Metric Learning: A Support Vector Approach

Nam Nguyen and Yunsong Guo

Department of Computer Science
Cornell University, USA
{nhnguyen,guoys}@cs.cornell.edu

Abstract. In this paper, we address the metric learning problem utilizing a margin-based approach. Our metric learning problem is formulated as a quadratic semi-definite programming problem (QSDP) with local neighborhood constraints, which is based on the Support Vector Machine (SVM) framework. The local neighborhood constraints ensure that examples of the same class are separated from examples of different classes by a margin. In addition to providing an efficient algorithm to solve the metric learning problem, extensive experiments on various data sets show that our algorithm is able to produce a new distance metric to improve the performance of the classical K-nearest neighbor (KNN) algorithm on the classification task. Our performance is always competitive and often significantly better than other state-of-the-art metric learning algorithms.

Keywords: metric learning, K-nearest neighbor classification, SVM.

1 Introduction

The distance metric learning problem, i.e. learning a distance measure over an input space, has received much attention in the machine learning community recently [1,2,3,4,5,6,7,8,9,10]. This is because designing a good distance metric is essential to many distance-based learning algorithms. For example, the K-nearest neighbor (KNN) [11], which is a classical classification method and requires no training effort, critically depends on the quality of the distance measures among examples. The classification of a new example is determined by the class labels of the K training examples with shortest distances. Traditionally the distance measure between two examples is defined to be the Euclidean distance between the features of the examples.

Although KNN is a simple method by nowadays standard, it is still an active research area [12,13], widely used in real-world applications [14], and serves as a basic component in more complex learning models [15,16]. Hence, improving KNN prediction accuracy would have a significant impact on this part of the machine learning community.

The performance of the KNN algorithm is influenced by three main factors: (i) the distance function or distance metric used to determine the nearest neighbors; (ii) the decision rule used to derive a classification from the K-nearest neighbors;

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 125–136, 2008.

and (iii) the number of neighbors used to classify the new example. Ideally the distance function should bring examples with the same labels closer to one another and push examples with different labels further apart. However, the ideal distance metric is hard to achieve. On the other hand, it is clear that with the right distance metric, KNN can perform exceptionally well using a simple majority-voting decision rule and a fixed number of nearest neighbors. Hence, in this paper we focus on the first main factor of KNN to derive a good distance function utilizing the maximum-margin approach [17].

Our approach presented in this paper is to learn a Mahalanobis distance function by minimizing a quadratic objective function subject to local neighborhood constraints. The goal of this optimization is, for every example, to bring examples of the same class close and to separate examples from other classes by a large margin. This goal is very similar to the one proposed in [7]. However, in [7] the authors solved a different optimization problem via semidefinite programming. In our framework, similar to most other support vector machine approaches our proposed algorithm is able to handle the non-linear separable cases by utilize the kernel trick. Finally, our extensive experiments with various data sets from the UCI repository show that our proposed metric learning algorithm is able to produce a better distance function that helps improve the performance of KNN classification in comparison to other state-of-the-art metric learning algorithms.

The paper is organized as follows: in section 2, we describe our metric learning algorithm via the support vector machine approach in detail; in section 3, we review other state-of-the-art metric learning algorithms; the experimental results and conclusions are presented in section 4 and 5.

2 Metric Learning: A Margin-Based Approach

In this section, we consider the following supervised learning framework. The learning algorithm takes a set of labeled training examples, $\mathbf{L} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$ as input, where $\mathbf{x}_i \in \mathcal{X}$ the input space and y_i belongs to a finite set of classes denoted by $\mathcal{Y}$. The goal of the metric learning problem is to produce a Mahalanobis distance function which can be used in computing nearest neighbors for the KNN classifier. In most metric learning algorithms, a positive semidefinite matrix $\mathbf{A} \succeq 0$, known as the Mahalanobis distance matrix, is learned as a transformation matrix in computing the (squared) distance between examples $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}_j).$$

Alternatively, we rewrite the distance function as,

$$d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{A}, (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \rangle,$$

where $\langle \cdot, \cdot \rangle$ represents the matrix inner-product.

A desirable distance function preserves the local neighborhood property that examples within the same class are separated from examples of different classes by a large margin in the new distance space. This translates to the following

local neighborhood constraint that, for each example its distance to all neighboring examples of the same class is smaller than its distance to any neighboring examples of different classes by at least a predefined margin, such as 1. Figure 1 demonstrates the idea how the neighborhood of a training example is transformed after applying the new distance function. Formally, the above constraint can be represented by the following inequalities,

$$\forall(\mathbf{x}_i, y_i) \in \mathbf{L}:$$

$$\min_{\substack{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ y_j \neq y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) \geq \max_{\substack{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ y_j = y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) + 1, \tag{1}$$

where $\mathcal{N}(\mathbf{x}_i)$ represents the set of neighbor examples of $\mathbf{x}_i$. In the absence of prior knowledge, the neighborhood set for each example is determined by the Euclidean distance. Along with the labeled training data $\mathbf{L}$, our metric learning algorithm also takes an integer K in the input which servers as the size of the constructed neighborhood $\mathcal{N}(\cdot)$, as well as the input to the KNN classification algorithm.

In our learning framework, we seek the positive semidefinite matrix that minimizes a quadratic objective function subject to the above constraints presented in inequality (1). Rather than using the hard margin constraints, we incorporate slack variables, ξ, to obtain the quadratic optimization problem with the soft margin constraints. Formally, the metric learning algorithm we propose

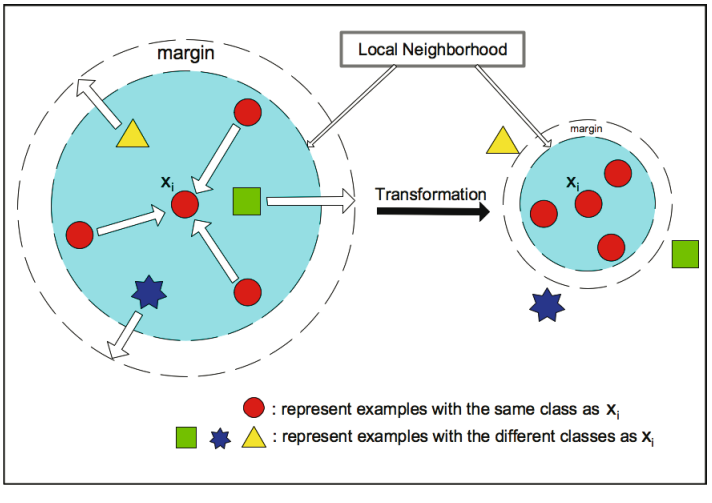

Fig. 1. Illustration of the local neighborhood property for an example $\mathbf{x}_i$ where before applying the transformation (on the left) the neighborhood of $\mathbf{x}_i$ contains both examples from the same and different classes; and after applying the transformation (on the right), examples within the same class of $\mathbf{x}_i$ cluster together and are separated from examples from different classes by a large margin

(MLSVM) learns a positive semidefinite matrix $\mathbf{A}$ and slack variables ξ via the following quadratic semi-definite programming problem (QSDP):

Optimization Problem: MLSVM

$$\min_{\mathbf{A}\succeq 0, \xi \geq 0} : \frac{\lambda}{2}\|\mathbf{A}\|_F^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i \tag{2}$$

subject to:
$\forall(\mathbf{x}_i, y_i) \in \mathbf{L}$:

$$\min_{\substack{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ y_j \neq \ y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) \geq \max_{\substack{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ y_j = \ y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) + 1 - \xi_i,$$

where $\|\cdot\|_F$ represents the Frobenius norm for matrices, $\|\mathbf{A}\|_F^2 = \sum_i \sum_j \mathbf{A}_{ij}^2$.

In order to solve MLSVM, we extend the Pegasos method from [18], which is an effective iterative algorithm for solving the SVM QP via gradient descent without the need to transform the formulation into its dual form as many other SVM based methods do [19,20]. Our algorithm is called MLSVM, as from the optimization problem. In each iteration, MLSVM follows three steps: gradient descent, positive semidefinite approximation, and projection steps. In the first step, the algorithm computes a set $\mathbf{V} \subset \mathbf{L}$ that contains examples violating the local neighborhood constraints. Then the matrix $\mathbf{A}$ is updated according to the violated constraint set $\mathbf{V}$. In the positive semidefinite approximation step, the algorithm finds a positive semi-definite matrix $\hat{\mathbf{A}}$ that is closest to the current matrix $\mathbf{A}$ in term of the Forbenius norm, i.e.

$$\hat{\mathbf{A}} = \underset{\tilde{\mathbf{A}}\succeq 0}{\operatorname{argmin}} \|\mathbf{A} - \tilde{\mathbf{A}}\|_F^2. \tag{3}$$

A solution to this problem is

$$\hat{\mathbf{A}} = \sum_{i=1}^{m} \max\{\lambda_i, 0\}\mathbf{v}_i\mathbf{v}_i^T,$$

where $\{\lambda_i\}_{i=1}^m$ and $\{\mathbf{v}_i\}_{i=1}^m$ represent the m eigenvalues and the corresponding m eigenvectors of the matrix $\mathbf{A}$ respectively. In other words, to obtain a nearest positive semidefinite matrix, we simply remove the terms in the eigenvector expansion that correspond to negative eigenvalues. In the projection step, the matrix $\mathbf{A}$ is projected to the sphere of radius $1/\sqrt{\lambda}$. Details of MLSVM are presented in Algorithm 1.

Kernelizing the Algorithm

We now consider kernelizing our metric learning algorithm. In Algorithm 1, if we initialize $\mathbf{A}_1 = 0$ then $\mathbf{A}_t$ can be expressed as

$$\mathbf{A}_t = \sum_{i=1}^{n}\sum_{j=1}^{n} \varphi_{ij}\mathbf{x}_i\mathbf{x}_j^T.$$

Algorithm 1. Metric Learning SVM (MLSVM)

Input: $\mathbf{L}$ - the labeled data
λ and T - parameters of the QP

Initialization: Choose $\mathbf{A}_1$ such that $\|\mathbf{A}_1\|_F \leq 1/\sqrt{\lambda}$
for $t = 1$ **to** T **do**
Set $\mathbf{V} = \{(\mathbf{x}_i, y_i) \in \mathbf{L} \mid \min_{\substack{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ y_j \neq y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) - \max_{\substack{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ y_j = y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) < 1\}$
Set $\eta_t = \dfrac{1}{\lambda t}$
Set $\mathbf{A}_{t+\frac{1}{3}} = (1 - \eta_t\lambda)\mathbf{A}_t + \dfrac{\eta_t}{n} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{V}} \Big[\left(\mathbf{x}_i - \mathbf{x}_i^-\right)\left(\mathbf{x}_i - \mathbf{x}_i^-\right)^T - \left(\mathbf{x}_i - \mathbf{x}_i^+\right)\left(\mathbf{x}_i - \mathbf{x}_i^+\right)^T \Big]$

where $\mathbf{x}_i^- = \operatorname{argmin}_{\substack{\mathbf{x}_i^- \in \mathcal{N}(\mathbf{x}_i) \\ y_i^- \neq y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_i^-)$,
$\mathbf{x}_i^+ = \operatorname{argmax}_{\substack{\mathbf{x}_i^+ \in \mathcal{N}(\mathbf{x}_i) \\ y_i^+ = y_i}} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_i^+)$

Set $\mathbf{A}_{t+\frac{2}{3}} = \operatorname{argmin}_{\mathbf{A} \succeq 0} \|\mathbf{A}_{t+\frac{1}{3}} - \mathbf{A}\|_F^2$
Set $\mathbf{A}_{t+1} = \min\left\{1, \dfrac{1/\sqrt{\lambda}}{\|\mathbf{A}_{t+\frac{2}{3}}\|_F}\right\} \mathbf{A}_{t+\frac{2}{3}}$
end for

Output: $\mathbf{A}_{T+1}$

Hence, we can incorporate the use of kernels when computing the matrix inner-product operation and the Frobenius norm:

$$\langle \mathbf{A}, \mathbf{x}\mathbf{x}'^T \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \varphi_{ij} \kappa(\mathbf{x}, \mathbf{x}_i) \kappa(\mathbf{x}', \mathbf{x}_j)$$

$$\|\mathbf{A}\|_F^2 = \sum_{i,j} \sum_{i',j'} \varphi_{ij} \varphi_{i'j'} \kappa(\mathbf{x}_i, \mathbf{x}_{i'}) \kappa(\mathbf{x}_j, \mathbf{x}_{j'}),$$

where $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ is the kernel function and $\Phi : \mathcal{X} \mapsto \mathcal{F}$ projects examples to a new space. In our experiment, we consider the polynomial kernel function, i.e. $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d$.

In the kernel space, the matrix $\mathbf{A}$ does not have an explicit form since $\mathbf{A}$ is only expressed as $\mathbf{A} = \sum_{i=1}^{n} \sum_{j=1}^{n} \varphi_{ij} \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_j)^T$ where $\Phi(\mathbf{x})$ may have infinite dimensions. Similar to the problem of Kernel Principal Component Analysis [21], in order to carry out the semidefinite matrix approximation step, we now have to find eigenvalues $\lambda \geq 0$ and corresponding eigenvectors $\mathbf{v} \in \mathcal{F} - \{\mathbf{0}\}$ satisfying

$$\lambda \mathbf{v} = \mathbf{A}\mathbf{v}. \tag{4}$$

By taking the inner-product of each $\Phi(\mathbf{x}_k)$ with both sides of equation 4, we obtain the set of equations

$$\lambda\langle\Phi(\mathbf{x}_k), \mathbf{v}\rangle = \langle\Phi(\mathbf{x}_k), \mathbf{A}\mathbf{v}\rangle, \ \forall k \in \{1, \ldots, n\}. \tag{5}$$

In addition, all eigenvectors $\mathbf{v}$ that correspond with non-zero eigenvalues must lie in the span of $\{\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2), ..., \Phi(\mathbf{x}_n)\}$. Hence there exists a set of coefficients $\{\alpha_i\}_{i=1}^n$ such that

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i) \tag{6}$$

By substituting $\mathbf{A} = \sum_{ij} \varphi_{ij} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)^T$ and equation (6) into the set of equations (5), we obtain the following set of equations

$$\lambda \sum_{i=1}^{n} \alpha_i \langle\Phi(\mathbf{x}_k), \Phi(\mathbf{x}_i)\rangle = \sum_{i,j,l=1}^{n} \varphi_{ij}\alpha_l \langle\Phi(\mathbf{x}_k), \Phi(\mathbf{x}_i)\rangle\langle\Phi(\mathbf{x}_l), \Phi(\mathbf{x}_j)\rangle, \ \forall k \in \{1, \ldots, n\} \tag{7}$$

By stacking the set of n equations (7), we get the simplified matrix equation,

$$\lambda\mathbf{K}\overline{\alpha} = \mathbf{K}\overline{\varphi}\mathbf{K}\overline{\alpha}, \tag{8}$$

where $\mathbf{K}$ is an $n \times n$ matrix whose elements $\mathbf{K}_{ij} = \langle\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle$; $\overline{\alpha}$ is the column vector $\overline{\alpha} = [\alpha_1, ..., \alpha_n]^T$; and $\overline{\varphi}$ is an $n \times n$ matrix with $\overline{\varphi}_{ij} = \varphi_{ij}$. To find a solution of equation (8), we solve the eigenvalue problem

$$\lambda\overline{\alpha} = (\overline{\varphi}\mathbf{K})\overline{\alpha} \tag{9}$$

for nonzero eigenvalues. This will give us all solutions of equation (8) that we are interested in.

Let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ and $\{\overline{\alpha}^1, \overline{\alpha}^2, ..., \overline{\alpha}^n\}$ denote the eigenvalues and the corresponding eigenvectors of $\overline{\varphi}\mathbf{K}$ (i.e. the solutions of equation (9)), with λ_p being the first positive eigenvalue. Since the solution eigenvectors of equation (4) must be unit vectors, i.e $\langle\mathbf{v}_k, \mathbf{v}_k\rangle = 1$ for all $k \in \{p, \ldots, n\}$, we normalize $\mathbf{v}_k$:

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} \overline{\alpha}_i^k \Phi(\mathbf{x}_i)}{\sum_{i,j=1}^{n} \overline{\alpha}_i^k \overline{\alpha}_j^k \langle\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle}. \tag{10}$$

The nearest positive semidefinite matrix of $\mathbf{A}$ can be computed as

$$\hat{\mathbf{A}} = \sum_{i=p}^{n} \lambda_i \mathbf{v}_k \mathbf{v}_k^T. \tag{11}$$

In summary, to compute the nearest positive semidefinite matrix of $\mathbf{A}$ we follow the three steps: first, compute the product matrix $\overline{\varphi}\mathbf{K}$; second, compute its eigenvalues and corresponding eigenvectors, after which normalize the eigenvectors of $\mathbf{A}$; finally, compute the nearest positive semidefinite matrix as shown in equation (11).

3 Related Work

Metric learning has been an active research topic in the machine learning community. Here we briefly review some representative recent works on this topic. The method proposed in [7] is the most related prior work to our learning method since both methods aim to bring examples of the same class closer together and to separate examples of different classes. The main difference between the two methods is that we formulate our objective function as a quadratic semi-definite programming problem (QSDP) instead of a semidefinite program described in [7]. In addition, our proposed algorithm is able to incorporate many different kernel functions easily for non-linearly separable cases.

An earlier work by [6] also used semidefinite programming based on similarity and dissimilarity constraints to learn a Mahalanobis distance metric for clustering. The authors proposed a convex optimization model to minimize the sum of squared distances between example pairs of similar labels subject to a lower bound constraint on the sum of distances between example pairs of different labels. While [6] focuses on a global constraint which brings all similar examples closer together and separates dissimilar examples, our proposed method enforces local neighborhood constraints to improve the KNN classification performance.

Most recently, [1] proposed an information-theoretic approach to learning a Mahalanobis distance function. The authors formulate the metric learning problem as minimizing the LogDet divergence, subject to similarity and dissimilarity constraints. In addition to the difference in the objective function formulation, the number of constraints for our proposed optimization is only linear with respect to the number of training labeled examples. In contrast, the number of similarity and dissimilarity constraints of the optimization formulation in [1] is potentially quadratic in the number of labeled training examples.

In [5], the authors proposed an online learning algorithm for learning a Mahalanobis distance function which is also based on similarity and dissimilarity constraints on example pairs. The online metric learning algorithm is based on successive projections onto the positive semi-definite cone subject to the constraint that all similarly labeled examples have small pairwise distances and all differently labeled examples have large pairwise distances. Instead of using global similarity and dissimilarity constraints, our proposed method enforces local neighborhood constraints for each labeled training example.

In [2], Maximally Collapsing Metric Learning (MCML) algorithm is proposed to learn a Mahalanobis distance metric based on similarity and dissimilarity

constraints. The authors construct a convex optimization problem which aims to collapse all examples in the same class into a single point and push examples in other classes infinitely far apart. The main difference between our proposed method and MCML is also the usage of the local neighborhood constraints versus the pairwise constraints between similarly labeled and differently labeled examples.

In [3], the authors proposed Neighborhood Component Analysis (NCA) to learn a distance metric specifically for the KNN classifier. The optimization aims at improving the leave-one-out performance of the KNN algorithm on the training data. The algorithm minimizes the probability of error under stochastic neighborhood assignments using gradient descent. Although both NCA and our method try to derive a distance function in order to improve the performance of the KNN algorithm, we employed different optimization problem formulations.

In [8], the authors proposed a framework for similarity metric learning by the energy-based model (EBM) from a pair of convolutional neural networks that shared the same learning parameters. The cost function incorporated into the EBM aimed at penalizing large distances among examples with the same label, and small distance among example pairs of different classes. In our work, the distance function is parameterized by a transformation matrix instead of a convolutional neural network.

4 Experiments

In this section, we evaluate our metric learning algorithm (MLSVM) on a number of data sets from the UCI repository [22] and the LIBSVM data [23]. A summary of the data sets is given in Table 1.

In our experiments, we compare MLSVM against recent proposed metric learning methods for KNN classification, namely the Large Margin Nearest Neighbor [7] (LMNN) and Information-Theoretic Metric Learning (ITML) [1]. In addition, we also compare our method against a benchmark margin-based multiclass classification method, i.e SVM [24]. All the compared methods learn a new distance metric before applying KNN except SVM, which is directly used as a classification method. The parameters of the MLSVM and SVM, λ and the degree of the polynomial kernel, are selected from the sets $\{10^i\}_{i=-3}^{3}$ and $\{1, 2, 3, 4, 5\}$, respectively. For ITML, the trade-off parameter γ is also chosen from the set $\{10^i\}_{i=-3}^{3}$. For all metric learning algorithms, the K-nearest neighbor value is set to 4 which is also used in [1]. The parameters are selected using two fold cross validation on the training data. For all data sets, 50% of data is randomly selected for the training phase and the rest is used for test. The experiment is repeated for 10 random trials. The mean performance and standard error of various methods are reported for each data set.

In Figure 2, we present the performance of KNN, LMNN, ITML, and SVM against the new metric learning method MLSVM for the 9 data sets. From the

Table 1. A summary of the data sets

Data sets	Classes	Size	Features
australian	2	690	14
balance	3	625	4
dermatology	6	179	34
german	2	1000	24
heart	2	270	13
ionosphere	2	351	34
iris	3	150	4
spambase	2	2300	57
wine	3	178	13

Table 2. Performance summary by the three metric learning algorithms and SVM

Data sets	Best Performance	Best Models
australian	0.1683	MLSVM, ITML
balance	0.0047	MLSVM, SVM
dermatology	0.0244	MLSVM, SVM
german	0.2383	MLSVM, SVM
heart	0.1770	MLSVM, LMNN
ionosphere	0.0835	MLSVM
iris	0.0192	MLSVM, SVM
spambase	0.1023	MLSVM, LMNN, ITML
wine	0.0383	MLSVM, ITML

figure we clearly see that all three metric learning methods (MLSVM, ITML and LMNN) significantly improved the KNN classification accuracy by comparing with the first column, which is the raw KNN performance, with the only exception for LMNN on the "german" dataset. SVM, as a non-metric-distance based method also performed competitively. MLSVM is consistently among the best performing methods across all datasets. We also present the significance test results in Table 2. In Table 2, the second column is the best error rate achieved by any method for a dataset, and column 3 shows all the methods whose results are not statistically significantly different from the best result by pair-wise t-test at the 95% level. It is interesting to notice that MLSVM is the only model that always perform the best even comparing with the non-metric-learning SVM classifier. This is an evidence that KNN, as a simple model, can achieve competitive or superior classification performance compared with SVM, if a proper distance metric can be learned. In comparison with the other two metric-distance learning methods, namely LMNN and ITML, in our conducted experiments MLSVM is more reliable in improving KNN's performance, as LMNN and ITML are only within the best models for 2 and 4 times respectively among the 9 datasets.

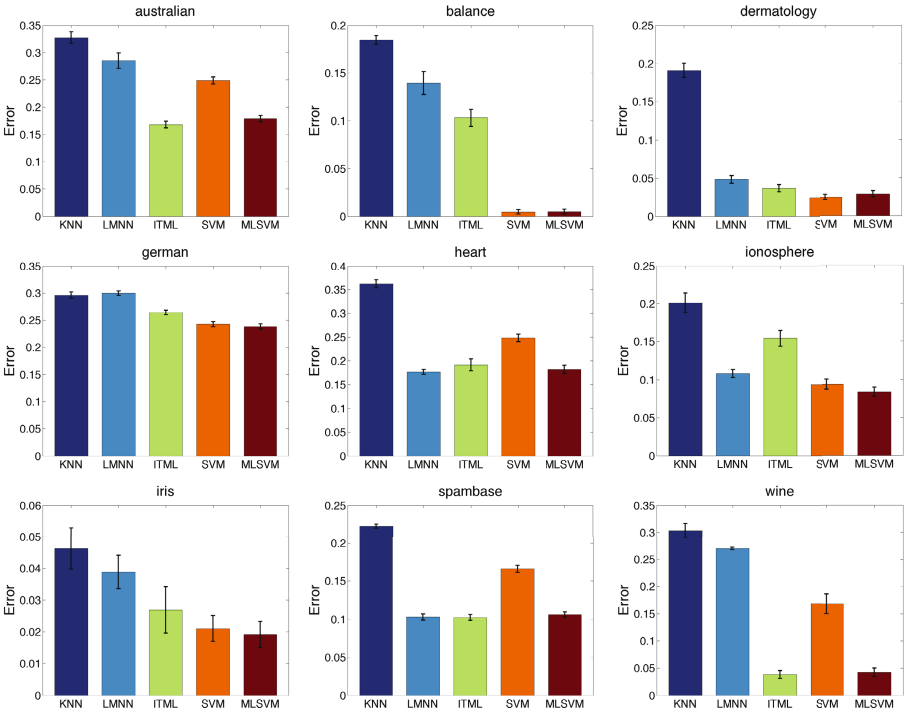

Fig. 2. Classification Error Rates of 6 different learning algorithms: KNN, LMNN, ITML, SVM, MLSVM for 9 data sets: wine, heart, dermatology, ionosphere, balance, iris, australian, german, and spambase

5 Conclusion

In this paper, we approached the metric learning problem utilizing the margin-based framework with the proposed method MLSVM. The problem is formulated as a quadratic semi-definite programming problem (QSDP) subject to local neighborhood constraints. We solve the optimization problem by iteratively relaxing the semidefiniteness constraint of the distance matrix and only approximate the closest semidefinite matrix after the gradient descent step. The SVM-based formulation allows MLSVM to incorporate various kernels conveniently. From the conducted experiments, MLSVM is able to produce a distance metric that helps improve this performance of the KNN classifier significantly. Our experiments also show that our metric learning algorithm always performs competitively and often better than the two other state-of-the-art metric learning approaches. Although KNN is arguably the most obvious application where metric distance learning plays an important role, we plan to adapt MLSVM to other interesting applications where a good distance metric is essential in obtaining competitive performances.

References

1. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Proceedings of the 24th International Conference on Machine learning, pp. 209–216 (2007)
2. Globerson, A., Roweis, S.: Metric learning by collapsing classes. In: Advances in Neural Information Processing Systems (NIPS) (2005)
3. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems (NIPS) (2004)
4. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: Advances in Neural Information Processing Systems (NIPS) (2004)
5. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudometrics. In: Proceedings of the 21st International Conference on Machine Learning (2004)
6. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems (NIPS) (2003)
7. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances ing Neural Information Processing Systems (NIPS) (2006)
8. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
9. Tsang, I.W., Kwok, J.T.: Distance metric learning with kernels. In: Proceedings of the International Conferences on Artificial Neural Networks (2003)
10. Shental, N., Hertz, T., Weinshall, D., Pavel, M.: Adjusting learning and relevant component analysis. In: Proceedings of the Seventh European Conference on Computer Vision (ECCV), pp. 776–792 (2002)
11. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
12. Domeniconi, C., Gunopulos, D., Peng, J.: Large margin nearest neighbor classifiers. IEEE Transactions on Neural Networks 16(4), 899–909 (2005)
13. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification and regression. In: Advances in Neural Information Processing Systems (NIPS), vol. 8, pp. 409–415. MIT Press, Cambridge (1996)
14. He, H., Hawkins, S., Graco, W., Yao, X.: Application of genetic algorithm and k-nearest neighbor method in real world medical fraud detection problem. Journal of Advanced Computational Intelligence and Intelligent Informatics 4(2), 130–137 (2000)
15. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2322 (2000)
16. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290 (2000)
17. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Chichester (1998)
18. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Primal estimated sub-gradient solver for SVM. In: Proceedings of the 24th International Conference on Machine Learning, pp. 807–814. ACM, New York (2007)
19. Joachims, T.: Making large-scale support vector machine learning practical. In: Schölkopf, B., Burges, C., Smola, A.J. (eds.) Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge (1998)

20. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A.J. (eds.) Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge (1999)
21. Schölkopf, B., Smola, A.J., Müller, K.R.: Kernel principal component analysis. Neural Computation 10(5), 1299–1319 (1998)
22. Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
23. Chang, C.C., Lin, C.J.: Libsvm data (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
24. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research 2, 265–292 (2001)