Report Number: WUCSE-2013-28

2013

Kernel Density Metric Learning

Authors: Yujie He, Wenlin Chen, and Yixin Chen

This paper introduces a supervised metric learning algorithm, called kernel density metric learning (KDML), which is easy to use and provides nonlinear, probability-based distance measures. KDML constructs a direct nonlinear mapping from the original input space into a feature space based on kernel density estimation. The nonlinear mapping in KDML embodies established distance measures between probability density functions, and leads to correct classification on datasets for which linear metric learning methods would fail. Existing metric learning algorithms, such as large margin nearest neighbors (LMNN), can then be applied to the KDML features to learn a Mahalanobis distance. We also propose an integrated optimization algorithm that learns not only the Mahalanobis matrix but also kernel bandwidths, the only hyper-parameters in the nonlinear mapping. KDML can naturally handle not only numerical features, but also categorical ones, which is rarely found in previous metric learning algorithms. Extensive experimental results on various benchmark datasets show that KDML significantly improves existing metric learning algorithms in terms of kNN classification accuracy.
... **Read complete abstract on page 2.**

Kernel Density Metric Learning

Complete Abstract:

This paper introduces a supervised metric learning algorithm, called kernel density metric learning (KDML), which is easy to use and provides nonlinear, probability-based distance measures. KDML constructs a direct nonlinear mapping from the original input space into a feature space based on kernel density estimation. The nonlinear mapping in KDML embodies established distance measures between probability density functions, and leads to correct classification on datasets for which linear metric learning methods would fail. Existing metric learning algorithms, such as large margin nearest neighbors (LMNN), can then be applied to the KDML features to learn a Mahalanobis distance. We also propose an integrated optimization algorithm that learns not only the Mahalanobis matrix but also kernel bandwidths, the only hyper-parameters in the nonlinear mapping. KDML can naturally handle not only numerical features, but also categorical ones, which is rarely found in previous metric learning algorithms. Extensive experimental results on various benchmark datasets show that KDML significantly improves existing metric learning algorithms in terms of kNN classification accuracy.

2013-28

Kernel Density Metric Learning

Authors: Yujie He, Wenlin Chen, Yixin Chen

Corresponding Author: wenlinchen@wustl.edu

Abstract: This paper introduces a supervised metric learning algorithm, called kernel density metric learning (KDML), which is easy to use and provides nonlinear, probability-based distance measures. KDML constructs a direct nonlinear mapping from the original input space into a feature space based on kernel density estimation. The nonlinear mapping in KDML embodies established distance measures between probability density functions, and leads to correct classification on datasets for which linear metric learning methods would fail. Existing metric learning algorithms, such as large margin nearest neighbors (LMNN), can then be applied to the KDML features to learn a Mahalanobis distance. We also propose an integrated optimization algorithm that learns not only the Mahalanobis matrix but also kernel bandwidths, the only hyper-parameters in the nonlinear mapping. KDML can naturally handle not only numerical features, but also categorical ones, which is rarely found in previous metric learning algorithms. Extensive experimental results on various benchmark datasets show that KDML significantly improves existing metric learning algorithms in terms of kNN classification accuracy.

Type of Report: Other

Kernel Density Metric Learning

Yujie He
Department of Computer Science and Engineering
Washington University, St. Louis, USA
yujie.he@wustl.edu

Wenlin Chen
Department of Computer Science and Engineering
Washington University, St. Louis, USA
wenlinchen@wustl.edu

Yixin Chen
Department of Computer Science and Engineering
Washington University, St. Louis, USA
chen@cse.wustl.edu

ABSTRACT

This paper introduces a supervised metric learning algorithm, called kernel density metric learning (KDML), which is easy to use and provides nonlinear, probability-based distance measures. KDML constructs a direct nonlinear mapping from the original input space into a feature space based on kernel density estimation. The nonlinear mapping in KDML embodies established distance measures between probability density functions, and leads to correct classification on datasets for which linear metric learning methods would fail. Existing metric learning algorithms, such as large margin nearest neighbors (LMNN), can then be applied to the KDML features to learn a Mahalanobis distance. We also propose an integrated optimization algorithm that learns not only the Mahalanobis matrix but also kernel bandwidths, the only hyper-parameters in the nonlinear mapping. KDML can naturally handle not only numerical features, but also categorical ones, which is rarely found in previous metric learning algorithms. Extensive experimental results on various benchmark datasets show that KDML significantly improves existing metric learning algorithms in terms of kNN classification accuracy.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Experimentation, Algorithms, Performance

Keywords

metric learning; k-nearest neighbor classification; kernel density estimation

1. INTRODUCTION

Learning a distance metric is a fundamental problem in machine learning and data mining. In many applications, once we have defined a good distance or similarity measure between all pairs of data points, the data mining tasks would become trivial. For example, with a perfect distance metric, the k-nearest neighbor (kNN) algorithm can achieve perfect classification. As a result, ever since metric learning is proposed by Xing et al. [24], there has been extensive research in this area [5, 10, 11, 21, 22]. These new methods greatly improved the performance of many metric-based algorithms and gained lots of popularity.

There are several basic desirable properties for any metric learning algorithm: 1) it must reflect the true distance or similarity between data samples; 2) it needs to be flexible to support different learning settings and data types; 3) it should be able to generalize to out-of-sample data; 4) it should be easy to use and does not require extensive parameter tuning. Few existing algorithms can satisfy all these requirements.

A vast majority of existing methods are based on a linear transformation. Namely, they learn a Mahalanobis distance between two data points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^D$ in the form of

$$d_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{L}(\mathbf{x}_i - \mathbf{x}_j) \|_2, \quad (1)$$

where $\| \cdot \|_2$ is the ℓ_2-norm and $\mathbf{L} \in \mathcal{R}^{D \times D}$ is a matrix. Therefore, $\mathbf{L}$ represents a linear transformation of the input space, which corresponds to rotating and scaling the data points. Many representative metric learning algorithms, such as distance metric learning [24], large margin nearest neighbors (LMNN) [22], information theoretic metric learning (ITML) [5], neighborhood components analysis (NCA) [11], and SEPAPH [17], are based on such linear transformation and ℓ_2 Euclidean distance.

A main reason for the popularity of linear metric learning is its good off-the-shelf usability. However, linear metric learning has inherent limits on their mapping capability. Nonlinear metric learning is more general and offers greater separation ability in theory. For example, for the four points in two classes in Figure 1.a), no linear metric learning methods can give correct kNN classification. For example, the linear transformation in Figure 1.c) only rotates and scales the data, so does the LMNN mapping in Figure 1.d). We can see that kNN classification on the mapped data points in Figure 1.c) and 1.d) still cannot separate the two classes correctly. However, our nonlinear transformation (to be explained in Section 6.1) can map the four points to the coordinates in Figure 1.b), which enable correct kNN classification.

Nonlinear metric learning methods, although more expressive, are far less popular than linear methods. Often, they are not easy to use, since they require complex computa-

KDD'13, August 11-14, 2013 in Chicago

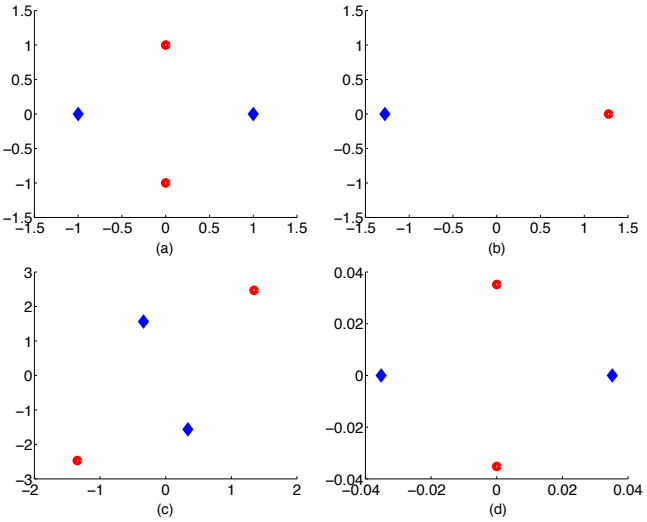

Figure 1: An toy example with four points in two classes, marked in different shapes. a) shows the original data; b) shows the data after our KDML mapping (the two points in each class are very close to each other); c) shows a random linear transformation; d) shows the data after LMNN mapping.

tion, not only for coefficient training, but also for model selection and hyper-parameter tuning. For example, kernelization methods [3,7,12,21] are inherently limited by the size of the kernel matrices. Neural network based methods [4] are also very expensive. Furthermore, these nonlinear methods often require tuning of many hyper-parameters. Their sensitivity to the parameter tuning further hinders their off-the-shelf usability, especially for unknown domains. Recently, Kedem et al. proposed two nonlinear metric learning algorithms χ^2-LMNN and GB-LMNN [13]. χ^2-LMNN still uses the linear transformation in (1) but employs a non-Euclidean distance. However, χ^2-LMNN has rather limited scope: it can only be applied when the input data are sampled from a simplex $S^D = \{\mathbf{x} \in \mathcal{R}^D | \mathbf{x} \geq 0, \mathbf{x}^T\mathbf{1} = 1\}$. GB-LMNN learns a nonlinear mapping $\phi(\mathbf{x})$ which is an ensemble over a number of regression trees with different heights. GB-LMNN applies gradient boosting to learn the nonlinear mapping in a function space.

In this paper, we propose a new metric learning framework called **kernel density metric learning (KDML)**. It uses kernel density regression to nonlinearly map each attribute to a new feature space. The distance is then defined as the Euclidean distance in the new space. Although we focus on integrating KDML with LMNN in this paper, this nonlinear metric learning framework is general and can be used to support many other metric learning algorithms such as NCA and ITML.

There are several salient advantages of the KDML approach. 1) It embodies excellent nonlinear distance measures with a sound probabilistic explanation. In fact, the Euclidean distance in the mapped feature space corresponds to established distance measures between probability density functions. As a result, such nonlinear mapping allows us to correctly classify datasets that are notoriously difficult to tackle by linear metric learning methods. 2) Compared to kernel-based nonlinear metric learning methods, KDML is easier to use and offers good off-the-shelf usability. In fact, end users do not need to tune any parameter in KDML since we can automatically learn its hyper-parameters using gradient descent. Moreover, KDML can be used as a preprocessing blackbox to map features and be integrated with any supervised metric learning algorithm. Thus, it allows us to leverage the extensive development on efficient and scalable linear metric learning methods. 3) Unlike most existing metric learning methods which require the attributes to be numerical, KDML is the first metric learning algorithm that can naturally handle both numerical, categorical, and mixed attributes in a unified fashion.

This paper contains the following contributions. We introduce KDML, a nonlinear metric learning algorithm, by proposing a novel nonlinear mapping which provides a good similarity measure based on kernel density estimation. It can naturally handle both numerical and categorical features and offers good out-of-the-box usability. Further, we integrate KDML with LMNN, and develop an optimization algorithm to train the model in a holistic way. The algorithm automatically finds optimal bandwidths for a Nadaraya-Watson kernel density estimator, which is absent in previous work. Finally we conduct extensive evaluation on a collection of datasets for multiway classification, with both numerical and categorical features. We show that KDML improves the performance of state-of-the-art metric learning methods for kNN classification tasks.

The rest of this paper is organized as follows. Section 2 gives some preliminaries for metric learning. Section 3 presents the proposed KDML model, including its nonlinear mapping and kernel density estimation. Section 4 combines KDML with LMNN and presents the optimization algorithm for learning the transformation matrix and kernel bandwidths. Section 5 surveys related work on metric learning. Section 6 presents experimental results of different metric learning algorithms on various benchmark datasets. Finally, Section 7 gives conclusions and discusses our future work.

2. PRELIMINARIES

In this paper, we focus on a supervised classification setting. The main ideas can also be extended to other settings such as weakly supervised, semi-supervised, or unsupervised ones.

We assume we are given a training data set

$$\mathcal{T} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\} \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_D \times \mathcal{C}, \quad (2)$$

where there the d^{th} feature is defined in a domain $\mathcal{D}_d$ and the label y_i's are from a set of C classes $\mathcal{C} = \{1, \cdots, C\}$. Note that our setting is more general than typical previous settings, because the domain $\mathcal{D}_i$ can either be a numerical set such as $\mathcal{R}$ or a categorical set.

We use large-margin nearest neighbors (LMNN) [22] as the basic metric learning method to be integrated with KDML. We briefly review the basics of LMNN here.

LMNN is a linear metric learning algorithm that is tailored for kNN classification. For each new input $\mathbf{x}$, kNN classifies $\mathbf{x}$ by a majority vote from the k neighbors that are closest to $\mathbf{x}$ under a certain distance metric. Therefore, kNN classification relies heavily on the distance metric and provides a most natural paradigm for evaluating various distance metric learning algorithms.

LMNN uses the linear transformation in (1). Equivalently, it learns a Mahalanobis distance:

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j), \quad (3)$$

where $\mathbf{M} = \mathbf{L}^T \mathbf{L}$ is a positive semi-definite matrix.

In LMNN, for each input $(\mathbf{x}_i, y_i)$, it specifies a number of *target neighbors* with the same label as y_i. Normally these m target neighbors are simply the m neighbors with the same label that are closest to $\mathbf{x}_i$ based on the Euclidean distance. We use $j \rightsquigarrow i$ to denote that $\mathbf{x}_j$ is a target neighbor of $\mathbf{x}_i$, and $y_{ij} \in \{0, 1\}$ to denote whether the labels y_j and y_i match ($y_{ij} = 1$ when $y_i = y_j$).

The objective function of LMNN is to minimize

$$\begin{aligned} E(\mathbf{M}) \quad = \quad & (1-\mu) \sum_{i, j \rightsquigarrow i} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \\ & \mu \sum_{i, j \rightsquigarrow i, l} (1 - y_{il}) \left[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)\right]_+ \end{aligned} \quad (4)$$

where $[z]_+ = \max(0, z)$ is the standard hinge loss and $\mu \in (0, 1)$ is a positive constant controlling the relative weights of the two terms. The first term minimizes the distance between each input and its target neighbors, and the second term, incorporating the idea of a margin as in SVM, penalizes the distances between those mismatched points that "invade" the neighborhood of each input.

It is shown that the optimization in (4) can be reformulated into a semidefinite program (SDP) [22]. Weinberger et al. have proposed a specialized subgradient descent algorithm to solve this SDP, by exploiting the sparsity of active invaders in the second term of (4). LMNN has received great attention and popularity due to its good kNN classification performance, efficiency, and easiness to use.

Another important work is the information-theoretic metric learning (ITML) [5]. ITML also uses the linear transformation in (3) but utilizes a one to one correspondence between the Mahalanobis distance parameterized by $\mathbf{M}$ and a multivariate Gaussian as

$$P(\mathbf{x}; \mathbf{M}) = \frac{1}{Z} \exp(-\frac{1}{2} d^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_0)), \quad (5)$$

where Z is a normalization factor and $\mathbf{x}_0$ is the mean of the Gaussian. Using this correspondence, the objective of ITML is to minimize

$$KL(p(\mathbf{x}; \mathbf{M}_0) \parallel p(\mathbf{x}, \mathbf{M})) = \int p(\mathbf{x}; \mathbf{M}_0) \ln \frac{p(\mathbf{x}; \mathbf{M}_0)}{p(\mathbf{x}; \mathbf{M})} d\mathbf{x},$$

where $\mathbf{M}_0$ is a fixed matrix such $\mathbf{I}$ or the inverse covariance matrix. The intuition is to regularize $\mathbf{M}$ by minimizing the Kullback-Leibler (KL) divergence [14] between the implied distribution $P(\mathbf{x}; \mathbf{M})$ and a prior distribution.

3. THE KDML FRAMEWORK

In this section, we propose the KDML framework for nonlinear metric learning. We assume that we are given inputs $(\mathbf{x}, y) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_D \times \mathcal{C}$, where the d^{th} feature is defined in a domain $\mathcal{D}_d$ and the label y is from a set of C classes $\mathcal{C} = \{1, \cdots, C\}$.

3.1 KDML feature mapping

Under the KDML framework, we propose two kinds of transformation for each input $(\mathbf{x}, y)$.

Density features. For each dimension $d = 1, \cdots, D$ and any $x_d \in \mathcal{D}_d$, there exists a conditional probability density function

$$P_d(c, x_d) = P(y = c | x_d), \quad c = 1, \cdots, C. \quad (6)$$

We use $P_d(x_d)$ to denote the vector $[P_d(1, x_d), \cdots, P_d(C, x_d)]$.

In this transformation, each input is transformed into a vector

$$\phi_P(\mathbf{x}) = [P_1(x_1); \cdots; P_D(x_D)], \quad (7)$$

which is a concatenation of all the D probability density vectors.

An alternative is to use the square roots of the probabilities.

$$S_d(c, x_d) = \sqrt{P(y = c | x_d)}, \quad c = 1, \cdots, C, \quad (8)$$

which leads to a corresponding feature vector $\phi_S(\mathbf{x})$.

Entropy features. For each dimension $d = 1, \cdots, D$ and any $x_d \in \mathcal{D}_d$, we compute the logarithm of the density

$$E_d(c, x_d) = \ln P(y = c | x_d), \quad c = 1, \cdots, C. \quad (9)$$

Let $E_d(x_d)$ denote the vector $[xE_d(1, x_d), \cdots, E_d(C, x_d)]$.

In this mapping, each input is transformed into a vector

$$\phi_E(\mathbf{x}) = [E_1(x_1); \cdots; E_D(x_D)], \quad (10)$$

which is a concatenation of all the entropy vectors.

In KDML, we choose a feature mapping from ϕ_P, ϕ_S, and ϕ_E and name it ϕ. We may also include the original variables in the feature vector to make it strictly more general than linear mapping. It then employs an existing linear metric learning method to learn a linear transformation $\mathbf{L}\phi(\mathbf{x})$ which gives rise to a Mahalanobis distance in the mapped feature space.

3.2 Implied distance measures

We now discuss the distance measures implied by using the above KDML features. We can see that they all correspond to some sound distance/similarity measures between two probability density functions. As a result, in many cases, the Euclidean distance in the feature space after mapping reflects a better distance measure than the Euclidean distance in the original input space.

One way to view KDML is that it first transforms the original input into a new space. In the new space, before any metric learning, the similarity between two data points are based on the Euclidean distance between their feature vectors:

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)). \quad (11)$$

In fact, since there are D dimensions, each data point corresponds to D probability density functions (PDFs), each with C possible values. That is, for an input $\mathbf{x}_i$, its PDF at the d^{th} dimension is

$$\text{PDF}_{i,d} = [P_d(1, x_{i,d}), \cdots, P_d(C, x_{i,d})], \quad (12)$$

where we use $x_{i,d}$ to denote the d^{th} attribute of $\mathbf{x}_i$. The distance in (11) can be viewed as the summation over the D

dimensions

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{D} \text{diff}(\text{PDF}_{i,d}, \text{PDF}_{j,d}), \quad (13)$$

where diff() measures the distance between two PDFs over the set $\mathcal{C}$.

Distance or similarity measures between PDFs have been extensively studied. A good survey of these measures can be found in [2]. To justify the features in KDML, we examine the underlying distance measures they imply.

When the density feature ϕ_S is used, considering two points $\mathbf{x}_i$ and $\mathbf{x}_j$, their Euclidean distance is

$$\begin{aligned} d^2(\mathbf{x}_i, \mathbf{x}_j) &= (\phi_S(\mathbf{x}_i) - \phi_S(\mathbf{x}_j))^T(\phi_S(\mathbf{x}_i) - \phi_S(\mathbf{x}_j)) \\ &= \sum_{d=1}^{D}\sum_{c=1}^{C}[\sqrt{P_d(c, x_{i,d})} - \sqrt{P_d(c, x_{j,d})}]^2 \end{aligned}$$

Therefore, using ϕ_S features, the implied distance measure between the PDFs is

$$\text{diff}_S(\text{PDF}_{i,d}, \text{PDF}_{j,d}) = \sum_{c=1}^{C}[\sqrt{P_d(c, x_{i,d})} - \sqrt{P_d(c, x_{j,d})}]^2.$$

The above equation is exactly the well-known squared-chord PDF distance measure [8], which is also the square of the Matusita distance measure [15]. Hence, the ϕ_S feature implies the squared-chord and Matusita PDF distance measures.

When the density feature ϕ_P is used, considering two points $\mathbf{x}_i$ and $\mathbf{x}_j$, their Euclidean distance is

$$\begin{aligned} d^2(\mathbf{x}_i, \mathbf{x}_j) &= (\phi_P(\mathbf{x}_i) - \phi_P(\mathbf{x}_j))^T(\phi_P(\mathbf{x}_i) - \phi_P(\mathbf{x}_j)) \\ &= \sum_{d=1}^{D}\sum_{c=1}^{C}[P_d(c, x_{i,d}) - P_d(c, x_{j,d})]^2. \end{aligned} \quad (14)$$

We can see that, using ϕ_P features, the implied distance measure between two PDFs is

$$\text{diff}_P(\text{PDF}_{i,d}, \text{PDF}_{j,d}) = \sum_{c=1}^{C}[P_d(c, x_{i,d}) - P_d(c, x_{j,d})]^2. \quad (15)$$

We can see that (15) is exactly the commonly used squared Euclidean distance measure between two PDFs [2].

Furthermore, the well-known Squared χ^2 PDF distance measure [18] is

$$\text{diff}_{\chi^2}(\text{PDF}_{i,d}, \text{PDF}_{j,d}) = \sum_{c=1}^{C} \frac{[P_d(c, x_{i,d}) - P_d(c, x_{j,d})]^2}{P_d(c, x_{i,d}) + P_d(c, x_{j,d})} \quad (16)$$

Comparing (15) with (16), we can see that diff_{χ^2} can be obtained if we apply a linear transformation to $\phi_P(\mathbf{x}_i)$ and $\phi_P(\mathbf{x}_i)$ (by dividing $\phi_P(d, c)$ by $\sqrt{P_d(c, x_{i,d}) + P_d(c, x_{j,d})}$) and then use the Euclidean distance as the distance measure. In this sense, the metric learning is more general since it learns a linear transformation $\mathbf{M}$, in the entire space of positive semi-definite matrices. Since $\mathbf{M}$ is learned under the guidance of some external objectives, such as optimizing the kNN classification accuracy, we expect it to give better metric for each specific data mining task than the fixed transformation in the squared χ^2 measure.

Finally, when the entropy feature ϕ_E is used, the Euclidean distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is

$$\begin{aligned} d^2(\mathbf{x}_i, \mathbf{x}_j) &= (\phi_E(\mathbf{x}_i) - \phi_E(\mathbf{x}_j))^T(\phi_E(\mathbf{x}_i) - \phi_E(\mathbf{x}_j)) \\ &= \sum_{d=1}^{D}\sum_{c=1}^{C}[\ln P_d(c, x_{i,d}) - \ln P_d(c, x_{j,d})]^2 \\ &= \sum_{d=1}^{D}\sum_{c=1}^{C}\left[\ln \frac{P_d(c, x_{i,d})}{P_d(c, x_{j,d})}\right]^2 \end{aligned} \quad (17)$$

Using ϕ_E features, the implied distance measure between the PDFs is

$$\text{diff}_E(\text{PDF}_{i,d}, \text{PDF}_{j,d}) = \sum_{c=1}^{C}\left[\ln \frac{P_d(c, x_{i,d})}{P_d(c, x_{j,d})}\right]^2, \quad (18)$$

which is not a known PDF distance measure to our knowledge but embodies, under a linear transformation that can be reflected in the Mahalanobis matrix $\mathbf{M}$, the following squared variant of KL divergence [14]

$$\text{diff}_{KL^2}(\text{PDF}_{i,d}, \text{PDF}_{j,d}) = \sum_{c=1}^{C} P_d(c, x_{i,d})\left[\ln \frac{P_d(c, x_{i,d})}{P_d(c, x_{j,d})}\right]^2$$

In summary, the proposed features correspond to some sound distance measures between two PDFs. We believe that they usually give a more reasonable distance measure than the original Euclidean distance. Performing metric learning on these transformed features may allow us to improve many learning algorithms.

3.3 Kernel density estimation for computing features

We have proposed the feature mappings ϕ_P, ϕ_S, and ϕ_E for KDML. Now we estimate the conditional probability densities in these features. From (6), (8) and (9), all of them require estimating $P(y = c|x_d)$, for each $x_d \in \mathcal{D}_d$ and $c = 1, \cdots, C$. Once we have all the $P(y = c|x_d)$, those features can be computed.

Given training data $\mathcal{T} = \{\mathbf{x}_i, y_i\}$, $i = 1, \cdots, N$, We partition $\mathcal{T}$ into C subsets $\mathcal{T}_1, \cdots, \mathcal{T}_C$, which contain data points with labels $y = 1, \cdots, y = C$, respectively.

To estimate $p(y = c|x_d)$, we distinguish the cases of categorical and numerical attributes. We use $\hat{p}(y = c|x_d)$ to denote the estimates.

Categorical attributes. If an attribute x_d takes categorical values, $p(y = c|x_d)$ can be estimated by the proportion of samples with $y = c$ among all the samples whose d^{th} attribute is x_d. Thus, it can be computed using:

$$\hat{p}(y = c|x_d) = \frac{|\mathcal{T}_k \bigcap T_{x_d}|}{|T_{x_d}|}, c = 1, \cdots, C \quad (19)$$

where $\mathcal{T}_{x_d} = \{\mathbf{x}_i \mid x_{i,d} = x_d, i = 1, \cdots, N\}$ is the set of samples in $\mathcal{T}$ whose d^{th} attribute is x_d.

Numerical attributes. If an attribute x_d takes numerical values, we propose to use a Nadaraya-Watson type kernel density regression to estimate $p(y = k|x_d), k = 0, 1$.

According to the Nadaraya-Watson estimator [1, 16], we have:

$$\hat{p}(y = c|x_d) = \frac{\sum_{i \in \mathcal{T}_c} K(\frac{x_d - x_{i,d}}{h_d})}{\sum_{i=1}^{N} K(\frac{x_d - x_{i,d}}{h_d})} \quad (20)$$

where $K(x)$ is a kernel function satisfying $K(x) \geq 0$ and $\int K(x)dx = 1$, and $h_d > 0$ is a parameter called the *bandwidth* of the kernel density function. In this paper, we choose the Gaussian kernel for $K(x)$, namely,

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}). \quad (21)$$

We can thus compute the KDML features by substituting the estimates in (19) and (20) into (6), (8), and (9). For example, the entropy features for categorical and numerical attributes are, respectively,

$$E_d(c, x_d) = \ln \left[\frac{|\mathcal{T}_c \bigcap \mathcal{T}_{x_d}|}{|\mathcal{T}_{x_d}|} \right], \quad (22)$$

and,

$$E_d(c, x_d) = \ln \left[\frac{\sum_{i \in \mathcal{T}_c} \exp(-\frac{(x_d - x_{i,d})^2}{2h_d^2})}{\sum_{i \in \mathcal{T}} \exp(-\frac{(x_d - x_{i,d})^2}{2h_d^2})} \right]. \quad (23)$$

We also comment on the difference between the assumptions of ITML and KDML. The main assumption of ITML is that all the data points are drawn from a single Gaussian distribution, centered at $\mathbf{x}_0$. Such an assumption may be too restrictive in some cases. KDML, in contrast, assumes a nonlinear distribution which is a mixture of multiple Gaussians at each dimension.

4. COMBINING KDML WITH LMNN

In principle, KDML is a general framework that can be combined with existing metric learning algorithms as a preprocessing step, which nonlinearly maps the features in the original space to a new space.

4.1 The KDML-LMNN approach

As a concrete application, we combine KDML with the LMNN algorithm and apply it to kNN classification. First, we map each training data $\mathbf{x}$ into a feature $\phi(\mathbf{x})$ (which may be $\phi_P(\mathbf{x})$, $\phi_S(\mathbf{x})$, or $\phi_E(\mathbf{x})$). Then, we use LMNN to learn a transformation $\mathbf{L}\phi(\mathbf{x})$ which leads to a Mahalanobis distance

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T \mathbf{M} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)), \quad (24)$$

where $\mathbf{M} = \mathbf{L}^T\mathbf{L}$ is a positive semi-definite matrix.

Applying LMNN to $\phi(\mathbf{x})$, we solve the problem of minimizing:

$$\begin{aligned} E(\mathbf{M}) &= (1-\mu) \sum_{i,j \rightsquigarrow i} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \\ &\quad \mu \sum_{i,j \rightsquigarrow i,l} (1 - y_{il}) \left[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) \right]_+, \end{aligned} \quad (25)$$

where $d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$ is defined in (24).

As a side note, we can also substitute $d^2_{\mathbf{M}}$ in (24) into (5) so that KDML is combined with ITML.

4.2 Optimization algorithm

The training of KDML-LMNN aims at learning the optimal values of the matrix $\mathbf{M}$ and the bandwidths in the Nadaraya-Watson estimator. For each numerical attributes x_d, there is a hyper-parameter h_d that needs to be chosen. One way to choose h_d is to use rules-of-thumb to set a heuristic h_d values. A popular one is the Silverman's rule of thumb [20]:

$$h_d^* = 1.06 \sigma N^{-1/5}, \quad (26)$$

where σ is the standard deviation of x_d.

Although such rules-of-thumb often give solid performance, we can in fact derive a novel way to automatically choose optimal h_d based on the KDML-LMNN objective. Such automatic tuning is absent in previous work. We propose to find the h_d that minimizes E in (25). For this minimization, a nice fact is that we can get the closed form of the subgradient $\frac{\partial E}{\partial h_d}$ and compute it efficiently.

There are two terms in (25). Let

$$E_1 = \sum_{i,j \rightsquigarrow i} D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j), \quad \text{and} \quad (27)$$

$$E_2 = \sum_{i,j \rightsquigarrow i,l} (1 - y_{il}) \left[1 + D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) \right]_+ \quad (28)$$

We have

$$E(\mathbf{M}) = (1-\mu)E_1 + \mu E_2 \quad (29)$$

We compute the gradients for these two terms separately. First, since

$$\frac{\partial D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} = \frac{\partial D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} = \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) \quad (30)$$

we have

$$\frac{\partial E_1}{\partial \mathbf{x}_k} = \sum_{j \rightsquigarrow k} \mathbf{M}(\mathbf{x}_k - \mathbf{x}_j) + \sum_{k \neq j, k \rightsquigarrow j} \mathbf{M}(\mathbf{x}_k - \mathbf{x}_j). \quad (31)$$

For E_2, note that $\frac{\partial E_2}{\partial \mathbf{x}_k}$ is a subgradient since it involves a hinge loss and it is non-differentiable whenever the term inside $[.]_+$ is zero. Therefore: when

$$[1 + D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)] < 0, \quad (32)$$

$\frac{\partial E_2}{\partial \mathbf{x}_k}$ is 0; otherwise, we have:

$$\begin{aligned} \frac{\partial E_2}{\partial \mathbf{x}_k} &= \sum_{k, j \rightsquigarrow k} \sum_l (1 - y_{kl}) \mathbf{M}(\mathbf{x}_l - \mathbf{x}_j) \\ &\quad + \sum_{i \neq k, k \rightsquigarrow i, l} (1 - y_{il}) \mathbf{M}(\mathbf{x}_k - \mathbf{x}_i) \\ &\quad - \sum_{i, k \rightsquigarrow i} (1 - y_{ik}) \mathbf{M}(\mathbf{x}_k - \mathbf{x}_i) \end{aligned} \quad (33)$$

Then, according to (29), for $d = 1, \cdots, D$, we have

$$\frac{\partial E}{\partial h_d} = \sum_k \left[(1-\mu) \frac{\partial E_1}{\partial \mathbf{x}_k} + \mu \frac{\partial E_2}{\partial \mathbf{x}_k} \right]^T \frac{\partial \mathbf{x}_k}{\partial h_d} \quad (34)$$

where $\frac{\partial \mathbf{x}_k}{\partial h_d} = \left[0, \cdots, \frac{\partial \phi_{c,d}(\mathbf{x}_k)}{\partial \mathbf{x}_k}, \cdots, 0 \right]^T$ is a column vector, where $\phi_{c,d}(\mathbf{x}_k)$ is the KDML feature value for the d^{th} dimension and class c of $\mathbf{x}_k$. Use the entropy feature in (9) as an example, we know

$$\phi_{c,d}(\mathbf{x}_k) = \ln \left[\frac{\sum_{i \in \mathcal{T}_c} \exp(-\frac{(x_{k,d} - x_{i,d})^2}{2h_d^2})}{\sum_{i \in \mathcal{T}} \exp(-\frac{(x_{k,d} - x_{i,d})^2}{2h_d^2})} \right]. \quad (35)$$

Algorithm 1 Optimization for KDML-LMNN learning

```
Initialize h using (26)
repeat
    compute the feature matrix Φ_h
    call LMNN to optimize M        ▷ under fixed h & Φ_h
    for d = 1 to D do              ▷ under fixed M
        if x_d is a numerical variable then
            h_d ← h_d − γ ∂E/∂h_d   ▷ gradient descent
        end if
    end for
until h converges
output h and M
```

We now compute $\frac{\partial \phi_{c,d}(\mathbf{x}_k)}{\partial h_d}$. Let $r_d = -1/(2h_d^2)$, we have:

$$\frac{\partial \phi_{c,d}(\mathbf{x}_k)}{\partial r_d} = \frac{\sum_{i\in\mathcal{T}_c}[(x_{k,d}-x_{i,d})^2 \cdot \exp(r_d(x_{k,d}-x_{i,d})^2)]}{\sum_{i\in\mathcal{T}_c}\exp(r_d(x_{k,d}-x_{i,d})^2)} - \frac{\sum_{i\in\mathcal{T}}[(x_{k,d}-x_{i,d})^2 \cdot \exp(r_d(x_{k,d}-x_{i,d})^2)]}{\sum_{i\in\mathcal{T}}\exp(r_d(x_{k,d}-x_{i,d})^2)} \tag{36}$$

and

$$\frac{\partial \phi_{c,d}(\mathbf{x}_k)}{\partial h_d} = \frac{1}{h_d^3}\frac{\partial \phi_{c,d}(\mathbf{x}_k)}{\partial r_d}. \tag{37}$$

Summarizing things together, we can get the closed form of $\frac{\partial E}{\partial h_d}$ by assembling (34), (31), (33), (36), and (37). The closed form of $\frac{\partial E}{\partial h_d}$ seems complex but in fact can be efficiently computed. $\frac{\partial E}{\partial h_d}$ has two parts, $\frac{\partial E_1}{\partial h_d}$ and $\frac{\partial E_2}{\partial h_d}$. $\frac{\partial E_1}{\partial h_d}$ has a simple form and only involves pairs of neighboring points satisfying $j \rightsquigarrow k$ or $k \rightsquigarrow j$.

For $\frac{\partial E_2}{\partial h_d}$, it is important to note that it is a subgradient. For E_2 in (28), for each point i, we only need to consider those "active" l that are invading the neighborhood of i so that the corresponding $[.]_+$ term is positive. There are typically few invaders. This is observed and exploited in LMNN to speed up its gradient computation [23]. In [23], it is found that the k target neighbors and the invaders do not change frequently over each iteration. The LMNN package maintains such information in a data structure for efficient updates during the optimization process. This data structure is adapted in our implementation to support efficient computation of $\frac{\partial E}{\partial h_d}$.

Let $\mathbf{h}$ to the vector of all those h_d for numerical attributes $x_d, d = 1, \cdots, D$. We show our optimization algorithm for training KDML+LMNN in Algorithm 1. It contains two levels of optimization: an outer loop which optimizes $\mathbf{h}$ using subgradient descent, and an inner loop which learns $\mathbf{M}$ using the original LMNN package under fixed $\mathbf{h}$.

In the outer level, at each iteration, the feature matrix $\mathbf{\Phi_h}$ composed of $\phi_{c,d}(\mathbf{x}_k)$ is updated based on the new $\mathbf{h}$. If a variable $\mathbf{x}_i$ is categorical, its feature is computed by (19). For a numerical variable x_d, we use the kernel density estimation in (20) to compute its feature. Then, entering the inner level, we use the original LMNN package to learn the $\mathbf{M}$ that minimizes $E(\mathbf{M})$ in (25) under fixed $\mathbf{h}$ and $\mathbf{\Phi_h}$. Finally, we optimize $\mathbf{h}$ for numerical attributes by performing descent along the subgradient direction in (34) based on the validation set. We use a line search with the Armijo rule to choose the step size γ.

As Algorithm 1 learns $\mathbf{M}$ and $\mathbf{h}$, there are no other hyper-parameters to tune for KDML-LMNN.

5. RELATED WORK

A number of prior works on metric learning have focused on learning a linear transformation in the original input space [5, 9, 11, 19, 23, 24]. They achieved great success in improving the performance of learning algorithms by obtaining better Mahalanobis distance measures. The concept of distance metric learning was first proposed by Xing et al. [24]. Their objective is to learn a Mahalanobis matrix such that similar points are clustered together subject to the constraints that distances between dissimilar points are larger than a lower bound. Inspired by this general idea, many works have been developed.

LMNN [23] identifies the local target neighbors in the original space for each point and learns a Mahalanobis matrix such that the non-target neighbors for each point are encouraged to be far away from all its target neighbors with a large margin. ITML [5] assumes that there exists a bijection between the Mahalanobis distance and a single multivariate Gaussian distribution. It minimizes the KL divergence between a prior distribution and the distribution implies by the Mahalanobis distance, subject to upper bound constraints on the distance between similar points and lower bound constraints on the distance between dissimilar points. The SEPAPH [17] approach also relies on a mapping from the Mahalanobis distance to a probability distribution but extends to semi-supervised metric learning based on regularization.

Neighborhood components analysis (NCA) [11] maximizes a softmax function that smooths the leave-one-out accuracy of kNN classification. However, it has a nonconvex objective function and suffers from local minima. Maximally collapsing metric learning (MCML) [9] constructs a convex objective based on the same softmax function to characterize the distribution. It minimizes for each point the KL divergence between a "bi-level" distribution and the desired distribution under the Mahalanobis distance, where the bi-level distribution is zero for similar points and non-zero for dissimilar points.

All the above methods look for a linear transformation. However, the linearly transformed features fail to have satisfactory expression on many cases, such as the example in Figure 1. Another example is the case when the two classes of data form two concentric circles [23], which we will illustrate in Section 6. Hence, there are also work on nonlinear distance metric learning.

One nonlinear extension is to kernelize existing methods and use the Representer's Theorem to represent the nonlinear transformation using the kernel matrix elements [3, 10, 21]. However, these methods have not replicated the success of linear methods and out-of-the-box packages based on such kernelization are lacking. In general, direct kernel methods are sensitive to hyper-parameters and their utility is limited inherently by the sizes of kernel matrices [13].

Another nonlinear approach, MM-LMNN [23], uses multiple metrics for different clusters of data to achieve global nonlinearity, where the clusters are obtained by the k-means algorithm. However, the transformation is locally linear with respect to each cluster and the cross-cluster distances cannot be easily learned. Two other nonlinear methods are proposed in [13]. χ^2-LMNN uses a nonlinear χ^2 distance

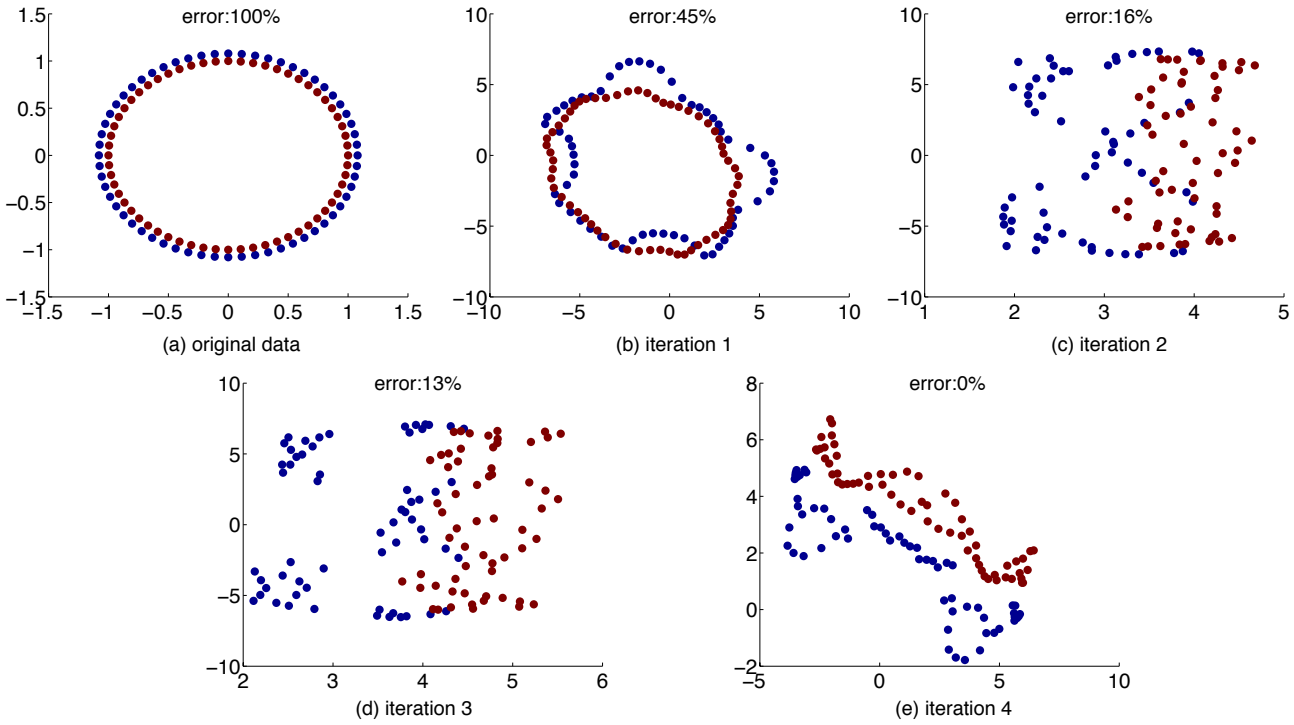

Figure 2: A toy example with two circles in two classes, marked in different colors. a) shows the original data; b) - e) show the data mapping and kNN classification error after each outer-loop iteration of Algorithm 1 which tunes h. The classification error quickly decreases to zero as h is optimized using subgradient descent.

measure. It is intended for histogram data and can only be applied when all the data lie on a simplex $S^D = \{\mathbf{x} \in \mathcal{R}^D | \mathbf{x} \geq 0, \mathbf{x}^T\mathbf{1} = 1\}$. KDML has much wider applicability than χ^2-LMNN since it can process any input, into histograms for categorical data and probability densities for numerical data. GB-LMNN uses a set of gradient boosting regression trees with different heights and optimizes the objective function of LMNN. GB-LMNN is shown to perform better than its linear counterpart LMNN and MM-LMNN [13].

6. EXPERIMENTAL RESULTS

We conduct extensive experiments to evaluate the KDML-LMNN approach in Algorithm 1, or KDML for short in this section. We evaluate KDML with ϕ_P, ϕ_S, or ϕ_E features (denoted as KDML$_P$, KDML$_S$, and KDML$_E$, respectively). Note that KDML can be applied to datasets with numerical, categorical, and mixed attributes.

For comparison, we also evaluate two state-of-the-art linear metric learning algorithms including LMNN [22] and ITML [5]. We also evaluate a nonlinear metric learning algorithm MM-LMNN [23], which first groups data into clusters and then uses multiple linear mappings for different clusters to achieve globally nonlinear mapping. Both LMNN and MM-LMNN are obtained from their website[1]. ITML code is obtained from http://www.cs.utexas.edu/∼pjain/itml/. We also evaluate using the original Euclidean distance as a baseline.

We implemented the KDML algorithm inside the LMNN package, which is implemented in Matlab. For full replicability of the experiments, our KDML code is made available online at http://www.cse.wustl.edu/∼chen/kdml/. All experiments are performed on a desktop computer with 2.67GHz CPU and 8G memory running Mac OS X 10.7.

6.1 Illustrations on toy cases

For sanity check and illustration, we first test on a simple example in Figure 1a). This data cannot be correctly separated by any linear metric learning algorithm. Since KDML maps the data to a higher dimensional space, to visualize the mapping in 2-D, we extract a 2-D transformation $\mathbf{L} \in \mathcal{R}^{2\times D}$ from $\mathbf{M}$ using eigendecomposition. Such dimensionality reduction is in fact another main utility of metric learning and already implemented in LMNN. Figure 1b) shows the 2-D mapping result by KDML, which clearly separates the two classes. Figures 1c) and 1d) show that linear transformations cannot separate the two classes.

We also test another toy example shown in Figure 2a). It contains two concentric circles of data from two different classes. It is a very difficult case since the nearest neighbor of any given data point is from the other class. It is a well-known example as no linear transformation can separate these two classes [13].

Figures 2b) to 2e) illustrate the process of KDML-LMNN in Algorithm 1 which automatically tunes the kernel bandwidth $\mathbf{h}$. For better visualization, the results in Figures 2b) to e) are obtained by applying Algorithm 1 and extracting a 2-D mapping using eigendecomposition of $\mathbf{M}$ at each outer-loop iteration. We can see that the kNN classification error quickly decreases from 45% after the initial KDML

[1]http://www.cse.wustl.edu/∼kilian/code/lmnn/lmnn.html

Type	Dataset	N	C	D_n	D_c
Numerical	Glass	214	7	10	0
	Wine	178	3	13	0
Mixed	Contraceptive	1473	3	2	7
	Statlog Heart	270	2	6	7
Categorical	Hayes-Roth	160	3	0	5
	Balance Scale	625	3	0	4
	Car	1728	4	0	6

Table 1: The number of instances N, number of classes C, number of numerical features D_n, and number of categorical features D_c of the tested UCI datasets.

mapping to 0% in just four major iterations of optimizing $\mathbf{h}$ using subgradient descent.

6.2 Results on numerical datasets

We test all the algorithms on benchmark datasets from the UCI repository [6]. We choose datasets mostly with multiple (≥ 3) classes since kNN has salient advantages over other methods such as SVM on multiway classification. For each dataset, we run a 10-fold cross validation with 90/10 splits and report the average results. We use $k = 3$ for kNN classification on all the cases.

Table 1 lists the main characteristics of the tested datasets. We can see that there are datasets with numerical, categorical, and mixed attributes.

Table 2 compares kNN classification errors of various algorithms on the numerical datasets. We observe that all KDML algorithms, with three different kinds of feature mappings, consistently perform significantly better than other algorithms in most cases.

6.3 Results on categorical and mixed datasets

Another major advantage of KDML is its ability to naturally handle categorical variables. We also evaluate our algorithms on datasets with categorical attributes and mixed data types from the UCI repository.

To deal with a categorical attribute x, KDML transforms x into numerical features defined as $\phi_P(x)$, $\phi_S(x)$ or $\phi_E(x)$ before. For other algorithms, we use a typical multinomial encoding to handle categorical variables. For each categorical attribute x that has m different categories, we transform it into m numbers with only one of the numbers being 1 and the others being 0.

Table 2 also lists the kNN classification results on datasets with categorical attributes. We observe that all KDML algorithms, with different features, again consistently perform much better than other algorithms on all the cases. KDML$_E$ is the overall winner with the best performance on all the categorical and mixed datasets, except for the Statlog Heart dataset where it is only slightly (well within one standard error) worse than ITML.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a novel kernel density metric learning (KDML) framework for nonlinear distance metric learning. KDML is fundamentally different from the previous metric learning algorithms since it introduces a nonlinear mapping from the original input space into a probability density space, based on Nadaraya-Watson kernel density estimation. We have shown that the nonlinear mapping in KDML embodies established distance measures between probability density functions, and leads to correct classification on datasets on which linear metric learning methods would fail. KDML can be used as a preprocessing step and combined with existing metric learning algorithms. We have integrated KDML with the LMNN algorithm. Under this framework, we have derived the closed form of the subgradients of the objective function with respect to the kernel bandwidths. We have then derived an integrated optimization algorithm for learning the Mahalanobis matrix and kernel bandwidths. Extensive results on real-world numerical and categorical data show that, KDML gives significantly better kNN classification quality than other linear and nonlinear metric learning algorithms. Unlike previous metric learning algorithms, KDML can naturally handle both numerical and categorical data. It is also easy to use and offers good off-the-shelf usability. These advantages make KDML an attractive general approach for metric learning.

Our ongoing work is focused on combining the nonlinear features in KDML with more expressive parametric forms of the distance function such as that in χ^2-LMNN and KL-divergence, instead of the simple Euclidean ℓ_2 form. The flexibility in both feature mappings and distance functions may enable us to construct superior distance/similarity measures for a wide range of applications.

Acknowledgment

This work is partially supported by the CNS-1017701 and CCF-1215302 grants from the National Science Foundation of the United States, a Microsoft Research New Faculty Fellowship, and a Barnes-Jewish Hospital Foundation grant.

8. REFERENCES

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] S. H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1:300–307, 2007.

[3] R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, and B. Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomput.*, 73(10-12):1570–1579, June 2010.

[4] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 539–546, Washington, DC, USA, 2005. IEEE Computer Society.

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 209–216, New York, NY, USA, 2007. ACM.

[6] A. Frank and A. Asuncion. UCI machine learning repository, http://archive.ics.uci.edu/ml, 2010.

Data set	Euclidean	ITML	LMNN	MM-LMNN	KDML$_P$	KDML$_S$	KDML$_E$
Glass	32.25± 5.08	32.75 ± 10.17	30.39± 4.47	29.48±8.69	**22.45± 4.93**	26.16± 7.56	28.03± 6.32
Wine	28.63 ± 3.33	6.14 ± 4.94	3.33± 3.04	3.92±3.15	4.51±3.20	3.95±3.21	**2.84±2.86**
Contraceptive	51.19±3.03	51.33 ± 3.39	50.65±1.81	52.41±3.66	**50.37±2.27**	50.44±2.53	**50.37±3.34**
Statlog Heart	38.15±7.81	**22.22±8.38**	24.44±4.22	26.30±6.85	25.56±1.55	26.67±4.46	22.96±3.84
Hayes-Roth	32.50±3.56	19.38±6.01	21.88±7.97	18.13±4.07	18.75±3.13	18.13±2.61	**17.50±1.71**
Balance Scale	28.80±2.99	18.40±4.42	21.76±3.94	12.64±10.33	17.12±2.44	14.56±2.22	**10.88±2.81**
Car	15.92±2.17	13.96±6.52	3.12±1.25	3.30±1.04	3.13±1.24	3.30±0.79	**3.07±0.93**

Table 2: KNN classification error (in %, ± standard deviation) of various methods on the UCI datasets, averaged over 10-fold 90/10 training-testing splits. Best results are shown in bold.

[7] C. Galleguillos, B. McFee, S. J. Belongie, and G. R. G. Lanckriet. Multi-class object localization by combining local contextual interactions. In *CVPR*, pages 113–120. IEEE, 2010.

[8] D. Gavin, W. Osward, E. Wahl, and J. Williams. A statistical approach to evaluating distance metrics and analog assignments for pollen records. 60:356–367, 2003.

[9] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Proc. NIPS*, 2005.

[10] A. Globerson and S. T. Roweis. Visualizing pairwise similarity via semidefinite programming. *Journal of Machine Learning Research - Proceedings Track*, 2:139–146, 2007.

[11] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Proc. NIPS*, pages 513–520, 2004.

[12] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 2012.

[13] D. Kedem, S. Tyree, K. Weinberger, F. Sha, and G. Lanckriet. Non-linear metric learning. In *Proc. NIPS*, 2012.

[14] S. Kullback and R. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22:79–86, 1951.

[15] K. Matusita. Decision rules, based on the distance, for problems of fit, two samples, and estimation. *Ann. Math. Statist.*, 26:631–640, 1955.

[16] E. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–142, 1964.

[17] G. Niu, B. Dai, M. Yamada, and M. Sugiyama. Information-theoretic semi-supervised metric learning via entropy regularization. In *Proceedings of the 29th international conference on Machine learning*, ICML '12, 2012.

[18] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Phil. Mag.*, 50:157–172, 1900.

[19] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 776–792, London, UK, UK, 2002. Springer-Verlag.

[20] B. W. Silverman and P. J. Green. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[21] L. Torresani and K. chih Lee. Large margin component analysis. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1385–1392. MIT Press, Cambridge, MA, 2007.

[22] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proc. NIPS*, 2005.

[23] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[24] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *Proc. NIPS*, pages 505–512, 2002.