Chapter 6

$\overline{\text{pin}}$-TSVM: A robust transductive support vector machine and its application to the detection of COVID-19 infected patients

Training a machine learning model on the data sets with missing labels is a challenging task. Not all models can handle the problem of missing labels. However, if these data sets are further corrupted with label noise, it becomes even more challenging to train a machine learning model on such data sets. In this chapter, a semi-supervised learning framework is discussed in contrast to the previous two chapters which contributed in supervised machine learning.

6.1 Introduction

In all the above-discussed works, it is assumed that all the class labels are available during the training of a model, i.e., they come under supervised machine learning. The assignment of labels during data set creation is one of the costly and error-prone tasks.

Therefore, in practice, we often come across data sets with missing labels. Training the models over such data sets comes under the category of semi-supervised learning. Transductive support vector machine (TSVM) is a semi-supervised variant of SVM [124]. It was first proposed in [125] and implemented in [26]. There are various applications in which TSVM is used for learning purposes when there are some unlabeled data samples. The survey article [126] best describes the rich literature of TSVM.

Similar to SVM, TSVM is also sensitive to the label noise. This is due to the presence of a noise-sensitive loss function, e.g., the hinge loss function. The novelty of the present study lies in the fact that the use of truncated pinball loss function with TSVM was proposed. The corresponding optimization problem was solved by implementing both the primal and dual forms.

Next, in Subsection 6.1.1, the conventional TSVM and the existing robust TSVMs are described. Robust TSVM handles noise sensitivity. In Subsection 6.1.2, the motivation behind this work are mentioned and the main contributions of this work are described.

6.1.1 A Brief Introduction of TSVM

For a set $\mathcal{L} = \{(x_1, y_1), \ldots, (x_L, y_L)\}, x \in \mathbb{R}^d, y \in \{+1, -1\}$ of L labeled training instances and U unlabeled instances $\mathcal{U} = \{x_{L+1}, \ldots, x_{L+U}\}$, the task is to find an optimal separating hyperplane defined by $\theta = (w, b)$, where w is the weight vector and b is the bias term. The decision function of the form

$$f_\theta(x) = w^T \phi(x) + b \tag{6.1}$$

is used to label new samples, where the kernel function, ϕ, maps the original data into a higher dimensional feature space.

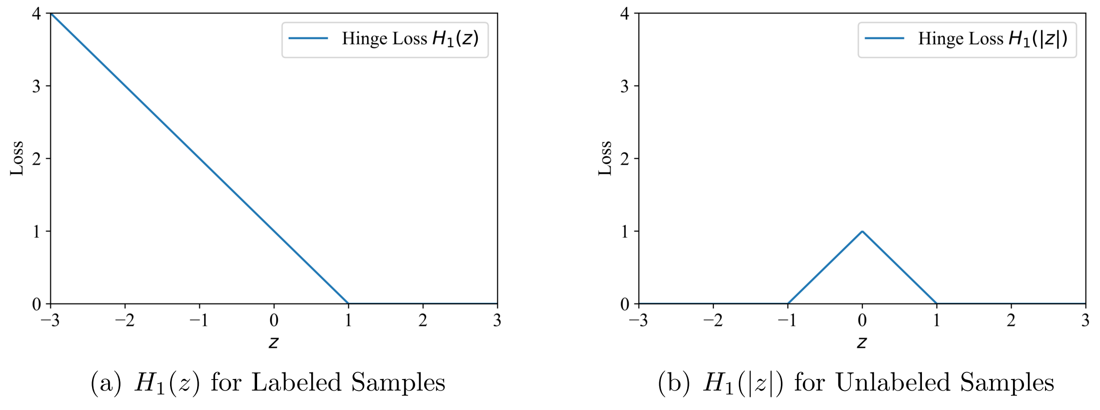
SVM is trained using $\mathcal{L}$ and the trained SVM provides the best separating hyperplane with the largest possible margin. It then assigns the labels to the U unlabeled instances of the set $\mathcal{U}$. TSVM is a combinatorial classifier of SVM and a constraint that the unlabeled samples should be as far as possible from the margin [125]. The optimization formulation corresponding to this combinatorial problem is

$$\begin{aligned} \underset{w,b}{\arg\min} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+U}\xi_i \\ \text{subject to} \quad & y_i(f_\theta(x_i)) \geq 1-\xi_i, \;\; i=1,\dots,L, \\ & |f_\theta(x_i)| \geq 1-\xi_i, \;\; i=L+1,\dots,L+U, \end{aligned} \tag{6.2}$$

where C and C^* are the weight controlling parameters corresponding to the labeled and unlabeled instances. The minimization problem (6.2) can be written as an unconstrained optimization problem of the form [26]

$$J(\theta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}H_1(y_i f_\theta(x_i)) + C^*\sum_{i=L+1}^{L+U}H_1(|f_\theta(x_i)|), \tag{6.3}$$

$H_1(z) = \max\{0, 1-z\}$ is the hinge loss function [1], $z = yf_\theta(x)$. In TSVM, $H_1(z)$ is used for the labeled samples while $H_1(|z|)$ is used for the unlabeled samples. These are shown in Figure 6.1. The TSVM has the limitation of assigning all the unlabeled



(a) $H_1(z)$ for Labeled Samples (b) $H_1(|z|)$ for Unlabeled Samples

Figure 6.1: Hinge Loss Function for Labeled and Unlabeled Samples

samples to one of the classes, leading to abysmal accuracy. To solve this problem, Chapelle and Zien [127] used a significant relaxed balancing constraint:

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(x_i) = \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{6.4}$$

TSVM is used in many applications, like cancer classifications [128], classification of mammographic abnormalities [129], glaucoma classification [130], image retrieval [131], ship category recognition [132], etc.

Li et al. [133] proposed a robust TSVM for multi-view classification. They observed that the multi-view representation of data from a different perspective could effectively improve the generalization performance [133]. Training a model on huge data sets is a tedious task. Training on small labeled sets is also challenging since the model has insufficient learning instances. Xu et al. [134] proposed an improved version of TSVM that can learn a small labeled training set well and applied this to the motor imagery based brain-computer interface.

Besides these, there are also many formulations in which researchers added robustness to the conventional TSVM by changing the loss functions. These methods are tabulated in Table 6.1. This table mentions the loss function for labeled and unlabeled samples that are used by various researchers to make the TSVM robust to noise. These loss functions include the conventional hinge loss function, symmetric sigmoid loss function [135] and the ramp loss function [47].

Table 6.1: Literature Survey of TSVM robust towards noise

S. No.	Reference	For labeled samples	For unlabeled samples
1	Semi-supervised SVM [125]	Hinge Loss	Symmetric Hinge Loss
2	Transductive Inference [136]	Hinge Loss	Symmetric Hinge Loss
3	Semi-supervised Classification by low density separation [127]	Hinge Loss	Symmetric Sigmoid Loss
4	Large Scale Transductive SVM [137]	Hinge Loss	Symmetric Ramp Loss
5	Large-scale Robust Transductive SVMs [63]	Ramp Loss	Symmetric Ramp Loss

A recent work on TSVM proposed to address its problem with Universum data [138]. In that work, Xiao et al. followed two steps: to select informative examples from the

Universum data and to use that data for semi-supervised classification [138]. They used Lagrange method to solve it further. Another recent work on TSVM handled the problem of lack of sparsity in LapSVM [139]. To do this, Zheng et al. [140] used L_1 norm in LapSVM. The method performed well (in terms of accuracy) on UCI data sets. Recently, SSL is also extended to various applications like fault identification in electricity distribution networks [141], for intrusion detection system [142] and enhanced prediction of heart disease [143].

In this work, the following three challenges were focused:

(i) To train the model in the presence of a significant number of unlabeled data.

(ii) To train the model to handle small as well as large data sets effectively.

(iii) To train the model under the varied amount of label noise in the data.

6.1.2 Motivation and Contribution

The robust behavior of the pinball loss function [57] is the primary motivation behind this work. The use of pinball loss function in other variants of SVM, like TWSVM [65], also made the model robust towards label noise. Since the use of pinball loss function affects the sparsity of a classifier [57], the truncated pinball loss function was used in this work, which leads to less computational time (shown experimentally in Section 6.3). The main contributions of this work are listed below:

(i) The truncated pinball loss function was used in place of the conventional hinge loss in TSVM. This made the model robust towards the label noise.

(ii) The use of the truncated pinball loss function made the model sparse, hence required fewer variables to contribute in the decision-making process of the model. This reduced the computational time of the model.

(iii) The proposed model was used for the detection of disease caused by the coronavirus in a human body. To do this, the model was trained on the chest X-ray images. The results (shown in Section 6.4) indicated that the model can be used to predict if a patient is infected by coronavirus or not.

6.1.3 Outline

In the next section, the proposed robust TSVM with a truncated pinball loss function is described. In Section 6.3, the results of the experiments performed using the proposed approach are reported and compared with the existing approaches. Further, in Section 6.4, it is shown that the proposed approach can be used to predict the COVID-19 infected patients. Finally, the work is concluded in Section 6.5.

6.2 $\overline{\text{pin}}$-TSVM: A Robust Transductive Support Vector Machine with Truncated Pinball Loss Function

In this section, the robust TSVM formulation using truncated pinball loss function is described. The truncated pinball loss function is

$$P_{\tau,s}(z) = H_{1+\tau}(z) - (H_\tau(z+s) - \tau s) = \begin{cases} \tau s, & \text{if} \quad z \geq 1+s \\ -\tau(1-z), & \text{if} \quad 1 < z < 1+s \\ 1-z, & \text{if} \quad z \leq 1 \end{cases} \tag{6.5}$$

where $0 \leq \tau \leq 1$. It is shown in Figure 6.2. Please note that $s > 0$ is the hinge point [57].

In (6.3), the hinge loss function was replaced by the truncated pinball loss function.

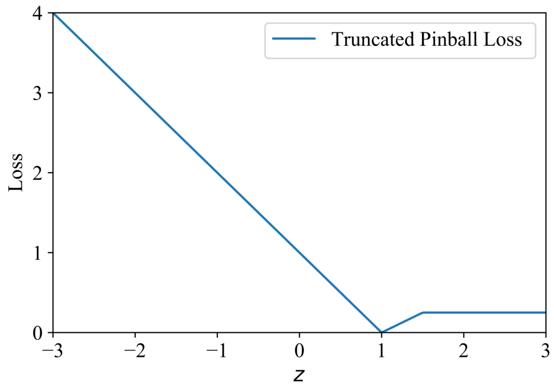


Figure 6.2: Truncated Pinball Loss Function with $\tau = s = 0.5$

Accordingly, $J(\theta)$ in (6.3) became

$$J(\theta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} P_{\tau,s}(z) + C^* \sum_{i=L+1}^{L+U} P_{\tau,s}(z). \tag{6.6}$$

To avoid the poor classification of the unlabeled samples, the same constraint as described earlier in (6.4) was used. On putting in (6.6), we get

$$\begin{aligned} J(\theta) =& \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} (H_{1+\tau}(y_i f_\theta(x_i)) - H_\tau(y_i f_\theta(x_i) + s) - \tau s) \\ &+ C^* \sum_{i=L+1}^{L+U} (H_{1+\tau}(y_i f_\theta(x_i)) - H_\tau(y_i f_\theta(x_i) + s) - \tau s). \end{aligned} \tag{6.7}$$

Now, each unlabeled sample is represented as two instances labeled with both positive and negative classes. This leads to the creation of new samples [63]

$$\begin{aligned} &y_i = +1, \ \ i \in [L+1, \ldots, L+U], \\ &y_i = -1, \ \ i \in [L+U+1, \ldots, L+2U], \\ &x_i = x_{i-U}, \ \ i \in [L+U+1, \ldots, L+2U]. \end{aligned} \tag{6.8}$$

This function was split into convex ($J_{\text{convex}}(\theta)$) and concave ($J_{\text{concave}}(\theta)$) parts [57]:

$$J_{\text{convex}}(\theta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} H_{1+\tau}(y_i f_\theta(x_i)) + C^*\sum_{i=L+1}^{L+2U} H_{1+\tau}(y_i f_\theta(x_i)) \tag{6.9}$$

and $$J_{\text{concave}}(\theta) = -C\sum_{i=1}^{L} H_\tau(y_i f_\theta(x_i)+s) + CL\tau s - C^*\sum_{i=L+1}^{L+2U} H_\tau(y_i f_\theta(x_i)+s) + C^*(2U)\tau s. \tag{6.10}$$

To perform the minimization of $J(\theta)$ with respect to $\theta = (w, b)$, the CCCP [144], as given by Algorithm 3, was used. CCCP decomposes the non-convex function into a concave and a convex part. It uses an iterative procedure where in each iteration, concave part is approximated by its tangent [63]. In Algorithm 3, $J'(\theta)$ represents $\frac{\partial J(\theta)}{\partial\theta}$. The convergence of CCCP algorithm is given in [144].

Algorithm 3 The Concave-Convex Procedure (CCCP) [144]

```
Input: $J_{\text{concave}}(\theta)$ and $J_{\text{convex}}(\theta)$
1: Initialize $\theta^0$.
2: repeat
3:     $\theta^{t+1} = \arg\min_\theta \left(J_{\text{convex}}(\theta) + J'_{\text{concave}}(\theta^t)\theta\right)$
4: until the convergence of $\theta^t$.
```

Next, the gradient of $J_{\text{concave}}(\theta)$ with respect to θ was computed

$$\Delta_\theta J_{\text{concave}}(\theta) = \frac{\partial}{\partial\theta}J_{\text{concave}}(\theta) = -C\sum_{i=1}^{L}\left(\frac{\partial H_\tau(\theta)}{\partial f_\theta(x_i)}\right)\left(\frac{\partial f_\theta(x_i)}{\partial\theta}\right) - C^*\sum_{i=L+1}^{L+2U}\left(\frac{\partial H_\tau(\theta)}{\partial f_\theta(x_i)}\right)\left(\frac{\partial f_\theta(x_i)}{\partial\theta}\right).$$

Now, $$\frac{\partial H_\tau(\theta)}{\partial\theta} = \tau\cdot\frac{\partial f_\theta(x_i)}{\partial\theta}\cdot(-y_i).$$

Therefore, $$\begin{aligned}\frac{\partial}{\partial\theta}J_{\text{concave}}(\theta) &= -C\sum_{i=1}^{L}\tau y_i\frac{\partial f_\theta(x_i)}{\partial\theta} - C^*\sum_{i=L+1}^{L+2U}\tau y_i\frac{\partial f_\theta(x_i)}{\partial\theta}\\ &= \sum_{i=1}^{L+2U}\beta_i y_i\frac{\partial f_\theta(x_i)}{\partial\theta},\end{aligned} \tag{6.11}$$

where

$$-\beta_i = \begin{cases} C\tau, & \text{if } 1 \leq i \leq L \\ C^*\tau, & \text{if } L+1 \leq i \leq L+2U. \end{cases} \tag{6.12}$$

Therefore, the problem (6.7) can now be stated [144] as the minimization of

$$\begin{aligned} & J_{\text{convex}}(\theta) + \frac{\partial J_{\text{concave}}(\theta)}{\partial\theta} \\ &= J_{\text{convex}}(\theta) + \left(\sum_{i=1}^{L+2U} \beta_i y_i \frac{\partial f_\theta(x_i)}{\partial\theta}\right)\theta \\ &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} H_{1+\tau}(y_i f_\theta(x_i)) + C^*\sum_{i=L+1}^{L+2U} H_{1+\tau}(y_i f_\theta(x_i)) + \sum_{i=1}^{L+2U} \beta_i y_i f_\theta(x_i). \end{aligned} \tag{6.13}$$

By introducing the slack variable, ξ in (6.13), the final minimization problem [57] obtained is

$$\begin{aligned} \min_{\theta,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=1}^{L+2U}\beta_i y_i f_\theta(x_i) \\ \text{subject to} \quad & y_i f_\theta(x_i) \geq 1 - \frac{1}{1+\tau}\xi_i, \quad \xi_i \geq 0, \quad i = 1,2,\ldots,L, \\ & \frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(x_i) = \frac{1}{L}\sum_{i=1}^{L} y_i. \end{aligned} \tag{6.14}$$

The final optimization problem (6.14) was solved by using SGD method [145] given in Algorithm 4. To implement (6.14) using SGD, data set, $D = \{x_i, y_i\}_{i=1}^{L+U}$ was required from which we get the value of L and U. Another inputs like λ, the learning rate of SGD and ϵ, the tolerance value were also required in the convergence of Algorithm 4.

Since the CCCP algorithm converged fast [63] in maximum five iterations in the experiments, wtherefore, $T = 5$ was considered in all the algorithms. However, the convergence conditions are also mentioned in the algorithms (Step 12 in Algorithm 4, Step 13 in Algorithm 5 and Algorithm 6).

The time complexity of Algorithm 4 is mainly due to the Step 2 and the conventional steps of SGD (Step 8 and Step 9). Step 2 was executed using the *svmtrain*() (LIBSVM) which has a time complexity of $O(n^3)$ [146]. However, the time complexity of SGD is $O(\bar{d}/\lambda\epsilon)$ [63], where $\bar{d}$ is used for the non-zero attributes of the data set, λ is the learning rate of SGD and ϵ is the tolerance value. Therefore, the overall time complexity of Algorithm 4 is $O(n^3) + O(\bar{d}/\lambda\epsilon)$, i.e. $O(n^3)$. The proposed approach, $\overline{\text{pin}}$-TSVM, was also implemented using the dual form of (6.14).

Algorithm 4 $\overline{\text{pin}}$-TSVM with Stochastic Gradient Method to Get Optimal Weight Vector and Bias Term

```
Input: $D = \{x_i, y_i\}_{i=1}^{L+U}$;
    $T$ is the maximum number of iterations;
    $\epsilon$ is the tolerance value;
    $L$ is the number of labeled instances in the data set $D$;
    $U$ is the number of unlabeled instances in the data set $D$;
    $\lambda_0 > 0$ is the learning rate of SGD;
Output: Optimal weight vector and bias term, $w_t$ and $b_t$ respectively.
1: Split the data set $D$ into training set and test set.
2: Train SVM on training set and get $w_0$ and $b_0$.
3: Initialize $t = 0$ and $\epsilon > 0$.
4: Compute $\beta_i^0$ for $i = 1, 2, \ldots, (L + 2U)$ using (6.12).
5: while $t \leq T$ do
6:     $\lambda_t \leftarrow \frac{\lambda_0}{t}$.
7:     for $i \in$ randperm$(L + 2U)$ do
8:         Compute the sub-gradient of (6.13) w.r.t $w$ and $b$
               $g_t = \begin{cases} -y_i(1+\tau)C(C^*)x_i + \beta_i y_i x_i, & \text{if } y_i(w^T x_i + b) \leq 1, \\ \beta_i y_i x_i, & \text{if } y_i(w^T x_i + b) > 1 \end{cases}$
           and
               $h_t = \begin{cases} -y_i(1+\tau)C(C^*) + \beta_i y_i, & \text{if } y_i(w^T x_i + b) \leq 1, \\ \beta_i y_i, & \text{if } y_i(w^T x_i + b) > 1. \end{cases}$
9:         Update parameters
               $\hat{w_t} \leftarrow w_t - \frac{\lambda_t}{L + 2U}(w_t + g_t)$
           and
               $\hat{b_t} \leftarrow b_t - \frac{\lambda_t}{L + 2U} h_t.$
10:        Set $w_t \leftarrow \hat{w_t}$ and $b_t \leftarrow \hat{b_t}$
11:    end for
12:    if $(t \geq 2)$ & $\|w_t - w_{t-1}\| \leq \epsilon$, break
13:    Compute $\beta_i^{t+1}$ using (6.12).
14:    Set $t = t + 1$.
15: end while
```

To get the dual form of (6.14), the Lagrangian function was obtained

$$\begin{aligned}
L(w,b,\xi,\alpha,\nu) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i \\
&+ \sum_{i=1}^{L+2U}\beta_i y_i f_\theta(x_i) - \alpha_0\left(\frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(x_i) - \frac{1}{L}\sum_{i=1}^{L} y_i\right) \\
&- \sum_{i=1}^{L+2U}\alpha_i\left(y_i(w^T x_i + b) - 1 + \frac{1}{1+\tau}\xi_i\right) - \sum_{i=1}^{L+2U}\nu_i\xi_i, \qquad (6.15)
\end{aligned}$$

where $\alpha_i, \nu_i \geq 0$, for $i = 1, 2, \ldots, (L+2U)$. The necessary Karush-Kuhn Tucker (KKT) optimality conditions for (6.15) are

$$\frac{\partial L}{\partial w} = w + \sum_{i=1}^{L+2U}\beta_i y_i \phi(x_i) - \frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\phi(x_i) - \sum_{i=1}^{L+2U}\alpha_i y_i \phi(x_i) = 0, \qquad (6.16)$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{L+2U}\beta_i y_i + \alpha_0 + \sum_{i=1}^{L+2U}\alpha_i y_i = 0 \qquad (6.17)$$

$$\frac{\partial L}{\partial \xi} = C - \frac{\alpha_i}{1+\tau} - \nu_i = 0, \quad 1 \leq i \leq L, \qquad (6.18)$$

$$\frac{\partial L}{\partial \xi} = C^* - \frac{\alpha_i}{1+\tau} - \nu_i = 0, \quad L+1 \leq i \leq L+2U. \qquad (6.19)$$

For simplification, a new sample was defined as

$$\phi(x_0) = \frac{1}{U}\sum_{i=L+1}^{L+U}\phi(x_i), \quad y_0 = 1. \qquad (6.20)$$

From (6.16), we get

$$\begin{aligned}
w &= \frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\phi(x_i) + \sum_{i=1}^{L+2U}\alpha_i y_i \phi(x_i) - \sum_{i=1}^{L+2U}\beta_i y_i \phi(x_i) \\
&= \alpha_0\phi(x_0) + \sum_{i=1}^{L+2U} y_i\phi(x_i)(\alpha_i - \beta_i) \\
&= \sum_{i=0}^{L+2U} y_i\phi(x_i)(\alpha_i - \beta_i), \qquad (6.21)
\end{aligned}$$

where $\beta_0 = 0$ and $y_0 = 1$. On putting the value of (6.21) in (6.15), we get

$$\begin{aligned}
L(b,\xi,\alpha,\nu) =& \frac{1}{2}\left(\sum_{i=0}^{L+2U} y_i\phi(x_i)(\alpha_i-\beta_i)\right)^T \left(\sum_{j=0}^{L+2U} y_j\phi(x_j)(\alpha_j-\beta_j)\right) \\
&+C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=1}^{L+2U}\beta_i y_i\left[\left(\sum_{j=0}^{L+2U} y_j\phi(x_j)(\alpha_j-\beta_j)\right)^T\phi(x_i)+b\right] \\
&-\alpha_0\left(\frac{1}{U}\sum_{i=L+1}^{L+U}\left[\left(\sum_{j=0}^{L+2U} y_j\phi(x_j)(\alpha_j-\beta_j)\right)^T\phi(x_i)+b\right]-\frac{1}{L}\sum_{i=1}^{L}y_i\right) \\
&-\sum_{i=1}^{L+2U}\alpha_i\left(y_i\left(\left[\sum_{j=0}^{L+2U} y_j\phi(x_j)(\alpha_j-\beta_j)\right]^T\phi(x_i)+b\right)-1+\frac{1}{1+\tau}\xi_i\right)-\sum_{i=1}^{L+2U}\nu_i\xi_i.
\end{aligned}$$

On simplification,

$$\begin{aligned}
L(b,\xi,\alpha,\nu) =& \frac{1}{2}\sum_{i,j=0}^{L+2U} y_iy_j(\alpha_i-\beta_i)(\alpha_j-\beta_j)\phi(x_i)^T\phi(x_j) + \sum_{i=1}^{L+2U}\beta_i y_i\left(\sum_{j=0}^{L+2U} y_j(\alpha_j-\beta_j)\phi(x_j)\right)\phi(x_i) \\
&+b\sum_{i=1}^{L+2U}\beta_i y_i - \frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\left(\sum_{j=0}^{L+2U} y_j(\alpha_j-\beta_j)\phi(x_j)\right)\phi(x_i)-\frac{\alpha_0}{U}b \\
&+\frac{\alpha_0}{L}\sum_{i=1}^{L}y_i - \sum_{i=1}^{L+2U}\alpha_i y_i\left(\sum_{j=0}^{L+2U} y_j\phi(x_j)(\alpha_j-\beta_j)\right)\phi(x_i) \\
&-b\sum_{i=1}^{L+2U}\alpha_i y_i + \sum_{i=1}^{L+2U}\alpha_i - \sum_{i=1}^{L+2U}\frac{\alpha_i}{1+\tau}\xi_i - \sum_{i=1}^{L+2U}\nu_i\xi_i \\
&+C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i. \qquad (6.22)
\end{aligned}$$

Now, adding and subtracting $\alpha_0 y_0\left(\sum_{j=0}^{L+2U} y_j\phi(x_j)(\alpha_j-\beta_j)\right)\phi(x_0)$ from (6.22), we get

$$\begin{aligned}
L(b,\xi,\alpha,\nu) =& -\frac{1}{2}\sum_{i,j=0}^{L+2U} y_iy_j(\alpha_i-\beta_i)(\alpha_j-\beta_j)\phi(x_i)^T\phi(x_j) \\
&+\sum_{i=1}^{L+2U}\beta_i y_i\left(\sum_{j=0}^{L+2U} y_j(\alpha_j-\beta_j)\phi(x_j)\phi(x_i)\right)+b\sum_{i=1}^{L+2U}\beta_i y_i
\end{aligned}$$

$$-\frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\left(\sum_{j=0}^{L+2U} y_j(\alpha_j-\beta_j)\phi(x_j)\right)\phi(x_i)-\frac{\alpha_0}{U}b$$
$$+\frac{\alpha_0}{L}\sum_{i=1}^{L} y_i - b\sum_{i=1}^{L+2U}\alpha_i y_i + \sum_{i=1}^{L+2U}\alpha_i$$
$$+\alpha_0 y_0\left(\sum_{j=0}^{L+2U} y_j(\alpha_j-\beta_j)\phi(x_j)\right)\phi(x_0). \tag{6.23}$$

Simplifying (6.23) using (6.18) and (6.20), we get

$$\begin{aligned}
\min_{\alpha}\quad & \frac{1}{2}\sum_{i,j=0}^{L+2U} y_i y_j(\alpha_i-\beta_i)(\alpha_j-\beta_j)\phi(x_i)^T/\phi(x_j)\\
& -\frac{\alpha_0}{L}\sum_{i=1}^{L} y_i - \sum_{i=1}^{L+2U}\alpha_i\\
\text{subject to}\quad & \sum_{i=0}^{L+2U} y_i(\alpha_i-\beta_i)=0,\\
& 0\le\alpha_i\le(1+\tau)C,\quad 1\le i\le L,\\
& 0\le\alpha_i\le(1+\tau)C^*,\quad L+1\le i\le L+2U.
\end{aligned}\tag{6.24}$$

Considering the kernel matrix, K such that $K_{ij}=\phi(x_i)^T/\phi(x_j)$ and $\hat{\alpha_i}=y_i(\alpha_i-\beta_i)$, we get the final dual problem as

$$\begin{aligned}
\min_{\hat{\alpha}}\quad & \frac{1}{2}\hat{\alpha}K\hat{\alpha}-\gamma^T\hat{\alpha}\\
\text{subject to}\quad & 0\le y_i\hat{\alpha_i}\le(1+\tau)C,\quad i=1,2,\ldots,L\\
& -\beta_i\le y_i\hat{\alpha_i}\le(1+\tau)C^*-\beta_i,\quad i=L+1,L+2,\ldots L+2U,\\
& \sum_{i=0}^{L+2U}\hat{\alpha_i}=0,
\end{aligned}\tag{6.25}$$

where $\gamma=y_i$ for $1\le i\le L+2U$ and $\gamma_0=\frac{1}{L}\sum_{i=1}^{L}y_i$. To solve (6.25), Algorithm 5 was followed to find the optimal weight vector and bias term. Next, the weight vector and the bias term were used to find the sign(w^Tx+b) in case of linear vector. In Algorithm

5, the SVM was first trained using *svmtrain*() function using LIBSVM whose time complexity is $O(n^3)$, where n represents the number of instances in a data set [146], and then *mlcv_quadprog*() [63] was used to implement Step 7 in Algorithm 5, whose time complexity is again $O(n^3)$ [147]. Therefore, the overall time complexity of the Algorithm 5 is $O(n^3)$.

Similarly, the dual optimization problem given in (6.25) can also be solved using the non-linear kernels. The Algorithm 6 shows the steps to be followed for non-linear kernels.

Algorithm 5 $\overline{\text{pin}}$-TSVM to get optimal weight vector and bias term (Linear Kernel)

```
Input: $D = \{x_i, y_i\}_{i=1}^{L+U}$;
    $T$ is the maximum number of iterations;
    $\epsilon_1$ and $\epsilon_2$ are the tolerance values;
    $L$ is the number of labeled instances of data set $D$;
    $U$ is the number of unlabeled instances of data set $D$;

Output: Optimal weight vector and bias term, $w_t$ and $b_t$ respectively.
1: Split the data set $D$ into training set and test set .
2: Train SVM on training set and get $w_0$ and $b_0$.
3: Initialize $t = 0$ and $\epsilon_1, \epsilon_2 > 0$.
4: Compute $\beta_i^0$ using (6.12).
5: Set $\gamma_i = y_i$ for $1 \le i \le L + 2U$ and $\gamma_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$.
6: while $t \le T$ do
7:     Solve the convex optimization problem given by (6.25).
8:     Compute $w$ using
           $$w = \sum_{i=0}^{L+2U} y_i(\alpha_i - \beta_i)x_i.$$
9:     Set $w_{t+1} = w$.
10:    Compute $b$ using the following constraints;
           $$\forall i \in \{1, \dots, L\}, 0 \le \alpha_i \le C\tau \implies y_i(w^T x_i + b) = 1,$$
       or
           $$\forall i \in \{L+1, \dots, L+2U\}, 0 \le \alpha_i \le C^*\tau \implies y_i(w^T x_i + b) = 1,$$
11:    Set $b_{t+1} = b$.
12:    Compute $\beta_i^{t+1}$ using (6.12).
13:    if $\Big((t \ge 2)$ & $(\|w_{t+1} - w_t\| \le \epsilon_1$ or $\left\|\beta^{t+1} - \beta^t\right\| \le \epsilon_2)\Big)$, break
14:    Set $t = t + 1$.
15: end while
```

It is noteworthy that the time complexity of both the algorithms, Algorithm 5 and Algorithm 6 is same as *mlcv_quadprog*() [63] was used to implement both the

Algorithm 6 $\overline{\text{pin}}$-TSVM to get Accuracy using Non-linear Kernel

```
Input: $D = \{x_i, y_i\}_{i=1}^{L+U}$;
    $T$ is the maximum number of iterations;
    $\epsilon_1$ and $\epsilon_2$ are the tolerance values;
    $L$ is the number of labeled instances of data set $D$;
    $U$ is the number of unlabeled instances of data set $D$;

Output: Accuracy
1: Split the data set $D$ into training set and test set.
2: Choose a non-linear kernel.
3: Compute kernel matrix, $K$ using the training features.
4: Train SVM on training set and get $\alpha_0$, $b_0$ and support vectors, $SV_{\text{initial}}$.
5: Initialize $t = 0$.
6: Compute $\beta_i^0$ using (6.12).
7: Set $\gamma_i = y_i$ for $1 \leq i \leq L + 2U$ and $\gamma_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$.
8: while $t \leq T$ do
9:     Solve the convex optimization problem (6.25) using $\alpha_0$, $b_0$ and $SV_{\text{initial}}$. Find $\hat{\alpha}$ and support vectors, $S$.
10:    Compute $b$ using the following constraints

           $\forall i \in \{1, \dots, L\}, 0 \leq \hat{\alpha_i} \leq C\tau \implies y_i(K * \hat{\alpha_i} + b) = 1,$

       or

           $\forall i \in \{L+1, \dots, L+2U\}, 0 \leq \hat{\alpha_i} \leq C^*\tau \implies y_i(K * \hat{\alpha_i} + b) = 1,$

11:    Set $b^{t+1} = b$ and $\hat{\alpha}^{t+1} = \hat{\alpha}$
12:    Compute $\beta_i^{t+1}$ using (6.12).
13:    if $\left((t \geq 2) \quad \& \quad \left(\left\|\hat{\alpha}^{t+1} - \hat{\alpha}^t\right\| \leq \epsilon_1 \text{ or } \left\|\beta^{t+1} - \beta^t\right\| \leq \epsilon_2\right)\right)$, break
14:    Set $t = t + 1$.
15: end while
16: Construct kernel matrix, $K'$ using test features and support vectors, $S$.
17: Evaluate $y_{\text{pred}} = \text{sign}(\hat{\alpha}K' + b)$.
18: Compute Accuracy by length(find($y_{\text{pred}}$ == test label))/length(test label)
```

algorithms.

6.3 Numerical Experiments

In this section, the results obtained by $\overline{\text{pin}}$-TSVM on various data sets are reported. The proposed model was compared with the standard SVM, TSVM and TSVM with ramp loss function (Ramp-TSVM). Firstly, the model performance was evaluated on synthetic data sets. Towards this direction, a two-dimensional synthetic data set of 100 samples was generated with 50 samples for both positive and negative classes. In this data set, a different amount of label noise was added to test the performance of the proposed method against the existing TSVM methods. To add $k\%$ noise to the data set, the $k\%$ labels of the labeled training data were switched from -1 to $+1$ and vice-versa. In this work, $k = 10, 15, 20$ and 25 were considered to add label noise in the synthetic data set. These results are reported in Table 6.2.

In Table 6.2, the best accuracies are marked in bold. Note that the labeled set

Table 6.2: Comparison of Various Techniques on Synthetic Data Set Using Linear Kernel

Methods	Noise-free Data	10% Noise	15% Noise	20% Noise	25% Noise
SVM	**94**	88	83	85	49
TSVM	93	86	85	88	44
Ramp-TSVM	93	**93**	90	91	**67**
$\overline{\text{pin}}$-TSVM-SG	93	**93**	**91**	**92**	**67**
$\overline{\text{pin}}$-TSVM-dual	**94**	87	84	88	45

Table 6.3: Small Data Sets Used for Experimentation Purposes

S. No.	Data Sets	Instances	Features	No. of Classes	Class Ratio
1	Sonar	208	61	2	3.00:1
2	Cleveland Heart	303	14	2	0.83:1
3	Haberman	306	4	2	0.36:1
4	WDBC	568	32	2	0.59:1
5	Australian	690	15	2	0.80:1
6	Pima Indians	738	9	2	0.74:1
7	CMC	1443	10	3	1.34:1
8	Spambase	4601	58	2	0.65:1

was used to train SVM since it is a supervised learning model. For the rest of the techniques, unlabeled test data was also used for training. It is noteworthy that for all the experiments, the weight adjusting parameters C and C^* from the set $\{1, 2, 3, 4, 5\}$ and $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, respectively, were used. The cross-validation was performed on 10% of the training data to optimally select the value of C and C^*. It is observed that the proposed method, $\overline{\text{pin}}$-TSVM-SG, shows better accuracy for most cases; however, the dual form of the proposed method lacks in terms of accuracy on this small synthetic data set. These experiments were performed on linear kernel. However, the algorithms for both linear and non-linear kernels (see Algorithm 4, 5 and 6) were provided. Similar trends were observed with other kernels as well.

6.3.1 Experiments on Real-world Data Sets

In this subsection, the performance of the existing TSVM techniques are compared with the proposed technique on real-world data sets. First, the experimental results on small real-world data sets are reported and then the techniques were evaluated on large real-world data sets. Small real-world data sets are listed in Table 6.3.

Table 6.4: Comparison of Various Techniques with 0% Noise in the Real-World Data Sets

Data Sets	Results	SVM	TSVM	Ramp-TSVM	$\overline{\text{pin}}$-TSVM-SG	$\overline{\text{pin}}$-TSVM-dual
Sonar	Accuracy	71.28±2.92	76.75±3.61	73.40±3.50	74.47±4.21	**78.72±3.14**
	Precision	0.8481	**0.8842**	0.8241	0.8335	0.8706
	Recall	0.8561	**0.8893**	0.8734	0.8861	0.8916
	Time	1.36	1.84	41.61	7.12	2.13
Cleveland	Accuracy	81.39±2.74	82.11±2.17	83.01±3.72	82.35±2.95	**83.09±2.14**
	Precision	0.8760	0.8828	0.8760	0.8795	**0.8829**
	Recall	0.9117	0.9239	0.9317	0.9333	**0.9339**
	Time	0.45	1.29	37.09	8.71	1.06
Haberman	Accuracy	72.46±2.01	75.36±9.61	72.46±2.24	73.46±2.14	**76.09±4.34**
	Precision	0.7246	0.8320	0.7246	0.7291	**0.8333**
	Recall	1	0.8889	1	1	1
	Time	0.9837	0.7631	12.216	1.26	0.8609
WDBC	Accuracy	96.05±1.34	96.88±1.36	93.75±1.14	94.92±1.31	**97.27±1.44**
	Precision	0.9760	0.9802	0.9836	0.9878	**0.9881**
	Recall	0.9643	**0.9880**	0.9524	0.9605	0.9842
	Time	0.0050	0.0861	109.71	18.285	1.326
Australian	Accuracy	85.91±1.94	84.84±1.57	84.52±1.72	**86.13±1.56**	84.84±0.88
	Precision	0.8926	0.9007	**0.9112**	0.9082	0.9007
	Recall	**0.9568**	0.9359	0.9193	0.9435	0.9359
	Time	0.0210	3.5915	72.052	19.958	2.068
Pima Indians	Accuracy	77.90±1.36	77.18±1.76	76.59±2.28	**78.61±2.05**	78.32±1.83
	Precision	0.9017	0.9100	0.9332	0.9351	**0.9400**
	Recall	0.8317	**0.8588**	0.7958	0.8344	0.8576
	Time	0.0113	1.109	141.874	22.357	1.292
CMC	Accuracy	66.16±2.78	66.38±1.63	65.95±1.98	64.87±1.39	**67.67±2.06**
	Precision	0.8016	0.8021	0.8407	0.8269	**0.8845**
	Recall	0.7912	0.7918	0.7537	0.7506	**0.7923**
	Time	0.5967	2.591	148.16	29.481	3.029
Spambase	Accuracy	89.76±1.03	90.40±1.12	89.54±0.53	89.69±0.26	**91.16±0.36**
	Precision	0.9712	0.9663	0.9696	**0.9766**	0.9762
	Recall	0.9221	0.8469	0.8601	0.8618	**0.9430**
	Time	0.7151	21.18	182.71	119.56	41.33

- *Description of the Data Sets*

In Table 6.3, the data sets are arranged in the increasing order of instances. A short description of each data set is given as follows:

Table 6.5: Comparison of Various Techniques with 15% Noise in the Real-World Data Sets

Data Sets	Results	SVM	TSVM	Ramp-TSVM	$\overline{\text{pin}}$-TSVM-SG	$\overline{\text{pin}}$-TSVM-dual
Sonar	Accuracy	62.77±3.48	69.15±2.61	59.67±8.23	63.24±4.11	**71.24±4.11**
	Precision	0.8551	0.8442	0.8235	**0.8587**	0.8571
	Recall	0.7024	0.7927	0.6829	0.7189	**0.7952**
	Time	0.0058	1.6075	38.22	7.74	1.42
Cleveland	Accuracy	80.62±3.61	80.15±2.66	77.49±1.50	78.68±3.14	**80.88±2.19**
	Precision	0.8505	0.8662	0.8346	0.8492	**0.8871**
	Recall	**0.9307**	0.8934	0.9011	0.9145	0.9016
	Time	0.0411	1.0062	39.83	8.998	1.311
Haberman	Accuracy	70.29±1.46	65.23±2.91	70.29±2.50	71.21±1.55	**73.91±1.62**
	Precision	0.7029	0.8182	0.7027	0.7313	**0.8361**
	Recall	0.8991	0.7627	0.8982	0.9011	**0.9264**
	Time	0.0027	0.7228	1.273	1.218	0.805
WDBC	Accuracy	92.97±1.02	94.53±1.13	91.02±1.34	91.80±1.20	**95.70±1.11**
	Precision	0.9123	0.9837	0.9957	**1**	0.9879
	Recall	0.9297	0.9603	0.9137	0.9180	**0.9684**
	Time	0.144	1.716	88.163	17.504	1.99
Australian	Accuracy	82.58±1.17	82.57±1.57	82.90±0.72	82.58±0.15	**83.61±0.17**
	Precision	0.8533	0.8534	0.8567	0.8633	**0.8717**
	Recall	0.9624	0.9642	0.9625	0.9624	**0.9732**
	Time	0.0226	2.0554	55.704	20.258	2.261
Pima Indians	Accuracy	71.25±3.64	73.12±1.53	71.54±1.53	72.83±3.12	**73.41±1.79**
	Precision	0.9214	0.8405	0.8729	**0.9265**	0.8469
	Recall	0.7645	0.7254	0.7404	0.7736	**0.8467**
	Time	0.0144	1.1704	132.14	22.274	1.506
CMC	Accuracy	65.73±2.16	65.19±2.67	66.81±2.17	**67.46±1.52**	66.16±2.21
	Precision	0.8356	0.8010	0.8517	0.8537	**0.8588**
	Recall	0.7550	0.7887	0.7961	0.7709	**0.7963**
	Time	0.0315	2.524	126.54	28.52	3.135
Spambase	Accuracy	87.10±1.16	89.08±1.32	60.95±1.05	89.30±0.89	**89.90±1.61**
	Precision	0.9609	0.9588	0.8106	**0.9808**	0.9568
	Recall	0.8943	0.9162	0.7842	0.8962	**0.9371**
	Time	1.0635	46.344	930.795	117.944	71.02

- **Sonar** [148]: To classify two types of sonar signals: one bounced off a roughly cylindrical rock and those sonar signals which bounced off a metal cylinder.

- **Cleveland Heart** [148]: To classify patients based on presence or absence of heart disease.
- **Haberman** [149]: Classification based on the survival status of patients (who died within 5 years or patients who survived 5 years or longer) who had undergone surgery for breast cancer.
- **WDBC** [148]: Features describe the characteristics of the nuclei of the cell present in the images. It classifies if the case is benign or malignant.
- **Australian** [148]: The task is to classify the applications approved for credit card.
- **Pima Indians**: Based on the women living in Arizona and the task is to classify them as diabetic or non-diabetic.
- **CMC** [148]: The task is to predict the contraceptive method choice of a woman based on her socio-economic and demographic characteristics.
- **Spambase** [148]: To classify an email as spam or non-spam.

To perform the experiments over these data sets, the data sets were divided into the ratio of 55:45, where 55% of data was used for training while the rest 45% of data was used for testing purposes. Only 55% labeled data was used to train the model. In this way, the performance of all the methods was tested with greater complexity. Therefore, 45% unlabeled data was used for training.

The SVM model was trained over the labeled training set (similar to the synthetic data set) using *svmtrain*() from LIBSVM [146]. The weight vector and the bias term obtained from training the SVM were then used to train other models. To check the robustness of the proposed model, noise was added to the labeled training data. In this part of the work, model's performance was evaluated by adding 0%, 15%, and 30% noise in the data sets. To add k% noise to the data set, the k% labels of the labeled training

Table 6.6: Comparison of Various Techniques with 30% Noise in the Real-World Data Sets

Data Sets	Results	SVM	TSVM	Ramp-TSVM	$\overline{\text{pin}}$-TSVM-SG	$\overline{\text{pin}}$-TSVM-dual
Sonar	Accuracy	61.70±3.31	62.06±4.57	57.45±6.98	58.51±6.77	**62.77±3.14**
	Precision	0.6988	**0.7754**	0.6207	0.6595	0.7708
	Recall	0.8406	0.7500	0.6471	0.7112	**0.8429**
	Time	0.0252	2.113	42.21	7.97	1.193
Cleveland	Accuracy	73.77±2.15	77.94±3.81	80.15±5.72	78.35±5.11	**79.41±3.32**
	Precision	0.8337	0.8669	0.8761	**0.9106**	0.8926
	Recall	0.8121	0.8619	0.8651	**0.8960**	0.8780
	Time	0.0868	1.562	41.346	9.408	1.343
Haberman	Accuracy	70.46±2.11	61.59±3.12	71.46±3.62	72.47±1.72	**73.19±1.72**
	Precision	0.7046	0.8333	0.7246	0.7249	**0.8050**
	Recall	0.8913	0.7246	0.7246	**1**	0.8947
	Time	0.0026	0.6961	2.167	2.289	0.8330
WDBC	Accuracy	92.19±2.91	92.58±1.69	91.80±0.61	91.80±0.51	**94.14±1.65**
	Precision	0.9958	0.9595	0.9476	0.9476	**0.9640**
	Recall	0.9256	0.9634	0.9671	0.9674	**0.9757**
	Time	0.0164	1.861	77.81	17.431	2.85
Australian	Accuracy	83.18±4.48	83.58±4.18	84.74±2.85	87.74±3.22	**87.82±2.83**
	Precision	0.8944	0.8947	0.8977	0.8978	**0.9064**
	Recall	0.8742	0.9748	0.9749	0.9841	**0.9861**
	Time	0.0211	2.278	62.214	20.701	20.58
Pima Indians	Accuracy	72.54±3.64	73.99±4.25	73.57±4.16	**75.14±3.53**	74.57±2.83
	Precision	0.8961	0.8366	0.8721	0.8814	**0.9350**
	Recall	0.7771	0.8439	0.8113	0.8660	**0.8746**
	Time	0.0159	1.194	125.98	22.342	1.532
CMC	Accuracy	64.09±2.20	65.30±2.41	64.44±2.57	64.87±1.47	**65.37±2.19**
	Precision	0.8216	0.8102	**0.8399**	0.8361	0.8090
	Recall	0.6991	0.7710	0.7346	0.7462	**0.7791**
	Time	0.088	2.934	110.924	29.31	4.057
Spambase	Accuracy	83.48±0.98	88.26±1.16	62.38±1.45	83.14±0.91	**88.41±1.11**
	Pecision	0.9686	0.9447	0.8243	**0.9690**	0.9443
	Recall	0.8580	0.9304	0.7918	0.8514	**0.9327**
	Time	1.462	109.073	849.58	118.602	190.886

data were changed from -1 to $+1$ and vice-versa. For experimentation, $k = 0, 15$ and 30 were considered for real-world small and large data sets. The performance of all these models were compared based on accuracy, precision [150] and recall [150]. The computational time (in seconds) of all the methods was also computed.

It is noteworthy that these experiments on small data sets were performed on a Lenovo laptop with Windows 10 operating system having 4GB RAM and RADEON graphics.

First, the results over data sets with 0% noise are reported in Table 6.4. The boldfaced accuracies, precision and recall values represent the best values corresponding to the data sets. It is observed that the dual form of $\overline{\text{pin}}$-TSVM perform better than rest of the techniques on most data sets. Note that for SVM and TSVM, the dual forms of these techniques were implemented for small data sets only as the computational time depends on the number of examples [147], so it was not used for large-scale data sets. The primal form using SGD (Algorithm 3) was used for large real-world data sets.

Next, 15% label noise was added to the data sets and the results are reported in Table 6.5. The $\overline{\text{pin}}$-TSVM-dual outperforms the other techniques in most cases. It was also observed that the decrease in the accuracy after adding 15% noise in the data sets was also less for $\overline{\text{pin}}$-TSVM-dual than the other techniques.

Next, the training data sets' label noise was further increased to 30% and the results are reported in Table 6.6. Fom Table 6.6, it was observed that the $\overline{\text{pin}}$-TSVM-dual still outperforms the rest of the methods in terms of accuracy, precision and recall for majority of the data sets. The method was also in close comparison to the TSVM in terms of computational time.

In all the above tables, Table 6.4, 6.5 and 6.6, the computational time for SVM was significantly less since only the labeled training set was used to train the model while in the rest of the techniques, labeled as well as the unlabeled set were used for training.

Next, the experiments were performed on large scale real-world data sets to compare the results of the proposed technique with the existing models. The data sets that were used are listed in Table 6.7. The CCCP form of TSVM, Ramp-TSVM, and the proposed approach were used to perform these experiments. $\overline{\text{pin}}$-TSVM was implemented using Algorithm 4.

Please note that the models were implemented on the master node of the IIT (BHU), Varanasi server with 96GB RAM to perform these experiments. These results are reported in Table 6.8. For image data sets like CIFAR-10, MNIST and Cat vs Dog, similar steps were performed as in [63]. The feature set was obtained from these images. The number of instances and the number of attributes of this feature set are listed in Table 6.7.

Table 6.7: Used Large Data Sets for Experimentation Purposes

S. No.	Data Sets	Instances	Features	No. of Classes
1	Banana	5300	3	2
2	Page Blocks	5473	11	5
3	Musk(version 2)	6568	168	2
4	Cat vs Dog	25000	1001	2
5	CIFAR-10	60000	16384	10
6	MNIST	70000	1570	10
7	Cover Type	581012	54	7

- *Description of the Data Sets listed in Table 6.7*

- **Banana** [151]: This data set is based on the two types of banana classification based on its shape.

- **Page Blocks** [148]: The task is to classify those page blocks of a document that has been detected by a segmentation process.

- **Musk (version2)** [148]: To classify whether the new molecules are musk or not.

- **Cat vs Dog** [152]: To classify the new image as cat image or dog image.

- **CIFAR-10** [153]: CIFAR-10 data set has 60,000 color images of size 32×32 pixels. These images belong to ten classes. To use this data set, we extract features from the image data set.

- **MNIST** [148]: MNIST database comprise of images of handwritten digits. The task is to identify the new digit based on the image.

- **Cover Type** [148]: The task is to predict the forest cover type based on the cartographic variables [148]. It includes a total of seven classes marked as integer 1 to 7 in the data set.

Similar to the experiments performed on small data sets, different levels of noise were addedd in these experiments also. For multi-class classification, the one versus rest approach was followed. From Table 6.8, it is observed that the proposed approach is close to the rest of the approaches for the noise-free data set. However, when noise was added to the data, the proposed approach outperformed significantly. Please note that the methods with empty values in Table 6.8 indicate that these methods have not produced any results in one month. The above-discussed techniques were also compared with convolutional neural network (CNN) on image data sets.

To compare the above-discussed techniques over the computational time, a data set with the maximum number of instances, Forest cover type, was chosen. The computational time of SVM is 6.67×10^3 minutes, TSVM is 6.81×10^5, Ramp-TSVM is 2.82×10^4 and the proposed method is 4.51×10^3 minutes. In the training time of the proposed method, $\overline{\text{pin}}$-TSVM is less than the others.

Table 6.8: Comparison of Various Techniques over Large Real-World Data Sets using Linear Kernel

Data Sets	Noise	Results	SVM	TSVM	Ramp-TSVM	$\overline{\text{pin}}$-TSVM	CNN
Banana	0% Noise	Accuracy	54.84	50.40	58.70	58.91	–
		Precision	0.5484	0.6952	0.5980	0.6893	
		Recall	0.6630	0.6469	0.9701	0.9716	
	15% Noise	Accuracy	54.68	50.07	57.19	57.06	
		Precision	0.5448	0.7028	0.5723	0.6926	
		Recall	0.9890	0.6513	0.9981	0.9985	

	30% Noise	Accuracy	55.22	48.97	58.41	58.96	
		Precision	0.5522	0.6775	0.5860	0.7054	
		Recall	0.9770	0.6381	0.9943	0.9946	
Page Blocks	0% Noise	Accuracy	92.11	94.78	89.21	95.09	-
		Precision	0.9238	0.9697	0.8921	0.9788	
		Recall	0.9631	0.9767	0.9535	1	
	15% Noise	Accuracy	91.47	94.84	89.21	94.92	
		Precision	0.9153	0.9652	0.8921	0.9661	
		recall	0.9994	0.9820	0.9414	1	
	30% Noise	Accuracy	92.05	93.84	90.55	94.61	
		Precision	0.9211	0.9615	0.9055	0.9714	
		Recall	0.9994	0.9635	0.9433	0.9996	
Musk	0% Noise	Accuracy	94.71	94.27	84.99	94.90	-
		Precision	0.9914	0.9717	0.9966	0.9791	
		Recall	0.9549	0.9623	0.8523	0.9681	
	15% Noise	Accuracy	89.17	91.24	87.92	91.76	
		Precision	0.9812	0.9770	0.9845	0.9790	
		Recall	0.9445	0.9512	0.8512	0.9578	
	30% Noise	Accuracy	80.63	83.83	86.09	91.24	
		Precision	0.9891	0.9576	0.9783	0.9791	
		Recall	0.9499	0.9688	0.8465	0.9581	
Cat vs Dog	0% Noise	Accuracy	50.10	96.25	96.59	97.25	95.71
		Precision	1	0.9797	0.9869	0.9881	0.9771
		Recall	0.5010	0.9725	0.9784	0.9841	9810
	15% Noise	Accuracy	50.09	96.02	93.81	96.18	93.78
		Precision	1	0.9680	0.9548	0.9858	0.9615
		Recall	0.5009	0.9736	0.9427	0.9743	0.9558

	30% Noise	Accuracy	48.48	95.50	89.79	95.80	76.20
		Precision	1	0.9856	0.9844	0.9870	0.8712
		Recall	0.4841	0.9686	0.9020	0.9732	0.8711
CIFAR-10	0% Noise	Accuracy	52.17	53.44	53.11	54.25	0.6508
		Precision	0.5684	0.6687	0.5995	0.6993	0.6991
		Recall	0.6783	0.6523	0.9712	0.9778	0.9872
	15% Noise	Accuracy	48.67	49.65	51.41	52.78	0.4513
		Precision	0.5432	0.6792	0.5980	0.7123	0.5810
		Recall	0.9771	0.6498	0.9956	0.9956	0.9762
	30% Noise	Accuracy	44.39	47.55	46.54	50.21	0.3767
		Precision	0.5211	0.5348	0.5312	0.6235	0.5210
		Recall	0.8965	0.6894	0.9873	0.9881	0.9621
MNIST	0% Noise	Accuracy	91.92	91.03	94.26	97.25	97.25
		Precision	0.9634	0.9601	0.9869	0.9891	1
		Recall	0.9775	0.9725	0.9784	0.9851	0.9812
	15% Noise	Accuracy	87.87	90.01	92.26	95.01	95.05
		Precision	0.9032	0.9227	0.9453	0.9721	0.9812
		Recall	0.9071	0.9436	0.9631	0.9878	0.8005
	30% Noise	Accuracy	82.21	89.98	91.56	94.34	94.01
		Precision	0.7898	0.9166	0.9229	0.9721	0.9651
		Recall	0.8445	0.9175	0.9246	0.9278	0.7927
Forest	0% Noise	Accuracy	63.57	76.14	69.18	77.47	
		Precision	0.8226	0.8977	0.8451	0.9092	
		Recall	0.9776	0.9712	0.9613	0.9842	
	15% Noise	Accuracy	58.22	-	67.99	76.34	-
		Precision	0.8002	-	0.8271	0.8956	

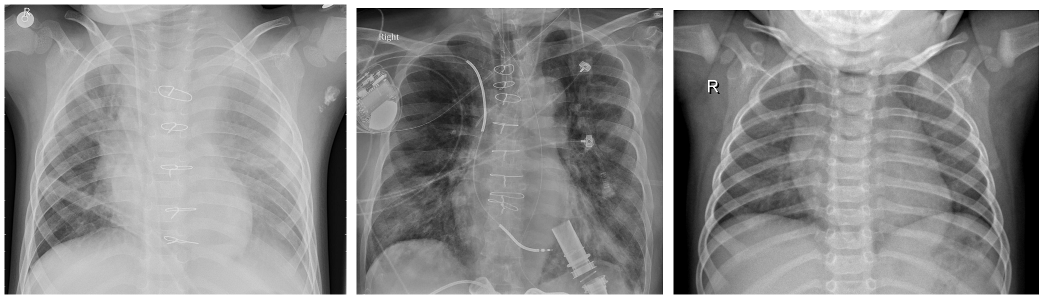
		Recall	0.9471	-	0.9606	0.9781
		Accuracy	51.21	-	65.46	73.21
	30% Noise	Precision	0.7911	-	0.8034	0.8676
		Recall	0.9251	-	0.9571	0.9661

As the proposed approach performed well on real-world data sets, this approach was also applied to the detection of disease due to the presence of novel coronavirus in the human body. Based on chest X-ray images, it was predicted that whether a person is infected or not. Since assigning these labels manually is time-consuming and difficult, especially during this pandemic, the labels using our proposed robust semi-supervised learning framework, $\overline{\text{pin}}$-TSVM.

6.4 Application to the Detection of Novel Coronavirus (COVID-19) Infected Patients using Chest X-ray Images

In this section, the use of the $\overline{\text{pin}}$-TSVM model to predict if a person is infected by COVID-19 is discussed. To do this, the model was trained using the chest X-ray images of humans.

It has been observed that the early detection of the disease with mild symptoms can help the patient in recovering from the disease. Therefore, it is required to detect the disease in its early stage. In this work, a semi-supervised machine learning model, TSVM, was used to detect the disease in humans using their chest X-ray images. Since labels come from human experts, and they do make mistakes, particularly in a pandemic like the situation where they are under considerable stress due to a large number of severe cases. The robust TSVM, $\overline{\text{pin}}$-TSVM, was used to detect the presence of COVID-19 in a human body. First, a data set was created using chest X-ray images of COVID patients, normal humans, and patients with bacterial infection. These images are shown in Figure 6.3. The pre-trained VGG19 model was used to extract features from the



(a) Patient having Bacterial Infection (b) Patient having Coronavirus (c) Normal X-ray

Figure 6.3: Chest X-ray Images of Humans with a) Bacterial Infection b) Coronavirus c) Normal X-ray

images [154]. In VGG19, the feature extraction part is from the first input layer to the max-pooling layer. The rest of the part of VGG19 is used for classification purposes. VGG19 uses multi-channel array signals to generate images and hence, it is superior than other machine learning models in terms of classification [155]. Therefore, VGG19 was used for feature extraction. To perform the experiments on the COVID-19 data set, the steps shown in Figure 6.4 were followed.

In these experiments, some of the labels of the training data (as described earlier in Section 6.3) were switched to test the robustness of $\overline{\text{pin}}$-TSVM on the COVID-19 data set. Therefore, when a few labels in the training set were wrong, the task was to formulate a model that is robust enough such that it maintains its accuracy to some extent, i.e., degrades gracefully rather than catastrophically. $\overline{\text{pin}}$-TSVM had proved its robustness through its performance on real-world data sets as discussed in the previous Section 6.3. This model was used on the COVID-19 data set (having chest X-ray images of humans). The results were computed in two ways: directly using the features obtained by applying the VGG19 model and extracting the essential features from this step using principal component analysis (PCA) [156]. The results are reported in Table 6.9. The accuracies and the computational time (in parenthesis) of the various techniques are also mentioned. Other performance metrics like sensitivity and specificity are also mentioned in Appendix E (Table 7.3). The last column of Table

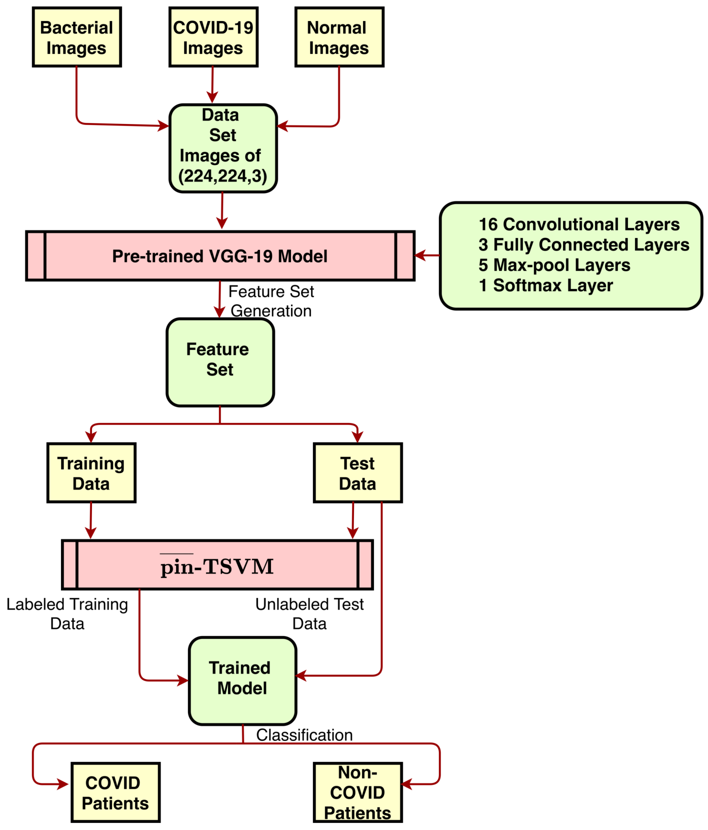


Figure 6.4: Steps to Extract Features From the COVID-19 Data Set and Training $\overline{\text{pin}}$-TSVM

6.9 represents the value of C used in these experiments. Note that the same value of C and C^* were used in these experiments.

From Table 6.9, it is observed that the proposed model has outperformed the existing techniques even after increasing noise in the data set. This method can also be used to assign labels to the unlabeled samples efficiently.

6.5 Summary

In this paper, an improved and robust TSVM was proposed towards label noise in the data set. The truncated pinball loss function was used instead of the conventional hinge loss function to introduce robustness in this framework (Section 6.2). In this work, both the primal form and the dual form of the proposed technique were implemented. The

Table 6.9: Comparison of Various Techniques over COVID-19 Data Set

Dats Sets	Results	SVM	TSVM	ramp-TSVM	$\overline{\text{pin}}$-TSVM	C
COVID-19 Data Set	0% Noise	93.63(0.2524)	94.62(5.70)	93.80(7.11)	**96.45(10.1)**	(1,1,2,1)
	10% Noise	92.15(0.2619)	93.10(7.10)	91.74(7.66)	**94.63(10.6)**	(1,1,2,3)
	15% Noise	89.67(0.3974)	92.17(14.73)	93.80(16.41)	**94.21(14.66)**	(2,2,1,1)
	30% Noise	68.77(0.3490)	91.74(18.18)	92.18(14.66)	**93.15(17.82)**	(2,2,2,2)
	40% Noise	68.18(0.3880)	90.32(13.97)	90.08(14.21)	**92.15(20.75)**	(2,2,2,3)
COVID-19 Data Set with PCA	0% Noise	92.89(0.0086)	92.15(0.9896)	91.74(1.6392)	**95.21(2.0552)**	(2,3,1,4)
	10% Noise	92.56(0.0079)	91.80(0.9192)	92.56(1.8664)	**93.89(1.8642)**	(2,3,1,1)
	15% Noise	91.95(0.0113)	92.98(1.7626)	92.28(1.6320)	**94.12(2.0590)**	(2,3,1,1)
	30% Noise	91.56(0.0124)	90.08(1.2117)	85.95(2.1602)	**92.15(2.6186)**	(2,2,1,3)
	40% Noise	70.25(0.0137)	86.78(1.7512)	86.36(2.4502)	**88.02(3.2693)**	(2,2,1,4)

CCCP was used on the primal form and implemented it using SGD (see Algorithm 4). The dual form was implemented using the *mlcv_quadprog*() function [63] in MATLAB (see Algorithms 5 and 6). In this work, algorithms for both linear and kernelized $\overline{\text{pin}}$-TSVM were provided. The proposed technique was also compared with the existing techniques on both the synthetic and real-world data sets. The proposed technique outperformed other techniques on the majority of the data sets.

The use of $\overline{\text{pin}}$-TSVM was also extended to the detection of coronavirus infected patients using their chest X-ray images. The proposed technique resulted in better accuracy, precision and recall even under the noisy environment. It is observed that the method can be efficiently used to detect the coronavirus infected patients using their chest X-ray images.