Formulating Context-dependent Similarity Functions

Gang Wu, Edward Y. Chang
Department of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106
{gwu,echang}@ece.ucsb.edu

Navneet Panda
Department of Computer Science
University of California
Santa Barbara, CA 93106
panda@cs.ucsb.edu

ABSTRACT

Tasks of information retrieval depend on a good distance function for measuring similarity between data instances. The most effective distance function must be formulated in a context-dependent (also application-, data-, and user-dependent) way. In this paper, we present a novel method, which learns a distance function by capturing the nonlinear relationships among contextual information provided by the application, data, or user. We show that through a process called the "kernel trick," such nonlinear relationships can be learned efficiently in a projected space. In addition to using the kernel trick, we propose two algorithms to further enhance efficiency and effectiveness of function learning. For efficiency, we propose a SMO-like solver to achieve O(N^2) learning performance. For effectiveness, we propose using unsupervised learning in an innovative way to address the challenge of lack of labeled data (contextual information). Theoretically, we substantiate that our method is both sound and optimal. Empirically, we demonstrate that our method is effective and useful.

Categories and Subject Descriptors: I.5.1 [Pattern Recognition]: models-*statistical*
General Terms: algorithms, performance
Keywords: function learning, kernel machines

1. INTRODUCTION

At the heart of information organization and retrieval is a distance function that measures *similarity* between data instances. To date, most applications employ a variant of the *Euclidean distance* for measuring similarity. However, to measure similarity meaningfully, an effective distance function ought to consider the idiosyncrasies of the application, data, and user (hereafter we refer to these factors as contextual information). The quality of the distance function significantly affects the success in organizing data or finding meaningful results [1, 2, 7, 10, 12].

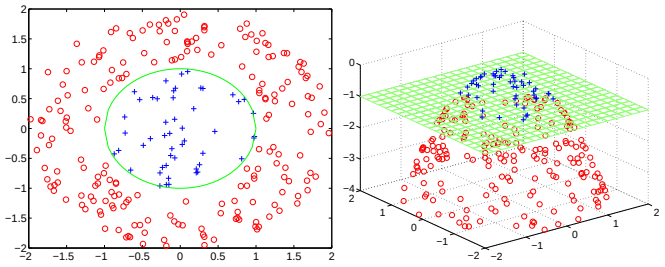
How do we consider contextual information in formulating a good distance function? One extension of the popular Euclidean distance (or more generally, the L_p-norm) is to weight the data attributes (features) based on their importance for a target task [2, 10, 26]. For example, for answering a *sunset* image-query, color features should be weighted higher. For answering an *architecture* image-query, shape and texture features may be more important. Weighting these features is equivalent to performing a *linear* transformation in the space formed by the features. Although linear models enjoy the twin advantages of simplicity of description and efficiency of computation, this same simplicity is insufficient to model similarity for many real-world datasets. For example, it has been widely acknowledged in the image/video retrieval domain that a query concept is typically a nonlinear combination of perceptual features (color, texture, and shape) [21, 24]. In this paper we first review a nonlinear-transformation framework [29] on the feature space to gain greater flexibility for mapping features to semantics; we then detail two companion algorithms, one for collecting contextual information and one for speeding up function learning.

We name our method *distance-function alignment* (DA$_{lign}$ for short). The inputs to DA$_{lign}$ are a prior distance function and *contextual information*. Contextual information can be conveyed in the form of *training data*. For instance, in the information-retrieval domain, Web users can convey information via relevance feedback showing which documents/images are relevant to their queries. In the biomedical domain, physicians can indicate which pairs of proteins may have similar functions. DA$_{lign}$ transforms the prior function to capture the nonlinear relationships among the *contextual information*. The similarity scores of unseen data-pairs can then be measured by the transformed function to better reflect the idiosyncrasies of the application, data, and user.

At first it might seem that capturing nonlinear relationships among contextual information can suffer from high computational complexity. DA$_{lign}$ avoids this concern by employing the *kernel trick*[1]. The kernel trick lets us generalize distance-based algorithms to operate in the *projected space* (defined next), usually nonlinearly related to the *input space*. The *input space* (denoted as $\mathcal{I}$) is the original space in which data vectors are located (e.g., in Figure 1(a)), and the *projected space* (denoted as $\mathcal{P}$) is that space to which the data vectors are projected, linearly or nonlinearly, (e.g., in Figure 1(b)). The advantage of using the *kernel trick* is that, instead of explicitly determining the coordinates of the data vectors in the projected space, the distance computation in $\mathcal{P}$ can be efficiently performed in $\mathcal{I}$ through a kernel function. Specifically, given two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ is defined as the inner product of $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, where ϕ is a basis function that maps the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ from $\mathcal{I}$ to $\mathcal{P}$. The inner product between two vectors can be thought of as a measure of their similarity. Therefore, $K(\mathbf{x}_i, \mathbf{x}_j)$ returns the similarity of $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{P}$.

[1]The *kernel trick* was first published in 1964 in the paper of M. Aizerman, E. Braverman, and L. Rozonoer's [3]. The kernel trick has been applied to several algorithms in statistics, including Support Vector Machines and kernel PCA.

MM'05, November 6–12, 2005, Singapore.



(a) Input Space (b) Projected Space

Figure 1: Clustering via the Kernel Trick.

The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in terms of the kernel is defined as

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2$$

$$= \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}. \quad (1)$$

Since a kernel function can be either linear or nonlinear, the traditional feature-weighting approach (e.g., [2, 26]) is just a special case of DA*lign*.

How does DA*lign* work? Given N unlabeled instances and a prior kernel function for measuring similarity, DA*lign* first generates an $N \times N$ gram matrix (similarity matrix) characterizing pairwise similarity of the N instances. To change the prior function, DA*lign* modifies the gram matrix. A critical challenge is to collect *sufficient* contextual information that can provide *maximal* information for modifying the similarity matrix, and thereby effectively align the prior function to the context. To achieve this aim, we propose measuring the *stability* of instance-pairs via perturbing clusters of data instances. Suppose we employ the spectral clustering algorithm [19] to cluster data instances using the prior kernel function. The cluster membership of an instance-pair is considered to be *stable* when the pair either always belong to the same cluster (i.e., they are similar) or always belong to different clusters (i.e., they are dissimilar), despite the various choices of the kernel parameter and the k value of the k-mean step in the spectral clustering algorithm. When we are uncertain of the similarity between a pair of instances, either because their cluster-membership is instable, or because their stability assessment disagrees with that provided by the prior kernel function, the instance-pair can be presented to the user to solicit a similarity score. The feedback scores, together with all the scores of the stable pairs, form the contextual information. The goal of DA*lign* is to transform the function in such a way that it can produce similarity scores in better agreement with the target task. More specifically, given a prior function (for example, a polynomial function or a Gaussian function) that produces default similarity scores between data instances, DA*lign* performs a linear transformation on the prior function in $\mathcal{P}$ by modifying the similarity matrix, based on the collected contextual information. Effectively, our DA*lign* procedure ensures that the distances between similar instance-pairs are decreased, and the distances between dissimilar pairs are increased, using an efficient SMO-like algorithm. Since performing a linear transformation in $\mathcal{P}$ can result in a nonlinear transformation in $\mathcal{I}$, DA*lign* achieves both model flexibility and computational efficiency. Theoretically, we prove that DA*lign* achieves optimal alignment to the *ideal* kernel defined by [9]. Empirically, we show that our experimental results back up our theoretical analysis.

In summary, we address in this paper a core problem of information organization and retrieval: formulating a context-dependent distance function to improve the accuracy of similarity measurement. DA*lign* achieves three significant contributions.

1. Thanks to the employment of the kernel trick and our proposed SMO-like solver, DA*lign* is an effective and efficient method for adapting a similarity measure to contextual information.

2. Wwe provide the proof of optimality of the DA*lign* algorithm.

3. We show how unsupervised learning algorithms can be used in an innovative way to assess membership stability of clusters for identifying most informative contextual information.

2. RELATED WORK

Distance-function learning approaches can be divided broadly into metric-learning and kernel-learning approaches. We discuss representative work using these two approaches.

2.1 Metric Learning

Metric learning attempts to find the optimal linear transformation for the given set of data vectors to better characterize the similarity between vectors. The transformation by itself is linear, but the data vectors may first be *explicitly* mapped to a new set of vectors using a nonlinear function $\phi(\mathbf{x})$. The transformation of the data vectors is equivalent to assigning weights to the features of the vectors; therefore, metric learning is often called *feature weighting*. The metric learning approach is given a set of data vectors $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{m}$ in $\Re^n$ and similarity information in the form of $(\mathbf{x}_i, \mathbf{x}_j) \in S$ (a similar set), if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar. Metric learning aims to learn a distance metric $d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$ between data vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ that respects the similarity information. Mathematically the distance metric can be represented as

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T \mathbf{M} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))},$$

where $\mathbf{M}$ needs to be positive (semi-) definite so as to satisfy metric properties—non-negativity and triangle inequality. More generally, $\mathbf{M}$ parameterizes a family of Mahalanobis distances over $\Re^n$. The choice of the basis function ϕ and the scaling matrix $\mathbf{M}$ will differentiate the various metric learning algorithms.

Wettschereck et al. [28] provide a review of the performance of feature-weighting algorithms with emphasis on their performance for the k-nearest neighbor classifier. Here, we discuss only a few representative algorithms. (For the other algorithms, please refer to [28].) A number of papers address the problem of learning distance metrics using contextual information[2] in the form of groups of similar vectors [5, 31]. Contextual information can be user-provided information on the similarity characteristics of a subset of data. Based on this information, the work of [5] uses Relevant Component Analysis (RCA) to efficiently learn a full rank Mahalanobis metric [18]. The authors use equivalence relations for the contextual information. They compute

$$\hat{C} = \frac{1}{p} \sum_{j=1}^{|G|} \sum_{i=1}^{|S_j|} (\mathbf{x}_{ji} - \hat{\mathbf{m}}_j)(\mathbf{x}_{ji} - \hat{\mathbf{m}}_j)^T,$$

where $\hat{\mathbf{m}}_j$ is the mean of the j-th group of vectors and $|G|$ and $|S_j|$ denote the number of groups and the number of samples in the j-th group, respectively. The matrix $W = \hat{C}^{-\frac{1}{2}}$ is used for transformation and the inverse of $\hat{C}$ as the Mahalanobis matrix. Xing

[2]Contextual information is also called side information in some papers such as [5, 31]

et al. [31] treat the same problem as a convex optimization problem, hence producing local-optimum-free algorithms. They present techniques for learning the weighting matrix both for the diagonal and for the full matrix case. The major difference between the two approaches is that RCA uses closed-form expressions, whereas [31] uses iterative methods that can be sensitive to parameter tuning and that are computationally expensive.

Metric learning aims to learn a good distance function by computing the optimal feature weightings in $\mathcal{I}$. Clearly, this linear transformation is restrictive in terms of modeling complex semantics. Although one can perform a non-linear transformation on the features via a basis function ϕ in $\mathcal{I}$, such a transformation is *explicit* and the resulting computational complexity renders this approach impractical. The kernel learning approach, which we discuss next, successfully addresses the concern about computational complexity.

2.2 Kernel Learning

Kernel-based methods attempt to *implicitly* map a set of data vectors $\mathcal{X} = \{\mathbf{x}\}_{i=1}^{m}$ in $\mathcal{I}$ to some other high-dimensional (possibly infinite) projected space $\mathcal{P}$, using a basis function (usually nonlinear) ϕ, where the mapped data can be separated by applying a linear procedure [25]. Kernel function K is defined as an inner product between two vectors in projected space $\mathcal{P}$, $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, as

$$K(\mathbf{x}_i, \mathbf{x}_j) = < \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) > .$$

Kernel-based methods use these inner products (K) as a similarity measure. (Theoretical justifications are presented in [22].) The kernel K provides an elegant way of dealing with nonlinear algorithms by reducing them to linear ones in $\mathcal{P}$. Any algorithm that can be expressed in terms of inner products can be made nonlinear by substituting kernel values for the inner products. A typical example is Support Vector Machines [25].

The requirement for choosing a valid K is that it should be positive (semi-) definite and symmetric. Three popular kernels employed are the polynomial kernel $(< \mathbf{x}_i, \mathbf{x}_j > +1)^p$, the Gaussian radial basis function (RBF) kernel $\exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2})$, and Laplacian RBF kernel $\exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel_1)$. In $\mathcal{P}$, the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ can then be computed via the kernel trick

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)},$$

where the validity of the kernel K ensures that the resulting distance function $d(\mathbf{x}_i, \mathbf{x}_j)$ will be a valid metric.

An important advantage of using kernels lies in the ease of computing the inner product (similarity measure) in $\mathcal{P}$ without actually having to know ϕ.There has been a lot of work [6, 22, 25] in classification, clustering, and regression methods, using the kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ for indirect computations of similarity measures (and hence the distance $d(\mathbf{x}_i, \mathbf{x}_j)$) in $\mathcal{P}$.

Due to the central role of the kernel, a poor kernel function can lead to significantly poor performance using kernel methods. Instead of choosing a pre-defined kernel for training, many recent efforts aim to learn a kernel from the training data [9, 16]. All these papers are based on the notion of *kernel alignment* proposed by Cristianini et al. [9] to measure the similarity between two kernel functions. Geometrically, when given a set of training instances $\mathcal{X}$, the *alignment* score is defined as the cosine of the angle between the two kernel matrices[3], after flattening the two matrices into vectors. They also proposed the notion of "ideal kernel" (K^*). Suppose $y(\mathbf{x}_i) \in \{1, -1\}$ is the class label of $\mathbf{x}_i$. K^* is defined as

$$K^*(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if } y(\mathbf{x}_i) = y(\mathbf{x}_j), \\ 0, & \text{if } y(\mathbf{x}_i) \neq y(\mathbf{x}_j), \end{cases} \quad (2)$$

which is the target kernel that a given kernel is supposed to align with. Kernel-alignment calculates the alignment score of a given kernel K to the ideal kernel K^* to indicate the degree to which the kernel matches the training data. Cristianini et al. [9] prove the connection between the alignment and the generalization performance of the resulting classifier. Basically, their work shows that with a high alignment on the training set, we can expect a good generalization performance of the resulting distance-based classifier.

Since a kernel K defines a pairwise distance from Eqn. 1, kernel learning has been recently applied to distance metric learning. Zhang [32] proposed to idealize a given kernel K using $\tilde{K} = K + K^*$, to achieve a good distance metric which is Euclideanly and Fisherly separable. Then, the achieved distance metric was embedded into a new Euclidean space via Multi-dimensional scaling (MDS) [27]. However, this iterative embedding procedure is computationally expensive, and its idealization model might not be optimal. Kwok et al. [16] considered the same problem of metric learning as a convex optimization problem. The approach works in both the input and the kernel-induced projected spaces. They modify the prior kernel using $\tilde{K} = K + \frac{\gamma}{2}K^*$ and derive distance functions using this modified kernel. Since the number of parameters is related to the number of patterns but not to the dimensionality of the patterns, the approach allows feature weighting to be done efficiently in the possibly infinite dimensional projected spaces. However, their learned distance metric cannot be guaranteed to be positive (semi-) definite. Hence, the induced distance function might not be a valid one. Moreover, their kernel transformation model, $\tilde{K} = K + \frac{\gamma}{2}K^*$, is not an optimal one for kernel idealization because $\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$ when two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar ($K^*(\mathbf{x}_i, \mathbf{x}_j) = 1$), which means the intra-class similarity scores remain unchanged using this transformation model.

Our DA$_{lign}$ algorithm transforms a prior kernel function[4] in a projected space $\mathcal{P}$ when given a set of training data. We theoretically prove that our method leads to a valid distance function. More importantly, we also show that since the optimization is performed in a convex space, the solution obtained is globally optimal, given contextual information.

3. DA$_{lign}$ ALGORITHM

Given the prior kernel function K and contextual information, DA$_{lign}$ transforms K. Kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, as discussed in Section 2.2, can be considered as a similarity measure between instances $\mathbf{x}_i$ and $\mathbf{x}_j$. We assume $0 \leq K(\mathbf{x}_i, \mathbf{x}_j) \leq 1$. A value of 1 indicates that the instances are identical while a value of 0 means that they are completely dissimilar. Commonly used kernels like the Gaussian and the Laplacian are normalized to produce a similarity measure between 0 and 1. The polynomial kernel, though not necessarily normalized, can easily be normalized by using

$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{K(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)K(\mathbf{x}_j, \mathbf{x}_j)}}. \quad (3)$$

[3]Given a kernel function K and a set of instances $\mathcal{X}$, the kernel matrix (Gram matrix) is the matrix of inner-products of all possible pairs from $\mathcal{X} \times X$, $\mathbf{K} = [k_{ij}]$, where $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

[4]The *kernel trick* in Eqn. 1 uniquely links the kernel functions with the distance function. The former (K) provides the pairwise-similarity measurement between two instances, whereas the latter (d) provides the pairwise-dissimilarity measurement between two instances. Therefore, when we say a transformation on a prior kernel function, it also means a transformation on a prior distance function, and vice versa.

The contextual information is represented by sets $\mathcal{S}$ and $\mathcal{D}$, where $\mathcal{S}$ denotes the set of similar pairs of instances, and $\mathcal{D}$ the set of dissimilar pairs of instances. Sets $\mathcal{S}$ and $\mathcal{D}$ can be constructed either directly or indirectly. Directly, users can return the information about whether two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar or dissimilar. In such cases, the similar set $\mathcal{S}$ can be written as $\{(\mathbf{x}_i, \mathbf{x}_j)|\mathbf{x}_i \sim \mathbf{x}_j\}$, and the dissimilar set $\mathcal{D}$ as $\{(\mathbf{x}_i, \mathbf{x}_j)|\mathbf{x}_i \nsim \mathbf{x}_j\}$. Indirectly, we may know only the class-label of instance $\mathbf{x}_i$ as y_i. In this case, we can consider $\mathbf{x}_i$ and $\mathbf{x}_j$ to be a similar pair if $y_i = y_j$, and a dissimilar pair otherwise.

In the remainder of this section, we first propose a transformation model to formulate the contextual information in terms of the prior kernel K (Section 3.1). Next, we discuss methods to generalize the model to compute the distance between unseen instances (Section 3.2).

3.1 Transformation Model

The goal of our transformation is to increase the kernel value for the similar pairs, but decrease the kernel value for the dissimilar pairs. DA*lign* performs transformation in $\mathcal{P}$, to modify the kernel from K to $\tilde{K}$. Let β_1 and β_2 denote the slopes of transformation curves for dissimilar and similar pairs, respectively. For a given $\mathcal{S}$ and $\mathcal{D}$, the kernel matrix $\mathbf{K}$, corresponding to the kernel K, is then modified as follows

$$\tilde{k}_{ij} = \begin{cases} \beta_1 k_{ij}, & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \\ \beta_2 k_{ij} + (1 - \beta_2), & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \end{cases} \tag{4}$$

where $0 \leq \beta_1, \beta_2 \leq 1$ and $\tilde{k}_{ij}$ is the ij-th component of the new kernel matrix $\tilde{\mathbf{K}}$.

In what follows, we prove two important theorems. Theorem 1 demonstrates that under some constraints on β_1 and β_2, our proposed similarity transformation model in Eqn. 4 ensures a valid kernel. Theorem 2 mathematically demonstrates that under the constraints from Theorem 1, the transformed $\tilde{\mathbf{K}}$ in Eqn. 4 guarantees a better alignment to the ideal $\mathbf{K}^*$ in Eqn. 8 than the $\mathbf{K}$ to the $\mathbf{K}^*$.

THEOREM 1. *Under the assumption that $0 \leq \beta_1 \leq \beta_2 \leq 1$, the transformed kernel $\tilde{K}$ is positive (semi-) definite if the prior kernel K is positive (semi-) definite. (The assumption $\beta_1 \leq \beta_2$ means that we place more emphasis on the decreasing similarity value (K) for dissimilar instance-pairs.)*

PROOF. The transformed kernel $\tilde{K}$ can be written as follows:

$$\tilde{K} = \beta_1 K + (\beta_2 - \beta_1) K \odot K^* + (1 - \beta_2) K^*, \tag{5}$$

which corresponds to the kernel matrix $\tilde{\mathbf{K}}$, associated with the training set $\mathcal{X}$, in Eqn. 4. If the prior K is positive (semi-) definite, using the fact that the ideal kernel K^* is positive (semi-) definite [9], we can derive that $\tilde{K}$ in Eqn. 5 is also positive (semi-) definite if $0 \leq \beta_1 \leq \beta_2 \leq 1$. Here, we use the closure properties of kernels, namely that the product and summation of valid kernels also give a valid kernel [22]. ■

THEOREM 2. *The kernel matrix $\tilde{\mathbf{K}}$ of the transformed kernel $\tilde{K}$ obtains a better alignment than the prior kernel matrix $\mathbf{K}$ to the ideal kernel matrix $\mathbf{K}^*$, if $0 \leq \beta_1 \leq \beta_2 \leq 1$. Moreover, a smaller β_1 or β_2 would induce a higher alignment score.*

PROOF. In [16], it has been proven that a kernel with the following form has a higher alignment score with the ideal kernel K^* than with the original kernel K,

$$\overline{K} = K + \gamma K^*, \quad \gamma > 0, \tag{6}$$

where we use $\overline{K}$ to distinguish with our $\tilde{K}$ defined in Eqn. 4. According to the definition of kernel target alignment [9], we have

$$\left[\frac{< \mathbf{K}, \mathbf{K}^* >}{\sqrt{< \mathbf{K}, \mathbf{K} >}}\right]^2 < \left[\frac{< \mathbf{K} + \gamma \mathbf{K}^*, \mathbf{K}^* >}{\sqrt{< \mathbf{K} + \gamma \mathbf{K}^*, \mathbf{K} + \gamma \mathbf{K}^* >}}\right]^2, \tag{7}$$

where the common item $\sqrt{< \mathbf{K}^*, \mathbf{K}^* >}$ is omitted at both sides of inequality, and the Frobenius norm of two matrices, say $\mathbf{M} = [m_{ij}]$ and $\mathbf{N} = [n_{ij}]$, is defined as $< \mathbf{M}, \mathbf{N} >= \sum_{i,j} m_{ij} n_{ij}$.

Cristianini et al. [9] proposed the notion of "ideal kernel" (K^*). Suppose $y(\mathbf{x}_i) \in \{1, -1\}$ is the class label of $\mathbf{x}_i$. K^* is defined as

$$K^*(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if } y(\mathbf{x}_i) = y(\mathbf{x}_j), \\ 0, & \text{if } y(\mathbf{x}_i) \neq y(\mathbf{x}_j), \end{cases} \tag{8}$$

which is the target kernel that a given kernel is supposed to align with. Employing Eqn. 6 and Eqn. 8, we expand (7) as follows

$$\frac{\left(\sum_{\mathcal{S}} k_{ij}\right)^2}{\sum_{\mathcal{S}} k_{ij}^2 + \sum_{\mathcal{D}} k_{ij}^2} < \frac{\left[\sum_{\mathcal{S}} (k_{ij} + \gamma)\right]^2}{\sum_{\mathcal{S}} (k_{ij} + \gamma)^2 + \sum_{\mathcal{D}} k_{ij}^2}. \tag{9}$$

Defining $\gamma = \frac{1-\beta 2}{\beta_2} > 0$, where β_2 is the parameter in Eqn. 4, we then rewrite the right side in (9) as follows

$$\begin{aligned} & \frac{\left[\sum_{\mathcal{S}} (k_{ij} + \frac{1-\beta_2}{\beta_2})\right]^2}{\sum_{\mathcal{S}} (k_{ij} + \frac{1-\beta_2}{\beta_2})^2 + \sum_{\mathcal{D}} k_{ij}^2} \\ = & \frac{\beta_2^2 \left[\sum_{\mathcal{S}} (k_{ij} + \frac{1-\beta_2}{\beta_2})\right]^2}{\beta_2^2 \sum_{\mathcal{S}} (k_{ij} + \frac{1-\beta_2}{\beta_2})^2 + \beta_2^2 \sum_{\mathcal{D}} k_{ij}^2} \\ = & \frac{\left[\sum_{\mathcal{S}} (\beta_2 k_{ij} + (1 - \beta_2))\right]^2}{\sum_{\mathcal{S}} (\beta_2 k_{ij} + (1 - \beta_2))^2 + \beta_2^2 \sum_{\mathcal{D}} k_{ij}^2} \qquad (10) \\ \leq & \frac{\left[\sum_{\mathcal{S}} (\beta_2 k_{ij} + (1 - \beta_2))\right]^2}{\sum_{\mathcal{S}} (\beta_2 k_{ij} + (1 - \beta_2))^2 + \beta_1^2 \sum_{\mathcal{D}} k_{ij}^2} \qquad (11) \\ = & \left[\frac{< \tilde{\mathbf{K}}, \mathbf{K}^* >}{\sqrt{< \tilde{\mathbf{K}}, \tilde{\mathbf{K}} >}}\right]^2, \qquad (12) \end{aligned}$$

where we apply the assumption of $\beta_1 \leq \beta_2$ from (10) to (11), and employ Eqn. 4 in the last step (12). Combining (7) and (12), we obtain

$$\frac{< \mathbf{K}, \mathbf{K}^* >}{\sqrt{< \mathbf{K}, \mathbf{K} >< \mathbf{K}^*, \mathbf{K}^* >}} < \frac{< \tilde{\mathbf{K}}, \mathbf{K}^* >}{\sqrt{< \tilde{\mathbf{K}}, \tilde{\mathbf{K}} >< \mathbf{K}^*, \mathbf{K}^* >}}.$$

Therefore, the transformed kernel $\tilde{\mathbf{K}}$ in Eqn. 4 can achieve a better alignment than the original kernel K under the assumption of $0 \leq \beta_1 \leq \beta_2 \leq 1$. Moreover, a greater γ in Eqn. 6 will have a higher alignment score [16]. Recall that $\beta_2 = \frac{1}{1+\gamma}$. Hence, a smaller β_2 will have a higher alignment. On the other hand, from (11) and (12), we can see that the alignment score of $\tilde{K}$ is a decreasing function w.r.t. β_1. Therefore, a smaller β_1 or β_2 will result in a higher alignment score. ■

For a prior kernel K, the inner product of two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ in $\mathcal{P}$. For simplicity, we denote $\phi(\mathbf{x}_i)$ as ϕ_i. The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{P}$ can thus be computed as $d_{ij}^2 = k_{ii} + k_{jj} - 2k_{ij}$, where $k_{ij} = \phi_i^T \phi_j$. Therefore, for the transformed kernel $\tilde{K}$ in Eqn. 4, the corresponding distance $\tilde{d}_{ij}^2 = \tilde{k}_{ii} + \tilde{k}_{jj} - 2\tilde{k}_{ij}$ in $\mathcal{P}$ can be written in terms of d_{ij}^2 as

$$\tilde{d}_{ij}^2 = \begin{cases} \beta_2 d_{ij}^2, & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \\ \beta_1 d_{ij}^2 + 2(1 - \beta_2) + (\beta_2 - \beta_1)(k_{ii} + k_{jj}), & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}. \end{cases} \tag{13}$$

Since K has been normalized as in Eqn. 3, we have the distance $d_{ij}^2 = 2 - 2k_{ij}$. Eqn. 13 can thus be rewritten as

$$\tilde{d}_{ij}^2 = \begin{cases} \beta_2 d_{ij}^2, & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \\ \beta_1 d_{ij}^2 + 2(1-\beta_1), & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}. \end{cases} \tag{14}$$

Using the property $0 \leq d_{ij} \leq 2$ and the condition $0 \leq \beta_1 \leq \beta_2 \leq 1$, we can see that $\tilde{d}_{ij}^2 \leq d_{ij}^2$ if the two instances are similar, and $\tilde{d}_{ij}^2 \geq d_{ij}^2$ if the two instances are dissimilar. In other words, the transformed distance metric in Eqn. 14 decreases the intra-class pairwise distance in $\mathcal{P}$, and increases the inter-class pairwise distance in $\mathcal{P}$. The developed distance metric (Eqn. 14) is a valid distance metric (non-negativity, symmetry, and triangle inequality) since the transformed kernel in Eqn. 4 is a valid kernel, according to Theorem 1.

3.2 Distance Metric Learning

In this subsection, we show how to generalize the model in Eqn. 14 to unseen instances without overfitting the contextual information. We use the feature-weighting method [13] by modifying the inner product $k_{ij} = \phi_i^T \phi_j$ in $\mathcal{P}$ as $\phi_i^T \mathbf{A}\mathbf{A}^T \phi_j$. Here, $\mathbf{A}_{m' \times m'}$ is the weighting matrix and m' is the dimension of $\mathcal{P}$. Based on the idea of feature reduction [13], we aim to achieve a small rank of $\mathbf{A}$, which means that the dimensionality of feature vector ϕ is kept small in projected space $\mathcal{P}$. The corresponding distance function thus becomes

$$\tilde{d}_{ij}^2 = (\phi_i - \phi_j)^T \mathbf{A}\mathbf{A}^T (\phi_i - \phi_j). \tag{15}$$

To solve for $\mathbf{A}\mathbf{A}^T$ so that the distance metric in Eqn. 15 equals the distance in Eqn. 14, we formulate the problem as a convex optimization problem whose objective function is to minimize the rank of the weighting matrix $\mathbf{A}$. However, it induces an NP-Hard problem by directly minimizing $\mathrm{rank}(\mathbf{A})$, the so-called zero-norm problem in [4]. Since minimizing the trace of a matrix tends to give a low-rank solution when the matrix is symmetric [11], in this paper, we approximate the rank of a matrix by its trace. Moreover, since $\mathrm{rank}(\mathbf{A}\mathbf{A}^T) = \mathrm{rank}(\mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{A}^T) \approx \mathrm{trace}(\mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{A}^T) = \|\mathbf{A}\mathbf{A}^T\|_{\mathrm{F}}^2$, we approximate the problem of minimizing the rank of $\mathbf{A}$ by minimizing its Frobenius norm $\|\mathbf{A}\mathbf{A}^T\|_{\mathrm{F}}^2$. The corresponding primal problem is formulated as

$$\min_{\mathbf{A}\mathbf{A}^T, \beta_1, \beta_2} \quad \frac{1}{2}\|\mathbf{A}\mathbf{A}^T\|_{\mathrm{F}}^2 + C_{\mathcal{D}}\beta_1 + C_{\mathcal{S}}\beta_2, \tag{16}$$

$$\begin{aligned} \text{s.t.} \quad \beta_2 d_{ij}^2 &= \tilde{d}_{ij}^2, && (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ \tilde{d}_{ij}^2 &= \beta_1 d_{ij}^2 + 2(1-\beta_1), && (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ \beta_2 &\geq \beta_1, \\ \beta_1 &\geq 0, \\ 1 - \beta_2 &\geq 0, \end{aligned} \tag{17}$$

where $C_{\mathcal{S}}$ and $C_{\mathcal{D}}$ are two non-negative hyper-parameters. Theorem 2 shows that a large β_1 or β_2 will induce a lower alignment score. However, on the contrary, $\beta_1 = \beta_2 = 0$ would overfit the training dataset. We hence add two penalty terms $C_{\mathcal{D}}\beta_1$ and $C_{\mathcal{S}}\beta_2$ to control the alignment degree. This strategy is similar to that used in Support Vector Machines [25], which limits the length of weight vector $\|\mathbf{w}\|^2$ in projected space $\mathcal{P}$ to combat the overfitting problem.

The constrained optimization problem above can be solved by considering the corresponding Lagrangian formulation

$$\begin{aligned} &\mathcal{L}(\mathbf{A}\mathbf{A}^{\mathbf{T}}, \beta_1, \beta_2, \alpha_v, \mu, \gamma, \pi) \\ &= \frac{1}{2}\|\mathbf{A}\mathbf{A}^{\mathbf{T}}\|_F^2 + C_{\mathcal{D}}\beta_1 + C_{\mathcal{S}}\beta_2 \\ &- \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} \alpha_{ij} \left[(\phi_i - \phi_j)^T (\beta_2 \mathbf{I} - \mathbf{A}\mathbf{A}^T)(\phi_i - \phi_j) \right] \\ &- \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij} \left[(\phi_i - \phi_j)^T (\mathbf{A}\mathbf{A}^T - \beta_1 \mathbf{I})(\phi_i - \phi_j) - 2(1-\beta_1) \right] \\ &- \mu\beta_1 - \gamma(\beta_2 - \beta_1) - \pi(1 - \beta_2), \end{aligned} \tag{18}$$

where the Lagrangian multipliers (dual variables) are $\boldsymbol{\alpha}_i, \mu, \gamma, \pi \geq 0$. This function has to be minimized w.r.t. the primal variables $\mathbf{A}\mathbf{A}^{\mathbf{T}}$, β_1, β_2, and maximized w.r.t. the dual variables $\boldsymbol{\alpha}$, μ, γ, π. To eliminate the primal variables, we set the corresponding partial derivatives to be zero, obtaining the following conditions:

$$\begin{aligned} \mathbf{A}\mathbf{A}^{\mathbf{T}} = &\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij} (\phi_i - \phi_j)(\phi_i - \phi_j)^T \\ &- \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} \alpha_{ij} (\phi_i - \phi_j)(\phi_i - \phi_j)^T, \end{aligned} \tag{19}$$

$$C_{\mathcal{S}} + \pi - \gamma = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} \alpha_{ij} (\phi_i - \phi_j)^T (\phi_i - \phi_j), \tag{20}$$

$$C_{\mathcal{D}} + \gamma - \mu = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij} \left[2 - (\phi_i - \phi_j)^T (\phi_i - \phi_j) \right], \tag{21}$$

Substituting the conditions of (20) and (21) into (18), we obtain the following dual formulation

$$\begin{aligned} \mathcal{W}(\boldsymbol{\alpha}, \pi) = &\frac{1}{2}\|\mathbf{A}\mathbf{A}^{\mathbf{T}}\|_F^2 + 2 \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij} \\ &- \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij} (\phi_i - \phi_j)^T \mathbf{A}\mathbf{A}^T (\phi_i - \phi_j) \\ &+ \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} \alpha_{ij} (\phi_i - \phi_j)^T \mathbf{A}\mathbf{A}^T (\phi_i - \phi_j) \\ &- \pi, \end{aligned} \tag{22}$$

which has to be maximized w.r.t. α_{ij}'s and $\pi \geq 0$. Actually, π can be removed from the dual function (22), since $\frac{\partial \mathcal{W}(\boldsymbol{\alpha}, \pi)}{\partial \pi} = -1 < 0$, which means that $\mathcal{W}(\boldsymbol{\alpha}, \pi)$ is a decreasing function w.r.t π. Hence, $\mathcal{W}(\boldsymbol{\alpha}, \pi)$ is maximal at $\pi = 0$.

Now, the dual formulation (22) becomes a convex quadratic function w.r.t. only α_{ij}. Next, we examine the constraints of dual formulation on α_{ij}. According to the KKT theorem [15], we have the following conditions

$$\alpha_{ij} \left[\beta_2 d_{ij}^2 - \tilde{d}_{ij}^2 \right] = 0, \quad (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \tag{23}$$

$$\alpha_{ij} \left[\tilde{d}_{ij}^2 - \beta_1 d_{ij}^2 - 2(1-\beta_1) \right] = 0, \quad (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \tag{24}$$

$$\gamma(\beta_2 - \beta_1) = 0, \tag{25}$$

$$\mu\beta_1 = 0, \tag{26}$$

$$\pi(1 - \beta_2) = 0. \tag{27}$$

The constraint (17) requires $\beta_2 \geq \beta_1$. In the case of $\beta_1 = \beta_2 = 0$, the training dataset would be overfitted. In addition, in the case of $\beta_1 = \beta_2 = 1$, $\tilde{d}_{ij}^2$ is exactly equal to the original distance metric d_{ij}^2 but we do not get any improvement. To avoid these cases, we then change (17) to be a strict inequality constraint of $\beta_2 > \beta_1$.

Therefore, we have $\gamma = 0$ from (25). Using the properties of $\gamma = 0$ and $\pi = 0$, we can then change the dual formulation (22) and its constraints of (20) and (21) by substituting the expansion form of $\mathbf{AA^T}$ in (19) as follows:

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \quad \mathcal{W}(\boldsymbol{\alpha}) \qquad (28)\\
&= 2 \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}\\
&- \frac{1}{2} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{S}} \alpha_{ij}\alpha_{kl} \left((\phi_i - \phi_j)^T(\phi_k - \phi_l)\right)^2\\
&- \frac{1}{2} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{D}} \alpha_{ij}\alpha_{kl} \left((\phi_i - \phi_j)^T(\phi_k - \phi_l)\right)^2\\
&+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{D}} \alpha_{ij}\alpha_{kl} \left((\phi_i - \phi_j)^T(\phi_k - \phi_l)\right)^2,
\end{aligned}$$

$$\text{s.t.} \quad \begin{cases} C_{\mathcal{S}} & = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\phi_i - \phi_j)^T(\phi_i - \phi_j), \\ C_{\mathcal{D}} & \geq \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij} \left[2 - (\phi_i - \phi_j)^T(\phi_i - \phi_j)\right], \\ 0 & \leq \alpha_{ij}. \end{cases}$$

where $\phi_i^T\phi_j = K(\mathbf{x}_i, \mathbf{x}_j)$ from the kernel trick. The dual formulation (28) is very similar to that of C-SVMs [25]. It is a standard convex quadratic programming, which can result in a global optimal solution without any local minima.

After solving the convex quadratic optimization problem in (28), we can then generalize our distance-metric model defined in Eqn. 19 to the unseen test instances. Suppose $\mathbf{x}$ and $\mathbf{x}'$ are two test instances with unknown class labels. Their pairwise distance $\tilde{d}^2(\mathbf{x}, \mathbf{x}')$ after feature weighting is computed as

$$\tilde{d}^2(\mathbf{x}, \mathbf{x}') = \left(\phi(\mathrm{x}) - \phi(\mathrm{x}')\right)^T \mathbf{AA}^T \left(\phi(\mathrm{x}) - \phi(\mathrm{x}')\right).$$

Substituting (19) into the above equation, we obtain

$$\begin{aligned}
&\tilde{d}^2(\mathbf{x}, \mathbf{x}') \qquad (29)\\
&= \left(\phi(\mathrm{x}) - \phi(\mathrm{x}')\right)^T \left(\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(\phi_i - \phi_j)(\phi_i - \phi_j)^T \right.\\
&\quad \left. - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\phi_i - \phi_j)(\phi_i - \phi_j)^T \right) \left(\phi(\mathrm{x}) - \phi(\mathrm{x}')\right)\\
&= \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(K_{\mathrm{x}\mathrm{x}_i} - K_{\mathrm{x}\mathrm{x}_j} - K_{\mathrm{x}_i\mathrm{x}'} + K_{\mathrm{x}_j\mathrm{x}'})^2\\
&\quad - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(K_{\mathrm{x}\mathrm{x}_i} - K_{\mathrm{x}\mathrm{x}_j} - K_{\mathrm{x}_i\mathrm{x}'} + K_{\mathrm{x}_j\mathrm{x}'})^2.
\end{aligned}$$

Remark: We note that here our learned distance function $\tilde{d}^2(\mathbf{x}, \mathbf{x}')$ is expressed in terms of the prior kernel K which has been chosen before we apply the algorithm. For example, such a prior kernel could be chosen as a Gaussian RBF function $\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\sigma^2}\right)$. $K_{\mathrm{x}\mathrm{x}_i}$ and $K_{\mathrm{x}\mathrm{x}_j}$ in Eqn. 29 can thus be computed as $\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|_2^2}{2\sigma^2}\right)$ and $\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_j\|_2^2}{2\sigma^2}\right)$, and so are $K_{\mathrm{x}_i\mathrm{x}'}$ and $K_{\mathrm{x}_j\mathrm{x}'}$. Therefore, even in the case where both $\mathbf{x}$ and $\mathbf{x}'$ are unseen test instances, their pairwise distance can still be calculated from Eqn. 29. Moreover, when a linear kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = < \mathbf{x}_i, \mathbf{x}_j >$, is employed, the projected space $\mathcal{P}$ is exactly equivalent to the original input space $\mathcal{I}$. DA$_{lign}$ then becomes a distance-function-learning in $\mathcal{I}$. Therefore, DA$_{lign}$ is a general algorithm which can learn a distance function in both $\mathcal{P}$ and $\mathcal{I}$. ■

4. DECOMPOSITION-BASED SOLUTION

The dual formulation in (28) is a typical convex *quadratic programming* (QP) problem. Usually, such a convex optimization problem can be effectively solved using the primal-dual interior-point method. However, the computational cost of ($O(n^3)$) is considerably high. We propose an SMO-like decomposition-based method [20] for speeding up DA$_{lign}$. Suppose we have M pairs of $(\mathbf{x}_i, \mathbf{x}_j)$ in $\mathcal{S}$ and N pairs of $(\mathbf{x}_k, \mathbf{x}_l)$ in $\mathcal{D}$. We denote their corresponding lagrangian multiplier $\boldsymbol{\alpha}$ vector as $\boldsymbol{\alpha}_{\mathcal{S}} = [\alpha_{\mathcal{S}_1}, ..., \alpha_{\mathcal{S}_M}]^T$ and $\boldsymbol{\alpha}_{\mathcal{D}} = [\alpha_{\mathcal{D}_1}, ..., \alpha_{\mathcal{D}_N}]^T$. Moreover, we define three matrices, $\mathbf{Q}_{\mathcal{S}}$ (M-by-M), $\mathbf{Q}_{\mathcal{D}}$ (N-by-N), and $\mathbf{Q}_{\mathcal{SD}}$ (M-by-N), whose uv-th components are as follows:

$$\begin{aligned}
(\mathbf{Q}_{\mathcal{S}})_{uv} &= \left(\Delta\phi_{s_u}^T \Delta\phi_{s_v}\right)^2 = \left((\phi_i - \phi_j)^T(\phi_k - \phi_l)\right)^2,\\
&\text{where } s_u = (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, s_v = (\mathbf{x}_k, \mathbf{x}_l) \in \mathcal{S}\\
(\mathbf{Q}_{\mathcal{D}})_{uv} &= \left(\Delta\phi_{d_u}^T \Delta\phi_{d_v}\right)^2 = \left((\phi_i - \phi_j)^T(\phi_k - \phi_l)\right)^2,\\
&\text{where } d_u = (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, d_v = (\mathbf{x}_k, \mathbf{x}_l) \in \mathcal{D}\\
(\mathbf{Q}_{\mathcal{SD}})_{uv} &= \left(\Delta\phi_{s_u}^T \Delta\phi_{d_v}\right)^2 = \left((\phi_i - \phi_j)^T(\phi_k - \phi_l)\right)^2,\\
&\text{where } s_u = (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, d_v = (\mathbf{x}_k, \mathbf{x}_l) \in \mathcal{D}.
\end{aligned}$$

$\mathcal{W}(\boldsymbol{\alpha})$ in (28) can thus be formulated in matrix form as follows:

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad \mathcal{W}(\boldsymbol{\alpha}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{D}}) =& 2\boldsymbol{\alpha}_{\mathcal{D}}^T\mathbf{1}_{N\times 1} - \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{S}}^T\mathbf{Q}_{\mathcal{S}}\boldsymbol{\alpha}_{\mathcal{S}} - \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{D}}^T\mathbf{Q}_{\mathcal{D}}\boldsymbol{\alpha}_{\mathcal{D}}\\
&+ \boldsymbol{\alpha}_{\mathcal{S}}^T\mathbf{Q}_{\mathcal{SD}}\boldsymbol{\alpha}_{\mathcal{D}} \qquad (30)\\
s.t. \quad & \boldsymbol{\alpha}_{\mathcal{S}}^T\mathbf{n}_{\phi\mathcal{S}} = C_{\mathcal{S}},\\
& \boldsymbol{\alpha}_{\mathcal{D}}^T(2 - \mathbf{n}_{\phi\mathcal{D}}) \leq C_{\mathcal{D}}, \qquad (31)\\
& \boldsymbol{\alpha}_{\mathcal{S}} \geq \mathbf{0}, \ \ \boldsymbol{\alpha}_{\mathcal{D}} \geq \mathbf{0},
\end{aligned}$$

where

$$\begin{aligned}
(\mathbf{n}_{\phi\mathcal{S}})_{\mathbf{s_u}} &= (\phi_i - \phi_j)^T(\phi_i - \phi_j), \ s_u = (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}\\
(\mathbf{n}_{\phi\mathcal{D}})_{\mathbf{d_u}} &= (\phi_i - \phi_j)^T(\phi_i - \phi_j), \ d_u = (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}.
\end{aligned}$$

Next, we select a set of s_u's from $\mathcal{S}$ and d_v's from $\mathcal{D}$ into the working set B, and all others into the non-working N. Denoting the sizes of s_u's and d_v's as I and J, respectively, we decompose the original QP problem with $(N+M)$ variables into a series of $(I+J)$ smaller QP problems. This is exactly how the decomposition-based methods, e.g., LibSVM and SVMLight, work to solve large-scale SVMs. Moreover, we will show that when $I = 2$ and $J = 1$, such a decomposition strategy becomes the version of SMO for DA$_{lign}$.

Let us permutate all vectors and matrices by moving the working set B ahead of N, e.g.,

$$\mathbf{Q}_{\mathcal{S}} = \begin{bmatrix} \mathbf{Q}_{\mathcal{S}_{BB}} & \mathbf{Q}_{\mathcal{S}_{BN}} \\ \mathbf{Q}_{\mathcal{S}_{NB}} & \mathbf{Q}_{\mathcal{S}_{NN}} \end{bmatrix}.$$

We can rewrite (30) as follows after removing all items only de-

pending on the non-working set N:

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \mathcal{W}(\boldsymbol{\alpha}_{\mathcal{S}_B}, \boldsymbol{\alpha}_{\mathcal{D}_B}) && (32)\\
=& -\frac{1}{2}\boldsymbol{\alpha}_{\mathcal{S}_B}^T \mathbf{Q}_{\mathcal{S}_{BB}} \boldsymbol{\alpha}_{\mathcal{S}_B} - \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{D}_B}^T \mathbf{Q}_{\mathcal{D}_B} \boldsymbol{\alpha}_{\mathcal{D}_B}\\
&+ \boldsymbol{\alpha}_{\mathcal{S}_B}^T (\mathbf{Q}_{\mathcal{S}\mathcal{D}_{BB}} \boldsymbol{\alpha}_{\mathcal{D}_B} + \mathbf{Q}_{\mathcal{S}\mathcal{D}_{BN}} \boldsymbol{\alpha}_{\mathcal{D}_N}^{old} - \mathbf{Q}_{\mathcal{S}_{BN}} \boldsymbol{\alpha}_{\mathcal{S}_N}^{old})\\
&+ \boldsymbol{\alpha}_{\mathcal{D}_B}^T (2 * \mathbf{1}_{\mathcal{D}_B} + \mathbf{Q}_{\mathcal{S}\mathcal{D}_{NB}}^T \boldsymbol{\alpha}_{\mathcal{S}_N}^{old} - \mathbf{Q}_{\mathcal{D}_{BN}} \boldsymbol{\alpha}_{\mathcal{D}_N}^{old})\\
s.t. \quad & \boldsymbol{\alpha}_{\mathcal{S}_B}^T \mathbf{n}_{\phi\mathcal{S}_\mathbf{B}} = (\boldsymbol{\alpha}_{\mathcal{S}_B}^{old})^T \mathbf{n}_{\phi\mathcal{S}_\mathbf{B}}, && (33)\\
& \boldsymbol{\alpha}_{\mathcal{D}_B}^T (2 - \mathbf{n}_{\phi\mathcal{D}_\mathbf{B}}) \le C_{\mathcal{D}} - (\boldsymbol{\alpha}_{\mathcal{D}_N}^{old})^T (2 - \mathbf{n}_{\phi\mathcal{D}_\mathbf{N}}), && (34)\\
& \boldsymbol{\alpha}_{\mathcal{S}_B} \ge \mathbf{0},\\
& \boldsymbol{\alpha}_{\mathcal{D}_B} \ge \mathbf{0}.
\end{aligned}$$

Next, we develop SMO for solving (32). From the linear equality constraint in (33) and the linear inequality constraint in (34), we can see that the minimal size of B would be two from $\mathcal{S}$ and one from $\mathcal{D}$. Without loss of generality, we denote them as $\alpha_{\mathcal{S}_1}$, $\alpha_{\mathcal{S}_2}$, and $\alpha_{\mathcal{D}_1}$. The corresponding decomposed form of $\mathcal{W}(\boldsymbol{\alpha}_{\mathcal{S}_B}, \boldsymbol{\alpha}_{\mathcal{D}_B})$ can then be organized as follows:

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \mathcal{W}(\alpha_{\mathcal{S}_1}, \alpha_{\mathcal{S}_2}, \alpha_{\mathcal{D}_1}) && (35)\\
=& 2\alpha_{\mathcal{D}_1} - \frac{1}{2}\alpha_{\mathcal{S}_1}^2 (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1})^2 - \frac{1}{2}\alpha_{\mathcal{S}_2}^2 (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2})^2\\
&- \frac{1}{2}\alpha_{\mathcal{D}_1}^2 (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_1})^2 + \alpha_{\mathcal{S}_1}\alpha_{\mathcal{S}_2} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_2})^2\\
&+ \alpha_{\mathcal{S}_1}\alpha_{\mathcal{D}_1} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{D}_1})^2 + \alpha_{\mathcal{S}_2}\alpha_{\mathcal{D}_1} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2\\
&- \alpha_{\mathcal{S}_1}\Theta_{\mathcal{S}_1} - \alpha_{\mathcal{S}_2}\Theta_{\mathcal{S}_2} + \alpha_{\mathcal{D}_1}\Theta_{\mathcal{D}_1},
\end{aligned}$$

where

$$\begin{aligned}
\Theta_{\mathcal{S}_1} &= \sum_{u=3}^{M} \alpha_{\mathcal{S}_u}^{old} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_u})^2 - \sum_{v=2}^{N} \alpha_{\mathcal{D}_v}^{old} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{D}_v})^2\\
\Theta_{\mathcal{S}_2} &= \sum_{u=3}^{M} \alpha_{\mathcal{S}_u}^{old} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_u})^2 - \sum_{v=2}^{N} \alpha_{\mathcal{D}_v}^{old} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_v})^2\\
\Theta_{\mathcal{D}_1} &= \sum_{u=3}^{M} \alpha_{\mathcal{S}_u}^{old} (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{S}_u})^2 - \sum_{v=2}^{N} \alpha_{\mathcal{D}_v}^{old} (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_v})^2.
\end{aligned}$$

We can see that the formulation of (35) depends on only $\alpha_{\mathcal{S}_1}$, $\alpha_{\mathcal{S}_2}$, and $\alpha_{\mathcal{D}_1}$. It can be analytically solved. Moreover, $\Theta_{\mathcal{S}_1}$, $\Theta_{\mathcal{S}_2}$, and $\Theta_{\mathcal{D}_1}$ can be also computed from $(\mathbf{A}\mathbf{A}^T)^{old}$ using $\boldsymbol{\alpha}_{\mathcal{S}}^{old}$ and $\boldsymbol{\alpha}_{\mathcal{D}}^{old}$. Let us rewrite $(\mathbf{A}\mathbf{A}^T)^{old}$ in (21) using the new notations.

$$(\mathbf{A}\mathbf{A}^T)^{old} = -\sum_{u=1}^{M} \alpha_{\mathcal{S}_u}^{old} \Delta\phi_{\mathcal{S}_u} \Delta\phi_{\mathcal{S}_u}^T - \sum_{v=1}^{N} \alpha_{\mathcal{D}_v}^{old} \Delta\phi_{\mathcal{D}_v} \Delta\phi_{\mathcal{D}_v}^T.$$

$\Theta_{\mathcal{S}_1}$, $\Theta_{\mathcal{S}_2}$, and $\Theta_{\mathcal{D}_1}$ can thus be computed as follows:

$$\begin{aligned}
\Theta_{\mathcal{S}_1} =& -\Delta\phi_{\mathcal{S}_1}^T (\mathbf{A}\mathbf{A}^T)^{old} \Delta\phi_{\mathcal{S}_1} - \alpha_{\mathcal{S}_1}^{old} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1})^2\\
&- \alpha_{\mathcal{S}_2}^{old} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_1})^2 + \alpha_{\mathcal{D}_1}^{old} (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{S}_1})^2\\
\Theta_{\mathcal{S}_2} =& -\Delta\phi_{\mathcal{S}_2}^T (\mathbf{A}\mathbf{A}^T)^{old} \Delta\phi_{\mathcal{S}_2} - \alpha_{\mathcal{S}_1}^{old} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_2})^2\\
&- \alpha_{\mathcal{S}_2}^{old} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2})^2 + \alpha_{\mathcal{D}_1}^{old} (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{S}_2})^2\\
\Theta_{\mathcal{D}_1} =& -\Delta\phi_{\mathcal{D}_1}^T (\mathbf{A}\mathbf{A}^T)^{old} \Delta\phi_{\mathcal{D}_1} - \alpha_{\mathcal{S}_1}^{old} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{D}_1})^2\\
&- \alpha_{\mathcal{S}_2}^{old} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2 + \alpha_{\mathcal{D}_1}^{old} (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_1})^2.
\end{aligned}$$

The optimal $\alpha_{\mathcal{S}_1}$, $\alpha_{\mathcal{S}_2}$, and $\alpha_{\mathcal{D}_1}$ are achieved by maximizing $\mathcal{W}(\alpha_{\mathcal{S}_1}, \alpha_{\mathcal{S}_2}, \alpha_{\mathcal{D}_1})$, with the constraints of

$$\begin{aligned}
\gamma &= \alpha_{\mathcal{S}_1} \Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1} + \alpha_{\mathcal{S}_2} \Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2} && (36)\\
&= \alpha_{\mathcal{S}_1}^{old} \Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1} + \alpha_{\mathcal{S}_2}^{old} \Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2}\\
C_{\mathcal{D}} &\ge \alpha_{\mathcal{D}_1} (2 - \Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_1}) + \sum_{v=2}^{N} \alpha_{\mathcal{D}_v} (2 - \Delta\phi_{\mathcal{D}_v}^T \Delta\phi_{\mathcal{D}_v}) && (37)\\
0 &\le \alpha_{\mathcal{S}_1}, \alpha_{\mathcal{S}_2}, \alpha_{\mathcal{D}_1}. && (38)
\end{aligned}$$

It can thus be induced that the range of three variables must be

$$\begin{aligned}
0 \le \quad & \alpha_{\mathcal{S}_1} \le \frac{\gamma}{\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1}},\\
0 \le \quad & \alpha_{\mathcal{D}_1} \le \frac{C_{\mathcal{D}} - \sum_{v=2}^{N} \alpha_{\mathcal{D}_v} (2 - \Delta\phi_{\mathcal{D}_v}^T \Delta\phi_{\mathcal{D}_v})}{2 - \Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_1}},\\
& \alpha_{\mathcal{S}_2} = \frac{\gamma - \alpha_{\mathcal{S}_1} \Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1}}{\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2}}. && (39)
\end{aligned}$$

Moreover, we have $\frac{\partial \mathcal{W}(\alpha_{\mathcal{S}_1}, \alpha_{\mathcal{D}_1})}{\partial \alpha_{\mathcal{S}_1}} = 0$ and $\frac{\partial \mathcal{W}(\alpha_{\mathcal{S}_1}, \alpha_{\mathcal{D}_1})}{\partial \alpha_{\mathcal{D}_1}} = 0$. Solving two linear equations of vanishing gradient, and utilizing (39), we can therefore solve $\alpha_{\mathcal{S}_1}$, $\alpha_{\mathcal{S}_2}$, and $\alpha_{\mathcal{D}_1}$. Their analytical forms are

$$\begin{aligned}
\alpha_{\mathcal{S}_1} &= \frac{\mathbf{O}_{\mathcal{S}_1}}{\mathbf{T}_{\mathcal{S}_1}},\\
\alpha_{\mathcal{S}_2} &= \mathbf{O}_{\mathcal{S}_2} + \alpha_{\mathcal{S}_1} \mathbf{T}_{\mathcal{S}_1},\\
\alpha_{\mathcal{D}_1} &= \mathbf{O}_{\mathcal{D}_1} + \alpha_{\mathcal{S}_1} \mathbf{T}_{\mathcal{D}_1},
\end{aligned}$$

where the parameters are calculated as

$$\begin{aligned}
\mathbf{O}_{\mathcal{D}_1} =& 2 + \Theta_{\mathcal{D}_1} + \frac{\gamma (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2}{\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2}}\\
\mathbf{T}_{\mathcal{D}_1} =& (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{D}_1})^2 - \frac{\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1}}{\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2}} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2\\
\mathbf{O}_{\mathcal{S}_2} =& \frac{\gamma}{\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2}}\\
\mathbf{T}_{\mathcal{S}_2} =& -\frac{\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1}}{\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2}}\\
\mathbf{O}_{\mathcal{S}_1} =& 2\mathbf{T}_{\mathcal{D}_1} - \gamma \Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1} + \mathbf{O}_{\mathcal{S}_2} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_2})^2\\
&- \mathbf{O}_{\mathcal{D}_1} \mathbf{T}_{\mathcal{D}_1} (\Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_1})^2 + \mathbf{O}_{\mathcal{D}_1} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{D}_1})^2\\
&+ \mathbf{O}_{\mathcal{D}_1} \mathbf{T}_{\mathcal{S}_2} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2 + \mathbf{O}_{\mathcal{S}_2} \mathbf{T}_{\mathcal{D}_1} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2\\
&- \Theta_{\mathcal{S}_1} - \mathbf{T}_{\mathcal{S}_2} \Theta_{\mathcal{S}_2} + \mathbf{T}_{\mathcal{D}_1} \Theta_{\mathcal{D}_1}\\
\mathbf{T}_{\mathcal{S}_1} =& (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_1})^2 + (\mathbf{T}_{\mathcal{S}_2} \Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{S}_2})^2\\
&- 2\mathbf{T}_{\mathcal{S}_2} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{S}_2})^2 + (\mathbf{T}_{\mathcal{D}_1} \Delta\phi_{\mathcal{D}_1}^T \Delta\phi_{\mathcal{D}_1})^2\\
&- 2\mathbf{T}_{\mathcal{D}_1} (\Delta\phi_{\mathcal{S}_1}^T \Delta\phi_{\mathcal{D}_1})^2 - 2\mathbf{T}_{\mathcal{S}_2} \mathbf{T}_{\mathcal{D}_1} (\Delta\phi_{\mathcal{S}_2}^T \Delta\phi_{\mathcal{D}_1})^2.
\end{aligned}$$

Figure 2 provides a high-level summary of the decomposition algorithm.

5. EXPERIMENTAL EVALUATION

We conducted empirical studies to examine the effectiveness and efficiency of our context-based distance-function learning algorithm in three aspects.

1. *Contextual information.* We compared the quality of performance for our learned distance function when given qualitatively different contextual information for learning. We were especially

Input : $\mathcal{S}$, $\mathcal{D}$, and the associated $\mathbf{K}$.
Output: $\boldsymbol{\alpha}_{\mathcal{S}}$ and $\boldsymbol{\alpha}_{\mathcal{D}}$
Repeat:

1. Set $\boldsymbol{\alpha}_{\mathcal{S}}^{(1)}$ and $\boldsymbol{\alpha}_{\mathcal{D}}^{(1)}$ as the initial feasible solution to (30) and let $k = 1$.
2. Return if $\boldsymbol{\alpha}_{\mathcal{S}}^{(k)}$ and $\boldsymbol{\alpha}_{\mathcal{D}}^{(k)}$ is an optimal solution of (30). Otherwise, find an I-$element$ working set $\mathcal{S}_B = \{1, \cdots, I\} \subset \{1, \cdots, N\}$, and a J-$element$ working set $\mathcal{D}_B = \{1, \cdots, J\} \subset \{1, \cdots, M\}$. Denote all other $\boldsymbol{\alpha}_{\mathcal{S}_u}$'s and $\boldsymbol{\alpha}_{\mathcal{D}_v}$'s as non-working set $\mathcal{S}_N$ and $\mathcal{D}_N$.
3. If $|\mathcal{S}_B| = 2$ and $|\mathcal{D}_B| = 1$, solve $(\boldsymbol{\alpha}_{\mathcal{S}_B}^{(k+1)}, \boldsymbol{\alpha}_{\mathcal{D}_B}^{(k+1)})$ using SMO on (35). Otherise, solve them using the the regular QP solver on (32).
4. Set $\boldsymbol{\alpha}_{\mathcal{S}}^{(k+1)} = ((\boldsymbol{\alpha}_{\mathcal{S}_B}^{(k+1)}, \boldsymbol{\alpha}_{\mathcal{S}_N}^{(k)}))$ and $\boldsymbol{\alpha}_{\mathcal{D}}^{(k+1)} = ((\boldsymbol{\alpha}_{\mathcal{D}_B}^{(k+1)}, \boldsymbol{\alpha}_{\mathcal{D}_N}^{(k)}))$. Set $k = k + 1$ and go to step 2.

End

Figure 2: Decomposition-based Solution to DA_{lign}

interested in finding out the quality of contextual information generated by our proposed unsupervised method.

2. *Speedup evaluation*. We evaluated the speedup that can be achieved by our proposed SMO-like decomposition algorithm.

2. *Function-learning performance*. We compared our DA_{lign} algorithm with the regular Euclidean metric, Kwok et al. [16], and Xing et al.'s [31] on clustering applications.

To conduct our experiments, we used three datasets: one toy (mixed) dataset, one video surveillance dataset, and one $2K$ image dataset, which are described as follows:

Mixed dataset The *mixed* artificial dataset was first introduced in [16]. We used it to compare the effectiveness of our method to other methods. This dataset has two classes and eleven features. The first feature of the first class was generated by using the Gaussian distribution $N(3, 1)$, and the first feature of the second class by $N(-3, 1)$; the other ten features were generated by $N(0, 25)$. Each feature was generated independently. The second row of Table 1 provides the detailed composition of the *mixed* dataset.

Video-surveillance dataset The *surveillance* dataset was gathered from 43 video clips from our video-surveillance project [30]. Each clip tracks a person wearing one of eleven shirt colors (*red*, *blue*, *black*, *green*, *pink*, *yellow*, *orange*, *brown*, *purple*, *white*, and *grey*), walking under various lighting conditions. The goal of the project was to correctly name the color of a shirt captured under different lighting conditions and in different cameras. We used a tracker to locate the locations of a shirt in video frames, and then characterized the shirt-color with a set of color histograms. We then trained a classifier and used the classifier to identify the color of a previously unseen shirt.

Image dataset The image dataset was collected from the Corel Image CDs. Corel images have been widely used by the computer vision and image-processing research communities for conducting various experiments. This set contains 2K representative images from fourteen categories: *architecture, bears, clouds, elephants, fabrics, fireworks, flowers, food, landscape, people, textures, tigers, tools,* and *waves*. Each image is represented by a vector of 144 dimensions including color, texture, and shape features [24].

We compared four distance functions in the experiments: our distance-function-alignment algorithm (DA_{lign}), the Euclidean distance function (Euclidean), the method developed by Kwok et al. [16] (Kwok), and the method developed by Xing et al. [31]. The latter two methods are presented in Section 2. We chose the methods of

dataset	# dim	# class	# instance	σ	γ
mixed	11	2	1,000	30.0	0.001
color	11	11	2,130	3.0	0.004
image	144	14	1,897	10.0	0.003

Table 1: Datasets Description.

Amount		Without-SMO	With-SMO
CPU time (seconds)	*mixed*	2,780	221
	color	10,314	606
	image	12,153	782
Clustering error (percents)	*mixed*	42.43	42.43
	color	5.26	5.30
	image	11.10	11.10

Table 2: Clustering performance of DA_{lign} with different strategy on selecting contextual information.

Kwok et al. and Xing et al. for two reasons. First, both are based on contextual information to learn a distance function, as is used in DA_{lign}. Second, the method of Xing et al. is a typical distance-function-learning algorithm in input space $\mathcal{I}$ that been seen as a yardstick method for learning distance functions, as mentioned in Section 2.1. The method of Kwok et al. is a new distance-function-learning algorithm in projected space $\mathcal{P}$ developed recently [16]. We compared DA_{lign} to both methods to test its effectiveness on learning a distance function.

Our evaluation procedure was as follows: First, we chose a prior kernel and derived a distance function via the kernel trick. We then ran DA_{lign}, Kwok's, and Xing's[5] algorithms, respectively, to learn a new distance function. Finally, we ran k-means using the prior and the learned distance functions, and compared their results. Three prior kernels we used in the experiments are linear ($\mathbf{x}^T\mathbf{x}'$), Gaussian ($\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\sigma^2}\right)$), and Laplacian ($\exp\left(-\gamma\|\mathbf{x}-\mathbf{x}'\|_1\right)$). For each dataset, we carefully tuned kernel parameters including σ, γ, $C_{\mathcal{S}}$ and $C_{\mathcal{D}}$ via cross-validation, where σ and γ were reported in Table 1. All measurements were averaged over 20 runs.

5.1 Evaluation on Contextual Information

We examined three different schemes for choosing contextual information for the k-means classifier. The first scheme, denoted as *random*, randomly samples a subset of contextual information from $\mathcal{S}$ and $\mathcal{D}$ sets.

The second scheme chooses the most uncertain boundary instances around decision boundary as contextual information. One way to find such uncertain instances is to run Support Vector Machines (SVMs) to identify the instances along the class boundary (support vectors), and sample these boundary instances to construct contextual information. Some other strategies can also be employed to help select the boundary instances. In this paper, we denote such a scheme as *SV*. We chose SVMs because it is easy to identify the boundary instances (support vectors).

[5]Xing's algorithm cannot be run when the dimensionality of $\mathcal{I}$ is very high or when nonlinear kernels, such as Gaussian and Laplacian, are employed. This is because its computational time does not scale well with the high dimensionality of input space and nonlinear kernels [16]. We thus did not report the corresponding results in these cases.

Dataset	Kernel	*random*	*SV*	*spectral*
mixed	Linear	48.27	47.02	**47.02**
	Gaussian	45.54	43.00	**42.43**
	Laplacian	36.79	33.22	**32.57**
color	Linear	8.10	7.10	**7.00**
	Gaussian	7.84	6.00	**5.26**
	Laplacian	7.54	5.70	**5.50**
image	Linear	14.00	13.00	**13.00**
	Gaussian	13.00	10.20	**10.10**
	Laplacian	11.57	9.87	**9.38**

Table 3: Clustering Error Rates with different strategy of selecting contextual information.

The third scheme, denoted as *spectral*, utilizes an unsupervised way to find such uncertain instances. As mentioned in Section 1 and in [8], we run several different unsupervised algorithm to identify the pairs of instances which are always clustered into one cluster as $\mathcal{S}$, and the pairs which are always clustered into different clusters as $\mathcal{D}$. In our experiments, we applied various spectral clustering algorithms on each dataset, and identified the appropriate pairs of instances into $\mathcal{S}$ and $\mathcal{D}$. Specifically, we ran the normalized-cut spectral algorithm by Shi and Malik (SM) [23], the variant normalized-cut algorithm by Kanna, Vemplala, and Vetta (KVV) [14], the spectral clustering algorithm of Ng, Jordan, and Weiss (NJW) [19], and a multi-cut algorithm by Meilă (Multicut) [17]. We also employed a different distance function for each spectral clustering algorithm. Notice that while *SV* requires manually labeling the entire dataset, *spectral* constructs $\mathcal{S}$ and $\mathcal{D}$ without any human effort.

We tested both contextual information selection schemes using our distance-learning algorithm on three prior kernels—linear, Gaussian, and Laplacian. For each scheme, we selected 10% contextual information. (The 10% sampling rate was determined based on our prior work on UCI datasets [29].) For *SV* and *spectral*, we selected the most uncertain instance pairs. Table 3 presents the clustering errors of k-means using three different schemes. Among all situations, *SV* and *spectral* outperform the *random* strategy, since both attempt to select the most useful information to learn a distance function. Moreover, since *spectral* involves no labeling effort and can perform slightly better than *SV* (which requires labeling the dataset), we are very encouraged by our innovative usage of the unsupervised approach for generating contextual information.

Remark: We did not use active learning with unsupervised learning to label data instances that cannot be distributed into $\mathcal{S}$ or $\mathcal{D}$. Nevertheless, the performance of *spectral* is already very promising.

5.2 Evaluation on Efficiency

We used all datasets to evaluate the efficiency of DA*lign* with the decomposition-based QP solution. We used Gaussian kernel to conduct the experiments. Table 4 reports the CPU time in seconds on running DA*lign* with and without decomposition on solving the QP problem. In our experiment, we only reported the results of the SMO version, where the working set for $\mathcal{S}$ has two pairs and the work set for $\mathcal{D}$ has one pair. We can see that with the help of SMO, the computational cost of DA*lign* can be dramatically reduced. Moreover, from the bottom of the table, we can see that the effectiveness of DA*lign* was not compromised.

5.3 Evaluation on Effectiveness

Amount		Without-SMO	With-SMO
CPU time (seconds)	*mixed*	2,780	221
	color	10,314	606
	image	12,153	782
Clustering error (percents)	*mixed*	42.43	42.43
	color	5.26	5.30
	image	11.10	11.10

Table 4: Clustering performance of DA*lign* with different strategy on selecting contextual information using Gaussian kernel.

Dataset	Kernel	Euclidean	Learned		
			DA*lign*	Kwok	Xing
mixed	Linear	49.35	**47.02**	47.48	47.53
	Gaussian	49.35	**42.43**	46.21	-
	Laplacian	47.67	**32.57**	38.02	-
color	Linear	8.15	**7.00**	7.49	8.00
	Gaussian	7.79	**5.26**	6.34	-
	Laplacian	7.89	**5.50**	6.58	-
image	Linear	14.30	**13.00**	**13.00**	13.27
	Gaussian	13.04	**10.10**	12.15	-
	Laplacian	11.98	**9.38**	10.65	-

Table 5: Clustering Error Rates. No results reported for Xing on Gaussian and Laplacian kernels since this algorithm can only work in input space $\mathcal{I}$.

We used the k-means algorithm to examine the effectiveness of our distance-learning algorithm on clustering problems. We used all three datasets to conduct the clustering experiments. The size of the contextual information was chosen as roughly 10% of the total number of all possible pairs. DA*lign* uses the contextual information to modify three prior distance functions: linear, Gaussian, and Laplacain. The value of k for k-means was set to be the number of classes for each dataset. To measure the quality of clustering, we used the clustering error rate defined in [31] as follows:

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

where $\{c_i\}_{i=1}^n$ denotes the true cluster to which $\mathbf{x}_i$ belongs, $\{\hat{c}_i\}_{i=1}^n$ denotes the cluster predicted by a clustering algorithm, n is the number of instances in the dataset, and $1\{\cdot\}$ the indicator function ($1\{\text{true}\} = 1$, and $1\{\text{false}\} = 0$).

Table 5 reports the k-means clustering results for the three datasets. From Table 5, we can see that DA*lign* achieves the best clustering results in almost all testing scenarios, tying with Kwok on *image* only while linear kernel was employed. DA*lign* performs better than Xing in all cases where the linear kernel is used.

6. CONCLUSION AND FUTURE WORK

In this paper, we have reported our study of an important issue—formulating a context-based distance function for measuring similarity. Our proposed DA*lign* method learns a distance function by considering the nonlinear relationships among the contextual information, without incurring high computational cost. Theoretically, we substantiated that our method achieves optimal alignment toward the ideal kernel. Empirically, we demonstrated that DA*lign*

improves similarity measures and hence leads to improved performance in clustering and classification applications.

Considering the contextual information in information retrieval has been identified as an important research area. Our work provides a mechanism that is more effective than the traditional ones because it achieves both model flexibility and computational efficiency. One of our future research tasks is to analyze theoretically the generalization ability of DA_{lign}. Another future project is to further investigate how we might best model context (for an application or for a user), and how we might effectively formulate and gather contextual information to take advantage of our proposed unsupervised (*spectral*) mechanism with active learning and reinforcement, under the unified learning paradigm (ULP), which we proposed in [8].

7. REFERENCES

[1] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional spaces. *In Proceedings of International Conference on Database Theory*, pages 420–434, 2001.

[2] C. C. Aggarwal. Towards systematic design of distance functions for data mining applications. *The Ninth ACM SIGKDD International Conference on Knowledge Discovery in Data and Data Mining*, 2003.

[3] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] E. Amaldi and V. Kann. On the approximability of minimizing non-zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[5] 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*, August 2003.

[6] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, March 2002.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1999.

[8] E. Chang, C.-H. Hoi, X. Wang, W.-Y. Ma, and M. Lyu. A unified machine learning paradigm for large-scale personalized information management. 5^{th} *IEEE Conference on Emerging Information Technology (invited paper)*, August 2005.

[9] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel target alignment. *Journal Machine Learning Research*, 1, 2002.

[10] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. *In Proceedings of ACM SIGMOD Conference on Management of Data*, pages 301–312, June 2003.

[11] M. Fazel. Matrix rank minimization with applications. *Ph.D. Thesis, Electrical Engineering Dept, Stanford University*, March 2002.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. *In Proceedings of the* $25th$ *VLDB Conference*, pages 518–529, 1999.

[13] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. *Advances in Neural Information Processing Systems 15*, 2003.

[14] R. Kanna, S. Vempala, and A. Vetta. On clusterings - good, bad and spectral. *In FOGS*, pages 367–377, 2000.

[15] H. W. Kuhn and A. W. Tucker. Nonlinear programming. *In Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probabilistics*, pages 481–492, 1951.

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

[17] M. Meila and J. Shi. Learning segmentation by random walks. *In Advances in Neural Information Processing Systems*, pages 873–879, 2000.

[18] M. Nadler and E. P. Smith. *Pattern Recognition Engineering*. New York: Wiley, 1993.

[19] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *In Advances in Neural Information Processing Systems*, pages 849–856, 2002.

[20] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[21] Y. Rui and T. Huang. Optimizing learning in image retrieval. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 236–245, June 2000.

[22] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[23] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[24] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *In Proceedings of ACM International Conference on Multimedia*, pages 107–118, 2001.

[25] V. Vapnik. *Statistical Learning Theroy*. John Wiley and Sons, 1998.

[26] T. Wang, Y. Rui, S. min Hu, and J. guang Sun. Adaptive tree similarity learning for image retrieval, 2003.

[27] A. R. Webb. Multidimensional scaling by iterative majorization using radial basis functions. *Pattern Recognition*, 28(5):753–759, 1995.

[28] D. Wettschereck, D. 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.

[29] G. Wu, N. Panda, and E. Chang. Formulating distance functions via the kernel trick. *ACM International Conference on Knowledge Discovery and Data Mining (poster)*, August 2005.

[30] G. Wu, A. Rahimi, K. Goh, C. Tsai, , Y. Wu, E. Chang, and Y.-F. Wang. Identifying color in motion in video sensors. *UCSB Technical Report*, 2005.

[31] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 15*, 2003.

[32] Z. Zhang. Learning metrics via discriminant kernels and multidimensional scaling : Towards expected euclidean representation. *In Proceedings of the Twentieth International Conference on Machine Learning*, August 2003.