Non-parallel Support Vector Classifiers with Different Loss Functions

Siamak Mehrkanoon[1], Xiaolin Huang, and Johan A.K. Suykens

KU Leuven, ESAT-STADIUS, Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium.

Abstract

This paper introduces a general framework of non-parallel support vector machines, which involves a regularization term, a scatter loss and a misclassification loss. When dealing with binary problems, the framework with proper losses covers some existing non-parallel classifiers, such as multisurface proximal support vector machine via generalized eigenvalues, twin support vector machines, and its least squares version. The possibility of incorporating different existing scatter and misclassification loss functions into the general framework is discussed. Moreover, in contrast with the mentioned methods, which applies kernel-generated surface, we directly apply the kernel trick in the dual and then obtain nonparametric models. Therefore, one does not need to formulate two different primal problems for the linear and nonlinear kernel respectively. In addition, experimental results are given to illustrate the performance of different loss functions.

Key words: non-parallel classifiers, least squares loss, pinball loss, hinge loss, kernel trick

1. Introduction

Support Vector Machines (SVM) is a powerful paradigm for solving pattern recognition problems [1, 2]. In this method one maps the data into a high dimensional feature space and then constructs an optimal separating hyperplane in the feature space. This method attempts to reduce the generalization error by maximizing the margin. The problem is formulated as a convex quadratic programming problem. Least squares support vector machines (LSSVMs) on the other hand have been proposed in [3] for function estimation, classification, unsupervised learning, and other tasks [3, 4]. In this case, the problem formulation involves equality instead of inequality constraints. Therefore in the dual one will deal with a system of linear equations instead of a quadratic optimization problem.

For binary classification problems, both SVMs and LSSVMs aim at constructing two parallel hyperplanes (or the hyperplanes in the feature space) to do classification. An extension is to consider non-parallel hyperplanes. The concept of applying two non-parallel hyperplanes was first introduced in [5], where two non-parallel hyperplanes were determined via solving two generalized eigenvalue problems and called GEPSVM. In this case one obtains two non-parallel hyperplanes where each one is as close as possible to the data points of one class and as far as possible from the data points of the other class. Recently many approaches, based on non-parallel hyperplanes, have been developed for classification, regression and feature selection tasks (see [6]–[11]).

The authors in [12] modified GEPSVM and proposed a non-parallel classifier called Twin Support Vector Machines (TWSVM), that obtains two non-parallel hyperplanes by solving a pair of quadratic programming problems. An improved TWSVM termed as TBSVM is given in [13] where the structural risk is minimized. Motivated by the ideas given in [3] and [14], recently least twin support vector machines (LSTSVM) is presented in [15], where the primal quadratic problems of TSVM is modified into least squares problem via replacing inequalities constraints by equalities.

In the above mentioned approaches, kernel-generated surfaces are used for designing a nonlinear classifier. In addition one has to construct different primal problems depending on whether a linear or nonlinear kernel is applied. It is the purpose of this paper to formulate a non-parallel support vector machine classifier for which we can directly apply the kernel trick and thus it enjoys the primal and dual properties as in classical support vector machines classifiers. A general framework of non-parallel support vector machine, which consists of a regularization term, a scatter loss and a misclassification loss is provided. The framework is designed for multi-class problems. Several choices for the losses are investigated. The corresponding non-parametric models are given via considering the dual problems and the kernel trick.

The paper is organized as follows. In Section 2, a non-parallel support vector machine classifier with a general form is given. In Section 3, several choices of losses are discussed. The guidelines for the user are provided in section 4. In Section 5, experimental results are given in order to confirm the validity and applicability of the proposed methods.

2. Non-parallel Support Vector Machine

Let us consider a given training dataset $\{x_i, y_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$, y_i is the label of the i-th data point and there are M number

[1]Corresponding author.
E-mail address: {*siamak.mehrkanoon,xiaolin.huang,johan.suykens*}*@esat.kuleuven.be*

of classes. Here the one-vs-all strategy is utilized to build the codebook, i.e., the training points belonging to the m-th class are labeled by +1 and all the remaining data from the rest of the classes are considered to have negative labels. The index set corresponding to class m is denoted by $\mathcal{I}_m$. We seek non-parallel hyperplanes in the feature space:

$$f_m(x) = w_m^T \varphi_m(x) + b_m = 0, \quad m = 1, 2, \ldots, M$$

each of which is as close as possible to the points of its own class and as far as possible from the data points of the other class.

2.1. General formulation

In the primal, the hyperplane $f_m(x) = 0$ for class m can be constructed by the following problem,

$$\begin{aligned} \min_{w_m, b_m, e, \xi} \quad & \frac{1}{2} w_m^T w_m + \frac{\gamma_1}{2} \sum_{i \in \mathcal{I}_m} L_{(1)}(e_i) + \frac{\gamma_2}{2} \sum_{i \notin \mathcal{I}_m} L_{(2)}(\xi_i) \\ \text{subject to} \quad & w_m^T \varphi_m(x_i) + b_m = e_i, \forall i \in \mathcal{I}_m \\ & 1 + \left(w_m^T \varphi_m(x_i) + b_m \right) = \xi_i, \forall i \notin \mathcal{I}_m. \end{aligned} \tag{1}$$

After solving (1) for $m = 1, 2, \ldots, M$, we obtain M non-parallel hyperplanes in the feature space. Then the label of the new test point x^* is determined depending on the perpendicular distances of the test points from the hyperplanes. Mathematically, the decision rule can be written as follows:

$$\text{Label}(x^*) = \arg \min_{m=1,2,\ldots,M} \{ d_m(x^*) \}, \tag{2}$$

where the perpendicular distance $d_m(x^*)$ is calculated by

$$d_m(x^*) = \frac{\left| w_m^T \varphi_m(x^*) + b_m \right|}{\|w_m\|_2}, \quad m = 1, 2, \ldots, M.$$

The target of (1) is to establish a hyperplane which is close to the points in class $\mathcal{I}_m$ and also is far away from the points that are not in this class. Therefore, any scatter loss function can be used for $L_{(1)}(\cdot)$ and at the same time any misclassification loss function can be utilized for $L_{(2)}(\cdot)$. Possible choices for $L_{(1)}(\cdot)$ include least squares, ϵ-insensitive tube, absolute, and Huber loss. For $L_{(2)}(\cdot)$, one can consider least squares, hinge, or squared hinge loss. Different loss has its own statistical properties and is suitable for different tasks. The proposed general formulation (1) is to handle multi-class problems, for which we essentially solve a series of binary problems. In the binary problem related to class m, we regard $x_i, i \in \mathcal{I}_m$ and the remaining points as two classes. Hence, the basic scheme of (1) for multi-class problems and binary problems is similar. For the convenience of expression, we focus on binary problems in theoretical discussion and evaluate multi-class problems in numerical experiments. Besides, for each class, one can apply different nonlinear feature mapping in (1). But in this paper, we discuss the case that unique $\varphi(x)$ is used for all the classes.

2.2. Related existing methods

For a binary problem, we assume that there are n_1 points in class 1 and n_2 points in class 2, i.e., there are n_1 elements in $\mathcal{I}_1$ and n_2 in $\mathcal{I}_2$. Suppose X_1 and X_2 is the matrix, of which each column is the vector $x_i, i \in \mathcal{I}_1$ and $x_i, i \in \mathcal{I}_2$, respectively. The corresponding matrices with feature mapping $\varphi(\cdot)$ are denoted by Φ_1 and Φ_2, i.e. the i-th row of Φ_1 is the vector $\varphi(x_i), i \in \mathcal{I}_1$, and so is Φ_2. Denote $Y_{n_1} = \text{diag}\{+1\}_{i=1}^{n_1} \in \mathbb{R}^{n_1 \times n_1}$, $Y_{n_2} = \text{diag}\{-1\}_{i=1}^{n_2} \in \mathbb{R}^{n_2 \times n_2}$, and 1_n as an n dimensional vector with all components equal to one. Then the non-parallel SVM (1) can be written in matrix formulation as the following two problems:

$$\begin{aligned} \min_{w_1, b_1, e, \xi} \quad & \frac{1}{2} w_1^T w_1 + \frac{\gamma_1}{2} L_{(1)}(e) + \frac{\gamma_2}{2} L_{(2)}(\xi) \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2} \left[\Phi_2 w_1 + b_1 1_{n_2} \right] + \xi = 1_{n_2}, \end{aligned} \tag{3}$$

and

$$\begin{aligned} \min_{w_2, b_2, e, \xi} \quad & \frac{1}{2} w_2^T w_2 + \frac{\gamma_1}{2} L_{(1)}(e) + \frac{\gamma_2}{2} L_{(2)}(\xi) \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1} \left[\Phi_1 w_2 + b_2 1_{n_1} \right] + \xi = 1_{n_1}. \end{aligned} \tag{4}$$

As discussed previously, $L_{(1)}(\cdot)$ could be any scatter loss function and any misclassification loss can be used in $L_{(2)}(\cdot)$. Some choices have been discussed. For example, if one chooses least squares loss for $L_{(1)}(\cdot)$ and hinge loss for $L_{(2)}(\cdot)$ and let $\gamma_1, \gamma_2 \to \infty$, the problem formulations (3) and (4), when a linear kernel is used, will reduce to TWSVM introduced in [12]:

$$\text{TWSVM1} \qquad \begin{aligned} \min_{w_1, b_1, \xi} \quad & \frac{1}{2} \|X_1 w_1 + b_1 1_{n_1}\|^2 + C_1 1_{n_2}^T \xi \\ \text{subject to} \quad & -(X_2 w_1 + b_1 1_{n_2}) + \xi \geq 1_{n_2}, \end{aligned} \tag{5}$$

$$\text{TWSVM2} \qquad \begin{aligned} \min_{w_2, b_2, \xi} \quad & \frac{1}{2} \|X_2 w_2 + b_2 1_{n_2}\|^2 + C_2 1_{n_1}^T \xi \\ \text{subject to} \quad & (X_1 w_2 + b_2 1_{n_1}) + \xi \geq 1_{n_1}. \end{aligned} \tag{6}$$

Another example is choosing least squares loss for both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Again, letting $\gamma_1, \gamma_2 \to \infty$ in (3) and (4) and using a linear kernel, one obtain the LSTSVM formulation reported in [15]

$$\text{LSTSVM1} \qquad \begin{aligned} \min_{w_1, b_1, \xi} \quad & \frac{1}{2} \|X_1 w_1 + b_1 1_{n_1}\|^2 + \frac{C_1}{2} \xi^T \xi \\ \text{subject to} \quad & -(X_2 w_1 + b_1 1_{n_2}) + \xi = 1_{n_2}, \end{aligned} \tag{7}$$

$$\text{LSTSVM2} \qquad \begin{aligned} \min_{w_2, b_2, \xi} \quad & \frac{1}{2} \|X_2 w_2 + b_2 1_{n_2}\|^2 + \frac{C_2}{2} \xi^T \xi \\ \text{subject to} \quad & (X_1 w_2 + b_2 1_{n_1}) + \xi = 1_{n_1}. \end{aligned} \tag{8}$$

In contrast with the classical support vector machines technique, TWSVM and LSTSVM do not take the structural risk

minimization into account. For TWSVM, the authors in [13] gave an improvement by adding a regularization term in the objective function aiming at minimizing the structural risk by maximizing the margin. This method is called TBSVM, where the bias term is also penalized. But penalizing the bias term will not affect the result significantly and will change the optimization problem slightly. From a geometric point of view it is sufficient to penalize the norm of w in order to maximize the margin.

Another noticeable point is that TWSVM, LSTSVM, and TBSVM use a kernel generated surface to apply nonlinear kernels. As opposed to these methods, in our formulation, the burden of designing another two optimization formulations, when nonlinear kernel is used, is reduced by applying the Mercer's theorem and kernel trick directly, which will be investigated in the following section.

3. Different Loss Functions

There are several possibilities for choosing the loss functions $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Our target is to make the points in one class clustered in the hyperplane by minimizing $L_{(1)}(\cdot)$, which hence should be a scatter loss. For this aim, we prefer to use the least squares loss for $L_{(1)}(\cdot)$, because the related problem is easy to handle. Its weak point is that the least squares loss is sensitive to large outliers, then one may also consider ℓ_1-norm or Huber loss under the proposed framework. For $L_{(2)}(\cdot)$, which penalties misclassification error to push the points in other classes away from the hyperplane, we need misclassification loss. In what follows, we illustrate the following loss functions used in (3) and (4). Other loss functions can be discussed similarly:

- Least squares loss for $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$ (will be referred to as LS-LS case).
- Least squares loss for $L_{(1)}(\cdot)$ and hinge loss for $L_{(2)}(\cdot)$ (will be referred to as LS-Hinge case).
- Least squares loss for $L_{(1)}(\cdot)$ and pinball loss for $L_{(2)}(\cdot)$ (will be referred to as LS-Pinball case).

The above-mentioned loss functions are depicted in Fig 1.

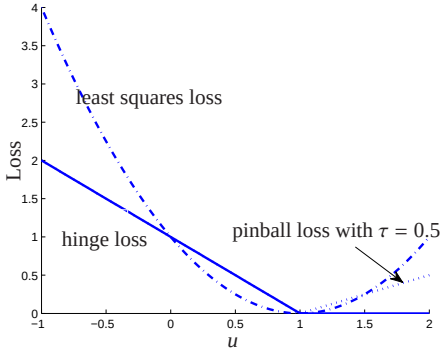


Figure 1: Some loss functions for $L_{(2)}(\cdot)$: hinge loss (solid line), least squares loss (dot-dashed line), and pinball loss with $\tau = 0.5$ (dotted line).

3.1. Case: LS-LS loss

We first investigate the case using least squares loss in both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Due to the fact that applying least squares loss will lead to a set of linear systems, this choice has much lower computational cost in comparison with other loss functions, which may result in solving quadratic programming problems or nonlinear systems of equations. Specifically, using least squares loss in (3) and (4) leads to the following problems:

$$\begin{aligned} \min_{w_1,b_1,e,\xi} \quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_2}{2}\xi^T \xi \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2}\Big[\Phi_2 w_1 + b_1 1_{n_2}\Big] + \xi = 1_{n_2}, \end{aligned} \tag{9}$$

and

$$\begin{aligned} \min_{w_2,b_2,e,\xi} \quad & \frac{1}{2}w_2^T w_2 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_2}{2}\xi^T \xi \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1}\Big[\Phi_1 w_2 + b_2 1_{n_1}\Big] + \xi = 1_{n_1}. \end{aligned} \tag{10}$$

In this case, problems (9) or (10) becomes a quadratic minimization under linear equality constraints, which enables a straightforward solution.

The obtained formulations (9) and (10) are closely related to LSTSVM (7) and (8). An important difference is that there are regularization terms involved in (9) and (10), which makes the kernel trick applicable to obtain nonparametric models. In [15], the kernel generated surfaces were introduced to LSTSVM, which does not consider structural risk minimization and also brings the burden of designing another two optimization formulations when nonlinear kernel is used. Our nonparametric model can be directly obtained from the dual problem of (9) and (10), illustrated below.

Theorem 3.1. *Given a positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(t,s) = \varphi(t)^T\varphi(s)$ and regularization constants $\gamma_1, \gamma_2 \in \mathbb{R}^+$, the dual problem of (9) is posed as:*

$$\left[\begin{array}{c|c|c} \Omega_{11} + I_{n_1}/\gamma_1 & \Omega_{12}Y_{n_2} & 1_{n_1} \\ \hline Y_{n_2}\Omega_{21} & \Omega_{22} + I_{n_2}/\gamma_2 & Y_{n_2}1_{n_2} \\ \hline 1_{n_1}^T & 1_{n_2}^T Y_{n_2} & 0 \end{array}\right]\left[\begin{array}{c} \alpha_1 \\ \hline \beta_1 \\ \hline b_1 \end{array}\right] = \left[\begin{array}{c} 0_{n_1} \\ \hline 1_{n_2} \\ \hline 0 \end{array}\right] \tag{11}$$

with $\alpha_1 \in \mathbb{R}^{n_1}, \beta_1 \in \mathbb{R}^{n_2}, \Omega_{11} = \Phi_1\Phi_1^T, \Omega_{22} = \Phi_2\Phi_2^T, \Omega_{12} = \Phi_1\Phi_2^T$ and $\Omega_{21} = \Omega_{12}^T$. In other words, the elements of Ω_{11} are calculated by $K(x_i, x_j)$, $i, j \in \mathcal{I}_1$, and so are Ω_{12}, Ω_{21} and Ω_{22}.

Proof. The Lagrangian of the constrained optimization problem (9) becomes

$$\begin{aligned} &\mathcal{L}(w_1, b_1, e, \xi, \alpha_1, \beta_1) = \\ &\frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_1}{2}\xi^T\xi - \alpha_1^T\Big(\Phi_1 w_1 + b_1 1_{n_1} - e\Big) - \\ &\beta_1^T\Big(Y_{n_2}\Big[\Phi_2 w_1 + b_1 1_{n_2}\Big] + \xi - 1_{n_2}\Big) \end{aligned}$$

where α_1 and β_1 are the Lagrange multipliers corresponding to the constraints in (9). Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial w_1} = 0 \rightarrow\ w_1 = \Phi_1^T\alpha_1 + \Phi_2^T Y_{n_2}\beta_1,\\
&\frac{\partial \mathcal{L}}{\partial b_1} = 0 \rightarrow\ 1_{n_1}^T\alpha_1 + 1_{n_2}^T Y_{n_2}\beta_1 = 0,\\
&\frac{\partial \mathcal{L}}{\partial e} = 0 \rightarrow\ e = -\frac{\alpha_1}{\gamma_1},\\
&\frac{\partial \mathcal{L}}{\partial \xi} = 0 \rightarrow\ \xi = \frac{\beta_1}{\gamma_2},\\
&\frac{\partial \mathcal{L}}{\partial \alpha_1} = 0 \rightarrow\ \Phi_1 w_1 + b_1 1_{n_1} - e = 0,\\
&\frac{\partial \mathcal{L}}{\partial \beta_1} = 0 \rightarrow\ Y_{n_2}\Big[\Phi_2 w_1 + b_1 1_{n_2}\Big] + \xi = 1_{n_2}.
\end{aligned}$$

After elimination of the primal variables w_1, e, ξ and making use of Mercer's Theorem, one can obtain the solution in the dual by solving linear system (11). □

Using a similar argument, one can show that the solution of optimization problem (4) can be obtained in the dual by solving the following linear system:

$$\left[\begin{array}{c|c|c} \Omega_{22} + I_{n_2}/\gamma_1 & \Omega_{21}Y_{n_1} & 1_{n_2} \\ \hline Y_{n_1}\Omega_{12} & \Omega_{11} + I_{n_1}/\gamma_2 & Y_{n_1}1_{n_1} \\ \hline 1_{n_2}^T & 1_{n_1}^T Y_{n_1} & 0 \end{array}\right]\left[\begin{array}{c} \alpha_2 \\ \hline \beta_2 \\ \hline b_2 \end{array}\right] = \left[\begin{array}{c} 0_{n_2} \\ \hline 1_{n_1} \\ \hline 0 \end{array}\right] \tag{12}$$

with $\alpha_2 \in \mathbb{R}^{n_2}, \beta_2 \in \mathbb{R}^{n_1}$.

Via solving (11) and (12), we obtain the optimal dual variables $\alpha_{1,2}$, $\beta_{1,2}$, and $b_{1,2}$. Then for the unseen test data points $\mathcal{D}^{\text{test}} = \{x_j^*\}_{j=1}^{n_{\text{test}}}$ the labels can be determined using (2) where

$$\begin{aligned}
d_1(\mathcal{D}^{\text{test}}) &= \frac{|\Phi_{\text{test}} w_1 + b_1 1_{n_{\text{test}}}|}{\|w_1\|_2}\\
&= \frac{|\Phi_{\text{test}}(\Phi_1^T\alpha_1 + \Phi_2^T Y_{n_2}\beta_1) + b_1 1_{n_{\text{test}}}|}{\|\Phi_1^T\alpha_1 + \Phi_2^T Y_{n_2}\beta_1\|_2},
\end{aligned}$$

and

$$\begin{aligned}
d_2(\mathcal{D}^{\text{test}}) &= \frac{|\Phi_{\text{test}} w_2 + b_2 1_{n_{\text{test}}}|}{\|w_2\|_2}\\
&= \frac{|\Phi_{\text{test}}(\Phi_2^T\alpha_2 + \Phi_1^T Y_{n_1}\beta_2) + b_2 1_{n_{\text{test}}}|}{\|\Phi_2^T\alpha_2 + \Phi_1^T Y_{n_1}\beta_2\|_2}.
\end{aligned}$$

Here $\Phi_{\text{test}} = [\varphi(x_1^*), \ldots, \varphi(x_{n_{\text{test}}}^*)]^T$. Thanks to the KKT optimality conditions, w_1 and w_2 are written in terms of Lagrange multipliers.

Next we will show that when we set $\gamma_1 = \gamma_2$, (11) and (12) reduce to least squares support vector machine classifier [4], given below,

$$\begin{aligned}
\min_{w,b,e}\quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e\\
\text{subject to}\quad & Y\Big[\Phi w + b1_N\Big] = 1_N - e,
\end{aligned} \tag{13}$$

where $N = n_1 + n_2$ is the number of training data in both class 1 and class 2.

Theorem 3.2. *Problems (9) and (10) are equivalent to the standard least squares support vector machine classifier (13) when $\gamma_1 = \gamma_2$.*

Proof. Consider problem (9) with least squares loss and $\gamma_1 = \gamma_2$. We introduce a new variable $\tilde{b}_1 = b_1 + 1/2$, and rewrite (9) as follows:

$$\begin{aligned}
\min_{w_1,\tilde{b}_1,e,\xi}\quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_2}{2}\xi^T\xi\\
\text{subject to}\quad & Y_{n_1}\Big[\Phi_1 w_1 + \tilde{b}_1 1_{n_1}\Big] - e = \frac{1}{2}1_{n_1}\\
& Y_{n_2}\Big[\Phi_2 w_1 + \tilde{b}_1 1_{n_2}\Big] + \xi = \frac{1}{2}1_{n_2},
\end{aligned} \tag{14}$$

where Y_{n_1} is defined as previously. Since $\gamma_1 = \gamma_2$, by combining the constraints, one can rewrite (14) as follows:

$$\begin{aligned}
\min_{w_1,\tilde{b}_1,\tilde{e}}\quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma}{2}\tilde{e}^T\tilde{e}\\
\text{subject to}\quad & \tilde{e} = \frac{1}{2}1_N - Y_N\Big[\Phi_2 w_1 + 2\tilde{b}_1 1_N\Big],
\end{aligned} \tag{15}$$

where $\Phi = \left[\begin{array}{c}\Phi_1\\ \Phi_2\end{array}\right], \tilde{e} = \left[\begin{array}{c} e \\ \xi\end{array}\right], Y_N = \left[\begin{array}{c} Y_{n_1} \\ Y_{n_2}\end{array}\right]$ and $1_N = \left[\begin{array}{c} 1_{n_1} \\ 1_{n_2}\end{array}\right]$.

Now let $\bar{w} = 2w_1$ and $\bar{b} = 2\tilde{b}$, then one can find that (15) is equivalent to the following optimization problem:

$$\begin{aligned}
\min_{\bar{w},\bar{b},\bar{e}}\quad & \frac{1}{2}(\bar{w})^T(\bar{w}) + \frac{\gamma}{2}(\bar{e})^T(\bar{e})\\
\text{subject to}\quad & \bar{e} = 1_N - Y_N\Big[\Phi\bar{w} + \bar{b}1_N\Big],
\end{aligned} \tag{16}$$

which is indeed the classical LS-SVM classifier formulation. □

Similarly one can demonstrate that (4) with least squares loss and $\gamma_1 = \gamma_2$ will be equivalent to (13). This relationship implies that the LS-LS is an extension to LS-SVM, from which we can start from LS-SVM and then improve the classifier using LS-LS model.

3.2. Case: LS-Hinge loss

In the non-parallel SVM framework (3) and (4), if we choose the least squares loss for $L_{(1)}(\cdot)$ and hinge loss for $L_{(2)}(\cdot)$, then the problem in the primal has the the following form

$$\begin{aligned}
\min_{w_1,b_1,e,\xi}\quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \gamma_2 1_{n_2}^T\xi\\
\text{subject to}\quad & \Phi_1 w_1 + b_1 1_{n_1} = e\\
& Y_{n_2}\Big[\Phi_2 w_1 + b_1 1_{n_2}\Big] + \xi \geq 1_{n_2}\\
& \xi \geq 0_{n_2},
\end{aligned} \tag{17}$$

and

$$\begin{aligned}
\min_{w_2,b_2,e,\xi}\quad & \frac{1}{2}w_2^T w_2 + \frac{\gamma_1}{2}e^T e + \gamma_2 1_{n_1}^T\xi\\
\text{subject to}\quad & \Phi_2 w_2 + b_2 1_{n_2} = e\\
& Y_{n_1}\Big[\Phi_1 w_2 + b_2 1_{n_1}\Big] + \xi \geq 1_{n_1}\\
& \xi \geq 0_{n_1}.
\end{aligned} \tag{18}$$

Following the similar technique in the last subsection, the dual problem of (17) can be constructed as

$$\begin{aligned} \max_{\mu_1} \quad & -\frac{1}{2}\mu_1^T H_1 \mu_1 + F_1 \mu_1 \\ \text{subject to} \quad & A_1\mu_1 = 0 \\ & 0 \le \beta_1 \le \gamma_2 1_{n_2}, \end{aligned} \tag{19}$$

where $H_1 = \left[\begin{array}{c|c} \Omega_{11} + \gamma_1^{-1} I_{n_1} & \Omega_{12} Y_{n_2} \\ \hline Y_{n_2}\Omega_{21} & Y_{n_2}\Omega_{22} Y_{n_2} \end{array}\right]$, $\mu_1 = [{\alpha_1}^T, {\beta_1}^T]^T$, $F_1 = [0_{n_1}^T, 1_{n_2}^T]$, and $A_1 = [1_{n_1}^T, 1_{n_2}^T Y_{n_2}]$.

Correspondingly, the dual problem of (18) is

$$\begin{aligned} \max_{\mu_2} \quad & -\frac{1}{2}\mu_2^T H_2 \mu_2 + F_2 \mu_2 \\ \text{subject to} \quad & A_2\mu_2 = 0 \\ & 0 \le \beta_2 \le \gamma_2 1_{n_1}, \end{aligned} \tag{20}$$

where $H_2 = \left[\begin{array}{c|c} \Omega_{22} + \gamma_1^{-1} I_{n_2} & \Omega_{21} Y_{n_1} \\ \hline Y_{n_1}\Omega_{12} & Y_{n_1}\Omega_{11} Y_{n_1} \end{array}\right]$, $\mu_2 = [{\alpha_2}^T, {\beta_2}^T]^T$, $F_2 = [0_{n_2}^T, 1_{n_1}^T]$, and $A_2 = [1_{n_2}^T, 1_{n_1}^T Y_{n_1}]$.

It can be seen that the formulations (19) and (20) differ from those given in [12] by

$$\begin{aligned} \min_{w_1,b_1,\xi} \quad & \left\|\bar{\Omega}^1 w_1 + b_1 1_{n_1}\right\|_2^2 + C_1 1_{n_2}^T \xi \\ \text{subject to} \quad & -\left(\bar{\Omega}^2 w_1 + b_1 1_{n_2}\right) + \xi \ge 1_{n_2} \\ & \xi \ge 0_{n_2}, \end{aligned} \tag{21}$$

and

$$\begin{aligned} \min_{w_2,b_2,\xi} \quad & \left\|\bar{\Omega}^2 w_2 + b_2 1_{n_2}\right\|_2^2 + C_2 1_{n_1}^T \xi \\ \text{subject to} \quad & \left(\bar{\Omega}^1 w_2 + b_2 1_{n_1}\right) + \xi \ge 1_{n_1} \\ & \xi \ge 0_{n_1}, \end{aligned} \tag{22}$$

$\bar{\Omega}^1$ and $\bar{\Omega}^2$ are $n_1 \times (n_1 + n_2)$ and $n_2 \times (n_1 + n_2)$ matrices respectively. $\bar{\Omega}^1_{ij} = K(x_i, x_j)$ with x_i, $i \in \mathcal{I}_1$ and x_j, $j \in \mathcal{I}_1 \cup \mathcal{I}_2$ and $\bar{\Omega}^2_{ij} = K(x_i, x_j)$ with x_i, $i \in \mathcal{I}_2$ and x_j, $j \in \mathcal{I}_1 \cup \mathcal{I}_2$.

and K is the kernel function. $\mathcal{I}_1$ and $\mathcal{I}_2$ have been defined previously in section 2.2.

In (19) and (20) the kernel generated surfaces are not used and our formulation enjoy the advantages of having primal and dual formulations with applying the kernel trick. Also the structural risk minimization is obtained by means of the regularization terms $w_1^T w_1$ and $w_2^T w_2$. Compared with the kernel generated surfaces, (19) and (20) also enjoys good optimization structure, since they are quadratic programming problems with box constraints. For such kind of problems, we can apply sequential minimal optimization (SMO, [16, 17]) technique, which is effective and is generally a popular solving method for SVMs.

If one uses least squares loss for both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$, then in the dual a set of linear systems have to be solved but no sparsity will be achieved. Whereas if one chooses typical SVM losses, e.g., ϵ-insensitive zone loss for $L_{(1)}(\cdot)$, and hinge loss for $L_{(2)}(\cdot)$, then in the dual the hyperparameters of the model can be obtained by solving a convex quadratic optimization problem. In this case sparsity is enhanced since the training points that are correctly classified and are far enough from the margins will have no influence on the decision boundary. One can also use Huber loss function for $L_{(1)}(\cdot)$ to cope with the noise or outliers in the data set.

3.3. Case: LS-Pinball loss

When the hinge loss is minimized, the distance that we maximize is related to the nearest points which is prone to be sensitive to noise. Therefore attempts have been made to overcome this weak point by changing the definition of the distance between two sets. For instance, if one uses the distance of the nearest 20% points to measure the distance between two sets, the result is more robust. Such distance is a kind of quantile value, which is closely related to pinball loss [18, 19, 20]. In classification, we consider the following definition of pinball loss:

$$L_\tau(u) = \begin{cases} u, & u \ge 0, \\ -\tau u, & u < 0. \end{cases}$$

The pinball loss has been used for classification problems in [21]. The advantage of using the pinball loss holds as well for non-parallel classifiers. The corresponding model can be formulated as the following quadratic programming problems,

$$\begin{aligned} \min_{w_1,b_1,e,\xi} \quad & \frac{1}{2}{w_1}^T w_1 + \frac{\gamma_1}{2} e^T e + \gamma_2 1_{n_2}^T \xi \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2}\left[\Phi_2 w_1 + b_1 1_{n_2}\right] + \xi \ge 1_{n_2} \\ & Y_{n_2}\left[\Phi_2 w_1 + b_1 1_{n_2}\right] - \frac{1}{\tau}\xi \le 1_{n_2}, \end{aligned} \tag{23}$$

and

$$\begin{aligned} \min_{w_2,b_2,e,\xi} \quad & \frac{1}{2}{w_2}^T w_2 + \frac{\gamma_1}{2} e^T e + \gamma_2 1_{n_1}^T \xi \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1}\left[\Phi_1 w_2 + b_2 1_{n_1}\right] + \xi \ge 1_{n_1} \\ & Y_{n_1}\left[\Phi_1 w_2 + b_2 1_{n_1}\right] - \frac{1}{\tau}\xi \le 1_{n_1}. \end{aligned} \tag{24}$$

Similarly to the previous discussions, we can derive the corresponding nonparametric model. The dual problem of (23) is

$$\begin{aligned} \max_{\mu_1} \quad & -\frac{1}{2}\mu_1^T H_1 \mu_1 + F_1 \mu_1 \\ \text{subject to} \quad & A_1\mu_1 = 0 \\ & -\tau\gamma_2 1_{n_2} \le \beta_1 \le \gamma_2 1_{n_2}, \end{aligned} \tag{25}$$

and that of (24) is

$$\begin{aligned} \max_{\mu_2} \quad & -\frac{1}{2}\mu_2^T H_2 \mu_2 + F_2 \mu_2 \\ \text{subject to} \quad & A_2\mu_2 = 0 \\ & -\tau\gamma_2 1_{n_1} \le \beta_2 \le \gamma_2 1_{n_1}. \end{aligned} \tag{26}$$

When $\tau = 0$, (25) and (26) reduces to (19) and (20), respectively. From this point of view, the LS-Pinball is an extension

to the LS-Hinge. This relationship also can be observed via comparing the hinge loss and pinball loss in the primal. As analyzed in [21], with a properly selected τ value , the pinball loss can bring noise-insensitivity to feature noise and stability to resampling. (25) and (26) are quadratic programming problems with box constraints, as LS-hinge. Therefore, we can also apply SMO or any SMO type algorithm such as SUMT proposed in [23] to solve LS-pinball.

Theorem 3.2 tells us that LS-LS with particular parameters reduces to LS-SVM. We are also interested in the relationship between other non-parallel classifiers and parallel ones. In parallel classification methods, only one loss function is minimized. In the proposed non-parallel framework (1), there are two loss functions involved. Only when we choose a unique loss for both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$, it is possible to reduce the non-parallel models to parallel ones. $L_{(1)}(\cdot)$ should be a scatter loss, that means asymmetric loss is needed. Hence, the hinge loss is not suitable for $L_{(1)}(\cdot)$ and it is hard to construct a non-parallel classifier from the SVM with hinge loss. One possible choice is to use ℓ_1 loss for $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Then a suitable parameter will lead (1) becomes pin-SVM [21] with $\tau = 1$. This relationship is applicable to establish effectively an improved method from the parallel methods.

4. Guidelines for the user

The proposed framework for constructing the non-parallel classifier consists of two types of loss functions: scatter and misclassification. As mentioned previously, any scatter loss function can be used for the $L_{(1)}(\cdot)$ and at the same time any misclassification loss can be employed for the $L_{(2)}(\cdot)$. Depending on the prior knowledge about the data under study, one may choose a specific scatter or misclassification loss function. For instance if the data is corrupted by label noise one may prefer to use the hinge or pinball loss misclassification which are less sensitive to outliers compared to least squares loss. In case no prior knowledge is available, then, in general, choosing the loss functions can be regarded as user defined choice. One may try different loss functions and select the one with minimum misclassification error on the validation set. Based on the statistical properties of each of the loss functions the following qualitative conclusion can be drawn.

Table 1: Qualitative conclusion for different loss functions

Type of noise	LS-LS	LS-Hinge	LS-Pinball
Label noise	✗	✓	✓
Feature noise	✓	✗	✓

Remark 1. One may notice that according to Theorem 3.2. the LS-SVM is a special case of LS-LS (with the ratio $r = 1$). Therefore in practice one can start with the LS-SVM algorithm and gradually change (tune) the ratio r, to obtain the non-parallel classifier with a better performance compared to the LS-SVM. After reaching the stage where the non-parallel classifier is built, one then can choose empirically the loss function that obtains the minimum misclassification error on the validation set.

Algorithm 1: Guidelines for the user

Input: Training data set $\mathcal{D} = \{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, the tuning parameters (if any)

Output: Class membership of test data points $\mathcal{D}^{test}$

1 Option 1. Try all combinations of the loss functions and choose the one with minimum misclassification error on the validation set.
2 Option 2. Start with the LS-SVM approach,
3 Employ Theorem 3.2. and obtain a non-parallel classifier,
4 Search for the best possible loss functions with minimum misclassification error on the validation set.

5. Numerical Experiments

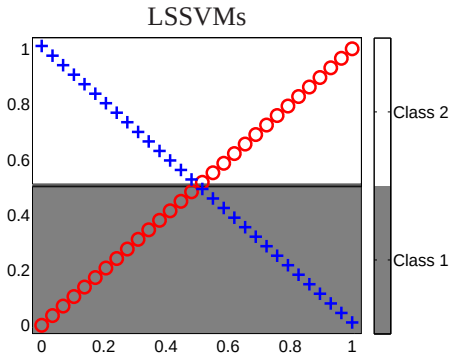
In this section experimental results on a synthetic data set so called "cross-planes" and real-life datasets from the UCI machine learning repository [22] are given. We compare the performance of the proposed methods (LS-Hinge, LS-LS, LS-pinball) with classical LSSVMs and method described in [15] over the above-mentioned datasets.

We first consider cross-planes data set for the relationship between LSSVMs and LS-LS, which has been studied in Theorem 3.2. The obtained results are depicted in Figure 1. LSSVMs with linear kernel are first tuned on this data set to obtain the optimal regularization parameter γ. Then the obtained γ is fed into the LS-LS formulation as γ_1 and regularization parameter γ_2 is set to $\frac{\gamma_1}{r}$.

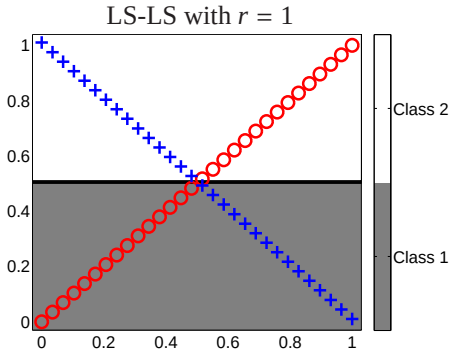
From Figure 1, it can be seen that the performance of the LS-LS when $r = 1$ ($\gamma_1 = \gamma_2$), is exactly equal to the performance of classical LSSVMs, i.e., we obtain two parallel hyperplanes. Whereas by changing the ratio r, which is defined as γ_1/γ_2, the classification accuracy is improved significantly. This is purely due to the ability of the proposed approach for designing two non-parallel hyperplanes. By changing the r value, hyperplanes start changing their directions. The optimal value for r is obtained by cross-validation method.

Figure 2, corresponds to the case when we have label noise, which can be regarded as outliers, in the data. As it was expected LS-LS is sensitive to noise whereas applying hinge or pinball loss functions will compensate the outliers to large extend.

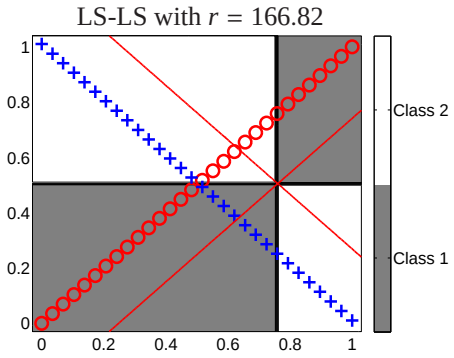
For UCI data sets, the parameters, including regularization constants γ_1, γ_2, kernel bandwidth σ, and in the case of pinball loss the parameter τ, are obtained using Coupled Simulated Annealing [24] approach initialized with 5 random sets of parameters. On every iteration step for CSA method we proceed with a 10-fold cross-validation. One may also use other existing techniques see [25, 26].



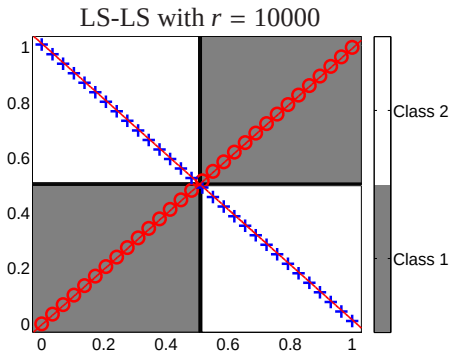
(a)



(b)

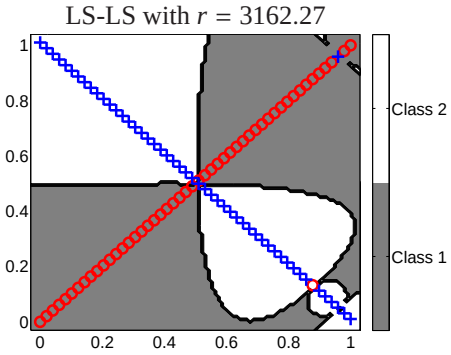


(c)

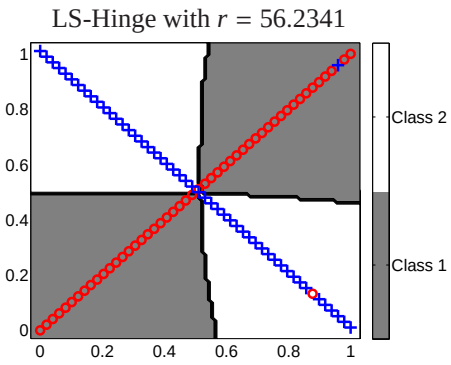


(d)

Figure 2: (a) Classification result obtained by LSSVMs with linear kernel, (b) Classification result obtained by LS-LS with linear kernel and $r = 1$, (c) Classification result obtained by LS-LS with linear kernel and $r = 166.82$, (d) Classification result obtained by LS-LS with linear kernel and $r = 10000$.



(a)



(b)

Figure 3: (a) Classification result obtained by LS-LS with nonlinear RBF kernel and, (b) Classification result obtained by LS-Hinge with nonlinear RBF kernel.

Descriptions of the used datasets from [22] can be found in Table 2. For Ecoli dataset some of the classes are merged in order to avoid unbalanced classes. One may consider the work in [27] to tackle the unbalanced classes.

Table 2: Dataset statistics

Dataset	# training data	# testing data	# attributes	# classes
Iris	105	45	4	3
Spect	80	187	21	2
Heart	135	135	13	2
Ecoli	100	236	7	5
Monk1	124	432	6	2
Monk2	169	132	6	2
Monk3	122	432	6	2
Ionosphere	176	175	33	2
Spambase	500	4101	57	2
Magic	500	18520	10	2
Seeds	147	63	7	3
Wine	125	53	13	3

We have artificially introduced random label and feature noise. To generate label noise, we randomly select 5% of samplings and change the observed labels. To generate feature noise, we add Gaussian noise to each feature and the signal-to-noise ratio is set to 20. All features for these data sets were normalized in a preprocessing step. We computed the means of the obtained accuracy over 10 simulation runs (every run includes 10 fold cross validation). The obtained results for RBF kernel are tabulated in Table 3, where the type of noise (no noise,

label noise, feature noise, both label and feature noise), dimension of the data, and the size of the training and testing sets are reported.

Table 3: Average binary classification accuracy on test sets with RBF kernel over 10 simulation runs with 5% label or/and feature noise

Datasets	Noise	**LSSVM** [3]	**LS-Hinge**	**LS-Pinball**	**LS-LS**	**LSTWSVM** [15]
Monk1	no noise	0.77	0.81	0.91	**0.96**	0.77
	label	0.78	**0.79**	**0.79**	0.78	0.78
	feature	0.72	**0.73**	0.72	0.72	0.64
	both	0.71	0.71	**0.73**	0.71	**0.73**
Monk2	no noise	0.87	0.86	0.87	**0.88**	**0.88**
	label	0.83	0.82	0.83	0.83	**0.84**
	feature	0.71	0.70	0.71	0.70	**0.72**
	both	0.69	**0.72**	**0.72**	0.70	0.71
Monk3	no noise	0.92	0.92	0.92	**0.93**	0.91
	label	0.90	0.91	**0.92**	0.90	0.88
	feature	0.85	0.85	**0.87**	0.83	0.81
	both	0.84	0.84	**0.86**	0.84	0.80
Spect	no noise	0.74	0.76	0.77	**0.84**	0.81
	label	0.77	**0.78**	0.75	0.77	0.77
	feature	0.71	0.77	0.74	0.78	**0.81**
	both	0.67	0.71	**0.77**	0.73	0.74
Ionosphere	no noise	**0.94**	**0.94**	**0.94**	**0.94**	0.93
	label	0.93	0.93	**0.94**	**0.94**	0.93
	feature	0.92	0.92	**0.93**	**0.93**	0.90
	both	0.89	0.92	**0.93**	**0.93**	0.92
Heart	no noise	0.83	0.82	0.81	**0.83**	0.70
	label	0.82	0.82	**0.82**	**0.82**	0.62
	feature	**0.86**	0.85	0.85	0.85	0.54
	both	0.82	0.82	0.82	**0.83**	0.63
Magic	no noise	0.78	**0.79**	**0.79**	0.78	0.59
	label	0.78	0.78	0.78	**0.79**	0.50
	feature	**0.78**	**0.78**	0.77	**0.78**	0.54
	both	0.77	0.71	0.77	**0.78**	0.51
Spambase	no noise	0.88	**0.91**	**0.91**	**0.91**	0.50
	label	0.89	**0.90**	**0.90**	**0.90**	0.50
	feature	0.88	0.88	**0.89**	**0.89**	0.51
	both	0.86	0.86	**0.88**	**0.88**	0.50

As discussed previously, the proposed non-parallel SVMs have more flexibility than the classical SVMs. The advantage of non-parallel classifiers is more obvious in the linear kernel than in the RBF kernel case, since the RBF kernel itself provides enough flexibility for many cases. Therefore, in many applications, the performance of classical SVMs and the non-parallel SVMs are similar. In Table 2, we only list the data sets with significant difference. The proposed non-parallel SVMs have different properties, due to the used loss functions. These properties have been discussed in Section 3. The least squares error is insensitive to feature noise but could be significantly affected by large outliers. Hence LS-LS generally performs well in feature noise cases but not in label noise cases. The LSTWSVM is also a kind LS-LS scheme and has similar performance as LS-LS. In contrast, the hinge loss is robust to outliers but only a few samples contribute the classifier. In this way the obtained classifier is robust to label noise but is sensitive to feature noise. The property of pinball loss used in classification has been discussed in [21]. Accordingly, LS-Pinball is a trade of between LS-LS and LS-Hinge and can give a good classifier when the data are contaminated by both label and feature noise.

As explained in Section 2.1, our non-parallel framework (1) is proposed for multi-class problems. In the next experiment, we consider four data sets from UCI machine learning repository. As for binary case, four scenarios: no noise, label noise, feature noise and feature/label noise are investigated. The average classification accuracy on test sets over 10 simulation runs are tabulated in Table 4. The performance of the proposed schemes on multi-class problem coincides with our explanation for binary classification tasks.

Table 4: Average multi-class classification accuracy on test sets with RBF kernel over 10 simulation runs with 5% label or/and feature noise.

Datasets	Noise	**LSSVM** [3]	**LS-Hinge**	**LS-Pinball**	**LS-LS**	**LSTWSVM** [15]
Ecoli	no noise	**0.85**	0.84	0.84	**0.85**	0.84
	label	0.81	**0.82**	0.81	0.79	0.81
	feature	**0.83**	0.82	**0.83**	0.81	0.80
	both	0.78	0.77	**0.79**	**0.79**	0.76
Iris	no noise	**0.97**	0.96	0.96	0.94	0.96
	label	0.93	0.94	**0.95**	0.93	0.93
	feature	0.93	0.93	**0.94**	**0.94**	0.93
	both	0.93	0.92	**0.94**	0.93	0.89
Seeds	no noise	**0.95**	0.93	0.94	**0.95**	**0.95**
	label	0.93	**0.94**	**0.94**	0.91	0.92
	feature	0.92	0.92	0.93	**0.94**	0.92
	both	0.89	0.90	**0.91**	0.90	0.89
Wine	no noise	0.98	**0.99**	**0.99**	0.97	0.98
	label	0.97	0.98	**0.99**	0.96	0.98
	feature	0.97	**0.98**	**0.98**	**0.9**8	0.97
	both	0.97	**0.98**	**0.98**	0.97	0.97

Recently several new algorithms have been reported in the literature for multiple output support vector regression task, see [28, 29, 30]. The adaptation of the proposed framework for regression is devoted to our future work.

6. Conclusions

In this paper, we gave a general framework for non-parallel classifier. As opposed to conventional approaches, the burden of formulating different optimization problem in the case of applying a non linear kernel, is avoided via utilizing the kernel trick in the dual. This framework enables the possibility of using different types of loss functions. Generally, different loss functions perform well for different problem, which is supported by numerical experiments. With the proposed non-parallel classifier, one can choose the suitable loss functions and achieve satisfactory performance for different distributions and different noise levels.

Acknowledgments

• EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. • Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants • Flemish Government: ○ FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants ○ IWT: projects: SBO POM (100031); PhD/Postdoc grants ○ iMinds Medical Information Technologies SBO 2014 • Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017). Johan Suykens is a professor at the KU Leuven, Belgium.

References

[1] V. Vapnik. *Statistic Learning Theory*. Cambridge University Press, 1998.

[2] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, 2002.

[3] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. Least Squares Support Vector Machines. 2002.

[4] J.A.K. Suykens and J. Vandewalle. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[5] O. L. Mangasarian and E. W. Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):69–74, 2006.

[6] Y.-P. Zhao, J. Zhao, and M. Zhao. Twin least squares support vector regression. *Neurocomputing*, 118:225–236, 2013.

[7] X. Peng. Efficient twin parametric insensitive support vector regression model. *Neurocomputing*, 79:26–38, 2012.

[8] X. Peng and D. Xu. Bi-density twin support vector machines for pattern recognition. *Neurocomputing* 99:134–143, 2013.

[9] Y. Shao, W. Chen, W. Huang, Z. Yang, and N. Deng. The best separating decision tree twin support vector machine for multi-class classification. *Procedia Computer Science*, 17:1032–1038, 2013.

[10] Y. Shao, N. Deng, and Z. Yang. Least squares recursive projection twin support vector machine for classification. *Pattern Recognition*, 45(6):2299–2307, 2012.

[11] Z. Yang, J. He, and Y. Shao. Feature selection based on linear twin support vector machines. *Procedia Computer Science*, 17:1039–1046, 2013.

[12] Jayadeva, R. Khemchandani, and S. Chandra. Twin support vector machines for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):905–910, 2007.

[13] Y. Shao, C. Zhang, X. Wang, and N. Deng. Improvements on twin support vector machines. *IEEE Transactions on Neural Networks*, 22(6):962–968, 2011.

[14] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proceedings of the 7-th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 77–86, 2001.

[15] M. Kumar and M. Gopal. Least squares twin support vector machines for pattern classification. *Expert Systems with Applications*, 36(4):7535–7543, 2009.

[16] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods – Support Vector Learning*, pages 185–208. MIT Press, 1999.

[17] R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

[18] R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

[19] I. Steinwart and A. Christmann. How SVMs can estimate quantiles and the median. *Advances in Neural Information Processing Systems*, 20:305 –312, 2008.

[20] I. Steinwart and A. Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1):211–225, 2011.

[21] X. Huang, L. Shi, and J.A.K. Suykens. Support vector machine classifier with pinball loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, doi:10.1109/TPAMI.2013.178, 2014.

[22] A. Frank and A. Asuncion. UCI machine learning repository. 2010.

[23] S. Joshi, G. Ramakrishnan and S. Chandra. Using Sequential Unconstrained Minimization Techniques to simplify SVM solvers. *Neurocomputing*, 77(1), 253–260, 2012.

[24] S. Xavier de Souza, J. A. K. Suykens, J. Vandewalle, and D. Bollé. Coupled simulated annealing. *IEEE Transations on System, Man, and Cybernetics, Part B*, 40(2):320–335, 2010.

[25] S. Li, M. Tan, Tuning SVM parameters by using a hybrid CLPSO-BFGS algorithm. *Neurocomputing*, 73, 2089–2096, 2010.

[26] Y. Bao, Z. Hu and T. Xiong. A PSO and pattern search based memetic algorithm for SVMs parameters optimization. *Neurocomputing*, 117, 98–106, 2013.

[27] X. Wang, Y. Niu. New one-versus-all ν-SVM solving intra-inter class imbalance with extended manifold regularization and localized relative maximum margin. *Neurocomputing*, 115, 106–121, 2013.

[28] Y. Bao, T. Xiong, Z. Hu. Multi-Step-Ahead Time Series Prediction using Multiple-Output Support Vector Regression. *Neurocomputing*, 129: 482–493, 2014.

[29] T. Xiong, Y, Bao, and Z. Hu. Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting. *Knowledge- Based Systems*, 55, 87–100, 2014.

[30] T. Xiong, Y. Bao, and Z. Hu. Does restraining end effect matter in EMD-based modeling framework for time series prediction? Some experimental evidences. *Neurocomputing*, 123, 174–184, 2014.