Distance Metric Learning: A Comprehensive Survey

Liu Yang
Advisor : Rong Jin
Department of Computer Science and Engineering
Michigan State University

May 19, 2006

Contents

Abstract

Many machine learning algorithms, such as K Nearest Neighbor (KNN), heavily rely on the distance metric for the input data patterns. Distance Metric learning is to learn a distance metric for the input space of data from a given collection of pair of similar/dissimilar points that preserves the distance relation among the training data. In recent years, many studies have demonstrated, both empirically and theoretically, that a learned metric can significantly improve the performance in classification, clustering and retrieval tasks. This paper surveys the field of distance metric learning from a principle perspective, and includes a broad selection of recent work. In particular, distance metric learning is reviewed under different learning conditions: supervised learning versus unsupervised learning, learning in a global sense versus in a local sense; and the distance matrix based on linear kernel versus nonlinear kernel. In addition, this paper discusses a number of techniques that is central to distance metric learning, including convex programming, positive semi-definite programming, kernel learning, dimension reduction, K Nearest Neighbor, large margin classification, and graph-based approaches.

1 Introduction

Learning a good distance metric in feature space is crucial in real-world application. Good distance metrics are important to many computer vision tasks, such as image classification and content-based image retrieval. For example, the retrieval quality of content-based image retrieval (CBIR) systems is known to be highly dependant on the criterion used to define similarity between images and has motivated significant research in learning good distance metrics from training data. Distance metrics are also critical in image classification applications. For instance, in the K-nearest-neighbor (KNN) classifier, the key is to identify the set of labeled images that are closest to a given test image in the space of visual features — again involving the estimation of a distance metric. Previous work [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] has shown that appropriately-designed distance metrics can significantly benefit KNN classification accuracy compared to the standard Euclidean distance.

There has been considerable research on distance metric learning over the past few years. Depending on the availability of the training examples, algorithms for distance metric learning can be divided into two categories: **supervised distance metric learning** and **unsupervised distance metric learning**. Unlike most supervised learning algorithms where training examples are given class labels, the training examples of supervised distance metric learning is cast into pairwise constraints: the equivalence constraints where pairs of data points that belong to the same classes, and inequivalence constraints where pairs of data points belong to different classes. The supervised distance metric learning can be further divided into two categories: the **global distance metric learning**, and the **local distance metric learning**. The first one learns the distance metric in a global sense, i.e., to satisfy *all* the pairwise constraints simultaneously. The second approach is to learn a distance metric in a local setting, i.e., only to satisfy *local* pairwise constraints. This is particularly useful for information retrieval and the KNN classifiers since both methods are influenced most by the data instances that are close to the test/query examples. Section 2 and Section 3 are devoted to the review of the supervised distance metric learning. The existing work for unsupervised distance metric learning methods is presented in section 4. In section 5, we will discuss the maximum margin based distance metric learning approaches. The kernel methods towards distance metrics is summarized in Section 6.

2 Supervised Global Distance Metric Learning

Approaches in this category attempt to learn metrics that keep *all* the data points within the same classes close, while separating *all* the data points from different classes far apart. The most representative work in this category is [11], which formulates distance metric learning as a constrained convex programming problem. It learns a global distance metric that minimizes the distance between the data pairs in the equivalence constraints subject to the constraint that the data pairs in the inequivalence constraints are well separated. This section is organized as the following. We start with the introduction of pairwise constraints. Then, we will review [11] in a framework of global distance metric learning. Finally, a probabilistic framework for global distance metric learning will be represented by the end of this section.

2.1 Pairwise Constraints

Unlike typical supervised learning, where each training example is annotated with its class label, the label information in distance metric learning is usually specified in the form of pairwise constraints on the data: (1) equivalence constraints, which state that the given pair are semantically-similar and should be close together in the learned metric; and (2) inequivalence constraints, which indicate that the given points are semantically-dissimilar and should not be near in the learned metric. Most learning algorithms try to find a distance metric that keeps all the data pairs in the equivalence constraints close while separating those in the inequivalence constraints. In [7], features weights are adjusted adaptively for each test point to reflect the importance of features in determining the class label of the test point. In [12], the distance function of a information geometry is learned from labeled examples to reflect the geometric relationship among labeled examples. In [11] and [13], the distance metric is explicitly learned to minimize the distance between data points within the equivalence constraints and maximize the distance between data points in the inequivalence constraints.

Let $\mathcal{C} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a collection of data points, where n is the number of samples in the collection. Each $\mathbf{x}_i \in \mathbb{R}^m$ is a data vector where m is the number of features. Let the set of equivalence constraints denoted by

$$\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j |\ \mathbf{x}_i \text{ and to } \mathbf{x}_j \text{ belong to the same class}\}$$

and the set of inequivalence constraints denoted by

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j |\ \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to different classes}\}$$

Let the distance metric denoted by matrix $\mathbf{A} \in \mathbf{R}^{m\times m}$, and the distance between any two data points $\mathbf{x}$ and $\mathbf{y}$ expressed by

$$d^2_{\mathbf{A}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2_{\mathbf{A}} = (\mathbf{x} - \mathbf{y})^T \mathbf{A} (\mathbf{x} - \mathbf{y})$$

2.2 Global Distance Metric Learning by Convex Programming

Given the equivalence constraints in $\mathcal{S}$ and the inequivalence constraints in $\mathcal{D}$, [11] formulated the problem of metric learning into the following convex programming problem [14]:

$$\begin{aligned} \min_{\mathbf{A}\in\mathbf{R}^{m\times m}} \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \|\mathbf{x}_i-\mathbf{x}_j\|_{\mathbf{A}}^2 \\ \text{s.t.} \quad & \mathbf{A}\succeq 0, \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \|\mathbf{x}_i-\mathbf{x}_j\|_{\mathbf{A}}^2 \geq 1 \end{aligned}$$

Note that the positive semi-definitive constraint $\mathbf{A}\succeq 0$ is needed to ensure the negative distance between any two data points and the triangle inequality. Although the problem in (1) falls into the category of convex programming, it may not be solved efficiently because of the following two reasons: First, it does not fall into any special class of convex programming, such as quadratic programming [15] and semi-definite programming [14]. As a result, it can only be solved by the generic approach, which is unable to take advantage of the special structure of the problem. Second, as pointed in [16], the number of parameters in (1) is almost quadratic in the number of features. This property makes (1) difficult to scale to a large number of features. Another disadvantage with (1) is that it is unable to estimate the probability for any data points to share the same class. This algorithm is further extended to the nonlinear case in [17] by the introduction of kernels. The authors also presented the dual formulism to reduce the computation complexity of the original optimization problem in [11]. We will discuss its kernel version in section 6.

2.3 A Probabilistic Approach for Global Distance Metric Learning

Given the computation complexity of the original optimization problem in [11], to simplify the calculation, a probabilistic framework for global distance metric can be set up based on the formulism in (1).

Following the idea of [12], we assume a logistic regression model when estimating the probability for any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ to share the same class, i.e.,

$$\Pr(y_{i,j}|\mathbf{x}_i,\mathbf{x}_j) = \frac{1}{1+\exp\left(-y_{i,j}(\|\mathbf{x}_i-\mathbf{x}_j\|_{\mathbf{A}}^2-\mu)\right)} \tag{1}$$

where

$$y_{i,j} = \begin{cases} 1 & (\mathbf{x}_i,\mathbf{x}_j)\in\mathbf{S} \\ -1 & (\mathbf{x}_i,\mathbf{x}_j)\in\mathbf{D} \end{cases}$$

Parameter μ is the threshold. Two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ will have the same class label only when their distance $\|\mathbf{x}_i-\mathbf{x}_j\|^2$ is less than the threshold μ. Then, the overall log likelihood for both the equivalence constraints $\mathcal{S}$ and the inequivalence constraints $\mathcal{D}$

is written as:

$$\begin{aligned}\mathcal{L}_g(\mathbf{A},\mu) &= \log \Pr(\mathcal{S}) + \log \Pr(\mathcal{D}) \\ = \; &- \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \log\left(1+\exp\left(-\|\mathbf{x}_i-\mathbf{x}_j\|_{\mathbf{A}}^2+\mu\right)\right) \\ &- \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \log\left(1+\exp\left(\|\mathbf{x}_i-\mathbf{x}_j\|_{\mathbf{A}}^2-\mu\right)\right) \qquad (2)\end{aligned}$$

Using the maximum likelihood estimation, we will cast the problem of distance metric learning into the following optimization problem

$$\begin{aligned}\min_{\mathbf{A}\in\mathbf{R}^{m\times m},\mu\in\mathbf{R}} \quad & \mathcal{L}_g(\mathbf{A},\mu) \\ \text{s. t.} \quad & \mathbf{A}\succeq 0,\ \mu \geq 0 \qquad (3)\end{aligned}$$

The difficulty with solving (3) lies in positive semi-definitive constraint $\mathbf{A} \succeq 0$. To simplify our computation, we will model the matrix $\mathbf{A}$ using the eigenspace of training instances $\mathbf{x}$s. Let $\mathcal{T} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ include all the training instances used by the constraints in $\mathcal{S}$ and $\mathcal{D}$. Let $\mathbf{M} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T$ include the pairwise correlation between any two features. Let $\{\mathbf{v}_i\}_{i=1}^K$ are the top K ($K \leq m$) eigenvectors of matrix $\mathbf{M}$. We then assume that $\mathbf{A}$ is a linear combination of the top K eigenvectors

$$\mathbf{A} = \sum_{i=1}^K \gamma_i \mathbf{v}_i\mathbf{v}_i^T, \gamma_i \geq 0,\ i = 1,\ldots,K \qquad (4)$$

where $(\gamma_i, \ldots, \gamma_K)$ are the non-negative weights for linear combination.

Using the parametric form in (4), we have (1) written as

$$\Pr(y_{i,j}|\mathbf{x}_i,\mathbf{x}_j) = \frac{1}{1+\exp\left(-y_{i,j}(\sum_{k=1}^K \gamma_k w_{i,j}^k - \mu)\right)} \qquad (5)$$

where

$$w_{i,j}^k = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{A}(\mathbf{x}_i - \mathbf{x}_j)$$

Then, the log-likelihood function $\mathcal{L}(\mathbf{A},\mu)$ in (2) becomes

$$\begin{aligned}&\mathcal{L}_g^e(\{\gamma_i\}_{i=1}^K,\mu) = \\ &- \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \log\left(1+\exp\left(-\sum_{k=1}^K \gamma_k w_{i,j}^k + \mu\right)\right) \\ &- \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \log\left(1+\exp\left(\sum_{k=1}^K \gamma_k w_{i,j}^k - \mu\right)\right) \qquad (6)\end{aligned}$$

Finally, the optimization problem in (3) is simplified into the following form:

$$\begin{aligned}\min_{\{\gamma_i\in\mathbf{R}\}_{i=1}^K,\mu\in\mathbf{R}} \quad & \mathcal{L}_g^e(\{\gamma_i\}_{i=1}^K,\mu) \\ \text{s. t.} \quad & \mu\geq 0,\ \gamma_i \geq 0, i = 1,\ldots,K \qquad (7)\end{aligned}$$

Notice that the above optimization problem is a convex programming problem, and can be solved directly using the Newton's method. Furthermore, the above framework allows for the incorporation of unlabeled data. This is because matrix $\mathbf{M}$ can be constructed using both the labeled data and the unlabeled data.

3 Supervised Local Distance Metric Learning

3.1 Local Adaptive Distance Metric Learning

In addition to general purpose algorithms for distance metric learning, several papers [7, 18, 16, 19, 6, 8] presented approaches to learn appropriate distance metrics for the KNN classifier. More specifically, these approaches tried to find feature weights that are adapted to individual test examples. We refer to these approaches as "**Local Adaptive Distance Metric Learning**".

3.1.1 Problem Setting

Consider a discrimination problem with J classes and n training data samples. The training dataset is $S = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, with the known class labels denoted as $\{y_1, \cdots, y_n\}$, where $\mathbf{x}_i \in \mathbb{R}^m$, and $y_i \in \{1, \cdots, J\}$. C_j is used to represent the set of training samples in class j. The goal is to predict the class label of a given testing sample with predictor vector $\mathbf{x}_0$.

Assume the data is generated from an unknown distribution $P(\mathbf{x}, y)$. For a given testing sample $\mathbf{x}_0$, the task reduces to estimating the class posterior probabilities$P(j|\mathbf{x}_0)_{j=1}^{J}$. Essentially speaking, the K-nearest neighbor approach assumes that $P(j|(\mathbf{x}_0+\delta\mathbf{x}_0)) \simeq P(j|\mathbf{x}_0)$, when δ is small enough. Then, $P(j|\mathbf{x}_0) \simeq (\sum_{x\in N(\mathbf{x}_0)} P(j|\mathbf{x}))/|N(\mathbf{x}_0)|$, where $N(\mathbf{x}_0)$ is a neighborhood of $\mathbf{x}_0$, and $|N(\mathbf{x}_0)|$ denotes the number of points in $N(\mathbf{x}_0)$. The simplest estimate is to use an indicator function as below:

$$\hat{P}(j|\mathbf{x}_0) = \frac{\sum\limits_{i=1}^{n} \theta(\mathbf{x}_i \in N(\mathbf{x}_0))\theta(y_i = j)}{\sum\limits_{i=1}^{n} \theta(\mathbf{x}_i \in N(\mathbf{x}_0))}$$

where $\theta(x)$ is an indicator function that returns 1 when the input argument is true, and 0 otherwise.

However, the assumption of smoothness within the neighborhood will not hold when the input observations approach class boundaries or when the dimensionality is large. Consequently, a modified local neighborhood where the posterior probabilities are approximately constant, need to be produced by locally adaptive metric techniques in the nearest neighbor classification setting.

3.1.2 Methodology

The k nearest neighbor method classifies $\mathbf{x}_0$ as the most frequent class among its k neighbors in the training set. It is an extremely flexible method, and does not make any assumption about the training data. Furthermore, the k nearest neighbor method is supported by the theoretical argument [20, 21], i.e., the asymptotic error rate of one nearest neighbor is at most twice the Bayes error rate, independent of the distance metric used. We refer the more detailed discussion of KNN to [20, 22, 23]. However, in the case when we have finite samples in a high dimensional space, the curse of dimensionality can hurt the nearest neighbor rule.

According to [6] and [18], a crucial assumption made by the KNN approach is that the class conditional probabilities in the local nearest neighbor is constant. This assumption can be relaxed by assuming that the conditional probabilities in the neighborhood of test examples is smooth, or a slow changing function. However, this is not necessarily true. For instance, for the area close to the decision boundary between two classes, we expect the class labels to change dramatically even within a range of short distance. In order to preserve the smoothness of neighborhood in terms of class conditional probability, we can elongate the distance where the change of label tends to be large such that the data points having inconsistent labeling as the query point are excluded from the neighbor of the query example. In the meantime, we can also squeeze the distance to include more points into the neighborhood of the query point if they share the same class labels as the query point . In other words, the goal of adaptive feature relevance learning is to obtain a neighborhood for a given testing point that have high consistency in assigning class labels.

Below gives two cases that may cause the inconsistency of the class conditional probabilities in the local neighborhood. The first case is the data sparseness caused by the curse of dimension, and the second case is bumpy class distribution in the local neighborhood when the query point is close to the decision boundary. Research has been motivated by the two cases, and significant work has been done on learning local distance metrics adaptive to each query point with modified spacial resolution. For the first case (e.g. [24]), local adaptive distance metric learning essentially determines the feature relevance for each query. The resulting neighborhood is elongated along less relevant feature dimensions and constricted along most influential ones. Consequently, the dimension with low feature relevance value will eventually be eliminated, which is similar to the feature selection adapted to each query point. For the second case (e.g. [25]), given a discriminative function learned by algorithms (e.g. SVM), we learn a distance metric to increase the spatial resolution around the decision surface, and in the meantime decrease the spatial resolution elsewhere. More specifically, along the direction that is perpendicular to the decision boundary, class labels are more likely to change dramatically and thus distance will be elongated to exclude the points that are likely to have inconsistent class labels in the neighborhood of query point; while in the direction along the decision boundary, class labels are less likely to change and therefore distance will be shrunk.

The concept of ”local feature relevance” originated from [18], which learns a flexible metrics to capture local feature relevance. Specifically, it uses a recursive partitioning strategy to adaptively shrink and shape rectangular neighborhood around the test point. Therefore, the learning approach they use combines the strength of the KNN method and the recursive partitioning method. [6] proposes another adaptive nearest-neighbor classification method based on the local Linear Discriminative Analysis (LDA). More specifically, the LDA analysis is applied for each query data point by weighting the training examples based on their distance to the query point. Then, only the discriminative directions identified by the local LDA are used for distance measurement. Interestingly, [6] illustrated the relationship between chi-square statistics and the local LDA, namely the local LDA can be viewed as an approximation of chi-square statistics if we assume a mixture Gaussian distribution for each class. A similar idea is proposed in [26, 27], in which the discriminative direction is computed as the line joining the centriods of the training examples in two different classes that are close to the given test example. [24] developed unified theory for adaptive metric that encompasses the strength of [18] and [6]. Inspired by the similar observation as [24], [8] developed a kernel version of adaptive local distance measurement. [28] presented a distance metric learning method to improve SVM, which increases the spatial resolution around the decision surface based on a Riemannian geometry. A parallel work of [28] is done by [25], which computes a local flexible metric using SVMs.

In the rest of this section, we will first review the concept of "local feature relevance" defined in [18]. We will then reveal the work in [6] in more details. Next, we will introduce the unified framework of Local Adaptive Distance Metric Learning that is presented in [24] and its kernel extension [8]. Finally, we also review algorithm for adaptive local distance metric using SVM in [25].

3.1.3 Local Feature Relevance

The concept of local feature relevance is first introduced by [18]. The motivation of feature relevance estimation comes from the need to exploit the differential relevance of the input measurement variables for class assignment. We briefly review the idea as below.

The least-squares estimate for predicting $f(\mathbf{x})$ is just its expected value over the joint probability density, i.e. $\mathrm{E}f = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}$. Then under the restriction that $x_i = z$, the least-squares prediction for $f(\mathbf{x})$ is

$$\mathrm{E}[f|x_i = z] \quad = \quad \int f(\mathbf{x})p(\mathbf{x}|x_i = z)d\mathbf{x}.$$

Here $p(\mathbf{x}|x_i = z)$ represents the probability density distribution of the input variables other than the ith variable, or

$$p(\mathbf{x}|x_i = z) \quad = \quad \frac{p(\mathbf{x})\delta(x_i - z)}{\int p(\mathbf{x}')\delta(x'_i - z)d\mathbf{x}}$$

where $\delta(x_i - z)$ is the Dirac "delta" function with the property $\delta(x - z) = 0$ if $x \neq z$ and $\int_{-\infty}^{\infty} \delta(x - z)dx = 1$.

Then the improvement in squared prediction error $I_i^2(z)$ associated with knowing the value z of the ith input variable $x_i = z$ is the following:

$$\begin{aligned} I_i^2(z) &= \mathrm{E}[(f(\mathbf{x}) - \mathrm{E}f)^2|x_i = z] - \mathrm{E}[(f(\mathbf{x}) - \mathrm{E}[f(\mathbf{x})|x_i = z])^2|x_i = z] \\ &= (\mathrm{E}f - \mathrm{E}[f|x_i = z])^2 \end{aligned} \tag{8}$$

Clearly, with assumptions $p(\mathbf{x}) = \Pi_{i=1}^p p_i(x_i)$ and $f(\mathbf{x}) = a_0 + \sum_{i=1}^p a_i x_i$, (8) can be easily calculated. In this case, $I_i^2(z) = a_i^2(z - \bar{x}_i)^2$ with $\bar{x}_i = \int x_i p(x_i)dx_i$ being the average value of the ith input variable. To be more general, suppose $f(\mathbf{x}) = \sum_{i=1}^{p} f_i(x_i)$, we get $I_i^2(z) = [f_i(z) - \mathrm{E}f_i]^2$. (8) reflects the influence of the ith input variable on the variation of $f(\mathbf{x})$ at the point $x_i = z$.

Consider an arbitrary point $\mathbf{z} = (z_1, \cdots, z_n)$ in the m-dimensional feature space. A measure of the relevance of the ith input variable x_i to the variation of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{z}$ is,

$$r_i^2(\mathbf{z}) = \frac{I_i^2(z_i)}{\sum_{k=1}^{m} I_k^2(z_k)} \tag{9}$$

$r_i^2(\mathbf{z}) = 0$ when $f(\mathbf{x})$ is independent of x_i at $\mathbf{z}$; and $r_i^2(\mathbf{z}) = 1$ when $f(\mathbf{x})$ only depends on x_i at $\mathbf{z}$.

We can generalized the definition of local feature relevance in (9) by changing the conditioning from a single point $\mathbf{z}$ to a subregion $R(\mathbf{x})$ that includes $\mathbf{z}$. More specifically, we define $r_i^2(R(\mathbf{x}))$ as

$$r_i^2(R(\mathbf{x})) = \int r_i^2(\mathbf{x}) p(x|R(\mathbf{x})) d\mathbf{z}$$

where

$$p(\mathbf{x}|\mathbf{x} \in R(\mathbf{z})) = \frac{p(\mathbf{x}) 1(\mathbf{x} \in R(\mathbf{z}))}{\int p(\mathbf{x}') 1(\mathbf{x}' \in R(\mathbf{z})) d\mathbf{x}'}.$$

Here function $1(\cdot) = 1$ when its input argument is true and 0 otherwise. Thus $r_i^2(R(\mathbf{z}))$ measures the relevance of the ith dimension to the variation of $f(\mathbf{x})$ within the region $R(\mathbf{z})$.

3.1.4 Local Linear Discriminative Analysis

To make the posterior probabilities in the neighborhood be more homogenous, [6] modifies the neighborhood by distance metric estimation, called *Local Linear Discriminative Analysis*. The estimated distance metric shrinks neighborhoods in directions orthogonal to these local decision boundaries, and elongates them parallel to the boundaries. Moreover, this paper discovers the elegant connection between local LDA and the chi-squared distance between the true and estimated posterior, which justify the the proposed metric for computing neighborhood. Details are given below.

Let us first briefly review the standard linear discriminant Analysis(LDA) classification procedure with J classes. As a discriminative feature transform, LDA finds eigenvectors of matrix $\mathbf{T} = \mathbf{S_w}^{-1}\mathbf{S_b}$. Here $\mathbf{S}_b$ denotes the between-class covariance matrix, i.e., the covariance matrix of class means, and $\mathbf{S}_w$ denotes the within-class covariance matrix, i.e. the weighted sum of covariance matrices of each class. $\mathbf{S}_w^{-1}$ captures the compactness of each class, and $\mathbf{S}_b$ represents the separation of the class means. Thus the principle eigenvectors of $\mathbf{T}$ will keep data points from the same classes close and meanwhile separate data points from different classes far apart. We then form a transform matrix $\mathbf{S_T}$ by stacking principle eigenvectors of $\mathbf{T}$ together, and the discriminative features $\mathbf{y}$ is computed as $\mathbf{y} = \mathbf{S_w x}$ where the $\mathbf{x}$ is the original input patterns of the test example.

Based on the standard LDA, [6] proposed to localize both $\mathbf{S}_b$ and $\mathbf{S}_w$ through a iterative procedure: It initializes the distance metric Σ as an identical matrix, i.e., a Euclidean distance metric. At the first step, it calculates $\mathbf{S}_b$ and $\mathbf{S}_w$ using the points that are in the neighborhood of the testing point $\mathbf{x}_0$ measured by the distance metric Σ. At the second step, the estimated $\mathbf{S}_b$ and $\mathbf{S}_w$ are used to update distance metric Σ as follows:

$$\begin{aligned}\Sigma &= \mathbf{S_w}^{-\frac{1}{2}}[\mathbf{S_w}^{-\frac{1}{2}}\mathbf{S_b}\mathbf{S_w}^{-\frac{1}{2}} + \epsilon\mathbf{I}]\mathbf{S_w}^{-\frac{1}{2}} \\ &= \mathbf{S_w}^{-\frac{1}{2}}[\mathbf{S_b}^* + \epsilon\mathbf{I}]\mathbf{S_w}^{-\frac{1}{2}} \qquad (10)\end{aligned}$$

The steps of computing local LDA and updating local distance metric will be iterated alternatively until Σ converges. Note that (10) essentially comes from the following equation

$$\begin{aligned}\Sigma &= \mathbf{S_w}^{-1}\mathbf{S_b}\mathbf{S_w}^{-1} \\ &= \mathbf{S_w}^{-\frac{1}{2}}(\mathbf{S_w}^{-\frac{1}{2}}\mathbf{S_b}\mathbf{S_w}^{-\frac{1}{2}}\mathbf{S_w}^{-\frac{1}{2}}) \\ &= \mathbf{S_w}^{-\frac{1}{2}}\mathbf{S_b}^*\mathbf{S_w}^{-\frac{1}{2}} \qquad (11)\end{aligned}$$

ϵ is introduced to prevent the neighborhood to be infinitely long in the complement of the sphered space. We denote $\mathbf{S}_b^*$ as the projection of $\mathbf{S_b}$ on the sphered space $\mathbf{S}_w$. Essentially, $\mathbf{S_w}$ is used to obtain the projection of the distance upon the sphered space, and $\mathbf{S_b}$ in the local neighborhood discloses the consistency of the class centroids. Consequently, the metric defined in (10), shrinks the neighborhood in directions in which the local centroids differs, and elongates the neighborhood in directions where the class centroids are close to each other. In this sense, the goal of [6] is consistent with the general goal of Local Adaptive Distance Metric Learning.

Furthermore, [6] justifies the proposed metric by showing that the first item in (11) approximates the Chi-squared distance $r(\mathbf{X}, \mathbf{x}_0)$ between the true and estimated posterior at the test point $\mathbf{x}_0$. Let $\mathbf{X}$ represent the neighborhood of a test point $\mathbf{x}_0$. Let $p(j|\mathbf{x})$ be the true probability of class j at point $\mathbf{x}$. Then, the Chi-square distance between the true estimation and approximate estimation of the posterior class probabilities at location $\mathbf{x}_0$, i.e., $r(\mathbf{X}, \mathbf{x}_0)$, is expressed as

$$r(\mathbf{X}, \mathbf{x}_0) = \sum_{j=1}^{J} \frac{[p(j|\mathbf{X}) - p(j|\mathbf{x}_0)]^2}{p(j|\mathbf{x}_0)} \qquad (12)$$

We assume the class conditional density to be Gaussian distribution with mean u_j ($j = 1, \cdots, J$), and a same covariance matrix Σ for each class. A first-order Taylor approximation of $\Pr(j|\mathbf{X})$ at point $\mathbf{x}_0$ is:

$$\Pr(j|\mathbf{X}) \approx \Pr(j|\mathbf{x_0}) - \Pr(j|\mathbf{x_0})(\mu_j - \bar{\mu})^T\Sigma^{-1}(\mathbf{X} - \mathbf{x}_0)$$

where $\bar{\mu} = \sum_j \Pr(j|\mathbf{x}_0)\mu_j$. Using this approximation, we have (12) simplified as

$$\begin{aligned} r(\mathbf{X}, \mathbf{x}_0) &= \sum_{j=1}^{J} \Pr(j|\mathbf{x}_0)[(\mu_j - \bar{\mu})\Sigma^{-1}(\mathbf{X} - \mathbf{x}_0))]^2 \\ &= (\mathbf{X} - \mathbf{x}_0)^T \Sigma^{-1} \sum_j \Pr(j|\mathbf{x}_0)(\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \Sigma^{-1}(\mathbf{X} - \mathbf{x}_0) \end{aligned}$$

Thus the approximated distance metric is $\Sigma^{-1} \sum\limits_j \Pr(j|\mathbf{x}_0)(\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \Sigma^{-1}$. Let $\mathbf{S_w}^{-\frac{1}{2}} = \Sigma^{-1}$ and $\mathbf{S_b}^* = \sum\limits_j \Pr(j|\mathbf{x}_0)(\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T$. This metric can be written into the exact form in (11). By assuming $\mu_j \sim \mathbf{N}(\nu_j, \epsilon\mathbf{I})$ in the sphered space, (10) can be obtained.

3.1.5 Locally Adaptive Feature Relevance Analysis

(12) computes the distance between the true and estimated posteriors. Consider $\mathbf{x}$ as a neighbor of the testing sample $\mathbf{x}_0$. [24] found that the Chi-squared distance can also tell us the extent to which the ith dimension can be relied on for predicting $\Pr(j|\mathbf{x})$. This is achieved by computing the expectation of $\Pr(j|\mathbf{x})$ conditioned at $\mathbf{x}_0$ along the ith dimension, and estimating the relevance of the i_{th} dimension by its ability of predicting the class posterior probabilities locally at $\mathbf{x}_0$. More specifically, [24] defines the measure of feature relevance for the testing point x_0, i.e., $r_i(\mathbf{z})$, as follows

$$r_i(\mathbf{z}) = \sum_{j=1}^{J} \frac{[\Pr(j|\mathbf{z}) - \overline{\Pr}(j|x_i = z_i)]^2}{\overline{\Pr}(j|x_i = z_i)}$$

where $\overline{\Pr}(j|x_i = z_i)$, i.e., the conditional expectation of $\Pr(j|x)$, is calculated as:

$$\overline{\Pr}(j|x_i = z_i) = \mathrm{E}(\Pr(j|\mathbf{x})|x_i = z_i) = \int \Pr(j|\mathbf{x})p(\mathbf{x}|\mathbf{x}_i = z_i)d\mathbf{x}$$

As indicated by the above definition, the feature relevance $r_i(\mathbf{x})$ measures the distance between $\Pr(j|\mathbf{z})$ and the conditional expectation of $\Pr(j|\mathbf{x})$ at location $\mathbf{z}$. The closer $\overline{\Pr}(j|x_i = z_i)$ is to $\Pr(j|\mathbf{z})$, the more information the ith dimension provides for predicting the class posterior probabilities locally at $\mathbf{z}$. Furthermore, [24] defined the expected relevance value for the ith feature by averaging $r_i(\mathbf{z})$ in the vicinity of query point $\mathbf{x}_0$, i.e.,

$$\bar{r}_i(\mathbf{x}_0) = \frac{1}{|N(\mathbf{x}_0)|} \sum_{\mathbf{z} \in N(\mathbf{x}_0)} r_i(\mathbf{z}),$$

where $N(\mathbf{x}_0)$ denotes the neighborhood of $\mathbf{x}_0$ according to a given metric. Similarly, a small $\bar{r}_i(\mathbf{x}_0)$ implies that the class posterior of the test example can be well approximated along the ith dimension in the vicinity of x_0.

Based on the above definition of $\bar{r}_i(\mathbf{x}_0)$, the relative relevance of the ith feature can be defined by

$$w_i(\mathbf{x}_0) = \frac{R_i(\mathbf{x}_0)^t}{\sum_{l=1}^{q} R_l(\mathbf{x}_0)^t}$$

where $R_i(\mathbf{x}_0) = (max_{j=1}^{q}\bar{r}_j(\mathbf{x}_0)) - \bar{r}_i(\mathbf{x}_0)$. Parameter t can be set to be either 1 or 2, which corresponds to linear and quadratic weighting. We also can define the relative relevance using the exponential weighting, i.e.,

$$w_i(\mathbf{x}_0) = \frac{\exp(cR_i(\mathbf{x}_0))}{\sum_{l=1}^{q} \exp(cR_l(\mathbf{x}_0))} \tag{13}$$

where c is chosen to adjust the influence of $\bar{r}_i$ on the relative relevance w_i. Compared to the polynomial weighting, the exponential weighting is more sensitive to the changes in the local feature relevance. Finally, the distance is computed by $D(x, y) = \sqrt{\sum_{i=1}^{q} w_i(x_i - y_i)^2}$, where weights w_i enable the neighborhood to elongate less important feature dimensions and, to constrict the most influential ones.

3.1.6 Adaptive Kernel Metric Nearest Neighbor Classification

As an extension of [24], [8] proposed a new kernel (*Quasiconformal kernel*) for nearest neighbor classification. In particular, the proposed Quasiconformal kernel adjusts the RBF kernel by introducing weights based on the both local consistency of class labels and labeling uncertainty.

First, the quasiconformal mapping is defined as:

$$\hat{\mathbf{K}}(\mathbf{x}, \mathbf{x_0}) = c(\mathbf{x})c(\mathbf{x_0})\mathbf{K}(\mathbf{x}, \mathbf{x_0}) \tag{14}$$

where $\mathbf{x}$ is a sample and $\mathbf{x_0}$ is a testing sample, and $c(\mathbf{x})$ is a positive function of $\mathbf{x}$. Assuming $\mathbf{K}(\mathbf{x}, \mathbf{x}) = 1$ for any $\mathbf{x}$, the corresponding quaciconformal kernel distance is,

$$\begin{aligned}\mathbf{D}(\mathbf{x}, \mathbf{x_0}) &= c(\mathbf{x})^2 - 2c(\mathbf{x})c(\mathbf{x_0})\mathbf{K}(\mathbf{x}, \mathbf{x_0}) + c(\mathbf{x_0})^2 \\ &= (c(\mathbf{x}) - c(\mathbf{x}_0))^2 + 2c(\mathbf{x})c(\mathbf{x}_0)(1 - \mathbf{K}(\mathbf{x}, \mathbf{x}_0))\end{aligned} \tag{15}$$

In [8], the weight function $c(\mathbf{x})$ is defined as

$$c(\mathbf{x}) = \frac{\Pr(j_{\bar{m}}|\mathbf{x})}{\Pr(j_m|\mathbf{x_0})} \tag{16}$$

where $j_m = \arg\max_j \Pr(j|\mathbf{x_0})$ and $j_{\bar{m}} = 1 - j_m$.

Using the above distance of $c(\mathbf{x})$, (15) can be written as follows

$$\mathbf{D}(\mathbf{x}, \mathbf{x_0}) = \left(\frac{\Pr(j_m|\mathbf{x_0}) - \Pr(j_m|\mathbf{x})}{\Pr(j_m|\mathbf{x_0})}\right)^2 + 2c(\mathbf{x})c(\mathbf{x_0})(1 - \mathbf{K}(\mathbf{x}, \mathbf{x_0})) \tag{17}$$

For computational efficiency, the RBF kernel in (17) is replaced by its Taylor expansion at the second order.

$$\mathbf{K}(\mathbf{x},\mathbf{x_0}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x_0})^T\Sigma^{-1}(\mathbf{x}-\mathbf{x_0})\right) \approx 1-\frac{1}{2}(\mathbf{x}-\mathbf{x_0})^T\Sigma^{-1}(\mathbf{x}-\mathbf{x_0})$$

This simplification leads to a weighted combination of Chi-squared distance and Mahalanobis distance.

$$\mathbf{D}(\mathbf{x},\mathbf{x_0}) \approx \left(\frac{\Pr(j_m|\mathbf{x_0})-\Pr(j_m|\mathbf{x})}{\Pr(j_m|\mathbf{x_0})}\right)^2 + c(\mathbf{x})c(\mathbf{x_0})(\mathbf{x}-\mathbf{x_0})^T\Sigma^{-1}(\mathbf{x}-\mathbf{x_0}) \quad (18)$$

$c(\mathbf{x}) \approx c(\mathbf{x_0})$ makes the first term in above expression to be zero, which reduces the distance $\mathbf{D}(\mathbf{x},\mathbf{x_0})$ to the Mahalanobis distance that is weighted by $c(\mathbf{x})c(\mathbf{x_0})$. It is interesting to note the different roles that weighting functions $c(\mathbf{x})$ and $c(\mathbf{x_0})$ play on the distance measure. In particular, $c(\mathbf{x})$ measures the consistency of class labels in the vicinity of $\mathbf{x}_0$; the smaller the $c(\mathbf{x})$ is, the more consistency the class labels are in the neighborhood of $\mathbf{x}_0$. $c(\mathbf{x}_0)$ measures the labeling uncertainty; the smaller the $c(\mathbf{x}_0)$ is, the less uncertain the class label is for $\mathbf{x}_0$. Thus, the product of $c(\mathbf{x})$ and $c(\mathbf{x}_0)$ measures both the labeling uncertainty and the label consistency in the neighborhood of $\mathbf{x}_0$.

Essentially, the sample risk $r(\mathbf{x_0},\mathbf{x})$ defined by [24] and $c(\mathbf{x})$ defined by [8] both represent class-consistency. $r^*(\mathbf{x_0})$ defined by [24] and $c(\mathbf{x_0})$ defined by [8] both measure the degree of uncertainty in labeling $\mathbf{x_0}$ with its maximum likelihood class. However, the shortage of the Chi-squared distance adopted by [24] is that it ignores the direction of variation of $\Pr(j|\mathbf{x})$ from $\Pr(j|\mathbf{x_0})$. As disclosed by (19), this information is captured by $c(\mathbf{x_0})$ and $c(\mathbf{x})$ in [8]. (19) shows that the dilation or contraction of the Mahalnobis distance due to the variation in $c(\mathbf{x})$, is proportional to the square root of the Chi-squared distance. The direction of variation of $\Pr(j|\mathbf{x})$ from $\Pr(j|\mathbf{x_0})$ is further used to decide to which degree the Mahalanobis distance should be modified to drive the neighborhood in the modified distance measure closer to homogeneous class posterior probabilities, or say to determine the dilation or contraction.

$$c(\mathbf{x}) = c(\mathbf{x_0}) \pm \sqrt{\left(\frac{\Pr(j_m|\mathbf{x_0})-\Pr(j_m|\mathbf{x})}{\Pr(j_m|\mathbf{x})}\right)^2} \quad (19)$$

Here $\pm$ represents the sign of $(\Pr(j_m|\mathbf{x_0})-\Pr(j_m|\mathbf{x}))$.

3.1.7 Local Adaptive Distance Metric Learning using SVM

[25] proposed a technique that computes a locally flexible distance metric using SVMs. Similar to the work [24, 7], [25] weights the features based on their relevance to the class conditional probabilities for the query example. However, unlike [24] where the feature relevance is computed based on the Chi-square distance, [25] measures the feature relevance based on the discriminative directions that are identified by SVMs. Specifically, the decision function constructed by SVMs is used to determine the most

discriminative direction in the neighborhood of the query example. The identified direction is then used to provide a local feature weighting scheme. Furthermore, the learned distance metric can be fed back to SVMs to further increase the performance of maximum margin classifers. Empirically, by combining the strength of [24] in local adaptive metric learning and the local discriminative direction obtained by SVMs, [25] improves the classification performance of SVM. By elongating the distance along the direction that is parallel to the decision boundary, and constricting the distance along the direction perpendicular to the decision boundary, the class conditional probabilities in the resulting neighbor tend to be constant.

Recall that SVMs classify patterns based on the classification function

$$f(\mathbf{x}) \quad = \quad \sum_{i=1}^{n_s} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - b$$

where n_s is the number of support vectors. The kernel function $K(\mathbf{x}, \mathbf{y})$ is defined as $K(\mathbf{x}, \mathbf{y}) = \phi^T(\mathbf{x}) \cdot \phi(\mathbf{y})$, where $\phi : R^m \to R^M$ $(m \leq M)$ is a mapping of the input space into a higher dimensional feature space. The class label of a test example $\mathbf{x}$ is determined by the sign of the classification function $f(\mathbf{x})$, and the decision boundary is determined by the equation $f(\mathbf{x}) = 0$. The gradient vector $\mathbf{n_d} = \bigtriangledown_{\mathbf{d}} f$ computed at any point $\mathbf{d}$ of the level curve $f(\mathbf{x}) = 0$, gives the perpendicular direction to the decision boundary in input space at $\mathbf{d}$. The vector $\mathbf{n_d}$ identifies the orientation in the input space on which the projected training data are well separated, locally over $\mathbf{d}$'s neighborhood. Hence, this information can be used to define a local measure of feature relevance.

We now review the definition of feature relevance measurement based on local discriminative direction given by [25]. Consider $\mathbf{x}_0$, a query point that is close to the decision boundary. Let $\mathbf{d}$ be the closest point to $\mathbf{x}_0$ on the boundary $f(\mathbf{x}) = 0$. Given that the class conditional probabilities are likely to change significantly along the direction $\mathbf{n}_d$, we need to increase the distance measurement for any direction that is close to $\mathbf{n}_d$ in order to exclude points that are likely to have different class labels as that of the query example $\mathbf{x}_0$. In the meantime, distance should be reduced where the direction is away from $\mathbf{n}_d$ because the class labels along those direction tends to be uniform. To this end, we follow the measure of feature relevance defined in (13), and define

$$R_i(\mathbf{x}_0) \quad = \quad |\mathbf{u}_i^T \cdot \mathbf{n_d}| = |n_{\mathbf{d},i}|$$

where $\mathbf{n_d} = (n_{\mathbf{d},1}, \cdots, n_{\mathbf{d},n})^T$, and $\mathbf{u}_i$ is the all-zero vector with the ith element being 1. Similar to the previous discussion, weights $R_i(\mathbf{x}_0)$ elongate the dimensions that are not informative, and constrict the dimensions that are critical to the prediction of class labels.

Below listed the detailed steps for the Local Flexible Metric Classification based on SVMs:

- Computer the approximated closest point $\mathbf{d}$ to $\mathbf{x}_0$ on the boundary
- Computer the gradient vector $\mathbf{n}_d = \bigtriangledown_{\mathbf{d}} f$

- Set feature relevance values $R_i(\mathbf{x}_0) = |n_{\mathbf{d},i}|$ for $i = 1, \dots, m$
- Estimate the distance of $\mathbf{x}_0$ from the boundary as follows:

$$B(\mathbf{x}_0) = \min_{\mathbf{S_i}} \|\mathbf{x}_0 - \mathbf{S}_i\|$$

- Compute the feature relevance as:

$$w_i(\mathbf{x}_0) = \frac{\exp(AR_i(\mathbf{x}_0))}{\sum_{i=1}^{m} \exp(AR_i(\mathbf{x}_0))},\ j = 1, \dots, m$$

  where $\mathbf{S}_i$s are nonzero support vectors, i.e., $0 < \alpha_i < C$. Variable A is defined as $A = D - B(\mathbf{x}_0)$, where D is the averaged minimum distance to the nonzero support vectors across the entire training dataset that is computed as follows:

$$D = \frac{1}{n_s} \sum_{\mathbf{x}_k} \min_{\mathbf{S}_i} \|\mathbf{x}_k - \mathbf{S}_i\|$$

  Note that the introduction of factor A will determine if the query example $\mathbf{x}_0$ is close to the decision boundary. More specifically, the larger the A, the closer the query example $\mathbf{x}_0$ is to the decision boundary.

- Use the resulting $\mathbf{w}(\mathbf{x}_0) = (w_1(\mathbf{x}_0), w_2(\mathbf{x}_0), \dots, w_m(\mathbf{x}_0))$ for the KNN classification at the query point $\mathbf{x}_0$.

3.1.8 Discussions

Below provide some discussions about the local adaptive distance metric learning approaches we have reviewed above.

- Although both [18] and [24] define relevance measure for features, they are driven by different motivation. [18] identify the dimensions along which the expected variation of class posterior probability is maximized, while [24] aims to find a dimension to minimize the difference between the true class probability distribution for a given query example and its estimation by the training examples in the neighborhood of the query example.
- The major difference between [6] and [24] is the following. The metric computed by [6] approximates the weighted Chi-squared distance by a Tylor series expansion, given the assumption that the class densities are Gaussian and have the same covariance matrix (which are not practical in real world application). Whereas [24] does not make such assumptions, but approximates the weighted Chi-squared distance using the Taylor expansion.

3.2 Neighborhood Components Analysis

Neighborhood Component Analysis (NCA) algorithm proposed in [9] learns a Mahalanobis distance metric for the KNN classifier by maximizing the leave-one-out cross validation. Below we briefly review the central idea of NCA.

Let the labeled data set denoted by $L = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$. To ensure the learned distance matrix to be symmetric positive semi-definite, [9] assumes the distance metric $\mathbf{Q}$ in the form $\mathbf{Q} = \mathbf{A}^T\mathbf{A}$ where $\mathbf{A}$ can be any matrix. This parametric form will guarantees that distance between any two data points $\mathbf{x}$ and $\mathbf{y}$ to be positive, given the fact that $d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T\mathbf{Q}(\mathbf{x} - \mathbf{y}) = (\mathbf{Ax} - \mathbf{Ay})^T(\mathbf{Ax} - \mathbf{Ay})$.

Given a point $\mathbf{x}_i$, a "soft" neighbor of $\mathbf{x}_i$ is defined by p_{ij}, which is the probability for $\mathbf{x}_j$ to be selected as the neighbor of $\mathbf{x}_i$, and shares the same class label with $\mathbf{x}_i$. in [9], the probability $p_{i,j}$ is defined as:

$$p_{ij} = \frac{\exp(-\|\mathbf{A}x_i - \mathbf{A}x_j\|^2)}{\sum\limits_{k\neq i} \exp(-\|\mathbf{A}x_i - \mathbf{A}x_k\|^2)}$$

Let the set of points that share the same class with $\mathbf{x}_i$ denoted by $C_i = \{j|c_i = c_j\}$. Then, the probability of classifying $\mathbf{x}_i$ correctly is expressed $p_i = \sum_{j\in C_i} p_{ij}$, and the expected number of correctly classified points is $f(\mathbf{A}) = \sum_{i=1}^{n} p_i$. Taking the first-order derivative of $f(\mathbf{A})$ with respect to $\mathbf{A}$, we have

$$\frac{\partial f}{\partial \mathbf{A}} = 2\mathbf{A}\sum_{i=1}^{n}\left(p_i \sum_{k\neq i} p_{i,k}(\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T - \sum_{j\in C_j} p_{i,j}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\right)$$

Instead of using the average classification accuracy, [9] suggests using the leave-one-out cross validation to be the objective function $f(\mathbf{A})$, i.e.,

$$f(\mathbf{A}) = \sum_{i=1}^{n} log(\sum_{j\in C_i} p_{i,j}))$$

NCA has the following three drawbacks:

- NCA suffers from the scalability problem because its objective function is differentiated w.r.t. the distance matrix and the number of parameters in $\mathbf{A}$ is quadratic in the number of features. Therefore, updating the distance matrix will become intractable with large dimensionality.
- The gradient ascent algorithm proposed by NCA does not guarantee to converge the local maxima.
- NCA tends to overfit the training data if the number of training examples is insufficient for learning a distance metric. This happens often when data points are represented in the high dimensional space.

3.3 Relevant Component Analysis

RCA is developed for unsupervised learning using equivalence relations. Founded on an information theoretic basis, and using only closed form expressions of the data, RCA algorithm [29] is a simple and efficient algorithm for learning a full ranked Mahalanobis distance metric. It constructs a Mahalanobis distance metric based on a weighted sum of in-class covariance matrices [29]. It is similar to PCA and linear discriminant analysis in that they depend on the second-order statistics. A number of studies have shown the success of learning distance metric learning using RCA.(e.g. [13] [9]). In this section, we will first give a brief review of RCA, followed by its application in distance metric learning. We will then discuss the kernel version of RCA based on the study ([30]).

3.3.1 A Brief Review of RCA

Relevant Component Analysis (RCA) applies a global linear transformation to assign large weights to relevant dimensions and low weights to irrelevant dimensions. These ”relevant dimensions” are estimated using chunklets. In RCA, a chunklet is defined as a subset of points that are known to belong to the same although unknown class. The detailed steps of RCA is the following:

- Substract the chunklet's mean from all of the points in a given chunklet
- Compute the covariance matrix of all the centered data points in the chunklets. If there are p points in k chunklets, and each chunklet j contains points $\{\mathbf{x}_{ji}\}_{i=1}^{n_j}$ with mean as $\hat{\mathbf{m}}_j$, the covariance matrix in RCA is computed:

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

- Apply the whitening transformation $\mathbf{W} = \hat{\mathbf{C}}^{-\frac{1}{2}}$ associated with this covariance matrix, to the original data points, $\mathbf{x}_{\text{new}} = \mathbf{W}\mathbf{x}$. The inverse of $\hat{\mathbf{C}}$ can be used as a Mahalanobis distance.

Information maximization under chunklet constraints [13] formulates the problem of RCA as a constrained optimization problem. Their work follows an information theoretic criterion proposed by [31]. It searches for $\mathbf{y}$, a transformation of input patterns $\mathbf{x}$, that maximizes the mutual information $\mathbf{I}(\mathbf{X}, \mathbf{Y})$ between $\mathbf{X}$ and transformed $\mathbf{Y}$ under suitable constraints. Accordingly, a set $\mathbf{X} = \{\mathbf{x}_l\}_{l=1}^{n}$ of data points in $\mathbf{R}^N$ is transformed into the set of points $\mathbf{Y} = \{f(\mathbf{x}_l)\}_{l=1}^{n}$ in $\mathbf{R}^M$. The goal is to find a function $f(\cdot)$ in the family of $\mathbf{F}$ that maximizes $\mathbf{I}(\mathbf{X}, \mathbf{Y})$. In addition, transformation function $f(\cdot)$ is required to keep all the data points in the chunklets close to each other. Consequently, the overall problem can be cast into the following optimization problem:

$$\begin{aligned} \max_{f\in\mathbf{F}} \quad & \mathbf{I}(\mathbf{X}, \mathbf{Y}) \\ \text{s.t.} \quad & \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}\|\mathbf{y}_{ji} - \mathbf{m}_j^y\|^2 \leq K \end{aligned} \tag{20}$$

where $\mathbf{m}_j^y$ denotes the mean of data points in the jth chunklet after the transformation, p is the total number of points in chunklets, and K is threshold constant. Given transformation function $f(\cdot)$ is deterministic, there will be no uncertainty regarding $\mathbf{Y}$ given $\mathbf{X}$, and therefore maximizing mutual information $\mathbf{I}(\mathbf{X}, \mathbf{Y})$ is equivalent to maximizing the entropy $\mathbf{H}(\mathbf{Y})$. Let $|J(x)|$ be the Jacobian of the transformation, and we have $p_y(y)dy = p_x(x)/|J(x)|dx$. Thus, $\mathbf{H}(\mathbf{Y})$ can be expressed in terms of $\mathbf{H}(\mathbf{X})$ as follows:

$$\begin{aligned}\mathbf{H}(\mathbf{Y}) &= -\int_y p(y)\log(p(y))dy = -\int_x p(x)\log\frac{p(x)}{|J(x)|}dx \\ &= \mathbf{H}(\mathbf{X}) + \langle \log|J(x)|\rangle_x\end{aligned}$$

In $\mathbf{I}(\mathbf{X},\mathbf{Y})$, the only term that depends on the transformation $\mathbf{A}$ is the Jacobian. The Jacobian is $|\mathbf{A}|$ for a linear transformation $\mathbf{Y}=\mathbf{A}\mathbf{X}$. Thus, (20) is written as:

$$\begin{aligned}\max_{\mathbf{A}} \quad & |\mathbf{A}| \\ \text{s.t.} \quad & \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}\|\mathbf{x}_{ji}-\mathbf{m}_j\|^2_{\mathbf{A}^T\mathbf{A}} \leq K\end{aligned} \tag{21}$$

Denote a Mahalanobis distance matrix as $\mathbf{B}=\mathbf{A}^T\mathbf{A}$, where $\mathbf{B}$ is positive definite, and $\log(|\mathbf{A}|)=\frac{1}{2}\log|\mathbf{B}|$. In this way, (21) can be further written into

$$\begin{aligned}\max_{\mathbf{B}} \quad & |\mathbf{B}| \\ \text{s.t.} \quad & \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}\|\mathbf{x}_{ji}-\mathbf{m}_j\|^2_{\mathbf{B}} \leq K, \mathbf{B}\succ 0\end{aligned} \tag{22}$$

Solving for the Lagrangian gives us the solution of RCA: $\mathbf{B}=\frac{K}{N}\hat{\mathbf{C}}^{-1}$, where $\hat{\mathbf{C}}$ is the average chunklets covariance matrix and N is the dimension of the feature space.

Apply RCA to Distance Metric Learning [13] addresses the problem of learning distance metrics with side-information using the RCA algorithm, which is a simple but efficient algorithm for learning a full ranked Mahalanobis metric [29]. This work demonstrates that with the permission of singular Mahalanobis matrix, RCA on the top of Fisher's linear discriminant is the optimal dimensionality reduction algorithm under the same criterion. [9] is another recent work related to RCA, and it will be reviewed in details in the later sections.

3.3.2 Kernel Relevant Component Analysis

Similar to the kernel PCA, [30] shows that RCA can be kernelized by some elegant matrix manipulations. Their experiments shows significant improvements can be achieved over the linear version, especially when nonlinearities are needed. Besides, [30] presents a learning algorithm for the incremental setting.

Assume that C chunklets are given, and each chunklet c_j comprises of n_j data points $\{x_{j,1},\cdots,x_{j,n_j}\}$. Each data point $x_{j,i}\in R^m$. Denote $\mathbf{1}_j$ be the n-dimensional vertor with

$$[\mathbf{1}_j]_i = \begin{cases} 1, & \text{pattern i} \in C_j \\ 0, & \text{otherwise}\end{cases}$$

and $\mathbf{I}_j$ be the $n\times n$ diagnal matrix diag($\mathbf{1_j}$). Also the chunklet patterns are stacked in a $m\times n$ matrix as: $\mathbf{X}=[\mathbf{x}_{1,1},\mathbf{x}_{1,2},\cdots,\mathbf{x}_{1,n1},\cdots,\mathbf{x}_{j,1},\mathbf{x}_{j,2},\mathbf{x}_{j,n_j}]$. Then $\mathbf{C}$ can be written in below matrix form,

$$\begin{aligned}\mathbf{C} &= \frac{1}{n}\sum_{j=1}^{C}\sum_{i=1}^{n_j}(\mathbf{x}_{j,i} - \frac{1}{n_j}(\mathbf{X}\mathbf{1}_j))(\mathbf{x}_{j,i} - \frac{1}{n_j}(\mathbf{X}\mathbf{1}_j))^T \\ &= \frac{1}{n}\sum_{j=1}^{C}\mathbf{X}(\mathbf{I}_j - \frac{2}{n_j}\mathbf{1}_j\mathbf{1}_j^T + \frac{1}{n_j}\mathbf{1}_j\mathbf{1}_j^T)\mathbf{X}^T \\ &= \frac{1}{n}\mathbf{X}\mathbf{H}\mathbf{X}^T\end{aligned}$$

where

$$\mathbf{H} = \sum_{j=1}^{C}(\mathbf{I}_j - \frac{1}{n_j}\mathbf{1}_j\mathbf{1}_j^T) \tag{23}$$

$\mathbf{H} \in R^{n\times n}$ is symmetric, block diagonal, and positive semi-definite. It plays similar role as the centering matrix $(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ in PCA.

Note that $\mathbf{C}$ in the above can be singular when $n \leq m$. To solve this problem, a regularizer $\epsilon\mathbf{I}$ is introduced to $\mathbf{C}$, where ϵ is a small positive constant. This leads to the following expression:

$$\hat{\mathbf{C}} = \epsilon\mathbf{I} + C = \epsilon\mathbf{I} + \frac{1}{n}\mathbf{X}\mathbf{H}\mathbf{X}^T \tag{24}$$

Using the Woodbury formula, i.e.,

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

the inverse of $\mathbf{C}$ can be computed efficiently as follows:

$$\hat{\mathbf{C}}^{-1} = (\epsilon\mathbf{I} + \frac{1}{n}\mathbf{X}\mathbf{H}\mathbf{X}^T)^{-1} = \frac{1}{\epsilon}\mathbf{I} - \frac{1}{n\epsilon^2}\mathbf{X}\mathbf{H}(\mathbf{I} + \frac{1}{n\epsilon}\mathbf{X}^T\mathbf{X}\mathbf{H})^{-1}\mathbf{X}^T$$

To extend the RCA to the kernelized version, we first compute the dot product between two patterns $\mathbf{x}$ and $\mathbf{y}$ as:

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

Given a nonlinear mapping $\varphi(\cdot)$ and the kernel function $k(\mathbf{x},\mathbf{y}) = \varphi(\mathbf{x})\varphi(\mathbf{y})$, the dot-product between $\varphi(\mathbf{x})$ and $\varphi(\mathbf{y})$ is calculated as:

$$k(\mathbf{x},\mathbf{y}) = \frac{1}{\epsilon}k(\mathbf{x},\mathbf{y}) - \mathbf{k}_x^T\left[\frac{1}{n\epsilon^2}\mathbf{H}(\mathbf{I} + \frac{1}{n\epsilon}\mathbf{K}\mathbf{H})^{-1}\right]\mathbf{k}_y \tag{25}$$

where $\mathbf{K} = [k(\mathbf{x}_i,\mathbf{x}_j)]_{ij}$ is the $n \times n$ kernel matrix defined on the chunkelt patterns, $\mathbf{k}_x = [k(\mathbf{x}_{1,1},\mathbf{x}),\cdots,k(\mathbf{x}_{j,n_j},\mathbf{x})]^T$ and $\mathbf{k}_y = [k(\mathbf{y}_{1,1},\mathbf{y}),\cdots,k(\mathbf{y}_{j,n_j},\mathbf{y})]^T$.

3.3.3 Incremental Learning

Equation (25) requires computing $\mathbf{H}(\mathbf{I}+\frac{1}{n\epsilon}\mathbf{KH})^{-1}$ whenever there is a new chuncklet, which is often computationally expensive. [30] presents an efficient learning algorithm for incremental updating without having to recompute the inverse of the large matrix.

By defining the value of ϵ, (24) can be written as $\hat{\mathbf{C}} = \frac{1}{n}(\mathbf{XHX}^T + \epsilon\mathbf{I})$. Using the Woodbury formula, $\hat{\mathbf{C}}^{-1} = \frac{n}{\epsilon}\mathbf{I} - \frac{n}{\epsilon^2}\mathbf{XH}(\mathbf{I} + \frac{1}{\epsilon}\mathbf{KH})^{-1}\mathbf{X}^T$. Below, the set of all processed chunklets are denoted by $\mathbf{A}$ containing $n_\mathbf{A}$ patterns, and the new chunklet as $\mathbf{B}$ containing $n_\mathbf{B}$ patterns. The corresponding $\mathbf{X}$ matrices are $\mathbf{X}_A \in R^{d\times n_A}$ and $\mathbf{X}_B \in R^{d\times n_B}$, respectively. Then $\mathbf{K}$ and $\mathbf{H}$ can be decomposed as:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \mathbf{K}_{AB}^T & \mathbf{K}_{BB} \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_A & 0 \\ 0 & \mathbf{H}_B \end{bmatrix}$$

where

$$\mathbf{K}_{AA} = \mathbf{X}_A^T\mathbf{X}_A$$
$$\mathbf{K}_{AB} = \mathbf{X}_A^T\mathbf{X}_B$$
$$\mathbf{K}_{BB} = \mathbf{X}_B^T\mathbf{X}_B$$

$\mathbf{H}_A$ is $\mathbf{H}$ in (23) that corresponds to all the processed chunklets, and $\mathbf{H}_B = \mathbf{I} - \frac{1}{n_B}\mathbf{1}\mathbf{1}^T$, where $\mathbf{1} \in R^{n_B}$ is a all-one vector.

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

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} = \begin{pmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{I} \end{pmatrix} \mathbf{P}^{-1}[-\mathbf{CA}^{-1} \;\; \mathbf{I}]$$

where $\mathbf{P} = \mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}$, we get,

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

Denote $\mathbf{Y}_A = \mathbf{H}_A\mathbf{Z}_A$, then

$$\begin{aligned}
\mathbf{H}(\mathbf{I} + \frac{1}{n\epsilon}\mathbf{KH})^{-1} &= \begin{pmatrix} \mathbf{Y}_A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \\
&+ \begin{pmatrix} -\frac{1}{\epsilon}\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H}_B \\ \mathbf{H}_B \end{pmatrix} (\mathbf{I} + \frac{1}{\epsilon}\mathbf{K}_{BB}\mathbf{H_B} - \frac{1}{\epsilon^2}\mathbf{K}_{AB}^T\mathbf{Y}_A\mathbf{K}_{AB}\mathbf{H_B})^{-1} \begin{pmatrix} -\frac{1}{\epsilon}\mathbf{K}_{AB}^T\mathbf{Y}_A & \mathbf{I} \end{pmatrix}
\end{aligned}$$

The complexity of this algorithm has been fully discussed in [30]. The total computational complexity is $o(n_A^2 n_B + n_A n_B^2 + n_B^3)$. Given the fact that $n_B \ll n_A$, the complexity is much less than that of the naive approach, which is $o((n_A + n_B)^3)$. Conclusively, the kernelized RCA extends the ability of RCA to produce nonlinear transforms of the input space. Moreover, the incremental update procedure allows the kernel RCA transform to be computed efficiently in an adaptive environment.

4 Unsupervised Distance Metric Learning

For unsupervised distance metric learning or called manifold learning, the main idea is to learn an underlying low-dimensional manifold where geometric relationships (e.g. distance) between most of the observed data are preserved. There is deep connection between unsupervised distance metric learning and dimension reduction. Every dimension reduction approach is essentially to learn a distance metric without label information. For example, as a classical dimension reduction approach, Principle Component Analysis (PCA) can be viewed as a special distance metric. Specifically, using the principle eigenvectors $\mathbf{u}_i$ of the covariance matrix, we can construct a distance metric $\mathbf{A} = \sum_i \mathbf{u}_i \mathbf{u}_i^T$, and the distance between $\mathbf{x}$ and $\mathbf{y}$ is measured by $d = (\mathbf{x} - \mathbf{y})^T \mathbf{A} (\mathbf{x} - \mathbf{y})$. Given this essential connection, our emphasis in this section is the review of dimension reduction.

4.1 Problem Setting

The dimension reduction problem is, given a data set $\mathbf{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \in R^M$, to find a set of points $\{\mathbf{y}_1, \cdots, \mathbf{y}_n\} \in R^n$ ($M \gg m$), such that each $\mathbf{y}_i$ "represents" its counterpart $\mathbf{x}_i$. For the convenience of presentation, we denote the matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ and correspondingly the matrix $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n]$. Here, we only consider the special case that $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathcal{M}$, where $\mathcal{M}$ is a manifold embedded in R_m.

Many dimension reduction algorithms have been proposed. Table 4.1 summarizes dimension reduction algorithms based on whether they are linear or nonlinear, global or local. We start our discussion with the linear dimension reduction approach, including Principle Component Analysis (PCA) and Multiple Dimension Scaling (MDS), followed by the discussion of nonlinear approaches, including ISOMAP [32], Local Linear Embedding (LLE) [33], and the Laplacian Eigenmap [34]. At the end of this section, we will present a unified framework for dimension reduction algorithms.

	Linear	nonlinear
Global	PCA, MDS	ISOMAP
Local	LLP , LLE, Laplacian Eigenmap	

Table 1: Algorithms for dimension reduction

4.2 Linear Methods

The task of dimensionality reduction is to find a small number of features to represent a large number of observed dimensions. The classical linear algorithms includes Principle Component Analysis (PCA) and Multidimensional Scaling (MDS).

Principal Component Analysis (PCA) finds the subspace that best preserves the variance of the data. Formally speaking, to find the orthonormal basis $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_m]$ that maximize the variance of $\mathbf{y}_i = \mathbf{U}^T \mathbf{x}_i$. Here $\mathbf{x}_i$ $(i = 1 \cdots n)$ has been centered by the mean of the input space

$$\begin{aligned}\text{Var}(\mathbf{Y}) &= \text{Var}(\mathbf{U}^T\mathbf{X}) \\ &= \text{E}[(\mathbf{U}^T\mathbf{X})(\mathbf{U}^T\mathbf{X})^T] \\ &= \text{E}[\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U}] \\ &= \mathbf{U}^T\text{E}[\mathbf{X}\mathbf{X}^T]\mathbf{U} \\ &= \mathbf{U}^T\Sigma\mathbf{U}\end{aligned}$$

where $\Sigma = \text{Var}(\mathbf{X})$. Assume $[\mathbf{V}^{\text{pca}}, \Lambda^{\text{pca}}] = \text{eig}(\Sigma)$, where $\mathbf{V}^{\text{pca}}$ is the eigen matrix containing eigenvectors of Σ as columns, and Λ^{pca} is a diagonal matrix whose diagonal elements are the corresponding eigenvalues of Σ. Then the matrix $\mathbf{U}$ that maximizes $\mathbf{U}^T\Sigma\mathbf{U}$ is composed by the first m eigenvectors of Σ, denoted as $\mathbf{V}_m^{\text{pca}}$, and the corresponding eigenvalues are denoted as Λ_m^{pca}. Consequently, the PCA projections of $\mathbf{X}$ with a rank of m is $\mathbf{Y} = \mathbf{V}_m^{pca}\mathbf{X}$.

Multidimensional Scaling (MDS) [35] finds the rank m projection that best preserves the inter-point distance (dissimilarity) given by the pairwise distance matrix $\mathbf{D}$. The detailed steps are the following:

- From $\mathbf{D}$, calculate $\mathbf{B} = \mathbf{X}^T\mathbf{X} = -\frac{1}{2}\mathbf{HDH}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ is the centering matrix, and $\mathbf{1} \in R^m$ is an all-one vector.
- Spectrally decompose $\mathbf{B} = \mathbf{V}^{\text{mds}}\Lambda^{\text{mds}}(\mathbf{V}^{\text{mds}})^T$, where $[\mathbf{V}^{\text{mds}}, \Lambda^{\text{mds}}] = \text{eig}(\mathbf{B})$.
- $\mathbf{X} = \mathbf{V}^{\text{mds}}(\Lambda^{\text{mds}})^{\frac{1}{2}}$.
- Rank m projections $\mathbf{Y}$ closet to $\mathbf{X}$ is $\mathbf{Y} = \mathbf{V}_m(\Lambda_m^{\text{mds}})^{\frac{1}{2}}$. Here $\mathbf{V}_m^{\text{mds}}$ and Λ_m^{mds}) are the top m eigenvectors and eigenvalues, respectively.

Relations between PCA and MDS can be revealed by the following equations:

$$\mathbf{V}^{\text{pca}} = \mathbf{X}\mathbf{V}^{\text{mds}}, \Lambda^{\text{pca}} = \Lambda^{\text{mds}}$$

$$\mathbf{Y}^{\text{pca}} = (\Lambda^{\text{pca}})^{\frac{1}{2}}\mathbf{Y}^{\text{mds}}$$

Conclusively, in the Euclidean case, MDS only differs from PCA by starting with $\mathbf{D}$ and calculating $\mathbf{X}$. Starting with $\mathbf{D}$ is useful when $\mathbf{D}$ is not Euclidean, but a dissimilarity function of the data.

PCA and Euclidean MDS are simple, efficient, and guarantee to optimize their criterions. However, as linear approaches, they cannot find nonlinear structure in the data. This motivates a series of nonlinear dimension reduction methods.

4.3 Nonlinear Methods

Significant research has been done on nonlinear dimension reduction. We follow the categorization scheme in [36] and divide these algorithms into below four categories.

- **Embedding Methods** Embedding algorithms can be divided as global and local. As a global algorithm, ISOMAP [32] assumes that isometric properties should be preserved in both the observation space and the intrinsic embedding space in the affine sense. According to this assumption, ISOMAP finds the subspace that best preserves the geodesic interpoint distances. Extensions of ISOMPA to conformal mappings is also discussed in [37]. Unlike ISOMAP that tries to preserve the geodesic distance for any pair of data points, Locally Linear Embedding (LLE) [38] and Laplacian Eigenamp [34] focus on the preservation of local neighbor structure. As an extension of [34], Locality Preserving Projection (LPP) [39] finds linear projective maps that optimally preserve the neighborhood structure of the data set. It is an optimal linear approximation to the eigen functions of the Laplace-Beltrami Operator on the manifold. Here Laplace-Beltrami Operator is a self-adjoint elliptic differential operator defined as $\Delta = d\delta + \delta d$, where d is the exterior derivative ($df = \sum\limits_i \frac{\partial f}{\partial x_i} dx_i$ written in a coordinate chart) and d and δ are adjoint to each other with respect to the inner product.
- **Mutual information methods** This group of methods assumes that the mutual information is a measurement of the differences of probability distribution between the observed space and the embedded space. Related work includes stochastic nearest neighborhood (SNE) [40] and manifold charting [41].
- **Projection methods** This group of methods is geometrically intuitive. The goal is to find principal surfaces passing through the middle of data, for example, the principal curves [42, 43]. Its difficulty lies in how to generalize the global variable-arc-length parameter into higher dimensional surface.
- **Generative methods** This group of methods adopts generative topology models [44, 45, 46], and hypothesizes that observed data are generated from the evenly spaced low dimensional latent nodes. Then the mapping relationship between the observation space and the latent space can be modeled. However, EM (Expectation-Maximization) algorithms makes the generative models easily fall into local minimum and have slow convergence rates.

In this survey, we will emphasize on nonlinear embedding methods, including the ISOMAP, LLE and Laplacian Eigenamp. We will first review each of these methods, and then discuss their underlying connections.

4.3.1 LLE(Locally Linear Embedding)

The locally linear embedding (LLE) algorithm [38] is a local method to establish the mapping relationship between the observed data and the corresponding low-dimensional data. The principle of the LLE algorithm is to preserve local order relation of data in both the embedding space and the intrinsic space. Each sample in the observation space is a linearly weighted average of its neighbors. The basic LLE algorithm based on local covering numbers can be described as follows:

- Find the n nearest neighbor for each $\mathbf{x}_i$ in the dataset. By assuming that each data point and its neighbors lie on a locally linear patch of the manifold, the local geometry of these patches can be characterized by linear coefficients that reconstruct each data point from its neighbors. Based on this assumption, the reconstruction error can be measured by adding up the squared distance between all the data points and their reconstructions, i.e.,

$$\varphi(\mathbf{W}) = \|\mathbf{x}_i - \sum_{j=1}^{K} \mathbf{W}_{ij} x_{ij}\|^2$$

  where the weight matrix $\mathbf{W} = [\mathbf{W}_{ij}]_{n\times n}$ summarizes the contribution of the jth data point to the ith reconstruction.
  The weighted matrix $\mathbf{W}$ can be obtained by solving this least square problem. There are two constraints regarding $\mathbf{W}$. One is $\sum_{j=1}^{n} \mathbf{W}_{ij} = 1$ for any data point $\mathbf{x}_i$. The other is $\mathbf{W}_{ij} = 0$ if $\mathbf{x}_j$ is not a neighbor of $\mathbf{x}_i$. These two constraints are important because the solution under these constraints characterizes the intrinsic geometric properties of each neighborhood, and is invariant to rotations, rescalings, and translations of a given data point and its neighbors.
- Construct the approximation matrix. Choose $\mathbf{W}_{ij}$ by minimizing

$$\sum_{i=1}^{n} \|\mathbf{x}_i - \sum_{j=1}^{n} \mathbf{W}_{ij}\mathbf{x}_{ij}\|$$

  under the condition that $\sum_j \mathbf{W}_{ij} = 1$ for each $\mathbf{x}_i$.
- Construct a neighbor-preserving mapping. The idea is, the reconstruction weights $\mathbf{W}_{ij}$ reflect intrinsic geometric properties of the data, and are invariant to the linear transformation from a high dimensional (M) coordinates of each neighborhood to global internal coordinates on a low dimensional (m, $M \gg m$) manifold. Therefore, the expectation is that $\mathbf{W}_{ij}$ that reconstruct the i_{th} data point in M dimensions should reconstruct its embedded manifold coordinated in m dimensions. This is equivalent to approximate the nonlinear manifold around point $\mathbf{x}_i$ by the linear hyperplane that passes through its neighbors $x_{i1}, \cdots, x_{ik}$. Therefore, by minimizing the cost function $\Phi(\mathbf{Y}) = \|\mathbf{y}_i - \sum_{j=1}^{K} \mathbf{W}^*_{ij} y_{ij}\|$, where $\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\, \varphi(\mathbf{W})$, we get $\mathbf{Y}^* = \underset{\mathbf{Y}}{\operatorname{argmin}}\, \Phi(\mathbf{Y})$. Here we have two constraints. One is $\sum_i \mathbf{y}_i = 0$ enforcing that the objective $\Phi(\mathbf{Y})$ is invariant to translation in Y. The other is $\sum_i \mathbf{y}_i\mathbf{y}_i^T/r = \mathbf{I}$ to avoid the degenerate solution of $\mathbf{Y} = 0$, where r is the number of local covering set.
  $\mathbf{Y}^* = \underset{\mathbf{Y}}{\operatorname{argmin}}\, \Phi(\mathbf{Y})$ can be further reduced to an eigenvector decomposition problem as $\underset{\mathbf{Y}}{\operatorname{argmin}}\, \|\mathbf{Y}^T(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})\mathbf{Y}\|^2$. Therefore to minimize the cost function, we only need to compute the embedding by eigen analysis of the symmetric positive semdefinite matrix$(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$. Only the $m+1$ lowest eigenvalues will be taken and the smallest eigenvector will be discarded considering the constraint terms. Thus, each high-dimensional $\mathbf{x}_i \in R^M$ is mapped to a

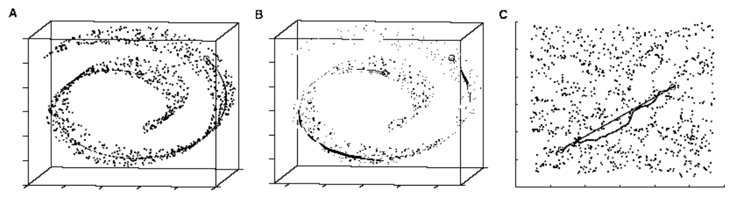

Figure 1: The Swiss roll data set, illustrating how ISOMAP exploits geodesic paths for nonlinear dimensionality reduction.

low-dimensional vector $\mathbf{y}_i \in R^m$. And $\mathbf{y}_i$ represents global internal coordinates on the manifold.

Disadvantage of LLE and Related Improvement LLE algorithm requires recomputing the eigen spectrum of the entire matrix $(I - W)^T(I - W)$ for every test example, which could be computationally expensive. An corresponding improvement has been made by [36]. According to weierstrass approximation theorem, any continuous function on a closed and bounded interval can be uniformly approximated on that interval by polynomials to any degree of accuracy. Based on this theorem, the MLA algorithm, [36] uses the gaussian RBF kernel to approximate the relationship. Given a testing point $\mathbf{x}'$, its internal coordinates on the manifold can be computed by $y' = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}')$ where $K(\mathbf{x}_i; \mathbf{x}') = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}'\|^2}{2\sigma^2})$, and α can be computed by the complete data $(\mathbf{X}; \mathbf{Y})$.

4.3.2 ISOMAP

The challenge of nonlinear dimension reduction is illustrated in [32] by the example with data lying on a two-dimensional Swiss roll. The leftmost picture in figure 1 shows that, for two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). Points far apart on the underlying manifold, as measured by their geodesic distances (i.e., the length of shortest path), may appear deceptively close in the high-dimensional input space, as measured by their straight-line Euclidean distance.

Only the geodesic distance is capable of revealing the true low-dimensional geometry of the manifold. However, both PCA and MDS can only see the Euclidean structure and fail to detect the intrinsic two dimensionality. To address the challenge, [32] propose an approach, named ISOMAP, to exploit the eigen analysis for nonlinear embedding. The detailed steps of the ISOMAP algorithm are given as follows:

- Set up neighbor relations on the manifold $\mathcal{M}$ based on the pairwise Euclidean distance $d_{\mathbf{X}}(i, j)$ in the input space $\mathbf{X}$. This can be realized either by a fixed neighbor size or by a fixed distance range. The identified neighbor relations are further represented by a weighted Graph $\mathcal{G}$ over the data points, with weight $d_{\mathbf{X}}(i, j)$ assigned to the corresponding edges. The picture in the middle of figure

1 shows the neighborhood graph $\mathcal{G}$ constructed in this step allows an approximation (solid segments) to the true geodesic path to be computed efficiently in the next step, as the shortest path in $\mathcal{G}$.

- Estimate the pairwise geodesic distance $d_{\mathcal{M}}(i, j)$ on the manifold $\mathcal{M}$. The geodesic distance is defined as the distance of the shortest path $d_{\mathcal{G}}(i, j)$ in the graph $\mathcal{G}$. For neighboring points, input distance is a good approximation to geodesic distance; for far away points, geodesic distance adding up a sequence of "short hops" between neighboring points which is actually the shortest paths in a graph with edges connecting neighboring data points.
- Apply MDS to the matrix of geodesic distance $\mathbf{D}_{\mathcal{G}}$, where $[\mathbf{D}_{\mathcal{G}}]_{i,j} = d_{\mathcal{G}}(i, j)$, to construct an embedding of the data in a m-dimensional Euclidean space $\mathbf{Y}$ that can best preserve the estimated intrinsic geometry of the manifold. The rightmost picture in figure 1 shows the two-dimensional embedding recovered by ISOMAP in this step, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (curves).

4.3.3 Laplacian Eigenmaps

Laplacian Eigenmaps method [34] is a locality-preserving nonlinear dimension reduction method. The eigenmaps are provided by the eigenvectors of the graph Laplacian, and the eigenfunctions of Laplace Beltrami operator on the manifold. And the map generated is a discrete approximation to a continuous map. Specifically, given a data set $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \in R^M$, we construct a weighted graph with n nodes, one for each point, and a set of edges by connecting neighboring points. Then the eigenvectors of the graph Laplacian is used to construct the embedding map from the input space to the embedded manifold. The detailed steps are listed as below:

- Constructing Adjacency Graph. Two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected if they are close. "Closeness" criterions can be $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 < \epsilon$ or n-nearest-neighbor (Node i and node j are connected by an edge if node i is among n nearest neighbors of node j or node j is among n nearest neighbor of node i)
- Weighting the edges. It could be rigid weighting and soft weighting. For rigid weighting, $\mathbf{W}_{ij} = 1$ if nodes i and j are connected; and $\mathbf{W}_{ij} = 0$, otherwise. For soft weighting, if nodes i and j are connected, $\mathbf{W}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\tau}}$, otherwise $\mathbf{W}_{ij} = 0$. Here τ need to be chosen carefully.
- Calculate Eigenmap. Given the connected graph $\mathbf{G}$ (if the graph $\mathbf{G}$ is not connected, go through each connected components of $\mathbf{G}$), solve a generalized eigen decomposition problem: $\mathcal{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f}$. Here $\mathbf{D}$ is a diagonal weight matrix, the entries of which are column sums of $\mathbf{W}$, i.e. $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ji}$, and $\mathcal{L}$ is the laplacian matrix. Let $0 = \lambda_0 \leq \cdots \leq \lambda_{k-1}$ be the eigenvalues, and $\mathbf{f}_0, \cdots, \mathbf{f}_{k-1}$ be the corresponding eigenvectors. By leaving the zero eigenvalue out, and using the top m eigenvectors, we obtain the following embedded mapping $\mathbf{x}_i \rightarrow (\mathbf{f}_1(i), \cdots, \mathbf{f}_m(i))$

Justification in the Simplest Case To reveals the scheme used by Laplacian Eigenmap to preserve locality in the dimension reduction process, [34] suggests the following simplest case. Given a dataset $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \in R^M$ and the connected weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Imagine mapping the weighted graph $\mathbf{G}$ to a line and get the corresponding map $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$, so that connected points stay as close as possible. In this case, minimizing the objective function

$$\sum_{i,j} (y_i - y_j)^2 \mathbf{W}_{ij}$$

ensures that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close then $\mathbf{y}_i$ and $\mathbf{y}_j$ are close as well. Precisely, if $\mathbf{W}_{ij}$ is large, which means $\mathbf{x}_i$ and $\mathbf{x}_j$ are close, the minimization of the objective function will force $(y_i - y_j)$ to be small, which means y_i and y_j are close as well.

In this simplest case, [34] also shows that, for any $\mathbf{y}$, $\frac{1}{2}\sum\limits_{i,j}(\mathbf{y}_i - y_j)^2 \mathbf{W}_{ij}$ can be further written into $\mathbf{y}^T \mathcal{L} \mathbf{y}$, given $\mathcal{L} = \mathbf{D} - \mathbf{W}$, Therefore, the minimization problem can be reduced to:

$$\begin{aligned} \underset{\mathbf{y}}{\text{argmin}} \quad & \mathbf{y}^T \mathcal{L} \mathbf{y} \\ \text{s.t.} \quad & \mathbf{y}^T \mathbf{D} \mathbf{y} = 1, \\ & \mathbf{y}^T \mathbf{D} \mathbf{1} = 0. \end{aligned}$$

The first constraint $\mathbf{y}^T \mathbf{D} \mathbf{y} = 1$ can prevent the minimization problem go to a trivial solution because of the scaling factors, because $\mathbf{D}_{ii}$ represents the importance of vertex i. For the second constraint,$\mathbf{y}^T \mathbf{D} \mathbf{1} = 0$ can remove a translation invariance of $\mathbf{y}$, to be sure that the solution is given by the eigenvectors with the smallest nonzero eigenvalue.

Generalization The above problem of embedding the graph $\mathbf{G}$ into a line can be further generalized into the problem of embedding the graph $\mathbf{G}$ into m-dimensional Euclidean space. Suppose $\mathbf{Y} = [\mathbf{y}_1\, \mathbf{y}_2, \cdots, \mathbf{y}_m]$, where the ith row $\mathbf{y}^{(i)}$ is m-dimension representation of $\mathbf{x}_i$. Minimizing

$$\sum_{i,j} \|\mathbf{y}^{(i)} - \mathbf{y}^{(j)}\|^2 \mathbf{W}_{ij}$$

can be reduced to find

$$\underset{\mathbf{Y}^T\mathbf{D}\mathbf{Y}=\mathbf{I}}{\text{argmin}} \ \text{trace} \mathbf{Y}^T \mathcal{L} \mathbf{Y})$$

For the m-dimensional case, the constraint $\mathbf{Y}^T \mathbf{D} \mathbf{Y} = \mathbf{I}$ prevents collapse onto a subspace of dimension less than $(m-1)$. The solution of this optimization problem is provided by the matrix of eigenvectors corresponding to the lowest nonzero eigenvalues of the generalized eigenvalue problem $\mathcal{L}\mathbf{y} = \lambda \mathbf{D} \mathbf{y}$.

4.3.4 Connections among ISOMAP, LLE, Laplacian Eigenmaps and Spectral Clustering

- One important observation from [34] is that, the process of dimensionality reduction that preserves locality yields the same solution as clustering. Essentially, local approaches such as LLE and Laplacian Eigenmaps only attempts to preserve neighborhood information, which can be interpreted as a soft clustering of the data. Therefore, local approaches to dimension reduction can also be deemed as a natural clustering of the data. And the solutions of both LLE and Laplacian Eigenmaps are closely tied to spectral clustering. Differently, as a global strategy, ISOMAP [32] attempts to approximate all the geodesic distances on the manifold.

- The strong connection between LLE [38] and Laplacian Eigenmaps [34]. Denote as the semi-definite matrix $\mathbf{E} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$ in LLE. And $\mathbf{E}$ and $\mathcal{L}$ are deemed as operators acting on functions defined on the data points. [34] has proved that, under certain conditions, $\mathbf{E}f = \frac{1}{2}\mathcal{L}^2 f$, where the eigenvectors of $\frac{1}{2}\mathcal{L}^2$ coincide with those of $\mathcal{L}$. Consequently, LLE's attempts to minimize $f^T(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})f$ can be reduced to find the eigenfunctions of $(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$, which is essentially to find the eigenfunctions of the iterated Laplacian $\mathcal{L}$.

4.4 A Unified Framework for Dimension Reduction Algorithms

Several unsupervised learning algorithms use an eigendecomposition for obtaining a lower-dimensional embedding of data lying on a non-linear manifold. Inspired by [47], we unify these algorithms into a common framework, based on the computation of an embedding for the training points obtained from the principal eigenvectors of a symmetric matrix. Given a data set $\mathbf{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and $\mathbf{x}_i \in R^M$, a $n \times n$ affinity matrix is first constructed and denoted as $\mathbf{H}$. Let S_i be the ith row sum of the affinity matrix $\mathbf{H}$:

$$S_i = \sum_j \mathbf{H}_{ij} \tag{26}$$

Then normalizing $\mathbf{H}$ by S_i gives us $\hat{\mathbf{H}}$. We further compute the m largest positive eigenvalues λ_t and eigenvector $\mathbf{v}_t$ of $\hat{\mathbf{H}}$. Then the embedding of $\mathbf{x}_i$ has two alternative solutions:

4.4.1 Solution 1

The embedding of $\mathbf{x}_i$ is $\mathbf{e}_i$ with $e_{it} = \sqrt{\lambda_t} v_{it}$, and $\langle \mathbf{e}_i, \mathbf{e}_j \rangle$ is the best approximation of $\hat{\mathbf{H}}_{ij}$ in the squared error sense. MDS(Multi-Dimensional Scaling) and ISOMAP belong to this category.

Multi-Dimensional Scaling(MDS) In Multi-Dimensional Scaling [35], a distance or affinity matrix is first computed between each pair of training examples. Then we normalize $\mathbf{H}$ to convert distances to equivalent dot-products by the "double-centering"

$$\hat{\mathbf{H}}ij = -\frac{1}{2}(\mathbf{H}_{ij} - \frac{1}{n}S_i - \frac{1}{n}\mathbf{S}_j + \frac{1}{n^2}\sum_k \mathbf{S}_k) \tag{27}$$

The embedding $\mathbf{e}_{it}$ of example $\mathbf{x}_i$ is given by $\sqrt{\lambda_t} v_{ti}$.

ISOMAP [32] simply generalizes MDS to non-linear manifolds. It replaces the Euclidean distance with an approximation of the geodesic distance on the manifold. It computes $\mathbf{M}_{ij}$ as the squared geodesic distance, then applies to this matrix the distance-to-dot-product transformation in (27), as for MDS. Similar with MDS, the embedding is $\sqrt{\lambda_t} v_{ti}$ instead of $y_{it} = v_{ti}$.

4.4.2 Solution 2

The embedding of $\mathbf{x}_i$ is $\mathbf{y}_i$ whose t-th element y_{it} is the i-th element of $\mathbf{v}_t$. Spectral clustering, Laplacian Eigenmap and LLE belong to this category.

Spectral Clustering In spectral clustering [48], normalization is done in the following way:

$$\hat{\mathbf{H}}_{ij} = \frac{\mathbf{H}_{ij}}{\sqrt{S_i \mathbf{S}_j}}$$

The first m principal eigenvectors of $\hat{\mathbf{H}}$ are computed. Then for clustering purpose, K-means is applied on the new unit-norm coordinates, obtained from the embedding $y_{it} = v_{ti}$.

Laplacian Eigenmap In the Laplacian Eigenmap algorithm [34], the Laplacian operator is approximated by heat kernel or simple-minded matrix whose element is one if the two vertices are connected and zero otherwise. Laplacian Eigenmap method solve a generalized eigenproblem:

$$(\mathbf{S} - \mathbf{H})\mathbf{v}_j = \lambda_j \mathbf{S} \mathbf{v}_j$$

with eigenvalues λ_j, eigenvectors $\mathbf{v}_j$ and $\mathbf{S}$ a diagonal matrix whose entries are given by S_i in (26). Those bottom eigenvectors except the smallest one are used for embedding.

Local Linear Embedding LLE [38] seeks an embedding to preserve the local geometry in the neighborhood of each data point. In this method, local predictive matrix $\mathbf{W}$ is first computed, subject to $\sum_j \mathbf{W}_{ij} = 1$, and $\mathbf{W}_{ij} = 0$ if $\mathbf{x}_j$ is not a k-nearest-neighbor of $\mathbf{x}_i$. Then minimizing $\| \sum_j \mathbf{W}_{ij}\mathbf{x}_j - \mathbf{x}_i \|^2$ gives us $\mathbf{H} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$. From the lowest eigenvectors of $\mathbf{H}$ (except for the smallest one with zero eigenvalue) $\mathbf{v}_t, t = 1, \cdots, m$, we get the embedding $y_{it} = v_{ti}$, where v_t.

4.5 Out-of-Sample Extensions

In this section, the reviewed unsupervised learning algorithms based on eigendecomposition, provide an embedding only for given training points, with no direct extension for testing examples short of recomputing eigenvectors. [47] have presented an extension to five unsupervised learning algorithms based on a spectral embedding of the data: MDS, spectral clustering, Laplacian eigenmaps, ISOMAP and LLE. This extension allows us to apply a trained model to testing points without having to recompute eigenvectors.

5 Distance Metric Learning based on SVM

To achieve good generalization in classification setting, a good distance metric should not only achieve high consistency in the neighborhood, but also maintain large margin at the boundaries between different classes. [25] and [10] both incorporate maximization margin into the process of distance metric learning. Moreover, to simplify computation, SVMs can be reformulated into a semidefinite programming problem (SDP). [49] provides discussion for systematically applying SDP methods to solve the kernelized margin maximization problem. In this section, we first give a brief review of SVM. Then we will describe the Large Margin Nearest Neighbor based distance metric learning methods [25, 10]. Finally, the SDP methods provided by [49] to solve the kernel version margin maximization problem will be summarized.

5.1 A Review of SVM

Consider a binary classification problem. We are given a labeled dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where the class label $y_i \in \{-1, +1\}$. The support vector machines [50, 51] require the solution of the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{\|\mathbf{w}\|_2^2}{2} + C\sum_{i=1}^{n} \xi_i, \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{28}$$

The training vectors $\mathbf{x}_i$ are mapped into a higher dimensional space by the function ϕ. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. The dual problem of (28) is:

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)\alpha_i\alpha_j + \sum_{j=1}^{N}\alpha_j \tag{29}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \forall i. \end{aligned}$$

with kernel trick $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Here α_is $(i = 1, \cdots, n)$ defined over the hypercube $[0, C]^n$ are the lagrange coefficients. The goal is to learn a set of parameters α that maximize the objective function in (29). And the obtained classifier is $y(\mathbf{x}) = f(\mathbf{x}, \alpha) = \text{sign}(\sum_i \alpha_i y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + b)$, and b is computed from data.

SVM have many desirable properties, and we address a few of them in the following.

Bounding issue and Maximum Margin Property Different form traditional methods whose aim is minimizing the empirical risk, SVMs aim at minimizing an upper bound of the generalization error. Specifically, SVMs learn the αs in $f(\mathbf{x}, \alpha)$ to make the trained machine to satisfy the maximum margin property, i.e., the decision boundary it represents has the maximum minimum distance from the closest training point. By defining the expected risk (expectation of the test error) for a training machine α as

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x}, \alpha)|dP(\mathbf{x}, y)$$

and the empirical risk(mean error rate measured over the training set) $R_{emp}(\alpha)$ as

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(\mathbf{x}_i, \alpha)|$$

we have the following bounds:

$$R(\alpha) \leq R_{emp}(\alpha) + Conf(h)$$

Here $R(\alpha)$ represents the accuracy attained on a particular training set; and $Conf(h)$ represents the ability of the machine to learn any training set without error. These two items drive the bias and variance of the generalization error, respectively. It is a tradeoff between $R_{emp}(\alpha)$ and $Conf(h)$ to get best generalization error.

The goal of SVM is to minimize an upper bound on the generalization error. For linearly separable case, SVM computes the hyperplane $\mathbf{w} \cdot \mathbf{x} - b = 0$ that has the minimum distance from the closest training point; while for the nonseparable case, SVM looks for the hyperplane that maximizes the margin and minimizes an upper bound of the error simultaneously.

Sparseness Representation SVMs has the sparseness representation of the decision function. Precisely, those training examples that lie far away from the hyperplane receive zero weight and do not participate in its specification and therefore. And only Training examples that lie close to the decision boundary(called support vectors) receive nonzero weights. Since the majority of the training examples will be safely ignored, SVMs can classify new examples efficiently. A small number of support vectors indicates that the two classes can be well separated.

5.2 Large Margin Nearest Neighbor Based Distance Metric Learning

Recent work [10] learns a Mahanalobis distance metric in the kNN classification setting by semidefinite programming. The learned distance metric enforces the k-nearest neighbors to always belong to the same class while examples from different classes are separated by a large margin. This algorithm makes no assumptions about the distribution of the data.

5.2.1 Objective Function

Given a training set of n labeled samples and the corresponding class labels $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$. Let $y_{ij} \in \{0, 1\}$ indicate whether or not the label y_i and y_j match. Beyond the class label information, the Euclidean distance based K-nearest-neighbor approach is used to specify k target neighbors for each input $\mathbf{x}_i$. And $\eta_{ij} \in \{0, 1\}$ is used to indicate whether $\mathbf{x}_j$ is a target neighbor of $\mathbf{x}_i$. Both y_{ij} and η_{ij} are fixed binary matrices during training. The goal is to learn a linear transformation $\mathbf{L} : \mathbf{R}^m \rightarrow \mathbf{R}$ that optimizes k-Nearest-Neighbor classification. This transformation is used to compute squared distance as

$$D(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{L}(\mathbf{x}_i, \mathbf{x}_j)\|^2 \tag{30}$$

On one hand, large distances between each input and its target neighbors should be penalized; on the other, small distances between each input and all other differently labeled inputs should also be penalized. Consequently, the cost function is given by:

$$\varepsilon(\mathbf{L}) = \sum_{ij} \eta_{ij} \|\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 + C \sum_{ijl} \eta_{ij}(1 - y_{il})[1 + \|\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)\|^2 - \|\mathbf{L}(\mathbf{x}_i - \mathbf{x}_l)\|^2]_+ \tag{31}$$

$[z]_+ = max(z, 0)$ denotes the standard hinge loss and the constant $C > 0$. The first term only penalizes large pairwise distances, instead of all similarly labeled examples. The second term in the cost function incorporates the idea of a margin. In particular, for each input $\mathbf{x}_i$, the hinge loss is incurred by differently labeled inputs whose distances do not exceed the distance from input $\mathbf{x}_i$ to any of its target neighbors by one absolute unit of distance (those exceed have no contribution to the cost function). The cost function thereby favors distance metrics in which differently labeled inputs maintain a large margin of distance and do not threaten to invade each other's neighborhoods.

The similarity of the cost function between this method and SVMs is two-fold:

- In both cost functions, one term penalizes the norm of the linear transformation $\mathbf{L}$ in the distance metric, while the other incurs the hinge loss for examples that violate the condition of unit margin.
- In SVMs, the hinge loss is only triggered by examples near the decision boundary, similarly, the hinge loss in (31) is only triggered by differently labeled examples that do not invade each other's neighborhoods.

5.2.2 Reformulation as SDP

To efficiently compute the global minimum of (31), [10] further reformulated it into a SDP problem, First, (30) is rewritten as $\mathbf{D}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i, \mathbf{x}_j)$, where the the Mahalanobis distance metric $\mathbf{M}$ is induced by the linear transformation $\mathbf{L}$ as $\mathbf{M} = \mathbf{L}^T \mathbf{L}$. In this way, the first term is already linear in $\mathbf{M}$. For the second item, a slack variables ξ_{ij} is introduced for all pairs labeled differently. And the resulting SDP is:

$$\begin{aligned}
\min_{\mathbf{M}} \quad & \sum_{ij} \eta_{ij}(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) + C \sum_{ijl} \eta_{ij}(1 - y_{il})\xi_{ijl} \\
\text{s.t.} \quad & (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) - (\mathbf{x}_i - \mathbf{x}_l)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_l) \geq 1 - \xi_{ijl}, \\
& \xi_{ijl} \geq 0, \\
& \mathbf{M} \succeq 0.
\end{aligned}$$

[25] (reviewed in section 3.1) purposes a margin based approach in the context of Local Adaptive Distance Metric Learning, which is comparable to [10]. It suggests

to use the decision boundaries of SVMs to induce a locally adaptive distance metric for kNN classification. Specifically, around each testing point, the decision function is constructed by SVMs to determine the most discriminant direction in its neighborhood. Based on this direction, a local feature weighting scheme is constructed to keep class conditional probabilities in the neighbor to be consistent. Comparing [25] and [10], we may find that, although they both hinge the distance metric learning on large margin, [10] does not involve the training of SVMs; but [25] need to train SVMs at the first place. In this sense [10] is more efficient than [25].

5.3 Cast Kernel Margin Maximization into a SDP Problem

Margin has been considered as a criterion to measure the separation of the labeled data. We first give a brief review of both hard margin and soft margin, then we present the kernelized margin maximization problem. Finally, the SDP methods used to solve the kernelized margin maximization problem, which is proposed by [49], will be reviewed in detail.

5.3.1 Definition of Margin

Hard Margin Given a labeled sample set $S_l = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, the maximal margin classifier with geometric margin $\gamma = \frac{1}{\|\mathbf{w}_*\|_2}$ can be realized by finding the hyperplane $(\mathbf{w}_*, b_*)$ that solves the optimization problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \langle \mathbf{w}, \mathbf{w} \rangle \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b) \geq 1, \ , i = 1, \cdots, n \end{aligned} \tag{32}$$

Here Φ maps $\mathbf{x}_i$ to a high dimensional space. γ corresponds to the distance between the convex hulls, i.e. the smallest convex sets that contain the data in each class. Its Lagrangian dual problem is given by:

$$\begin{aligned} w(\mathbf{K}) &= \frac{1}{\gamma^2} = \langle \mathbf{w}_*, \mathbf{w}_* \rangle \\ &= \max_{\alpha} 2\alpha^T \mathbf{e} - \alpha^T G(\mathbf{K})\alpha : \alpha > 0, \alpha^T \mathbf{y} = 0 \end{aligned}$$

where $\mathbf{e} \in R^n$ is the vector of ones. $\alpha \in R^n$. $G(\mathbf{K})$ is defined by $G_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)y_i y_j$.

Soft Margin The hard margin solution exists only when the labeled sample is linearly separable in feature space. Therefore soft margin are defined for a nonlinearly separable labeled sample set S_l. Here we consider both the 1-norm and the 2-norm soft margins. In this section, we use w_{S1} and w_{Ss} to represent the cost functions for 1-Norm Soft Margin and 2-Norm Soft Margin, respectively.

1-norm Soft Margin Given a labeled sample set $S_l = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, the 1-norm soft margin classifier with geometric margin $\gamma = \frac{1}{\|\mathbf{w}_*\|_2}$ can be realized by solving the optimization problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \langle \mathbf{w}, \mathbf{w} \rangle \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \ i = 1, \cdots, n, \\ & \xi_i \geq 0, \ i = 1, \cdots, n. \end{aligned} \tag{33}$$

Its corresponding Lagrangian dual problem is:

$$\begin{aligned} w_{S_1}(\mathbf{K}) &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle + C \sum_{i=1}^{n} \xi_{i,*} \\ &= \max_{\alpha} 2\alpha^T \mathbf{e} - \alpha^T G(K) \alpha : C \geq \alpha \geq 0, \alpha^T \mathbf{y} = 0 \end{aligned} \tag{34}$$

2-norm Soft Margin Given a labeled sample set $S_l = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, the 1-norm soft margin classifier with geometric margin $\gamma = \frac{1}{\|\mathbf{w}_*\|_2}$ can be realized by solving the optimization problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \langle \mathbf{w}, \mathbf{w} \rangle \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \ , i = 1, \cdots, n \\ & \xi_i \geq 0, \ i = 1, \cdots, n \end{aligned}$$

Its corresponding Lagrangian dual problem is:

$$\begin{aligned} w_{S_2}(\mathbf{K}) &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle + C \sum_{i=1}^{n} \xi_{i,*}^2 \\ &= \max_{\alpha} 2\alpha^T \mathbf{e} - \alpha^T (G(K) + \frac{1}{C} I_n) \alpha : \alpha \geq 0, \alpha^T \mathbf{y} = 0 \end{aligned}$$

The difference between 1-norm Soft Margin and 2-norm Soft Margin is two-fold. First, for the error tolerance item about ξ, they use different norm of ξ: 1-norm and 2-norm. Second, the upper bound of lagrange multiplier α is introduced directly by $C \geq \alpha \geq 0$ in the 1-norm soft margin and indirectly by the item $\frac{1}{C} I_n$ in the 2-norm soft margin.

5.3.2 Cast into SDP problem

First, semidefinite programming can be adopted to minimize the generalized performance measure

$$w_{C,\tau}(K) = \max_{\alpha} 2\alpha^T \mathbf{e} - \alpha^T (G(K) + \tau I) \alpha : C \geq \alpha \geq 0, \alpha^T \mathbf{y} = 0 \tag{35}$$

with $\tau \geq 0$ on the training data w.r.t the kernel matrix $\mathbf{K}$, where $\mathbf{K}$ is in the positive semidefinite cone ($\mathcal{K}$) with constraint trace$(K) = c$, i.e.

$$\begin{aligned} \min_{k \in \mathcal{K}} \quad & w_{C,\tau}(K_{tr}) \\ \text{s.t.} \quad & \text{trace}(K) = c \end{aligned} \tag{36}$$

Below we will show how to realize the optimization problem in (35) in a semidefinite programming framework.

Proposition 1 $w_{C,\tau}(K) = \max_{\alpha} 2\alpha^T\mathbf{e} - \alpha^T(G(K)+\tau I)\alpha : C \geq \alpha \geq 0, \alpha^T\mathbf{y} = 0$ is convex in K, because that $\max_{\alpha} 2\alpha^T\mathbf{e} - \alpha^T(G(K)+\tau I)\alpha$ is a convex objective function. And the constraints $C \geq \alpha \geq 0, \alpha^T\mathbf{y} = 0$ are also convex. Therefore (35) is a convex optimization problem.

Theorem A Given a labeled sample set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted by $\mathbf{y} \in R^{n_{tr}}$, the kernel matrix $\mathbf{K} \in \mathcal{K}$ that optimizes (35), with $\tau \geq 0$, can be found by solving the following convex optimization problem:

$$\begin{aligned} \min_{K,t,\lambda,\nu,\delta} \quad & t \\ \text{s.t.} \quad & \operatorname{trace}(K) = c, \\ & K \in \mathcal{K}, \ \nu \geq 0, \ \delta \geq 0, \\ & \begin{pmatrix} G(K_{tr} + \tau I_{n_{tr}}) & \mathbf{e} + \nu - \delta + \lambda\mathbf{y} \\ (\mathbf{e} + \nu - \delta + \lambda\mathbf{y})^T & t - 2C\delta^T\mathbf{e} \end{pmatrix} \succeq 0. \end{aligned} \tag{37}$$

Theorem B Given a labeled sample set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted by $\mathbf{y} \in R^{n_{tr}}$, the kernel matrix $\mathbf{K} = \sum_{i=1}^{g} \mu_i K_i$ ($\mathbf{K}_i$ is some initial guess of $\mathbf{K}$) that optimizes (35), with $\tau \geq 0$, under the additional constraint $\mu \geq 0$ can be found by solving the following convex optimization problem, and considering its dual solution

$$\begin{aligned} \max_{\alpha,t} \quad & 2\alpha^T\mathbf{e} - \tau\alpha^T\alpha - ct \\ \text{s.t.} \quad & t \geq \frac{1}{r_i}\alpha^T G(K_{i,tr})\alpha, i = 1, \cdots, m \\ & \alpha^T\mathbf{y} = 0, \ C \geq \alpha \geq 0 \end{aligned}$$

where $r \in R^m$ with $\operatorname{trace}(K_i) = r_i$.

5.3.3 Apply to Hard Margin

Maximizing the margin of a hard margin SVMs with respect to the kernel matrix can be realized as a special case of the semidefinite programming framework derived in Theorem A.

$$\min_{K \in \mathcal{K}} w(K_{tr}) \ \text{ s.t. } \ \operatorname{trace}(K) = c \tag{38}$$

Here $w(K_{tr}) = w_{\infty,0}(K_{tr})$.

Theorem C Given a linearly separable labeled sample set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted by $y \in R^{n_{tr}}$, the kernel matrix $K \in \mathcal{K}$ that optimizes (38) can be found by solving the following problem:

$$\begin{aligned}\min_{K,t,\lambda,\mu} \quad & t \\ \text{s.t.} \quad & \text{trace}(K) = c, \\ & K \in \mathcal{K}, \;\; \mu \geq 0, \\ & \begin{pmatrix} G(K_{tr}) & \mathbf{e}+\mu+\lambda\mathbf{y} \\ (\mathbf{e}+\mu+\lambda\mathbf{y})^T & t \end{pmatrix} \succeq 0. \end{aligned}$$

Cast into QCQP For the specific case in which the $\mathbf{K}_i$ are rank-one matrices $\mathbf{K}_i = \mathbf{v}_i\mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal (e.g., the normalized eigenvectors of an initial kernel matrix K_0), the semidefinite program reduces to a QCQP:

$$\begin{aligned}\max_{\alpha,t} \quad & 2\alpha^T\mathbf{e} - ct \\ \text{s.t.} \quad & t \geq (\hat{\mathbf{v}}_i^T\alpha)^2, i = 1, \cdots, m, \\ & \alpha^T\mathbf{y} = 0, \\ & \alpha \geq 0. \end{aligned}$$

with $\hat{\mathbf{v}}_i = diag(\mathbf{y}\mathbf{v}_i(1 : n_{tr}))$, given the fact that for $K_i = \mathbf{v}_i\mathbf{v}_i^T$ with $\mathbf{v}_i^T\mathbf{v}_i = \delta_{ij}$, $\sum_{i=1}^{m} \mu_i K_i \succeq 0 \Longleftrightarrow \mu \geq 0$; and $\frac{1}{r_i}\alpha^T G(K_{i,tr}\alpha) = (\hat{\mathbf{v}}_i^T\alpha)^2$ by applying Theorem B with $\tau = 0$ and $C = \infty$.

The benefits provided by the restriction $K = \sum_{i=1}^{m} \mu_i K_i, \mu \geq 0$ are three-fold: First, positive weights μ_i yields a smaller set of kernel matrices, Second, the general SDP can be reduced to a QCQP. Third, the constraint can result in improved numerical stability, because that it prevents the algorithm from using large weights with opposite sign that cancel.

Theorem D Given a linearly separable labeled sample set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted by $y \in R^{n_{tr}}$, the kernel matrix $K = \sum_{i=1}^{m} \mu_i K_i$ that optimizes (35) with $\tau \geq 0$, under the additional constrain $\mu \geq 0$ can be found by solving the following convex optimization problem, and considering its dual solution:

$$\begin{aligned}\max_{K,t,\lambda,\mu} \quad & 2\alpha^T\mathbf{e} - ct \\ \text{s.t.} \quad & t \geq \frac{1}{r_i}\alpha^T G(K_{i,tr})\alpha, \\ & \alpha^T\mathbf{y} = 0, \;\; \alpha \geq 0. \end{aligned}$$

where $r \in R_m$ with $\text{trace}(K_i) = r_i$. This can be proved by applying Theorem B for $C = \infty$ and $\tau = 0$.

5.3.4 Apply to Soft Margin

Apply to 1-Norm Soft Margin For the case of nonlinearly separable data, we can consider the 1-norm soft margin cost function. The goal is to optimize this quantity with respect to the kernel matrix K, i.e.

$$\min_{K \in \mathcal{K}} w_{S1}(K_{tr}) \text{ s.t. } \text{trace}(K) = c \tag{39}$$

This is again a convex optimization problem.

Theorem E Given a labeled sample set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted by $y \in R^{n_{tr}}$, the kernel matrix $\mathbf{K} \in \mathcal{K}$ that optimizes (39), can be found by solving the following convex optimization problem:

$$\begin{aligned} \max_{K,t,\lambda,\mu,\delta} \quad & t \\ \text{s.t.} \quad & \text{trace}(K) = c, \\ & k \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) & \mathbf{e} + \nu - \delta + \lambda \mathbf{y} & \succeq 0 \\ (\mathbf{e} + \nu - \delta + \lambda \mathbf{y})^T & t - 2C\delta^T \mathbf{e} & \end{pmatrix} \\ & \nu \geq 0, \\ & \delta \geq 0. \end{aligned}$$

Here $w_{S1}(K_{tr}) = w_{C,0}(K_{tr})$. The above formula can be obtained y applying Theorem A for $\tau = 0$. Adding the additional constraint that K is a linear combination of fixed kernel matrices leads to the following SDP:

$$\begin{aligned} \min_{\mu_i,t,\lambda,\nu,\delta} \quad & t \\ \text{s.t.} \quad & \text{trace}(\sum_{i=1}^{m} \mu_i K_i) = c, \\ & \sum_{i=1}^{m} \mu_i K_i \succeq 0, \\ & \begin{pmatrix} G(\sum_{i=1}^{m} \mu_i K_{i,tr} & \mathbf{e} + \nu - \delta + \lambda \mathbf{y} \\ (\mathbf{e} + \nu - \delta + \lambda \mathbf{y})^T & t - 2C\delta^t \mathbf{e} \end{pmatrix} \succeq 0, \\ & \nu, \delta \geq 0. \end{aligned}$$

In the case that $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal, the SDP reduces to a QCQP using Theorem B, with $\tau = 0$, similar to the hard margin case:

$$\begin{aligned} \max_{\alpha,t} \quad & 2\alpha^T \mathbf{e} - ct \\ \text{s.t.} \quad & t \geq (\hat{\mathbf{v}}_i^T \alpha)^2, i = 1, \cdots, m, \\ & \alpha^T \mathbf{y} = 0, \\ & C \geq \alpha \geq 0. \end{aligned}$$

with $\hat{\mathbf{v}}_i = \text{diag}(\mathbf{y}) \mathbf{v}_i (1 : n_{tr})$).

Apply to 2-Norm Soft Margin For the case of nonlinearly separable data, we can also consider the 2-norm soft margin cost function. Optimize this quantity with respect to the kernel matrix $\mathbf{K}$

$$\min_{K\in\mathcal{K}} w_{S2}(K_{tr}) \text{ s.t. } \text{trace}(K) = c \tag{40}$$

is again a convex optimization problem, and can be restated as follows.

Theorem F Given a labeled sample set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted by $y \in R^{n_{tr}}$, the kernel matrix $K \in \mathcal{K}$ that optimizes (40) with $\tau \geq 0$ can be found by solving the following convex optimization problem:

$$\begin{aligned}
\min_{K,t,\lambda,\mu} \quad & t \\
\text{s.t.} \quad & \text{trace}(K) = c, \\
& K \in \mathcal{K}, \\
& \begin{pmatrix} G(K_{tr}) + \frac{1}{C} I_{n_{tr}} & \mathbf{e} + \nu + \lambda\mathbf{y} \\ (\mathbf{e} + \nu + \lambda\mathbf{y})^T & t \end{pmatrix} \succeq 0, \\
& \nu \geq 0.
\end{aligned}$$

Here $w_{S2}(K_{tr}) = w_{\infty,\tau}(K_{tr})$. The above formulas can be obtained by applying Theorem A for $C = \infty$.
If K is restricted to be a linear combination of fixed kernel matrices, we obtain

$$\begin{aligned}
\min_{\mu_i,t,\lambda,\nu} \quad & t \\
\text{s.t.} \quad & \text{trace}(\sum_{i=1}^{m} \mu_i K_i) = c, \\
& \sum_{i=1}^{m} \mu_i K_i \succeq 0, \\
& \begin{pmatrix} G(\sum_{i=1}^{m} \mu_i K_{i,tr} + \frac{1}{C} I_{n_{tr}}) & \mathbf{e} + \nu + \lambda\mathbf{y} \\ (\mathbf{e} + \nu + \lambda\mathbf{y})^T & t \end{pmatrix} \succeq 0, \\
& \nu \geq 0.
\end{aligned}$$

Further, in the case that K_is are rank-one matrices, $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal, we obtain a QCQP:

$$\begin{aligned}
\max_{\alpha,t} \quad & 2\alpha^T \mathbf{e} - \frac{1}{C}\alpha^T\alpha - ct \\
\text{s.t.} \quad & t \geq (\hat{\mathbf{v}}_i^T \alpha)^2, i = 1, \cdots, m \\
& \alpha^T \mathbf{y} = 0 \\
& \alpha \geq 0
\end{aligned} \tag{41}$$

6 Kernel Methods for Distance Metrics Learning

The last decade has witnessed substantial developments in kernel-based learning methods (e.g., [52]). Since an appropriate kernel choice can often generate dramatic improvements in classification accuracy, kernel learning has recently attracted significant interest. Early work in this area assumed parametric forms for kernels and learned optimal parameters for kernel functions by minimizing the classification error. For these parametric kernels method, a brief description of the main ideas of kernel feature spaces can be found in [53]; and [54] also gives an introduction to kernel-based learning algorithms. However, our emphasis in this survey is the more recent work on non-parametric approaches that allow for any positive kernel matrix. Representative work in this category includes kernel alignment [55] and its semi-definite programming approach [49]. We first give a review of kernel methods. Then details of [55] and [49]'s work are provided. In the end, we also consider learning the idealized kernel [17], which is an extension of [55] and [49].

6.1 A Review of Kernel Methods

We first would like to provide the general idea of Kernel Methods. Detailed introduction can be found in [56] and [57]. Kernel-based learning algorithms attempts to embed the data into a Hilbert space, and searching for linear relations in such a space. Performed implicitly, the embedding specifies the inner product between each pair of points instead of giving their coordinates explicitly. The advantages of kernel methods come from the fact that often the inner product in the embedding space can be computed much more easily than the coordinates of the points themselves.

Suppose we have a dataset $\mathbf{X}$ with n samples, and a map $\Phi : \mathbf{X} \to \mathbf{F}$, where $\mathbf{F}$ is an embedding space. the kernel matrix is defined as $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n$, and $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. In this way, $\mathbf{K}$ completely determines the relative positions between those points in the embedding space.

Proposition: Every positive definite and symmetric matrix is a kernel matrix, that is, an inner product matrix in some embedding space. Conversely, every kernel matrix is symmetric and positive definite.

6.2 Kernel Alignment

Kernel Alignment, as a measure of similarity between two kernel functions or between a kernel and a target function, is originally introduced by [55]. This quantity is designed to capture the degree of agreement between a kernel and a given learning task. [55] also proved that Kernel Alignment is sharply concentrated around its expected values.

Given an unlabeled sample set $\mathbf{S} = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{x}_i \in R^m$, we denote the inner product between its two kernel matrices based on kernel k_1 and kernel k_2 as,

$$\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F = \sum_{i,j=1}^{n} k_1(\mathbf{x}_i, \mathbf{x}_j) k_2(\mathbf{x}_i, \mathbf{x}_j)$$

where $\mathbf{K}_i$ is the kernel matrix for the sample $\mathbf{S}$ using kernel k_i. Then the alignment of kernel k_1 and kernel k_2 with respect to the sample $\mathbf{S}$ is defined as:

$$\hat{\mathbf{A}}(\mathbf{S}, k_1, k_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}} \tag{42}$$

When the label vector $\mathbf{y}$ ($y_i = \pm 1, \ \ i = 1, \cdots, n$) for the sample is known, $\mathbf{K}_2 = \mathbf{y}\mathbf{y}^T$ can be considered as the target kernel where $k_2(\mathbf{x}_i; \mathbf{x}_j) = 1$ if $y_i = y_j$ and $k_2(\mathbf{x}_i; \mathbf{x}_j) = -1$ if $y_i \neq y_j$. The alignment of a kernel k_1 with k_2 with respect to the sample set S can be considered as a quality measure for kernel k_1:

$$\hat{\mathbf{A}}(\mathbf{S}, \mathbf{K}_1, yy^T) = \frac{\langle \mathbf{K}_1, yy^T \rangle}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle \langle yy^T, yy^T \rangle}} = \frac{\langle \mathbf{K}_1, yy^T \rangle}{m\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle}}$$

since $\langle yy^T, yy^T \rangle = m^2$.

Concentration and Generalization Concentration means the alignment is not too dependent on the training set S, or say, the probability of an empirical estimate deviating from its mean can be bounded as an exponentially decaying function of that deviation. This suggests that the alignment on a training set can be optimized and expected to keep high alignment and good performance on a test set. Details about the proof of concentration and generalization for kernel alignment can be found in [55].

6.3 Kernel Alignment with SDP

Based on the concept of kernel alignment defined by [55], [49] considers to optimizing the alignment between a set of labels and a kernel matrix using SDP in a transductive setting. [49] also shows that, if K is a class of linear combinations of fixed kernel matrices, this problem can still be cast as an SDP.

Suppose we work in a transduction setting, where the labeled training set is $S_{n_{tr}} = \{(\mathbf{x}_1; y_1), \cdots, (\mathbf{x}_{n_{tr}}; y_{n_{tr}})\}$, and the unlabeled testing set is $T_{n_t} = \{(\mathbf{x}_{n_{tr}+1}, \cdots, \mathbf{x}_{n_{tr}+n_t})\}$. The corresponding kernel matrix is

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{tr} & \mathbf{K}_{tr,t} \\ \mathbf{K}_{tr,t}^T & \mathbf{K} \end{pmatrix}$$

where $\mathbf{K}_{ij} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, i, j = 1, \cdots, n_{tr}, n_{tr}+1, \cdots, n_{tr}+n_t$.

Consistent with the general goal of transductive learning, which is to label the unlabeled data, our specific goal in this setting is to learn the optimal mixed block $\mathbf{K}_{tr,t}$, and the optimal testing data block $\mathbf{K}_t$, by optimizing a cost function over the training data block $\mathbf{K}_{tr}$. In this optimization problem, training and testing data blocks are entangled, and optimizing the embedding of training data entries in $\mathbf{K}$ results in the automatic tuning of testing data entries.

The capacity of the search space of possible kernel matrices should be controlled to prevent overfitting and achieve good generalization on test data. Below we first provide a general optimization problem in which the kernel matrix $\mathbf{K}$ is restricted to a convex subset, the positive semidefinite cone. Then we consider a specific example, where the search space of $\mathcal{K}$ lies in the intersection of a low-dimensional linear subspace with the positive semidefinite cone.

Case 1: The kernel matrix $\mathbf{K}$ is restricted to the positive semidefinite cone. We denote $\mathcal{K} = \{\mathbf{K} \succeq 0\}$ as the set of all positive semidefinite matrices. The kernel matrix $\mathbf{K} \in \mathcal{K}$ which is maximally aligned with the label vector $\mathbf{y} \in R^{n_{tr}}$ can be found by solving the following optimization problem:

$$\begin{aligned} \underset{\mathbf{K}}{max} \quad & \hat{\mathbf{A}}(\mathbf{S}, \mathbf{K}_1, yy^T) \\ \text{s.t.} \quad & \mathbf{K} \in \mathcal{K}, \ \text{trace}(\mathbf{K}) = 1. \end{aligned}$$

which is equivalent to

$$\begin{aligned} \underset{\mathbf{A},\mathbf{K}}{max} \quad & \langle \mathbf{K}_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{s.t.} \quad & \text{trace}(\mathbf{K}) = 1, \\ & \mathbf{K} \in \mathcal{K}, \ \langle \mathbf{K}, \mathbf{K} \rangle_F = 1. \end{aligned} \tag{43}$$

By introducing a matrix $\mathbf{A}$ that satisfies $\mathbf{K}^T\mathbf{K} \preceq \mathbf{A}$ and trace$(A) \leq 1$, (43) can be equivalently written as

$$\begin{aligned} \underset{\mathbf{A},\mathbf{K}}{max} \quad & \langle \mathbf{K}_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{s.t.} \quad & \text{trace}(\mathbf{A}) \leq 1, \\ & \begin{pmatrix} \mathbf{A} & \mathbf{K}^T \\ \mathbf{K} & I_n \end{pmatrix} \succeq 0, \\ & \mathbf{K} \in \mathcal{K}. \end{aligned} \tag{44}$$

where I_n is the $n \times n$identity matrix.

Case 2: $\mathbf{K}$ belongs to the set of positive semidefinite matrices with bounded trace that can be expressed as a linear combination of fixed kernel matrices from the set $\{\mathbf{K}_1, \cdots, \mathbf{K}_g\}$, and the parameters μ_i are constrained to be non-negative. In this way, $\mathcal{K}$ is the subset satisfying

$$\begin{aligned} & \mathbf{K} = \sum_{i=1}^{g} \mu_i \mathbf{K}_i \\ & \mathbf{K} \succeq 0 \\ & \mu_i \geq 0, \ i = 1, \cdots, g \\ & \text{trace}(\mathbf{K} \leq c) \end{aligned} \tag{45}$$

The set $\{\mathbf{K}_1, \cdots, \mathbf{K}_g\}$ is the inintial guess of the kernel matrix $\mathbf{K}$. It could be linear, Gaussian or polynomial kernels with different kernel parameter values. It can also be rank-one matrices $\mathbf{K}_i = \mathbf{v}_i\mathbf{v}_i^T$, where $\mathbf{v}_i$ some set of orthogonal vectors. Consequently, the task turns out to be optimizing the weights in the linear combination of the obtained kernel matrices. And as we can see, this optimization problem has significantly reduced computational complexity.

Adding the additional constraint (45) to (43) leads to

$$\begin{aligned} \max_{\mathbf{K}} \quad & \langle \mathbf{K}_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{s.t.} \quad & \langle \mathbf{K}, \mathbf{K} \rangle_F \leq 1, \\ & \mathbf{K} \succeq 0, \\ & \mathbf{K} = \sum_{i=1}^{g} \mu_i \mathbf{K}_i. \end{aligned} \tag{46}$$

And (46) can be written in the standard form of a SDP, in a similar way as for (44).

$$\begin{aligned} \max_{\mathbf{A},\mu_i} \quad & \langle \sum_{i=1}^{g} \mu_i \mathbf{K}_{i,tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{s.t.} \quad & \text{trace}(\mathbf{A}) \leq 1, \\ & \begin{pmatrix} \mathbf{A} & \sum_{i=1}^{g} \mu_i \mathbf{K}_i^T \\ \sum_{i=1}^{g} \mu_i \mathbf{K}_i & I_n \end{pmatrix} \succeq 0, \\ & \sum_{i=1}^{g} \mu_i \mathbf{K}_i \succeq 0. \end{aligned}$$

For the specific case where $\mathbf{K}_i$ are rank-one matrices $\mathbf{K}_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthnormal, the semidefinite program reduces to a QCQP:

$$\begin{aligned} \max_{\mu_i} \quad & \sum_{i=1}^{g} \mu_i (\hat{\mathbf{v}}_i^T \mathbf{y})^2 \\ \text{s.t.} \quad & \sum_{i=1}^{g} \mu_i^2 \leq 1, \\ & \mu_i \geq 0, \ i = 1, \cdots, g. \end{aligned}$$

with $\hat{\mathbf{v}}_i = \mathbf{v}_i(1 : n_{tr})$.

6.4 Learning with Idealized Kernel

Derived from the work in [55] and [49], [17] applys non-parametric kernel methods to distance metric learning. Their intuition is, although obtaining the ideal kernel function is infeasible, however, in classification problem, ideal kernel matrix can be defined on the training patterns based on the provided label information. Then they idealize a given kernel by making it more similar to the ideal kernel matrix. They formulate this as a distance metric learning problem that searches for a suitable linear transform in the kernel-induced feature space. The challenging issue is how to generalize this to patterns outside the training dataset.

6.4.1 Ideal Kernel

Kernel defines the pairwise similarity between patterns. For a two-class classfication problem with training set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i \in R^m$ and $y \in \{-1, +1\}$. Ideally, two patterns should be considered similar iff they belong to the same class. Therefore, [17] introduced the concept of "ideal kernel" $\mathbf{K}^*$ as below.

$$\mathbf{K}^*_{ij} = \mathbf{K}^*(x_i, x_j) = \begin{cases} 1 & y(x_i) = y(x_j) \\ 0 & y(x_i) \neq y(x_j) \end{cases} \tag{47}$$

[17] and uses "alignment" to access the quality of a kernel. Following the definition of kernel alignment in (42), the alignment of a kernel k_1 with k_2 w.r.t. the sample is

$$A(k_1, k_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle}\sqrt{\langle \mathbf{K}_2, \mathbf{K}_2 \rangle}}$$

where $\mathbf{K}_i$ is the corresponding kernel matrix of the kernel k_i.

When $\mathbf{K}_2^*$ is ideal, this kernel target alignment can be used to measure the similarity between the kernel and the target, and we can idealize a given kernel $\mathbf{K}_1$ by making it more similar to the ideal kernel. Moreover, if a high alignment on the training dataset is obtained, we also expect a high alignment on the test dataset, which means high alignment implies good generalization performance of the resulting classifier.

6.4.2 Idealized Kernel

To idealize a given kernel $\mathbf{K}$ means to make it more similar to the ideal kernel $\mathbf{K}^*$. A simple way to modify $\mathbf{K}$ is,

$$\hat{\mathbf{K}} = \mathbf{K} + \frac{\gamma}{2}\mathbf{K}^* \tag{48}$$

where $\gamma \geq 0$ is a parameter to be determined. For a two-class problem, this leads to

$$\hat{\mathbf{K}}_{ij} = \begin{cases} \mathbf{K}_{ij} + \frac{\gamma}{2} & y_i = y_j \\ \mathbf{K}_{ij} & y_i \neq y_j \end{cases}$$

As both $\mathbf{K}$ and $\mathbf{K}^*$ are valid kernels, so is the idealized kernel $\hat{\mathbf{K}}$.

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

As $\gamma > 0$, so long as $\langle \mathbf{K}, \mathbf{K}^* \rangle \geq 0$ (i.e. K is aligned in the "right" direction), or slightly wrongly ($\langle \mathbf{K}, \mathbf{K}^* \rangle$ is a small negative number, then the idealized kernel $\hat{\mathbf{K}}$ will have an increased alignment.

Extending to C classes ($C > 2$), $\mathbf{K}^*$ will take a form of block diagonal as below:

$$K^* = \begin{bmatrix} \mathbf{11}'_{n_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{11}'_{n_2} & \cdots & \mathbf{0} \\ \vdots & \cdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{11}'_{n_c} \end{bmatrix}$$

where n_i is the number of patterns belonging to the i_{th} class, and $\mathbf{1}\mathbf{1}'_{n_i}$ is a $n_i \times n_i$ matrix with all ones. Both $\mathbf{K}^*$ and $\hat{\mathbf{K}}$ are valid kernels, and $\hat{\mathbf{K}}$ will have a higher alignment than $\mathbf{K}$ if $\gamma > -\frac{\langle \mathbf{K},\mathbf{K}^*\rangle}{\sum_{i=1}^{C} n_i^2}$.

6.4.3 Formulation as Distance Metric Learning

First, we look at the linear kernel case. Assume that the original inner product for any two pattern $\mathbf{x}_i$, $\mathbf{x}_j$ in input space R^m is defined as $s_{ij} = \mathbf{K}_{ij} = \mathbf{x}_i^T \mathbf{M} \mathbf{x}_j$, where $\mathbf{M}_{m\times m}$ is a positive semi-definite matrix. The the corresponding squared distance is $d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)$. By changing $\mathbf{K}$ to $\hat{\mathbf{K}}$, this squared distance will be changed to

$$\hat{\mathbf{K}}_{ii} + \hat{\mathbf{K}}_{jj} - 2\hat{\mathbf{K}}_{ij} = \begin{cases} d_{ij}^2 & y_i = y_j \\ d_{ij}^2 + \gamma & y_i \neq y_j \end{cases} \tag{49}$$

We modify the inner product s_{ij} to $\hat{s}_{ij} = \mathbf{x}_i \mathbf{A}\mathbf{A}^T \mathbf{x}_j$, where $A_{m\times m} = [\mathbf{a}_1, \cdots, \mathbf{a}_m]$. Here $\mathbf{a}_i$'s are a set of "useful" directions in the input space. Consequently, the corresponding distance metric becomes:

$$\hat{d}_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A}\mathbf{A}' (\mathbf{x}_i - \mathbf{x}_j) \tag{50}$$

Here $\mathbf{A}\mathbf{A}^T$ is positive semi-definite, and $\hat{d}$ is always a valid distance metric.

Now we search for a matrix $\mathbf{A}$ such that

$$\hat{d}_{ij}^2 \begin{cases} \leq d_{ij}^2 & y_i = y_j \\ \geq d_{ij}^2 + \gamma & y_i \neq y_j \end{cases} \tag{51}$$

In another word, patterns in different classes will be pulled apart by an amount at least equal to γ under the modified distance metric, while those in the same class may get close together. This is consistent with the slogan of "reducing intra-class variability and increasing inter-class variability". To extend the above formulation to the case where only similarity information is available, we denote the sets containing similar and dissimilar pairs by S and D repectively. Then (51) can be modified to:

$$\hat{d}_{ij}^2 - d_{ij}^2 \begin{cases} \leq 0 & (\mathbf{x}_i, \mathbf{x}_j) \in S, \\ \geq \gamma & (\mathbf{x}_i, \mathbf{x}_j) \in D. \end{cases} \tag{52}$$

6.4.4 Primal and Dual

Matrix $\mathbf{A}$ performs a projection onto a set of useful features. This set should ideally be small, which means that a small rank for $\mathbf{A}$ or $\mathbf{A}\mathbf{A}^T$ will be desirable (given that $rank(\mathbf{A}) = rank(\mathbf{A}\mathbf{A}^T)$). Assume that the eigen decomposition of $\mathbf{A}\mathbf{A}^T$ is $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$. Then $rank(\mathbf{A}\mathbf{A}^T) = rank(\Sigma) = \|\Sigma\|_0$. To simplify the problem, $\|\Sigma\|_0$ is approximated by the Euclidean norm $\|\Sigma\|_2 = \|\mathbf{A}\mathbf{A}^T\|_2$. This leads to the following primal problem:

$$\begin{aligned}
\min_{\mathbf{A},\gamma,\xi_{ij}} \quad & \frac{1}{2}\|\mathbf{A}\mathbf{A}^T\|_2^2 + \frac{C_S}{N_S}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S}\xi_{ij} + C_D(-\nu\gamma + \frac{1}{N_D}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in D}\xi_{ij}) \\
\text{s.t.} \quad & d_{ij}^2 \geq \hat{d}_{ij}^2 - \xi_{ij},\ (\mathbf{x}_i,\mathbf{x}_j)\in S, \\
& \hat{d}_{ij}^2 - d_{ij}^2 \geq \gamma - \xi_{ij},\ (\mathbf{x}_i,\mathbf{x}_j)\in D, \\
& \xi_{ij} \geq 0, \\
& \gamma \geq 0.
\end{aligned}$$

Here N_S and N_D are the number of pairs in S and D respectively, and C_S, C_D, ν are non-negative adjustable parameters. Its dual problem is,

$$\begin{aligned}
\max_{\alpha_{ij}} \quad & \{-\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S}\alpha_{ij}(\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i-\mathbf{x}_j) + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in D}\alpha_{ij}(\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i-\mathbf{x}_j) \\
- \quad & \frac{1}{2}\sum_{(\mathbf{x}_i,\mathbf{x}_j),(\mathbf{x}_k,\mathbf{x}_l)\in S}\alpha_{ij}\alpha_{kl}((\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_k,\mathbf{x}_l))^2 \\
- \quad & \frac{1}{2}\sum_{(\mathbf{x}_i,\mathbf{x}_j),(\mathbf{x}_k,\mathbf{x}_l)\in D}\alpha_{ij}\alpha_{kl}((\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_k,\mathbf{x}_l))^2 \\
+ \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S}\sum_{(\mathbf{x}_k,\mathbf{x}_l)\in D}\alpha_{ij}\alpha_{kl}((\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_k-\mathbf{x}_l))^2\} \\
\text{s.t.} \quad & \\
& \frac{1}{C_D}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in D}\alpha_{ij} \geq \nu, \\
& 0 \leq \alpha_{ij} \leq \begin{cases} \frac{C_S}{N_S} & (\mathbf{x}_i,\mathbf{x}_j)\in S \\ \frac{C_D}{N_D} & (\mathbf{x}_i,\mathbf{x}_j)\in D \end{cases}
\end{aligned}$$

where α_{ij}'s are the lagrangian multipliers. This is a standard quadratic programming (QP) problem with $N_S + N_D$ variables which is independent of the dimension of $\mathbf{x}$, and will not suffer from local optimum.

Comparable to [17], [11] is another distance metric learning algorithm that learns a full distance matrix bases on similarity information. We have reviewed [11] in section 2. When a full matrix is used, [11]'s method leads to a convex programming problem with m^2 variables. It also involves an iterative procedure comprising projection and eigen decomposition. When m is high, [11] is more costly, compared with the idealized kernel learning method.

6.4.5 Kernelization

As we may notice, only inner products are required in [17]'s formulation. We replace all the $\mathbf{x}$ with $\varphi(\mathbf{x})$, where φ is the feature map corresponding to the original kernel $\mathbf{K}$. By applying the kernel trick, the idealized kernel $\hat{\mathbf{K}}$ is given by:

$$\hat{\mathbf{K}}(\mathbf{x}_a,\mathbf{x}_b) = -\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S}\alpha_{ij}(\mathbf{K}_{ai}-\mathbf{K}_{aj})(\mathbf{K}_{bi}-\mathbf{K}_{jb}) + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in D}\alpha_{ij}(\mathbf{K}_{ai}-\mathbf{K}_{aj})(\mathbf{K}_{bi}-\mathbf{K}_{jb})$$

and the distance metric is

$$\hat{d}^2(\mathbf{x}_a, \mathbf{x}_b) = -\sum_{(\mathbf{x}_a,\mathbf{x}_b)\in S} \alpha_{ij}(\mathbf{K}_{ai} - \mathbf{K}_{aj} - \mathbf{K}_{bi} + \mathbf{K}_{bj})^2$$

$$+ \sum_{(\mathbf{x}_a,\mathbf{x}_b)\in D} \alpha_{ij}(\mathbf{K}_{ai} - \mathbf{K}_{aj} - \mathbf{K}_{bi} + \mathbf{K}_{bj})^2$$

7 Conclusions

This survey provides a comprehensive review of problems and algorithms in distance metric learning that look particularly interesting from a principle point of view. We categorize the disparate issues in distance metric learning. Within each category, we summarize existing work, disclose their essential connections, strengths and weaknesses. Listed below are categories we have addressed:

- Supervised distance metric learning, which contains supervised global distance metric learning, local adaptive supervised distance metric learning, Neighborhood Components Analysis, and Relevant Component Analysis.
- Unsupervised distance metric learning, covering linear (PCA, MDS)and nonlinear embedding methods (ISOMAP, LLE, and Laplacian Eigenmaps). We further unify these algorithms into a common framework based on the embedding computation.
- Maximum margin based distance metric learning approaches, including the large margin nearest neighbor based distance metric learning methods and SDP methods to solve the kernelized margin maximization problem.
- Kernel methods towards distance metrics, covering kernel alignment and its SDP approaches, and also the extension work of learning the idealized kernel.

Although significant work has been done in this field, a range of problems requiring practical and feasible solution that is truly optimal, have not been well addressed. A few of them are listed as below.

- Unsupervised distance metric learning. Although dimension reduction has been well explored, unsupervised distance metric learning is a more general problem than dimension reduction. Beyond dimension reduction, it has many other purposes, such as clustering and maximizing margin. Currently, no general principle framework has been set up to learn a distance metric without pairwise constraints and side-information.
- Going local in a principle manner. Previous research on distance metric learning has mainly focused on finding a linear distance metric that optimizes the data compactness and separability in a *global* sense, namely bringing *all* of the data points from the same classes close together as well as separating *all* of the data points from different classes. However, a global distance metric may not be realistic given that the two optimization goals often conflict with each other when classes exhibit multi-modal data distributions. Instead, focusing on local compactness is an effective way to avoid this conflict and enhance metric quality. Local Adaptive methods in [18, 7] are a group of work trying to go local, to find feature weights adapted to individual test examples. However, their local

distance metrics are hand crafted and are therefore unable to take full advantage of the available training data. Another related work NCA [9] suffers from the scalability problem. Consequently, we need more principle approaches to learn local distance metrics.

- Learn an explicit nonlinear distance metric in the local sense. No current nonlinear dimension reduction approaches can learn an explicit nonlinear metric. Therefore, an interesting question to ask is how to learn an explicit nonlinear distance metric from the local pairwise constraints by optimizing the *local compactness* and *local separability*.
- Efficiency issue. Recall the global distance metric learning method [11], Neighborhood Component Analysis (NCA) [9] and Large Margin Nearest Neighbor classifier [10]. They all formulate distance metric learning as a constrained convex programming problem, and attempt to learn complete distance metrics from the training data. This is computationally expensive and prone to overfitting, especially when applied to tasks with limited number of training samples, and the feature dimension is large, e.g. online Medical Image retrieval. Therefore, one tough problem of distance metric learning algorithms is, how to simplify the learning problem while improving the performance with less training samples and larger dimensions.

Conclusively, as an essential technique in machine learning research, distance metric learning problem has not been well addressed so far. There is still much work to be done and many interesting questions remaining.

References

[1] X. He, O. King, W.-Y. Ma, M. Li, and H. J. Zhang, “Learning a semantic space from user’s relevance feedback for image retrieval,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 1, 2003.

[2] X. He, W.-Y. Ma, and H.-J. Zhang, “Learning an image manifold for retrieval,” in *Proc. ACM Multimedia*, 2004.

[3] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, “Manifold ranking based image retrieval,” in *Proc. ACM Multimedia*, 2004.

[4] H. Muller, T. Pun, and D. Squire, “Learning from user behavior in image retrieval: Application of market basket analysis,” *International Journal of Computer Vision*, vol. 56, no. 1–2, 2004.

[5] R. Yan, A. Hauptmann, and R. Jin, “Negative pseudo-relevance feedback in content-based video retrieval,” in *Proc. ACM Multimedia*, 2003.

[6] T. Hastie and R. Tibshirani, “Discriminant adaptive nearest neighbor classification,” *IEEE Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, 1996.

[7] C. Domeniconi and D. Gunopulos, “Adaptive nearest neighbor classification using support vector machines,” *Proc. NIPS*, 2002.

[8] J. Peng, D. Heisterkamp, and H. Dai, “Adaptive kernel metric nearest neighbor classification,” *Proc. International Conference on Pattern Recognition*, 2002.

[9] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, “Neighbourhood components analysis,” in *Proc. NIPS*, 2005.

[10] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. NIPS*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 1475–1482.

[11] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Proc. NIPS*, 2003.

[12] G. Lebanon, "Flexible metric nearest neighbor classification," in *Proc. Uncertainty in Artificial Intelligence*, 2003.

[13] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning distance functions using equivalence relations," in *Proc. International Conference on Machine Learning*, 2003.

[14] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, 1996.

[15] P. Gill, W. Murray, and M. Wright, *Practical Optimization*. Academic Press, 1981.

[16] Z. Zhang, J. Kwok, and D. Yeung, "Parametric distance metric learning with label information," in *Proc. International Joint Conference on Artificial Intelligence*, 2003.

[17] J. T. Kwok and I. W. Tsang, "Learning with idealized kernels," *Proc. International Conference on Machine Learning*, 2003.

[18] J. Friedman, "Flexible metric nearest neighbor classification," Stanford University Statistics Department., Tech. Rep., 1994.

[19] K. Zhang, M. Tang, and J. T. Kwok, "Applying neighborhood consistency for fast clustering and kernel density estimation." in *Proc. Computer Vision and Pattern Recognition*, 2005, pp. 1001–1007.

[20] R. O.Duda and P. E. Hart, *Pattern classification and scene analysis*. New York: Wiley, 1973.

[21] T. Cover and P. Hart, "Nearest neighbor pattern classification," in *IEEE transaction on Information Theory*, 1967, pp. 21–27.

[22] T.M.Cover, "Rates of covergence of nearest neighbor procedures," in *Hawaii Inter. Conf. on System Sciences*, Western Periodicals, Honolulu, 1968, pp. 413–415.

[23] G. J. Mclachlan, *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley, 1992.

[24] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally adaptive metric nearest-neighbor classificaton," *IEEE Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1281–1285, 2002.

[25] S. Bermejo and J. Cabestany, "Large margin nearest neighbor classifiers," in *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks*. London, UK: Springer-Verlag, 2001, pp. 669–676.

[26] R. Short and K.Fukanaga, "A new nearest distance measure," in *Proc. International Conference on Pattern Recognition*, 1980, pp. 81–86.

[27] R. Short and K. Fukanaga, "The optimal distanc measure for nearest neighbor classification," in *IEEE transaction on Information Theory*, 1981, pp. 622–627.

[28] A. M. C. Atkeson and S. Schaal, "Locally weighted learning," *AI Review*, 1996.

[29] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in *Proc. of the European Conference on Computer Vision*. London, UK: Springer-Verlag, 2002, pp. 776–792.

[30] J. T. K. Ivor W. Tsang, Pak-Ming Cheung, "Kernel relevant component analysis for distance metric learning," in *Proc. of the International Joint Conference on Neural Networks*, 2005.

[31] R. Linsker, "An application of the principle of maximum information preservation to linear systems," in *Proc. NIPS*, 1989, pp. 186–194.

[32] J. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[33] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[34] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[35] T. Cox and M. Cox, "Multidimensional scaling," in *London: Chapman and Hall*, 1994.

[36] J. Zhang, S. Li, and J. Wang, "Manifold learning and applications in recognition," in *Intelligent Multimedia Processing with Soft Computing. Springer-Verlag, Heidelberg.*, 2004.

[37] V. de Silva and J. Tenenbaum, "Unsupervised learning of curved manifolds," in *the MSRI workshop on nonlinear estimation and classification*. New York: Springer Verlag, 2002.

[38] S. T. Roweis and K. S. Lawrance, "Nonlinear dimensionality reduction by locally linear embedding," in *Science*, vol. 290, 2000, pp. 2323–2326.

[39] X. He and P. Niyogi, "Locality preserving projections," in *Proc. NIPS*, 2003.

[40] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," 2002, pp. 833–840.

[41] M. Brand, "Charting a manifold," in *Proc. NIPS*, vol. 15, 2003, pp. 961–968.

[42] T. Hastie and W. Stuetzle, "Principla curves," in *Journal of the American Statistical Association,84(406)*, 1988, pp. 502–516.

[43] T. L. B. Kégl, A. Krzyzak and K. Zeger, "Learning and design of principal curves," in *IEEE Pattern Analysis and Machine Intelligence*, no. 3, 2000, pp. 281–297.

[44] M. S. C.M. Bishop and C. Williams, "Gtm:the generative topographic mapping," in *Neural Computation*, vol. 10, 1998, pp. 215–234.

[45] K. Chang and J. Ghosh, "A unified model for probabilistic principal surfaces," in *IEEE Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, 2001, pp. 22–41.

[46] A. J. Smola, R. C. Williamson, S. Mika, and B. Schölkopf, "Regularized principal manifolds," in *the 4th European Conference on Computational Learning Theory*, 1999, pp. 214–229.

[47] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," in *Proc. NIPS*. Cambridge, MA: MIT Press, 2004.

[48] J. M. I. Ng, A. Y. and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. NIPS*, 2002.

[49] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[50] C. Cortes and V. Vapnik, "Support-vector network," in *Machine Learning*, 1995, pp. 273–297.

[51] I. G. Boser, B. and V. Vapnik, "A training algorithm for optimal margin classifiers," in *the Fifth Annual Workshop on Computational Learning Theory*, 1992.

[52] B. Schölkopf and A. Smola, *Learning with Kernels*. MIT Press., 2002.

[53] B. Schölkopf, *Statistical learning and kernel methods. Data Fusion and Perception Technical report*. Redmond, WA: Springer, 2000.

[54] G. R. K. T. K.R.muller, S. Mika and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Neural Networks*, vol. 12(2), pp. 181–201, 2001.

[55] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *NIPS*, 2001, pp. 367–373.

[56] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press, 2000.

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