Machine Learning (2024) 113:1567–1594
https://doi.org/10.1007/s10994-022-06238-0

Parametric non-parallel support vector machines for pattern classification

Sambhav Jain[1] · **Reshma Rastogi**[1]

Received: 30 May 2022 / Revised: 15 August 2022 / Accepted: 12 September 2022 /
Published online: 19 October 2022

Abstract
This paper proposes Parametric non-parallel support vector machines for binary pattern classification. Through an intelligent redesigning of the Support vector machine optimisation, not only do we bring noise resilience into the model, but also retain its sparsity. Our model exhibits properties similar to Support vector machines, hence many SVM related learning algorithms can be extended to make it scalable for large scale problems. Experimental results on several benchmark UCI datasets validate our claims.

Keywords Support vector machines · Twin support vector machines · Pinball loss · Noise insensitivity · Sparsity

1 Introduction

Support vector machines (SVM) is a celebrated binary classifier built upon the direct practical application of the Statistical Learning Theory. The practicality and strong theoretical backing of SVMs have enabled them to be successfully adapted in a diverse variety of real-world applications ranging from bioinformatics (Yin et al., 2015), scene classification (Subasi, 2013), to power applications (Hao & Lewin, 2010). SVM has also attracted many researchers and inspired the development of new classifiers, like Generalised eigenvalue proximal support vector machines (GEPSVM) (Mangasarian & Wild, 2005), Twin Support vector machines (TSVM) (Khemchandani & Chandra, 2007). Each of them gave rise to an entirely new plethora of research.

TSVM showed that a two hyperplane rule is better adept at dealing with XOR(cross-planes) dataset, without the use of kernel methods. Also, leveraging the divide and conquer rule to solve individual quadratic programming problems (QPPs) for each

Editors: Yu-Feng Li, Prateek Jain.

✉ Sambhav Jain
sambhavjain0712@gmail.com

Reshma Rastogi
reshma.khemchandani@sau.ac.in

[1] Computer Science, South Asian University, Chanakyapuri, Delhi 110021, India

class, reduced the constraints by half and hence were around four times faster than the conventional SVMs. Moreover, TSVM was able to output comparable performance with SVMs.

Nevertheless, TSVM also has its own set of limitations like matrix inversion, sparsity and sensitivity to outliers due to L_2-norm. Each of these problems has captivated researchers' attention resulting in a new set of research.

TSVM adopts two hyperplane rule to classify samples which results in better accuracy than the SVM. However, the inconsistency between the training and prediction processes leads to sub-optimal results in certain cases. Authors in Shao et al. (2014), inculcated the two hyperplane prediction rule into their optimisation to address this issue. Similar to the TSVM their model minimises the respective class variances simultaneously maximising the projection difference between the classes.

Authors in Tian et al. (2013) presented an interesting extension to TSVM, that avoids matrix inversion, is sparse, and shows uniformity with the linear kernel and non-linear classifier. The first two sets of constraints in their optimisation enforce the respective class patterns to lie in the ϵ-bands. Their formulation shows correspondence with L_1-norm if ϵ tends to zero. The last set of constraints models the hinge loss to allow optimal separation from the opposite class patterns. Since the data points are implicitly modelled in the optimisation objective, it avoids matrix inversion. While on the other hand, explicit modulation of the data points in the constraints bestows the model properties such as sparsity as well as uniformity. Further to speed up the training process, fast SVM solvers can be adopted with little modification for their model.

All the aforementioned models in a way implement the hinge loss function which is unstable in the presence of noise. Traditionally, this problem has been dealt with by replacing the hinge loss with the pinball loss function. The pinball loss function focuses on maximising the quantile distance between the classes in contrast to hinge loss which focuses on the maximising closest set of points in the convex hull of the two classes. Alternatively, pinball loss grants noise resilience by penalising the correctly classified samples by a small amount and hence minimising the scatter as much as permissible. However, pinball loss trades noise resilience in exchange for the sparsity of the classifier. To attain back the sparsity, several pinball loss variants have been proposed, the most popular of them being the ϵ-pinball loss, truncated pinball loss etc. They do succeed in equipping the classifier with noise resilience and sparsity by increasing the number of variables, constraints and parameters of the model. The explicit impact is a more complicated optimisation having enormous time complexity.

In this research, we focus on some set of problems in SVM which are critical and a solution to them can help make SVM better applicable to real-world applications such as being noise resilience simultaneously maintaining sparsity and being time-efficient than the current state of the art pinball loss models.

Taking inspiration from the TSVM's proximal classification rule and through intelligent redesigning of the SVM optimisation, we achieve the solution to the problems above and our model stands in comparison with pinball loss models.

The present work is described in the following sections. Related work is reviewed in Sect. 2. Our proposed model PN-SVM is introduced in Sect. 3 and subsequently talks about the motivation behind the proposed model and its connection with the existing approaches. The experimental results are reported in Sect. 4, and finally, Sect. 5 concludes the paper.

2 Related work

In this paper, we denote class A as the set of samples having positive label and class B as the set of samples having negative label, i.e $A = \{(x_i, y_i = 1)\}$, and $B = \{(x_i, y_i = -1)\}$, $i = 1, 2, \dots, m_1 + m_2$, $D = [A;B]$, where $x_i \in X \subset R^N$. Labels of the samples are $y_i = +1$ for $i = 1, 2, \dots, m_1$, $y_i = -1$ for $i = m_1 + 1, m_1 + 2, \dots, n = m_1 + m_2$. For simplicity we write $Y = diag(y_1, \dots, y_{m_1}, y_{m_1+1}, \dots, y_{m_1+m_2})$. Here note that number of samples in class A, class B are denoted as m_1, m_2 respectively.

2.1 Support vector machines

State of the art binary classification algorithm, SVM is backed by statistical learning theory's (Vapnik, 1999) structural risk minimization principle. For a binary classification dataset D, SVM tends to find a hyperplane from a family of separating hyperplanes such that the width of the separation between the two classes is maximised. Consider the optimal hyperplane as: $J(w, b) = w^T x + b = 0$, where $b \in R$ is the bias term, $w \in R^N$ is the normal vector of $J(w, b)$. The optimal hyperplane J can be obtained by solving the following optimisation problem:

$$\begin{aligned} \min_{w,b,q} \quad & \frac{1}{2}\|w\|_2^2 \ + \ ce^T q, \\ s.t. \quad & Y(Dw \ + \ eb) + q \ \geq e, \\ & q \ \geq \ 0. \end{aligned} \tag{1}$$

here the minimization of first term (or the L_2-norm regularisation) ensures maximising margin or separation between the classes. e is the vector of ones of appropriate dimension. The second term is the sum of errors(q) of incorrectly positioned samples in the dead-zone or are incorrectly classified. The parameter c controls the importance of the two terms in the objective function. The first set of constraints dictates the data points projections to hyperplane, be at least unit distance away. However, if this is violated, then the error variable takes minimum value to satisfy the constraint and hence results in soft-margin hyperplane. For more details we advise the reader to refer to the original SVM paper (Cortes & Vapnik, 1995).

The dual solution, is generally sparse, implying that the SVM need only a fraction of the total data points to determine the optimal hyperplane. This property is called sparsity. There are several research works that tend to identify these data points from the training set. Since the classifier model is dependent on these points only, the training set can be reduced thereby making the model faster (Jung & Kim, 2013; Nalepa & Kawulok, 2019)

2.2 Support vector machines with pinball loss function

Authors in Huang et al. (2014) proposed Support vector machines with Pinball loss (Pin-SVM) taking into consideration the problems with SVMs namely: sensitivity towards noise and instability with respect to resampling. Their idea was to penalise the correctly classified samples by introducing a parameter $\frac{1}{\tau}$, so as to keep the correctly classified samples close to the separating hyperplane at the same time maximising the margin as in SVMs. The Pin-SVM formulation is as follows:

$$\begin{aligned} \min_{w,b,q} \quad & \frac{\|w\|_2^2}{2} + ce^T q, \\ s.t. \quad & Y(Dw + eb) + q \geq e, \\ & Y(Dw + eb) - \frac{q}{\tau} \leq e. \end{aligned} \tag{2}$$

Here c and τ are model parameters. As it is eminent from the above set of equations, Pin-SVM loses sparsity in exchange for noise insensitivity and resampling robustness. The issue of sparsity is handled by introducing a ϵ zone which is mentioned in the subsequent subsection.

2.3 Modified support vector machine with pinball loss

Authors in Rastogi et al. (2018) motivated by the properties of ϵ Pin-SVM, built a data-driven, asymmetric ϵ zone Pin-SVM termed as (ϵ_1, ϵ_2) Modified Pin Support Vector Machine (Mod-Pin-SVM). Please note that here we take into consideration the version 2 mentioned in their paper, we advise the reader to Rastogi et al. (2018) for more details.

$$\begin{aligned} \min_{w,b,q,\epsilon_2} \quad & \frac{\|w\|_2^2}{2} + c_1\epsilon_2 + c_2 e^T q, \\ s.t. \quad & Y(Dw + eb) + \left(\frac{q + \epsilon_1}{\tau_1}\right) \geq e, \\ & Y(Dw + eb) - \left(\frac{q + \epsilon_2}{\tau_2}\right) \leq e, \\ & q \geq 0, \quad \epsilon_2 \geq 0. \end{aligned} \tag{3}$$

Here c_1, c_2 τ_1, τ_2 and ϵ_1 being the model parameters. This model optimisation enforces the data points to lie in the respective ϵ zones, wherein they have zero value for the error variable q. Thus the obtained solution is naturally sparse. From the results reported, it is evident that their model achieves sparsity, also retaining the noise sensitivity and robustness to resampling properties. The downside is many model parameters, and increased training time due to the increased number of constraints and variables.

2.4 Twin support vector machines

Twin support vector machines (TSVM) find a pair of hyperplanes such that each hyperplane is proximal to the patterns of its own class and opposite class patterns are atleast unit distance away. It solves the following two sets of quadratic problems:

$$\begin{aligned} \min_{w_1,b_1,q_1} \quad & \frac{1}{2}\|Aw_1 + e_1 b_1\|_2^2 + c_1 e_2^T q_1, \\ s.t. \quad & -(Bw_1 + e_2 b_1) + q_1 \geq e_2, \\ & q_1 \geq 0. \end{aligned} \tag{4}$$

For brevity, we avoid writing the equation for the second hyperplane. Here q_1 is the error vector, and c_1 parameter controls the effectiveness between the L_2-norm square and the sum of errors. TSVM deals with only half of the constraints in each QPP, hence are four times faster than conventional SVM (Khemchandani & Chandra, 2007).

TSVM loses sparsity because of the two-loss functions (L_2 loss, hinge loss) used on the classes. Moreover, for large datasets, matrix inversion might be problematic. To overcome this, an appropriate small value ϵI (i.e. $(H^T H + \epsilon I)$) is added to make the matrices invertible. Once the class proximal hyperplanes $J_1(x) = w_1^T x + b_1$ and $J_2(x) = w_2^T x + b_2$ are found, the test data can be annotated according to the rule:

$$y = sign(abs(J_2(x)) - abs(J_1(x))). \tag{5}$$

To overcome the matrix inversion step in the dual, many alternatives have been put forward by the researchers (Chen et al., 2020; Shao et al., 2011). One solution to avoid matrix inversion is replacing L_2-norm with L_1-norm. Another reason for using L_1-norm in the primal objective of TSVM, rather than using L_2-norm is that the latter is highly influenced by outliers and noise in comparison to the former. Also L_1-norm is more robust than L_2-norm from experimental and statistical point of view (Li et al., 2016; Kwak, 2008).

Moreover, the use of L_1-norm can make the twin solution sparse. Note that a technique to solve L_1-norm has been applied in Twin Support Vector Clustering (Wang et al., 2015). It has been shown to converge to the optimal solution, but it would require more number iterations to converge, in a way increasing the complexity of the model.

2.5 L_1-norm twin support vector machines

Authors in Peng et al. (2016) motivated by the problems due to the L_2-norm, introduced L_1-norm based TSVM (L_1TSVM). The optimisation is as follows:

Hyperplane 1 (J_1):

$$\begin{aligned}
&\min_{w_1, b_1, q_1^+, q_1^-, q_2} \quad \frac{1}{2}(\|w_1\|_2^2 + b_1^2) + c_1 e_1^T (q_1^+ + q_1^-) + c_2 e_2^T q_2, \\
&s.t. \quad Aw_1 + e_1 b_1 = (q_1^+ - q_1^-), \\
&\qquad -(Bw_1 + e_2 b_1) + q_2 \geq e_2, \\
&\qquad q_1^{\pm} \geq 0, \; q_2 \geq 0.
\end{aligned} \tag{6}$$

This formulation avoids the matrix inversion step and the solution obtained is partly sparse. From the empirical observations reported in their paper (Peng et al., 2016), it is evident their model becomes less sensitive to outliers with the introduction of L_1-norm. The problem with L_1-norm TSVM is that it is highly dependent on the parameters. Looking at the constraints, it has n constraints in both optimisation Eqs. (6) and (7). So they lose the time efficacy of original TSVM, in-fact it requires almost twice the training time as compared to conventional SVM. Overall, they do succeed in introducing partial sparsity in TSVM, simultaneously avoiding the matrix inversion step.

2.6 Twin parametric margin support vector machine

Authors in Peng (2011) inspired from TSVM framework introduced Twin parametric margin support vector machines (TPMSVM). The main optimisation of their model is as follows:

$$
\begin{aligned}
\min_{w_1,b_1,q_1} \quad & \frac{1}{2}\|w_1\|_2^2 + \frac{v_1}{m_2}e_2^T(Bw_1 + e_2b_1) + \frac{c_1}{m_1}e_2^Tq_1, \\
s.t. \quad & (Aw_1 + e_1b_1) + q_1 \geq 0, \\
& q_1 \geq 0.
\end{aligned} \tag{7}
$$

Here v_1, c_1 being the model parameters. For the sake of brevity, we avoid writing the set of equations for the second hyperplane. The constraints dictate the Class A samples to lie on the positive half-space. Due to the minimisation of the second term in the objective function, class B samples will lie on the negative half-space keeping them as far as possible from the hyperplane. Notice that in this formulation the class A is kept in constraint and class B is kept in objective while calculating the hyperplane for class A. TPMSVM obtains marginal hyperplanes but TSVM obtains Proximal hyperplanes. A good advantage of this formulation is that it doesn't require any matrix inversion, simultaneously enjoying similar time complexity as TSVM. This model mainly finds applicability in scenarios where heteroscedastic noise is prevalent.

2.7 Twin-support vector machines with pinball loss

Authors in Xu et al. (2016) introduced pinball loss in Twin support vector machines (Pin-TSVM) to make it less sensitive to noise. They take Twin Parametric Margin Support Vector Machine (TPMSVM) as the building block of their research and model it to inculcate the pinball loss function. In their paper they have shown how Pin-TSVM is an extension to TPMSVM, we advise the reader to reference (Peng, 2011; Xu et al., 2016) for more details. The optimisation of Pin-TSVM is as follows:

$$
\begin{aligned}
\min_{w,b,q} \quad & \frac{1}{2}\|w\|_2^2 + \frac{v_1}{m_2}(Bw + e_2b) + \frac{c_1}{m_1}(e_1^Tq), \\
s.t. \quad & (Aw + e_1b) + q \geq e_1 * 0, \\
& (Aw + e_1b) - (\frac{q}{\tau_1}) \leq e_1 * 0.
\end{aligned} \tag{8}
$$

For the sake of brevity, we avoid writing the set of equations for the second hyperplane. Among the pinball loss models, Pin-TSVM does seem to be faster in terms of training time if there is no data imbalance, but it does not have a sparse solution.

3 Proposed model

3.1 Motivation

Conventional SVM has proved to be effective in many practical applications owing to its SRM principle which avoids overfitting.

SVM has cubic complexity which means it requires humongous training time for large datasets. TSVM is around 4 times faster than SVM due to the reduction in the number of constraints. TSVM still needs to solve two QPPs, as their decision rule dictates, to classify samples. Moreover, it is sensitive to outliers and needs to invert matrices due to the usage of squared loss on the proximal class. The L_1-norm TSVM addresses these problems by replacing L_2-norm by L_1-norm. However, it manages to be only partly sparse. This model served as the primal point of our research i.e. to further improve upon sparsity in the TSVM framework. Surprisingly, the design of PN-SVM turned out to be very similar to the Pinball loss models instead. Also, our proposed model is approximately four times (or more) faster than the Pin-SVM (and its generations). To sum up, not only does PN-SVM manages to address the noise instability problem in the SVM framework, but is also sparse, faster, and less complicated than Pin-SVM based models.

3.2 Formulation

3.2.1 Linear model

The mathematical formulation of our proposed model is as follows:

Hyperplane 1 (J_1):

$$\begin{aligned} \min_{w_1,b_1,q_1,q_2} \quad & \frac{1}{2}\|w_1\|_2^2 + \frac{v_1}{m_1}e_1^T(Aw_1 + e_1b_1) + \frac{c_1}{m_1+m_2}(e_1^Tq_1 + e_2^Tq_2), \\ & s.t.(Aw_1 + e_1b_1) + q_1 \geq 0 * e_1, \\ & \quad -(Bw_1 + e_2b_1) + q_2 \geq \epsilon * e_2, \\ & \quad q_1 \geq 0, \; q_2 \geq 0. \end{aligned} \tag{9}$$

Hyperplane 2 (J_2):

$$\begin{aligned} \min_{w_2,b_2,q_1',q_2'} \quad & \frac{1}{2}\|w_2\|_2^2 + \frac{v_1}{m_2}e_2^T(Bw_2 + e_2b_2) + \frac{c_1}{m_1+m_2}(e_1^Tq_1' + e_2^Tq_2'), \\ s.t. \quad & (Bw_2 + e_2b_2) + q_1' \geq 0 * e_2, \\ & -(Aw_2 + e_1b_2) + q_2' \geq \epsilon * e_1, \\ & q_1' \geq 0, \; q_2' \geq 0. \end{aligned} \tag{10}$$

Consider the equation for Hyperplane 1 (Eq. 9), the second term in the optimisation forces the hyperplane to be close to class A so that the mean of class A projections is minimised. The first constraint dictates the hyperplane to keep class A projections to be above the hyperplane, if not, q_1 (the slack vector) takes the minimum value to satisfy the constraint. Similarly, class B projections need to be at least ϵ unit away from the hyperplane. q_2 is the slack vector for class B samples. The minimisation of the first term in optimisation along with the third term adds the SRM principle to the model to avoid overfitting. Notice that the Eq. (9), without considering the second term is simply the bounding hyperplane equation for the SVM for class A. The minimisation of the mean of class A projections, in the objective, dictates the hyperplane to keep close to class A. Note that the nature of the hyperplane is still bounding if the sum of training errors is emphasised more than the mean of the projections i.e. $c_1 > v_1$. Please note that if $c_1 >> v_1$, or if the effectiveness of the second term in the objective is very small the hyperplanes (i.e. Eqs. 9 and 10) becomes parallel and hence converge to the conventional SVM. The minimisation of the mean allows the bounding hyperplanes to minimise the scatter of the classes individually. Since this is similar to penalising the correctly classified samples by some amount, it points towards an analogy with the pinball loss popularly used to equip SVMs with noise robustness.

For class A, the points which lie on the opposite side of the class A respective hyperplanes and the class B points which lie in the ϵ-margin of the hyperplane, contribute towards the training error and hence only few points contribute to the model building for class A, allowing sparsity. Similarly for class B.

The Eq. (9) can be simplified to as follows:

$$\begin{aligned} &\min_{w_1,b_1,q} \quad \frac{1}{2}\|w_1\|_2^2 + \frac{v_1}{m_1}e_1^T(Aw_1 + e_1b_1) + \frac{c_1}{m_1+m_2}e^Tq, \\ &s.t. \quad Y*(D_1w_1 + eb_1) + q \geq p_1, \\ &\qquad\quad q \geq 0. \end{aligned} \tag{11}$$

here $q = [q_1\,; q_2]$, $e = [e_1\,; e_2]$, $D_1 = [A;B]$, $p_1 = [0 * e_1\,; \epsilon * e_2]$

For calculating the dual of Eq. (11), we write the Lagrangian L_1 as follows:

$$\begin{aligned} L_1(w_1, b_1, q) &= \frac{1}{2}\|w_1\|_2^2 + \frac{v_1}{m_1}e_1^T(Aw_1 + e_1b_1) + \frac{c_1}{m_1+m_2}e^Tq \\ &- \alpha_1^T(Y*(D_1w_1 + eb_1) + q - p_1) - \beta_1^Tq = 0. \end{aligned} \tag{12}$$

Here α_1 and β_1 are the Lagrangian multipliers. The Karush-Kuhn-Tucker(KKT) necessary and sufficient conditions of Eq. (11) are given by:

$$\frac{\partial L_1}{\partial w_1} = \frac{v_1}{m_1}A^Te_1 + w_1 - (Y*D_1)^T\alpha_1 \Rightarrow w_1 = (Y*D_1)^T\alpha_1 - \frac{v_1}{m_1}A^Te_1, \tag{13}$$

$$\frac{\partial L_1}{\partial b_1} = \frac{v_1}{m_1}e_1^Te_1 - (Y*e)^T\alpha_1 = 0 \Rightarrow \alpha_1^TY = v_1, \tag{14}$$

$$\frac{\partial L_1}{\partial q} = \frac{c_1}{m_1+m_2}e - \alpha_1 - \beta_1 \Rightarrow \frac{c_1}{m_1+m_2}e = \alpha_1 + \beta_1, \tag{15}$$

$$\alpha_1^T(Y*(D_1w_1 + eb_1) + q - p_1) = 0,\ \alpha_1 \geq 0, \tag{16}$$

$$\beta_1^Tq = 0,\ \beta_1 \geq 0. \tag{17}$$

Substituting Eqs. (13)– (17) in Eq. (12), we get the dual of primal problem Eq. (11) which is as follows:

$$\begin{aligned} \min_{\alpha_1} \quad & \frac{\alpha_1^T Q_1 \alpha_1}{2} + F_1^T \alpha_1, \\ s.t. \quad & 0 \le \alpha_1 \le \frac{c_1}{m_1 + m_2} e, \\ & \alpha_1^T Y = v_1. \end{aligned} \tag{18}$$

Here $Q_1 = (Y * D_1)(Y * D_1)^T$ and $F_1 = -(\frac{v_1}{m_1}(Y * D_1)A^T e_1 + p_1)$.

Once the optimal α_1 is obtained, w_1 can be obtained from Eq. (13) and b_1 for the required hyperplane can be obtained by substituting w_1 in Eq. (16) as:

$$b_1 = \frac{1}{m_1 + m_2} e^T (diag(Y * p_1) - D_1 w_1). \tag{19}$$

Note that the diag() function column vectorizes the diagonal elements of the square matrix. Hence the required hyperplane is: $J_1(w_1, b_1) = w_1^T x + b_1 = 0$.

Similarly, the hyperplane for class B can be calculated.

$$\begin{aligned} \min_{\alpha_2} \quad & \frac{\alpha_2^T Q_2 \alpha_2}{2} + (F_2)^T \alpha_2, \\ s.t. \quad & 0 \le \alpha_2 \le \frac{c_1}{m_1 + m_2} e, \\ & \alpha_2^T \bar{Y} = v_1. \end{aligned} \tag{20}$$

Here $Q_2 = (\bar{Y} * D_2)(\bar{Y} * D_2)^T$ and $F_2 = -(\frac{v_1}{m_2}(\bar{Y} * D_2)B^T e_2 + p_2)$.

$p_2 = [0 * e_2; \epsilon * e_1]$, $\bar{Y} = -Y$, and we redefine $D_2 = [B;A]$.

$$\begin{aligned} w_2 &= (\bar{Y} * D_2)^T \alpha_2 - \frac{v_1}{m_2} B^T e_2, \\ b_2 &= \frac{1}{m_1 + m_2} e^T (diag(\bar{Y} * p_2) - D_2 w_2). \end{aligned} \tag{21}$$

Hence the required hyperplane for class B is: $J_2(w_2, b_2) = w_2^T x + b_2$. The test data sample, x can be annotated according to the rule as in Eq. (5).

3.2.2 Nonlinear model

In this section we extend the linear model to the nonlinear case. Here we directly use kernel matrices to deal with the non-linear data. The nonlinear model optimizes the following optimization problem:

$$\begin{aligned} \min_{w_1, b_1, q} \quad & \frac{1}{2}\|w_1\|_2^2 + \frac{v_1}{m_1} e_1^T (K(A, D_1) w_1 + e_1 b_1) + \frac{c_1}{m_1 + m_2} e^T q, \\ s.t. \quad & Y * (K(D_1, D_1) w_1 + e b_1) + q \ge p_1, \\ & q \ge 0. \end{aligned} \tag{22}$$

$$\begin{aligned} &\min_{w_2,b_2,q'} \quad \frac{1}{2}\|w_2\|_2^2 + \frac{v_1}{m_1}e_2^T(K(B,D_2)w_2 + e_2b_2) + \frac{c_1}{m_1+m_2}e^Tq', \\ &s.t. \quad \bar{Y} * (K(D_2,D_2)w_2 + eb_2) + q' \geq p_2, \\ &\qquad q' \geq 0. \end{aligned} \tag{23}$$

Here note that we have used the rectangular kernel technique, here it indicates that the linear kernel model and nonlinear formulations are non-uniform.

Similar to the linear case we can obtain the dual of the primal problem Eq. (22) which is as follows:

$$\begin{aligned} &\min_{\alpha_1} \frac{\alpha_1^T\hat{Q}_1\alpha_1}{2} + \hat{F}^T\alpha_1, \\ &s.t. \quad 0 \leq \alpha_1 \leq \frac{c_1}{m_1+m_2}e, \\ &\alpha_1^TY = v_1. \end{aligned} \tag{24}$$

Here $\hat{Q}_1 = (Y * \hat{D}_1)(Y * \hat{D}_1)^T$, $\hat{F} = -(\frac{v_1}{m_1}(Y * \hat{D}_1))K(A,D_1)^Te_1 + p_1)$ and $\hat{D}_1 = K(D_1,D_1)$.

Solving we get the above Eq. (24):

$$\begin{aligned} w_1 &= \alpha_1(Y * \hat{D}_1)^T - \frac{v_1}{m_1}e_1K(A,D_1)^T, \\ b_1 &= \frac{1}{m_1 + m_2}e^T(Y. * p_1 - \hat{D}_1w_1). \end{aligned} \tag{25}$$

Similarly we can obtain the dual of the primal problem Eq. (23) which is as follows:

$$\begin{aligned} &\min_{\alpha_2} \frac{\alpha_2^T\hat{Q}_2\alpha_2}{2} + \hat{F_2}^T\alpha_2, \\ &s.t. \quad 0 \leq \alpha_2 \leq \frac{c_1}{m_1+m_2}e, \\ &\alpha_2^T\bar{Y} = v_1. \end{aligned} \tag{26}$$

Here $\hat{Q}_2 = (\bar{Y} * \hat{D}_2)(\bar{Y} * \hat{D}_2)^T$, $\hat{F}_2 = -(\frac{v_1}{m_1}(\bar{Y} * \hat{D}_2))K(B,D_2)^Te_2 + p_2)$ and $\hat{D}_2 = K(D_2,D_2)$.

Solving the above Eq. (26) we get:

$$\begin{aligned} w_2 &= \alpha_2(\bar{Y} * \hat{D}_2)^T - \frac{v_1}{m_1}e_1K(B,D_2)^T, \\ b_2 &= \frac{1}{m_1 + m_2}e^T(\bar{Y}. * p_2 - \hat{D}_2w_2). \end{aligned} \tag{27}$$

The test data sample, x can be annotated according to the kernelized version of rule Eq. (5), which is as follows:

$$y = sign(abs(K(x,D) * w_2 + b_2) - abs(K(x,D) * w_1 + b_1)). \tag{28}$$

3.3 Algorithm

The algorithm for the proposed model is as follows:

Algorithm 1 Linear PN-SVM

1. Input Training Data A, B, D_1, D_2, Class labels Y, and model parameter v_1, c_1.
2. Define: $Q_1 = (Y * D_1)(Y * D_1)^T$, $\quad Q_2 = (\bar{Y} * D_2)(\bar{Y} * D_2)^T$,
   $F_1 = -(\frac{v_1}{m_1}(Y * D_1)A^T e_1 + p_1)$, $\quad F_2 = -(\frac{v_1}{m_2}(\bar{Y} * D_2)B^T e_2 + p_2)$.
3. Solve eq. (18), eq. (20) to get α_1, α_2
4. Obtain w_1,w_2 and b_1, b_2 as:
   $w_1 = (Y * D_1)^T \alpha_1 - \frac{v_1}{m_1} A^T e_1$,
   $w_2 = (\bar{Y} * D_2)^T \alpha_2 - \frac{v_1}{m_2} B^T e_2$,
   $b_1 = \frac{1}{m_1+m_2} e^T (Y. * p_1 - D_1 w_1)$,
   $b_2 = \frac{1}{m_1+m_2} e^T (\bar{Y}. * p_2 - D_2 w_2)$.
5. For any test sample x, annotate it by rule eq. (5).

Algorithm 2 Non-Linear PN-SVM

1. Input Training Data A, B, D_1, D_2, Suitable kernel function K, Class labels Y, and model parameter v_1, c_1.
   $\hat{D_1} = K(D_1, D_1)$, $\hat{D_2} = K(D_2, D_2)$.
2. Define: $\hat{Q_1} = (Y * \hat{D_1})(Y * \hat{D_1})^T$, $\quad \hat{Q_2} = (\bar{Y} * \hat{D_2})(\bar{Y} * \hat{D_2})^T$,
   $\hat{F_1} = -(\frac{v_1}{m_1}(Y * \hat{D_1}))K(A, D_1)^T e_1 + p_1)$, $\quad \hat{F_2} = -(\frac{v_1}{m_1}(\bar{Y} * \hat{D_2}))K(B, D_2)^T e_2 + p_2)$,
3. Solve eq. (24), eq. (26) to get α_1, α_2
4. Obtain w_1,w_2 and b_1, b_2 as:
   $w_1 = \alpha_1 (Y * \hat{D_1})^T - \frac{v_1}{m_1} e_1 K(A, D_1)^T$,
   $w_2 = \alpha_2 (\bar{Y} * \hat{D_2})^T - \frac{v_1}{m_1} e_1 K(B, D_2)^T$,
   $b_1 = \frac{1}{m_1 + m_2} e^T (Y. * p_1 - \hat{D}_1 w_1)$,
   $b_2 = \frac{1}{m_1 + m_2} e^T (\bar{Y}. * p_2 - \hat{D}_2 w_2)$.
5. For any test sample x, annotate it by rule eq. (28).

3.4 Connection with existing approaches

In this section, we briefly describe how our proposed model shares its connection with the existing approaches.

3.4.1 L_1-norm TSVM versus PN-SVM

In L_1-norm TSVM, the first set of constraints has an equality sign (Eq. 7) so as to develop a class A centric hyperplane. In our case, we don't keep the hyperplane centric to class A but instead, keep it close to the boundary of class A simultaneously minimising class A projections. This modification not only makes our model sparse but also allows it to act as a marginal/bounding or even a proximal hyperplane.

3.4.2 SVM versus PN-SVM

Our proposed model seems to have a similar formulation as SVM but the difference is the margin and the introduction of the second term in the optimisation Eq. (9). It is simply the bounding hyperplane equation for class A in SVM, if the mean of class A is ignored. It is evident that the onset of this change in optimisation makes SVM capture optimal variance of the respective class, hence the name PN-SVM.

3.4.3 Pin-SVM versus PN-SVM

The introduction of pinball loss pull-down sparsity in SVMs. To bring back sparsity ϵ zone-based Pin-SVM models (Rastogi et al., 2018) are developed. This in turn makes pinball loss models complicated. The addition of class A projections into the objective explicitly makes this process much simpler and time-efficient. Along similar lines, our model can be regarded as an improved version of Pin-TSVM on the advent of sparsity.

3.4.4 MCM versus PN-SVM

Minimal Complexity Machines (MCM) (Jayadeva,, 2015) also share homology with Pin-SVM models. Our model can be extended as a two hyperplane version of MCM, as it minimises the scatter of two classes separately. However, it does not have the time efficacy relationship as TSVM has compared with SVM. This can be a subject of further research.

3.4.5 TPMSVM versus PN-SVM

TPMSVM extends the idea of Parametric Margin-SVM (Hao, 2010) by solving two independent QPPs. Their model does not implement the SRM principle, thus resulting in sub-optimal results in certain cases. On the other hand, similar to TPMSVM, PN-SVM also tries to model marginal hyperplanes but is not limited to it, i.e. it can also model proximal hyperplanes on a suitable choice of parameters. Moreover, PN-SVM gives regard to the variance of the respective classes, simultaneously implementing the SRM principle and maintaining sparsity. In TPMSVM, the mean of the opposite class is used in the objective to model separation from the opposite class patterns. However, in PN-SVM mean of the respective class is used to model proximity to its own class patterns. On the basis of the aforementioned points, PN-SVM can be also regarded as an improved version of TPMSVM analogously. For more details on the TSVM based models we advise the reader to refer to a recently published survey paper (Tanveer et al., 2022).

3.5 Computation complexity

In this section, we compare the computational cost of our proposed PN-SVM with other related algorithms. During the training phase, similar to conventional SVM, the proposed model needs to find coefficients of n constraints for solving the dual QPP. Hence the computation of the proposed model is twice that of SVM is $2\mathcal{O}(n^3)$. As for the TSVM it needs to optimise a pair of smaller QPPs with approximately $n/2$ constraints, it has the computational complexity $2\mathcal{O}(\frac{n^3}{8})$, which is four times faster than the SVM. It should be pointed out that TSVM has some extra computational cost since it needs to invert a pair of matrices. For TPMSVM it only needs to solve two QPPs hence it has the computational complexity $2\mathcal{O}(\frac{n^3}{8})$. As for the pinball loss models it has $2n$ constraints so it should have computation of order $\mathcal{O}((2n)^3)$ except for Pin-TSVM which has a computation of approximately $2\mathcal{O}(n^3)$ only if the datasets are balanced (i.e. $m_1 = m_2$). Here note that the PN-SVM is theoretically four times faster than Pin-SVM.

During the testing phase, since the PN-SVM and SVM both are sparse, hence their computational cost depends only on the non-zero coefficients (or support vectors, sv) that is $\mathcal{O}(sv)$ Generally $sv << n$, hence due to the sparsity these models are faster in the testing phase. For the other models, TSVM and other non-sparse models testing complexity is simply $\mathcal{O}(n)$.

4 Experiments

The experiments are performed on well known diverse datasets using ten-fold cross-validation in MATLAB (Matlab, 2012) version 9.4 under Microsoft Windows environment on a machine with 3.40 GHz i7 CPU and 16 GB RAM. The optimal value of user-defined parameters in different models is obtained by fine-tuning a validation set generated using ten percent of training data. After the model parameters are known the validation set is sent back to training data for retraining. All datasets are normalised in the range $[1, -1]$. All comparing models are implemented in Matlab. Quadprog() function is used for all models to provide uniformity for comparison. For all two hyperplane models, parameters for both hyperplanes are kept the same. For measuring the sparsity of sparse models, we use Gini index. In Hurley and Rickard (2009) authors show that Gini index is well suited for evaluating sparsity for sparse models. Note that the lower the Gini index, the better the sparsity of the model. Note that TSVM, TPMSVM, Pin-TSVM and Pin-SVM are non-sparse models hence the Gini index for such non-sparse models is explicitly reported as 1 without any calculation. For measuring the time (seconds) of the algorithms we use tic-toc (Matlab). Note that we measure only the time taken by quadprog function by each algorithm for uniformity's sake. The training time taken is reported in seconds. The best testing accuracy and Gini index are highlighted in bold, and training accuracy is reported to check for overfitting. We use gaussian kernel ($K(x_1, x_2) = e^{-\rho\|x_1 - x_2\|^2}$) for all kernelised models, ρ is tuned in range $[0.1 : 0.1 : 1]$.

For our proposed model the setting of model parameters is very important. We tune v_1, c_1 in range $[0.1 : 0.1 : 0.9]$ and $[1 : 1 : 10]$ respectively. For brevity, we select the same value for the second hyperplane. However, the parameters for both hyperplanes can be tuned separately for better accuracy. Note that c_1 should be sufficiently larger that v_1 (i.e. $c_1 > v_1$).

For all our experiments we fix $\epsilon = 0.1$ just to avoid trivial solution for the optimisation.

4.1 Illustration on toy dataset

To show how our proposed model works, we show its plot on a synthetic two-dimensional dataset. The synthetic data is generated using mvnrnd function Matlab. We take the mean of class A as $[0.1, -1.8]^T$ and the mean of class B as $[-0.1, 1.8]^T$. Covariance matrices are $[0.3, 0.2; 0.2, 0.3]$ and $[0.3, -0.2; -0.2, 0.3]$ for the two classes respectively. The size of the two classes is taken to be 200 each. Refer to Fig. 1, it is easy to observe the differences among the various two hyperplane classifiers. Our proposed model does not run through the centre of the respective classes but it runs along the margin of the class such that the sum of projections of the respective class is minimised. Most of the samples lie on one side of the hyperplane but the trade-off between the second and third term in Eq. (9) forces some noisy samples to lie on the opposite side of the hyperplane (note that we have kept the parameter for the third term to be higher than that of the second term).

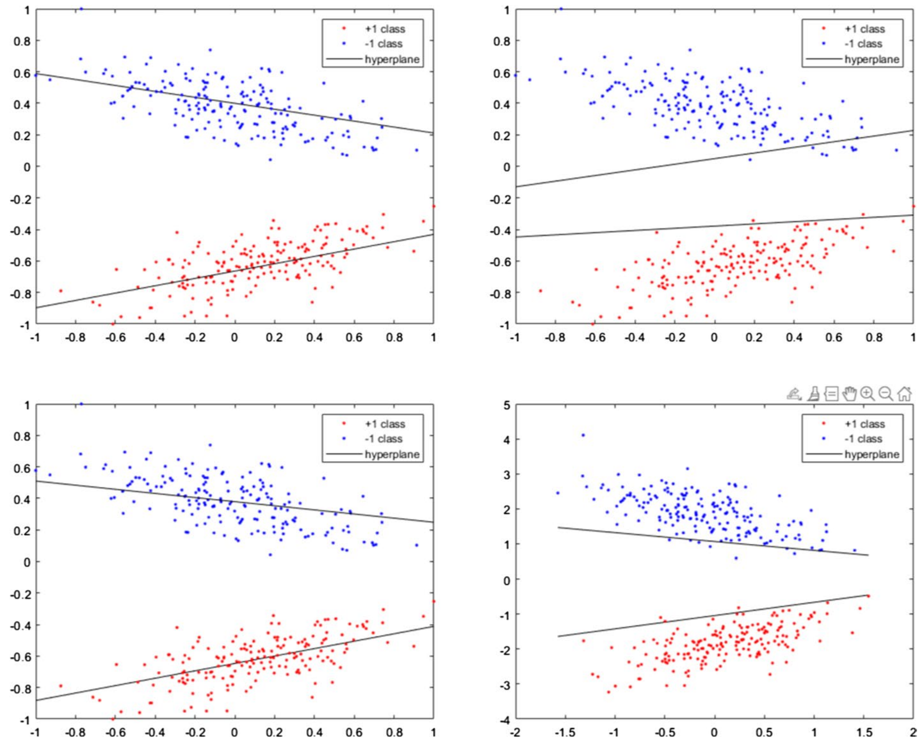

Fig. 1 Figures starting from top left to bottom right, are TSVM, TPMSVM, Pin-TSVM, and PN-SVM

The separation can be made better by significantly increasing the value of c_1. Doing so will allow the hyperplane to focus on more separation between the classes. In fact, if $c_1 >> v_1$ then it can be seen that the hyperplanes become parallel and the PN-SVM effectively behaves as conventional-SVM.

Look at Fig. 2, notice how increasing the value of c_1 changes the hyperplanes and the results for PN-SVM. So it's very important to select the parameters correctly.

4.2 Illustration on noise-corrupted datasets

The Fig. 3 shows the effect of noise on the classification accuracy of the two comparing models, namely PN-SVM and Pin-SVM. For a more accurate representation for comparison, natural log of testing accuracy is reported. Datasets are corrupted by adding noise using the mvrnd function Matlab. Dataset mean and covariance are used as input to the mvrnd function. Finally, some percentage of samples are corrupted by this generated noise. From the plots shown, it is evident that PN-SVM behaves approximately as Pin-SVM in the presence of noise. However, a comparison between their optimisations points toward functional analogy.

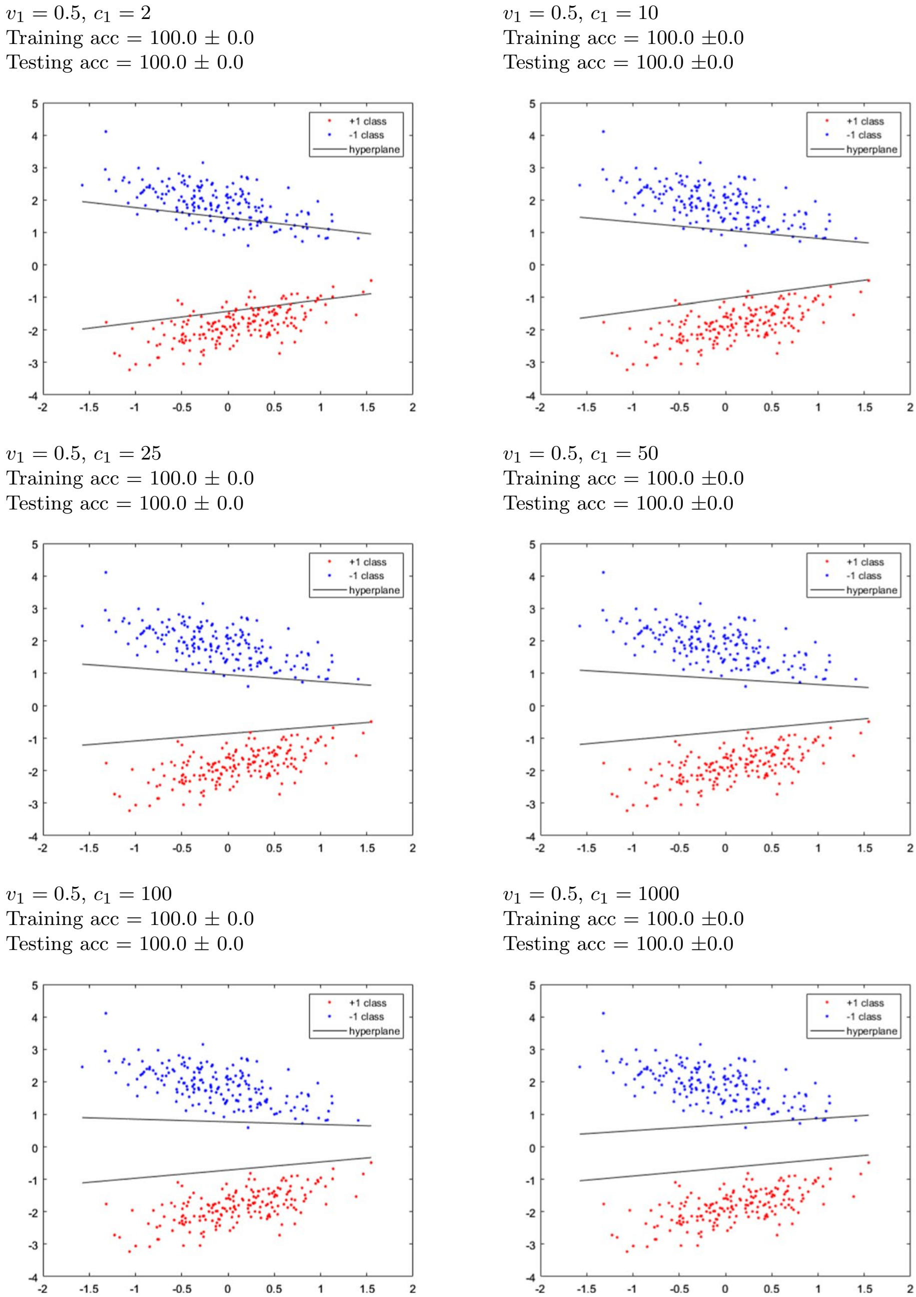

Fig. 2 Plots showing the effect of c_1 on PN-SVM

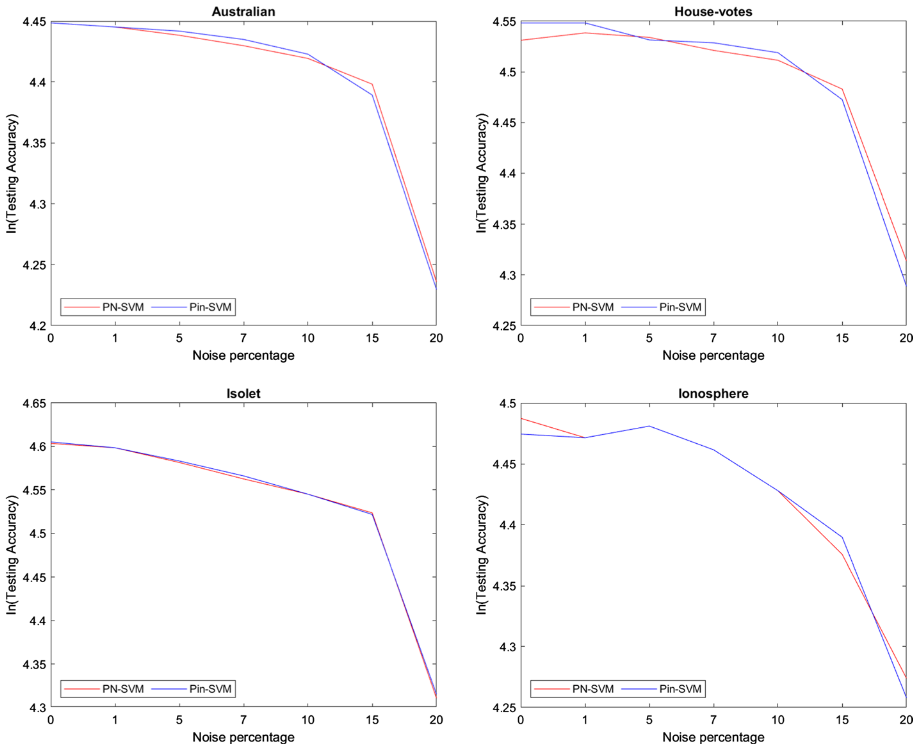

Fig. 3 Figures show the comparison between PN-SVM and Pin-SVM on varying percentage of noise on datasets

4.3 Datasets

The datasets are selected keeping into consideration the diversity of sizes. The characteristics of the datasets (Instances × features) used are described alongside the results. Datasets used are obtained from UCI machine learning repository (Asuncion & Newman, 2007) and Gunnar Rätsch's repository (Diethe, 2015).

4.4 Results for linear models

The Table 1 show the linear results for our proposed models PN-SVM against the comparing models. In comparison with TSVM and SVM, PN-SVM needs more training time. On the other hand, PN-SVM has comparable performance with these two and in many cases, it outputs greater Accuracy. Compared with TSVM and PN-SVM, the latter introduces sparsity into its framework which is comparable to the sparsity of SVM. In many cases, it exhibits superiority in this regard.

In comparison with pinball loss models, i.e. Pin-SVM, Pin-TSVM and Mod-Pin-SVM, the time efficacy of PN-SVM is clearly visible. Since pinball loss adds another set of constraints, it makes these models slower. Please note that our model is not the first attempt to introduce sparsity into the Pinball loss based models. However, it can be regarded as a

Table 1 Linear results on datasets

	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Thyroid (215 ×5)							
Time	0.003 ± 0	0.003 ± 0	0.007 ± 0	0.045 ± 0.003	0.004 ± 0	0.122 ± 0.01	0.109 ± 0.006
Training acc	81.707 ± 0.302	84.496 ± 1.316	87.288 ± 0.265	88.165 ± 0.342	86.978 ± 0.259	78.811 ± 0.267	79.174 ± 0.212
Testing acc	80.974 ± 2.685	81.818 ± 2.047	**86.58 ± 2.37**	86.515 ± 1.873	85.649 ± 2.564	78.139 ± 2.705	78.593 ± 2.89
Gini-Index	1 ± 0	1 ± 0	0.569 ± 0.004	1 ± 0	**0.528 ± 0.005**	0.76 ± 0.003	1 ± 0
Heart-statlog (270 × 13)							
Time	0.004 ± 0	0.004 ± 0	0.014 ± 0.002	0.057 ± 0.005	0.004 ± 0	0.202 ± 0.005	0.172 ± 0.007
Training acc	85.35 ± 0.302	83.991 ± 0.263	85.267 ± 0.306	83.621 ± 0.28	85.144 ± 0.339	85.432 ± 0.359	82.099 ± 1.188
Testing acc	83.704 ± 2.222	84.444 ± 2.194	84.444 ± 2.124	82.222 ± 2.457	**84.815 ± 2.435**	84.074 ± 2.533	76.296 ± 2.222
Gini-Index	1 ± 0	1 ± 0	**0.514 ± 0.003**	1 ± 0	0.576 ± 0.004	0.791 ± 0.003	1 ± 0
Breast cancer (277 ×9)							
Time	0.004 ± 0	0.003 ± 0	0.01 ± 0	0.067 ± 0.003	0.006 ± 0	0.922 ± 0.354	0.128 ± 0.009
Training acc	72.964 ± 0.973	75.851 ± 0.306	73.124 ± 0.251	67.51 ± 0.833	73.004 ± 0.329	73.164 ± 0.232	73.244 ± 0.3
Testing acc	72.183 ± 2.4	**75.396 ± 2.563**	72.54 ± 2.481	67.884 ± 1.929	71.429 ± 2.648	71.415 ± 2.928	72.526 ± 2.608
Gini-index	1 ± 0	1 ± 0	0.433 ± 0.004	1 ± 0	**0.432 ± 0.005**	0.712 ± 0.006	1 ± 0
Ionosphere (351 × 33)							
Time	0.005 ± 0	0.007 ± 0	0.017 ± 0	0.115 ± 0.004	0.008 ± 0.001	0.402 ± 0.014	0.361 ± 0.017
Training acc	88.382 ± 0.366	86.61 ± 1.217	92.118 ± 0.285	89.174 ± 0.192	91.01 ± 0.296	91.516 ± 0.253	89.49 ± 0.384
Testing acc	86.024 ± 1.513	86.888 ± 2.264	88.31 ± 1.784	87.175 ± 1.554	**88.881 ± 1.678**	88.31 ± 2.11	87.738 ± 2.005
Gini-index	1 ± 0	1 ± 0	**0.621 ± 0.003**	1 ± 0	0.638 ± 0.004	0.832 ± 0.002	1 ± 0
House votes (435 × 16)							
Time	0.009 ± 0.001	0.008 ± 0	0.025 ± 0.001	0.192 ± 0.007	0.012 ± 0	0.613 ± 0.021	0.498 ± 0.027
Training acc	94.611 ± 0.199	89.731 ± 0.173	94.483 ± 0.132	89.706 ± 0.228	94.56 ± 0.152	95.147 ± 0.161	94.483 ± 0.132
Testing acc	94.26 ± 1.283	89.423 ± 1.367	**94.958 ± 1.164**	88.964 ± 1.52	93.113 ± 1.169	93.8 ± 1.178	94.503 ± 1.182
Gini-index	1 ± 0	1 ± 0	**0.505 ± 0.003**	1 ± 0	0.793 ± 0.002	0.914 ± 0.002	1 ± 0
Isolet (600 × 51)							

Table 1 (continued)

	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Time	0.013 ± 0.001	0.012 ± 0	0.057 ± 0.003	0.315 ± 0.006	0.025 ± 0.001	1.469 ± 0.057	1.687 ± 0.111
Training acc	100 ± 0	99.148 ± 0.092	99.963 ± 0.025	99.315 ± 0.04	99.944 ± 0.028	100 ± 0	99.963 ± 0.025
Testing acc	**100 ± 0**	99.333 ± 0.272	99.833 ± 0.167	99 ± 0.444	99.667 ± 0.222	99.833 ± 0.167	99.833 ± 0.167
Gini-index	1 ± 0	1 ± 0	**0.593 ± 0.001**	1 ± 0	0.876 ± 0.001	0.964 ± 0.001	1 ± 0
Australian (690 × 14)							
Time	0.017 ± 0.001	0.011 ± 0	0.08 ± 0.005	0.503 ± 0.011	0.056 ± 0.004	3.498 ± 0.124	1.41 ± 0.069
Training acc	81.127 ± 0.532	87.165 ± 0.133	85.507 ± 0.12	85.507 ± 0.12	85.507 ± 0.12	85.507 ± 0.12	85.507 ± 0.12
Testing acc	79.855 ± 1.519	**87.246 ± 1.053**	85.507 ± 1.08	85.507 ± 1.08	85.507 ± 1.08	85.507 ± 1.08	85.507 ± 1.08
Gini-index	1 ± 0	1 ± 0	**0.297 ± 0.002**	1 ± 0	0.691 ± 0.005	0.855 ± 0.001	1 ± 0
Pimadiabetes (768 × 8)							
Time	0.017 ± 0	0.026 ± 0.001	0.092 ± 0.004	0.749 ± 0.022	0.04 ± 0.002	3.083 ± 0.101	2.547 ± 0.065
Training acc	76.389 ± 0.219	75.607 ± 0.261	77.054 ± 0.229	75.014 ± 0.272	76.664 ± 0.223	77.879 ± 0.163	77.575 ± 0.234
Testing acc	76.044 ± 1.571	75.521 ± 1.309	75.916 ± 1.791	74.621 ± 1.905	75.918 ± 1.941	76.56 ± 1.714	**76.562 ± 1.632**
Gini-index	1 ± 0	1 ± 0	**0.321 ± 0.002**	1 ± 0	0.376 ± 0.003	0.648 ± 0.003	1 ± 0
German (1000 × 20)							
Time	0.045 ± 0.004	0.031 ± 0.001	0.166 ± 0.01	1.795 ± 0.03	0.077 ± 0.004	6.112 ± 0.237	5.68 ± 1.077
Training acc	73.967 ± 0.399	74.244 ± 0.473	76.622 ± 0.152	71.878 ± 0.286	76.089 ± 0.157	77.078 ± 0.178	70 ± 0.194
Testing acc	72.9 ± 1.716	73 ± 1.92	**75.6 ± 1.166**	70.7 ± 1.984	75.1 ± 1.917	75.6 ± 1.634	70 ± 1.745
Gini-index	1 ± 0	1 ± 0	**0.362 ± 0.002**	1 ± 0	0.447 ± 0.003	0.685 ± 0.002	1 ± 0
Flare-solar (1066 × 9)							
Time	0.033 ± 0.002	0.05 ± 0.002	0.165 ± 0.006	1.379 ± 0.029	0.091 ± 0.007	97.807 ± 0.292	4.065 ± 0.288
Training acc	60.601 ± 0.306	67.542 ± 0.133	67.542 ± 0.134	57.557 ± 0.198	67.542 ± 0.134	67.542 ± 0.134	67.542 ± 0.134
Testing acc	60.136 ± 1.046	**67.55 ± 1.205**	**67.551 ± 1.205**	57.328 ± 1.257	**67.551 ± 1.205**	**67.551 ± 1.205**	**67.551 ± 1.205**
Gini-index	1 ± 0	1 ± 0	0.278 ± 0.002	1 ± 0	**0.274 ± 0.002**	0.675 ± 0.001	1 ± 0
CMC (1473 × 9)							
Time	0.052 ± 0.004	0.072 ± 0.003	0.467 ± 0.018	4.209 ± 0.074	0.22 ± 0.011	20.345 ± 0.488	17.597 ± 0.754

Table 1 (continued)

	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Training acc	67.994 ± 0.157	62.993 ± 0.139	67.361 ± 0.191	62.39 ± 0.398	66.893 ± 0.187	67.361 ± 0.198	66.018 ± 0.167
Testing acc	67.016 ± 1.438	63.143 ± 1.394	**67.288 ± 1.711**	61.037 ± 1.408	66.811 ± 1.435	66.743 ± 1.288	65.725 ± 1.299
Gini-index	1 ± 0	1 ± 0	**0.225 ± 0.002**	1 ± 0	0.24 ± 0.002	0.548 ± 0.002	1 ± 0
Image (2310 × 18)							
Time	0.246 ± 0.008	0.421 ± 0.004	1.619 ± 0.029	15.726 ± 0.16	0.72 ± 0.027	67.973 ± 2.293	87.802 ± 3.595
Training acc	74.728 ± 0.337	69.624 ± 1.538	83.098 ± 0.118	70.188 ± 0.267	83.213 ± 0.098	83.603 ± 0.101	74.411 ± 0.1
Testing acc	74.113 ± 1.06	70.043 ± 1.235	82.727 ± 0.843	70.563 ± 0.91	82.771 ± 0.727	**83.593 ± 0.751**	74.329 ± 0.901
Gini-index	1 ± 0	1 ± 0	**0.429 ± 0.001**	1 ± 0	0.461 ± 0.001	0.733 ± 0.001	1 ± 0
Krvskp (3196 × 36)							
Time	0.404 ± 0.013	0.735 ± 0.015	3.201 ± 0.066	30.456 ± 0.456	1.866 ± 0.058	1722.95 ± 169.771	210.822 ± 25.778
Training acc	93.791 ± 0.458	83.931 ± 0.068	94.086 ± 0.035	92.539 ± 0.067	95.286 ± 0.035	95.293 ± 0.036	90.707 ± 0.038
Testing acc	93.616 ± 0.552	83.728 ± 0.504	94.086 ± 0.316	92.397 ± 0.269	**95.213** ± 0.276	**95.213 ± 0.276**	90.707 ± 0.339
Gini-index	1 ± 0	1 ± 0	**0.585 ± 0.002**	1 ± 0	0.84 ± 0.001	0.95 ± 0.001	1 ± 0
Splice (3175 × 60)							
Time	0.447 ± 0.015	0.611 ± 0.005	3.498 ± 0.058	28.058 ± 0.454	1.696 ± 0.028	239.438 ± 9.606	407.888 ± 178.936
Training acc	84.591 ± 0.099	82.988 ± 0.066	85.533 ± 0.102	84.64 ± 0.059	85.676 ± 0.123	85.963 ± 0.079	85.071 ± 0.072
Testing acc	84.001 ± 0.542	82.52 ± 0.779	**84.726 ± 0.754**	84.032 ± 0.608	84.694 ± 0.879	84.725 ± 0.706	84.159 ± 0.642
Gini-index	1 ± 0	1 ± 0	**0.572 ± 0.001**	1 ± 0	0.641 ± 0.001	0.804 ± 0.001	1 ± 0
Waveform (5000 × 21)							
Time	1.944 ± 0.072	2.92 ± 0.041	12.819 ± 0.218	286.301 ± 7.634	5.122 ± 0.061	869.604 ± 22.482	620.592 ± 18.47
Training acc	79.184 ± 0.064	84.493 ± 0.05	87.287 ± 0.066	83.122 ± 0.035	79.976 ± 0.048	88.569 ± 0.035	82.511 ± 0.053
Testing acc	79.14 ± 0.571	84.56 ± 0.357	87.08 ± 0.467	83.14 ± 0.359	80 ± 0.468	**88.22 ± 0.329**	82.46 ± 0.612
Gini-index	1 ± 0	1 ± 0	**0.31 ± 0**	1 ± 0	0.679 ± 0.001	0.825 ± 0.001	1 ± 0
Statlog (6435 × 36)							
Time	6.048 ± 0.099	5.185 ± 0.113	27.28 ± 0.369	471.211 ± 7.303	10.645 ± 0.215	1936.922 ± 68.712	1542.161 ± 55.481

Table 1 (continued)

	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Training acc	98.429 ± 0.02	89.47 ± 0.051	98.662 ± 0.017	96.445 ± 0.037	98.097 ± 0.017	98.648 ± 0.013	98.358 ± 0.025
Testing acc	98.337 ± 0.129	89.51 ± 0.507	**98.632 ± 0.148**	96.55 ± 0.258	98.011 ± 0.165	98.539 ± 0.145	98.337 ± 0.191
Gini-index	1 ± 0	1 ± 0	**0.762 ± 0**	1 ± 0	0.935 ± 0	0.97 ± 0	1 ± 0
Twonorm (7400 × 20)							
Time	4.266 ± 0.21	6.171 ± 0.038	40.794 ± 0.323	237.057 ± 4.982	13.478 ± 0.19	3978.745 ± 111.452	2134.632 ± 137.898
Training acc	97.866 ± 0.021	97.665 ± 0.02	97.896 ± 0.022	97.761 ± 0.018	97.899 ± 0.024	97.94 ± 0.019	97.784 ± 0.021
Testing acc	97.824 ± 0.182	97.689 ± 0.164	97.811 ± 0.191	97.757 ± 0.192	97.838 ± 0.188	**97.865 ± 0.182**	97.716 ± 0.176
Gini-index	1 ± 0	1 ± 0	**0.689 ± 0**	1 ± 0	0.891 ± 0	0.946 ± 0	1 ± 0

The best results are highlighted in bold

Table 2 Linear results on NDC datasets

	TSVM	PN-SVM	Pin-TSVM	SVM	Pin-SVM	Mod-Pin-SVM
NDC-550 ($550 \times 32 \times 0.519$)						
Time	0.013 ± 0.001	0.05 ± 0.004	1.365 ± 0.031	0.021 ± 0.002	0.964 ± 0.024	1.25 ± 0.041
Training acc	87.273 ± 0.221	88.04 ± 0.25	86.606 ± 0.259	87.798 ± 0.222	88.101 ± 0.336	89.394 ± 0.228
Testing acc	85.273 ± 1.446	85.636 ± 1.745	85.091 ± 1.377	84.364 ± 1.585	**86.182 ± 1.414**	85.818 ± 1.688
Gini-index	1 ± 0	**0.585 ± 0.002**	1 ± 0	0.714 ± 0.003	1 ± 0	0.858 ± 0.002
NDC-770 ($770 \times 32 \times 0.565$)						
Time	0.022 ± 0.002	0.102 ± 0.004	3.385 ± 0.091	0.054 ± 0.004	2.638 ± 0.176	3.826 ± 0.122
Training acc	85.325 ± 0.205	86.364 ± 0.26	84.315 ± 0.215	86.392 ± 0.131	86.609 ± 0.216	87.244 ± 0.197
Testing acc	83.636 ± 1.527	82.857 ± 1.393	82.597 ± 1.438	82.597 ± 1.287	84.156 ± 1.617	**85.065 ± 1.359**
Gini-index	1 ± 0	**0.567 ± 0.002**	1 ± 0	0.698 ± 0.003	1 ± 0	0.84 ± 0.002
NDC-1100 ($1100 \times 32 \times 0.594$)						
Time	0.042 ± 0.002	0.225 ± 0.007	10.176 ± 0.12	0.109 ± 0.005	7.211 ± 0.236	9.198 ± 0.155
Training acc	85.374 ± 0.094	86.455 ± 0.086	83.778 ± 0.08	85.828 ± 0.093	86.242 ± 0.123	86.737 ± 0.152
Testing acc	83.727 ± 0.934	84.273 ± 0.949	83.182 ± 0.86	83.455 ± 0.811	**84.909 ± 0.804**	84.818 ± 0.791
Gini-index	1 ± 0	**0.559 ± 0.001**	1 ± 0	0.686 ± 0.001	1 ± 0	0.827 ± 0.001
NDC-1650 ($1650 \times 32 \times 0.594$)						
Time	0.104 ± 0.004	0.616 ± 0.012	34.648 ± 0.336	0.29 ± 0.007	20.827 ± 0.698	29.468 ± 0.686
Training acc	85.717 ± 0.082	85.892 ± 0.165	84.411 ± 0.115	85.502 ± 0.108	85.65 ± 0.094	86.316 ± 0.128
Testing acc	84.606 ± 0.621	84.424 ± 0.745	83.818 ± 0.6	83.939 ± 0.672	84.788 ± 0.941	**85.091 ± 0.724**
Gini-index	1 ± 0	**0.548 ± 0.001**	1 ± 0	0.664 ± 0.002	1 ± 0	0.815 ± 0.001
NDC-2200 ($2200 \times 32 \times 0.609$)						
Time	0.228 ± 0.007	1.305 ± 0.019	89.123 ± 0.995	0.727 ± 0.021	51.355 ± 2.212	71.275 ± 1.49
Training acc	85.924 ± 0.1	86.747 ± 0.082	84.753 ± 0.104	86.111 ± 0.095	86.97 ± 0.093	87.172 ± 0.116
Testing acc	85.091 ± 0.704	**85.955 ± 0.786**	84.227 ± 0.922	85.182 ± 0.702	85.727 ± 0.56	85.455 ± 0.701
Gini-index	1 ± 0	**0.559 ± 0.001**	1 ± 0	0.683 ± 0.002	1 ± 0	0.824 ± 0.001
NDC-2750 ($2750 \times 32 \times 0.613$)						

Table 2 (continued)

	TSVM	PN-SVM	Pin-TSVM	SVM	Pin-SVM	Mod-Pin-SVM
Time	0.409 ± 0.01	2.234 ± 0.042	169.986 ± 1.413	1.237 ± 0.04	96.591 ± 6.049	133.296 ± 3.099
Training acc	86.133 ± 0.103	86.453 ± 0.096	84.735 ± 0.081	85.903 ± 0.084	86.4 ± 0.082	86.873 ± 0.128
Testing acc	85.782 ± 0.722	85.927 ± 0.566	84.073 ± 0.698	85.164 ± 0.565	86.036 ± 0.874	**86.618 ± 0.768**
Gini-index	1 ± 0	**0.553 ± 0.00**1	1 ± 0	0.677 ± 0.002	1 ± 0	0.821 ± 0.001
NDC-3300 (3300 × 32 × 0.620)						
Time	0.628 ± 0.017	3.999 ± 0.054	297.632 ± 3.739	2.02 ± 0.041	157.506 ± 4.588	252.195 ± 7.458
Training acc	86.131 ± 0.088	86.316 ± 0.067	85.013 ± 0.07	85.734 ± 0.062	86.276 ± 0.059	86.865 ± 0.085
Testing acc	85.576 ± 0.837	85.727 ± 0.623	84.727 ± 0.611	84.97 ± 0.656	86 ± 0.519	**86.182 ± 0.539**
Gini-index	1 ± 0	**0.554 ± 0.001**	1 ± 0	0.677 ± 0.001	1 ± 0	0.822 ± 0.001
NDC-3850 (3850 × 32 × 0.610)						
Time	1.021 ± 0.037	5.868 ± 0.156	491.897 ± 11.915	3.254 ± 0.093	283.294 ± 14.544	361.685 ± 10.747
Training acc	85.789 ± 0.073	85.893 ± 0.062	84.716 ± 0.063	85.668 ± 0.05	86.115 ± 0.06	86.58 ± 0.054
Testing acc	85.403 ± 0.402	85.506 ± 0.456	84.597 ± 0.476	85.039 ± 0.52	85.61 ± 0.46	**85.818 ± 0.419**
Gini-index	1 ± 0	**0.554 ± 0.001**	1 ± 0	0.674 ± 0.001	1 ± 0	0.817 ± 0
NDC-4400 (4400 × 32 × 0.622)						
Time	1.381 ± 0.026	8.487 ± 0.246	752.603 ± 30.17	4.521 ± 0.127	417.903 ± 28.569	553.392 ± 16.922
Training acc	85.821 ± 0.055	86.318 ± 0.061	84.929 ± 0.049	85.967 ± 0.053	86.035 ± 0.047	86.702 ± 0.054
Testing acc	85.318 ± 0.443	85.818 ± 0.476	84.841 ± 0.462	85.568 ± 0.52	85.705 ± 0.433	**86 ± 0.455**
Gini-index	1 ± 0	**0.552 ± 0.001**	1 ± 0	0.672 ± 0.001	1 ± 0	0.817 ± 0.001

The best results are highlighted in bold

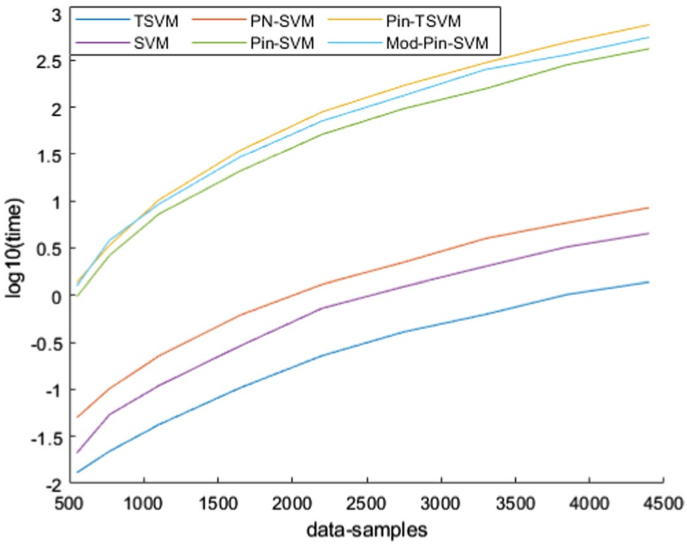

Fig. 4 Time comparison on NDC datasets

better, simpler model, in terms of training time, accuracy and sparsity as compared with Pinball loss based models. Our model has superior sparsity in comparison with Mod-Pin-SVM. Our model adopts the sparsity due to the SVM framework and noise resilience is granted by the minimisation of the mean of respective class projections.

4.5 Results for NDC datasets

To clearly demonstrate the effect of dataset size on the training time of the linear classifiers we conduct experiments on NDC datasets (Musicant, 1998). The characteristic of these datasets is described in the Table 2 as samples × dimension × Imbalance. Imbalance is defined as the ratio of number of positive labelled samples to the number of negative labelled samples in the dataset. The Table 2 show the linear results for our proposed models namely PN-SVM against the comparing models. The dimension of the NDC datasets is fixed (i.e. 32) while the sample size is varied. For all the datasets the model parameters are fixed to study the change of time with increasing data size. It can be seen that as the imbalance ratio increases, the pin-TSVM model becomes more time-consuming. In fact from the Fig. 4, it can be seen that its curve grows the fastest among all the other models. Also, it is clear that the pinball loss models are very heavy on training time compared to our proposed model PN-SVM. Among these three, TSVM is the fastest, followed by SVM and PN-SVM. So it is important to improve upon the time requirement of the pinball models.

4.6 Results for non-linear models

The Table 3 show the kernelised results for our proposed models namely PN-SVM against the comparing models. Due to the higher computational requirement of Pinball models, we experiment on medium-sized datasets. Our proposed model PN-SVM is able to output comparable results and achieves superior sparsity. Similar to the observations in the linear

Table 3 Non-linear results on datasets

	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Thyroid (215 × 5)							
Time	0.006 ± 0	0.003 ± 0	0.014 ± 0.001	0.075 ± 0.003	0.005 ± 0	0.398 ± 0.013	0.281 ± 0.014
Training acc	97.21 ± 0.191	92.972 ± 0.299	89.717 ± 0.288	91.68 ± 0.22	89.768 ± 0.272	89.82 ± 0.274	82.637 ± 0.506
Testing acc	**96.775 ± 1.369**	92.684 ± 2.262	89.848 ± 2.424	91.688 ± 2.007	89.848 ± 2.424	89.848 ± 2.424	81.861 ± 2.82
Gini-index	1 ± 0	1 ± 0	0.647 ± 0.003	1 ± 0	**0.633 ± 0.005**	0.844 ± 0.002	1 ± 0
Heart-statlog (270 × 13)							
Time	0.009 ± 0.001	0.003 ± 0	0.015 ± 0	0.097 ± 0.002	0.006 ± 0	0.462 ± 0.015	0.485 ± 0.017
Training acc	96.379 ± 0.286	83.374 ± 0.353	85.391 ± 0.283	77.984 ± 0.215	84.774 ± 0.342	84.691 ± 0.28	83.58 ± 0.557
Testing acc	78.889 ± 2.533	83.703 ± 2.222	83.704 ± 2.601	77.778 ± 1.991	83.704 ± 2.542	**84.074 ± 2.28**	77.407 ± 2.025
Gini-index	1 ± 0	1 ± 0	**0.51 ± 0.003**	1 ± 0	0.571 ± 0.004	0.787 ± 0.002	1 ± 0
Breast cancer (277 × 9)							
Time	0.009 ± 0	0.004 ± 0	0.017 ± 0.001	0.117 ± 0.003	0.008 ± 0	0.903 ± 0.269	1.468 ± 0.666
Training acc	84.077 ± 0.673	76.654 ± 0.261	77.417 ± 0.34	68.472 ± 0.411	76.896 ± 0.515	75.612 ± 0.305	75.371 ± 0.292
Testing acc	70.794 ± 2.491	76.124 ± 2.018	**76.151 ± 2.693**	67.156 ± 3.736	75.053 ± 2.781	73.598 ± 2.978	73.981 ± 2.897
Gini-index	1 ± 0	1 ± 0	**0.453 ± 0.005**	1 ± 0	0.459 ± 0.005	0.728 ± 0.003	1 ± 0
Ionosphere (351 × 33)							
Time	0.016 ± 0.001	0.006 ± 0	0.027 ± 0.001	0.28 ± 0.006	0.012 ± 0.001	1.048 ± 0.028	1.123 ± 0.057
Training acc	99.462 ± 0.068	94.238 ± 0.168	96.011 ± 0.096	68.345 ± 0.393	95.283 ± 0.138	94.777 ± 0.118	94.397 ± 0.095
Testing acc	93.452 ± 1.277	94.023 ± 1.498	**95.444 ± 0.969**	68.667 ± 2.361	93.167 ± 1.658	93.452 ± 1.593	93.738 ± 1.259
Gini-index	1 ± 0	1 ± 0	**0.706 ± 0.002**	1 ± 0	0.794 ± 0.003	0.897 ± 0.003	1 ± 0
House votes (435 × 16)							
Time	0.023 ± 0.001	0.006 ± 0	0.043 ± 0.001	0.446 ± 0.008	0.019 ± 0.001	1.942 ± 0.149	9.309 ± 2.23
Training acc	99.642 ± 0.057	91.392 ± 0.126	95.07 ± 0.142	86.539 ± 0.172	94.662 ± 0.158	95.096 ± 0.233	94.458 ± 0.147
Testing acc	**94.487 ± 1.338**	89.423 ± 1.636	93.805 ± 1.441	86.427 ± 1.699	93.34 ± 1.092	93.113 ± 1.776	93.584 ± 1.391
Gini-index	1 ± 0	1 ± 0	**0.703 ± 0.001**	1 ± 0	0.808 ± 0.002	0.888 ± 0.002	1 ± 0
Isolet (600 × 51)							

Table 3 (continued)

	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Time	0.048 ± 0.005	0.007 ± 0	0.098 ± 0.002	0.575 ± 0.015	0.043 ± 0.004	4.279 ± 0.127	5.211 ± 0.231
Training acc	100 ± 0	99.463 ± 0.051	99.704 ± 0.03	97.722 ± 0.133	99.667 ± 0.025	99.667 ± 0.025	99.815 ± 0.028
Testing acc	**100 ± 0**	99.5 ± 0.355	99.667 ± 0.222	97.833 ± 0.558	99.667 ± 0.222	99.667 ± 0.222	99.5 ± 0.255
Gini-index	1 ± 0	1 ± 0	**0.77 ± 0.001**	1 ± 0	0.933 ± 0.001	0.967 ± 0.001	1 ± 0
Australian (690 × 14)							
Time	0.063 ± 0.004	0.009 ± 0	0.123 ± 0.006	1.369 ± 0.043	0.06 ± 0.004	7.749 ± 0.219	8.906 ± 0.479
Training acc	91.514 ± 0.125	86.392 ± 0.202	88.1 ± 0.177	85.491 ± 0.201	85.62 ± 0.096	85.556 ± 0.132	85.507 ± 0.12
Testing acc	85.652 ± 0.928	**87.101 ± 1.111**	85.652 ± 1.068	85.362 ± 1.024	85.507 ± 1.08	85.507 ± 1.08	85.507 ± 1.08
Gini-index	1 ± 0	1 ± 0	**0.607 ± 0.002**	1 ± 0	0.704 ± 0.002	0.853 ± 0.001	1 ± 0
German (1000 × 20)							
Time	0.164 ± 0.004	0.024 ± 0	0.275 ± 0.006	3.11 ± 0.041	0.132 ± 0.004	18.826 ± 0.697	34.621 ± 10.211
Training acc	95.067 ± 0.111	80.755 ± 0.125	79.656 ± 0.169	68.633 ± 0.384	78.822 ± 0.175	78.922 ± 0.154	77.533 ± 0.291
Testing acc	73 ± 1.563	75.3 ± 1.445	**76.2 ± 1.062**	66.6 ± 0.833	75.6 ± 1.641	75.9 ± 1.215	75.5 ± 1.241
Gini-index	1 ± 0	1 ± 0	**0.43 ± 0.002**	1 ± 0	0.473 ± 0.003	0.701 ± 0.001	1 ± 0
Pimadiabetes (768 × 8)							
Time	0.101 ± 0.015	0.022 ± 0	0.157 ± 0.007	1.581 ± 0.032	0.074 ± 0.003	8.72 ± 0.24	8.418 ± 0.339
Training acc	79.876 ± 0.282	73.263 ± 0.254	78.212 ± 0.225	69.343 ± 0.142	76.939 ± 0.194	77.503 ± 0.209	77.778 ± 0.267
Testing acc	77.09 ± 1.432	74.742 ± 1.161	**77.21 ± 1.652**	69.287 ± 1.546	76.309 ± 1.926	76.564 ± 1.652	76.557 ± 1.741
Gini-index	1 ± 0	1 ± 0	**0.424 ± 0.003**	1 ± 0	0.439 ± 0.003	0.669 ± 0.003	1 ± 0
Flare-solar (1066 × 9)							
Time	0.162 ± 0.009	0.069 ± 0.001	0.325 ± 0.008	3.249 ± 0.066	0.157 ± 0.007	190.059 ± 37.481	113.171 ± 38.685
Training acc	67.897 ± 0.164	65.791 ± 0.864	67.511 ± 0.142	65.499 ± 0.105	67.542 ± 0.134	67.542 ± 0.134	67.542 ± 0.134
Testing acc	66.324 ± 1.259	67.174 ± 1.033	67.078 ± 1.244	65.577 ± 1.055	**67.551 ± 1.205**	**67.551 ± 1.205**	**67.551 ± 1.205**
Gini-index	1 ± 0	1 ± 0	**0.271 ± 0.002**	1 ± 0	0.318 ± 0.002	0.674 ± 0.001	1±0

The best results are highlighted in bold

Table 4 Ranks with linear classifiers for datasets

Datasets	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Thyroid	5	4	**1**	2	3	7	6
Heart-statlog	5	2	2	6	**1**	4	7
Breast cancer	4	**1**	2	7	5	6	3
Ionosphere	7	6	2	5	**1**	2	4
House votes	3	6	**1**	7	5	4	2
Isolet	**1**	6	2	7	5	2	2
Australian	7	**1**	2	2	2	2	2
Pimadiabetes	3	6	5	7	4	2	**1**
German	5	4	**1**	6	3	**1**	7
Flare-solar	6	5	**1**	7	**1**	**1**	**1**
CMC	2	6	**1**	7	3	4	5
Image	5	7	3	6	2	**1**	4
Krvskp	4	7	3	5	**1**	**1**	6
Splice	6	7	**1**	5	3	2	4
Waveform	7	3	2	4	6	**1**	5
Statlog	3	7	**1**	6	5	2	3
Twonorm	3	7	4	5	2	**1**	6
Avg rank	4.47	5.00	**2.00**	5.53	3.06	2.53	4.00

The best results are highlighted in bold

Table 5 Ranks with non-linear classifiers for datasets

Datasets	TSVM	TPMSVM	PN-SVM	Pin-TSVM	SVM	Mod-Pin-SVM	Pin-SVM
Thyroid	**1**	2	4	3	4	4	7
Heart-statlog	5	4	2	6	2	**1**	7
Breast cancer	6	2	**1**	7	3	5	4
Ionosphere	4	2	**1**	7	6	4	3
House votes	**1**	6	2	7	4	5	3
Isolet	**1**	5	2	7	2	2	5
Australian	2	**1**	2	7	4	4	4
Pimadiabetes	2	6	**1**	7	5	3	4
German	6	5	**1**	7	3	2	4
Flare-solar	6	4	5	7	**1**	**1**	**1**
Avg rank	3.4	3.7	**2.1**	6.5	3.4	3.1	4.2

The best results are highlighted in bold

case, the time requirement of our model, here is higher than SVM and TSVM, but still lesser in comparison to Pinball models.

4.7 Statistical analysis

We use ranking criteria to evaluate better the results obtained from the algorithms. The ranks are allotted to each algorithm according to the order in which they perform w.r.t to a metric, i.e. if a model has the best results, it is ranked 1; if it has a second-best result, it is given rank 2 on a particular dataset. If two algorithms have the same results, they are given the same rank. Mean rank obtained via averaging ranks for all datasets w.r.t to a metric is reported in Tables 4 and 5. It can be seen that the proposed algorithm PN-SVM achieves the best rank amongst the comparing algorithms for linear as well as non-linear classifiers.

5 Conclusions

In this paper, we have proposed the PN-SVM model to handle binary classification, wherein we effectively address the problem of sparsity and noise resilience through reformulation of the SVM optimisation framework by minimising average projections of the respective classes. The results on benchmark datasets prove the efficacy of the proposed model against the comparing approaches. Further, our proposed model presents a faster and much simpler model than the Pinball loss models. PN-SVM exhibits homology to SVM. Hence the fast SVM type approaches can be modified to adapt to our model. Further, it would be interesting to develop its extension for multiclass classification.

Acknowledgements Authors of this manuscript would like to express gratitude to Dr S. Chandra for their valuable feedback and Dr Aman Pal for providing Matlab codes for Mod-Pin-SVM, Pin-SVM. We would also like to thank South Asian University for monthly stipend to Sambhav Jain, which helped conduct this research.

Author contributions SJ: Conceptualization, data curation, formal analysis, investigation, methodology, software, supervision, validation, visualization, writing—original draft, writing - review and editing. RR: Conceptualization, formal analysis, investigation, methodology, software, supervision, validation, visualization, writing— review and editing.

Funding No funding was received for conducting this study.

Data availibility The datasets used in this study are freely available online at UCI machine learning repository (Asuncion and Newman, 2007) and Gunnar Rätsch's repository (Diethe, 2015).

Code availability The code for the proposed model can be made available on request from the corresponding author.

Declarations

Conflict of interest There are no conflicts of interest in this study.

Ethics approval This research paper does not involve any studies with human participants or animals performed by any authors.

References

Asuncion, A., & Newman, D. (2007). *Uci machine learning repository*.

Chen, W. J., Shao, Y. H., Li, C. N., et al. (2020). ν-projection twin support vector machine for pattern classification. *Neurocomputing, 376*, 10–24.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273–297.

Diethe, T. (2015). 13 benchmark datasets derived from the UCI, DELVE and STATLOG repositories. https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets/, https://doi.org/10.5281/zenodo.18110

Hao, L., & Lewin, P. (2010). Partial discharge source discrimination using a support vector machine. *IEEE Transactions on Dielectrics and electrical Insulation, 17*(1), 189–197.

Hao, P. Y. (2010). New support vector algorithms with parametric insensitive/margin model. *Neural Networks, 23*(1), 60–73.
Huang, X., Shi, L., & Suykens, J. (2014). Support vector machine classifier with pinball loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2013.178
Hurley, N., & Rickard, S. (2009). Comparing measures of sparsity. *IEEE Transactions on Information Theory, 55*(10), 4723–4741.
Jayadeva. (2015). Learning a hyperplane classifier by minimizing an exact bound on the vc dimensioni. *Neurocomputing, 149*, 683–689.
Jung, H. G., & Kim, G. (2013). Support vector number reduction: Survey and experimental evaluations. *IEEE Transactions on Intelligent Transportation Systems, 15*(2), 463–476.
Khemchandani, R., Chandra, S., et al. (2007). Twin support vector machines for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(5), 905–910.
Kwak, N. (2008). Principal component analysis based on l1-norm maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(9), 1672–1680.
Li, C. N., Shao, Y. H., & Deng, N. Y. (2016). Robust l1-norm non-parallel proximal support vector machine. *Optimization, 65*(1), 169–183. https://doi.org/10.1080/02331934.2014.994627
Mangasarian, O. L., & Wild, E. W. (2005). Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(1), 69–74.
Matlab, S. (2012). *Matlab*. Natick, MA: The MathWorks.
Musicant, D.R. (1998). NDC: Normally distributed clustered datasets. Www.cs.wisc.edu/dmi/svm/ndc/
Nalepa, J., & Kawulok, M. (2019). Selecting training sets for support vector machines: A review. *Artificial Intelligence Review, 52*(2), 857–900.
Peng, X. (2011). Tpmsvm: A novel twin parametric-margin support vector machine for pattern recognition. *Pattern Recognition, 44*(10–11), 2678–2692.
Peng, X., Xu, D., Kong, L., et al. (2016). L1-norm loss based twin support vector machine for data recognition. *Information Sciences*. https://doi.org/10.1016/j.ins.2016.01.023.
Rastogi, R., Pal, A., & Chandra, S. (2018). Generalized pinball loss SVMs. *Neurocomputing, 322,* 151–165.
Shao, Y. H., Zhang, C. H., Wang, X. B., et al. (2011). Improvements on twin support vector machines. *IEEE Transactions on Neural Networks, 22*(6), 962–968.
Shao, Y. H., Chen, W. J., & Deng, N. Y. (2014). Nonparallel hyperplane support vector machine for binary classification problems. *Information Sciences, 263*, 22–35.
Subasi, A. (2013). Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders. *Computers in Biology and Medicine, 43*(5), 576–586.
Tanveer, M., Rajani, T., Rastogi, R., et al. (2022). Comprehensive review on twin support vector machines. *Annals of Operations Research*, 1–46.
Tian, Y., Qi, Z., Ju, X., et al. (2013). Nonparallel support vector machines for pattern classification. *IEEE Transactions on Cybernetics, 44*(7), 1067–1079.
Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks, 10*(5), 988–999.
Wang, Z., Shao, Y. H., Bai, L., et al. (2015). Twin support vector machine for clustering. *IEEE Transactions on Neural Networks and Learning Systems, 26*(10), 2583–2588.
Xu, Y., Yang, Z., & Pan, X. (2016). A novel twin support-vector machine with pinball loss. *IEEE Transactions on Neural Networks and Learning Systems, 28*(2), 359–370.
Yin, H., Jiao, X., Chai, Y., et al. (2015). Scene classification based on single-layer SAE and SVM. *Expert Systems with Applications, 42*(7), 3368–3380.