Neural Comput & Applic (2018) 30:2205–2215
https://doi.org/10.1007/s00521-016-2791-9

ORIGINAL ARTICLE

Improved least squares support vector machine based on metric learning

Dewei Li[1,2] · Yingjie Tian[2,3]

Received: 28 December 2015 / Accepted: 18 December 2016 / Published online: 29 December 2016
© The Natural Computing Applications Forum 2016

Abstract As two kinds of popular data mining methods, metric learning and SVM have a interesting and valuable internal relationship. The basic idea of metric learning is to learn a data-dependent metric, instead of Euclidean metric, to shrink the distances between similar points and extend the distances between dissimilar points. From a different view, LSSVM can reach a similar goal as metric learning. It finds two parallel hyperplanes to make the distances between points and corresponding hyperplane as small as possible and the distance between two hyperplanes as large as possible. LSSVM can be looked as a slack version of metric learning. Then, it can be improved by modifying the way in measuring between-class distance, lead to the raise of our novel approach ML-LSSVM, which adds constraints of inter-class distance into LSSVM. Alternating direction method of multipliers algorithm was implemented to solve ML-LSSVM effectively, much faster than handling the original quadratic convex programming problem. Experiments were made to validate the efficacy of ML-LSSVM and prove that different measurements of intra-class distance and inter-class distance have significant impact on classification. At last, the relation between LMNN and ML-LSSVM was discussed to illustrate that the local formulation of LMNN is equivalent to ML-LSSVM.

Keywords Metric learning · Least square-SVM · LMNN · Classification · Distance

1 Introduction

Support vector machines (SVMs) are very excellent algorithms in classification and regression problems which have shown state-of-the-art performance in a large number of applications [4, 6, 15, 16]. SVM is first proposed by Vapnik et al. [3, 19] twenty years ago and has always been a hot spot till today since its good generalization ability. Based on margin maximal principle, SVM aims to find two support hyperplanes, the distance between which can be expanded as large as possible. Later, least squares SVM (LSSVM) [14] was presented to convert the inequality constraints into equalities, leading to solving a system of linear equations. LSSVM seeks for two proximal hyperplanes for each class to make the distances between points and their corresponding proximal hyperplanes as small as possible. Meanwhile, LSSVM wishes to maximize the distance of the two hyperplanes. The solving speed of LSSVM is much faster than SVM because of avoiding handling quadratic programming problem, but its classification performance is slight worse than SVM [12, 13, 18].

As another popular research area, metric learning has attracted significant attention recently [1, 20, 24]. The aim of metric learning is to learn a data-dependent metric matrix M to redefine the distance between two points x_1, x_2 as

$$\sqrt{(x_1 - x_2)^{\mathrm{T}} M (x_1 - x_2)}$$

✉ Yingjie Tian
tyj@ucas.ac.cn

Dewei Li
lidewei15@mails.ucas.ac.cn

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

[2] Research Center on Fictitious Economy and Data Science, Chinese Academy of Sciences, Beijing 100190, China

[3] Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing 100190, China

In Euclidean distance, M is an identity matrix. The goal of the learned matrix is to expand inter-class distance and shrink intra-class distance. Classification can be easier implemented after metric learning, and the performance can be improved naturally [2, 5, 9, 10]. Much work related with metric learning has been done to improve the performance of SVMs [25, 27] and k-NN [8, 23, 26] algorithms. SVM can be improved by learning new RBF kernel [25] since RBF kernel function in nonlinear SVM is a function of distance. Large margin nearest neighbor (LMNN) classification [17, 21, 22] is viewed as a metric learning-based counterpart to SVM. It replaces linear classification in SVM with k-NN classification. MLSVM[11] constructs a new metric learning problem with local neighborhood constraints based on the formulation of SVM. PCML and NCML[28] are formulated as a kernel classification problem which can be solved by training SVMs iteratively.

In fact, SVM has close relationship with metric learning. In the view of metric learning, the margin maximal principle in SVM is equivalent to maximizing inter-class distance. But SVM can not minimize intra-class distance which had been promoted in [7]. ε-SVM has been introduced to improve SVM based on metric learning. It not only penalizes the points violating the margin, but also punishes the points far from the support hyperplanes. In this paper, we argue that LSSVM has closer relation with metric learning than traditional SVM. LSSVM makes the two proximal hyperplanes be proximal to their corresponding classes, which can be seen as minimizing within-class distance in metric learning perspective. Pulling the two proximal hyperplanes as far as possible is equivalent to maximizing between-class distance. The difference is that metric learning aims at learning a matrix M (full or sparse), while LSSVM only learns a vector w (or diagonal matrix W with $\mathrm{diag}(W) = w$). So LSSVM can be regarded as a slack version of metric learning algorithm. LSSVM can be improved by consolidating its formulation in measuring between-class distance. The $1/\|w\|^2$ term can not represent inter-class distance well since the two center hyperplanes have invaded the interior of each class. And it is just a regularized term in metric learning view. In traditional SVM, the principal of maximal margin is implemented, which seeks for two boundary hyperplanes, each one corresponds to a class, and maximizes the distance between the two hyperplanes. It can be regarded as expanding inter-class distance as much as possible in metric learning view. However, SVM does not make errors in shrinking intra-class distance, resulting in the claim that LSSVM is closer to metric learning than SVM.

In this paper, we will clearly analyze the relation between LSSVM and metric learning. Metric learning problem can be transformed as LSSVM by relaxing the pairwise constraints. In light of the advantages of metric learning in measuring between-class distance, we add constraints into the primary problems of LSSVM to control inter-class distance. ADMM algorithm is used to solve the new proposed approach, ML-LSSVM. ADMM can solve convex programming problem by breaking it into many smaller pieces and obtain the solution of each subproblem in much less time. Discussion on the relation between ML-LSSVM and LMNN is presented to show that the formulation of ML-LSSVM is similar with local LMNN.

We will organize the rest of the paper as follows. Background on metric learning and SVM is introduced in Sect. 2. The relation between LSSVM and metric learning, ML-LSSVM and LMNN will be discussed in Sect. 3, and Solving ML-LSSVM by ADMM will also be performed. In Sect. 4, we will make plenty of numerical experiments to compare our new method with several algorithms and verify the advantages of ML-LSSVM. Conclusions are made in Sect. 5.

2 Background

For a training set with c classes

$$T = \{(x_1, y_1), \ldots, (x_m, y_m)\}, \tag{2.1}$$

where $(x_i, y_i) \in R^n \times \{1, 2, \ldots, c\}, i = 1, \ldots, m$ and m is the total number of samples, n is the number of features. Define a index set $I = \{1, \ldots, m\}$ and the following pairwise data sets

$$S = \{(x_i, x_j) | y_i = y_j\} \tag{2.2}$$

$$D = \{(x_i, x_j) | y_i \neq y_j\} \tag{2.3}$$

where S contains data pairs with the same label and D contains data pairs with different labels.

2.1 Metric learning (ML)

The purpose of metric learning is to find a proper distance metric to measure the distance of all the data pairs instead of Euclidean metric. For a new learned distance metric $M \in R^{n\times n}$, the distance between two data points z_1 and z_2 in terms of M is represented by

$$d_M(z_1, z_2) = \sqrt{(z_1 - z_2)^{\mathrm{T}} M (z_1 - z_2)} \tag{2.4}$$

where M is a positive semi-definite matrix and satisfies the following properties

$$d_M(z_1, z_2) \geq 0 \tag{2.5}$$

$$d_M(z_1, z_2) = d_M(z_2, z_1) \tag{2.6}$$

$$d_M(z_1, z_2) + d_M(z_1, z_3) \geq d_M(z_1, z_2) \tag{2.7}$$

$$d_M(z_1, z_2) = 0 \Leftrightarrow z_1 = z_2 \tag{2.8}$$

In supervised metric learning, one of the most representative work is implemented by Xing with the paper [23], in which the method of pairwise constraints is proposed. The algorithm, formulated as a convex programming problem, aims to find a global data-dependent distance metric which minimizes the sum of distances between data pairs in (2.2) constrained by that data pairs in (2.3) are well separated. The following convex optimization problems are constructed

$$\min_{M} \sum_{(x_i,x_j)\in S} d_M^2(x_i, x_j) \tag{2.9}$$

$$\text{s.t.} \quad \sum_{(x_i,x_j)\in D} d_M^2(x_i, x_j) \geq 1, \tag{2.10}$$

$$M \succeq 0 \tag{2.11}$$

The above problem can been classified into two cases, the case of diagonal matrix M and full matrix M, which have been solved by Newton–Rapson method and gradient ascent with iterative projection, respectively.

2.2 Large margin nearest neighboor (LMNN)

LMNN learns a Mahanalobis distance metric to improve the performance of k-NN classification. Based upon the local view, LMNN defines a neighborhood for every point and aims to make the metric meet the goal that pull the samples in the neighborhood with the same label nearer and push the samples with different labels away by a large margin. The primal semi-definite programming problem is stated as follows

$$\min_{M,\xi} \sum_{il} \eta_{il} d_M^2(x_i, x_l) + C \sum_{ijl} \eta_{il}(1 - y_{jl})\xi_{ijl}, \tag{2.12}$$

$$\text{s.t.} \quad d_M^2(x_j, x_l) - d_M^2(x_i, x_l) \geq 1 - \xi_{ijl}, \tag{2.13}$$

$$\xi_{ijl} \geq 0, \quad M \succeq 0 \tag{2.14}$$

where

$$y_{jl} = \begin{cases} 1, & y_j = y_l, \\ 0, & y_i \neq y_l \end{cases}, \quad \eta_{il} = \begin{cases} 1, & y_{il} = 1, x_i \in t(x_l) \\ 0, & y_{il} = 0 \end{cases}$$

and $t(x_l)$ denotes the neighborhood of x_l. LMNN can make significant improvements on k-NN classification with Euclidean distance, verifying that the algorithm can obtain a better distance metric to measure the similarity of any two points.

2.3 Support vector machine (SVM)

SVM intends to seek the best hyperplane to classify the training set (2.1) by maximizing the margin between different classes. The primal problem of standard SVM

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i, \tag{2.15}$$

$$\text{s.t.} \quad y_i(w^{\mathrm{T}} x_i + b) \geq 1 - \xi_i, i \in I \tag{2.16}$$

$$\xi_i \geq 0, \quad i \in I \tag{2.17}$$

where $\xi = (\xi_1, \xi_2, \ldots, \xi_l)^{\mathrm{T}}$ and $C > 0$ is a penalty parameter. In the view of metric learning, SVM only makes efforts in separating different classes but ignores gathering the points in the same class. It is considered that SVM can be boosted by adding constraints of minimizing the within-class distance. So ε-SVM [7] is constructed as follows

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i + \lambda \sum_{i=1}^{m} \varepsilon_i, \tag{2.18}$$

$$\text{s.t.} \quad 1 + \varepsilon_i \geq y_i(w^{\mathrm{T}} x_i + b) \geq 1 - \xi_i, \tag{2.19}$$

$$\xi_i, \varepsilon_i \geq 0, \quad i \in I \tag{2.20}$$

which has double number of constraints than SVM, resulting in even lower speed than SVM.

2.4 Least square support vector machine (LSSVM)

In light of the idea of least squares, equality constraints are considered in SVM approach which lead to the presentation of LSSVM. The method can avoid solving quadratic programming by solving a set of linear equations. LSSVM searches for two parallel hyperplanes $w^{\mathrm{T}} x + b = 1, w^{\mathrm{T}} x + b = -1$ to minimize the distance between the data points and the corresponding hyperplane and maximize the margin of the two hyperplanes. The formulation of LSSVM algorithm is

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} e_i^2 \tag{2.21}$$

$$\text{s.t.} \quad y_i(w^{\mathrm{T}} x_i + b) = 1 + e_i, \quad i \in I \tag{2.22}$$

and the final decision hyperplane is $w^{\mathrm{T}} x + b = 0$. In fact, LSSVM makes two sides endeavors in adjusting the between-class and within-class distance. But its formation of measuring within-class and between-class distance are different from metric learning which will be promoted in later section.

The four methods have different forms in presenting within-class distance and between-class distance [14]. We list the formulas in Table 1. For within-class distance, we argue that

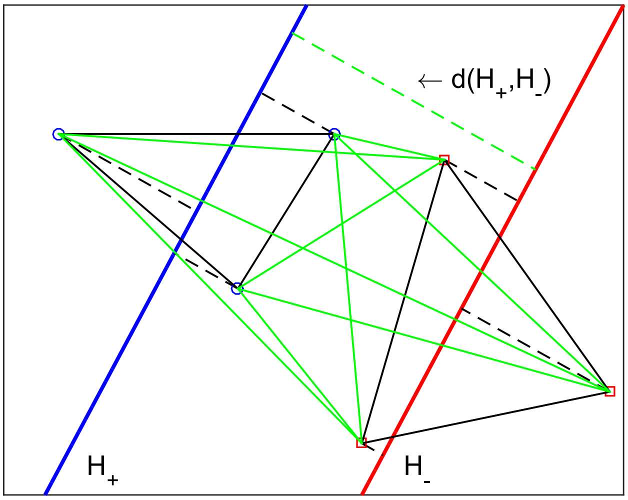
$$\sum_{y_i = y_j} d_M(x_i, x_j)$$

is more strict than

$$\sum_{k=\pm} \sum_{y_i = k} d(x_i, H_k)$$

Table 1 Representations of within-class distance and between-class distance

Algorithm	ML	LMNN	SVM	LSSVM
Within-class distance	$\sum_{y_i=y_j} d_M(x_i,x_j)$	$\sum_{y_i=y_j} d_M(x_i,x_j)$	N/A	$\sum_{k=\pm}\sum_{y_i=k} d(x_i,H_k)$
Between-class distance	$\sum_{y_i\neq y_l} d_M(x_i,x_l)$	$\min_{y_i\neq y_l} d(x_i,x_l)$	$\min_{y_i\neq y_l} d(x_i,x_l)$	$d(H_+,H_-)$



(a)

Fig. 1 The relation between LSSVM and metric learning. The *blue circles* belong to positive class, and the *red squares* belong to negative class. The points are mapped from primal space by the mapping: $\hat{x} = Wx$. The *black line* segments represent within-class distance and the *green ones* represent between-class distance. In these distances, *solid line* segments correspond to metric learning and *dotted line* segments corresponds to LSSVM. **a** LSSVM (color figure online)

in minimizing within-class distance. In a fixed space, minimizing $\sum_{y_i=k} d(x_i,H_k)$ can only make the distance of two points in the direction of perpendicular to H_k as short as possible without restricting the distance in the direction of parallel to H_k (Fig. 1). For between-class distance, $\min_{y_i\neq y_l} d(x_i,x_l)$ is considered to be more rigid than $\sum_{y_i\neq y_l} d_M(x_i,x_l)$. Maximizing the former can separate two different classes distinctly by a large margin, contributing to better classification performance, but the latter can not ensure it.

3 Metric learning-based-LSSVM

3.1 The relation between metric learning and LSSVM

In this subsection, the claim that LSSVM has a strong relation with metric learning will be proved as follows. For the metric learning problem (2.9)–(2.11), let $M = W^{\mathrm{T}}W$, where W is a diagonal matrix and $\mathrm{diag}(W) = w$, and define a linear transformation $R^n \to H : \hat{x} = Wx$. Given two fixed hyperplanes $H_+ : 1^{\mathrm{T}}\hat{x} + b = \sqrt{n}/2, H_- : 1^{\mathrm{T}}\hat{x} + b = -\sqrt{n}/2$

$$\begin{aligned}\sum_{(x_i,x_j)\in S} d_M^2(x_i,x_j) &= \sum_{(x_i,x_j)\in S} (x_i - x_j)^{\mathrm{T}} ww^{\mathrm{T}} (x_i - x_j)\\ &= \sum_{(x_i,x_j)\in S} (w^{\mathrm{T}}x_i - w^{\mathrm{T}}x_i)^2\\ &= \sum_{y_i=y_j=1} (1^{\mathrm{T}}Wx_i + b - \sqrt{n}/2\\ &\quad - (1^{\mathrm{T}}Wx_j + b - \sqrt{n}/2))^2\\ &\quad + \sum_{y_i=y_j=-1} (1^{\mathrm{T}}Wx_i + b + \sqrt{n}/2\\ &\quad - (1^{\mathrm{T}}Wx_j + b + \sqrt{n}/2))^2\\ &\leq n \sum_{y_i=y_j=1} (d(\hat{x_i},H_+) + d(\hat{x_j},H_+))^2\\ &\quad + n \sum_{y_i=y_j=-1} (d(\hat{x_i},H_-) + d(\hat{x_j},H_-))^2\\ &\leq 2n(p-1)\sum_{y_i=1} d^2(\hat{x_i},H_+)\\ &\quad + 2n(q-1)\sum_{y_j=-1} d^2(\hat{x_j},H_-)\\ &\leq t(\sum_{y_i=1} d^2(\hat{x_i},H_+) + \sum_{y_j=-1} d^2(\hat{x_j},H_-))\end{aligned} \tag{3.1}$$

where $t = 2n \cdot \max(p-1, q-1)$. In addition, we hope that the projected points of each class lie on two sides of corresponding hyperplane symmetrically. Then, we have

$$n\sum_{y_i=1} d^2(\hat{x_i},H_+) + n\sum_{y_j=-1} d^2(\hat{x_j},H_-) \leq \sum_{(x_i,x_j)\in S} d_M^2(x_i,x_j) \tag{3.2}$$

$$d^2(H_+,H_-) \leq \sum_{(x_i,x_j)\in D} d_M^2(x_i,x_j) \tag{3.3}$$

The Eqs. (3.2) and (3.3) are trivial results and we can explain them in Fig. 1.

From the Eqs. (3.1) and (3.2), it is notable that minimizing $\sum_{y_i=1} d^2(\hat{x_i},H_+) + \sum_{y_j=-1} d^2(\hat{x_j},H_-)$ in the H space is a slack way to minimize $\sum_{y_i=y_j} d_M^2(x_i,x_j)$ in R^n space. So the following optimization problems in H space are considered

$$\min_{W} \sum_{y_i=1} d^2(\hat{x_i},H_+) + \sum_{y_j=-1} d^2(\hat{x_j},H_-), \tag{3.4}$$

$$\text{s.t.} \quad d^2(H_+, H_-) = 1 \tag{3.5}$$

which can be rewritten as

$$\min \quad \sum_{i=1}^{m} e_i^2, \tag{3.6}$$

$$\text{s.t.} \quad y_i(w^{\mathrm{T}}x_i + b) = \sqrt{n}/2 + e_i, \quad i \in I \tag{3.7}$$

But the problem is scaling with $\|w\|$, we add a regularized term into the object function and standardize the constraints by dividing $\sqrt{n}/2$ on both sides. Then, LSSVM method is obtained

$$\min \quad \|w\|_2^2 + C\sum_{i=1}^{l} e_i^2, \tag{3.8}$$

$$\text{s.t.} \quad y_i(w^{\mathrm{T}}x_i + b) = 1 + e_i, \quad i \in I \tag{3.9}$$

It is considered that the $\|w\|$ term in the objective function of LSSVM is only a regularization term in metric learning view which can not maximize between-class margin well. In Fig. 2, H_+, H_- are the two hyperplanes that LSSVM seeking for and H_0 is the final decision hyperplane. There are two drawbacks in LSSVM: (1) The decision hyperplane is sensitive to outliers. In Fig. 2, there is an outlier in the top right corner which pull H_+ further from H_-, and the final decision hyperplane is not proper; (2) The between-class distance is measured by the distance between H_+ and H_-, $1/\|w\|$. But in metric learning view, between-class distance measured by $\min_{y_i \neq y_l} d_M^2(x_i, x_l)$ is more proper. So we can improve LSSVM by maximizing

$$\min_{y_i \neq y_l} \quad d_M^2(x_i, x_l)$$

which can be converted into maximizing the distance between two boundary lines (the two solid lines in Fig. 2).

Then, we can construct the following problems, termed as ML-LSSVM:

$$\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} e_i^2 - \lambda t, \tag{3.10}$$

$$\text{s.t.} \quad y_i(w^{\mathrm{T}}x_i + b - y_i) = e_i, i \in I \tag{3.11}$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq t, \quad i \in I \tag{3.12}$$

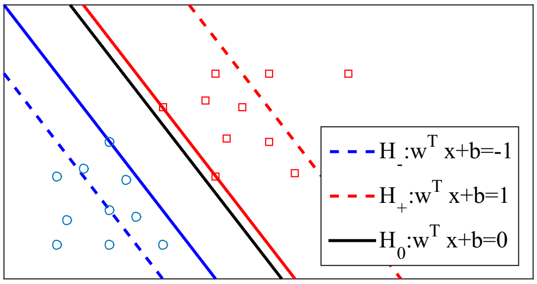


Fig. 2 Explanation for LSSVM and its drawbacks

In the above problem, maximizing t means that the points in two classes lie in two sides of H_0 and be far away from H_0 as much as possible. So the goal of maximizing between-class distance can be obtained.

3.2 Solving ML-LSSVM via ADMM

In this subsection, we try to solve the primary problem of ML-LSSVM in an effective way. The algorithm of ADMM (alternating direction method of multipliers) solves convex optimization problem by splitting them into many smaller scale optimization problems, each of which can be easier handled. It has been used in many applications recently. We will first transform our method into the standard formulation of ADMM and then introduce the solving process.

First, the slack variable $\eta = \{\eta_1, \ldots, \eta_m\} \leq 0$ is introduced to convert the inequations into equations, and then, we have

$$\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} e_i^2 - \lambda t, \tag{3.13}$$

$$\text{s.t.} \quad y_i(w^{\mathrm{T}}x_i + b - y_i) = e_i, i \in I \tag{3.14}$$

$$y_i(w^{\mathrm{T}}x_i + b) = t + \eta_i, \quad i \in I \tag{3.15}$$

$$\eta \geq 0 \tag{3.16}$$

Define an indicative function

$$h(\eta) = \begin{cases} +\infty, & \eta < 0, \\ 0, & \eta \geq 0 \end{cases} \tag{3.17}$$

and let $\pi = (w^{\mathrm{T}}, b, e^{\mathrm{T}}, t)^{\mathrm{T}}$, where $e = (e_1, \ldots, e_m)^{\mathrm{T}}$, we have the following optimization problem in matrix form

$$\min \quad \frac{1}{2}\pi^{\mathrm{T}}Q\pi + q^{\mathrm{T}}\pi + h(\eta), \tag{3.18}$$

$$\text{s.t.} \quad A_1\pi = 1_m, \tag{3.19}$$

$$A_2\pi - \eta = 0_m \tag{3.20}$$

where

$$Q = \begin{pmatrix} E & & & \\ & 0 & & \\ & & 2E & \\ & & & 0 \end{pmatrix}, \tag{3.21}$$

$$A_1 = (\,\mathrm{diag}(y)X \quad y \quad -E \quad \boldsymbol{0}_m\,), \tag{3.22}$$

$$A_2 = (\,\mathrm{diag}(y)X \quad y \quad O_{m\times m} \quad -\boldsymbol{1}_m\,), \tag{3.23}$$

$q^{\mathrm{T}} = (\boldsymbol{0}_{m+n+1}^{\mathrm{T}} - \lambda)$ and E is $m \times m$ identity matrix, $\mathbf{1}, \mathbf{0}$ are vectors of ones and zeros, respectively. For $X \in R^{m\times n}$, each row is a training instance. Then, we can get the following standard optimization problem for

$$\min \quad \frac{1}{2}\pi^{\mathrm{T}}Q\pi + q^{\mathrm{T}}\pi + h(\eta), \tag{3.24}$$

$$\text{s.t.} \quad A\pi + B\eta = c, \tag{3.25}$$

where $A = (A_1^{\mathrm{T}}\ A_2^{\mathrm{T}})^{\mathrm{T}}$, $B = (O_{m\times m}^{\mathrm{T}}\ -E)^{\mathrm{T}}$, $c = (\boldsymbol{1}^{\mathrm{T}}\ \boldsymbol{0}^{\mathrm{T}})^{\mathrm{T}}$. The problem (3.24)–(3.25) can be solved by Algorithm 1.

To extend our method into nonlinear case, the following kernel-based surface is considered

$$K(x, X)u + b = 0 \tag{3.26}$$

where $K(x, X) = (\Phi(x) \cdot \Phi(X))$. We can introduce (3.26) into the primal problem (3.10)–(3.12) directly and achieve the standard formulation for ADMM easily.

Algorithm 1 ADMM for ML-LSSVM

```
1. Given training set (2.1) and penalty parameters $C > 0, \lambda > 0$, learning rate $\rho$, iteration number $K$, stop criterion $\varepsilon_1, \varepsilon_2$; Given initial values $\eta^0, u^0$;
2. Set $k = 0$ and compute
     $\pi^{k+1} = (Q + \rho A^\top A)^{-1}(\rho A^\top(c - B\eta^k - u^k))$
     $\eta^{k+1} = (B^\top(c - A\pi^{k+1} - u^k))_+$
     $u^{k+1} = u^k + A\pi^{k+1} + B\eta^{k+1} - c$
3. Compute the primal residual and the dual residual at iteration $k + 1$
     $r^{k+1} = A\pi^{k+1} - B\eta^{k+1} - c$
     $s^{k+1} = \rho A^\top B(u^{k+1} + Bu^k)$
   If $\|r^{k+1}\| \le \varepsilon_1, \|s^{k+1}\| \le \varepsilon_2$, get the solution $(\pi^*, \eta^*) = (\pi^{k+1}, \eta^{k+1})$; Else go to step 2.
```

3.3 The relation on LMNN and ML-LSSVM

We will explore the relation between LMNN and ML-LSSVM in this subsection. The problems (2.12)–(2.14) can be generalized as

$$\min \quad \sum_{il}\eta_{il}d_M^2(x_i, x_l) - \lambda\sum_{l}\gamma_l, \tag{3.27}$$

$$\text{s.t.} \quad d_M^2(x_j, x_l) - d_M^2(x_i, x_l) \ge \gamma_l, \tag{3.28}$$

$$M \succeq 0 \tag{3.29}$$

which can be rewritten as

$$\min \quad \sum_{il}d_M^2(x_i, x_l) - \lambda\sum_{l}\gamma_l, \tag{3.30}$$

$$\text{s.t.} \quad d_M^2(x_i, x_l) = e_i^2, \quad \forall l, x_i \in t(x_l) \tag{3.31}$$

$$d_M^2(x_j, x_l) \ge \gamma_l + \max_i d_M^2(x_i, x_l), \tag{3.32}$$

$$\forall x_l, x_i \in t(x_l), \quad y_j \ne y_l, \quad M \succeq 0 \tag{3.33}$$

where $\lambda > 0$ is an adaptive parameter.

For every x_l, the above problem can be broken up into

$$\min \quad \sum_{i}d_M^2(x_i, x_l) - \lambda\gamma_l, \tag{3.34}$$

$$\text{s.t.} \quad d_M^2(x_i, x_l) = e_i^2, \quad x_i \in t(x_l) \tag{3.35}$$

$$d_M^2(x_j, x_l) \ge \gamma_l + \max_i d_M^2(x_i, x_l), \tag{3.36}$$

$$y_j \ne y_l, M \succeq 0 \tag{3.37}$$

Similar as [7], we introduce a nonlinear transformation

$$\tilde{x} = \Phi(x) = (x_1^2, \ldots, x_n^2, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n, x_1, x_2, \ldots, x_n) \tag{3.38}$$

then $d_M^2(x, x_l) = w_l^{\mathrm{T}}\Phi(x) + b_l$. The problem (3.34)–(3.37) can be transformed as

$$\min \quad \sum_{i}(e_i^2 - \gamma_l/2) - \lambda\gamma_l/2, \tag{3.39}$$

$$\text{s.t.} \quad w_l^{\mathrm{T}}\Phi(x_i) + b_l' + 1 = 1 - \max_i d_M^2(x_i, x_l) + e_i^2 - \gamma_l/2, \tag{3.40}$$

$$w_l^{\mathrm{T}}\Phi(x_j) + b_l' \ge \gamma_l/2, \tag{3.41}$$

$$y_j \ne y_l, x_i \in t(x_l) \tag{3.42}$$

where $b_l' = b_l - \max_i d_M^2(x_i, x_l) - \gamma_l/2$.

The formulation of problem (3.39)–(3.41) is equivalent to ML-LSSVM, except their different training sets. Given any x_l, the training set contains two classes, one class consists of the points located in the neighborhood of x_l that own the same label with x_l and the other class consists of the points with different labels from x_l. We denote the two classes by the sets TS_s and TS_d. The problem minimizes the within-class distance by forcing the data in TS_s to be as near as possible from x_l and maximize between-class distance by making the margin as large as possible. It can be seen that LMNN contains m local ML-LSSVM with constraining that every w_l is dependent on the corresponding x_l. So local information is embedded in LMNN, but ML-LSSVM utilizes global information to obtain the best hyperplane. The relation between LMNN and ML-LSSVM can be explained in Fig. 3.

4 Numerical experiments

In this section, numerical experiments in different aspects will be made to evaluate the ability of ML-LSSVM. The experimental design, including the selected datasets and compared algorithms, the parameters to be optimized, will be clearly specified. We will introduce a toy example first to show that our method can reduce the negative impact of outliers and scale within-class and between-class distance properly. Then, classifications on binary class and multi-class are implemented and the CPU time is compared.

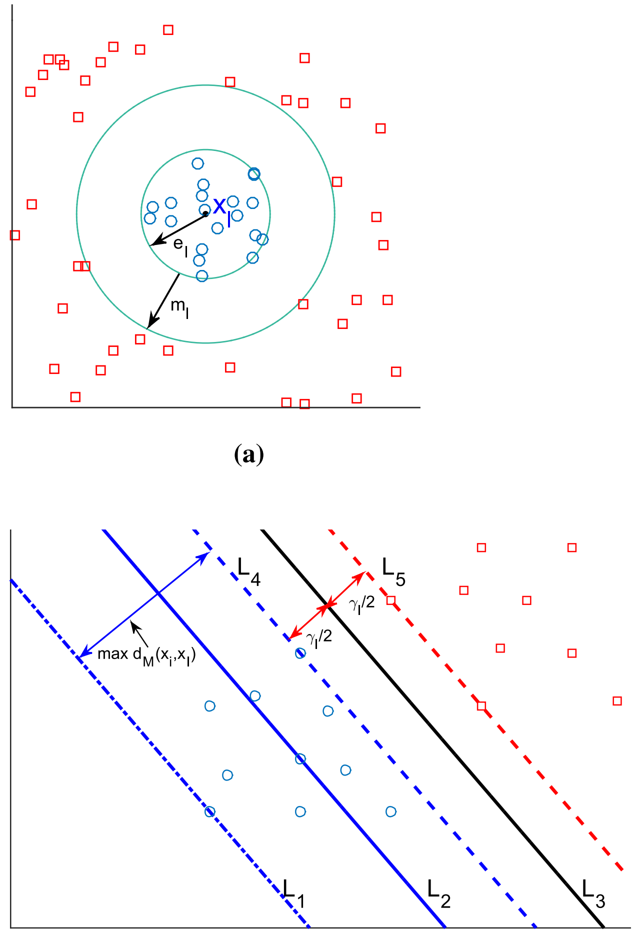
4.1 A toy example for ML-LSSVM

Given an artificial training set: negative class

$$N = \{(0.5, 0.5), (0.7, 0.3), (0.3, 0.7), (1, 1), (1.4, 0.6), (0.6, 1.4)\}$$

and positive class

$$P = \{(2, 2), (2.8, 1.2), (1.2, 2.8), (2.5, 2.5), (3.5, 1.5), (1.5, 3.5)\}$$



(a)

(b)

Fig. 3 The relation between local LMNN (**a**) and ML-LSSVM (**b**). **a** The points are in the original space. For every x_l, the *inner circle* denotes the extent of its class and the radius is $e_l = \max_i d_M^2(x_i, x_l)$. The margin $m_l = \sqrt{\gamma_l + \max_i d_M^2(x_i, x_l)} - e_l$. Local LMNN aims to minimize e_l and maximize γ_l. **b** The points are in the mapped space: $\tilde{x} = \Phi(x)$. $\mathrm{L}_1 : w_l^{\mathrm{T}}\Phi(x) + b_l = 0$; $\mathrm{L}_2 : w_l^{\mathrm{T}}\Phi(x) + b_l' = -1$; $\mathrm{L}_3 : w_l^{\mathrm{T}}\Phi(x) + b_l' = 0$. Minimizing $\sum_i e_i^2$ in the Eq. (3.39) is equivalent to making the distances between *blue points* and L_2 as small as possible. Maximizing γ_l is to extend the distance between the two boundary lines as much as possible ($\gamma_l/2$ is equals to t in Eq. (3.12)) (color figure online)

The former three points in N are in the line $x1 + x2 = 1$ and the last three in $x1 + x2 = 2$. The former three points in P are in the line $x1 + x2 = 4$ and the last three in $x1 + x2 = 5$. So the best separating hyperplane should be $x1 + x2 = 3$ and the two corresponding center lines are $x1 + x2 = 1.5$, $x1 + x2 = 4.5$.

If adding an outlier (5, 5) into P, LSSVM will get two center lines $x1 + x2 = 0.23$, $x1 + x2 = 6.38$ (the blue and red dotted lines in Fig. 4) and the decision line $x1 + x2 = 3.3$ (the black dotted line in Fig. 4), but ML-LSSVM can get $x1 + x2 = 1.6$, $x1 + x2 = 4.4$ (the blue and red solid lines in Fig. 4) and the decision line $x1 + x2 = 3$ (the green solid line in Fig. 4). In fact, the two lines $x1 + x2 = 0.23$, $x1 + x2 = 6.38$ have not minimized the within-class distance owing to the outlier. Also $1/\|w\|^2$ loses its meaning in representating the between-class distance. But ML-LSSVM can eliminate the effect of the outlier by adjusting the parameter λ. Besides, the two center lines of ML-LSSVM play the role of minimizing within-class distance well.

4.2 Datasets information and experimental setups

Benchmark datasets were selected to evaluate the classification performance. We selected 20 binary class datasets and 9 multi-class datasets from UCI Machine Learning Repository and the LIBSVM DATASETS. The characteristics of the 29 datasets, including the number of instances, features and classes, are displayed in Table 2. All the datasets are scaled into the interval [0, 1]. The instance numbers of the datasets are from 62 to 1000 and the feature numbers are from 4 to 7129.

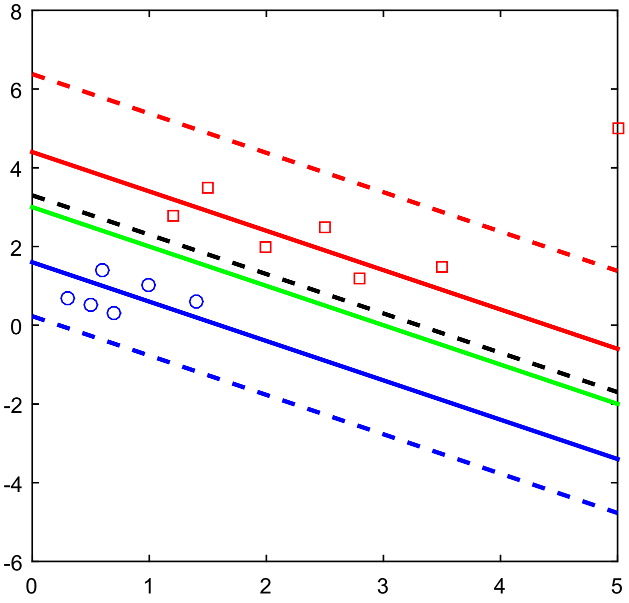


Fig. 4 A toy example for ML-LSSVM

Table 2 Characteristics of selected datasets

Category	Datasets	Instance	Feature	Class	Datasets	Instance	Feature	Class
Binary class	Coloncancer	62	2000	2	Ionosphere	351	34	2
	Leuke	72	7129	2	Dermatology	366	34	2
	Hepatitis	155	19	2	Votes	435	16	2
	WPBC	198	33	2	Arrhythmia	452	279	2
	Sonar	208	60	2	Clean1	476	166	2
	Spectf	267	44	2	WDBC	569	30	2
	Heart	270	13	2	Australian	690	14	2
	Hungarian	294	13	2	Blood	748	4	2
	Heartc	303	13	2	Pima	768	8	2
	Bupa-liver	345	6	2	German	1000	20	2
Multi-class	Circle	200	2	4	Libras	360	90	15
	Iris	150	4	3	Gem	712	3	4
	Wine	178	13	3	Vehicle	846	18	4
	Seeds	210	7	3	Vowel	528	10	11
	Thyroid	215	5	3				

We compared our method with three methods, ε-SVM, LSSVM, LMNN, using classification error to evaluate performance. ADMM algorithm was also implemented in the former two methods. All the experiments were made on MATLAB 2015a(Inter Core i5, 4G RAM).

In binary classification, to test the performance of ML-LSSVM comprehensively, the four algorithms except LMNN were implemented in four kernels: linear, polynomial, RBF and sigmoid. All the best parameters were selected with fivefold cross-validation. For all the kernels, C is searched from the set $\{10^{-3},\ldots,10^{3}\}$. In polynomial kernel $K(u,v)=(u'v+1)^{d}$, the degree d is selected from $\{2,3,4\}$. In RBF kernel $K(u,v)=exp(-\gamma\|u-v\|^{2})$ and Sigmoid kernel $K(u,v)=tanh(ku'v+1)$, the parameters γ and k are both chosen from $\{10^{-2},\ldots,10^{2}\}$. For LMNN, only experiments in linear case were made since the method can not be extended into nonlinear case directly. For $\varepsilon-SVM$, λ is set to be $C/3$ and in ML-LSSVM, $\lambda=2C$, both settings mean to penalize the points violate inter-class distance constraints more than intra-class distance constraints.

In multi-class classification, we compared the four methods in three cases: linear kernel, polynomial kernels with $d=2$ and $d=3$. Similarly, LMNN is only performed in linear kernel. The settings of C,λ are the same as that in binary classification. One versus One algorithm was used in ε-SVM, LSSVM, ML-LSSVM.

4.3 Experimental results and analysis

The average error rates in binary classification are displayed in Table 3 and the best results are in bold-face. In 20 comparisons of each kernel, ML-LSSVM got 13, 12, 14,12 times best performance in linear, polynomial, RBF, sigmoid kernel, respectively, which demonstrate that ML-LSSVM can handle binary classes task effectively and the improvements on LSSVM are advantageous. LMNN performed worst in linear case. In Fig. 5, The CPU time comparisons were processed on ε-SVM, LSSVM, ML-LSSVM. The datasets are ranked in ascend order of the number of instances. For each subfigure in Fig. 5, the horizontal axis is from 1 to 20, denotes the rank of each dataset. It is notable that LSSVM is the fastest method, and ML-LSSVM is slower than LSSVM slightly. For the running speed in linear, RBF and sigmoid kernel, LSSVM is three times faster than ML-LSSVM. In polynomial kernel, the speed of ML-LSSVM is as high as LSSVM. But ML-LSSVM performed much better than LSSVM. ε-SVM is much slower than LSSVM and ML-LSSVM since its constraints in the primary problems are four times as much as LSSVM.

Figure 6 shows the average error rates in multi-class classification. In linear case, ML-LSSVM performed best in 4 of 9 and LMNN took the first place in three times. And in polynomial kernel with two degrees, ML-LSSVM got 15 best results of 18. ε-SVM performed second after ML-LSSVM.

The performance of ML-LSSVM verify that within-class and between-class distance are very important in classification tasks and the way in measuring the two types of distance can affect prediction results markedly. The relation on metric learning and LSSVM is helpful in making improvements on LSSVM.

5 Conclusions

In this paper, we explore the relation between metric learning and LSSVM. LSSVM can be regarded as a slack version of the method with pairwise constraints, which is

Table 3 Error rate of binary classification in linear and nonlinear kernel

Dataset	Linear				Poly		
	ε-SVM	LSSVM	ML-LSSVM	LMNN	ε-SVM	LSSVM	ML-LSSVM
Coloncancer	19.52	19.52	**16.19**		17.62	**11.19**	13.33
Leuke	2.86	2.86	**1.43**		1.43	1.43	**1.25**
Hepatitis	16.13	16.13	**14.84**	17.42	19.35	19.35	**12.90**
WPBC	21.76	21.72	**19.78**	26.15	20.29	**20.26**	20.29
Sonar	28.74	27.79	**21.59**	15.86	18.66	19.63	**18.17**
Spectf	20.58	**19.82**	20.60	20.59	25.48	23.96	**23.26**
Heart	17.41	16.67	**15.19**	22.22	**17.78**	19.63	18.15
Hungarian	**17.58**	17.90	17.66	21.78	17.94	18.61	**17.71**
Heartc	16.14	16.17	**15.14**	20.14	**16.48**	20.14	17.49
Bupa	31.30	31.88	**30.72**	34.20	26.38	26.09	**25.51**
Ionosphere	17.09	**11.95**	11.96	15.11	11.97	15.68	**10.82**
Dermatology	3.83	2.73	**2.46**	3.00	3.27	3.55	**3.00**
Votes	**4.37**	5.29	5.29	5.06	4.14	4.60	**4.14**
Arrhythmia	24.33	23.46	**22.14**	28.97	36.25	43.12	**34.94**
Clean1	**15.76**	18.07	17.85	9.88	22.27	30.87	**21.00**
WDBC	6.50	3.70	**3.69**	4.20	4.05	3.52	**3.14**
Australian	14.35	**13.77**	13.91	16.38	**12.32**	13.48	13.62
Blood	23.14	22.73	**22.59**	29.02	**19.80**	20.74	20.99
Pima	30.47	**21.88**	22.67	25.90	**22.28**	22.28	22.53
German	25.3	24.40	**23.50**	29.30	36.90	**25.70**	28.30

Dataset	Rbf			Sigmoid		
	ε-SVM	LSSVM	ML-LSSVM	ε-SVM	LSSVM	ML-LSSVM
Coloncancer	14.52	17.62	**14.29**	35.48	**35.24**	35.48
Leuke	**20.71**	30.18	33.04	34.82	34.46	**34.46**
Hepatitis	17.42	20.81	**14.19**	14.84	18.71	**14.19**
WPBC	19.30	20.81	**17.77**	23.33	20.66	**20.22**
Sonar	14.30	16.25	**11.51**	22.49	23.96	**21.55**
Spectf	18.72	**14.97**	18.35	20.58	20.56	**20.56**
Heart	15.56	15.56	**15.19**	16.30	16.30	**15.56**
Hungarian	18.65	18.24	**17.39**	**18.63**	19.37	18.74
Heartc	15.51	16.49	**15.14**	15.49	16.16	**15.46**
Bupa	28.12	26.09	**26.09**	28.99	29.57	**28.99**
Ionosphere	5.41	4.56	**4.56**	**11.67**	13.09	12.24
Dermatology	2.46	2.46	**2.18**	**2.45**	2.46	2.72
Votes	**4.14**	**4.14**	4.37	4.37	4.14	**4.14**
Arrhythmia	22.79	22.34	**21.03**	**26.31**	31.67	29.01
Clean1	7.56	5.88	**4.83**	**22.50**	25.41	23.94
WDBC	1.76	1.76	**1.75**	3.35	2.11	**1.76**
Australian	13.77	**12.75**	13.33	13.77	13.48	**13.48**
Blood	20.46	**20.34**	20.98	**20.20**	20.59	21.25
Pima	**21.89**	22.13	22.53	22.80	**22.14**	22.67
German	23.9	23.3	**22.9**	24.80	23.90	**23.10**

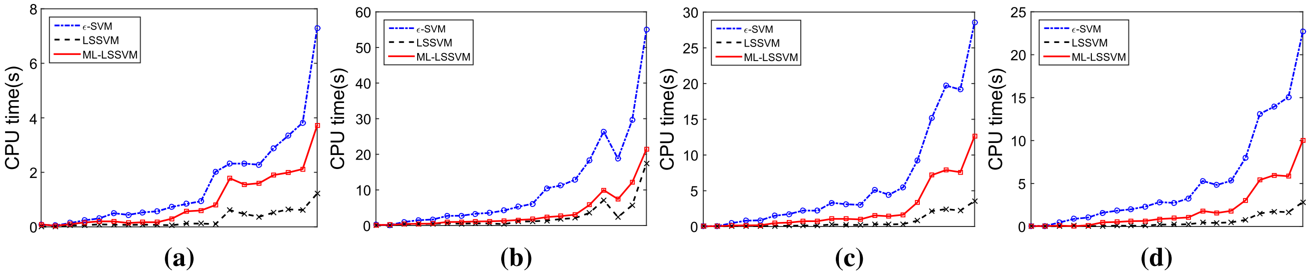


Fig. 5 Comparisons of CPU time. **a** Linear, **b** Poly, **c** Rbf, **d** Sigmoid

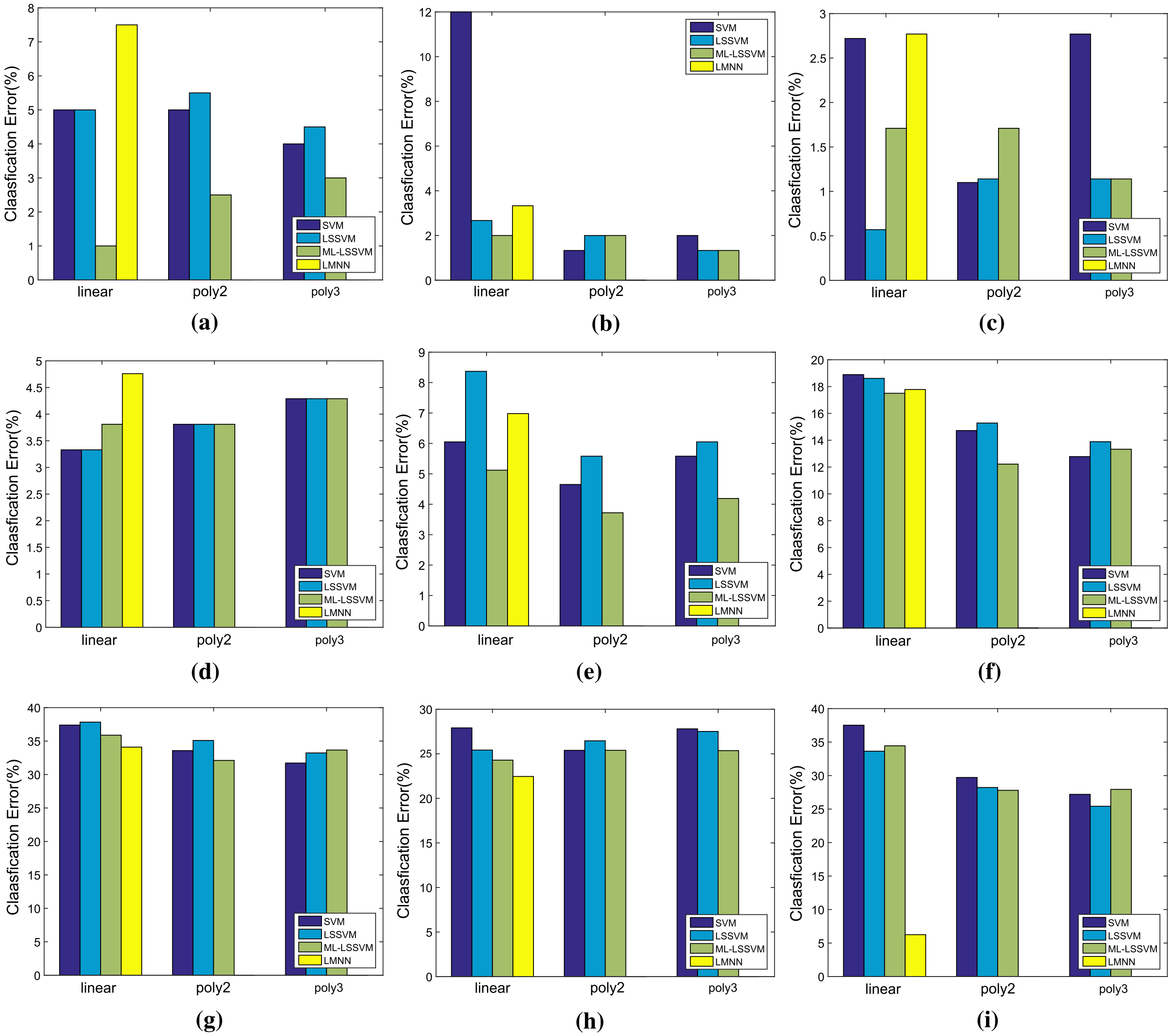


Fig. 6 Error rate of Multi-class classification. **a** Circle, **b** iris, **c** wine, **d** seeds, **e** thyroid, **f** libras, **g** gem, **h** vehicle, **i** vowel

one of the earliest work in metric learning. Within-class distance and between-class distance are defined by the sum of pairwise distances with respect to a new learned Mahalanobis matrix. But LSSVM seeks for two parallel hyperplanes, treated as a center marks for each class, respectively. Within-class distance is measured by the sum of distances between points and corresponding hyperplane. The distance between the two center marks is the between-class distance. In fact, LSSVM implements the idea of metric learning essentially. Though, LSSVM can be improved by revising between-class distance. A novel method, called ML-LSSVM, is presented, which add constraints of inter-class into the primary LSSVM. ML-LSSVM can be solved effectively by ADMM algorithm, breaking large convex problems into smaller ones. Further, LMNN has an inner relation with ML-LSSVM and its local version is equivalent to ML-LSSVM, just different in training sets. Numerical experiments shown that the extra constraints in ML-LSSVM are advantageous in improving classification performance. In the future, we will investigate the relation of metric learning with more variants of SVM for improved performance.

Acknowledgements This work has been partially supported by grants from National Natural Science Foundation of China (Nos. 61472390, 11271361, 71331005, and 11226089), Major International (Regional) Joint Research Project (No. 71110107026) and the Beijing Natural Science Foundation (No.1162005).

Compliance with ethical standards

Conflict of interest The authors declared that they have no conflicts of interest to this work.

References

1. Bellet A, Habrard A, Sebban M (2013) A survey on metric learning for feature vectors and structured data. arXiv:1306.6709
2. Bilenko M, Basu S, Mooney RJ (2004) Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the twenty-first international conference on machine learning. ACM, p 11
3. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
4. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
5. Davis JV, Kulis B, Jain P, Sra S, Dhillon IS (2007) Information-theoretic metric learning. In: Proceedings of the 24th international conference on machine learning. ACM, pp 209–216
6. Deng N, Tian Y, Zhang C (2012) Support vector machines: optimization based theory, algorithms, and extensions. CRC Press, Boca Raton
7. Do H, Kalousis A, Wang J, Woznica A (2012) A metric learning perspective of SVM: on the relation of LMNN and SVM. In: International conference on artificial intelligence and statistics, pp 308–317
8. Goldberger J, Hinton GE, Roweis ST, Salakhutdinov R (2004) Neighbourhood components analysis. In: Advances in neural information processing systems, MIT Press, pp 513–520
9. Guillaumin M, Verbeek J, Schmid C (2009) Is that you? Metric learning approaches for face identification. In: 2009 IEEE 12th international conference on computer vision. IEEE, pp 498–505
10. Koestinger M, Hirzer M, Wohlhart P, Roth PM, Bischof H (2012) Large scale metric learning from equivalence constraints. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 2288–2295
11. Nguyen N, Guo Y (2008) Metric learning: a support vector approach. In: Joint european conference on machine learning and knowledge discovery in databases. Springer, Berlin, Heidelberg, pp 125–136
12. Suykens JA, De Brabanter J, Lukas L, Vandewalle J (2002) Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing 48(1):85–105
13. Suykens JA, Lukas L, Vandewalle J (2000) Sparse approximation using least squares support vector machines. In: The 2000 IEEE international symposium on circuits and systems, 2000. Proceedings. ISCAS 2000 Geneva, vol 2. IEEE, pp 757–760
14. Suykens JA, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J (2002) Least squares support vector machines. World Scientific, Singapore
15. Tan J, Zhang C, Deng N (2010) Cancer related gene identification via p-norm support vector machine. In: The 4th international conference on computational systems biology, vol 1, pp 101–108
16. Tian Y, Shi Y, Liu X (2012) Recent advances on support vector machines research. Technol Econ Dev Econ 18(1):5–33
17. Torresani L, Lee K-c (2006) Large margin component analysis. In: Advances in neural information processing systems, MIT Press, pp 1385–1392
18. Van Gestel T, Suykens JA, Baesens B, Viaene S, Vanthienen J, Dedene G, De Moor B, Vandewalle J (2004) Benchmarking least squares support vector machine classifiers. Mach Learn 54(1):5–32
19. Vapnik V (1998) Statistical learning theory. Wiley-Interscience, New York
20. Wang F, Sun J (2014) Survey on distance metric learning and dimensionality reduction in data mining. Data Min Knowl Discov 29(2):534–564
21. Weinberger KQ, Saul LK (2008) Fast solvers and efficient implementations for distance metric learning. In: Proceedings of the 25th international conference on machine learning. ACM, pp 1160–1167
22. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res 10:207–244
23. Xing EP, Jordan MI, Russell S, Ng AY (2002) Distance metric learning with application to clustering with side-information. In: Advances in neural information processing systems, MIT Press, pp 505–512
24. Yang L, Jin R (2006) Distance metric learning: a comprehensive survey, vol 2, Michigan State University, USA
25. Zaidi N, Squire D (2010) SVMs and data dependent distance metric. In: 2010 25th International conference of image and vision computing New Zealand (IVCNZ). IEEE, pp 1–7
26. Zaidi NA, Squire DM, Suter D (2011) A gradient-based metric learning algorithm for k-nn classifiers. In: AI 2010: advances in artificial intelligence. Springer, pp 194–203
27. Zou P-C, Wang J, Chen S, Chen H (2014) Bagging-like metric learning for support vector regression. Knowl Based Syst 65:21–30
28. Zuo W, Wang F, Zhang D, Lin L, Huang Y, Meng D, Zhang L (2015) Iterated support vector machines for distance metric learning. arXiv:1502.00363