Chapter 3

RSVM-PDProx: Adding Robustness to SVM Using Rescaled α-hinge Loss Function

3.1 Introduction

SVM is a popular supervised ML algorithm. Besides its several advantages, there are some limitations of SVM too. Sensitivity to label noise is one of them. In Chapter 2, an extensive literature survey is presented on robust statistics based SVM. It is also described how different authors have tried to use robust loss functions in their formulation. From the survey in Chapter 2, it can be concluded that a rich literature is there to tackle the issue of sensitivity towards noise but the sparsity is sacrificed at the same time [37]. However, the proposal of truncated pinball loss function with SVM [57] tried to mitigate this problem, and demonstrated for the first time that both the robustness and the sparsity can be improved.

In this chapter, a robust SVM with enhanced sparseness is proposed. As SVM is used extensively in machine learning applications, it becomes essential to obtain a sparse model robust to noise in the data set. Although many researchers have presented

different approaches to get a robust SVM, the work on robust SVM based on rescaled hinge loss function (RSVM-RHHQ) [78] has attracted a great deal of attention. Using correntropy with hinge loss function has added a noticeable amount of robustness to the model. However, it was noticed that the sparsity of the model can be further increased with improved robustness towards label noise. In this chapter, correntropy [78] is applied to the α-hinge loss function, which results in a better loss function than the rescaled hinge loss function. Correntropy is the term which is mostly used in robust learning. It is a metric used to measure local similarity. Towards this direction, rescaled α-hinge loss has been proposed, which is bilinear in the dual variable α and the weight vector w. A non-smooth regularizer with a non-convex and non-smooth loss function is used in the proposed formulation. This non-smooth and non-convex problem is solved using the primal-dual proximal method [79]. It is observed that this combination not only adds sparsity to the model, but is also better than the existing robust SVM methods in terms of robustness towards label noise. Thus, this chapter discusses a proposed approach in which the SVM is made sparse and, at the same time, robust to label noise. The optimization problem which is used frequently in machine learning problems is

$$\min_{w\in\mathbb{R}^d} \; F(w) = \frac{1}{n}\sum_{i=1}^{n} l(w; x_i, y_i) + \lambda R(w). \tag{3.1}$$

Here, d denotes the number of features. $l(w; x_i, y_i)$ is the loss function with w as the weight vector, x_i and y_i are the i^{th} input and target values, respectively, of the input vector, X and the target vector, y. $R(w)$ is the regularization term used with λ as a regularization parameter. Equation (3.1) is converted into a non-smooth optimization problem of the form given by (see [79])

$$\min_{w\in Q_w}\Big[F(w) = \max_{\alpha\in Q_\alpha} L(w, \alpha; X, y) + \lambda R(w)\Big]. \tag{3.2}$$

In (3.2), the loss function of (3.1) changes after incorporating the dual variable into it,

and this changed loss function is denoted by $L(w, \alpha; X, y)$. In this equation, parameters w in domain Q_w and α in domain Q_α are referred to as primal and dual variables, respectively [79]. Now, the equivalent min-max formulation of (3.2) is (see [79])

$$\min_{w \in Q_w} \max_{\alpha \in Q_\alpha} F(w, \alpha) = L(w, \alpha) + \lambda R(w). \tag{3.3}$$

In the proposed method, the loss function is the rescaled α-hinge loss, and this function is obtained by applying correntropy to the α-hinge loss function [79]. In the proposed formulation, $\|\cdot\|_2$ norm of w is used as the regularizer. Both the loss function and the regularizer used here are non-smooth. Both the loss function and the regularizer used here are non-smooth and are discussed in detail in Section 3.3.

3.1.1 Motivation and Contribution

The bilinear nature of α-hinge loss is the primary motivation behind this work. Also, applying correntropy to this loss function makes the C-loss bounded, ultimately leading to a lower impact of outliers on the model training.It is observed that the use of $\|\cdot\|_2$ regularization makes the standard models label noise-robust [80], so $\|\cdot\|_2$ regularization is used in the problem formulation. It improved the robustness and the sparseness of the model, which is shown experimentally in Section 3.5. Next, the main contributions of this work are listed below:

(i) Correntropy is applied to α-hinge loss (named it as a rescaled α-hinge loss), which made the C-loss function bounded, monotonic, and non-convex. This rescaled α-hinge loss function is robust to outliers (shown experimentally in Section 3.5). In this chapter, the hinge loss function is not rescaled directly. The bilinear hinge loss function with dual variable α is rescaled, which made the proposed algorithm different from existing techniques.

(ii) A non-smooth regularizer, $\|\cdot\|_2$, is also added with rescaled α-hinge loss func-

tion, which made the overall function non-smooth. This non-smooth optimization problem is solved using the PDProx algorithm (Algorithm 1 [79]).

(iii) The experiments are performed on synthetic as well as real-world data sets. For experimentation purposes, twelve real-world data sets are used. Various levels of data noise are added to the training set to test the robustness of the model.

(iv) The proposed approach added more robustness and sparseness to SVM (shown experimentally in Section 3.5).

(v) The convergence of the PDProx method is theoretically proved for the optimization problem under consideration.

3.1.2 Outline

The rest of the chapter is organized as follows. The next section, Section 3.2 mentions the related concepts to this work. Section 3.3 discusses the rescaled form of α-hinge loss function. In this section, the detailed formulation of the proposed approach, RSVM-PDProx, is given. In the next section, Section 3.4, the detailed analysis of RSVM-PDProx is given. Section 3.4 analyzes the convergence rate and the time complexity of the PDProx dual algorithm. Section 3.5 presents the experimental results demonstrating the superiority of the proposed approach over the existing methods in terms of robustness and sparsity. In Section 3.6, the dual variable and the regularization parameter effect are discussed. The work is concluded in Section 3.7.

3.2 Related Concepts

For 'maximum-margin' classification, traditionally, the hinge loss function is used which is given by

$$l_{\text{hinge}}(w, b) = \max(0, 1 - y(w^T x + b)). \tag{3.4}$$

In the above expression, y is the target variable whose value corresponding to every instance (with input vector x) is either $+1$ or -1. Here $w^T x + b$ is the equation of the separating hyperplane, where w and b are the weight vector and bias term, respectively. This loss function is convex and is not differentiable, but its subgradient can be computed. Yang et al. in [79] proposed to change the non-smooth loss function of the form (3.1) into the form given by

$$l(w; x_i, y_i) = \max_{\alpha_i \in \Delta_\alpha} f(w, \alpha_i; x_i, y_i), \tag{3.5}$$

where Δ_α is the domain of the variable vector α. Also, $f(w, \alpha; , x, y)$ is a bilinear function in w and α. Using (3.5), problem (3.1) can be converted into (3.2) with $L(w, \alpha; X, y)$ as below (see [79]):

$$L(w, \alpha; X, y) = \frac{1}{n} \sum_{i=1}^{n} f(w, \alpha_i; x_i, y_i), \tag{3.6}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ being defined in the domain

$$Q_\alpha = \left\{ \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T \;\middle|\; \alpha_i \in \Delta_\alpha \right\}.$$

By (3.5), the α-hinge loss can be re-written as

$$l_{\alpha\text{-hinge}}(w, \alpha) = \max_{\alpha \in [0,1]} \alpha \left(1 - y(w^T x)\right). \tag{3.7}$$

α is the dual variable. Note that both (3.4) and (3.7) represent the classical hinge loss function, but in (3.7), only the weight vector, w, is considered, and the bias term, b, is neglected.

As discussed in [78], correntropy is a term used in robust learning. It was used for classification for the first time in [81], and the expression for the correntropy loss

function (C-loss) is

$$l_C(z) = \beta \left[1 - e^{\left(-\frac{(1-z)^2}{2\sigma^2} \right)} \right], \tag{3.8}$$

where σ is a window width [81] and $\beta = [1 - e^{(-1/(2\sigma^2))}]^{-1}$ is a normalizing constant; here, z is the margin variable, and this is equal to $y(w^T x)$. Interestingly, this function looks similar to the Welsch loss function [82]. The C-loss function is unbounded and non-monotonic [78]. Xu et al. [78] combined the C-loss function and the hinge loss function (3.4) to get the rescaled hinge loss function (see Figure 3.1) and this is given by

$$l_C(z) = \beta \left[1 - e^{\left(-\frac{l_{\text{hinge}}(z)}{2\sigma^2} \right)} \right]. \tag{3.9}$$

In this work, we considered $\eta = 1/2\sigma^2 \geq 0$. This formulation is used for the proposed approach as well. In Figure 3.1, z is plotted on the x-axis which is equal to $y(w^T x + b)$

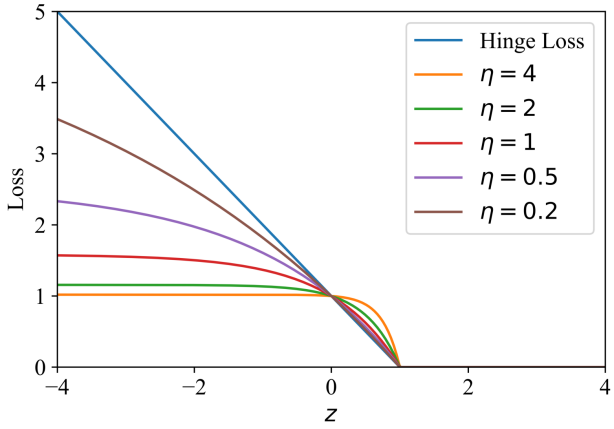


Figure 3.1: Rescaled Hinge Loss Function with Different η Values

and the 'Loss' corressponding to z values is plotted on the y-axis. From Figure 3.1, it can be observed that as η value increases, the value of 'Loss' (on y- axis) decreases.

3.3 RSVM-PDProx

In this section, the proposed formulation is described. This formulation consists of the non-smooth loss function and the non-smooth regularizer. The non-smooth loss function is rescaled α-hinge loss function.

3.3.1 Rescaled α-hinge Loss Function

Correntropy (3.8) is applied to α-hinge loss that is given in (3.7) to obtain the rescaled α-hinge loss function. After applying correntropy, the rescaled α-hinge loss is obtained as

$$l_{\alpha\text{-rhinge}}(z) = \beta\left[1 - e^{\left(-\eta l_{\alpha\text{-hinge}}(z)\right)}\right], \tag{3.10}$$

where $\eta = \frac{1}{2\sigma^2} \geq 0$ is a scaling constant and $\beta = \left(1 - e^{(-\eta)}\right)^{-1}$. After applying the C-loss function to α-hinge loss, it makes the overall function bounded, monotonic, and non-convex (see Figure 3.2). A separate plot of both the loss functions at $\eta = 4$ is

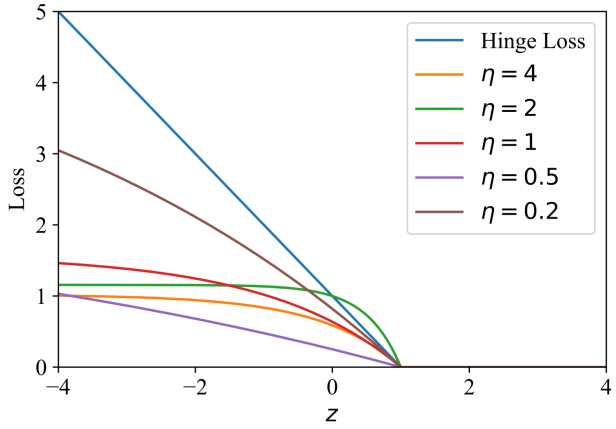


Figure 3.2: Rescaled α-hinge Loss Function with Different η Values

shown in Figure 3.3. Figure 3.3 shows that the proposed loss function is either equal or less than the rescaled hinge loss function. The red curve in the plot is lower than the blue curve for various marginal parameters z. Hence, corresponding to the different z, the value of 'Loss' is less for the proposed function. It improved the accuracy of the

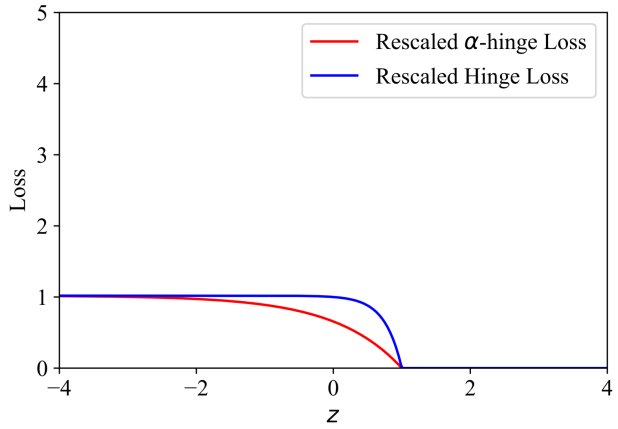


Figure 3.3: Comparsion of Rescaled α-hinge Loss Function and Rescaled Hinge Loss Function at η=4

proposed classifier. The analytical proof showing rescaled α-hinge loss is better than rescaled hinge loss is given in Appendix C.

3.3.2 Non-smooth Regularizer, $\|.\|_2$

The proposed loss function is combined with a non-smooth $\|\cdot\|_2$ regularizer and thereby obtained altogether a new optimization problem:

$$\min_{w\in\mathbb{R}^d} \mathcal{F}(w), \text{ where } \mathcal{F}(w) = \sum_{i=1}^{n} \beta \left(1 - e^{\left(-\eta\left(l_{\alpha\text{-hinge}}(w,x_i,y_i)\right)\right)}\right) + \lambda \|w\|_2 , \tag{3.11}$$

where λ is a regularization parameter. With the help of (3.7), the objective function in (3.11) can be expressed as follows:

$$\mathcal{F}(w) = \max_{\alpha\in[0,1]^n} L(w,\alpha;X,y) + \lambda R(w),$$

where

$$L(w,\alpha;X,y) = \frac{1}{n}\sum_{i=1}^{n} f(w,\alpha_i;x_i,y_i), \;\; R(w) = \|w\|_2,$$

and

$$f(w, \alpha_i; x_i, y_i) = \frac{1}{1 - e^{-\eta}} \left[1 - e^{-\eta\alpha_i(1 - y_i w^T x_i)}\right], \ i = 1, 2, \ldots, n.$$

Note that

$$L(w, \alpha; X, y) = \beta \left[n - \sum_{i=1}^{n} e^{-\eta\alpha_i(1 - y_i w^T x_i)}\right],$$

where

$$\beta = \frac{1}{1 - e^{-\eta}} \geq 0.$$

Thus, the optimization problem (3.11) becomes

$$\begin{aligned} \min_{w \in \mathbb{R}^d} \mathcal{F}(w) = \ & \beta \max_{\alpha \in [0,1]^n} \left[1 - \frac{1}{n} \sum_{i=1}^{n} e^{-\eta\alpha_i(1 - y_i w^T x_i)}\right] + \lambda \left\|w\right\|_2 \\ = \ & \max_{\alpha \in [0,1]^n} l(w, \alpha; X, y) + \lambda \left\|w\right\|_2, \end{aligned} \tag{3.12}$$

where

$$l(w, \alpha; X, y) = \beta \left[1 - \frac{1}{n} \sum_{i=1}^{n} e^{-\eta\alpha_i(1 - y_i w^T x_i)}\right].$$

Therefore, the problem (3.11) is equivalent to the following min-max problem:

$$\min_{w \in \mathbb{R}^d} \max_{\alpha \in [0,1]^n} F(w, \alpha), \text{ where } F(w, \alpha) = l(w, \alpha; X, y) + \lambda \left\|w\right\|_2. \tag{3.13}$$

This is the final optimization problem of the proposed approach. In (3.13), $l(w, \alpha; X, y)$ represents the proposed loss function, the rescaled α-hinge loss function. Note that the loss function and the regularizer in (3.13) are non-smooth functions. Due to this, a non-smooth optimization technique, PDProx dual [79], is used to solve this problem. The steps that are followed to solve (3.13) are mentioned in Algorithm 1. The approach

is named as RSVM-PDProx.

Algorithm 1 PDProx-Dual algorithm for (3.13) to get $\hat{w}$ and $\hat{\alpha}$

Input: $\{x_i, y_i\}_{i=1}^n$;

λ is the regularization parameter;

η is the rescaling parameter;

α is the dual variable;

T is the total number of iterations

Output: $\hat{w}_T$ is the final weight vector and $\hat{\alpha}_T$ is the final α vector.

1: $\gamma \leftarrow \sqrt{\frac{1}{2c}}$ where c can be computed from (3.14)

2: $w_0 \leftarrow 0$, $\beta_0 \leftarrow 0$, $t \leftarrow 0$ ▷ Initializations required before entering the loop

3: **for** $t = 1$ to T **do**

4: $\alpha_t = \prod_{Q_\alpha = \Delta_\alpha} [\, \beta_{t-1} + \gamma\, G_\alpha(w_{t-1},\, \beta_{t-1})]$

5: $w_t = \underset{w \in \{Q_w = \mathbb{R}^d\}}{\operatorname{argmin}} \frac{1}{2} \|w - (w_{t-1} - \gamma G_w(w_{t-1}, \alpha_t))\|_2^2 + \gamma\lambda \|w\|_2$ ▷ FISTA (backtracking) [83].

6: $\beta_t = \prod_{Q_\alpha} [\beta_{t-1} + \gamma G_\alpha(w_t,\, \alpha_t)]$

7: **end for**

8: **return** $\hat{w}_T = \sum_{t=1}^T \frac{w_t}{T}$ and $\hat{\alpha}_T = \sum_{t=1}^T \frac{\alpha_t}{T}$ ▷ $\hat{w}$ is used in the equation of separating hyperplane and also in the convergence proof along with $\hat{\alpha}$

In Step 1 of Algorithm 1, to estimate a value of c, the following inequality is used (see [79]):

$$\|G_\alpha(w_1, \alpha_1) - G_\alpha(w_2, \alpha_2)\|^2 \leq c \|w_1 - w_2\|^2 \tag{3.14}$$

where G_α and G_w represents the gradients of the objective function F w.r.t. α and w, respectively.

After implementing this algorithm, new data points, x are categorized into '+1' and '-1' classes based on the sign of $\hat{w}^T x$ ($w^T x + b$ can also be considered. This would amplify the convergence rate [79] by a constant of $\sqrt{2}$).

Since the proposed approach is the improved version of RSVM-RHHQ [78], the difference between the proposed method, RSVM-PDProx, and RSVM-RHHQ, is presented before proceeding to the next section. The mathematical differences in the proposed approach and RSVM-RHHQ [78] are:

(i) (*Difference in the loss functions*). The loss function used in RSVM-RHHQ [78] is (refer to Figure 3.1)

$$\beta\left[1-e^{\left(-\eta l_{\text{hinge}}(z)\right)}\right],$$

while the loss function in the proposed approach is (refer to Figure 3.2)

$$\beta\left[1-e^{\left(-\eta l_{\alpha\text{-hinge}}(z)\right)}\right].$$

The incorporation of dual variable α limits the loss that is less than the rescaled hinge loss. This is clearly depicted in Figure 3.3.

(ii) (*Difference in the regularizers*). RSVM-RHHQ [78] used a smooth regularizer, $\|w\|_2^2$. In contrast, the proposed approach uses a non-smooth regularizer, $\|w\|_2$. Therefore, the number of support vectors are amplified in RSVM-RHHQ than the proposed approach.

(iii) (*Difference in the used optimization techniques*). The optimization technique used in RSVM-RHHQ is the half-quadratic optimization technique [78]. In the proposed approach, since both the loss function and regularizer are non-smooth, the proximal dual optimization technique, PDProx dual [79] is used, given in Algorithm 1.

3.4 Analysis of RSVM-PDProx

In this section, the rate of convergence and the time complexity analysis of RSVM-PDProx are discussed.

3.4.1 Rate of Convergence of RSVM-PDProx

Before the convergence analysis of PDProx method for the optimization problem (3.11), the following two lemmas are presented that are useful to arrive at the convergence proof.

Lemma 3.1 (See [79]). *The updates of α_t, w_t and β_t in Algorithm 1 are equivalent to the below mentioned gradient mappings:*

$$\begin{pmatrix} \alpha_t \\ w_t \end{pmatrix} = \prod_{[0,1]^n,\ \mathbb{R}^d} \begin{pmatrix} \beta_{t-1} + \gamma G_\alpha(u_{t-1}, \beta_{t-1}) \\ u_{t-1} - \gamma(G_w(u_{t-1}, \alpha_t) + \lambda v_t) \end{pmatrix}$$

and

$$\begin{pmatrix} \beta_t \\ u_t \end{pmatrix} = \prod_{[0,1]^n,\ \mathbb{R}^d} \begin{pmatrix} \beta_{t-1} + \gamma G_\alpha(w_t, \alpha_t) \\ u_{t-1} - \gamma(G_w(w_t, \alpha_t) + \lambda v_t) \end{pmatrix},$$

with the initialization $u_0 = w_0$, where $v_t \in \partial R(w_t)$, the subgradient of $R(w_t)$.

Lemma 3.2 (See [79]).

$$\gamma \begin{pmatrix} G_w(w_t, \alpha_t) + \lambda v_t \\ -G_\alpha(w_t, \alpha_t) \end{pmatrix}^T \begin{pmatrix} w_t - w \\ \alpha_t - \alpha \end{pmatrix} \leq \frac{1}{2} \left\| \begin{pmatrix} w - u_{t-1} \\ \alpha - \beta_{t-1} \end{pmatrix} \right\|_2^2 - \frac{1}{2} \left\| \begin{pmatrix} w - u_t \\ \alpha - \beta_t \end{pmatrix} \right\|_2^2 + \gamma^2 \left\| G_\alpha(w_t, \alpha_t) - G_\alpha(u_{t-1}, \beta_{t-1}) \right\|_2^2 - \frac{1}{2} \left\| w_t - u_{t-1} \right\|_2^2.$$

Theorem 3.1 *The output $\hat{\alpha}_T$ and $\hat{w}_T$ of Algorithm 1 satisfies*

$$F(\hat{w}_T, \alpha) - F(w, \hat{\alpha}_T) \leq \frac{\eta e^\eta (1 + \sqrt{\eta}) \sqrt{2c}}{T}$$

for T number of iterations.

Proof: Let $F_1(w, \alpha_i) = l_1(w, \alpha_i; X, y) + \lambda \left\| w \right\|_2$, where $l_1(w, \alpha_i; X, y) = \frac{1}{n} \sum_{i=1}^n \alpha_i (1 - y_i w^T x_i)$. Then,

$$\frac{F(w_t, \alpha) - F(w, \alpha)}{F_1(w_t, \alpha) - F_1(w, \alpha)} = \frac{l(w_t, \alpha; X, y) - l(w, \alpha; X, y)}{l_1(w_t, \alpha; X, y) - l_1(w, \alpha; X, y)}$$

$$
\begin{aligned}
&= \frac{\beta\Big[1-\frac{1}{n}\sum_{i=1}^{n} e^{-\eta\alpha_i(1-y_i w_t^T x_i)}\Big] - \beta\Big[1-\frac{1}{n}\sum_{i=1}^{n} e^{-\eta\alpha_i(1-y_i w^T x_i)}\Big]}{\frac{1}{n}\sum_{i=1}^{n}\alpha_i(1-y_i w_t^T x_i) - \frac{1}{n}\sum_{i=1}^{n}\alpha_i(1-y_i w^T x_i)} \\
&= \frac{\sum_{i=1}^{n}\Big[e^{-\eta\alpha_i(1-y_i w^T x_i)} - e^{-\eta\alpha_i(1-y_i w_t^T x_i)}\Big]}{\sum_{i=1}^{n}\alpha_i(y_i w^T x_i - y_i w_t^T x_i)} \\
&= \frac{\sum_{i=1}^{n}\Big(-\eta\alpha_i(y_i w^T x_i - y_i w_t^T x_i) + \frac{\eta^2\alpha_i^2}{2}\Big((y_i w^T x_i)^2 - (y_i w_t^T x_i)^2\Big) - \dots\Big)}{\sum_{i=1}^{n}\alpha_i(y_i w^T x_i - y_i w_t^T x_i)} \\
&= \frac{\sum_{i=1}^{n}\Big\{-\eta\alpha_i(y_i w^T x_i - y_i w_t^T x_i)\Big\}\Big[1-\frac{\eta\alpha_i}{2}(y_i w^T x_i + y_i w_t^T x_i) + \dots\Big]}{\sum_{i=1}^{n}\alpha_i(y_i w^T x_i - y_i w_t^T x_i)} \\
&\le \frac{\Big\{-\eta\sum_{i=1}^{n}\alpha_i(y_i w^T x_i - y_i w_t^T x_i)\Big\}\Big\{1+\frac{n}{2}(2R)\,\|w\| + \frac{n^2}{3!}(3R^2)\,\|w\| + \dots\Big\}}{\sum_{i=1}^{n}\alpha_i(y_i w^T x_i - y_i w_t^T x_i)}.
\end{aligned}
$$

As for all $i \in \{1,\dots,n\}$, we have $\|x_i\| \le R$, thus

$$
\frac{F(w_t,\alpha) - F(w,\alpha)}{F_1(w_t,\alpha) - F_1(w,\alpha)} = -\eta\Big\{1 + \eta R\,\|w\| + \frac{\eta^2 R^2}{2!}\,\|w\| + \dots\Big\} = -\eta e^{\eta R\|w\|}.
$$

Therefore,

$$
|F(w_t,\alpha) - F(w,\alpha)| \le \eta |F_1(w_t,\alpha) - F_1(w,\alpha)|
$$

Next,

$$
\begin{aligned}
\frac{F(w,\alpha_t) - F(w,\alpha)}{F_1(w,\alpha_t) - F_1(w,\alpha)} &= \frac{l(w,\alpha_t;X,y) - l(w,\alpha;X,y)}{l_1(w,\alpha_t;X,y) - l_1(w,\alpha;X,y)} \\
&= \frac{\beta\Big[1-\frac{1}{n}\sum_{i=1}^{n} e^{-\eta\alpha_{it}(1-y_i w^T x_i)}\Big] - \beta\Big[1-\frac{1}{n}\sum_{i=1}^{n} e^{-\eta\alpha_i(1-y_i w^T x_i)}\Big]}{\frac{1}{n}\sum_{i=1}^{n}\alpha_i t(1-y_i w^T x_i) - \frac{1}{n}\sum_{i=1}^{n}\alpha_i(1-y_i w^T x_i)} \\
&= \frac{\sum_{i=1}^{n}\Big[e^{-\eta\alpha_{it}(1-y_i w^T x_i)} - e^{-\eta\alpha_i(1-y_i w^T x_i)}\Big]}{\sum_{i=1}^{n}(\alpha_{it}-\alpha_i)(1-y_i w^T x_i)} \\
&\le \frac{-\eta\sum_{i=1}^{n}\Big[-\eta(\alpha_{it}-\alpha_i)(1-y_i w^T x_i)\Big]\Big[1+\frac{n}{2}2(1+R)\,\|w\| + \frac{n^2}{3!}3(1+R)^2\,\|w\| + \dots\Big]}{\sum_{i=1}^{n}(\alpha_{it}-\alpha_i)(1-y_i w^T x_i)}
\end{aligned}
$$

$$= \eta\Big\{1 + \eta(1+R)\,\|w\| + \frac{\eta^2}{2!}(1+R^2)\,\|w\| + \dots\Big\}$$
$$= -\eta e^{\eta(1+R\|w\|)} \leq -\eta e^{\eta}.$$

Therefore,

$$|F(w,\alpha_t) - F(w,\alpha)| \leq \eta e^{\eta} |F_1(w,\alpha_t) - F_1(w,\alpha)|.$$

Hence,

$$\begin{aligned}
|F(\hat{w}_T,\alpha) - F(w,\hat{\alpha}_T)| &\leq |F(\hat{w}_T,\alpha) - F(\hat{w}_T,\hat{\alpha}_T)| + F(\hat{w}_T,\hat{\alpha}_T) - F(w,\hat{\alpha}_T)|, \text{ where } \hat{w}_T = \frac{1}{T}\sum_{t=1}^{T} w_t \text{ and } \hat{\alpha}_t = \frac{1}{T}\sum_{t=1}^{T}\alpha_t \\
&\leq \eta e^{\eta} |F_1(\hat{w}_T,\hat{\alpha}_T) - F_1(\hat{w}_T,\alpha)| + \eta |F_1(\hat{w}_T,\hat{\alpha}_T) - F_1(w,\hat{\alpha}_T)| \\
&\leq \eta e^{\eta}\Big\{|F_1(\hat{w}_T,\hat{\alpha}_T) - F_1(\hat{w}_T,\alpha)| + |F_1(\hat{w}_T,\hat{\alpha}_T) - F_1(w,\hat{\alpha}_T)|\Big\} \\
&\leq \eta e^{\eta} \frac{\|\hat{w}_T\| + \|\alpha\| + \|\hat{\alpha}_T\| + \|w\|}{\sqrt{2/c}\,T}.
\end{aligned}$$

As $\|w\|_2^2$ and $\|\hat{w_T}\|_2^2$ are in $O(1/\lambda)$, $\|\alpha\|_2^2 \leq n$, and $\|\hat{\alpha}_T\|_2^2 \leq n$ (see [79]), we get

$$\begin{aligned}
|F(\hat{w}_T,\alpha) - F(w,\hat{\alpha}_T)| &\leq \eta e^{\eta} \frac{1 + \sqrt{n} + \sqrt{n} + 1}{\sqrt{2/c}\,T} \\
&= \frac{\eta e^{\eta}(1+\sqrt{n})\sqrt{2c}}{T}.
\end{aligned}$$

Therefore,

$$F(\hat{w}_T,\alpha) - F(w,\hat{\alpha}_T) \leq \frac{\eta e^{\eta}(1+\sqrt{n})\sqrt{2c}}{T}.$$

□

Note 1 By Theorem 3.1, it can be observed that the convergence rate of RSVM-PDProx is $O(1/T)$.

Note 2 Theorem 3.1 shows that as the number of iterations $T \to \infty$, $|F(\hat{w}_T,\alpha) - F(w,\hat{\alpha}_T)| \to 0$; and hence, the strong duality holds. Therefore, $(\hat{\alpha}_T, \hat{w}_T)$ is an optimum solution to the problem (3.11) with $O(1/T)$ duality gap; the larger the number of itera-

tions, the smaller the duality gap. Importantly, Theorem 3.1 reports that to reach at a solution of (3.11) with ϵ-duality-gap, Algorithm 1 requires at least $\frac{1}{\epsilon}\left(\eta e^{\eta}(1+\sqrt{n})\sqrt{2c}\right)$ number of iterations. This estimate explicitly reveals the robustness of the algorithm with respect to the choice of T. If fewer iterations are chosen, the duality gap will be higher, and hence the solution will show a low performance.

3.4.2 Time Complexity of RSVM-PDProx

To find the time complexity of Algorithm 1, the time complexity of each step of Algorithm 1 is executed for each iteration. In Algorithm 1, Steps 4 to 6 significantly contribute to the time complexity.

In Step 1, $\gamma = \sqrt{1/2c}$ is to be calculated with the help of an estimated value of c. To estimate a value of c, one can use the inequality (3.14). Here, $c = R^2/n$ for the given loss function (see appendix of [79]), where $\|x\|_2 \leq R$. Therefore, there are n operations to square all the elements, $n-1$ sums, and 2 square roots are required. Therefore, the overall complexity corresponding to Step 1 is $O(n)$ where n denotes the number of instances in a data set.

Step 2 in Algorithm 1 is the initialization step with two assignments, which lead to constant asymptotic time complexity [84].

In the next step, the loop begins. In Step 4, the projection operation is performed. Towards this direction, the gradient $G_\alpha(w_{t-1}, \beta_{t-1})$, is computed first. After that, the gradient is multiplied with γ, computed in Step 1. Now, $\gamma\ G_\alpha(w_{t-1}, \beta_{t-1})$ is added with β_{t-1}. Now, the projection operationis performed on the obtained result. Therefore, the time complexity is $(O(1) + O(n^2 d)) \times O(n)$.

The next step, Step 5, is the calculation of w_t for which the Fast Iterative Shrinkage Thresholding Algorithm (FISTA) [83] is used. The time complexity of Step 5 is $O(n^3)$ (see appendix/supplementary material of [85]).

In Step 6, the time complexity is similar to Step 4, $(O(1) + O(n^2 d)) \times O(n)$. In the

next step of Algorithm 1, the loop ends. Finally, the optimal weight and α vectors, $\hat{w}_T$ and $\hat{\alpha}_T$ respectively, are calculated with constant time complexity in Step 8.

After considering the time complexity of each step of Algorithm 1, it is observed that the final time complexity of the algorithm is $O(n^3)$.

3.5 Numerical Experiments and Results

In this section, the superiority of our method over existing techniques is proved experimentally. The experiments are performed over synthetic and real-world data sets. To perform the experiments, the data set is divided into the ratio of 70:30, where the significant part is for training the model (Training set), and the rest is for testing the model (Test set). All the experiments are performed on Spyder UI under Anaconda3-2019.07. These experiments are conducted on a Windows operating system with 16 GB RAM.

In a training set, some of the labels are randomly switched to test the robustness of the proposed approach. For example, for experiments with 10% noise in the data set, 10% labels of the training set [78] are switched (from positive class to negative class and vice-versa). Please note that a linear kernel is used to perform these experiments, but the technique can be easily used with the non-linear kernels.

To test the sparseness of all the methods, the ratio of a number of support vectors to the total number of instances (defined as support vector ratio) is computed. The model with a smaller support vector ratio is more sparse than the others.

3.5.1 Experiments on Synthetic Data Sets

For experiments, 100 instances were generated in the synthetic data set (using *make_blobs*() function in python) with two features (x_1, x_2) and this data set is divided into the ratio of 70:30. Figures 3.4, 3.5 and 3.6 shows this comparison. Experiments with increased number of instances were also performed. Results are also reported in tabular form

on page 167, Appendix B. For generating accuracies and the corresponding standard deviations, all the experiments were performed ten times.

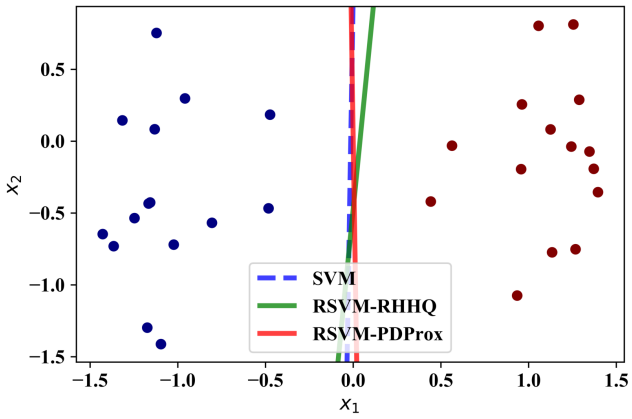


Figure 3.4: Comparison of SVM, RSVM-RHHQ and RSVM-PDProx over Synthetic Data Set with 0% Noise

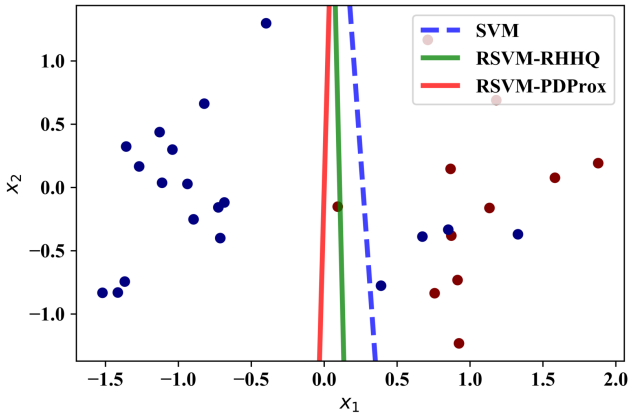


Figure 3.5: Comparison of SVM, RSVM-RHHQ and RSVM-PDProx over Synthetic Data Set with 15% Noise

In Figures 3.4, 3.5 and 3.6 experiments were performed on data with no noise, data with 15% noise, and data with 30% noise respectively. When the data is noise-free, all the methods performed almost similar to each other and hence classifying the blue class instances (left side) and the red class instances (right side) correctly.

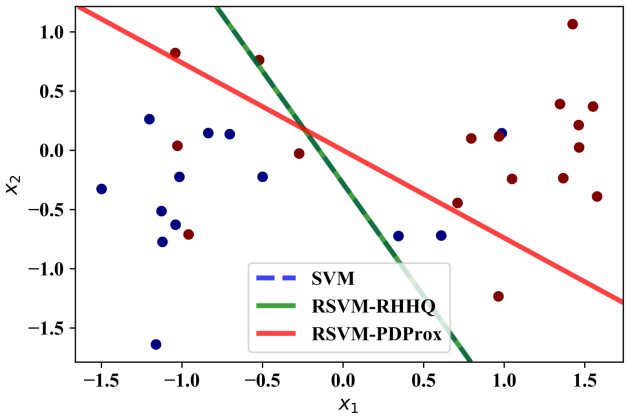


Figure 3.6: Comparison of SVM, RSVM-RHHQ and RSVM-PDProx over Synthetic Data Set with 30% Noise

When 15% label noise is added to the training set, the proposed method classifies both the classes more accurately, while this is not so with the conventional SVM and RSVM-RHHQ (see Figure 3.5).

When 30% label noise is added to the training set, the proposed approach again classifies better than the rest of the methods. It is evident from Figure 3.6 where the existing techniques are classifying the red instances wrong while RSVM-PDProx correctly classifies these instances.

3.5.2 Experiments on Real-world Data Sets

In this section, the proposed method is compared with the conventional SVM [9], SVM using PDProx optimization technique [79], SVM with the truncated pinball loss function [57] and the SVM with a rescaled hinge loss function [78]. The method is compared with these techniques based on both robustness towards label noise and sparsity. To perform the experiments, twelve real-world data sets were used. These data sets were obtained from diverse sources to test the method over a wide variety of applications. Also, data sets with a small number of instances, a large number of instances, and data sets with a large number of features were used for experimentation purposes. Table 3.1

enlists these data sets. The number of instances, features, and the number of classes corresponding to each data set is mentioned in this table. There is one more column in this table, 'Class Ratio.' Since the proposed approach is for two-class classification problems, this column specifies the positive class ratio to the negative class in the data set. Please note that the entries of Table 3.1 are arranged according to the increasing number of instances. Similarly, all the results reported following the same order of the data sets. For

Table 3.1: Data Sets Used for Experimentation Purposes

S. No.	Data Sets	Instances	Features	No. of Classes	Class Ratio
1	SCADI	70	206	7	3.37:1
2	Sonar	208	61	2	3.00:1
3	Cleveland Heart	303	14	2	0.83:1
4	Haberman	306	4	2	0.36:1
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
9	Banana	5300	3	2	1.23:1
10	Page Blocks	5473	11	5	8.77:1
11	Musk(version 2)	6568	168	2	0.18:1
12	Mushrooms	8124	22	2	0.93:1

3.5.2.1 Robustness

The term robustness is used as the property by which the performance of the classifier is least affected even in the presence of label noise. Therefore, if a model is robust than the others, it shows better accuracy than the others. Hence, accuracy is used as a parameter to check the robustness of different methods.

First, the experiments were performed on the data sets with no noise. These results are reported in Table 3.2. For RSVM-RHHQ and RSVM-PDProx, the results were computed for different values of η (0.2, 0.5, 1, 2 and 3). Please note that in Tables

Table 3.2: Results of RSVM-PDProx and the Existing Methods with 0% Noise in the Real-World Data Sets

Data Sets	Results	SVM	SVM-PDProx	$\overline{\text{pin}}$-SVM	RSVM-RHHQ (η-val)	RSVM-PDProx					
						η=0.2	η=0.5	η=1	η=2	η=3	Parameter
SCADI	Accuracy	90.00±3.95	90.00±6.54	90.47±4.34	89.04±6.75 (0.5)	89.04±5.23	90.47±5.63	88.57±7.12	89.04±5.65	**93.33±4.85**	
	SVs	35.71	31.63	98.57	40.81	12.65	6.93	2.85	3.26	13.06	10^{-2}
	Time (in sec)	0.0007	0.1815	0.97s	0.0049	0.88	0.90	0.90	0.90	0.98	
Sonar	Accuracy	75.55±4.01	75.07±4.43	76.41±4.82	74.60±3.75 (2)	**76.82±3.19**	76.66± 4.26	74.92±5.21	75.87±4.24	75.07± 5.26	
	SVs	54.41	44.82	63.88	62.75	44.34	47.03	49.10	46.27	44.34	10^{-6}
	Time (in sec)	2.380	0.1815	0.007s	0.011	2.86	2.76	2.73	2.72	2.72	
Cleveland	Accuracy	81.64±2.55	83.62±1.56	47.77±3.21	82.96±2.36 (2)	83.29± 2.13	84.28±3.77	84.28±4.50	83.40±2.66	**86.37±4.99**	
	SVs	38.01	53.01	99.52	36.32	18.16	22.45	15.47	11.79	9.85	10^{-3}
	Time (in sec)	0.5962	0.32	0.0036	0.191	0.7	0.84	0.90	0.92	0.90	
Haberman	Accuracy	71.30±3.77	73.34±1.76	68.83±3.46	72.06±2.66 (1)	73.04±2.94	72.60±5.60	73.05±3.71	**73.47±2.39**	72.93±2.85	
	SVs	56.07	84.25	83.09	63.08	61.91	41.86	23.08	15.70	15.74	10^{-2}
	Time (in sec)	0.02	1.49	0.003s	0.089	1.5	1.62	1.81	1.87	1.88	
Australian	Accuracy	85.14±1.94	86.18±2.08	81.55±3.24	85.84±1.55 (1)	**86.57±1.85**	86.13±1.60	86.03±1.96	85.89±1.36	85.16±2.10	
	SVs	29.35	43.12	22.19	31.46	16.16	12.13	7.701	6.79	7.61	10^{-2}
	Time (in sec)	141.16	5.007	365.36	102.15	15.45	15.84	16.35	16.57	16.14	
Pima Indians	Accuracy	75.93±1.38	74.02±2.14	**77.13±6.87**	76.53±2.33 (3)	73.82±1.57	74.32±2.52	73.76±2.13	75.15±2.41	73.59±2.24	
	SVs	51.85	58.10	61.00	51.02	25.17	17.15	15.81	13.01	11.43	10^{-3}
	Time (in sec)	4.25	13.05	2.71	2.10	10.48	10.72	11.17	11.21	11.51	
CMC	Accuracy	67.01±2.42	64.34±1.60	66.67±1.45	67.17±0.98 (2)	66.85±1.87	65.27±2.58	66.93±1.69	**67.64±1.24**	65.99±2.02	
	SVs	71.21	85.77	67.66	71.17	48.81	54.09	44.09	33.97	25.78	10^{-2}
	Time (in sec)	0.18	2.40	0.078	0.07	1.28	2.99	3.11	3.23	3.24	
Spambase	Accuracy	92.70±1.73	91.006±0.92	79.49±0.11	92.17±0.41 (3)	92.17±0.71	92.33±0.65	92.47±0.65	**92.97±0.36**	92.07±0.51	
	SVs	17.98	63.94	60.08	21.42	51.97	37.94	20.57	12.32	9.009	10^{-3}
	Time (in sec)	898.65	28.46	1.17	182.79	73.06	123.67	126.12	126.75	126.51	
Banana	Accuracy	54.52±2.98	56.99±1.31	**80.48±0.98**	55.27±1.17 (0.2)	56.704±2.37	56.33±1.69	55.23±2.44	55.78±2.88	58.01±3.55	
	SVs	89.75	100	24.50	88.84	88.84	50.93	42.15	33.34	15.05	10^{-2}
	Time (in sec)	0.18	11.30	18.37	0.11	9.90	10.48	10.79	10.50	10.68	
Page Blocks	Accuracy	94.09±0.89	93.52±0.78	92.44±0.64	94.52±0.67 (0.2)	94.57±0.72	**94.67±0.61**	94.38±0.77	93.80±0.44	93.19±0.34	
	SVs	7.82	88.35	98.17	9.47	2.00	2.91	3.77	2.76	2.00	10^{-3}
	Time (in sec)	188.02	89.36	0.56	206.39	176.41	177.31	184.12	209.65	187.03	
Musk (version 2)	Accuracy	95.10±0.68	84.94±0.34	91.15±0.28	99.56±0.16 (2)	99.58±0.10	99.49±0.06	99.57±0.09	99.51±0.09	**99.61±0.09**	
	SVs	12.38	62.98	98.54	0.71	0	0.11	0	0.01	0.01	10^{-5}
	Time (in sec)	2.055	174.13	4.67	0.664	31.17	20.94	20.96	21.22	22.04	
Mushrooms	Accuracy	96.69±0.41	96.86±0.87	99.98±0.20	96.59±0.96 (1)	100±0.00	99.99±0.01	99.95±0.06	100±0.00	**100±0.00**	
	SVs	9.69	15.96	23.07	14.66	2.41	2.52	2.49	2.54	2.55	10^{-8}
	Time (in sec)	0.81	154.12	11.51	1.31	15.96	15.87	15.87	16.15	16.87	

3.2, 3.3 and 3.4, under the column named 'RSVM-RHHQ (η-val)' (η-val in parenthesis indicates the η value with the best accuracy), the best accuracy is mentioned amongst different η values. In all the results, the best accuracy in a row is marked bold. The support vector ratio and the computational time are also reported according to the same η value. In the case of RSVM-PDProx, all the results corresponding to the different η values are provided. Please note that all the experiments are performed ten times, so the mean and standard deviation (std) of accuracies on those ten runs can computed.

In Table 3.2, the accuracy, the support vector ratio (SVs), and the computational time (in sec) are mentioned. The table also comprises the value of the regularization

parameter used for RSVM-PDProx in the last column of Table 3.2. The optimal regularization parameter is selected using 10-fold cross-validation on 10% of the data set. The set from which the regularization parameter selected is

$$\lambda \in \left\{10^{-a} : a \in \{-10, -9, -8, \ldots, -1\}\right\}.$$

The same set of λ is considered for all the experiments.

Similarly, the experiments were conducted over the data sets with different label noise. First, the training set was corrupted, with a 15% label noise. Table 3.3 reports the results with a 15% label noise. In all the results, the best accuracy in a row is marked bold. Please note that all the experiments were performed ten times, so the mean and standard deviation (std) of accuracies on those ten runs are computed. These are reported in the format 'mean±std.' For SVs and the computational time, the mean of all the ten runs is reported.

Next, the experiments were performed after switching 30% of labels in the training set. The results of this experiment are reported in Table 3.4. Please note that for the numerical computation of $\overline{\text{pin}}$-SVM in all the experiments, the legacy of [57] is followed in which Shen et al. get the best accuracy at τ=0.5; therefore, in all the experiments, the same value of τ is used.

Table 3.2 shows that the proposed approach RSVM-PDProx shows better accuracy on ten data sets out of twelve data sets. On rest two data sets, $\overline{\text{pin}}$-SVM shows better accuracy than all the methods. $\overline{\text{pin}}$-SVM performs exceptionally well on the Banana data set but performs the worst on the Cleveland data set. The proposed approach shows a significant variation of accuracies on the data sets like SCADI, Sonar, Cleveland, Australian, and mushrooms.

When 15% label noise was added in the training data, the accuracy of all the methods decreased to some extent. The drop in accuracy in RSVM-PDProx is slighter than the rest of the methods (see Table 3.3). The best accuracies on all the data sets are

Table 3.3: Results of RSVM-PDProx and the Existing Methods with 15% Noise in the Real-World Data Sets

Data Sets	Results	SVM	SVM-PDProx	$\overline{\text{pin}}$-SVM	RSVM-RHHQ (η-val)	RSVM-PDProx					
						η=0.2	η=0.5	η=1	η=2	η=3	Parameter
SCADI	Accuracy	73.33±9.80	89.04±4.78	84.30±5.55	78.57±10.48 (0.2)	**92.38±5.71**	89.04±7.92	87.61±8.83	88.57±4.85	89.52±5.55	
	SVs	56.53	77.55	98.59	53.06	78.16	48.57	35.30	31.02	24.69	10^{-2}
	Time (in sec)	0.0009	0.086	1.31	0.008	0.14	0.17	0.25	0.24	0.25	
Sonar	Accuracy	71.11±5.109	70.37±5.98	74.08±4.91	72.53±4.97 (1)	72.53±5.01	73.49±5.89	**74.76±3.40**	74.12±6.19	72.58±6.40	
	SVs	69.17	53.58	75	71.72	54.65	76.34	49.72	47.79	41.86	10^{-6}
	Time (in sec)	0.001	2.23	0.0016s	0.001	2.72	2.62	2.65	2.72	2.60	
Cleveland	Accuracy	79.76±2.51	80.00±3.83	43.95±4.67	80.98±2.60 (0.2)	79.89±3.18	81.09±3.50	80.43±3.60	79.78±2.83	**81.53±3.67**	
	SVs	57.12	77.87	99.56	64.15	18.86	10.00	8.96	7.40	6.60	10^{-3}
	Time (in sec)	1.07	0.27	0.0036	0.357	5.94	6.32	6.45	6.50	6.50	
Haberman	Accuracy	72.60±4.47	73.26±3.15	67.03±4.28	73.36±2.71 (3)	71.63±5.98	**73.91±3.82**	72.195±4.24	71.41±2.70	72.50±3.70	
	SVs	67.42	86.77	86.38	64.95	27.71	16.96	9.20	7.71	7.28	10^{-4}
	Time (in sec)	0.038	1.39	0.038	0.001	2.02	6.98	7.13	7.19	7.15	
Australian	Accuracy	79.80±2.10	83.86±1.92	71.36±3.24	83.23±2.42 (0.5)	85.31±2.005	84.87±1.79	**85.50±1.85**	84.68±2.20	85.41±1.78	
	SVs	44.80	81.71	50.20	51.34	15.007	11.71	7.92	7.88	6.52	10^{-4}
	Time (in sec)	185.03	3.51	69.58	126.10	8.02	8.40	9.04	9.18	8.85	
Pima Indians	Accuracy	75.75±1.06	73.33±2.09	71.93±5.71	**75.88±2.35** (0.5)	73.46±1.43	73.81±2.36	75.32±2.13	72.68±2.04	73.73±2.76	
	SVs	66.10	69.53	93	65.36	22.36	16.38	16.20	16.18	16.35	10^{-4}
	Time (in sec)	5.55	10.74	2.92	3.20	47.38	40.24	48.47	48.17	48.43	
CMC	Accuracy	65.42±2.23	64.25±2.72	63.01±1.02	65.99±2.47 (1)	65.75±1.80	65.63±2.20	**66.38±2.17**	65.47±1.66	65.58±1.89	
	SVs	80.38	87.66	75.72	80.79	53.22	16.68	14.25	14.04	13.20	10^{-3}
	Time (in sec)	0.15	2.59	0.08	0.02	21.59	22.88	23.36	23.51	23.45	
Spambase	Accuracy	89.22±0.98	90.36±0.77	71.73±0.82	89.87±1.21 (3)	90.35±0.87	90.78±0.64	**91.16±0.70**	89.03±4.75	88.27±4.73	
	SVs	47.95	91.72	65.95	55.49	87.73	83.78	75.27	49.84	35.56	10^{-3}
	Time (in sec)	886.56	29.39	1.80	487.12	43.06	43.40	44.31	44.87	44.05	
Banana	Accuracy	56.16±2.11	55.16±2.78	**81.67±0.98**	55.70±1.05 (3)	56.71±1.40	56.16±2.61	50.18±5.92	53.97±3.57	50.64±5.67	
	SVs	94.60	100	65.95	92.58	95.99	56.60	73.22	28.20	18.89	10^{-7}
	Time (in sec)	0.26	10.95	1.80	0.11	25.49	14.41	18.73	26.25	26.79	
Page Blocks	Accuracy	91.23±0.89	89.22±2.56	91.03±0.72	92.35±0.69 (0.2)	**92.91±0.64**	91.63±0.74	88.14±2.53	87.89±2.49	87.56±0.59	
	SVs	29.83	95.34	98.21	34.35	15.30	17.20	12.53	11.25	10.11	10^{-5}
	Time (in sec)	490.31	89.21	0.56	617.78	252.26	253.64	258.78	260.55	260.88	
Musk (version 2)	Accuracy	93.71±0.76	82.26±0.46	88.07±1.11	99.57±0.07 (0.2)	99.49±0.07	99.55±0.09	**99.67±0.011**	99.48±0.07	99.60±0.03	
	SVs	41.81	83.94	96.40	83.23	53.47	51.009	36.12	6.28	6.27	10^{-6}
	Time (in sec)	13.37	204.81	4.53	996.57	58	59	58.12	57.66	60.18	
Mushrooms	Accuracy	94.77±0.36	95.11±0.38	78.82±2.11	94.71±0.36 (0.5)	95.11±0.38	95.82±0.71	**96.01±0.52**	95.11±0.42	94.59±0.37	
	SVs	51.99	54.28	46.15	55.90	82.28	66.97	33.74	7.45	7.72	10^{-6}
	Time (in sec)	7.08	159.73	119.97	5.25	22.19	21.77	23.15	22.69	21.73	

shown in boldface. The majority of boldfaced accuracies are seen with RSVM-PDProx. It again shows that the proposed method outperforms even in the presence of significant label noise. For conducting the experiments on noisy data set, the optimal parameter was obtained beforehand.

When 30% label noise is added to the training data, the accuracy of all the methods further decrease. This drop in the accuracy is most significant for $\overline{\text{pin}}$-SVM and the least significant for our proposed method, RSVM-PDProx (see Table 3.4).

Table 3.4: Results of RSVM-PDProx and the Existing Methods with 30% Noise in the Real-World Data Sets

Data Sets	Results	SVM	SVM-PDProx	$\overline{\text{pin}}$-SVM	RSVM-RHHQ (η-val)	RSVM-PDProx					
						η=0.2	η=0.5	η=1	η=2	η=3	Parameter
SCADI	Accuracy	67.61±8.98	86.66±5.12	78.50±9.28	68.09±11.47 (0.2)	86.66±7.61	85.23±7.81	**90.00±3.95**	84.76±8.19	85.23±5.81	
	SVs	68.57	84.89	98.42	77.55	80.40	66.32	49.79	35.91	43.26	10^{-6}
	Time (in sec)	0.0011	0.14	1.02	0.001	0.17	0.21	0.18	0.20	0.24	
Sonar	Accuracy	67.30±6.77	66.98±4.24	50.00±3.79	64.12±5.77 (0.2)	68.25±7.37	67.77±5.40	**71.42±4.65**	65.71±6.19	64.28±5.46	
	SVs	78.68	65.86	88.19	61.37	76.89	70.41	70.68	56.08	44.55	10^{-6}
	Time (in sec)	0.001	2.10	0.001	0.004	2.38	2.39	2.35	2.44	2.40	
Cleveland	Accuracy	75.71±3.20	77.25±2.69	46.15±2.91	75.27±3.78 (2)	78.68±3.34	78.02±3.57	77.58±4.63	**78.90±5.85**	78.68±4.23	
	SVs	67.78	90.23	99.52	76.41	22.83	11.83	10.42	8.20	6.12	10^{-3}
	Time (in sec)	0.93	0.26	0.004	0.54	6.14	6.47	6.39	6.53	6.49	
Haberman	Accuracy	71.63±4.25	60.32±9.46	61.53±2.69	72.17±3.41 (2)	71.52±3.16	**72.41±3.26**	71.08±3.71	70.86±4.23	71.30±4.53	
	SVs	81.49	92.009	88.26	91.58	23.87	17.89	17.19	19.06	13.50	10^{-5}
	Time (in sec)	0.022	1.11	0.0026	0.007	16.48	19.52	17.03	17.65	13.94	
Australian	Accuracy	81.59±2.82	84.63±1.69	59.12±3.11	82.80±2.97 (3)	84.20±1.60	83.81±1.28	81.15±5.09	83.57±2.97	**84.78±2.90**	
	SVs	58.13	94.94	67.42	72.04	14.24	11.80	13.47	8.34	8.19	10^{-4}
	Time (in sec)	140.10	3.42	92.43	74.11	16.90	16.70	17.14	17.50	17.60	
Pima Indians	Accuracy	**75.70±2.98**	72.46±2.42	61.52±6.31	72.20±4.26 (0.2)	72.68±2.03	70.21±2.06	72.55±1.28	71.47±3.00	73.16±2.37	
	SVs	77.16	78.99	100.00	84.72	22.94	17.72	17.54	18.28	16.55	10^{-2}
	Time (in sec)	5.40	9.11	3.05	0.008	47.58	48.03	48.21	48.37	48.42	
CMC	Accuracy	64.47±3.42	**66.80±2.28**	56.96±1.51	63.86±3.60 (1)	65.04±2.27	64.29±2.96	64.14±3.47	63.98±2.48	63.59±1.91	
	SVs	86.30	94.80	84.27	89.21	22.94	28.98	21.65	18.50	17.50	10^{-2}
	Time (in sec)	0.16	2.57	0.085	0.092	47.58	20.03	20.22	20.40	10.49	
Spambase	Accuracy	87.90±0.76	89.39±1.81	61.08±1.84	86.42±1.71 (2)	89.13±0.83	89.68±1.14	**90.23±0.89**	84.16±6.89	80.14±12.38	
	SVs	64.62	98.54	78.68	76.10	96.03	93.66	70.56	65.40	31.39	10^{-3}
	Time (in sec)	951.04	2.57	2.24	489.12	54.02	54.44	55.62	56.07	45.28	
Banana	Accuracy	53.58±3.91	55.109±4.004	**80.23±1.98**	55.47±0.78 (2)	56.17±3.50	54.89±3.64	54.69±4.30	53.88±5.59	51.11±3.19	
	SVs	94.55	96.55	76.86	94.59	96.19	82.45	85.02	51.22	24.31	10^{-5}
	Time (in sec)	0.178	81.61	751.08	0.12	25.00	26.16	26.24	26.48	26.51	
Page Blocks	Accuracy	89.57±3.02	83.08±3.34	86.17±1.07	**90.32±1.43** (0.5)	86.94±3.11	85.38±2.29	85.14±3.02	79.64±1.27	84.83±3.19	
	SVs	41.24	97.55	98.25	55.07	30.45	15.67	15.93	14.72	13.30	10^{-5}
	Time (in sec)	412.61	88.63	0.65	497.97	268.73	277.10	178.67	279.76	222.69	
Musk (version 2)	Accuracy	92.07±1.46	82.26±0.46	85.49±0.97	99.50±0.17 (0.5)	99.58±0.20	99.55±0.06	99.54±0.07	99.51±0.07	**99.61±0.13**	
	SVs	59.54	91.38	98.09	91.72	98.61	98.69	99.12	93.17	50.12	10^{-6}
	Time (in sec)	16.58	203.43	4.77	1263.26	62.24	62.35	64.11	61.96	71.49	
Mushrooms	Accuracy	92.91±020	92.41±1.40	64.25±3.47	91.96±2.66 (2)	91.29±1.89	85.11±2.36	91.68±0.92	93.24±1.19	**95.44±0.18**	
	SVs	66.69	71.57	64.48	78.67	90.69	80.39	33.32	16.09	14.06	10^{-8}
	Time (in sec)	7.68	159.67	412.45	7.91	22.34	22.85	23.58	23.53	23.76	

3.5.2.2 Sparseness

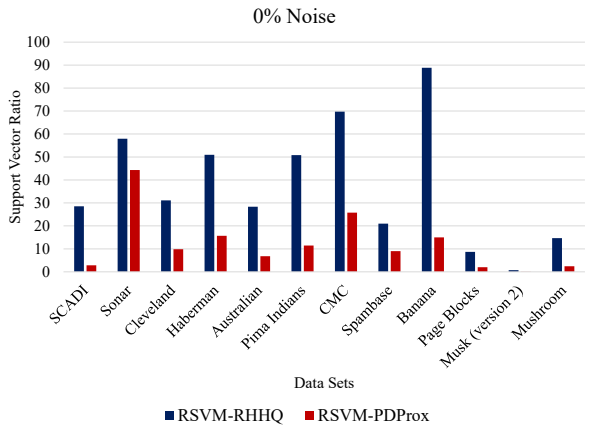
Sparsity is the property based on the number of variables used by the classifier in the classification. If the coefficient of a variable is 0, it does not affect the model. It is required to make the classifier dependent on as few variables as possible as the classification time of the classifier directly depends upon the non-zero dual variables. It also minimizes the tendency of overfitting, resulting in better generalizability. The use of a non-smooth regularizer improves the sparsity of the classifier. Therefore, in this section, the sparsity of the proposed classifier is under focus, and the support vector ratio is calculated for measuring sparseness in a model.

From the Tables 3.2, 3.3, and 3.4, it is evident that the support vector ratio for RSVM-PDProx is less than the rest of the methods for all the data sets. Another interesting observation is the decrease in support vector ratio when η is increased from $\eta = 0.2$ to $\eta = 3$. Also, as the noise increases in the data sets, the support vector ratio generally increases. Therefore, the support vector ratio is positively correlated with the label noise in the data set.

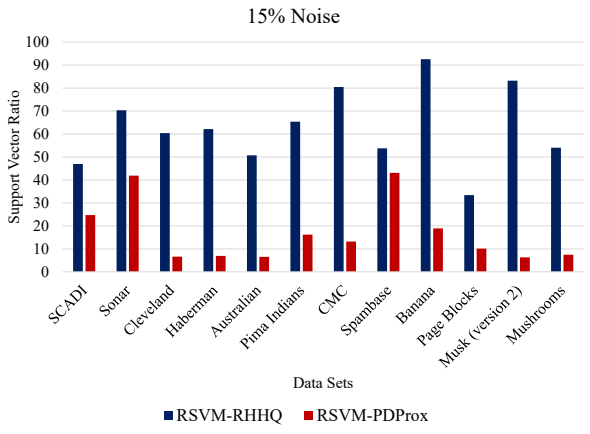
Figure 3.7 explicitly compares the support vector ratio of RSVM-PDProx and the RSVM-RHHQ method for all the three cases where data sets are noise-free, data sets with 15% noise, and data sets with 30% noise. In all the cases, RSVM-PDProx outperforms RSVM-RHHQ in terms of the support vector ratio.

The average support vector ratio of all the above-compared methods are listed in Table 3.5. From this table, it is evident that the proposed method is more sparse than the rest of the methods, and RSVM-PDProx shows the lower ratio of support vectors at $\eta = 3$, which is boldfaced in Table 3.5.

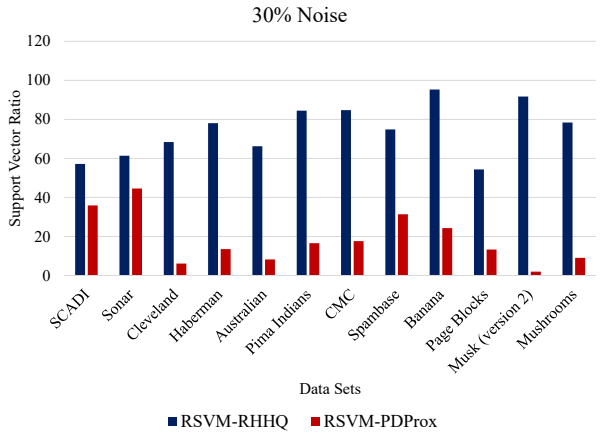
It should be noted here that the proposed approach is much more sparse than the existing methods; therefore, this method can be implemented in devices with low computational power and low memory.



(a) 0% Noisy Data Set



(b) 15% Noisy Data Set



(c) 30% Noisy Data Set

Figure 3.7: Comparison of RSVM-RHHQ and RSVM-PDProx Based on Support Vector Ratio over Real-World Data Sets

Table 3.5: Mean Support Vector Ratios Over All the Data Sets

Cases	SVM	SVM-PDProx	$\overline{\text{pin}}$-SVM	RSVM-RHHQ					RSVM-PDProx				
				η=0.2	η=0.5	η=1	η=2	η=3	η=0.2	η=0.5	η=1	η=2	η=3
0% noise	39.51	60.99	66.68	41.04	39.87	40.49	40.28	39.20	31.03	24.66	18.92	15.14	**13.03**
15% noise	58.97	79.99	79.25	65.42	63.31	65.76	65.29	66.65	50.39	39.34	31.03	19.58	**16.25**
30% noise	70.39	88.10	86.87	78.77	77.79	77.41	79.46	77.91	56.34	49.65	42.05	33.74	**23.57**

3.6 Discussion about dual variable and regularization parameter

The dual variable, α, in the equation of rescaled hinge loss, is updated by gradient mapping with another dual variable β. These variables are updated according to Steps 4 and 6 in Algorithm 1.

In this subsection, the effect of the regularization parameter, λ, on the support vector ratio and the accuracy of RSVM-PDProx is discussed. As the regularization parameter was obtained from the set, $\lambda \in \{10^{-a} : a \in \{-10, -9, -8, \cdots, -1\}\}$, the accuracy and the support vector ratio were computed on all the λ values of this set. For this experimentation, the synthetic data set was used. All the experiments are performed with a constant $\eta = 3$.

(i) *The effect of the regularization parameter on the accuracy of a classifier*

The accuracy graph is plotted over various regularization parameters. This plot is shown in Figure 3.8. The a from the set of regularization parameters is plotted on the x-axis, and the accuracy is plotted on the y-axis. From Figure 3.8, it is

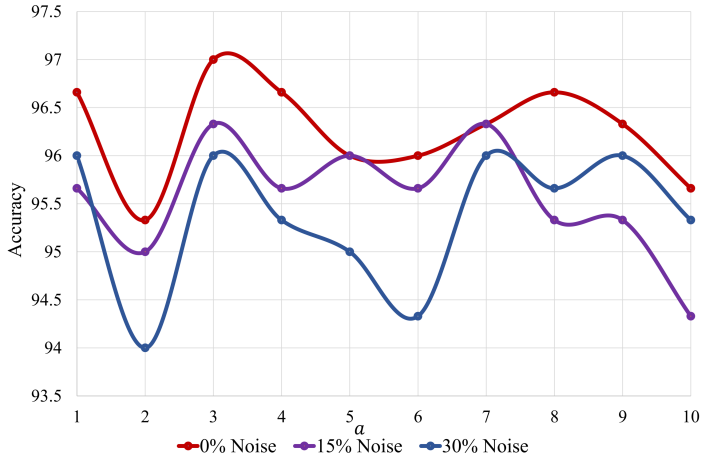


Figure 3.8: Effect of λ on the Accuracy at $\eta = 3$

observed that the maximum accuracy is generally achieved at $a = 3$, which implies 10^{-3} is the optimal parameter for this data set. However, the regularization parameter is data set dependent. Therefore, in all the experiments, the optimal parameter is selected first, even after adding noise to the same data set.

(ii) *The effect of the regularization parameter on the support vector ratio*

Here also, the a values from the set of regularization parameters are plotted on the x-axis, and now, it is plotted against the support vector ratio on the y-axis. This is shown in Figure 3.9. From Figure 3.9, it is observed that the support vector ratio first decreases up to $a = 6$, and after that, there is uncertainty in the support vector ratio. Another point observed from Figure 3.9 is the increase in support vector ratio as the noise in the data set increases.

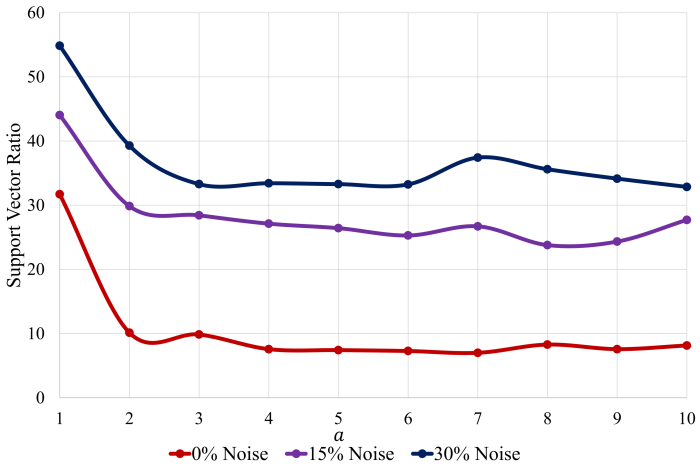


Figure 3.9: Effect of λ on the Support Vector Ratio at $\eta = 3$

3.7 Summary of the Work

In this work, an improved SVM version is proposed, which is robust towards label noise and has better sparseness. The rescaled α-hinge loss is used as the loss function (see (3.10)). This rescaled α-hinge loss is combined with a non-smooth regularizer to obtain a non-smooth objective function. This non-smooth function is optimized using the PDProx technique [79] which not only adds sparseness to the classifier but also better robustifies the classifier towards label noise (see Subsections 3.5.2.1 and 3.5.2.2). The time complexity of the optimization technique, which is $O(n^3)$, where n denotes the number of instances, is also provided. To apply the PDProx technique, FISTA is used. Therefore, the time taken by the proposed approach is comparatively more for some data sets. For larger data sets (Spambase, page blocks, and musk, etc.), where other techniques take quite long, RSVM-PDProx converges faster with better results in terms of both robustness and sparsity. The proposed technique outperforms other techniques for the majority of the data sets. Also, the support vector ratio has been noticed to be lesser in the case of RSVM-PDProx. Besides, it has been observed that the support

vector ratio is inversely proportional to the η values. Although, with the growing noise in the data sets, the support vector ratio increases for the proposed approach. However, it is much lesser than the existing techniques (see Table 3.5).