Learning Similarity Metric with SVM

Xiaoqiang Zhu*, Pinghua Gong*, Zengshun Zhao† and Changshui Zhang*
*State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Automation, Tsinghua University, Beijing 100084, China
†College of Information and Electrical Engineering, Shandong University of Science and Technology, Qingdao, China
{zhuxq05, gph08}@mails.tsinghua.edu.cn, zhaozengshun@gmail.com, zcs@mail.tsinghua.edu.cn

***Abstract*—In this paper, we show how to learn a good similarity metric for SVM classification. We present a novel approach to simultaneously learn a Mahalanobis similarity metric and an SVM classifier. Different from previous approaches, we optimize the Mahalanobis metric directly for minimizing the SVM classification error. Our formulation generalizes the traditional large margin principle used in standard SVM, that is, we maximize the *margin-radius-ratio*. The learned similarity metric significantly improves the classification performance of standard SVM. Empirical studies on real datasets show the proposed approach achieves higher or comparable classification accuracies compared with state-of-the-art similarity learning methods.**

I. Introduction

In many machine learning and data mining tasks, the raw input features are roughly collected. Due to the absence of prior knowledge, using Euclidean metric to measure the similarity among raw data points is a popular choice for these tasks. This often fails to generate discriminative representations for given problems. In this way, even state-of-the-art algorithms, such as k-nearest neighbor (KNN) and support vector machine (SVM), cannot achieve optimal performances [1] [2] [3] [4]. Hence, similarity learning, which aims to learn a data adaptive similarity metric to generate more discriminative representations for given problems, has drawn many researchers attention.

To learn a good similarity metric, several approaches from different aspects have been proposed. Generally speaking, these similarity learning approaches can be roughly divided into two categories.

The first category is to directly tune a Mahalanobis similarity metric from labeled examples in the feature space. Mahalanobis metric, which is a sufficiently powerful class of metrics that work on many real-world problems [1] [2] [3], generalizes the Euclidean metric by admitting arbitrary linear scalings and rotations of the feature space. Representative work include neighborhood component analysis (NCA) [5], large margin nearest neighbor (LMNN) [6], information-theoretic metric learning (ITML) [7], etc. These approaches optimize the Mahalanobis metric by exploiting partial empirical knowledge among data points. For example, they constrain the distances between data points in different classes to be larger than the distances between data points in the same classes. However, most of these approaches are designed for improving the KNN performance only.

The second category is kernel learning approaches. Unlike Mahalanobis similarity metric learning methods, these approaches implicitly learn a generalized similarity metric defined by the kernel function. They then design classification algorithms, e.g., SVM, that only depends on the kernel function. Well-performing approaches include kernel alignment [8], simpleMKL [9], radius based kernel learning (RKL) [4] and so on. However, most of these approaches are either restricted to learn the kernel function specified to a linear combination of several base kernels or limited to estimate the kernel matrix in a transductive setting and cannot naturally generalize to new data points [10].

In this paper, we present a new similarity metric learning scenario. We focus on the task of learning a good similarity metric for SVM classification. Inspired by the success of previous work, we propose a novel approach to learn a Mahalanobis similarity metric simultaneously with the training of SVM classifier. We optimize the Mahalanobis metric directly for minimizing the SVM classification error. It results in more discriminative representations of input data and significantly improves the classification performance of standard SVM. Besides, our approach is inductive and can be easily extend to the out-of-sample points.

Our formulation generalizes the traditional large margin principle used in standard SVM. We propose to maximize the *margin-radius-ratio*, that is, the ratio between the margin and radius of minimum enclosing ball (MEB). It is proved in [11] that the *margin-radius-ratio* bounds the estimation error of SVM, which denotes the gap between the expected error and the empirical error. Large *margin-radius-ratio* leads to small estimation error.

We learn the Mahalanobis similarity metric by estimating a linear transformation matrix in the feature space. This allows us to (1) learn a low-dimension projection matrix in high dimensional classification tasks which saves us considerable computational cost; (2) kernelize our algorithm to learn a nonlinear similarity metric.

II. Linear Similarity Metric Learning with SVM

A. SVM

Standard SVM aims to maximize the generalization ability of learned classifier [12]. It trains the classifier by employing the large margin principle.

Specifically, given dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in R^D$, $y_i \in \{+1, -1\}$, standard SVM [13] is to find an optimal classification hyperplane with the largest margin. The hyperplane is $\mathbf{w}^T\mathbf{x} + b = 0$, with $(\mathbf{w}, b)$ calculated by:

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

where parameter $p > 0$. $p = 1$ and $p = 2$ are two common choices, named L_1 loss and L_2 loss respectively [14].

The objective function in Eq. (1) is to maximize the margin (margin $\gamma = 1/\|\mathbf{w}\|_2$) as well as minimize the empirical error (defined by term $\sum_i \xi_i^q$). C is a tuneable tradeoff parameter.

Given an unlabeled data $\mathbf{x}$, the prediction output label is:

$$\hat{y} = \text{sign}\left(\sum_{i=1}^n \alpha_i y_i \underbrace{\mathbf{x}_i^T\mathbf{x}}_{} + b\right), \tag{2}$$

where $\alpha_i (i = 1, 2, ..n)$ are the dual variables and b is a bias. They can be obtained by solving the dual formulation of Eq. (1) using a quadratic programming solver.

Eq. (2) shows that the SVM classifier relies on the inner product (say, similarity[1]) between data points. Here we present a novel approach to learn an appropriate similarity metric for SVM to improve its performance.

B. Learning a linear similarity metric with SVM

1. Similarity Metric

Our approach learns a Mahalanobis similarity metric via a linear transformation L in the input space. Similarity among data points is defined as the inner product after applying the transformation L to input data:

$$S_L(\mathbf{x}_i, \mathbf{x}_j) = (L\mathbf{x}_i)^T(L\mathbf{x}_j) = \mathbf{x}_i^T L^T L \mathbf{x}_j.$$

Learning the transformation L is equivalent to learning the Mahalanobis matrix $M = L^T L$ [16]. In high dimensional classification tasks, such as face recognition and document classification, we can constrain L to be a $d \times D$ $(d \ll D)$ matrix, that is, a low-dimension projection matrix. This will save us considerable computational cost. Because directly estimating the Mahalanobis matrix M needs to learn $\frac{1}{2}D\times(D+1)$ parameters (M is a $D \times D$ symmetric matrix). In our approach learning the transformation L needs only $d \times D$ parameters.

2. Learn the metric by maximizing margin-radius-ratio

Embedding the above similarity metric into standard SVM, our approach simultaneously learns the linear transformation L and an SVM classifier $(\mathbf{w}, b)$:

$$\begin{aligned}(L^*, \mathbf{w}^*, b^*) = \arg\min_{(L,\mathbf{w},b)} \quad & \frac{1}{2}R(L)^2\|\mathbf{w}\|_2^2 + C\sum_{i=1}^n \xi_i^2, \\ s.t. \quad \forall i, y_i(\mathbf{w}^T L\mathbf{x}_i + b) \geq 1 - \xi_i, & \quad \xi_i \geq 0.\end{aligned} \tag{3}$$

[1]Treating inner product as the similarity measure is a popular approach to similarity-based classification [15].

Here $R(L)$ is the radius of minimum enclosing ball (MEB) of data in the transformed space. The MEB is the smallest ball that encloses all data points, whose radius $R(L)$ can be obtained by:

$$R(L) = \min_{y,c} \quad y, \qquad s.t. \quad \forall i, y \geq \|L\mathbf{x}_i - c\|_2. \tag{4}$$

The idea of Eq. (3) is to maximize the *margin-radius-ratio* in the transformed space, that is, the ratio between the margin and the radius of minimum enclosing ball. This is a generalization of traditional large margin principle. According to [11], the estimation error of SVM, which denotes the gap between the expected error and the empirical error, is bounded by the *margin-radius-ratio*: $\sqrt{O(R^2\gamma^{-2})/n}$. γ is the margin of SVM classifier. n is the number of data points. R is the radius of MEB of data. We can see that large *margin-radius-ratio* γ/R leads to small estimation error.

In standard SVM, the radius R of MEB is a constant and we can safely employ the large margin principle to minimize $\|\mathbf{w}\|_2^2$ (as well as the empirical error). But in our similarity learning scenario, the radius R changes with different learned transformations applied to input data. Large margin γ itself can not guarantee small estimation error. To get a good generalization performance, we propose to maximize the *margin-radius-ratio* γ/R directly. That is, to minimize $R^2/\gamma^2 = R^2\|\mathbf{w}\|_2^2$ (as well as the empirical error).

3. Margin maximization in unit-enclosing-ball

Directly solving Eq. (3) is difficult, which contains a complicated term $R(L)$ defined by Eq. (4). Here we derive an equivalent formulation which can tackle this difficulty.

We notice that in Eq. (4), c is the center of data points. Hence, with a preprocessing step to centralize the input data: $\mathbf{x}_i \longleftarrow \mathbf{x}_i - \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$, $R(L)$ can be easily calculated as:

$$R(L) = \max_i \ ||L\mathbf{x}_i||_2.$$

In the following, without special declaration the data are supposed to be centralized.

Now we can reformulate Eq. (3) as Eq. (5):

$$\begin{aligned}(L^*, \mathbf{w}^*, b^*) = \arg\min_{(L,\mathbf{w},b)} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^n \xi_i^2, \\ s.t. \quad \forall i, y_i(\mathbf{w}^T L\mathbf{x}_i + b) \geq 1 - \xi_i, & \quad \xi_i \geq 0, \\ & \forall i, \|L\mathbf{x}_i\|_2^2 \leq 1.\end{aligned} \tag{5}$$

Eq. (5) eliminates the complicated computation of the radius R of MEB from the objective and adds an *unit-enclosing-ball* constraint, which forces the radius R of MEB to be unit in the transformed space. Theorem 1 guarantees that these two equations have equivalent optimal solutions.

Theorem 1: Eq. (5) is equivalent to Eq. (3). That is to say, for any optimal solution obtained in Eq. (3), there exists an equivalent optimal solution in Eq. (5), and vice versa.

We defer the proof of Theorem 1 to Appendix A. Theorem 1 brings us an interesting view to see the formulation of Eq. (3) and Eq. (5). In Eq. (3) we maximize the *margin-radius-ratio*,

which explicitly considers the MEB of data in the objective. Theorem 1 says we can find an equivalent solution by maximizing the margin under an *unit-enclosing-ball* constraint. This leads to an easier optimization problem.

4. Mechanism of tuning the Mahalanobis similarity metric

The objective of Eq. (5) implies the mechanism of tuning the Mahalanobis similarity metric. Assume we "freeze" the classifier $(\mathbf{w}, b)$, then optimizing Eq. (5) is to find an optimal transformation L which minimizes the empirical error $\sum_{i=1}^{n} \xi_i^2$. On the other hand, by "freezing" the transformation L the optimal classifier $(\mathbf{w}, b)$ is obtained by margin maximization in the transformed space. In practice, the transformation and the classifier are optimized simultaneously, towards to minimizing of the SVM classification error directly.

5. Optimization

The problem of Eq. (5) is non-convex, since variables L and $\mathbf{w}$ couple together. We use the gradient-projection method [17] to solve it. By eliminating the variables ξ_i, Eq. (5) can be reformulated as Eq. (6).

$$\begin{aligned} (L^*, \mathbf{w}^*, b^*) = \arg\min_{(L,\mathbf{w},b)} \quad & f(L, \mathbf{w}, b), \\ s.t. \quad & \forall i, \|L\mathbf{x}_i\|_2^2 \leq 1, \end{aligned} \tag{6}$$

where

$$f(L, \mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \max\left[0, 1 - y_i(\mathbf{w}^T L\mathbf{x}_i + b)\right]^2.$$

5.1 Gradients computation

Differentiating $f(L, \mathbf{w}, b)$ with respect to the transformation matrix L and classifier $(\mathbf{w}, b)$ gives the following gradients:

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{w}} &= \mathbf{w} - 2C\sum_{i=1}^{n} \max\left[0, 1 - y_i(\mathbf{w}^T L\mathbf{x}_i + b)\right](y_i L\mathbf{x}_i), \\ \frac{\partial f}{\partial b} &= -2C\sum_{i=1}^{n} \max\left[0, 1 - y_i(\mathbf{w}^T L\mathbf{x}_i + b)\right] y_i, \\ \frac{\partial f}{\partial L} &= -2C\sum_{i=1}^{n} \max\left[0, 1 - y_i(\mathbf{w}^T L\mathbf{x}_i + b)\right](y_i \mathbf{w}\mathbf{x}_i^T). \end{aligned} \tag{7}$$

5.2 Updating with projection

We update the variables using gradient decent method with Armijo rule to search a suitable step size α.

$$\begin{aligned} \mathbf{w}^{t+1} &= \mathbf{w}^t - \alpha \frac{\partial f}{\partial \mathbf{w}}\Big|_{\mathbf{w}^t}, \\ b^{t+1} &= b^t - \alpha \frac{\partial f}{\partial b}\Big|_{b^t}, \\ L^{t+1} &= \left[L^t - \alpha \frac{\partial f}{\partial L}\Big|_{L^t}\right]^+. \end{aligned} \tag{8}$$

Here $[\cdot]^+$ means projection to the constrains, which is typically defined by Eq. (9):

$$[L]^+ = \arg\min_{A} \|A - L\|_F^2, \quad s.t. \quad \forall i, \|A\mathbf{x}_i\|_2^2 \leq 1. \tag{9}$$

The optimization problem in Eq. (9) is a Quadratic Constraint Quadratic Programming(QCQP) problem. In practice we find it converges slowly. Since the projection is a critical subroutine in the optimization procedure, it will be called for many times and greatly affects the convergence speed of the whole algorithm. To speed it up, we propose a heuristic projection strategy, defined by Eq. (10):

$$[L]^+ = \frac{L}{R}, \quad \text{where } R = \max_i \|L\mathbf{x}_i\|_2. \tag{10}$$

It is a straightforward projection by scaling L with respect to the radius R. We find this simple projection strategy really works experimentally. It leads to much faster convergence speed than using projection of Eq. (9).

6. Algorithm implementation

Algorithm 1 gives the implementation details. We call it **Linear-MSVM** (**Linear M**etric learning with **SVM**).

Algorithm 1: Linear-MSVM

Input : Data set: $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
A start point: $(L_0, \mathbf{w}_0, b_0)$.

Initialize: $\beta(0 < \beta < 1), \epsilon$(typically 10^{-6}) ;
Centralize the input data: $\mathbf{x}_i \longleftarrow \mathbf{x}_i - \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$;
for $t = 0, 1, 2, \cdots$ **do**
  Compute the gradient $\frac{\partial f}{\partial (L_t, \mathbf{w}_t, b_t)}$ by Eq. (7) ;
  for $m = 0, 1, 2, \cdots$ **do**
    $\alpha = \beta^m$;
    Update $(L_{t+1}, \mathbf{w}_{t+1}, b_{t+1})$ by Eq. (8) ;
    if *Armijo rule is satisfied* **then**
      break;
    end
  end
  if $|\frac{f(L_t, \mathbf{w}_t, b_t) - f(L_{t+1}, \mathbf{w}_{t+1}, b_{t+1})}{f(L_t, \mathbf{w}_t, b_t)}| < \epsilon$ **then**
    $L^* = L_{t+1}, \mathbf{w}^* = \mathbf{w}_{t+1}, b^* = b_{t+1}$, break;
  end
end

Output: $L^*, \mathbf{w}^*, b^*$

III. Nonlinear Similarity Metric Learning with SVM

In this section we show how to kernelize **Linear-MSVM** to learn a nonlinear similarity metric. We first map the input data into a kernel space F, then learn a linear transformation L in F. This is equivalent to learning a nonlinear similarity metric in the input space. Here kernel space F refers to the transformed space by a nonlinear kernel map $\phi : R^D \to F$.

We parameterize the linear transformation in kernel space to be $L = H\Phi_X$, where $\Phi_X = [\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_n)]^T$ and H is a weight matrix which allows us to parameterize L as the linear combination of the feature points $\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_n)$. Similar ideas can be found in previous work such as [18] and [10]. This kind of parameterization benefits us to easily compute the similarity in the transformed kernel space. Define

$$\begin{aligned} \mathbf{z}_i &= \Phi_X \phi(\mathbf{x}_i) \\ &= [\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_n)]^T \phi(\mathbf{x}_i) \\ &= [k(\mathbf{x}_1, \mathbf{x}_i), ..., k(\mathbf{x}_n, \mathbf{x}_i)]^T. \end{aligned}$$

Here $k(\cdot\ ,\cdot)$ is a kernel function,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j).$$

Hence,

$$L\phi(\mathbf{x}_i) = H\Phi_X \phi(\mathbf{x}_i) = H\mathbf{z}_i.$$

Similarity among data in the kernel space with the similarity metric defined by parameterized transformation L is:

$$\begin{aligned} S_L(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) &= (L\phi(\mathbf{x}_i))^T (L\phi(\mathbf{x}_j)) \\ &= (H\mathbf{z}_i)^T (H\mathbf{z}_j) \\ &= \mathbf{z}_i^T H^T H \mathbf{z}_j. \end{aligned}$$

A more inspiring result is that we can directly apply the Linear-MSVM algorithm to learn the nonlinear metric, by mapping input dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ to be $\{\mathbf{z}_i, y_i\}_{i=1}^n$, which is declared in the following proposition:

Proposition 1: We can directly apply the Linear-MSVM algorithm to learn a nonlinear metric with SVM by mapping the inputs using function $\mathcal{L} : \mathbf{x}_i \rightarrow \mathbf{z}_i$, $\mathbf{z}_i = \mathcal{L}(\mathbf{x}_i) = [k(\mathbf{x}_1, \mathbf{x}_i), ..., k(\mathbf{x}_n, \mathbf{x}_i)]^T$. *$x_i$ is an input point and z_i is the corresponding mapping point. $k(\cdot\ ,\cdot)$ is a kernel function.*

Proof: Since $L\phi(\mathbf{x}_i) = H\mathbf{z}_i$, treating $\mathbf{z}_i$ as the mapping point corresponding to input point $\mathbf{x}_i$ and H as the transformation matrix, it falls into the Linear-MSVM paradigm. That is to say, with the parameterized transformation L, we are in fact mapping the input dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ to be $\{\mathbf{z}_i, y_i\}_{i=1}^n$. The mapping function $\mathcal{L} : \mathbf{x}_i \rightarrow \mathbf{z}_i$ is defined as $\mathbf{z}_i = \mathcal{L}(\mathbf{x}_i) = \Phi_X \phi(\mathbf{x}_i) = [k(\mathbf{x}_1, \mathbf{x}_i), ..., k(\mathbf{x}_n, \mathbf{x}_i)]^T$. $k(\cdot\ ,\cdot)$ is some kernel function, e.g., RBF kernel.

To classify a new data $\mathbf{x}_q$, we first map it using function $\mathcal{L}$: $\mathbf{z}_q = \mathcal{L}(\mathbf{x}_q) = [k(\mathbf{x}_1, \mathbf{x}_q), ..., k(\mathbf{x}_n, \mathbf{x}_q)]^T$. then classify it using the rule $\hat{y} = \text{sign}(\mathbf{w}^T L\phi(\mathbf{x}_q) + b) = \text{sign}(\mathbf{w}^T H\mathbf{z}_q + b)$.

Hence, to learn a nonlinear metric in the input space, we need only first map the input dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ to be $\{\mathbf{z}_i, y_i\}_{i=1}^n$ in a parameterized space using the function $\mathcal{L}$, then apply the Linear-MSVM algorithm to learn a linear transformation in the parameterized space. ■

Algorithm 2: Nonlinear-MSVM

Input : Data set: $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
A kernel function $k(\cdot\ ,\cdot)$;
A start point H_0.

1 Map the input dataset by computing the Gram-Matrix K using $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and $k(\cdot\ ,\cdot)$;

2 Take $\{\mathbf{z}_i, y_i\}_{i=1}^n, H_0$ as inputs to compute $H^*, \mathbf{w}^*, b^*$ using Algorithm 1, $\mathbf{z}_i$ is the i-th column vector of K;

Output: $H^*, \mathbf{w}^*, b^*$

Now we present the nonlinear metric learning algorithm. Note that $\mathbf{z}_i (i = 1, 2, ...n)$ is the i-th column vector of the Gram-Matrix $K, K(i, j) = k(\mathbf{x}_i, \mathbf{x}_j)$. Hence, to learn a nonlinear metric in the input data space, we first pre-compute the Gram-Matrix K using the input data and a given kernel function $k(\cdot\ ,\cdot)$, then use Linear-MSVM algorithm to find the optimal similarity metric and the SVM classifier, taking $\{\mathbf{z}_i, y_i\}_{i=1}^n$ as inputs. Details are described in Algorithm 2, which we denote as **Nonlinear-MSVM**.

IV. Experiments

A. Experimental settings

In this section we test the classification performances of our proposed similarity metric learning algorithms on real world datasets. All the comparisons are divided into two groups:

- **Linear similarity metric methods:** As Linear-MSVM learns a linear similarity metric via a linear transformation directly in the input space, we compare it with two state-of-the-art Mahalanobis metric learning algorithms: NCA[2] and LMNN[3]. NCA learns a linear transformation as Linear-MSVM does, and LMNN estimates a Mahalanobis matrix in the input space. Besides, linear SVM is used as the baseline of the proposed Linear-MSVM.
- **Nonlinear similarity metric methods:** Our Nonlinear-MSVM algorithm estimates a nonlinear similarity metric by learning a linear transformation in the kernel space. We compare it with two state-of-the-art kernel learning methods SimpleMKL[4] and RKL[4]. Nonlinear SVM is also taken as the baseline of Nonlinear-MSVM.

All the experimental results are averaged over 10 runs. In each run, we randomly generate 70/30 splits of the data, with 70% training and 30% testing. To tune the parameters, we use 5-fold cross-validation in the training set.

For NCA and LMNN which use KNN classifiers for classification, we cross validate the number of neighbors K in $\{1, 3, 5, 7, 9\}$.

For SVM based methods (all above mentioned methods except NCA and LMNN), parameter C is cross-validated in $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$.

For Nonlinear-MSVM and Nonlinear SVM, we use RBF kernel function with parameter γ cross-validated in $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$.

For kernel learning methods SimpleMKL and RKL, 12 base kernels are used, including 7 RBF kernels with parameter $\gamma \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$ and 5 polynomial kernels with order $r \in \{1, 2, 3, 4, 5\}$.

NCA may trap into local optimums. Hence, we use three strategies for initializing of NCA: identity matrix, PCA transformation matrix and random matrix. We report the best results of NCA in three cases. Our proposed algorithms may trap into local optimums too. But in the next sub-section we will show that different initializations of our algorithms result in similar performances. Thus in all our experiments, we simply initialize the transformation L of our algorithms to be identity matrix.

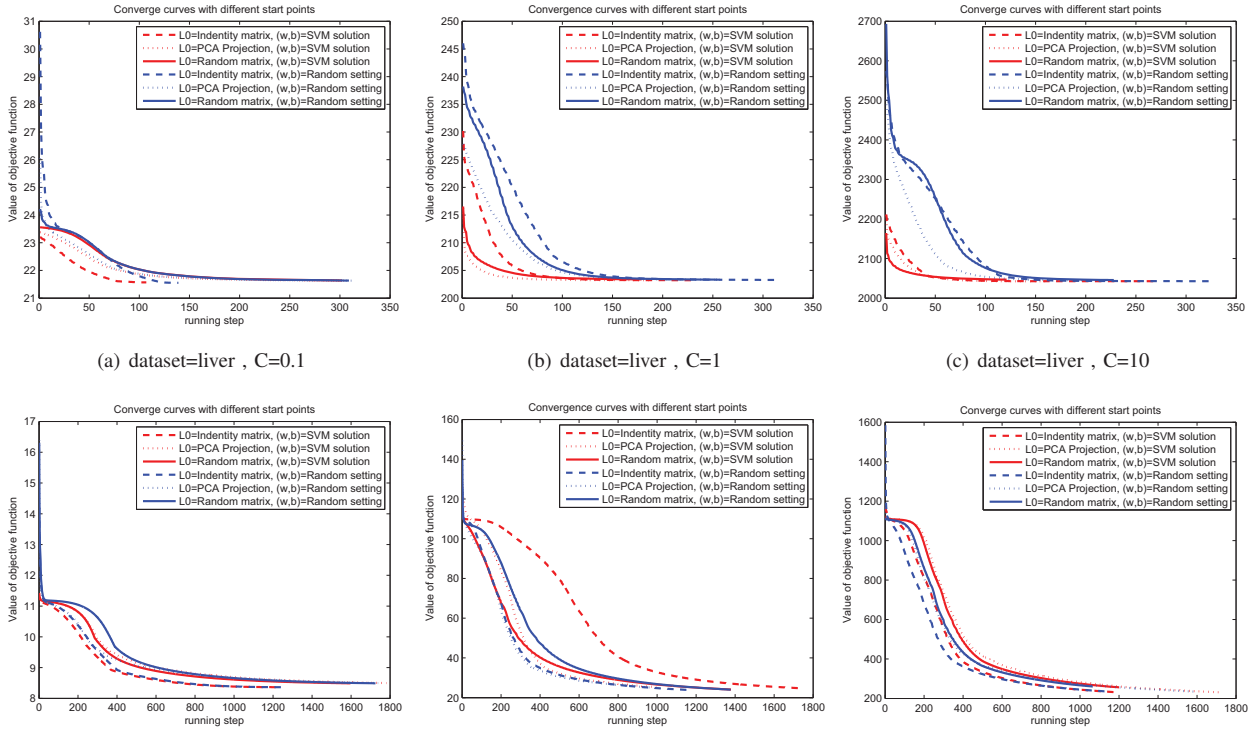
B. Influence of initialization

As pointed out in the optimization section, our proposed formulation is non-convex. We study experimentally the influence of different initializations on two real-world datasets:

[2]http://www.cs.berkeley.edu/∼fowlkes/software/nca/

[3]http://www.cse.wustl.edu/ kilian/Downloads/LMNN.html

[4]http://www.mloss.org



(a) dataset=liver , C=0.1 (b) dataset=liver , C=1 (c) dataset=liver , C=10

(d) dataset=TechTC-100(No.68) , C=0.1 (e) dataset=TechTC-100(No.68) , C=1 (f) dataset=TechTC-100(No.68) , C=10

Fig. 1. Convergence curves with different initializations. The top is liver dataset, and bottom is Tech-100(N0.68) dataset. Parameter C is the tradeoff coefficient in Eq. (5), which is set with three different values. It is observed that initialization has no significant influence on the convergence speed.

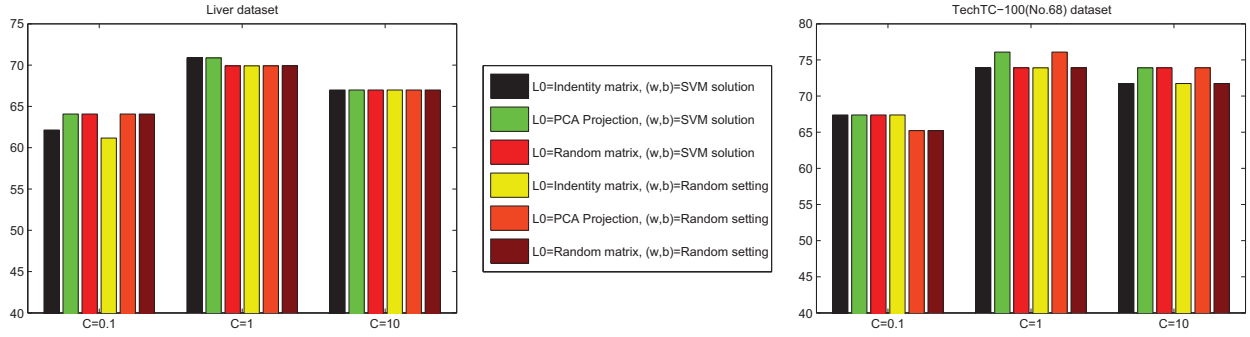


Fig. 2. Classification accuracies with different initializations. The left is liver dataset, and right is TechTC-100(N0.68) dataset. Parameter C is the tradeoff coefficient in Eq. (5). It is observed that initialization has no significant influence on the classification performance of the final convergent classifiers.

"liver" dataset (with low-dimensional data) and and "TechTC-100(N0.68)" document dataset (with high-dimensional data). See Table I and Table III for detailed properties of them.

In our proposed approaches, Linear-MSVM is the subroutine of Nonlinear-MSVM. Hence, we evaluate the performance of Linear-MSVM carefully with six simple and natural strategies for initialization:

(1) L_0=Identity matrix,$(\mathbf{w}_0, b_0)$=SVM solution[5];
(2) L_0=PCA projection matrix,$(\mathbf{w}_0, b_0)$=SVM solution;
(3) L_0=Random matrix,$(\mathbf{w}_0, b_0)$=SVM solution;
(4) L_0=Identity matrix,$(\mathbf{w}_0, b_0)$=Random vector;
(5) L_0=PCA projection matrix,$(\mathbf{w}_0, b_0)$=Random vector;
(6) L_0=Random matrix,$(\mathbf{w}_0, b_0)$=Random vector.

[5]By first applying transformation L_0 to input data: $\mathbf{x} \rightarrow L_0\mathbf{x}$ and then solving the standard SVM defined in Eq. (1).

The convergence curves under six initialization strategies are shown in Figure 1, and corresponding classification accuracies are shown in Figure 2. We can see that different initializations result in similar convergence speeds and classification accuracies. None of the six initializations significantly outperforms the others.

Thus in all our experiments, we simply set the initializations of our algorithms as: L_0=Identity matrix and $(\mathbf{w}_0, b_0)$=SVM solution.

C. Evaluation on UCI datasets

We evaluate the classification performances of our proposed algorithms on six low dimensional benchmark UCI datasets from UCI Machine Learning Repository[6]. These datasets are summarized in Table I. We learn a $D \times D$ transformation matrix L here.

TABLE I
PROPERTIES OF SIX BENCHMARK UCI DATASETS. ALL OF THEM ARE COLLECTED FOR BINARY CLASSIFICATION TASK.

	wpbc	wdbc	splice	pima	liver	ionosphere
Dimension	33	30	60	8	6	33
#Sample	194	569	1000	768	345	351

Table II shows the classification accuracies of both linear and nonlinear similarity metric learning methods.

(1) On all 6 datasets our proposed algorithms significantly outperform standard SVM (linear and nonlinear respectively). Moreover, on 4 out of 6 datasets, our Linear-MSVM defeats nonlinear SVM surprisingly. It indicates: a) the input features are not the most representable, as we argued, thus decrease the performance of standard SVM; b) with the learning of transformations in the feature space towards to minimizing the classification error, our approaches do find more discriminative representations of input data which provides helpful similarity metrics to improve the performance of standard SVM. Besides, we notice that on these datasets nonlinear SVM and linear SVM have similar performances, but our NonLinear-MSVM performs significantly better than Linear-MSVM. This shows our approach can properly capture the nonlinear structure of input on these datasets.

(2) Compared with state-of-the-art similarity learning methods, we observe that: a) in the linear metric group, our Linear-MSVM algorithm achieves higher accuracies on 4 datasets, and comparable accuracies on the left 2 datasets. b) in the nonlinear metric group, our NonLinear-MSVM algorithm achieves higher accuracies on all 6 datasets. Note that SimpleMKL and RKL use richer kernels than our NonLinear-MSVM but still are defeated by NonLinear-MSVM. We believe this is mainly attributed to our criteria of directly optimizing the Mahalanobis similarity metric for minimizing classification error.

D. Evaluation on document datasets

We also test our algorithms on higher dimensional document classification tasks with benchmark datasets from TechTC-100[7]. TechTC-100 is collected from real-world application, with 100 binary classification problems whose categorization difficulty (as measured by baseline SVM accuracy) is uniformly distributed. We randomly pick four of the most difficult ones, which we believe the inputs are not well representable. They are summarized in Table III.

[6]Available at http://archive.ics.uci.edu/ml/

[7]Available at http://techtc.cs.technion.ac.il/techtc100/

TABLE III
FOUR DATASETS FOR DOCUMENT CLASSIFICATION IN TECHTC-100. THE FIRST COLUMN LISTS THEIR ID NUMBERS.

TechTC-100	Dimension	#Sample
No.15	16384	153
No.24	17400	175
No.68	16248	158
No.87	20450	156

As the dimensionality of document data is high, here we learn a low-dimensional projection with a $d \times D$ matrix L for linear metric learning methods. D is the dimensionality of input space (in these tasks $D > 15,000$). d is the dimensionality of projection space (no more than 200 experimentally), which is calculated according to the energy percentage E those used dimensions hold: $E = \sum_{i=1}^{d} \lambda_i^2 / \sum_{i=1}^{D} \lambda_i^2$. λ_i corresponds to the i-th largest eigenvalue of the covariance matrix of inputs.

As LMNN can only learn a square Mahalanobis matrix in the input space, we project data into d-dimensional space using PCA for it. This is also done for linear SVM.

Kernel based methods have no explicit setting to learn a low-dimensional projection. We test their performance just with the original input data.

Figure 3 illustrates the classification results. As shown in the top row, our Linear-MSVM algorithm significantly outperforms state-of-the-art linear similarity metric methods (NCA, PCA+LMNN) and the baseline PCA+SVM-Linear. With dimensionality d decreasing, the classification performances of other three methods decrease rapidly, while our Linear-MSVM algorithm performs consistently well. This demonstrates again the effectiveness of our proposed algorithms.

On the other hand, our Nonlinear-MSVM algorithm performs better than SimpleMKL and RKL. As is shown in the bottom row of Figure 3, on 3 datasets our Nonlinear-MSVM algorithm gets better or comparable accuracies than SimpleMKL and RKL, and on TechTC-100(No.24) dataset it is defeated. However, we notice that all the nonlinear metric based methods, including our Nonlinear-MSVM, are inferior to our Linear-MSVM algorithm. A reasonable explanation is that for high dimensional and sparse document data, linear classifiers in the input space are good enough, with no need to map the data into higher dimensional kernel space.

Also, it should be pointed out that our proposed Linear-MSVM and Nonlinear-MSVM algorithms outperform standard linear SVM and nonlinear SVM respectively on all the four document datasets again.

V. RELATED WORK

We notice that an early work [19] proposed an algorithm to train an SVM classifier with scaling each feature of inputs. It is equivalent to learning a diagonal transformation matrix. Our proposed approach differs from it in two aspects: (1) We learn a more generalized transformation matrix. (2) They tune the metric by minimizing the SVM empirical error. Our approach directly minimizes the estimation error bound to optimize the generalization performance of SVM.

TABLE II

THE TESTING CLASSIFICATION ACCURACIES WITH STANDARD DEVIATIONS (IN PARENTHESES) ON UCI DATASETS. ALL THE COMPARISONS ARE DIVIDED INTO TWO GROUPS: THE FIRST GROUP LEARNS A LINEAR SIMILARITY METRIC IN THE INPUT SPACE DIRECTLY, THE SECOND LEARNS METRIC IN KERNEL SPACE, WHICH IS EQUIVALENT TO LEARNING A NONLINEAR SIMILARITY METRIC. IN EACH GROUP THE BEST RESULTS ARE SET TO BE BOLD.

data set	Linear similarity metric methods				Nonlinear similarity metric methods			
	NCA	LMNN	SVM-Linear	Linear-MSVM	SimpleMKL	RKL	SVM-NonLinear	NonLinear-MSVM
wpbc	75.01(5.22)	75.55(6.46)	75.38(6.16)	**81.25(3.44)**	77.03(1.23)	77.03(1.23)	75.90(2.00)	**84.17(3.16)**
wdbc	96.61(2.21)	97.08(0.92)	96.43(1.70)	**98.07(1.02)**	97.59(0.89)	98.30(0.70)	97.14(1.27)	**98.37(1.06)**
splice	74.00(2.58)	**83.77(1.15)**	79.639(1.66)	80.40(1.86)	87.78(1.61)	87.91(1.55)	83.77(2.03)	**89.07(1.80)**
pima	75.35(2.40)	74.56(2.62)	77.48(2.93)	**77.74(2.81)**	78.12(2.13)	78.51(2.35)	76.87(3.45)	**79.04(2.19)**
liver	67.66(3.93)	64.02(4.46)	69.12(1.89)	**70.18(2.01)**	66.71(3.01)	67.48(2.63)	69.58(3.99)	**73.65(2.16)**
ionosphere	89.66(2.71)	**90.12(3.17)**	86.89(2.07)	89.84(1.70)	94.79(2.17)	96.11(1.22)	93.83(2.12)	**96.20(0.63)**

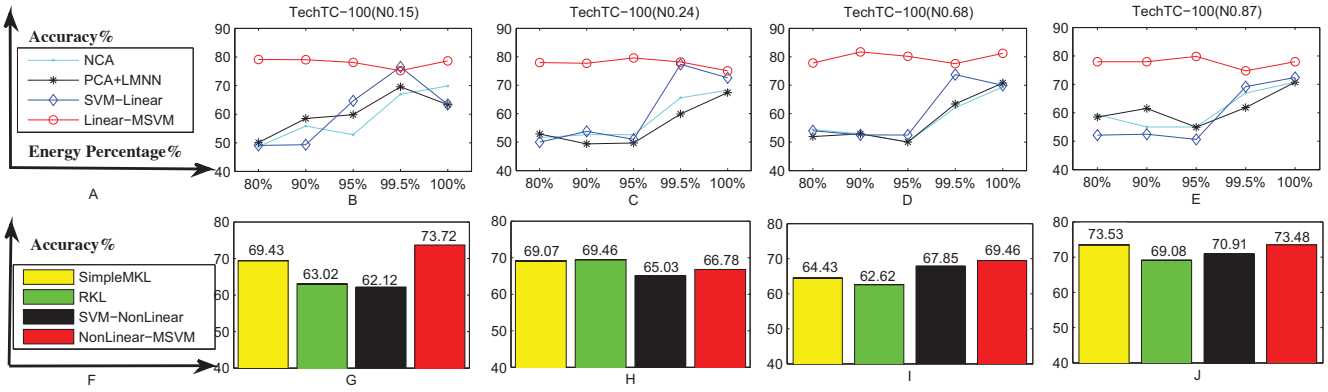


Fig. 3. Classification accuracies on four TechTC-100 datasets. Subplot A and F illustrates the legends of each row. 1) top row shows classification accuracies of linear similarity metric methods when projecting high dimensional document data into low dimensional(d-dim) spaces. d is the least number of dimensions which holds a given energy, i.e.: $E = \sum_{i=1}^{d} \lambda_i^2 / \sum_{i=1}^{D} \lambda_i^2$. λ_i corresponds to the i-th largest eigenvalue of the covariance matrix of input data. 2) bottom row shows classification accuracies of nonlinear similarity metric learning methods. On all four datasets, our Linear-MSVM gets the best performances.

A recently proposed kernel learning method [4] notes the effect of minimum enclosing ball varying in different kernel spaces. They learn a kernel function which is a linear combination of several given base kernels. In their method, *margin-radius-ratio* is used to measure the goodness of a learned kernel. To optimize the formulation, they propose a complicated multilevel optimization strategy. Our approach employs the principle of maximization of the *margin-radius-ratio* in a different setting. We inductively learn a Mahalanobis metric and an SVM classifier. Our optimization strategy is different from [4]. Instead of optimizing the complicated *margin-radius-ratio* directly, we derive an equivalent formulation by eliminating the radius of MEB in the objective. This leads to an easier optimization problem.

VI. CONCLUSION

In this paper, we propose a novel approach to simultaneously learn a Mahalanobis similarity metric and an SVM classifier. Our formulation generalizes the traditional large margin principle and proposes to maximize the *margin-radius-ratio*. We estimate the Mahalanobis metric by learning a linear transformation, which is optimized by minimizing the SVM classification error. Empirical study shows this criteria results in more discriminative representations of input data and significantly improves the performances of standard SVM (linear and nonlinear respectively). We also show how to kernelize the algorithm to learn a nonlinear similarity metric. Experiments on real world datasets show that our algorithms achieves better or comparable classification performances than those state-of-the-art similarity learning methods. For applications with low dimensional data, we suggest to use the Nonlinear-MSVM algorithm. For applications with high dimensional sparse data, such as document classification, our Linear-MSVM algorithm seems to be a better choice.

Our future work will concentrate on two directions: (1) to extend the proposed approach into multi-class classification tasks; (2) to speed up the proposed algorithms in high dimensional tasks by employing the sparsity of data.

ACKNOWLEDGMENT

This work is supported by NingBo Science and Technology Innovation Foundation (201001B7201008, 201001B7101013), Zhejiang Provincial Natural Science Foundation of China (No. Y1110661) and Visiting Scholarship Foundation for University Key Teacher by the Shandong Province of China (Grant No. ZR2010FM027).

REFERENCES

[1] A. Frome, Y. Singer, F. Sha, and J. Malik, "Learning globally-consistent local distance functions for shape-based image retrieval and classification," in *IEEE 11th International Conference on Computer Vision(ICCV)*, 2007.

[2] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *The Journal of Machine Learning Research(JMLR)*, vol. 11, pp. 1109–1135, 2010.
[3] M. Slaney, K. Weinberger, and W. White, "Learning a metric for music similarity," in *Proceedings of the 9th International Conference of Music Information Retrieval(ICMIR)*, 2008.
[4] K. Gai, G. Chen, and C. Zhang, "Learning Kernels with Radiuses of Minimum Enclosing Balls," in *Advances in neural information processing systems (NIPS)*, 2010.
[5] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems (NIPS)*, 2004.
[6] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems (NIPS)*, 2005.
[7] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning(ICML)*, 2007.
[8] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, "On kernel target alignment," *Innovations in Machine Learning(IML)*, pp. 205–256, 2006.
[9] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research(JMLR)*, vol. 9, pp. 2491–2521, 2008.
[10] I. D. Prateek Jain, Brian Kulis, "Inductive regularized learning of kernel functions," in *Advances in neural information processing systems (NIPS)*, 2010.
[11] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *Advances in neural information processing systems (NIPS)*, 2001.
[12] V. Vapnik, *The nature of statistical learning theory*. Springer Verlag, 2000.
[13] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning(ML)*, vol. 20, no. 3, pp. 273–297, 1995.
[14] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research(JMLR)*, vol. 9, pp. 1871–1874, June 2008.
[15] Y. Chen, E. Garcia, M. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based classification: Concepts and algorithms," *The Journal of Machine Learning Research(JMLR)*, vol. 10, pp. 747–776, 2009.
[16] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems (NIPS)*, 2003.
[17] D. Bertsekas, *Nonlinear programming*. Athena Scientific Belmont, 1999.
[18] L. Torresani and K. Lee, "Large margin component analysis," in *Advances in neural information processing systems (NIPS)*, 2007.
[19] Y. Grandvalet and S. Canu, "Adaptive scaling for feature selection in SVMs," in *Advances in neural information processing systems (NIPS)*, 2003.

APPENDIX A
PROOF OF THEOREM 1.

Proof: The proof consists of two steps: first, we prove that Eq. (3) is equivalent to Eq. (11); second, we show Eq. (11) can change into Eq. (5) by variable substitution.

$$\begin{aligned}(L^*, \mathbf{w}^*, b^*, R^*) = \underset{(L,\mathbf{w},b,R)}{\arg\min} \quad & \frac{1}{2}R^2\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i^2, \\ s.t. \quad & \forall i, y_i(\mathbf{w}^T L\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \\ & \forall i, \|L\mathbf{x}_i\|_2^2 \leq R^2.\end{aligned} \tag{11}$$

(1). Note that in Eq. (11), R becomes a variable, not the function of L anymore. However, if we can show $R^* = R(L^*)$ holds at the optimal point of Eq. (11), then the optimal solution of Eq. (11) is also the optimal solution of Eq. (3), and vice versa. In fact, it holds. To prove this, we only need to show that:

$$R^* = \max_i \ ||L^*\mathbf{x}_i||_2, \tag{12}$$

which means at the optimal point, the equality of the second constraint in Eq. (11) can be strictly met for some i.

This is true. Assume $(L_1^*, \mathbf{w_1}^*, b_1^*, R_1^*)$ is the optimal solution, but it doesn't satisfy Eq. (12), i.e.,$\forall i,\ ||L_1^*\mathbf{x}_i||_2 < R_1^*$. Denote: $r = \max_i \ ||L_1^*\mathbf{x}_i||_2$. Hence, $0 < r < R_1^*$. Construct another solution $(L_2^*, \mathbf{w_2}^*, b_2^*, R_2^*)$, where:

$$L_2^* = \frac{R_1^*}{r}L_1^*, \quad \mathbf{w}_2^* = \frac{r}{R_1^*}\mathbf{w}_1^*,$$
$$b_2^* = b_1^*, \quad R_2^* = R_1^*.$$

It is easy to verify that:

$$R_2^* = \max_i \ ||L_2^*\mathbf{x}_i||_2, \mathbf{w}_2^{*T}L_2^* = \mathbf{w}_1^{*T}L_1^*.$$

Hence, solution $(L_2^*, \mathbf{w_2}^*, b_2^*, R_2^*)$ satisfies the two constraints in Eq. (11). $\xi_i s$ in solution $(L_2^*, \mathbf{w_2}^*, b_2^*, R_2^*)$ are the same as in the solution $(L_1^*, \mathbf{w_1}^*, b_1^*, R_1^*)$ for all i.

Now let's check the value of objective function of Eq. (11), which we denote as f for brevity.
As

$$f_1^* = \frac{1}{2}(R_1^*)^2\|\mathbf{w}_1^*\|_2^2 + C\sum_{i=1}^{n}\xi_i^2,$$

and

$$\begin{aligned}f_2^* &= \frac{1}{2}(R_2^*)^2\,\|\mathbf{w}_2^*\|_2^2 + C\sum_{i=1}^{n}\xi_i^2 \\ &= \frac{1}{2}(R_1^*)^2\left\|\frac{r}{R_1^*}\mathbf{w}_1^*\right\|_2^2 + C\sum_{i=1}^{n}\xi_i^2 \\ &= \frac{1}{2}r^2\|\mathbf{w}_1^*\|_2^2 + C\sum_{i=1}^{n}\xi_i^2.\end{aligned}$$

Because $r < R_1^*$, hence $f_2^* < f_1^*$. This contradicts our assumption that $(L_1^*, \mathbf{w_1}^*, b_1^*, R_1^*)$ is an optimal solution. Thus at the optimal point, $R^* = \max_i \ ||L^*\mathbf{x}_i||_2$, i.e., $R^* = R(L^*)$. Hence we prove that Eq. (11) and Eq. (3) have the same optimal solutions.

(2). Now we just need to show that Eq. (11) is equivalent to Eq. (5). Denote:

$$\widetilde{\mathbf{w}} = R\mathbf{w}, \quad \widetilde{L} = \frac{1}{R}L.$$

Substitute $\widetilde{\mathbf{w}}, \widetilde{L}$ instead of $\mathbf{w}, L$ in Eq. (11), it is clearly that Eq. (11) turns to be Eq. (5), in which the variable R is eliminated. If $(L^*, \mathbf{w}^*, b^*, R^*)$ is the optimal solution of Eq. (11), then respectively $(\frac{1}{R^*}L^*, R^*, \mathbf{w}^*, b^*)$ is the optimal solution of Eq. (5).

According to (1) and (2), we have proved that Eq. (3) and Eq. (5) have equivalent solutions. That is, if $(L^*, \mathbf{w}^*, b^*)$ is the optimal solution of Eq. (3), then respectively $(\frac{1}{R(L^*)}L^*, R(L^*)\mathbf{w}^*, b^*)$ is the optimal solution of Eq. (5), and vice versa.

■