Formulating Distance Functions via the Kernel Trick

Gang Wu
Electrical Engineering
University of California
Santa Barbara, CA
gwu@ece.ucsb.edu

Edward Y. Chang
Electrical Engineering
University of California
Santa Barbara, CA
echang@ece.ucsb.edu

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

ABSTRACT

Tasks of data mining and 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 propose to learn 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. Theoretically, we substantiate that our method is both sound and optimal. Empirically, using several datasets and applications, we demonstrate that our method is effective and useful.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Distance function, kernel trick

1. INTRODUCTION

At the heart of data-mining and information-retrieval tasks 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, 5, 9, 11].

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, 9, 18]. For example, for answering an *ocean* 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) [16]. In this paper we propose performing a *nonlinear* transformation on the feature space to gain greater flexibility for mapping features to semantics.

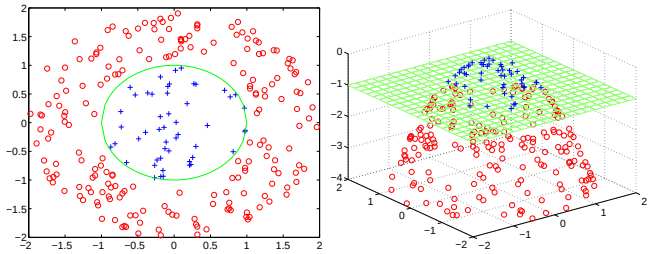
We name our method *distance-function alignment* (DA_{lign} for short). The inputs to DA_{lign} are a prior distance function, and *contextual information*. Contextual information can be conveyed in the form of *training data* (discussed in detail in Section 2). For instance, in the information-retrieval domain, Web users can convey information via relevance feedback showing which documents 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* [3]. 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 $\mathcal{P}$. The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in terms of the kernel is defined as

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

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

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.



(a) Input Space (b) Projected Space

Figure 1: Clustering via the Kernel Trick.

Since a kernel function can be either linear or nonlinear, the traditional feature-weighting approach (e.g., [2, 18]) is just a special case of DA$_{lign}$.

We use an example in Figure 1 to illustrate the effectiveness of DA$_{lign}$. The figure shows how DA$_{lign}$ learns a nonlinear distance function using projection. Figure 1(a) shows two clusters of data, one in circles and the other in crosses. These two clusters of data obviously are not linearly separable in the two-dimensional input space $\mathcal{I}$. After we have used the kernel trick to implicitly project the data onto a three-dimensional projected space $\mathcal{P}$ (shown in Figure 1(b)), the two clusters can be separated by a linear hyperplane in the projected space. What DA$_{lign}$ accomplishes is to learn the distance function in the projected space based on contextual information. From the perspective of the input space, $\mathcal{I}$, the learned distance function captures the nonlinear relationships among the training data.

In summary, we address in this paper a core problem of data mining and information retrieval: formulating a context-based distance function to improve the accuracy of similarity measurement. In Section 2, we propose DA$_{lign}$, an efficient method for adapting a similarity measure to contextual information, and also provide the proof of optimality of the DA$_{lign}$ algorithm. We empirically demonstrate the effectiveness of DA$_{lign}$ on clustering and classification in Section 3. Finally, we offer our concluding remarks and suggestions for future work in Section 4.

2. DA$_{lign}$ ALGORITHM

Given the prior kernel function K and contextual information, DA$_{lign}$ transforms K. Kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ 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

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

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 2.1). Next, we discuss methods to generalize the model to compute the distance between unseen instances (Section 2.2).

2.1 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

$$\tilde{k}_{ij} = \begin{cases} \beta_1 k_{ij}, & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \\ \beta_2 k_{ij} + (1 - \beta_2), & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \end{cases} \quad (3)$$

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. 3 ensures a valid kernel. Theorem 2 mathematically demonstrates that under the constraints from Theorem 1, the transformed $\tilde{\mathbf{K}}$ in Eqn. 3 guarantees a better alignment to the ideal $\mathbf{K}^*$ in [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:

$$\tilde{K} = \beta_1 K + (\beta_2 - \beta_1) K \odot K^* + (1 - \beta_2) K^*, \quad (4)$$

which corresponds to the kernel matrix $\tilde{\mathbf{K}}$, associated with the training set $\mathcal{X}$, in Eqn. 3. If the prior K is positive (semi-) definite, using the fact that the ideal kernel K^* is positive (semi-) definite [7], we can derive that $\tilde{K}$ in Eqn. 4 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 [15]. ■

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

$$\overline{K} = K + \gamma K^*, \quad \gamma > 0, \quad (5)$$

where we use $\overline{K}$ to distinguish with our $\tilde{K}$ defined in Eqn. 3. According to the definition of kernel target alignment [7], we have

$$\left[\frac{< \mathbf{K}, \mathbf{K}^* >}{\sqrt{< \mathbf{K}, \mathbf{K} >}}\right]^2 < \left[\frac{< \mathbf{K} + \gamma \mathbf{K}^*, \mathbf{K}^* >}{\sqrt{< \mathbf{K} + \gamma \mathbf{K}^*, \mathbf{K} + \gamma \mathbf{K}^* >}}\right]^2, \quad (6)$$

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. [7] 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

$$K^*(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if } y(\mathbf{x}_i) = y(\mathbf{x}_j), \\ 0, & \text{if } y(\mathbf{x}_i) \neq y(\mathbf{x}_j), \end{cases} \quad (7)$$

which is the target kernel that a given kernel is supposed to align with. Employing Eqn. 5 and Eqn. 7, we expand (6) as follows

$$\frac{\left(\sum_{\mathcal{S}} k_{ij}\right)^2}{\sum_{\mathcal{S}} k_{ij}^2 + \sum_{\mathcal{D}} k_{ij}^2} < \frac{\left[\sum_{\mathcal{S}}(k_{ij}+\gamma)\right]^2}{\sum_{\mathcal{S}}(k_{ij}+\gamma)^2 + \sum_{\mathcal{D}} k_{ij}^2}. \tag{8}$$

Defining $\gamma = \frac{1-\beta_2}{\beta_2} > 0$, where β_2 is the parameter in Eqn. 3, we then rewrite the right side in (8) as follows

$$\begin{aligned}
& \frac{\left[\sum_{\mathcal{S}}(k_{ij} + \frac{1-\beta_2}{\beta_2})\right]^2}{\sum_{\mathcal{S}}(k_{ij} + \frac{1-\beta_2}{\beta_2})^2 + \sum_{\mathcal{D}} k_{ij}^2} \\
= & \frac{\beta_2^2\left[\sum_{\mathcal{S}}(k_{ij} + \frac{1-\beta_2}{\beta_2})\right]^2}{\beta_2^2\sum_{\mathcal{S}}(k_{ij} + \frac{1-\beta_2}{\beta_2})^2 + \beta_2^2\sum_{\mathcal{D}} k_{ij}^2} \\
= & \frac{\left[\sum_{\mathcal{S}}\left(\beta_2 k_{ij} + (1-\beta_2)\right)\right]^2}{\sum_{\mathcal{S}}\left(\beta_2 k_{ij} + (1-\beta_2)\right)^2 + \beta_2^2\sum_{\mathcal{D}} k_{ij}^2} \qquad (9) \\
\leq & \frac{\left[\sum_{\mathcal{S}}\left(\beta_2 k_{ij} + (1-\beta_2)\right)\right]^2}{\sum_{\mathcal{S}}\left(\beta_2 k_{ij} + (1-\beta_2)\right)^2 + \beta_1^2\sum_{\mathcal{D}} k_{ij}^2} \qquad (10) \\
= & \left[\frac{<\tilde{\mathbf{K}}, \mathbf{K}^*>}{\sqrt{<\tilde{\mathbf{K}}, \tilde{\mathbf{K}}>}}\right]^2, \qquad (11)
\end{aligned}$$

where we apply the assumption of $\beta_1 \leq \beta_2$ from (9) to (10), and employ Eqn. 3 in the last step (11). Combining (6) and (11), 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. 3 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. 5 will have a higher alignment score [14]. Recall that $\beta_2 = \frac{1}{1+\gamma}$. Hence, a smaller β_2 will have a higher alignment. On the other hand, from (10) and (11), 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. 3, 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

$$\tilde{d}_{ij}^2 = \begin{cases} \beta_2 d_{ij}^2, \text{ if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \\ \beta_1 d_{ij}^2 + 2(1-\beta_2) + (\beta_2 - \beta_1)(k_{ii} + k_{jj}), \text{ otherwise.} \end{cases} \tag{12}$$

Since K has been normalized as in Eqn. 2, we have the distance $d_{ij}^2 = 2 - 2k_{ij}$. Eqn. 12 can thus be rewritten as

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

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. 13 decreases the intra-class pairwise distance in $\mathcal{P}$, and increases the inter-class pairwise distance in $\mathcal{P}$. The developed distance metric (Eqn. 13) is a valid distance metric (non-negativity, symmetry, and triangle inequality) since the transformed kernel in Eqn. 3 is a valid kernel, according to Theorem 1.

2.2 Distance Metric Learning

In this subsection, we show how to generalize the model in Eqn. 13 to unseen instances without overfitting the contextual information. We use the feature-weighting method [12] 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 [12], 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 $\tilde{d}_{ij}^2 = (\phi_i - \phi_j)^T\mathbf{A}\mathbf{A}^T(\phi_i - \phi_j)$.

To solve for $\mathbf{A}\mathbf{A}^T$ so that the above distance metric equals the distance in Eqn. 13, 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 [10], 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

$$\min_{\mathbf{A}\mathbf{A}^T, \beta_1, \beta_2} \quad \frac{1}{2}\|\mathbf{A}\mathbf{A}^T\|_{\mathrm{F}}^2 + C_{\mathcal{D}}\beta_1 + C_{\mathcal{S}}\beta_2, \tag{14}$$

$$\begin{aligned}
\text{s.t.} \quad \beta_2 d_{ij}^2 &= \tilde{d}_{ij}^2, && (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\
\tilde{d}_{ij}^2 &= \beta_1 d_{ij}^2 + 2(1-\beta_1), && (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\
\beta_2 &\geq \beta_1, \qquad (15) \\
\beta_1 &\geq 0, \\
1 - \beta_2 &\geq 0,
\end{aligned}$$

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 [17], 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

$$\begin{aligned}
& \mathcal{L}(\mathbf{A}\mathbf{A}^{\mathbf{T}}, \beta_1, \beta_2, \boldsymbol{\alpha}, \mu, \gamma, \pi) \qquad (16) \\
& = \frac{1}{2}\|\mathbf{A}\mathbf{A}^{\mathbf{T}}\|_F^2 + C_{\mathcal{D}}\beta_1 + C_{\mathcal{S}}\beta_2 \\
& - \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} \alpha_{ij}\left[(\phi_i - \phi_j)^T(\beta_2\mathbf{I} - \mathbf{A}\mathbf{A}^T)(\phi_i - \phi_j)\right] \\
& - \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \alpha_{ij}\left[(\phi_i - \phi_j)^T(\mathbf{A}\mathbf{A}^T - \beta_1\mathbf{I})(\phi_i - \phi_j) - 2(1-\beta_1)\right] \\
& - \mu\beta_1 - \gamma(\beta_2 - \beta_1) - \pi(1-\beta_2),
\end{aligned}$$

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:

$$\mathbf{AA^T} = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(\phi_i-\phi_j)(\phi_i-\phi_j)^T - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\phi_i-\phi_j)(\phi_i-\phi_j)^T, \quad (17)$$

$$C_{\mathcal{S}} + \pi - \gamma = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\phi_i-\phi_j)^T(\phi_i-\phi_j), \quad (18)$$

$$C_{\mathcal{D}} + \gamma - \mu = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}\left[2-(\phi_i-\phi_j)^T(\phi_i-\phi_j)\right], \quad (19)$$

Substituting the conditions of (18) and (19) into (16), we obtain the following dual formulation

$$\begin{aligned}\mathcal{W}(\boldsymbol{\alpha},\pi) =& \frac{1}{2}\|\mathbf{AA^T}\|_F^2 + 2\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij} \\ &- \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(\phi_i-\phi_j)^T\mathbf{AA}^T(\phi_i-\phi_j) \\ &+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\phi_i-\phi_j)^T\mathbf{AA}^T(\phi_i-\phi_j) \\ &- \pi, \end{aligned} \quad (20)$$

which has to be maximized w.r.t. α_{ij}'s and $\pi \geq 0$. Actually, π can be removed from the dual function (20), 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 (20) 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 [13], we have the following conditions

$$\alpha_{ij}\left[\beta_2 d_{ij}^2 - \tilde{d}_{ij}^2\right] = 0, \ \ (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S} \quad (21)$$

$$\alpha_{ij}\left[\tilde{d}_{ij}^2 - \beta_1 d_{ij}^2 - 2(1-\beta_1)\right] = 0, \ \ (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D} \quad (22)$$

$$\gamma(\beta_2-\beta_1) = 0, \quad (23)$$

$$\mu\beta_1 = 0, \quad (24)$$

$$\pi(1-\beta_2) = 0. \quad (25)$$

The constraint (15) 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, and we do not get any improvement. To avoid these cases, we then change (15) to be a strict inequality constraint of $\beta_2 > \beta_1$. Therefore, we have $\gamma = 0$ from (23). Using the properties of $\gamma = 0$ and $\pi = 0$, we can then change the dual formulation (20) and its constraints of (18) and (19) by substituting the expansion form of $\mathbf{AA^T}$ in (17) as follows:

$$\begin{aligned}&\max_{\boldsymbol{\alpha}} \ \ \mathcal{W}(\boldsymbol{\alpha}) \quad (26)\\ &= 2\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij} \\ &- \frac{1}{2}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}\sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{S}} \alpha_{ij}\alpha_{kl}\left((\phi_i-\phi_j)^T(\phi_k-\phi_l)\right)^2 \\ &- \frac{1}{2}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}\sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{D}} \alpha_{ij}\alpha_{kl}\left((\phi_i-\phi_j)^T(\phi_k-\phi_l)\right)^2 \\ &+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}\sum_{(\mathbf{x}_k,\mathbf{x}_l)\in\mathcal{D}} \alpha_{ij}\alpha_{kl}\left((\phi_i-\phi_j)^T(\phi_k-\phi_l)\right)^2, \end{aligned}$$

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

where $\phi_i^T\phi_j = K(\mathbf{x}_i,\mathbf{x}_j)$ from the kernel trick. The dual formulation (26) is very similar to that of C-SVMs [17]. 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 (26), we can then generalize our distance-metric model defined in Eqn. 17 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 (17) into the above equation, we obtain

$$\begin{aligned}&\tilde{d}^2(\mathbf{x},\mathbf{x}') \quad (27)\\ &= \left(\phi(\mathrm{x})-\phi(\mathrm{x}')\right)^T\left(\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(\phi_i-\phi_j)(\phi_i-\phi_j)^T \right.\\ &\quad \left. - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(\phi_i-\phi_j)(\phi_i-\phi_j)^T\right)\left(\phi(\mathrm{x})-\phi(\mathrm{x}')\right) \\ &= \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \alpha_{ij}(K_{\mathrm{xx}_i} - K_{\mathrm{xx}_j} - K_{\mathrm{x}_i\mathrm{x}'} + K_{\mathrm{x}_j\mathrm{x}'})^2 \\ &\quad - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \alpha_{ij}(K_{\mathrm{xx}_i} - K_{\mathrm{xx}_j} - K_{\mathrm{x}_i\mathrm{x}'} + K_{\mathrm{x}_j\mathrm{x}'})^2. \end{aligned}$$

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_{xx_i} and K_{xx_j} in Eqn. 27 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. 27. 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 $\mathcal{I}$. ■

3. EXPERIMENTAL EVALUATION

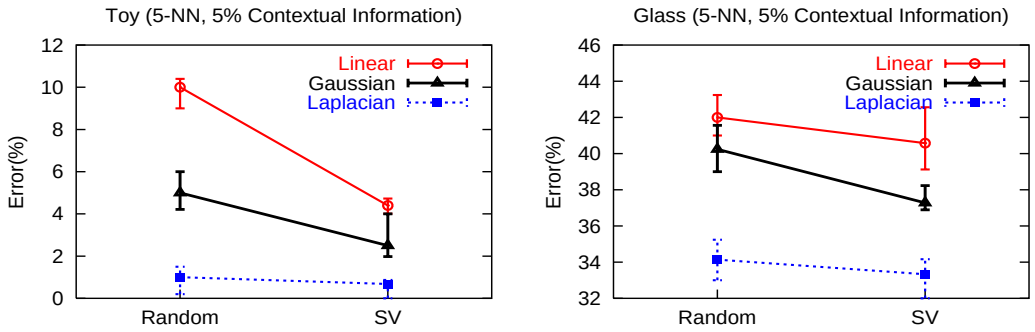
We conducted an extensive empirical study to examine the effectiveness of our context-based distance-function learning algorithm in two aspects.

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

2. *Learning effectiveness*. We compared our DA$_{lign}$ algorithm with the regular Euclidean metric, Kwok et al. [14], and Xing et al.'s [20] on classification and clustering applications.

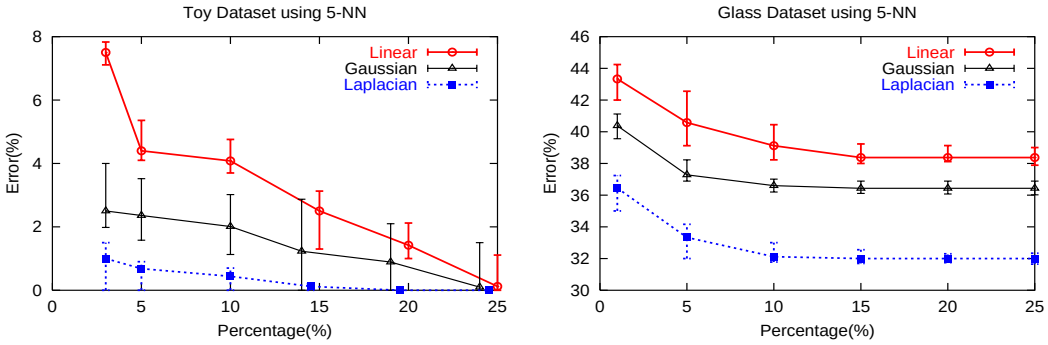
To conduct our experiments, we used five datasets: one toy dataset, and four UCI benchmark datasets.

One toy dataset The *toy* dataset was first introduced in [14]. We used it to compare the effectiveness of our method to other methods. The toy dataset has two classes and eleven features. The



(a) Toy (b) Glass

Figure 2: Quality of Contextual Information vs. Classification Error



(a) Toy (b) Glass

Figure 3: Percentage of Contextual Information vs. Classification Error

Dataset	Kernel	Euclidean	Learned		
			DA$_{lign}$	Kwok	Xing
toy	Linear	50.5	**48.2**	48.5	48.9
	Gaussian	50.5	**43.2**	47.1	-
	Laplacian	48.5	**33.5**	38.9	-
soybean	Linear	33.2	**23.0**	24.2	25.8
	Gaussian	32.6	27.0	**22.4**	-
	Laplacian	32.1	**22.8**	34.8	-
wine	Linear	37.1	36.0	**35.8**	36.3
	Gaussian	36.3	**35.7**	35.9	-
	Laplacian	36.3	**26.8**	32.0	-
glass	Linear	31.5	**31.3**	37.0	35.5
	Gaussian	**31.0**	33.6	40.8	-
	Laplacian	31.5	**30.3**	33.3	-
seg	Linear	43.3	**24.2**	25.3	33.0
	Gaussian	40.5	**19.6**	34.4	-
	Laplacian	36.2	20.3	**14.1**	-

Table 2: Clustering Error Rates on Five Datasets. No results reported for Xing on Gaussian and Laplacian kernels since this algorithm can only work in input space $\mathcal{I}$.

Dataset	Kernel	Euclidean	Learned		
			DA$_{lign}$	Kwok	Xing
toy	Linear	10.00	**4.40**	4.80	5.32
	Gaussian	10.00	**2.22**	2.61	-
	Laplacian	10.00	**0.68**	1.10	-
soybean	Linear	2.50	**0.00**	0.12	0.39
	Gaussian	2.50	**1.00**	1.34	-
	Laplacian	2.50	**0.30**	1.19	-
wine	Linear	5.00	**4.86**	4.94	4.98
	Gaussian	5.00	**3.30**	3.50	-
	Laplacian	6.67	**1.65**	1.68	-
glass	Linear	40.58	**39.67**	42.03	40.34
	Gaussian	40.58	**37.28**	47.83	-
	Laplacian	37.68	**33.10**	33.33	-
seg	Linear	2.94	**1.47**	1.51	1.86
	Gaussian	2.94	**0.10**	0.31	-
	Laplacian	**0.00**	**0.00**	2.94	-

Table 3: Classification Error Rates on Five Datasets. No results reported for Xing on Gaussian and Laplacian kernels since this algorithm can only work in input space $\mathcal{I}$.

3.2.2 Classification Results

For classification experiment, we used the toy dataset and the four UCI datasets. The size of the contextual information chosen was again about 5% of the total number of all possible pairs.

When performing classification, we employed different distance functions: linear, Gaussian, and Laplacian. We compared the performance of using these distance functions before and after applying DA$_{lign}$, and competing methods. For k-NN, we randomly extracted 80% of the dataset as training data, and used the remaining data 20% as testing data. (Notice that the 80% training data here is for training k-NN, not for modifying distance function.) Such a training/testing ratio was empirically chosen via cross-validation so that the classifier using the regular Euclidean metric performed best for a fair comparison. We set k in the k-NN algorithm to be 5. Our