Non-parallel Support Vector Classifiers with Different Loss Functions

Siamak Mehrkanoon, 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 T SVM 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

---

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

Preprint submitted to Elsevier

May 17, 2014
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 $I_m$. We seek non-parallel hyperplanes in the feature space:

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

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,

$$\begin{align*}
\min_{w_m, b_m, \xi} & \quad \frac{1}{2} \|w_m\|^2 + \frac{\gamma_1}{2} \sum_{i \in I_m} L_1(e_i) + \frac{\gamma_2}{2} \sum_{i \not\in I_m} L_2(\xi_i) \\
\text{subject to} & \quad w_m^T \varphi_m(x_i) + b_m = e_i, \forall i \in I_m \\
& \quad 1 + (w_m^T \varphi_m(x_i) + b_m) = \xi_i, \forall i \not\in I_m.
\end{align*}$$

(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:

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

(2)

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

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

The target of (1) is to establish a hyperplane which is close to the points in class $I_m$ 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, $\ell_1$-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 $x_i, i \in I_m$ 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 $I_1$ and $n_2$ in $I_2$. Suppose $X_1$ and $X_2$ is the matrix, of which each column is the vector $x_i, i \in I_1$ and $x_i, i \in I_2$, respectively. The corresponding matrices with feature mapping $\varphi(\cdot)$ are denoted by $\Phi_1$ and $\Phi_2$, i.e. the i-th row of $\Phi_1$ is the vector $\varphi(x_i), i \in I_1$, and so is $\Phi_2$. Denote $Y_m = \text{diag}(+1)_{n_1} \in \mathbb{R}^{n_1 \times n_1}$, $Y_n = \text{diag}(-1)_{n_2} \in \mathbb{R}^{n_2 \times n_2}$, and $I_n$ 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:

$$\begin{align*}
\min_{w_1, b_1, \xi} & \quad \frac{1}{2} \|w_1\|^2 + \frac{\gamma_1}{2} L_1(e_1) + \frac{\gamma_2}{2} L_2(\xi) \\
\text{subject to} & \quad \Phi_1 w_1 + b_1 = e_1 \\
& \quad Y_n [\Phi_2 w_1 + b_1] + \xi = 1_{n_2},
\end{align*}$$

(3)

and

$$\begin{align*}
\min_{w_2, b_2, \xi} & \quad \frac{1}{2} \|w_2\|^2 + \frac{\gamma_1}{2} L_1(e_2) + \frac{\gamma_2}{2} L_2(\xi) \\
\text{subject to} & \quad \Phi_2 w_2 + b_2 = e_2 \\
& \quad Y_n [\Phi_1 w_2 + b_2] + \xi = 1_{n_2}.
\end{align*}$$

(4)

As discussed previously, $L_{11}(\cdot)$ could be any scatter loss function and any misclassification loss can be used in $L_{22}(\cdot)$. Some choices have been discussed. For example, if one chooses least squares loss for $L_{11}(\cdot)$ and hinge loss for $L_{22}(\cdot)$ 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]:

$$\begin{align*}
\text{TWSVM1} & \quad \min_{w_1, b_1, \xi} \frac{1}{2} \|X_1 w_1 + b_1 \|_2^2 + C_1 1_{n_1}^T \xi \\
\text{subject to} & \quad - (X_2 w_1 + b_1) + \xi \geq 1_{n_2},
\end{align*}$$

(5)

and

$$\begin{align*}
\text{TWSVM2} & \quad \min_{w_2, b_2, \xi} \frac{1}{2} \|X_2 w_2 + b_2 \|_2^2 + C_2 1_{n_2}^T \xi \\
\text{subject to} & \quad (X_1 w_2 + b_2) + \xi \geq 1_{n_1}.
\end{align*}$$

(6)

Another example is choosing least squares loss for both $L_{11}(\cdot)$ and $L_{22}(\cdot)$. 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]:

$$\begin{align*}
\text{LSTSVM1} & \quad \min_{w_1, b_1, \xi} \frac{1}{2} \|X_1 w_1 + b_1 \|_2^2 + C_1 1_{n_1}^T \xi \\
\text{subject to} & \quad - (X_2 w_1 + b_1) + \xi \geq 1_{n_2},
\end{align*}$$

(7)

and

$$\begin{align*}
\text{LSTSVM2} & \quad \min_{w_2, b_2, \xi} \frac{1}{2} \|X_2 w_2 + b_2 \|_2^2 + C_2 1_{n_2}^T \xi \\
\text{subject to} & \quad (X_1 w_2 + b_2) + \xi \geq 1_{n_1}.
\end{align*}$$

(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 opti-
mization 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 \( \ell_1 \)-norm or Huber
loss under the proposed framework. For \( L_{(2)}(\cdot) \), which penalties
classification error to push the points in other classes a way
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 func-
tions, which may result in solving quadratic programming pro-
blems or nonlinear systems of equations. Specifically, using least
squares loss in (3) and (4) leads to the following problems:

\[
\begin{aligned}
\min_{w, b, \varepsilon, \xi} & \quad \frac{1}{2} w^T w_1 + \frac{\gamma_1}{2} e^T e + \frac{\gamma_2}{2} \xi^T \xi \\
\text{subject to} & \quad \Phi_1 w_1 + b_1 1_{n_1} = e \\
& \quad Y_n [\Phi_2 w_1 + b_1 1_{n_2}] + \xi = 1_{n_2},
\end{aligned}
\]

and

\[
\begin{aligned}
\min_{w, b, \varepsilon, \xi} & \quad \frac{1}{2} w^T w_2 + \frac{\gamma_1}{2} e^T e + \frac{\gamma_2}{2} \xi^T \xi \\
\text{subject to} & \quad \Phi_2 w_2 + b_1 1_{n_2} = e \\
& \quad Y_n [\Phi_1 w_2 + b_1 1_{n_1}] + \xi = 1_{n_1}.
\end{aligned}
\]

In this case, problems (9) or (10) becomes a quadratic mini-
mization 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 for-
mulations 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 \( K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \),
with \( K(t, s) = \phi(t) \phi(s) \) and regularization constants \( \gamma_1, \gamma_2 \in \mathbb{R}^+ \),
the dual problem of (9) is posed as:

\[
\begin{bmatrix}
\Omega_{11} + I_{n_1}/\gamma_1 \\
Y_n \Omega_{21} \\
1^T_{n_1}
\end{bmatrix}
\begin{bmatrix}
\Omega_{12} Y_{n_2} \\
Y_n \Omega_{22} + I_{n_2}/\gamma_2 \\
1^T_{n_2} Y_{n_2}
\end{bmatrix}
\begin{bmatrix}
l_{n_1} \\
l_{n_2} \\
0
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\beta_1 \\
1_{n_2}
\end{bmatrix}
= \begin{bmatrix}
0_{n_1} \\
0_{n_2} \\
0
\end{bmatrix}
\]

(11)

with \( \alpha_1 \in \mathbb{R}^{n_1}, \beta_1 \in \mathbb{R}^{n_2}, \Omega_{11} = \Phi_1 \Phi_1^T, \Omega_{21} = \Phi_1 \Phi_2^T, \Omega_{12} = \Phi_2 \Phi_1^T \) and \( \Omega_{22} = \Phi_2 \Phi_2^T \). In other words, the elements of \( \Omega_{11} \)
are calculated by \( K(x_i, x_j), i, j \in I_1 \), and so are \( \Omega_{12}, \Omega_{21} \) and \( \Omega_{22} \).

**Proof.** The Lagrangian of the constrained optimization prob-
lem (9) becomes

\[
L(w_1, b_1, e, \xi, \alpha_1, \beta_1) =
\frac{1}{2} w_1^T w_1 + \frac{\gamma_1}{2} e^T e + \frac{\gamma_2}{2} \xi^T \xi - \alpha_1^T (\Phi_1 w_1 + b_1 1_{n_1} - e) - \\
\beta_1^T (Y_n [\Phi_2 w_1 + b_1 1_{n_2}] + \xi - 1_{n_2})
\]
where $\alpha_1$ and $\beta_1$ are the Lagrange multipliers corresponding to the constraints in (9). Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

\[
\begin{align*}
\frac{\partial L}{\partial w_1} = 0 &\rightarrow w_1 = \Phi^T_1 \alpha_1 + \Phi^T_2 Y_0 \beta_1, \\
\frac{\partial L}{\partial b_1} = 0 &\rightarrow 1^T_n \alpha_1 + 1^T_n Y_0 \beta_1 = 0, \\
\frac{\partial L}{\partial e} = 0 &\rightarrow e = -\frac{\alpha_1}{\gamma_1}, \\
\frac{\partial L}{\partial \xi} = 0 &\rightarrow \xi = \frac{\beta_1}{\gamma_2}, \\
\frac{\partial L}{\partial \alpha_1} = 0 &\rightarrow \Phi_1 w_1 + b_1 1_n - e = 0, \\
\frac{\partial L}{\partial \beta_1} = 0 &\rightarrow Y_n \left[ \Phi_2 w_1 + b_1 1_n \right] + \xi = 1_n.
\end{align*}
\]

After elimination of the primal variables $w_1, e, \xi$ 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:

\[
\begin{bmatrix}
\Omega_{22} + I_{n_2} / \gamma_1 \\
\frac{Y_n \Omega_{12}}{\gamma_2} \\
\frac{1}{\gamma_2} \\
\frac{Y_1 \Omega_{11}}{\gamma_1} \\
\frac{1}{\gamma_1} \\
\frac{1}{\gamma_1} \\
\end{bmatrix} \begin{bmatrix}
\alpha_2 \\
\beta_2 \\
1_{n_1} \\
1_{n_1} \\
0 \\
0 \\
\end{bmatrix} = \begin{bmatrix}
0_{n_2} \\
0_n \\
1_{n_1} \\
1_{n_1} \\
0 \\
0 \\
\end{bmatrix}
\]

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 $\mathcal{D}_{\text{test}} = \{x^*_{j_{\text{test}}}: j_{\text{test}} \}$ the labels can be determined using (2) where

\[
\phi = \begin{bmatrix}
\Phi_1 \\
\Phi_2 \\
\end{bmatrix}, \quad \hat{e} = \begin{bmatrix}
e \\
Y \end{bmatrix} = \begin{bmatrix}
Y_{n_1} \\
Y_{n_2} \\
1_{n_1} \\
1_{n_2} \\
\end{bmatrix} \quad \text{and} \quad \hat{y} = \begin{bmatrix}
Y_{n_1} \\
Y_{n_2} \\
\end{bmatrix}.
\]

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

\[
\begin{align*}
\min_{w_{\epsilon}, \delta} &\quad \frac{1}{2} (\tilde{w})^T (\tilde{w}) + \frac{\gamma_1}{2} \tilde{e}^T \tilde{e} \\
\text{subject to} &\quad \tilde{e} = 1_N - Y \Phi_{\tilde{w}} + \tilde{b} 1_N,
\end{align*}
\]

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

\[
\begin{align*}
\min_{w_{\epsilon}, \delta} &\quad \frac{1}{2} w^T \Phi_1 w + \frac{\gamma_1}{2} e^T e + \gamma_2 1_n \xi \\
\text{subject to} &\quad \Phi_1 w + b_1 1_n = e, \\
&\quad Y_n \left[ \Phi_2 w + b_1 1_n \right] + \xi \geq 1_n, \\
&\quad \xi \geq 0_n,
\end{align*}
\]

and

\[
\begin{align*}
\min_{w_{\epsilon}, \delta} &\quad \frac{1}{2} w^T \Phi_1 w + \frac{\gamma_1}{2} e^T e + \gamma_2 1_n \xi \\
\text{subject to} &\quad \Phi_1 w + b_1 1_n = e, \\
&\quad Y_n \left[ \Phi_2 w + b_1 1_n \right] + \xi \geq 1_n, \\
&\quad \xi \geq 0_n.
\end{align*}
\]

**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:

\[
\begin{align*}
\min_{w_{\epsilon}, \delta} &\quad \frac{1}{2} w^T \Phi_1 w + \frac{\gamma_1}{2} e^T e + \gamma_2 1_n \xi \\
\text{subject to} &\quad Y_n \left[ \Phi_1 w + \tilde{b}_1 1_n \right] - e = \frac{1}{2} 1_n, \\
&\quad Y_n \left[ \Phi_2 w + \tilde{b}_1 1_n \right] + \xi = \frac{1}{2} 1_n,
\end{align*}
\]

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

\[
\begin{align*}
\min_{w_{\epsilon}, \delta} &\quad \frac{1}{2} w^T \Phi_1 w + \frac{\gamma_1}{2} e^T e + \gamma_2 1_n \xi \\
\text{subject to} &\quad \tilde{e} = \frac{1}{2} 1_N - Y \Phi_{\tilde{w}} + \tilde{b} 1_N.
\end{align*}
\]

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

$$\begin{align*}
\max_{\mu_i} \quad & -\frac{1}{2} \mu_i^T H_1 \mu_1 + F_1 \mu_1 \\
\text{subject to} \quad & A_1 \mu_1 = 0 \\
& 0 \leq \mu_i \leq \gamma_2 \nu_i,
\end{align*}$$

where $H_1 = \begin{bmatrix} \Omega_1 + \gamma_1 \nu_i \n_i & \Omega_{12} \nu_i \\ \Omega_{12} \nu_i & \gamma_2 \nu_i \n_i \end{bmatrix}$, $\mu_1 = [\alpha_1^T, \beta_1^T]^T$, $F_1 = [0_{n_1}, 1_{n_1}^T]$, and $A_1 = [1_{n_1}, 1_{n_1}^T]$. Correspondingly, the dual problem of (18) is

$$\begin{align*}
\max_{\mu_i} \quad & -\frac{1}{2} \mu_i^T H_2 \mu_2 + F_2 \mu_2 \\
\text{subject to} \quad & A_2 \mu_2 = 0 \\
& 0 \leq \mu_i \leq \gamma_2 \nu_i,
\end{align*}$$

where $H_2 = \begin{bmatrix} \Omega_{22} + \gamma_1 \nu_i \n_i & \Omega_{21} \nu_i \\ \Omega_{21} \nu_i & \gamma_2 \nu_i \n_i \end{bmatrix}$, $\mu_2 = [\alpha_2^T, \beta_2^T]^T$, $F_2 = [0_{n_1}, 1_{n_1}^T]$, and $A_2 = [1_{n_1}, 1_{n_1}^T]$. It can be seen that the formulations (19) and (20) differ from those given in [12] by

$$\begin{align*}
\min_{w_1, b_1, \xi} \quad & \|\hat{\Omega}_1 w_1 + b_1 \|_2^2 + C_1 \xi \\
\text{subject to} \quad & -\hat{\Omega}_1 w_1 + b_1 \geq \xi \\
& \xi \geq 0,
\end{align*}$$

and

$$\begin{align*}
\min_{w_2, b_2, \xi} \quad & \|\hat{\Omega}_2 w_2 + b_2 \|_2^2 + C_2 \xi \\
\text{subject to} \quad & \hat{\Omega}_2 w_2 + b_2 \geq \xi \\
& \xi \geq 0,
\end{align*}$$

where $\hat{\Omega}_1$ and $\hat{\Omega}_2$ are $n_1 \times (n_1 + n_2)$ and $n_2 \times (n_1 + n_2)$ matrices respectively. $\hat{\Omega}_1 = K(x_i, x_j)$ with $x_i \in I_1$ and $x_j \in I_1 \cup I_2$ and $\hat{\Omega}_2 = K(x_i, x_j)$ with $x_i \in I_2$ and $x_j \in I_1 \cup I_2$, and $K$ is the kernel function. $I_1$ and $I_2$ 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 $w_i^T w_i$ and $w_i^T w_i$. 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 $L_{12}(\cdot)$ and $L_{22}(\cdot)$, 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., $\epsilon$-insensitive zone loss for $L_{12}(\cdot)$, and hinge loss for $L_{22}(\cdot)$, 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 $L_{12}(\cdot)$ to cope with the noise or outliers in the data set.

### 3.3: Case: LS-Pinball

When the hinge loss is minimized, the distance that we maximize is related to the nearest points which is prone to being 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:

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

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,

$$\begin{align*}
\min_{w_1, b_1, \xi} \quad & \frac{1}{2} w_1^T w_1 + \gamma_1 \xi^2 e + \gamma_2 \xi^T \nu_i \\
\text{subject to} \quad & \Phi_{w_1} + b_1 \|_1 \geq \mu_1 \\
& \nu_i \left[ \Phi_{w_1} + b_1 \|_1 \right] - \frac{1}{r} \xi \leq 1_{n_1},
\end{align*}$$

and

$$\begin{align*}
\min_{w_2, b_2, \xi} \quad & \frac{1}{2} w_2^T w_2 + \gamma_1 \xi^2 e + \gamma_2 \xi^T \nu_i \\
\text{subject to} \quad & \Phi_{w_2} + b_2 \|_1 \geq \mu_2 \\
& \nu_i \left[ \Phi_{w_2} + b_2 \|_1 \right] - \frac{1}{r} \xi \leq 1_{n_1}.
\end{align*}$$

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

$$\begin{align*}
\max_{\mu_1} \quad & -\frac{1}{2} \mu_1^T H_1 \mu_1 + F_1 \mu_1 \\
\text{subject to} \quad & A_1 \mu_1 = 0 \\
& -r \gamma_2 \|_1 \mu_1 \leq \beta_1 \leq \gamma_2 \|_1 \mu_1,
\end{align*}$$

and that of (24) is

$$\begin{align*}
\max_{\mu_2} \quad & -\frac{1}{2} \mu_2^T H_2 \mu_2 + F_2 \mu_2 \\
\text{subject to} \quad & A_2 \mu_2 = 0 \\
& -r \gamma_2 \|_1 \mu_2 \leq \beta_2 \leq \gamma_2 \|_1 \mu_2.
\end{align*}$$

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 $r$ value, the pinball loss can bring noise-insensitivity to feature noise and stability to re-sampling. (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 $\ell_{1}$ loss for $L_{1}(\cdot)$ and $L_{2}(\cdot)$. Then a suitable parameter will lead (1) becomes pin-SVM [21] with $r = 1$. 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>
<thead>
<tr>
<th>Type of noise</th>
<th>LS-LS</th>
<th>LS-Hinge</th>
<th>LS-Pinball</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label noise</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Feature noise</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

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 $D = \{x_i\}_{i=1}^{N}$, labels $\{y_i\}_{i=1}^{N}$, the tuning parameters (if any)

**Output:** Class membership of test data points $D_{test}$

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 $\gamma$. Then the obtained $\gamma$ is fed into the LS-LS formulation as $\gamma_{1}$ and regularization parameter $\gamma_{2}$ is set to $\frac{\gamma_{1}}{2}$.

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 $\gamma_{1}/\gamma_{2}$, 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 $\gamma_{1}, \gamma_{2}$, kernel bandwidth $\sigma$, and in the case of pinball loss the parameter $r$, 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].
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 $r = 10000$.

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Monk1</td>
<td>no noise</td>
<td>0.77 0.81 0.91 0.96 0.77</td>
<td>label 0.78 0.79 0.79 0.78 0.78</td>
<td>feature 0.72 0.73 0.72 0.72 0.64</td>
<td>both 0.71 0.71 0.72 0.71 0.73</td>
<td></td>
</tr>
<tr>
<td>Monk2</td>
<td>no noise</td>
<td>0.87 0.86 0.87 0.88 0.88</td>
<td>label 0.83 0.82 0.83 0.83 0.84</td>
<td>feature 0.71 0.70 0.71 0.70 0.72</td>
<td>both 0.69 0.72 0.72 0.70 0.71</td>
<td></td>
</tr>
<tr>
<td>Monk3</td>
<td>no noise</td>
<td>0.92 0.92 0.92 0.93 0.91</td>
<td>label 0.90 0.91 0.92 0.90 0.88</td>
<td>feature 0.85 0.85 0.87 0.83 0.81</td>
<td>both 0.84 0.84 0.86 0.84 0.80</td>
<td></td>
</tr>
<tr>
<td>Spect</td>
<td>no noise</td>
<td>0.74 0.76 0.77 0.84 0.81</td>
<td>label 0.77 0.78 0.75 0.77 0.77</td>
<td>feature 0.71 0.77 0.74 0.78 0.81</td>
<td>both 0.67 0.71 0.77 0.73 0.74</td>
<td></td>
</tr>
<tr>
<td>Ionosphere</td>
<td>no noise</td>
<td>0.94 0.94 0.94 0.94 0.93</td>
<td>label 0.93 0.93 0.94 0.94 0.93</td>
<td>feature 0.92 0.92 0.93 0.93 0.90</td>
<td>both 0.89 0.92 0.93 0.93 0.92</td>
<td></td>
</tr>
<tr>
<td>Heart</td>
<td>no noise</td>
<td>0.83 0.82 0.81 0.83 0.70</td>
<td>label 0.82 0.82 0.82 0.82 0.62</td>
<td>feature 0.86 0.85 0.85 0.85 0.54</td>
<td>both 0.82 0.82 0.82 0.83 0.63</td>
<td></td>
</tr>
<tr>
<td>Magic</td>
<td>no noise</td>
<td>0.78 0.79 0.79 0.78 0.59</td>
<td>label 0.78 0.78 0.78 0.79 0.50</td>
<td>feature 0.78 0.78 0.77 0.78 0.54</td>
<td>both 0.77 0.71 0.77 0.78 0.51</td>
<td></td>
</tr>
<tr>
<td>Spambase</td>
<td>no noise</td>
<td>0.88 0.91 0.91 0.91 0.50</td>
<td>label 0.89 0.90 0.90 0.90 0.50</td>
<td>feature 0.88 0.88 0.89 0.89 0.51</td>
<td>both 0.86 0.86 0.88 0.88 0.50</td>
<td></td>
</tr>
</tbody>
</table>

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-DATA4DRIVE-B (250923). 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 FPV/10/002 (OPTEC), BIL/12/11T, PhD/Postdoc grants • Flemish Goverment: • FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensord 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 (DYSICO, Dynamical systems, control and optimization, 2012-2017). Johan Seykens is a professor at the KU Leuven, Belgium.
References


