A Scalable Kernel-Based Semi-Supervised Metric Learning Algorithm with Out-of-Sample Generalization Ability

Dit-Yan Yeung[†], Hong Chang[‡], Guang Dai[†]

[†] Department of Computer Science and Engineering

Hong Kong University of Science and Technology

Clear Water Bay, Hong Kong, China

[‡] Chinese Academy of Sciences

China

January 15, 2008

To appear in *Neural Computation*

Abstract

In recent years, metric learning in the semi-supervised setting has aroused a lot of research interests. One type of semi-supervised metric learning utilizes supervisory information in the form of pairwise similarity or dissimilarity constraints. However, most methods proposed so far are either limited to linear metric learning or unable to scale up well with the data set size. In this paper, we propose a nonlinear metric learning method based on the kernel approach. By applying low-rank approximation to the kernel matrix, our method can handle significantly larger data sets. Moreover, our low-rank approximation scheme can naturally lead to out-of-sample generalization. Experiments performed on both artificial and real-world data show very promising results.

Keywords: metric learning, semi-supervised learning, kernel learning, low-rank approximation, out-of-sample extension.

1 Introduction

1.1 Semi-Supervised Learning

In supervised learning, we are given a training sample in the form of input-output pairs. The learning task is to find a functional relationship that maps any input to an output such that disagreement with future input-output observations is minimized. Classification and regression problems

are the most common supervised learning problems for discrete-valued and continuous-valued outputs, respectively. In unsupervised learning, we are given a training sample of objects with no output values, with the aim of extracting some structure from them to obtain a concise representation or to gain some understanding of the process that generated the data. Clustering, density estimation, and novelty detection problems are common unsupervised learning problems.

Over the past decade or so, there has been growing interest in exploring new learning problems between the supervised and unsupervised learning extremes. These methods are generally referred to as *semi-supervised learning* methods, although there exist large variations in both the problem formulation and the approach to solve the problem. The semi-supervised learning literature is too large to do a comprehensive review here. Interested readers are referred to some good surveys, e.g., (Seeger, 2000; Grira, Crucianu, & Boujemaa, 2005; Zhu, 2006).

One way to categorize many, though not all, semi-supervised learning methods is to consider the type of supervisory information available for learning. Unlike unsupervised learning tasks, supervisory information is available in semi-supervised learning tasks. However, the information is in a form that is weaker than that available in typical supervised learning tasks. One type of (weak) supervisory information assumes that only part (usually a limited part) of the training data are labeled. This scenario is commonly encountered in many real-world applications. One example is the automatic classification of web pages into semantic categories. Since labeling web pages

is very labor-intensive and hence costly, unlabeled web pages are far more plentiful on the web. It would be desirable if a classification algorithm can take advantage of the unlabeled data in increasing the classification accuracy. Many *semi-supervised classification* methods (Zhu, 2006) belong to this category.

Another type of supervisory information is even weaker in that it only assumes the existence of some pairwise constraints indicating similarity or dissimilarity relationships between training examples. In video indexing applications, for example, temporal continuity in the data can be naturally used to impose pairwise similarity constraints between successive frames in video sequences. Another example is in proteomic analysis, where protein-protein interactions can naturally be represented as pairwise constraints for the study of proteins encoded by the genes (e.g., *Database of Interacting Proteins* (DIP), `http://dip.doe-mbi.ucla.edu/`). Yet another example is the anti-spam problem. Recent studies show that more than half of the e-mail in the Internet today is spam, or unsolicited commercial e-mail. The relentless rise of spam is posing a significant problem to users and providers of e-mail services. One recent approach to spam detection is based on the trustworthiness of social networks (Boykin & Roychowdhury, 2005). Such social networks can naturally be represented as graphs with edges between nodes representing pairwise relationships in the social networks.

While supervisory information in the form of limited labeled data can be transformed into pairwise similarity and dissimilarity constraints, inverse transformation is in general not possible except for the special case of two-

class problems. In this sense, the second type of supervisory information is of a weaker form and hence the corresponding learning problem is more difficult to solve. The focus of our paper is on this category of semi-supervised learning problems.

1.2 Semi-Supervised Metric Learning Based on Pairwise Constraints

While many semi-supervised learning methods assume the existence of limited labeled data (Zhu, 2006), there are much fewer methods that can work with pairwise constraints only. We survey the most representative methods in this subsection.

Very often, the pairwise constraints simply state whether two examples belong to the same class or different classes.[1] (Wagstaff & Cardie, 2000) and (Wagstaff, Cardie, Rogers, & Schroedl, 2001) first used such pairwise information for semi-supervised clustering tasks by modifying the standard k-means clustering algorithm to take into account the pairwise similarity and dissimilarity constraints. Extensions have also been made to model-based clustering based on the expectation-maximization (EM) algorithm for Gaussian mixture models (Shental, Bar-Hillel, Hertz, & Weinshall, 2004; Lu & Leen, 2005). However, these methods do not explicitly learn a distance func-

[1] In principle, it is also possible to incorporate pairwise constraints that quantify the degree of similarity or dissimilarity as well to provide more informative knowledge for learning. Such pairwise constraints may be regarded as constituting part of the dissimilarity matrix in multidimensional scaling (MDS).

tion but seek to satisfy the constraints, typically for clustering tasks. Hence, they are sometimes referred to as constraint-based clustering methods.

As a different category, some methods have been proposed to learn a Mahalanobis metric or some other distance function based on pairwise constraints. (Xing, Ng, Jordan, & Russell, 2003) formulated a convex optimization problem with inequality constraints to learn a Mahalanobis metric and demonstrated performance improvement in a subsequent clustering task. Solving a similar problem to learn a Mahalanobis metric, the relevant component analysis (RCA) algorithm (Bar-Hillel, Hertz, Shental, & Weinshall, 2003, 2005) was proposed as a simpler and more efficient algorithm than that of (Xing et al., 2003). However, RCA can make use of similarity constraints only. (Schultz & Joachims, 2004) proposed a method for learning essentially the same metric, but they made use of a different type of pairwise information which compares the pairwise relationships between two pairs of examples. Specifically, each relational constraint states that example A is closer to B than A is to C. (Hertz, Bar-Hillel, & Weinshall, 2004) proposed a distance function learning method called DistBoost. However, there is no guarantee that the distance function learned is a metric. (Bilenko, Basu, & Mooney, 2004) explored the possibility of integrating constraint-based clustering and semi-supervised clustering based on distance function learning. The distance function learning methods reviewed above either learn a Mahalanobis metric that corresponds to linear transformation only, or learn a distance function that is not a metric. In our previous work (Chang & Yeung, 2004), we proposed a metric learning method that corresponds to nonlinear

transformation. While this method is more powerful than the linear methods, the optimization problem is non-convex and hence is more complicated to solve.

In this paper, we focus on the distance function learning approach because distance functions are central to many learning models and algorithms. This also makes it easier to achieve out-of-sample generalization. Moreover, we focus on learning metrics because this allows us to formulate the metric learning problem based on the kernel approach (Vapnik, 1998; Schölkopf & Smola, 2002), which provides a disciplined, computationally appealing approach to nonlinear metric learning.

1.3 Organization of Paper

In Section 2, we propose a simple and efficient kernel-based metric learning method based on pairwise similarity constraints. Like many kernel methods, a limitation of this method is that it does not scale up well with the sample size. In Section 3, we address the scalability issue by applying low-rank approximation to the kernel matrix. This extended method can also naturally give rise to out-of-sample generalization, which is addressed in Section 4. We present some experimental results in Section 5 to demonstrate the effectiveness of our metric learning algorithm. Finally, Section 6 concludes the paper.

2 Kernel-Based Metric Learning

2.1 Problem Setup

Let $\{\mathbf{x}_i\}_{i=1}^n$ be a set of n data points in some input space $\mathcal{X}$. Suppose we have a Mercer kernel $\hat{k}$ which induces a nonlinear feature map $\hat{\phi}$ from $\mathcal{X}$ to some reproducing kernel Hilbert space $\mathcal{H}$ (Vapnik, 1998; Schölkopf & Smola, 2002). The corresponding set of feature vectors in $\mathcal{H}$ is $\{\hat{\phi}(\mathbf{x}_i)\}_{i=1}^n$ and the kernel matrix is $\hat{\mathbf{K}} = [\hat{k}(\mathbf{x}_i, \mathbf{x}_j)]_{n\times n} = [\langle\hat{\phi}(\mathbf{x}_i), \hat{\phi}(\mathbf{x}_j)\rangle]_{n\times n}$. Choices for $\hat{k}$ include the Gaussian RBF kernel and the polynomial kernel. We apply a centering transform such that the feature vectors in $\mathcal{H}$ have zero mean. The resulting kernel matrix $\mathbf{K}$ can be computed as $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{n\times n} = [\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle]_{n\times n} = \mathbf{H}\hat{\mathbf{K}}\mathbf{H}$, where the centering matrix $\mathbf{H} = \mathbf{I} - (1/n)\mathbf{1}\mathbf{1}^T$ with $\mathbf{I}$ being the $n \times n$ identity matrix and $\mathbf{1}$ being the $n \times 1$ vector of ones.

We consider a type of semi-supervised metric learning in which the supervisory information is given in the form of pairwise similarity constraints.[2] Specifically, we are given a set of point pairs, which is a subset of $\mathcal{X} \times \mathcal{X}$, as $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to the same class}\}$. Our goal is to make use of $\mathcal{S}$ to learn a better metric through modifying the kernel so that the performance of some subsequent task (e.g., clustering, classification) based on the metric is improved after kernel learning.

[2] As we will see later in the formulation of the optimization problem for kernel learning, these constraints are "soft" constraints rather than "hard" constraints in that they are only preferred, not enforced. This makes it easy to handle noisy constraints, i.e., erroneous supervisory information, if we so wish.

2.2 Kernel Learning

Since the kernel matrix $\mathbf{K}$ is symmetric and positive semi-definite, we can express it as

$$\mathbf{K} = \sum_{r=1}^{p} \lambda_r \mathbf{v}_r \mathbf{v}_r^T = \sum_{r=1}^{p} \lambda_r \mathbf{K}_r, \tag{1}$$

where $\lambda_1 \geq \cdots \geq \lambda_p > 0$ are the $p \leq n$ positive eigenvalues of $\mathbf{K}$, $\mathbf{v}_1, \ldots, \mathbf{v}_p$ are the corresponding normalized eigenvectors, and $\mathbf{K}_r = \mathbf{v}_r \mathbf{v}_r^T$.

We consider a restricted form of kernel matrix learning by modifying $\mathbf{K}$ through changing the λ_r's while keeping all $\mathbf{K}_r$'s fixed. This approach has also been used in some semi-supervised kernel learning methods based on labeled data, typically for classification problems, e.g., (Cristianini, Shawe-Taylor, Elisseeff, & Kandola, 2002; Lanckriet, Cristianini, Bartlett, El Ghaoui, & Jordan, 2002; Bousquet & Herrmann, 2003; Tsuda, Akaho, & Asai, 2003; Zhang, Yeung, & Kwok, 2004). Ideally, one would like to have as much flexibility as possible in modifying $\mathbf{K}$. Unfortunately, for many kernel learning settings including ours, the amount of supervisory information in the form of pairwise constraints or labeled data is typically very limited. Allowing too much flexibility in the model while having only limited supervisory information for learning is likely to lead to model overfitting. A common approach to this problem is to regularize the model using model bias. Fixing all $\mathbf{K}_r$'s while allowing the eigenvalues to change can be regarded as a form of regularization by constraining the search space of possible kernel matrices to the eigenspace defined by the eigenvectors $\mathbf{v}_r$'s of $\mathbf{K}$.

To ensure that the eigenvalues are nonnegative, we rewrite $\mathbf{K}$ as $\mathbf{K}_{\boldsymbol{\beta}} =$

$[k_{\boldsymbol{\beta}}(\mathbf{x}_i, \mathbf{x}_j)]_{n\times n} = [\langle \phi_{\boldsymbol{\beta}}(\mathbf{x}_i), \phi_{\boldsymbol{\beta}}(\mathbf{x}_j)\rangle]_{n\times n}$:

$$\mathbf{K}_{\boldsymbol{\beta}} = \sum_{r=1}^{p} \beta_r^2 \mathbf{K}_r, \tag{2}$$

which represents a family of kernel matrices parameterized by $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$.

We perform kernel learning such that the mean squared Euclidean distance induced by $\mathbf{K}_{\boldsymbol{\beta}}$ between feature vectors in $\mathcal{H}$ corresponding to point pairs in $\mathcal{S}$ is reduced. Thus the criterion function for optimization is

$$\begin{aligned} J_{\mathcal{S}}(\boldsymbol{\beta}) &= \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} [(\mathbf{K}_{\boldsymbol{\beta}})_{ii} + (\mathbf{K}_{\boldsymbol{\beta}})_{jj} - 2(\mathbf{K}_{\boldsymbol{\beta}})_{ij}] \\ &= \sum_{r=1}^{p} \beta_r^2 \left[\frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} (\mathbf{b}_i - \mathbf{b}_j)^T \mathbf{K}_r (\mathbf{b}_i - \mathbf{b}_j) \right] \\ &= \boldsymbol{\beta}^T \mathbf{D}_{\mathcal{S}} \boldsymbol{\beta}, \end{aligned} \tag{3}$$

where $\mathbf{b}_i$ is the ith column of the $p \times p$ identity matrix[3] and $\mathbf{D}_{\mathcal{S}}$ is a $p \times p$ diagonal matrix with diagonal entries

$$\begin{aligned} (\mathbf{D}_{\mathcal{S}})_{rr} &= \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} (\mathbf{b}_i - \mathbf{b}_j)^T \mathbf{K}_r (\mathbf{b}_i - \mathbf{b}_j) \\ &= \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} [(\mathbf{b}_i - \mathbf{b}_j)^T \mathbf{v}_r]^2 \geq 0. \end{aligned} \tag{4}$$

To prevent $\boldsymbol{\beta}$ from degenerating to the zero vector $\mathbf{0}$ and to eliminate the scaling factor, we minimize the convex function $J_{\mathcal{S}}(\boldsymbol{\beta})$ subject to the linear constraint $\mathbf{1}^T\boldsymbol{\beta} = c$ for some constant $c > 0$. This is a simple convex optimization problem with a quadratic objective function and a linear equality

[3]This indicator variable will be "overloaded" later to refer to a column of any identity matrix whose size is clear from the context.

constraint. We introduce a Lagrange multiplier ρ to minimize the following Lagrangian:

$$J_{\mathcal{S}}(\boldsymbol{\beta}, \rho) = J_{\mathcal{S}}(\boldsymbol{\beta}) + \rho(c - \mathbf{1}^T\boldsymbol{\beta}). \tag{5}$$

The optimization problem can be solved easily to give the optimal value of $\boldsymbol{\beta}$ as the following closed-form solution:

$$\boldsymbol{\beta} = \frac{c\mathbf{D}_{\mathcal{S}}^{-1}\mathbf{1}}{\mathbf{1}^T\mathbf{D}_{\mathcal{S}}^{-1}\mathbf{1}}. \tag{6}$$

Note that $\mathbf{D}_{\mathcal{S}}^{-1}$ exists as long as all the diagonal entries of $\mathbf{D}_{\mathcal{S}}$ are positive, which is usually true.[4] We set the constant $c = \sum_{r=1}^{p}\sqrt{\lambda_r}$.

3 Scalable Kernel Learning

The kernel learning method described above requires performing eigendecomposition on $\mathbf{K}$. In case n is very large and hence $\mathbf{K}$ is a large matrix, operations such as eigendecomposition on $\mathbf{K}$ are computationally demanding. In this section, we apply low-rank approximation to extend the kernel learning method so that it scales up well with n.

3.1 Low-Rank Approximation

We apply low-rank approximation (Ye, 2005) to approximate $\mathbf{K}$ by another $n \times n$ symmetric and positive semi-definite matrix $\widetilde{\mathbf{K}}$:

$$\mathbf{K} \simeq \widetilde{\mathbf{K}} = \mathbf{W}\mathbf{L}\mathbf{W}^T, \tag{7}$$

[4] In case $\mathbf{D}_{\mathcal{S}}$ is really singular (though a rare case), a common way to make it invertible is to add a term $\epsilon\mathbf{I}$ to $\mathbf{D}_{\mathcal{S}}$ where ϵ is a small positive constant.

where $\mathbf{W} \in \mathbb{R}^{n \times m}$ is an $n \times m$ matrix and $\mathbf{L} \in \mathbb{R}^{m \times m}$ is an $m \times m$ symmetric and positive semi-definite matrix, with $m \ll n$.

There are different ways to construct $\mathbf{L}$ for low-rank approximation. We consider one way which constructs $\mathbf{L}$ using a subset of the n data points. We refer to these points as *landmarks* (de Silva & Tenenbaum, 2003; Weinberger, Packer, & Saul, 2005; Silva, Marques, & Lemos, 2006). Without loss of generality, we assume that the n points are ordered in such a way that the first m points form the set of landmarks $\{\mathbf{x}_i\}_{i=1}^m$. We should ensure that all points involved in $\mathcal{S}$ are chosen as landmarks. Other landmarks are randomly sampled from all the data points. Similar to $\mathbf{K}$ in the previous section, $\mathbf{L}$ is obtained here by applying the centering transform to $\hat{\mathbf{L}}$, as $\mathbf{L} = \mathbf{H}\hat{\mathbf{L}}\mathbf{H}$, where $\hat{\mathbf{L}} = [\hat{k}(\mathbf{x}_i, \mathbf{x}_j)]_{m \times m} = [\langle \hat{\phi}(\mathbf{x}_i), \hat{\phi}(\mathbf{x}_j) \rangle]_{m \times m}$ is the upper-left $m \times m$ submatrix of $\hat{\mathbf{K}}$.

We apply eigendecomposition on $\mathbf{L}$ and express it as

$$\mathbf{L} = \sum_{r=1}^{q} \mu_r \boldsymbol{\alpha}_r \boldsymbol{\alpha}_r^T = \mathbf{V}_{\boldsymbol{\alpha}} \mathbf{D}_{\boldsymbol{\mu}} \mathbf{V}_{\boldsymbol{\alpha}}^T, \tag{8}$$

where $\mu_1 \geq \cdots \geq \mu_q > 0$ are the $q \leq m$ positive eigenvalues of $\mathbf{L}$, $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q$ are the corresponding normalized eigenvectors, $\mathbf{D}_{\boldsymbol{\mu}} = \mathrm{diag}(\mu_1, \ldots, \mu_q)$, and $\mathbf{V}_{\boldsymbol{\alpha}} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q]$. Substituting (8) into (7), we can rewrite $\widetilde{\mathbf{K}}$ as

$$\begin{aligned} \widetilde{\mathbf{K}} &= \mathbf{W} \left(\sum_{r=1}^{q} \mu_r \boldsymbol{\alpha}_r \boldsymbol{\alpha}_r^T \right) \mathbf{W}^T \\ &= \sum_{r=1}^{q} \mu_r (\mathbf{W}\boldsymbol{\alpha}_r)(\mathbf{W}\boldsymbol{\alpha}_r)^T = \sum_{r=1}^{q} \mu_r \widetilde{\mathbf{K}}_r, \end{aligned} \tag{9}$$

where $\widetilde{\mathbf{K}}_r = (\mathbf{W}\boldsymbol{\alpha}_r)(\mathbf{W}\boldsymbol{\alpha}_r)^T$.

3.2 Kernel Learning

We apply low-rank approximation to devise a scalable kernel learning algorithm which can be seen as an extension of the algorithm described in Section 2. We use $\widetilde{\mathbf{K}}$ to approximate $\mathbf{K}$ and define the following parameterized family of kernel matrices:

$$\widetilde{\mathbf{K}}_{\boldsymbol{\beta}} = \sum_{r=1}^{q} \beta_r^2 \widetilde{\mathbf{K}}_r. \tag{10}$$

Note, however, that $\boldsymbol{\beta}$ is now a $q \times 1$ vector rather than a $p \times 1$ vector.

The optimal value of $\boldsymbol{\beta}$ has the same form as (6), except that the constant c is set to $\sum_{r=1}^{q} \sqrt{\mu_r}$ and $\mathbf{D}_{\mathcal{S}}$ is now a $q \times q$ diagonal matrix with diagonal entries

$$\begin{aligned} (\mathbf{D}_{\mathcal{S}})_{rr} &= \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} (\mathbf{b}_i - \mathbf{b}_j)^T \widetilde{\mathbf{K}}_r (\mathbf{b}_i - \mathbf{b}_j) \\ &= \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} [(\mathbf{b}_i - \mathbf{b}_j)^T (\mathbf{W} \boldsymbol{\alpha}_r)]^2 \geq 0. \end{aligned} \tag{11}$$

3.3 Computing the Embedding Weights

A question that remains to be answered is how to obtain $\mathbf{W}$ for low-rank approximation. We use a method that is similar to locally linear embedding (LLE) (Roweis & Saul, 2000; Saul & Roweis, 2003), with two differences. First, we only use the first part of LLE to obtain the weights for locally linear fitting. Second, we perform locally linear fitting in the kernel-induced feature space $\mathcal{H}$ rather than the input space $\mathcal{X}$.

Let $\mathbf{W} = [w_{ij}]_{n \times m}$ and $\mathbf{w}_i = (w_{i1}, \ldots, w_{im})^T$. If $\mathbf{x}_i$ is a landmark, i.e.,

$1 \leq i \leq m$, then

$$w_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

If $\mathbf{x}_i$ is not a landmark, then we minimize the following function to obtain $\mathbf{w}_i$:

$$E(\mathbf{w}_i) = \left\| \phi(\mathbf{x}_i) - \sum_{\phi(\mathbf{x}_j) \in \mathcal{N}_i} w_{ij} \phi(\mathbf{x}_j) \right\|^2, \tag{13}$$

where $\mathcal{N}_i$ is the set of K nearest landmarks of $\phi(\mathbf{x}_i)$ in $\mathcal{H}$, subject to the constraints $\sum_{\phi(\mathbf{x}_j) \in \mathcal{N}_i} w_{ij} = \mathbf{1}^T \mathbf{w}_i = 1$ and $w_{ij} = 0$ for all $\phi(\mathbf{x}_j) \notin \mathcal{N}_i$. We can rewrite $E(\mathbf{w}_i)$ as

$$\begin{aligned} E(\mathbf{w}_i) &= \sum_{\phi(\mathbf{x}_j), \phi(\mathbf{x}_k) \in \mathcal{N}_i} w_{ij} w_{ik} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_k)) \\ &= \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i, \end{aligned} \tag{14}$$

where

$$\begin{aligned} \mathbf{G}_i &= [(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_k))]_{K \times K} \\ &= [k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_k) - k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{x}_k)]_{K \times K} \end{aligned} \tag{15}$$

is the local Gram matrix of $\mathbf{x}_i$ in $\mathcal{H}$.

To prevent $\mathbf{w}_i$ from degenerating to $\mathbf{0}$, we minimize $E(\mathbf{w}_i)$ subject to the constraints $\sum_{\phi(\mathbf{x}_j) \in \mathcal{N}_i} w_{ij} = \mathbf{1}^T \mathbf{w}_i = 1$ and $w_{ij} = 0$ for all $\phi(\mathbf{x}_j) \notin \mathcal{N}_i$. As above for the kernel learning problem, we solve a convex optimization problem with a quadratic objective function and a linear equality constraint. The Lagrangian with a Lagrange multiplier α is as follows:

$$L(\mathbf{w}_i, \alpha) = \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i + \alpha(1 - \mathbf{1}^T \mathbf{w}_i). \tag{16}$$

The closed-form solution for this optimization problem is given by

$$\mathbf{w}_i = \frac{\mathbf{G}_i^{-1}\mathbf{1}}{\mathbf{1}^T\mathbf{G}_i^{-1}\mathbf{1}} \tag{17}$$

if $\mathbf{G}_i^{-1}$ exists.[5]

Instead of performing matrix inversion, a more efficient way of finding the solution is to solve the linear system of equations

$$\mathbf{G}_i\hat{\mathbf{w}}_i = \mathbf{1} \tag{18}$$

for $\hat{\mathbf{w}}_i$ and then compute $\mathbf{w}_i$ as

$$\mathbf{w}_i = \frac{\hat{\mathbf{w}}_i}{\mathbf{1}^T\hat{\mathbf{w}}_i} \tag{19}$$

to ensure that the equality constraint $\mathbf{1}^T\mathbf{w}_i = 1$ is satisfied.

3.4 Iterative Extension of Basic Algorithm

We assume above that the neighborhood relationships between points in $\mathcal{H}$ and the local Gram matrices remain fixed during the kernel learning process. A simple extension of this basic algorithm is to repeat the above procedure iteratively using the learned kernel at each iteration. Thus the basic algorithm is just a special case of this iterative extension when the number of iterations is equal to one.

3.5 Computational Complexity

A major motivation for our scalable kernel learning method based on low-rank approximation is to reduce the computational complexity so that the

[5]Similar to $\mathbf{D}_{\mathcal{S}}$ above, we may add $\epsilon\mathbf{I}$ to make sure that $\mathbf{G}_i$ is invertible.

algorithm can exhibit better scalability. In this subsection, we compare the computational complexity of the initial kernel learning method presented in Section 2 with that of our scalable method presented in this section.

Without exploiting any structure in the kernel matrix $\mathbf{K}$, eigendecomposition of $\mathbf{K}$ to express it in the form of (1) takes $O(n^3)$ time. The diagonal matrix $\mathbf{D}_{\mathcal{S}}$ can be computed according to (4) in $O(p|\mathcal{S}|)$ time and the optimal value of $\boldsymbol{\beta}$ can be computed according to (6) in $O(p)$ time. Note that $p \leq n$ and typically $|\mathcal{S}| < n$ or even $|\mathcal{S}| \ll n$. So the overall complexity of the method is $O(n^3)$ which is dominated by the complexity of the eigendecomposition step.

As for our method, eigendecomposition of $\mathbf{L}$ in (8) has $O(m^3)$ time complexity. The complexity of computing $\mathbf{D}_{\mathcal{S}}$ is $O(qm|\mathcal{S}|)$ and the complexity of computing $\boldsymbol{\beta}$ is $O(q)$. Moreover, we also need to compute $\mathbf{W}$. The rows corresponding to the landmarks are computed according to (12) in $O(m^2)$ time and those corresponding to the other data points take $O((n-m)(K^3+m))$ time. Note that $q \leq m$ and $K \leq m$. Hence the overall complexity is $O(m^3 + m^2|\mathcal{S}| + nK^3 + nm)$. As $m \ll n$, the complexity of our method is significantly lower than that of the original method.

Figure 1 shows the CPU time required by the initial and the scalable kernel learning methods on a digit data set which contains digits "0" and "1". The total number of data points $n = 4000$ (2000 data points from each digit) and the number of similarity constraints $|\mathcal{S}| = 50$. The empirical results agree well with the complexity analysis above. As shown in Figure 1(a), the speedup is more significant when the data set is large. The CPU time only

increases noticeably with the number of landmarks (Figure 1(b)).

4 Out-of-Sample Generalization

The exact form of out-of-sample generalization depends on the operation we want to perform. For example, the n given points are first clustered into $C \geq 2$ classes after kernel learning, and then a new data point $\mathbf{x}$ is classified into one of the C classes. We are interested in the case where both the clustering of the n points and the classification of new points are based on the same Euclidean metric in $\mathcal{H}$.

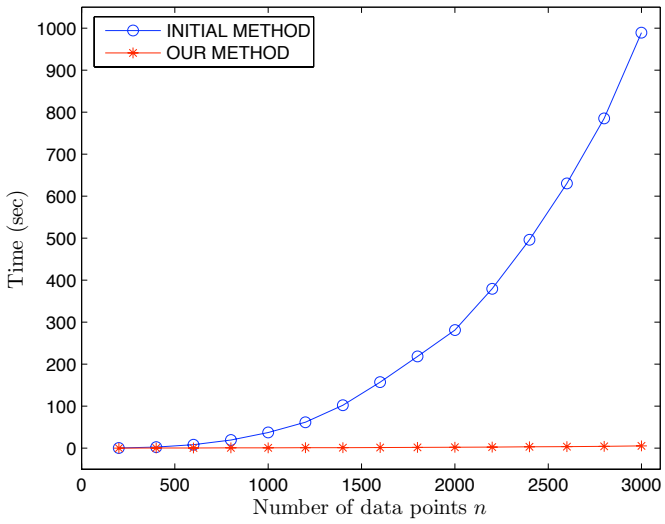
The key idea of our out-of-sample generalization scheme rests on the observation that kernel principal component analysis (KPCA) (Schölkopf, Smola, & Müller, 1998) can be performed on $\{\mathbf{x}_i\}_{i=1}^m$ to obtain an embedding in a q-dimensional subspace $\mathcal{Y}$ of $\mathcal{H}$, so that the non-landmark points $\{\mathbf{x}_i\}_{i=m+1}^n$ and any out-of-sample point $\mathbf{x}$ can be embedded into $\mathcal{Y}$ in the same way.

Let $\{\mathbf{u}_1, \ldots, \mathbf{u}_q\}$ be an orthonormal basis with each $\mathbf{u}_r \in \mathcal{H}$ being a unit vector along the direction of the rth principal component. We define $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_q]$. Then each landmark $\mathbf{x}_i \, (i = 1, \ldots, m)$ can be embedded into $\mathcal{Y}$ to give a q-dimensional vector

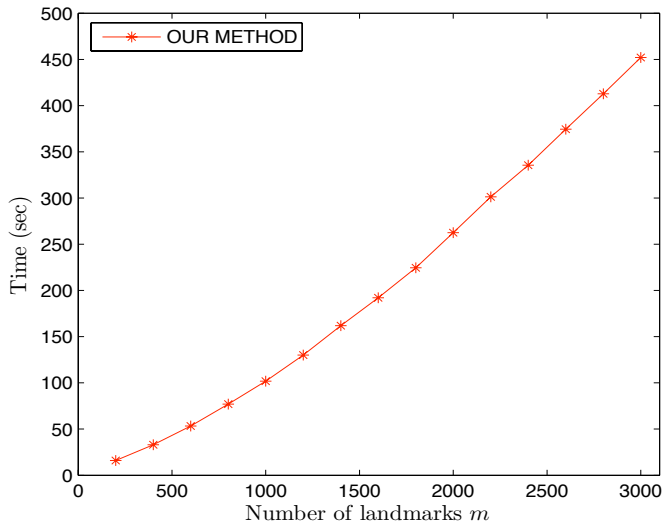
$$\mathbf{y}_i = \mathbf{U}^T \phi(\mathbf{x}_i). \tag{20}$$

Since

$$\mathbf{u}_r = \frac{1}{\sqrt{\mu_r}} \sum_{j=1}^{m} \alpha_{jr} \phi(\mathbf{x}_j), \tag{21}$$



(a)



(b)

Figure 1: CPU time required by the initial and the scalable kernel learning methods on a digit data set. (a) CPU time vs. data set size n for the two kernel learning methods with the number of landmarks m fixed at 100; (b) CPU time vs. number of landmarks m for the scalable method.

we can express $\mathbf{y}_i$ as

$$\begin{aligned}\mathbf{y}_i &= \mathbf{D}_{\boldsymbol{\mu}}^{-1/2}\mathbf{V}_{\boldsymbol{\alpha}}^T\mathbf{L}\mathbf{b}_i \\ &= \mathbf{D}_{\boldsymbol{\mu}}^{-1/2}\mathbf{V}_{\boldsymbol{\alpha}}^T\mathbf{V}_{\boldsymbol{\alpha}}\mathbf{D}_{\boldsymbol{\mu}}\mathbf{V}_{\boldsymbol{\alpha}}^T\mathbf{b}_i \\ &= \mathbf{D}_{\boldsymbol{\mu}}^{1/2}\mathbf{V}_{\boldsymbol{\alpha}}^T\mathbf{b}_i. \end{aligned} \tag{22}$$

Let $\mathbf{Y}_m = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$. So

$$\mathbf{Y}_m = \mathbf{D}_{\boldsymbol{\mu}}^{1/2}\mathbf{V}_{\boldsymbol{\alpha}}^T. \tag{23}$$

For the non-landmark points $\{\mathbf{x}_i\}_{i=m+1}^n$, from (13), we use $\widetilde{\phi}(\mathbf{x}_i) = \sum_{\phi(\mathbf{x}_j)\in\mathcal{N}_i} w_{ij}\phi(\mathbf{x}_j)$ to approximate $\phi(\mathbf{x}_i)$ and $\widetilde{\mathbf{y}}_i$ to denote the embedding of $\widetilde{\phi}(\mathbf{x}_i)$ in $\mathcal{Y}$. Thus we have

$$\begin{aligned}\widetilde{\mathbf{y}}_i = \mathbf{U}^T\widetilde{\phi}(\mathbf{x}_i) &= \sum_{\phi(\mathbf{x}_j)\in\mathcal{N}_i} w_{ij}\mathbf{U}^T\phi(\mathbf{x}_j) \\ &= \sum_{\phi(\mathbf{x}_j)\in\mathcal{N}_i} w_{ij}\mathbf{y}_j \\ &= \mathbf{Y}_m\mathbf{W}^T\mathbf{b}_i \\ &= \mathbf{D}_{\boldsymbol{\mu}}^{1/2}\mathbf{V}_{\boldsymbol{\alpha}}^T\mathbf{W}^T\mathbf{b}_i. \end{aligned} \tag{24}$$

Similarly, for any out-of-sample example $\mathbf{x}$, we use $\widetilde{\phi}(\mathbf{x}) = \sum_{\phi(\mathbf{x}_j)\in\mathcal{N}_i} w_j\phi(\mathbf{x}_j)$ to approximate $\phi(\mathbf{x})$ and $\widetilde{\mathbf{y}}$ to denote the embedding of $\widetilde{\phi}(\mathbf{x})$ in $\mathcal{Y}$. The embedding weights $\mathbf{w} = (w_1, \ldots, w_m)^T$ can be determined as in Section 3.3. Similar to the non-landmark points $\{\mathbf{x}_i\}_{i=m+1}^n$, we can obtain

$$\widetilde{\mathbf{y}} = \mathbf{D}_{\boldsymbol{\mu}}^{1/2}\mathbf{V}_{\boldsymbol{\alpha}}^T\mathbf{w}. \tag{25}$$

Based on (24) and (25), the squared Euclidean distance between $\widetilde{\mathbf{y}}_i$ and $\widetilde{\mathbf{y}}$ in $\mathcal{Y}$ before kernel learning can be expressed as

$$\begin{aligned} d^2(\mathbf{x}_i, \mathbf{x}) &= \|\widetilde{\mathbf{y}}_i - \widetilde{\mathbf{y}}\|^2 \\ &= (\mathbf{b}_i^T\mathbf{W} - \mathbf{w}^T)\mathbf{V}_{\boldsymbol{\alpha}}\mathbf{D}_{\boldsymbol{\mu}}\mathbf{V}_{\boldsymbol{\alpha}}^T(\mathbf{W}^T\mathbf{b}_i - \mathbf{w}) \\ &= (\mathbf{b}_i^T\mathbf{W} - \mathbf{w}^T)\mathbf{L}(\mathbf{W}^T\mathbf{b}_i - \mathbf{w}). \end{aligned} \tag{26}$$

The squared Euclidean distance after kernel learning is

$$d^2_{\boldsymbol{\beta}}(\mathbf{x}_i, \mathbf{x}) = (\mathbf{b}_i^T\mathbf{W} - \mathbf{w}^T)\mathbf{V}_{\boldsymbol{\alpha}}\mathbf{D}_{\boldsymbol{\beta}}\mathbf{V}_{\boldsymbol{\alpha}}^T(\mathbf{W}^T\mathbf{b}_i - \mathbf{w}), \tag{27}$$

where $\mathbf{D}_{\boldsymbol{\beta}} = \mathrm{diag}(\beta_1^2, \ldots, \beta_q^2)$.

5 Experimental Results

In this section, we present some experiments we have performed based on both artificial and real-world data.

5.1 Experimental Setup

We compare two versions of our kernel-based metric learning method described in Sections 2 and 3 with three metric learning methods. The first method is RCA (Bar-Hillel et al., 2003, 2005), which is a promising linear metric learning method that usually performs equally well as other computationally more demanding methods. The second method is locally linear metric adaptation (LLMA) (Chang & Yeung, 2004), which is more general in that it is linear locally but nonlinear globally. The third method is large

margin nearest neighbor (LMNN) (Weinberger, Blitzer, & Saul, 2006), which learns a global Mahalanobis metric for nearest neighbor classification with the goal that data points from different classes are separated by a large margin. For baseline comparison, we also include two metrics without metric learning. They are the Euclidean metric in the input space and the Euclidean metric in the feature space induced by a Gaussian RBF kernel.

For each data set, we randomly generate 10 different $\mathcal{S}$ sets. For small data sets, we can learn $\mathbf{K}_{\boldsymbol{\beta}}$ without low-rank approximation. For large data sets, in addition to the data points involved in $\mathcal{S}$, we also randomly select some other points as landmarks for learning $\widetilde{\mathbf{K}}_{\boldsymbol{\beta}}$. We use the iterative extension of the scalable kernel learning algorithm with the number of iterations equal to 3. We also measure the change in metric learning performance as the number of landmarks increases.

5.2 An Illustrative Example

Figure 2 uses the XOR data set to compare the performance of different metrics, some with metric learning. Figure 2(a) shows 360 data points in the input space. The points with the same color and mark belong to the same class. The randomly generated point pairs corresponding to the similarity constraints in $\mathcal{S}$ are shown in solid lines (Figure 2(b)). Forty new data points are shown in Figure 2(c), with different marks used to distinguish them from the original input data points. The results obtained by RCA, LMNN, LLMA, RBF kernel and two versions of our method are shown in Figure 2(d)–(i).

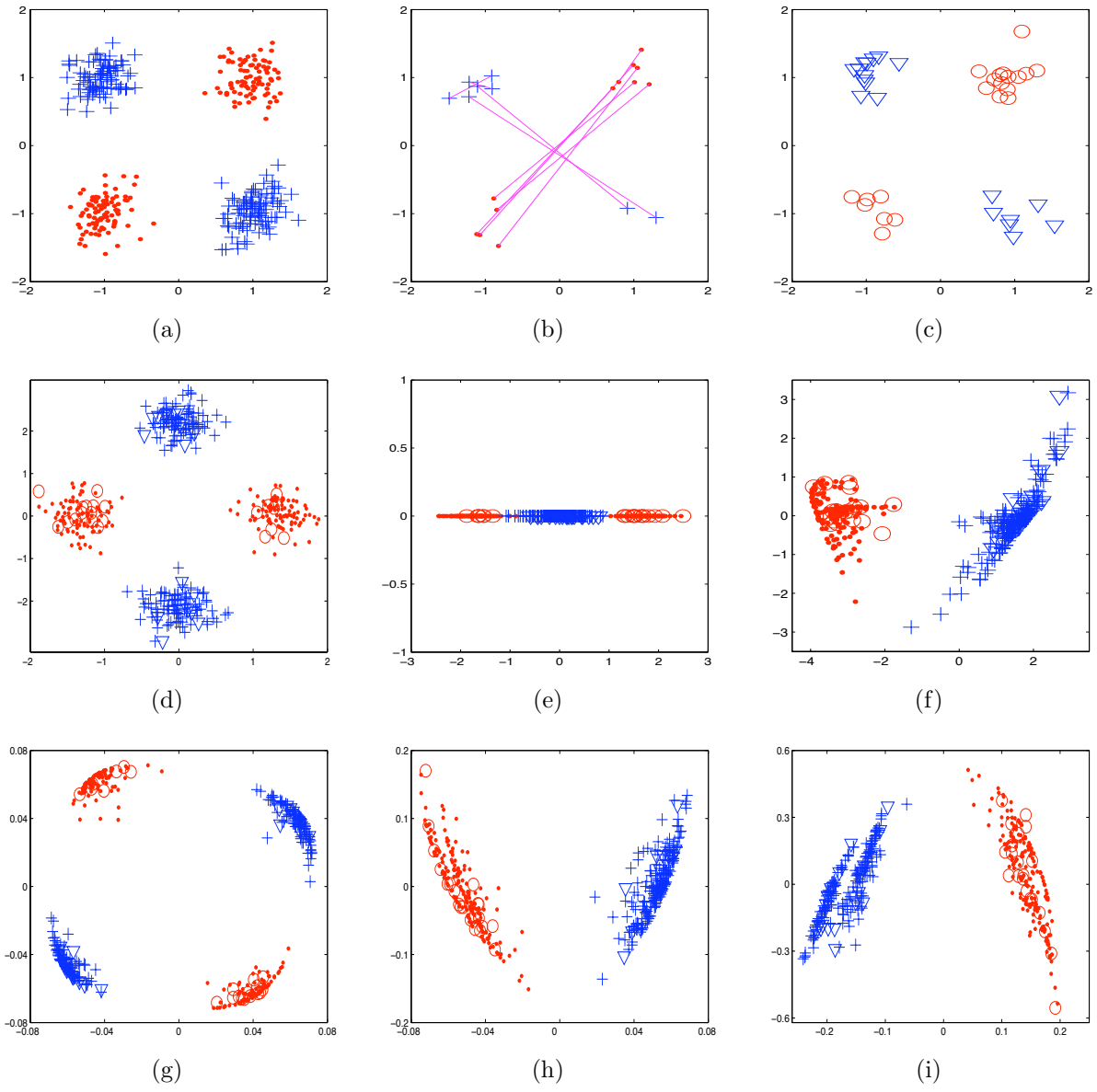


Figure 2: XOR illustration. (a) input data; (b) similarity constraints; (c) new data; (d) RCA; (e) LMNN; (f) LLMA; (g) RBF; (h) $\mathbf{K}_{\boldsymbol{\beta}}$; (i) $\widetilde{\mathbf{K}}_{\boldsymbol{\beta}}$ ($m = 40$).

RCA, LMNN, and LLMA perform metric learning directly in the input space, which can be generalized to new data using the learned transformation. For the kernel methods, we apply KPCA using the (learned) kernel matrix to embed the data points to a 2-dimensional space, as shown in Figure 2(g)–(i). New data points can be embedded using the generalization method described in Section 4. As expected, neither RCA nor LMNN performs satisfactorily for the XOR data set since it can only perform linear transformation. LLMA can group the points according to their class membership, although the topology of the data set is not preserved well after metric adaptation. On the other hand, our kernel-based metric learning methods can give the best results. The result of the scalable kernel learning method with low-rank approximation based on 40 landmarks is almost as good as that of the basic algorithm based on all 400 data points. Moreover, the result on embedding of new data points verifies the effectiveness of our out-of-sample generalization method.

5.3 Quantitative Performance Comparison

5.3.1 Internal Performance Measure

Let $\{y_i\}_{i=1}^n$ be the set of true class labels of the n data points. We define the following performance measure:

$$J = \frac{\overline{d}_b}{\overline{d}_w}, \tag{28}$$

where $\overline{d}_b = (1/n_b) \sum_{y_i \neq y_j} \|\mathbf{x}_i - \mathbf{x}_j\|$ is the mean between-class distance with n_b being the number of point pairs with different class labels, and $\overline{d}_w =$

$(1/n_w)\sum_{y_i=y_j}\|\mathbf{x}_i-\mathbf{x}_j\|$ is the mean within-class distance with n_w being the number of point pairs with the same class label. Note that J is closely related to the optimization criterion $J_{\mathcal{S}}$ in (3), except that $J_{\mathcal{S}}$ is defined for the labeled data (i.e., data involved in the pairwise constraints) only while J is for all data assuming the existence of true class labels. Since LLMA is not efficient for large data sets, we only compare our kernel-based methods with RCA and RBF kernel in this subsection. For kernel methods, we use $\phi(\mathbf{x}_i)$ or $\widetilde{\phi}(\mathbf{x}_i)$ in place of $\mathbf{x}_i$ and apply the kernel trick to compute the mean distances and hence J. A larger value of J corresponds to a better metric due to its higher class or cluster separability.

We first perform some experiments on a much larger XOR data set with 8,000 data points. We randomly select 50 similarity constraints to form $\mathcal{S}$ and measure the metric learning performance in terms of the J value for an increasing number of landmarks. Table 1 shows the results for different metrics. For the metric learning methods (i.e., RCA and our method), we show for each trial the mean (upper) and standard deviation (lower) over 10 random runs corresponding to different $\mathcal{S}$ sets. From the results, we can see that our method outperforms the other methods significantly. Moreover, using more landmarks generally gives better results.

We further perform some experiments on real-world data sets. One of them is the Isolet data set from the UCI Machine Learning Repository which contains 7,797 isolated spoken English letters belonging to 26 classes, with each letter represented as a 617-dimensional vector. Other data sets are

Table 1: Performance comparison in terms of J value for XOR data set ($|\mathcal{S}| = 50$, m = 100:100:800).

Input data	RBF	RCA	Our method			
			$m = 100$	$m = 200$	$m = 300$	$m = 400$
1.2460	1.4253	1.2552 ±0.19	**2.8657** ±1.28	**3.0886** ±0.42	**3.5640** ±0.78	**3.8422** ±1.25

$m = 500$	$m = 600$	$m = 700$	$m = 800$
4.2378 ±0.99	**4.6395** ±0.68	**4.8334** ±0.90	**4.7463** ±0.65

handwritten digits from the MNIST database.[6] The digits in the database have been size-normalized and centered to 28×28 gray-level images. Hence the dimensionality of the input space is 784. In our experiments, we randomly choose 2,000 images for each digit from a total of 60,000 digit images in the MNIST training set. We use 50 similarity constraints and 100 landmarks in the experiments. The results for Isolet and different digit data sets are shown in Table 2. From the results, we can again see that the metric learned by our method is the best in terms of the J measure.

To assess the performance of different metric learning methods with respect to the number of similarity constraints $|\mathcal{S}|$, we perform more experi-

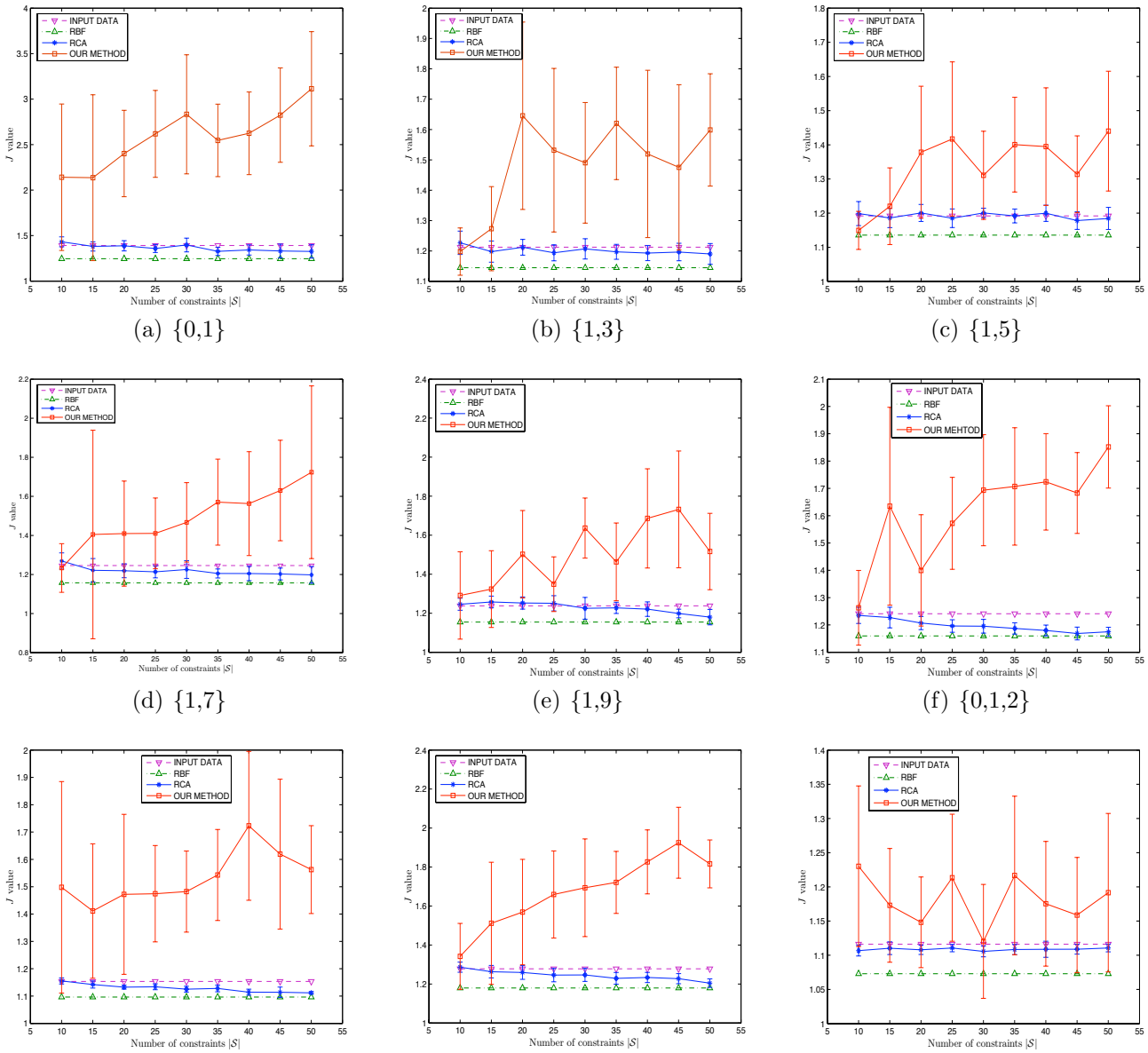
[6] `http://yann.lecun.com/exdb/mnist/`

ments on the MNIST data sets with different sizes $|\mathcal{S}|$. Figure 3 shows the results for different MNIST data sets with the J value used as performance measure. As we can see, the performance of RCA cannot improve with more similarity constraints. On the other hand, the J value of our method usually increases with the number of constraints, with some fluctuations reflecting the randomness involved in choosing the constraint sets.

5.3.2 External Performance Measure

Besides using the generic performance measure J which is not designed for a specific machine learning task, we further conduct external performance evaluation on some semi-supervised learning tasks to see how much the metric learning methods can help to improve the learning performance. First, we report k-means clustering results based on the learned distance metrics for the above Isolet and MNIST data sets. Then, we compare our kernel methods with state-of-the-art metric learning methods on semi-supervised classification and clustering tasks in Sections 5.4 and 5.5, where we use classification accuracy and *Rand index* (Rand, 1971), respectively, as external performance measures.

The Rand index reflects the agreement of the clustering result with the ground truth. Let n_s be the number of point pairs that are assigned to the same cluster (i.e., matched pairs) in both the resultant partition and the ground truth, and n_d be the number of point pairs that are assigned to different clusters (i.e., mismatched pairs) in both the resultant partition and the ground truth. The Rand index is defined as the ratio of $(n_s + n_d)$ to the



(a) {0,1} (b) {1,3} (c) {1,5}

(d) {1,7} (e) {1,9} (f) {0,1,2}

(g) {6,7,8} (h) {0,1,9} (i) {3,4,5,6}

Figure 3: Performance comparison in terms of J value vs. number of similarity constraints $|\mathcal{S}|$ for MNIST data sets.

total number of point pairs, i.e., $n(n-1)/2$. When there are more than two clusters, however, the standard Rand index will favor assigning data points to different clusters. We modify the Rand index as in (Xing et al., 2003) so that matched pairs and mismatched pairs are assigned weights to give them equal chance of occurrence (0.5).

We first externally evaluate the effectiveness of our kernel-based metric learning method for k-means clustering on the Isolet and MNIST data sets. As in (Bar-Hillel et al., 2003) and (Chang & Yeung, 2004), we use the Rand index to measure the clustering quality. The comparison results are shown in Table 3. As we can see from the table, our kernel-based method generally outperforms the other methods in terms of the Rand index, though the improvements are not as large as those based on the internal performance measure J.

5.4 Evaluation on Semi-Supervised Classification Tasks

In this subsection, we perform experiments on semi-supervised classification and compare our kernel-based method with RCA and LMNN. For each data set and percentage of training data, we randomly select 10 different subsets of the labeled data.[7] Classification accuracy is used as performance measure.

The performance comparison in terms of classification accuracy for the Isolet and MNIST data sets is shown in Table 4. As we can see, while LMNN can dramatically improve the classification performance, our method gives

[7]While LMNN directly makes use of the labeled data, we extract pairwise similarity constraints from the label information for RCA and our kernel-based method.

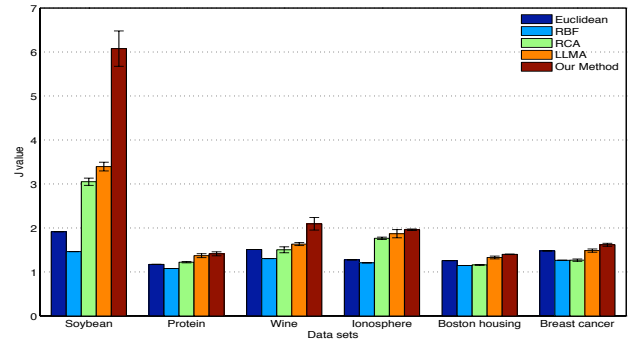
even better results.

5.5 Evaluation on Semi-Supervised Clustering Tasks

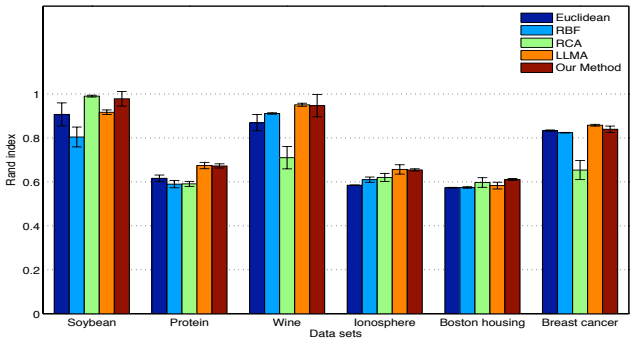
In this subsection, we conduct performance evaluation on some semi-supervised clustering tasks and use the Rand index as the external performance measure for clustering quality.

We perform semi-supervised clustering experiments on six real-world data sets from the UCI Machine Learning Repository: Soybean (47/35/4/10), Protein (116/20/6/15), Wine (178/13/3/20), Ionosphere (351/34/2/30), Boston housing (506/13/3/40), and Breast cancer (569/31/2/50). The numbers inside the brackets ($n/d/c/|\mathcal{S}|$) summarize the characteristics of the data sets, including the number of data points n, number of features d, number of clusters c, and number of randomly selected point pairs for the similarity constraints. Note that these UCI data sets are much smaller than those used in Section 5.3, so LLMA can also be included here for experimental comparison.

The semi-supervised clustering results for the six UCI data sets are shown in Figure 4 in terms of both the J value and the Rand index. As we can see, our kernel-based metric learning method significantly outperforms all other methods in terms of the J value. As for semi-supervised clustering, LLMA and the kernel method are comparable in performance in terms of the Rand index, with the kernel method being slightly better than LLMA. More specifically, based on a paired t-test with significance level 0.05, the



(a) J value



(b) Rand Index

Figure 4: Performance comparison in terms of J value and Rand index for six UCI data sets.

kernel method is better than LLMA on the Soybean and Boston housing data sets but worse on the Breast cancer data set.

6 Concluding Remarks

We have presented a simple and efficient kernel-based semi-supervised metric learning method based on supervisory information in the form of pairwise

similarity constraints. Not only does it scale up well with the data set size, it can also naturally lead to out-of-sample generalization. Although previous studies by other researchers showed that pairwise dissimilarity constraints usually cannot help much in many real-world applications, there are situations when incorporating them may still be helpful and hence we plan to extend our method to incorporate dissimilarity constraints as well. In our low-rank approximation scheme, besides including those points involved in $\mathcal{S}$, we also randomly sample some other points as landmarks. A recent study (Silva et al., 2006) shows that non-uniform sampling of landmarks for manifold learning can give parsimonious approximations using only very few landmarks. We will pursue research along this direction in our future work.

Acknowledgments

This research has been supported by Competitive Earmarked Research Grant (CERG) 621706 from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China.

References

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2003, 21–24 August). Learning distance functions using equivalence relations. In *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 11–18). Washington, DC, USA.

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2005). Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, *6*, 937–965.

Bilenko, M., Basu, S., & Mooney, R. (2004, 4–8 July). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning* (pp. 81–88). Banff, Alberta, Canada.

Bousquet, O., & Herrmann, D. (2003). On the complexity of learning the kernel matrix. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15* (pp. 399–406). Cambridge, MA, USA: MIT Press.

Boykin, P., & Roychowdhury, V. (2005). Leveraging social networks to fight spam. *IEEE Computer*, *38*(4), 61–68.

Chang, H., & Yeung, D. (2004, 4–8 July). Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning* (pp. 153–160). Banff, Alberta, Canada.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. (2002). On kernel-target alignment. In T. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14* (pp. 367–373). Cambridge, MA, USA: MIT Press.

de Silva, V., & Tenenbaum, J. (2003). Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15* (pp.

705–712). Cambridge, MA, USA: MIT Press.

Grira, N., Crucianu, M., & Boujemaa, N. (2005, August). *Unsupervised and semi-supervised clustering: a brief survey* (Tech. Rep.). France: INRIA Rocquencourt.

Hertz, T., Bar-Hillel, A., & Weinshall, D. (2004, 4–8 July). Boosting margin based distance functions for clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning* (pp. 393–400). Banff, Alberta, Canada.

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. (2002, 8–12 July). Learning the kernel matrix with semi-definite programming. In *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 323–330). Sydney, Australia.

Lu, Z., & Leen, T. (2005). Semi-supervised learning with penalized probabilistic clustering. In L. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems 17* (pp. 849–856). Cambridge, MA, USA: MIT Press.

Rand, W. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*, 846–850.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Saul, L., & Roweis, S. (2003). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, *4*, 119–155.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels*. Cambridge, MA,

USA: MIT Press.

Schölkopf, B., Smola, A., & Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.

Schultz, M., & Joachims, T. (2004). Learning a distance metric from relative comparisons. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems 16.* Cambridge, MA, USA: MIT Press.

Seeger, M. (2000). *Learning with labeled and unlabeled data* (Tech. Rep.). Edinburgh, UK: Institute for Adaptive and Neural Computation, University of Edinburgh.

Shental, N., Bar-Hillel, A., Hertz, T., & Weinshall, D. (2004). Computing Gaussian mixture models with EM using equivalence constraints. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems 16.* Cambridge, MA, USA: MIT Press.

Silva, J., Marques, J., & Lemos, J. (2006). Selecting landmark points for sparse manifold learning. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in Neural Information Processing Systems 18* (pp. 1241–1248). Cambridge, MA, USA: MIT Press.

Tsuda, K., Akaho, S., & Asai, K. (2003). The *em* algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, *4*, 67–81.

Vapnik, V. (1998). *Statistical Learning Theory.* New York, NY, USA: Wiley.

Wagstaff, K., & Cardie, C. (2000, 29 June – 2 July). Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Con-*

ference on Machine Learning (pp. 1103–1110). Stanford, CA, USA.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001, 28 June – 1 July). Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 577–584). Williamstown, MA, USA.

Weinberger, K., Blitzer, J., & Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in Neural Information Processing Systems 18* (pp. 1473–1480). Cambridge, MA, USA: MIT Press.

Weinberger, K., Packer, B., & Saul, L. (2005, 6–8 January). Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics* (pp. 381–388). Barbados.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15* (pp. 505–512). Cambridge, MA, USA: MIT Press.

Ye, J. (2005). Generalized low rank approximations of matrices. *Machine Learning*, *61*(1–3), 167–191.

Zhang, Z., Yeung, D., & Kwok, J. (2004, 4–8 July). Bayesian inference for transductive learning of kernel matrix using the Tanner-Wong data augmentation algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning* (pp. 935–942). Banff, Alberta,

Canada.

Zhu, X. (2006, December 9 (last modified)). *Semi-supervised learning literature survey* (Tech. Rep. No. 1530). Department of Computer Science, Madison, Wisconsin, USA: University of Wisconsin – Madison.

Table 2: Performance comparison in terms of J value for Isolet and MNIST data sets ($|\mathcal{S}| = 50$, $m = 100$).

	Isolet	MNIST				
		$\{0,1\}$	$\{1,3\}$	$\{1,5\}$	$\{1,7\}$	$\{1,9\}$
Input data	1.3888	1.3914	1.2124	1.1920	1.2458	1.2379
RBF	1.2477	1.2460	1.1447	1.1357	1.1570	1.1548
RCA	1.3049	1.2970	1.1779	1.1705	1.1820	1.2086
	$\pm$0.0030	$\pm$0.0324	$\pm$0.0270	$\pm$0.0283	$\pm$0.0399	$\pm$0.0391
Our method	**2.7938**	**2.9078**	**1.7070**	**1.4015**	**1.5463**	**1.7023**
	$\pm$0.0430	$\pm$0.4614	$\pm$0.2252	$\pm$0.1400	$\pm$0.2055	$\pm$0.3567

	$\{0,1,2\}$	$\{6,7,8\}$	$\{0,1,9\}$	$\{3,4,5,6\}$
Input data	1.2408	1.1534	1.2779	1.1162
RBF	1.1598	1.0963	1.1796	1.0729
RCA	1.1844	1.1207	1.2087	1.0793
	$\pm$0.0311	$\pm$0.0160	$\pm$0.0200	$\pm$0.0110
Our method	**1.8620**	**1.6233**	**1.9608**	**1.2945**
	$\pm$0.3160	$\pm$0.1996	$\pm$0.1884	$\pm$0.0667

Table 3: Performance comparison in terms of Rand index for Isolet and MNIST data sets ($|\mathcal{S}| = 50$, $m = 100$).

	Isolet	MNIST		
		$\{0, 1, 2\}$	$\{6, 7, 8\}$	$\{3, 4, 5, 6\}$
Input data	0.7374	0.8779	0.9214	0.7287
	$\pm$0.0143	$\pm$0.0037	$\pm$0.0002	$\pm$0.0003
RBF	0.6848	0.8055	0.8908	0.6840
	$\pm$0.0113	$\pm$0.0032	$\pm$0.0001	$\pm$0.0551
RCA	0.7627	**0.8936**	0.9243	0.7875
	$\pm$0.0183	$\pm$0.0138	$\pm$0.0032	$\pm$0.0338
Our method	**0.7712**	0.8855	**0.9310**	**0.7951**
	$\pm$0.0269	$\pm$0.0161	$\pm$0.0245	$\pm$0.0322

Table 4: Performance comparison in terms of classification accuracy for Isolet and MNIST data sets.

	Isolet				MNIST $\{0,1,2,3,4\}$	
training data	5%	10%	15%	20%	5%	10%
Input data	0.6123	0.7017	0.7518	0.7624	0.2768	0.3592
	±0.0106	±0.0024	±0.0037	±0.0062	±0.0106	±0.0212
RCA	0.8654	0.8851	0.7531	0.7655	0.2811	0.8431
	±0.0083	±0.0033	±0.0045	±0.0018	±0.0117	±0.0157
LMNN	0.8673	0.9125	0.9234	**0.9328**	0.7370	0.8683
	±0.0067	±0.0032	±0.0031	±0.0020	±0.0247	±0.0142
Our method	**0.8810**	**0.9189**	**0.9237**	0.9276	**0.7874**	**0.8851**
	±0.0143	±0.0061	±0.0100	±0.0087	±0.0225	±0.0140

training data	15%	20%
Input data	0.4336	0.5024
	±0.0238	±0.0144
RCA	0.8801	0.8931
	±0.0078	±0.0079
LMNN	0.9126	**0.9349**
	±0.0068	±0.0070
Our method	**0.9223**	0.9306
	±0.0112	±0.0087