, and dimensionality reduction is often required before RCA can be run. Third, RCA relies on a vectorial representation of the data. However, some objects, such as protein sequences, graphs and trees, may not have a natural feature vector representation and thus cannot benefit from RCA. Finally, in an adaptive environment, there is a continual input of new information. For example, in the surveillance example mentioned earlier, there is a continual arrival of new video clips. Thus, it is desirable to have RCA being able to operate in an incremental setting, such that the computations do not have to start from scratch every time.

In this paper, we propose the use of kernels in RCA. The

0-7803-9048-2/05/$20.00 ©2005 IEEE



(a) Original. (b) After the RCA transform.

Fig. 1. An simple XOR example that RCA fails.

use of kernels has been highly successful in various aspects of machine learning, such as classification, regression, clustering, ranking and principal component analysis [12], [15]. A well-known example in supervised learning is the support vector machines (SVMs). 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").

The rest of this paper is organized as follows. RCA is first reviewed in Section II. Then, in Section III, we show that RCA can be kernelized and also can be computed in an incremental fashion. Experimental results are presented in Section IV, and the last section gives some concluding remarks.

II. Relevant Component Analysis

In this Section, we will briefly review relevant component analysis (RCA) as introduced in [14]. Here, similarity information is provided in the form of *chunklets*. Patterns in a chunklet are similar, i.e., belonging to the same class, though its exact class label is not known. Moreover, each chunklet may only contain a small number of patterns. As mentioned in Section I, the patterns are further assumed to be in some vectorial representation.

In the following, we assume that C chunklets are given, with chunklet c containing n_c patterns $\{\mathbf{x}_{c,1}, \ldots, \mathbf{x}_{c,n_c}\}$, where each $\mathbf{x}_{c,i} \in \mathbb{R}^d$. In RCA, the chunklets are first centered. Then, the covariance matrix of the centered patterns in all the chunklets is obtained, as:

$$\mathbf{C} = \frac{1}{n} \sum_{c=1}^{C} \sum_{i=1}^{n_c} (\mathbf{x}_{c,i} - \bar{\mathbf{x}}_c)(\mathbf{x}_{c,i} - \bar{\mathbf{x}}_c)'. \tag{1}$$

Here, $\bar{\mathbf{x}}_c$ denotes the mean of chunklet c and $n = \sum_{c=1}^{C} n_c$. Finally, a whitening transform associated with this chunklet covariance matrix $\mathbf{C}$ is applied, and each pattern $\mathbf{x}$ in the data set is transformed as

$$\mathbf{x} \mapsto \mathbf{C}^{-\frac{1}{2}} \mathbf{x}. \tag{2}$$

RCA is thus similar to principal component analysis (PCA), except that it does not aim at finding directions that represent most of the variance in the data. Computationally, RCA only involves one matrix inversion of the $d \times d$ matrix $\mathbf{C}$. Thus, it is very efficient when the input dimensionality d is low. However, on high-dimensional data sets where d is large, computing the chunklet covariance matrix explicitly can be expensive, if not prohibitive.

III. Kernel RCA

With the use of kernels, the patterns will first be implicitly mapped from the input space to the kernel-induced feature space $\mathcal{F}$. The dimensionality of $\mathcal{F}$ is usually very high, sometimes even infinite (such as when the Gaussian kernel is used). Thus, a direct computation of the chunklet covariance matrix in (1) is not feasible.

In Section III-A, we first show how RCA can be kernelized. An incremental procedure for updating the (kernel) RCA transform is then proposed in Section III-B.

A. Chunklet Covariance Matrix

For class $c = 1, \ldots, C$, let $\mathbf{1}_c$ be the n-dimensional vector with $[\mathbf{1}_c]_i = \begin{cases} 1, & \text{pattern } i \in c \\ 0, & \text{otherwise} \end{cases}$, and $\mathbf{I}_c$ be the $n \times n$ diagonal matrix $\mathrm{diag}(\mathbf{1}_c)$. We also stack the chunklet patterns together in a $d \times n$ matrix, as

$$\mathbf{X} = [\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \ldots, \mathbf{x}_{1,n_1}, \ldots \mathbf{x}_{C,1}, \mathbf{x}_{C,2}, \ldots, \mathbf{x}_{C,n_C}]. \tag{3}$$

Then, $\mathbf{C}$ in (1) can be written in matrix form, as

$$\begin{aligned}
\mathbf{C} &= \frac{1}{n} \sum_{c=1}^{C} \sum_{i=1}^{n_c} \left(\mathbf{x}_{c,i} - \frac{1}{n_c} \mathbf{X} \mathbf{1}_c \right) \left(\mathbf{x}_{c,i} - \frac{1}{n_c} \mathbf{X} \mathbf{1}_c \right)' \\
&= \frac{1}{n} \sum_{c=1}^{C} \sum_{i=1}^{n_c} \Bigg(\mathbf{x}_{c,i} \mathbf{x}_{c,i}{}' - \frac{1}{n_c} \mathbf{X} \mathbf{1}_c \mathbf{x}_{c,i}{}' \\
&\quad - \frac{1}{n_c} \mathbf{x}_{c,i} \mathbf{1}_c{}' \mathbf{X}' + \frac{1}{n_c^2} \mathbf{X} \mathbf{1}_c \mathbf{1}_c{}' \mathbf{X}' \Bigg) \\
&= \frac{1}{n} \sum_{c=1}^{C} \mathbf{X} \left(\mathbf{I}_c - \frac{2}{n_c} \mathbf{1}_c \mathbf{1}_c{}' + \frac{1}{n_c} \mathbf{1}_c \mathbf{1}_c{}' \right) \mathbf{X}' \\
&= \frac{1}{n} \sum_{c=1}^{C} \mathbf{X} \left(\mathbf{I}_c - \frac{1}{n_c} \mathbf{1}_c \mathbf{1}_c{}' \right) \mathbf{X}' \\
&= \frac{1}{n} \mathbf{X} \mathbf{H} \mathbf{X}',
\end{aligned} \tag{4}$$

where

$$\mathbf{H} = \sum_{c=1}^{C} \left(\mathbf{I}_c - \frac{1}{n_c} \mathbf{1}_c \mathbf{1}_c{}' \right) \tag{5}$$

is an $n \times n$ matrix that is similar[1] to the conventional centering matrix $\left(\mathbf{I} - \frac{1}{n} \mathbf{1}\mathbf{1}' \right)$.

[1] In particular, $\mathbf{H}$ is symmetric, block diagonal, idempotent and positive semi-definite.

Note that $\mathbf{C}$ will be singular when $n \leq d$. Therefore, we add a regularizer $\epsilon\mathbf{I}$ to $\mathbf{C}$, where ϵ is a small positive constant. On using (4), we then have

$$\hat{\mathbf{C}} \;=\; \epsilon\mathbf{I} + \mathbf{C} = \epsilon\mathbf{I} + \frac{1}{n}\mathbf{X}\mathbf{H}\mathbf{X}'. \tag{6}$$

Using the Woodbury formula

$$(\mathbf{A}+\mathbf{B}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1},$$

we obtain the inverse of $\mathbf{C}$ as required in (2):

$$\begin{aligned}\hat{\mathbf{C}}^{-1} &= \left(\epsilon\mathbf{I} + \frac{1}{n}\mathbf{X}\mathbf{H}\cdot\mathbf{X}'\right)^{-1} \\ &= \frac{1}{\epsilon}\mathbf{I} - \frac{1}{n\epsilon^2}\mathbf{X}\mathbf{H}\left(\mathbf{I}+\frac{1}{n\epsilon}\mathbf{X}'\mathbf{X}\mathbf{H}\right)^{-1}\mathbf{X}'.\end{aligned}$$

The dot product between any two patterns $\mathbf{x}$ and $\mathbf{y}$ in the RCA transformed space is then equal to

$$\begin{aligned}&(\hat{\mathbf{C}}^{-\frac{1}{2}}\mathbf{x})'(\hat{\mathbf{C}}^{-\frac{1}{2}}\mathbf{y}) \\ &= \mathbf{x}'\hat{\mathbf{C}}^{-1}\mathbf{y} \\ &= \mathbf{x}'\left[\frac{1}{\epsilon}\mathbf{I} - \frac{1}{n\epsilon^2}\mathbf{X}\mathbf{H}\left(\mathbf{I}+\frac{1}{n\epsilon}\mathbf{X}'\mathbf{X}\mathbf{H}\right)^{-1}\mathbf{X}'\right]\mathbf{y}.\end{aligned}$$

Obviously, the computations above only involve dot products between patterns, and the kernel trick can be readily applied. Specifically, suppose that RCA is performed in the feature space $\mathcal{F}$ corresponding to a kernel function k (and a nonlinear map φ), then the dot product between $\varphi(\mathbf{x})$ and $\varphi(\mathbf{y})$ after running RCA in $\mathcal{F}$ is

$$\begin{aligned}\tilde{k}(\mathbf{x},\mathbf{y}) &= \frac{1}{\epsilon}k(\mathbf{x},\mathbf{y}) \\ &\quad -\mathbf{k}'_{\mathbf{x}}\left[\frac{1}{n\epsilon^2}\mathbf{H}\left(\mathbf{I}+\frac{1}{n\epsilon}\mathbf{K}\mathbf{H}\right)^{-1}\right]\mathbf{k}_{\mathbf{y}},\end{aligned} \tag{7}$$

where $\mathbf{K} = [k(\mathbf{x}_i,\mathbf{x}_j)]_{ij}$ is the $n \times n$ kernel matrix defined on the chunklet patterns, $\mathbf{k}_{\mathbf{x}} = [k(\mathbf{x}_{1,1},\mathbf{x}),\ldots,k(\mathbf{x}_{C,n_C},\mathbf{x})]'$ and $\mathbf{k}_{\mathbf{y}} = [k(\mathbf{x}_{1,1},\mathbf{y}),\ldots,k(\mathbf{x}_{C,n_C},\mathbf{y})]'$.

Notice that this kernelization does not incur additional costs. Thus, while most existing distance metric learning methods that are capable of utilizing similarity information are based on convex programming (Section I), an unique advantage of kernel RCA is that it is much faster, which may be crucial in some applications.

B. Incremental Update

As mentioned in Section I, in many applications, it is typical to have a continual input of new chunklets. As the computation in (7) involves the matrix inversion

$$\mathbf{H}\left(\mathbf{I}+\frac{1}{n\epsilon}\mathbf{K}\mathbf{H}\right)^{-1}, \tag{8}$$

it could become expensive if this matrix inversion had to be performed on the arrival of every chunklet. In this Section, we show that (8) can be computed in an efficient, incremental manner instead.

By redefining the value of ϵ, (6) can be rewritten as

$$\hat{\mathbf{C}} = \frac{1}{n}(\mathbf{X}\mathbf{H}\mathbf{X}' + \epsilon\mathbf{I}).$$

Using the Woodbury formula again, we obtain

$$\hat{\mathbf{C}}^{-1} = \frac{n}{\epsilon}\mathbf{I} - \frac{n}{\epsilon^2}\mathbf{X}\mathbf{H}\left(\mathbf{I}+\frac{1}{\epsilon}\mathbf{K}\mathbf{H}\right)^{-1}\mathbf{X}'. \tag{9}$$

In the following, we denote the set of all processed chunklets by A (containing a total of n_A patterns) and the new chunklet by B (with n_B patterns). Denote the corresponding $\mathbf{X}$'s in (3) by $\mathbf{X}_A$ and $\mathbf{X}_B$ (which are of size $d \times n_A$ and $d \times n_B$ respectively). Then, $\mathbf{K}$ and $\mathbf{H}$ can be decomposed as

$$\mathbf{K} = \begin{bmatrix}\mathbf{K}_{AA} & \mathbf{K}_{AB}\\ \mathbf{K}'_{AB} & \mathbf{K}_{BB}\end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix}\mathbf{H}_A & \mathbf{0}\\ \mathbf{0} & \mathbf{H}_B\end{bmatrix},$$

where $\mathbf{K}_{AA} = \mathbf{X}'_A\mathbf{X}_A, \mathbf{K}_{AB} = \mathbf{X}'_A\mathbf{X}_B, \mathbf{K}_{BB} = \mathbf{X}'_B\mathbf{X}_B$, $\mathbf{H}_A$ is the matrix $\mathbf{H}$ in (5) that corresponds to all the processed chunklets, and $\mathbf{H}_B = \mathbf{I} - \frac{1}{n_B}\mathbf{1}\mathbf{1}'$, where $\mathbf{1}$ is a n_B-dimensional vector of all ones.

Denote $\mathbf{Z}_A = (\mathbf{I} + \frac{1}{\epsilon}\mathbf{K}_{AA}\mathbf{H}_A)^{-1}$. Using the fact that

$$\begin{aligned}\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} &= \begin{bmatrix}\mathbf{A}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} + \\ &\quad \begin{bmatrix}-\mathbf{A}^{-1}\mathbf{B}\\ \mathbf{I}\end{bmatrix}\mathbf{P}^{-1}\begin{bmatrix}-\mathbf{C}\mathbf{A}^{-1} & \mathbf{I}\end{bmatrix},\end{aligned}$$

where $\mathbf{P} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ and $\mathbf{0}$ is a zero matrix with the appropriate dimensions, we have

$$\begin{aligned}&\left(\mathbf{I}+\frac{1}{\epsilon}\mathbf{K}\mathbf{H}\right)^{-1} \\ &= \left(\mathbf{I}+\frac{1}{\epsilon}\begin{bmatrix}\mathbf{K}_{AA}\mathbf{H}_A & \mathbf{K}_{AB}\mathbf{H}_B\\ \mathbf{K}'_{AB}\mathbf{H}_A & \mathbf{K}_{BB}\mathbf{H}_B\end{bmatrix}\right)^{-1} \\ &= \begin{bmatrix}\mathbf{I}+\frac{1}{\epsilon}\mathbf{K}_{AA}\mathbf{H}_A & \frac{1}{\epsilon}\mathbf{K}_{AB}\mathbf{H}_B\\ \frac{1}{\epsilon}\mathbf{K}'_{AB}\mathbf{H}_A & \mathbf{I}+\frac{1}{\epsilon}\mathbf{K}_{BB}\mathbf{H}_B\end{bmatrix}^{-1} \\ &= \begin{bmatrix}\mathbf{Z}_A & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} + \begin{bmatrix}-\frac{1}{\epsilon}\mathbf{Z}_A\mathbf{K}_{AB}\mathbf{H}_B\\ \mathbf{I}\end{bmatrix} \\ &\quad \left(\mathbf{I}+\frac{1}{\epsilon}\mathbf{K}_{BB}\mathbf{H}_B - \frac{1}{\epsilon^2}\mathbf{K}'_{AB}\mathbf{H}_A\mathbf{Z}_A\mathbf{K}_{AB}\mathbf{H}_B\right)^{-1} \\ &\quad \begin{bmatrix}-\frac{1}{\epsilon}\mathbf{K}'_{AB}\mathbf{H}_A\mathbf{Z}_A & \mathbf{I}\end{bmatrix}.\end{aligned}$$

Denote $\mathbf{Y}_A = \mathbf{H}_A\mathbf{Z}_A$. Then, (8) becomes (on replacing $n\epsilon$ by ϵ):

$$\begin{aligned}&\mathbf{H}\left(\mathbf{I}+\frac{1}{\epsilon}\mathbf{K}\mathbf{H}\right)^{-1} \\ &= \begin{bmatrix}\mathbf{Y}_A & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} + \begin{bmatrix}-\frac{1}{\epsilon}\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H}_B\\ \mathbf{H}_B\end{bmatrix} \\ &\quad \left(\mathbf{I}+\frac{1}{\epsilon}\mathbf{K}_{BB}\mathbf{H}_B - \frac{1}{\epsilon^2}\mathbf{K}'_{AB}\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H}_B\right)^{-1} \\ &\quad \begin{bmatrix}-\frac{1}{\epsilon}\mathbf{K}'_{AB}\mathbf{Y}_A & \mathbf{I}\end{bmatrix}.\end{aligned} \tag{10}$$

Note that $\mathbf{Y}_A$ has already been obtained during the processing of chunklet A. Hence, on the arrival of the new chunklet

B, we have to compute the following terms in (10) with complexities[2]:

1) $\mathbf{K}_{AB}\mathbf{H}_B$: $O(n_A n_B^2)$;
2) $\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H}_B$: $O(n_A^2 n_B)$;
3) $\mathbf{K}_{BB}\mathbf{H}_B$: $O(n_B^3)$;
4) $\mathbf{K}'_{AB}\mathbf{Y}_A$: $O(n_A^2 n_B)$;
5) $\mathbf{K}'_{AB}\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H}_B$: $O(n_A n_B^2)$;
6) $\left(\mathbf{I} + \frac{1}{\epsilon}\mathbf{K}_{BB}\mathbf{H}_B - \frac{1}{\epsilon^2}\mathbf{K}'_{AB}\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H}_B\right)^{-1}$: $O(n_B^3)$;
7) multiplying the matrices together: $O((n_A+n_B)n_B^2)$ and $O((n_A+n_B)^2 n_B)$.

Ignoring the matrix additions, which take at most $O((n_A + n_B)^2)$, the total computational complexity is thus

$$O(n_A^2 n_B + n_A n_B^2 + n_B^3). \tag{11}$$

Typically, $n_B \ll n_A$. The complexity incurred in (11) is thus much less than that of the naive approach, which is $O((n_A + n_B)^3)$.

IV. Experiments

In this Section, experiments are performed on a number of toy and real-world data sets. Section IV-A first demonstrates the performance of kernel RCA on the XOR problem that motivated this work. As in [14], our kernelized version is then also evaluated in retrieval (Section IV-B) and clustering settings (Section IV-C). Performance gains on using the incremental update procedure is illustrated in Section IV-D.

A. XOR Problem

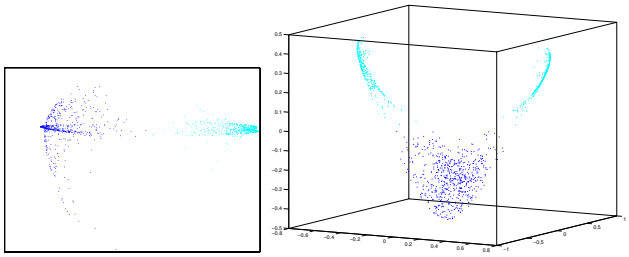
The same XOR problem as mentioned in Section I is used here. Thirty patterns are drawn from each cluster to form a total of four chunklets. As shown in Figure 1, linear RCA fails on this simple data set. Here, we apply kernel RCA with the Gaussian kernel. As the feature space induced by the Gaussian kernel is infinite-dimensional, so, for visualization purposes, we project the transformed patterns on the two leading principal axes obtained by kernel PCA [11]. As can be seen in Figure 2, the two classes now become well separated.

To quantitatively evaluate the qualities of the distance metrics, we perform the constrained k-means clustering algorithm [17] using both the original and the transformed metrics. Clustering accuracy is measured by the Rand index, which is defined as [20]:

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

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. To reduce statistical variability, results here are based on averages over 300 random repetitions.

Table I shows that kernel RCA can attain superb clustering performance on this data set. Note that even though a support vector machine with the Gaussian kernel can have perfect classification result on this data set, kernel clustering using the metric induced by the same Gaussian kernel leads to disappointing results. This can be illustrated by projecting the mapped patterns (in the feature space) to a three-dimensional space obtained by kernel PCA (Figure 2). The first class has both of its clusters in the upper part while the two clusters from the other class are in the lower part, thus leading to perfect classification. However, clusters belonging to the same class are still not close together, thus explaining the poor clustering result.



(a) After the KRCA transform.

(b) Feature space induced by the Gaussian kernel.

Fig. 2. Projections of the transformed XOR data.

TABLE I
Clustering results on the XOR data set.

kernel	(K)RCA	Rand index
linear	no	0.503
	yes	0.502
Gaussian	no	0.528
	yes	**1.000**

B. Retrieval Experiment

In this Section, we perform retrieval experiments on the Structural Classification of Proteins (SCOP) database (v.1.65) [10]. The two most commonly requested sequence subsets on the Astral database[3] [6], namely the 40%-subset which contains sequences with 40% or less identity to each other and the 95%-subset with sequences having 95% or less identity to each other, are used. From the 40%-subset, we choose the 47 largest superfamilies, resulting in a total of 1778 sequences. Similarly, from the 95%-subset, we choose the 22 largest superfamilies, with a total of 2540 sequences.

As protein sequences do not have a natural vectorial representation, only kernel RCA, but not the traditional RCA, can be performed. We adopt the spectrum kernel on sequences [9], which is defined as

$$\langle x, y \rangle = \sum_{\mathbf{s} \in \mathcal{A}^*} \text{num}_{\mathbf{s}}(x)\text{num}_{\mathbf{s}}(y),$$

[2]Here, we assume that each kernel evaluation takes $O(1)$ time. Moreover, the complexity for multiplying a $p \times q$ matrix by another $q \times r$ matrix is $O(pqr)$, while computing the inverse of an $n \times n$ matrix takes $O(n^3)$ time.

[3]The Astral database can be downloaded from http://astral.berkeley.edu.

where $\mathcal{A}^*$ is the k-spectrum[4] of the alphabet, and $\text{num}_{\mathbf{s}}(x), \text{num}_{\mathbf{s}}(y)$ are the numbers of occurrences of subsequence $\mathbf{s}$ in sequences x and y respectively. In this experiment, we choose $k = 3$. Moreover, as in [18], this spectrum kernel is further normalized as

$$k(x, y) = \frac{\langle x, y \rangle}{\sqrt{\langle x, x \rangle \langle y, y \rangle}}, \tag{12}$$

so that each feature vector has length 1.

The experimental setup is as follows. For each query in each superfamily, we use the spectrum kernel in (12) to retrieve varying numbers[5] of nearest neighbors. These are then labeled and grouped as chunklets for input to the kernel RCA. The following metrics are compared:

1) metric induced by the spectrum kernel;
2) metric learned by kernel RCA with the spectrum kernel;
3) metric induced by spectrum kernel and a further nonlinear map via the Gaussian kernel;
4) metric learned by kernel RCA using the combined kernel in 3 above.

Performance is evaluated based on the ROC (receiver operating characteristic) graph [5], which plots the true positive (TP) rate on the Y-axis and the false positive (FP) rate on the X-axis[6]. To provide a single numerical measure, the AUC, which is the area under the ROC curve, will be reported.

Table II shows the results. As can be seen, the use of kernel RCA leads to substantial improvement over the original kernel on both data sets.

TABLE II
RETRIEVAL RESULTS (AUC VALUES) ON THE SCOP DATA SET.

data set	method		spectrum kernel	spectrum kernel + Gaussian kernel
40%-subset	no KRCA		0.568	0.539
	w/ KRCA	20 neighbors	0.677	0.678
		30 neighbors	0.682	0.685
		40 neighbors	0.685	0.690
		50 neighbors	0.687	0.694
95%-subset	no KRCA		0.630	0.600
	w/ KRCA	20 neighbors	0.769	0.771
		30 neighbors	0.774	0.780
		40 neighbors	0.778	0.787
		50 neighbors	0.779	0.791

C. Clustering Experiments

In this Section, we perform clustering experiments by using the constrained k-means clustering algorithm[7] on the USPS handwritten digits dataset and a number of data sets from the UCI machine learning repository[8] (Table III) [4]. The experimental setup is similar to that in [20]. The similarity information is generated as a random subset of all pairs of patterns belonging to the same class. In the case of "little" side information, its size is chosen such that the number of resulting connected components is roughly 70% of the size of the original data set; whereas in the case of "much" side information, this is increased to 90%. The following distance metrics are compared:

1) Euclidean metric on the input space;
2) metric learned by RCA;
3) metric induced by the Gaussian kernel;
4) metric learned by kernel RCA with the Gaussian kernel.

TABLE III
DATA SETS USED IN THE CLUSTERING EXPERIMENTS.

data set	#features	#classes	#patterns
USPS	88	10	2500
breast-cancer	9	2	683
german	20	2	1000
glass	9	6	214
heart	13	2	270
image	18	7	2310
ionosphere	33	2	351
letter	16	26	2600
pima	8	2	768
satellite	36	6	2400
sonar	60	2	208
vehicle	18	4	846

Results are shown in Table IV. As can be seen, the use of (kernel) RCA in both the input space and kernel-induced feature space leads to improvements over the original metric. Moreover, the combined use of kernels and RCA as in kernel RCA yields the best performance on most data sets. This is particularly the case on data sets with many input features, in which it is likely that some of them are not related to the class labels.

D. Incremental Update

In this Section, we demonstrate the performance gains that can result from adopting the incremental update procedure in Section III-B. As an illustration, we only perform experiments on the USPS data set used in Section IV-C. We keep on adding chunklets of size 2, and measure the CPU time required to compute the expression in (8). The procedure is implemented in MATLAB and the experiment is run on a Pentium-4 3.2GHz machine, with 512MB RAM, running Windows XP. Results here are based on averages over 20 random repetitions.

Results are shown in Figure 3. As expected, the incremental updating approach leads to significant speedup.

V. CONCLUSION

In this paper, we showed that RCA, which is a simple yet powerful distance metric learning method capable of

[4]For any given $k \geq 1$, the *k-spectrum* of a sequence is the set of all the length-k contiguous subsequences that it contains.

[5]In this experiment, we have experimented with 20, 30, 40 and 50 nearest neighbors.

[6]TP and FP are defined by TP $= \frac{\text{positives correctly classified}}{\text{total positives}}$ and FP $= \frac{\text{positives incorrectly classified}}{\text{total negatives}}$, respectively.

[7]Here, k is set to the number of classes in each data set.

[8]For the letter and satellite data sets, we only use random subsets with 2,600 and 2,400 patterns (with equal number of patterns in each class) respectively.

TABLE IV

CLUSTERING RESULTS (AS MEASURED BY THE RAND INDEX) ON THE USPS AND UCI DATA SETS (ASTERISKS MEAN THAT THE MARKED METHOD OUTPERFORMS THE OTHER THREE AT A 95% LEVEL OF SIGNIFICANCE).

data set	side info	input space: no RCA	input space: with RCA	Gaussian kernel: no KRCA	Gaussian kernel: with KRCA
USPS	much	0.843	0.895	0.846	**0.927***
	little	0.785	0.810	0.793	**0.896***
breast-cancer	much	0.955	0.955	**0.961***	0.948
	little	0.934	0.921	**0.945***	0.899
german	much	0.515	**0.549***	0.518	0.517
	little	0.501	0.511	0.503	**0.512**
glass	much	0.635	0.644	0.636	**0.691***
	little	0.614	0.616	0.616	**0.651***
heart	much	**0.799***	0.738	0.784	0.627
	little	**0.737***	0.580	0.718	0.623
image	much	0.790	**0.837***	0.789	0.818
	little	0.746	**0.814**	0.749	0.811
ionosph	much	0.624	0.690	0.682	**0.807***
	little	0.586	0.661	0.632	**0.830***
letter	much	0.637	0.724	0.641	**0.777***
	little	0.602	0.671	0.607	**0.701***
pima	much	0.636	**0.661***	0.636	0.557
	little	0.577	0.567	**0.585***	0.548
satellite	much	0.795	0.817	0.800	**0.857***
	little	0.758	0.765	0.756	**0.824***
sonar	much	0.529	0.517	0.521	**0.626***
	little	0.511	0.532	0.509	**0.543***
vehicle	much	0.557	0.800	0.566	**0.834***
	little	0.540	0.761	0.544	**0.792***

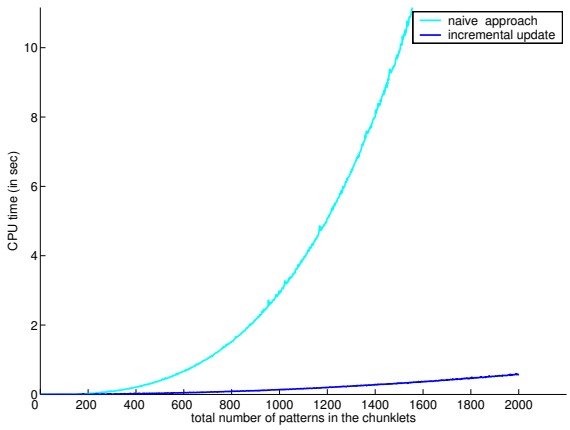


Fig. 3. CPU time (in seconds) vs the total number of patterns in the chunklets.

utilizing similarity information, can be efficiently kernelized. This not only extends the ability of RCA to produce nonlinear transforms of the input space, but also allows learning of distance metrics for structural objects, such as protein sequences. Moreover, the proposed incremental update procedure allows the kernel RCA transform to be computed efficiently in an adaptive environment. Experiments on a number of real-world data sets yield improved performance on both retrieval and clustering tasks.

Here, we have focused on finding a global metric which is used for all patterns. More generally, the metric can be local in nature and adaptive to the query or even to each individual pattern. This will be further investigated in the future.

REFERENCES

[1] D.W. Aha. Feature weighting for lazy learning algorithms. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer, Norwell, MA, 1998.

[2] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 11–18, Washington, D.C., USA, August 2003.

[3] S. Basu, M. Bilenko, and R Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *ICML 2003 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning & Data Mining*, Washington, D.C., USA, August 2003.

[4] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html University of California, Irvine, Department of Information and Computer Sciences.

[5] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[6] S.E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

[7] G. Chechik and N. Tishby. Extracting relevant structures with side information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[8] J.T. Kwok and I.W. Tsang. Learning with idealized kernels. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 400–407, Washington, D.C., USA, August 2003.

[9] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, Lihue, Hawaii, USA, 2002.

[10] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[11] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[12] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[13] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[14] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of the Seventh European Conference on Computer Vision*, volume 4, pages 776–792, Copenhagen, Denmark, 2002.

[15] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[16] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1103–1110, Standord, CA, USA, 2000.

[17] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584, Williamstown, MA, USA, 2001.

[18] J. Weston, B. Schölkopf, E. Eskin, C. Leslie, and S.W. Noble. Dealing with large diagonals in kernel matrices. *Principles of Data Mining and Knowledge Discovery, Springer Lecture Notes in Computer Science 243*, 2002.

[19] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.

[20] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.