EBSVM: A Sparsification Method From LP-Based SVM via Stochastic Functional ξ-Analysis

Rezaul Karim*, Zakir Hossain, Amit Kumar Kundu, Ali Ahmed Ave
Department of Electrical and Electronic Engineering, Uttara University, Dhaka, Bangladesh
E-mail: rezkar@uttarauniversity.edu.bd
*Corresponding author

Keywords: Classifier, stochastic, LP, QP, sparse, SVM, EBSVM (efficiency boosted SVM), accuracy, MAC, GFR, compact, data, mapping

Received: March 28, 2025

While SVM being a solid mathematical and state-of-the-art sparse but powerful stochastic machine with top accuracy, Vapnik's LP based SVM is sparser than the QP generated one though offers similar accuracy. Yet, further sparsity is essential to efficiently work with very complicated and large datasets. In our work, we propose a novel efficient method, EBSVM by imposing additional sparsification on this sparser LPSVM maintaining optimal complexity of the machine to further reduce classification cost while preserving similar accuracy by analyzing the stochastic auxiliary variable, ξ after a compact deterministic mapping of data. Experiments with Gunnar Rätsch benchmark data as well as synthetic data reveals that our model demands classification cost much smaller than the cutting-edge prominent classifiers while posing similar accuracy despite of being simpler to be implemented. On average, it costs 12.4% of the standard QPSVM, 50.3% of the VLPSVM, 68.1% of the EESVM [1], 98.1% of the greedy SpSVM-2 [2]. It insists the least MAC (Machine Accuracy Cost) value as well as GFR(Generalization Failure Rate) considering these machines. Numerical calculus based noise-sensitivity analysis proves our model's potentiality in working with noisy data. This machine is also expected to work with high efficiency in case of very large and complex data satisfying constraints in real-time.

Povzetek: Članek predstavi EBSVM za učinkovitejšo klasifikacijo s pomočjo stohastične ξ-analize in redčenja LP-SVM, ki odstranita zavajajoče KCV ter optimizirata pristranskost, omogočata bolj kvalitetno posploševanje kot metoda QPSVM, z hitrejšimi odločitvami klasifikacije.

1 Introduction & selected related work

In stochastic learning, the principal objective in a probabilistic data-classification problem is in general to model a discriminator function that effectively represents the generalized relationship of the input objects or patterns with their matching class-labels applying the given dataset having some example patterns with their associated class-labels. While in such indeterministic frameworks, performance from kernel-based classifiers is influenced by the size, randomness of distribution and topological variations of the dataset, the complexity of constructing such classifiers and determining its decision boundary by involving kernel operating patterns is also related to the probabilistic nature of data, classifiers' geometry and available computational resources. Thus different machines show different performances on data detection demanding different computational cost. In this regard while Support Vector Machines (SVMs) lead the field of data classification by offering top level accuracy through finding global optima, a key stochastic property of SVMs is their sparsity, which enables the construction of the optimal classifier using only a probabilistically selected subset of training patterns, known as support vectors (SVs). This sparsity reduces computational demands but as the number of SVs increases, computational cost from nonlinear SVM classifiers also gets higher. However, although Vapnik's Linear Programming (LP) based SVM is even sparser and demands much less computational cost, it also demands the expensive kernel execution task. Moreover, as science and technology have advanced, the size as well as the complexity of data have grown rapidly, and as a result, the number of support vectors usually rises with the growth of the the size of training dataset [3]. This makes the further but compact sparsity of the support vector set a critical and continuous consideration for large-scale datasets, especially in scenarios constrained by computational resources and time. As a result, developing methods to address this challenge has become a central focus of research, aiming optimal performance by balancing stochastic generalization with computational efficiency.
Tom Downs et al. [4] offers a method to detect linearly dependent support vectors (SVs), retaining only one and discarding the rest but the number of this linearly dependent SVs tends to decrease as data dimensionality and complexity increase. Joachims-Yu [5], Keerthi et al. [2], and Cot-

ter et al. [6] develop iterative algorithms for reduced SVMs that deliver impressive results, demanding only a very small amount of kernel execution to classify a single pattern whereas [6] reported a memory run out from [2], [5] applying on bulky dataset while their approach introduces a notable deviation from the original methodology with massive parameter choosing. Romdhani et al. [7] and Wu et al. [8] propose to approximate the decision function of a full SVM by identifying a novel denser set of patterns to replace the full set of support vectors whereas Raetsch et al. [9] explore wavelet-based estimations for these vectors to efficiently compute dot products between the kernel generating pattern vectors. These approaches, however, rely heavily on some complex optimization methods that are indeed prone to the starting point, learning rate, etc. Heisele et al. [10] suggests a structured SVM-oriented classification method that uses a series of linear SVMs in the early stages and a nonlinear SVM in the last stage while optimizing thresholds for enhancing both classification time and accuracy whereas Karim et al. [11] presents that a carefully organized cascade of two nonlinear SVMs considerably lowers classification costs compared to using a single nonlinear SVM. Arreola et al. [12] models a linear SVM-based decision tree. Sahbi-Geman [13] designs a tree-structured hierarchical SVM using threshold selection as well as reduced set techniques, applied across partitioned pattern spaces. Moreover, there are also some other noteworthy works focusing on developing cost-efficient SVM variants. Where, Maji et al. [14] presents that histogram and additive kernels based SVMs outperform linear SVMs in speed and accuracy. Ladicky and Torr [15] propose a locally linear SVM with bounded curvature and a smooth decision boundary, balancing the number of anchor points against classifier complexity to prevent overfitting and runtime issues. Xu et al. [16] suggests an additional training algorithm to further compress the trained SVMs using a small set of parameters. Li and Zhang [17] designs a fast object detector using cascaded stages with logistic regression acting as a weak base learner focusing on efficient training method. Raykar et al. [18] develops a cascaded system where the classifiers discard patterns based on probabilities calculated in previous stages achieving a balance between feature learning costs and accuracy. EESVM by Kundu-Karim [1], HESVM by Karim-kundu [19] achieve a significant advancement in SVM efficiency, which is particularly suitable for applications involving large-scale and complex datasets though their implementation complexity is also notable. Fu et al. [20] combines linear SVMs to classify complex data showing that the prediction phase efficiency of linear SVMs rivals that of nonlinear SVMs. Karim and Kundu [21] propose a second-order SVM by performing quadratic and linear programming in sequence, achieving compactness and high classification accuracy. Later, they introduce a different method [22] for support vector reduction by LP and computational analysis, demonstrating on benchmark datasets that their model achieves comparable accuracy to standard nonlinear SVMs with significantly improved prediction efficiency. Franc et al. [23] proposed Reduced SVMs (RSVMs) with lower computational costs compared to standard SVMs to classify patterns by leveraging a small subset from the support vectors of the conventional SVM while Gu et al. [24] introduced Clustered SVMs, employing weighted combinations of LSVMs that are trained on the subsets of the training data which are clustered to achieve localized separation. Karim et al. [25] introduces a novel similarity function to enhance the efficiency of Linear Programming based Support Vector Machine. Zhou [26] proposes an approach to enhance the efficiency of SVMs through sparsity constraints that offers significant computational speed while using a practically challenging assumption in the optimization method. Dries et al. [27] offers innovative approaches for support vectors reduction in SVM classifiers through two different methods that gives some computational benefits. However, their modification of training data limits their applicability to non-separable datasets. Wang et al. [28] offers a significant advancement in linear SVM algorithm by tackling fundamental issues with traditional hinge loss but initiates new challenges, specially in terms of optimization and scalability. Liu et al. [29] offers a significant advancement in kernel SVM methodology by incorporating the 0-1 soft margin loss into a nonlinear kernel framework that gives a promising alternative for complex classification tasks while the inherent challenges associated with their non-convex optimization and algorithmic complexity limits the model's practicality and robustness. Wu et al. [30] proposes an innovative approach to SVM classification by excluding the involvement of kernel functions and directly incorporating the 0-1 loss function that gives improved performance. However, on top of the challenges from their non-convex optimization and algorithmic complexity, their assumption that a quadratic hyper-surface is suitable for separating classes, which may not hold true for all datasets, potentially bounds its applicability. Akhtar et al. [31] gains generalization performance by coupling their RoBoSS loss function in SVM while issues about implementation complexity, parameter tuning, scalability, and data quality should be taken into account seriously during its application. Lin et al. [32] gets support vector reduction from modification of conventional SVMs by integrating the ℓ_0-norm into the hinge loss function but the challenges regarding their non-convex optimization, computational complexity should be considered when applying this approach. Wang et al. [33], [34], [35], [36] propose effective mechanisms for handling large-scale classification problems by balancing robustness, sparsity, and efficiency but their non-convex models bring challenges during implementation.

In spite of the efficiency of these classifiers, further sparsification with convenient implementation is very necessary for datasets that are complex and very large.

In this work, we focus to build up an exceptionally sparse classifier within a stochastic framework. By analyzing the SVM learning process through stochastic principles with pattern space spanning and leveraging computational and

probabilistic insights, we propose a simple but very productive learning algorithm under uncertainty. Being generated from the outcome of the analysis of a stochastic variable, this algorithm employs methods to model a highly efficient classifier from a very compact and dense set of stochastic decision points that significantly reduces classification costs while preserving statistical robustness and strong generalization capabilities.
The remaining part of the paper is arranged in the following way: Section 2 covers the relevant foundations while Section 3 details the proposed algorithm with an in-depth analysis. The experimental results are presented in Section 4 whereas Section 5 concludes with a discussion and suggestions for future work.

2 Related basics

2.1 Stochastic learning fundamentals

Stochastic learning lies in the Machine Learning system to train the machine where model parameters are determined under uncertainty using the training data (figure 1). With constant model complexity, for small training sets, the model may get insufficient information to capture underlying patterns, just memorizes the training dataset well rather than learning generalizable patterns, leading to high variance or overfitting and underperform while with larger sample size, the generalization error typically decreases because the model is trained on more representative data, improving its ability to generalize and reducing overfitting. Further, when the model has low dimensionality, it may not have the capacity to fit the training data adequately, resulting in high bias (underfitting) where the generalization error gets highly influenced by the inability to capture data complexity whereas when the model has high dimensionality, it can perfectly fit the training data, including noise, leading to high variance (overfitting) and the generalization error increases as the model fails to perform well on test data. Hansen [37], Karim [38] showed that to avoid higher generalization error in Stochastic Linear Learning, for a problem with sample size N, model dimension d, one must have $N >> d$ and this error goes unbounded for $N \leq d + 1$. However, the ideal balance is achieved when the model dimension matches the effective complexity of the data, given a sufficient sample size and the relationship among generalization error, sample size, and model dimension in stochastic learning is governed by the interplay among bias, variance, and the complexity of the model.

Generalization error while modeling a stochastic machine

Problem set up

The key objective in machine learning is to approximate an unknown function $f(x)$ that assigns an output y for an input

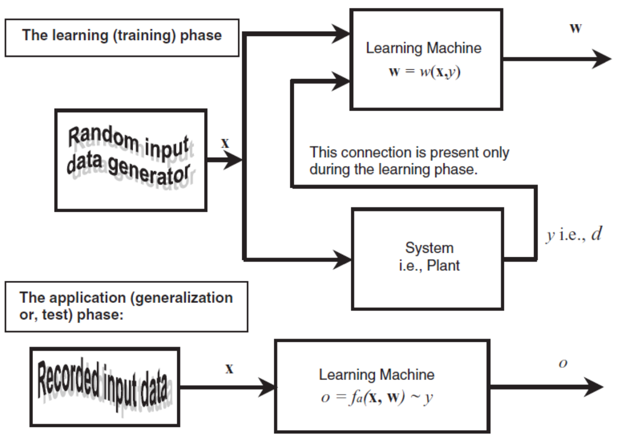

The learning (training) phase
Random input data generator
x
Learning Machine
w = w(x,y)
w
This connection is present only during the learning phase.
System i.e., Plant
y i.e., d
The application (generalization or, test) phase:
Recorded input data
x
Learning Machine
o = f_a(x, w) ~ y
o

Figure 1: Training and generalization phase of a statistical learning machine model [39]

x while the observed outputs y are noisy due to inherent randomness or measurement errors:

$$y = f(x) + \epsilon$$

where $f(x)$ is the true underlying function, ϵ is the random noise following a zero-mean normal distribution with variance $\sum_{\epsilon}^{2}$: $\epsilon \sim \mathcal{N}(0, \sum_{\epsilon}^{2})$.

We approximate $f(x)$ using a model $\hat{f}(x)$, which is learned from data, aiming at measuring how well $\hat{f}(x)$ performs by analyzing its expected squared error.

Expected error forming

The expected squared-error at a point x can be calculated as:

$$\mathbb{E}[(y - \hat{f}(x))^2]$$

where the expectation is taken over the randomness in both $\hat{f}(x)$ and y (due to noise ϵ).

Substituting $y = f(x) + \epsilon$ in the above equation we get

$$\mathbb{E}[(y - \hat{f}(x))^2] = \mathbb{E}[(f(x) + \epsilon - \hat{f}(x))^2]$$

Expanding the squared term we get

$$(f(x)+\epsilon-\hat{f}(x))^2 = (f(x)-\hat{f}(x))^2+2(f(x)-\hat{f}(x))\epsilon+\epsilon^2$$

Now taking the expectation and using $\mathbb{E}[\epsilon] = 0$, the middle term becomes $\mathbb{E}[2(f(x) - \hat{f}(x))\epsilon] = 0$ while the last term gives $\mathbb{E}[\epsilon^2] = \sum_{\epsilon}^{2}$, which is the noise variance and the whole expected squared error simplifies to

$$\mathbb{E}[(y - \hat{f}(x))^2] = \mathbb{E}[(f(x) - \hat{f}(x))^2] + \sum_{\epsilon}^{2}$$

Error term decomposition

The term $\mathbb{E}[(f(x) - \hat{f}(x))^2]$ representing the error in approximating $f(x)$ with $\hat{f}(x)$, can be further decomposed by adding and subtracting $\mathbb{E}[\hat{f}(x)]$ (the expected prediction of the model over all possible datasets) as

$$\mathbb{E}[(f(x)-\hat{f}(x))^2] = \mathbb{E}[(f(x)-\mathbb{E}[\hat{f}(x)]+\mathbb{E}[\hat{f}(x)]-\hat{f}(x))^2]$$

Now, expanding the squared term, we get

$$(f(x)-\mathbb{E}[\hat{f}(x)]+\mathbb{E}[\hat{f}(x)]-\hat{f}(x))^2 = (f(x)-\mathbb{E}[\hat{f}(x)])^2 + 2(f(x)-\mathbb{E}[\hat{f}(x)])(\mathbb{E}[\hat{f}(x)]-\hat{f}(x)) + (\mathbb{E}[\hat{f}(x)]-\hat{f}(x))^2$$

Now applying the expectation, we get the cross-term $\mathbb{E}[2(f(x) - \mathbb{E}[\hat{f}(x)])(\mathbb{E}[\hat{f}(x)] - \hat{f}(x))] = 0$ as $\mathbb{E}[\hat{f}(x)] - \hat{f}(x) = 0$. whereas the first term $(f(x) - \mathbb{E}[\hat{f}(x)])^2$ is the **Bias squared**, the second term $\mathbb{E}[(\mathbb{E}[\hat{f}(x)] - \hat{f}(x))^2]$ is the **Variance**.

Thus, we can write

$$\mathbb{E}[(f(x) - \hat{f}(x))^2] = (\text{Bias})^2 + (\text{Variance}),$$

where, $\text{Bias}^2 = (f(x) - \mathbb{E}[\hat{f}(x)])^2$, $\text{Variance} = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$.

Eventually, the full expression for the expected squared error becomes

$$\mathbb{E}[(y - \hat{f}(x))^2] = (\text{Bias})^2 + (\text{Variance}) + \sum_{\epsilon}^{2}$$

Generalization introducing input distribution

Considering all inputs x, we have to take the expectation over the input distribution $p(x)$ as

$$\text{Generalization Error} = \mathbb{E}_{x,\epsilon}[(y - \hat{f}(x))^2]$$

Expanding this we can write the the final expression as

$$\mathbb{E}_{x,\epsilon}[(y - \hat{f}(x))^2] = \underbrace{\mathbb{E}_x[(f(x) - \mathbb{E}[\hat{f}(x)])^2]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_x[\mathbb{E}_\epsilon[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]]}_{\text{Variance}} + \underbrace{\sum_{\epsilon}^{2}}_{\text{Irreducible Error}} \quad (1)$$

Intuition behind each term & key observations

- High bias takes place while the model is undertrained or too simple to capture data complexity and as model complexity increases, **Bias** decreases because the model can better approximate $f(x)$. Underfitting from high bias can lead to a situation, where both training and test errors are high.
- **Variance** increases because the model turns to be more responsive to the training data. Variance measures how sensitive the model is to changes in the training data. High variance can result in overfitting, where the model performs well on the training data but poorly on unseen test data.
- **Irreducible error** ($\sum^2$) remains constant, independent of model complexity.
- A model having high bias typically has low variance and vice versa. The optimal model is a properly tuned model that achieves good generalization with reasonable training performance minimizing the total error by balancing bias and variance. This is the **bias-variance tradeoff**, a fundamental concept in machine learning that explains the relationship between a model's ability to generalize to unseen data and its performance on training data. It is very important to consider during the construction of a stochastic machine.

Next we discuss about Support Vector Machine (SVM), which is very prominent managing this bias-variance trade off successfully.

2.2 Support vector machine (SVM)

Due to their robustness, strong generalization capability, and convexity of the mathematical model to find unique global optimum, SVMs probably have gained the highest popularity among machine learning approaches for supervised learning though having very simple principle. The problem that drove its initial development occurs in various forms as the bias variance trade-off [40], capacity control [41] , overfitting [42] while the core idea is the same [43]. Thus, it is mathematically quite strong basing on the concept of statistical learning theory as well as optimization.

2.2.1 SVM with statistical learning theory

Suppose that we have two classes of objects. Now we have received a novel object for which we are to allocate a class from one of these two classes. Such a problem can be considered as a *(binary, due to two classes) pattern recognition* problem that can be formed in the following way: having a set of observational data

$(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{S} \times \{\pm 1\}$,

we need to design a *decision (discriminator) function* $f : \mathcal{S} \to \{\pm 1\}$, where $\mathcal{S}$ is a set of elements from which the x_i patterns have been drawn, generally referred to as the *domain* whereas y_i are designated as the *targets* or *class-labels*. A well defined decision function is supposed to be able to *generalize* to the unseen novel data instances that poses a possibly low value while measuring the *risk*

$$R[f] = \frac{1}{2} \int |f(x) - y| \, dP(x, y) \quad (2)$$

where $dP(x, y)$ can be written as $p(x, y)dxdy$ when a density $p(x, y)$ exists.
Expressing linguistically, on average across an unspecified distribution P that is considered to generate both of the training and test data, we aim to achieve a low error. Here, we calculate the error using the *binary loss function* $q(x, y, f(x)) := \frac{1}{2}|f(x) - y|$, which gives 1 when (x, y) is not correctly classified and 0 when it is correctly classified. It is to note that up to now, the patterns can take almost any form , and no assumptions have been made on $\mathcal{S}$ except it is a set equipped by a probability measure P. Additionally, without having an estimate of $P(x, y)$, although (2) represents the true mean error in a nice way, it does not tell about the way either to find a function posing a small risk or even to assess the risk of a specified function, and thus becomes not so useful. Hence,for risk minimization, we need to move towards an induction principle . Now, we incorporate an additional form of structure related to the information we actually have, that is the training data. And while the quantity $R[f]$ is usually referred to as the expected risk, or simply the risk or here, we term it as the actual risk, this newly introduced quantity will be called the empirical risk (the training error) , noted by $R_{\text{emp}}[f]$, which is defined as the calculated mean error rate over the full training set (considering a fixed and finite number of observations); that is,

$$R_{\text{emp}}[f] = \frac{1}{m}\sum_{i=1}^{m}|f(x_i) - y_i| \tag{3}$$

It is to be noted that, here, no probability distribution shows up and $R_{\text{emp}}[f]$ is a fixed number for a specific f and for a specific training data set $\{x_i, y_i\}$.
However, when the number of training examples m is limited, minimizing this empirical risk does not assure small actual risk as small error over the training set does not automatically imply good generalization ability, that is, a small amount of error on any independent and novel test data. This phenomenon is commonly known as overfitting [44]. To get the maximum benefit from a limited amount of data, various statistical methods are being developed over many years. Vapnik-Chervonenkis, through their VC theory [45] in Statistical Learning Theory, emphasis on the necessity of limiting the set of functions from which f is selected to one having a *capacity* that well fits the available training data. It establishes *bounds* on the test error, which depends on the empirical risk as well as the capacity of the function class while minimizing these bounds guides to the *structural risk minimization* principle [46]. Considering the discussed problem, SVMs can be regarded as an application of this principle where it is tried to minimize the "Structural Risk", which is a combination of the *empirical risk*, $R_{\text{emp}}[f]$ and a "capacity term" found for the set of hyperplanes from a dot product space $\mathcal{H}$ as stated next. For some η such that $0 \le \eta \le 1$ and $m > h$, with probability $1 - \eta$, the following bound holds :

$$R[f] \le R_{\text{emp}}[f] + \sqrt{\frac{h\left(\log(2m/h) + 1\right) - \log(\eta/4)}{m}} \tag{4}$$

where h, VC dimension (or "model complexity"), which is an integer that cannot be negative and is used to assess the above mentioned capacity. Right-hand side of (4) is called the the "risk bound" and the second term of this part is called the "model complexity penalty" or "VC confidence". Several key points can be noted about this bound. First, it is free of the distribution $P(x, y)$ and here only assumption is made that both the training and test data are taken independently following this distribution. Second, the left hand side of the bound is usually not possible to compute whereas the right hand side of it can be computed if we know h. Third, learning process is said to be consistent if both of the expected risk and the empirical risk converge to the minimum value of the risk, which can be found for $m \to \infty$ and at last, for given some diverse learning machines (equivalently, family of functions), selecting the one, which minimizes the right hand part of (4) will be taking the machine that gives the least upper bound imposed on the actual risk in its left hand part.
Moreover, going into a bit more explicit about the error or risk of the machine depending on its model complexity we see that the VC dimension, h of a set of functions describing their *capacity* controls the main characteristic of a machine. SVM, by applying structural risk minimization (SRM) [47] principle, which is an inductive method that chooses a model to learn from a training dataset with finite examples, acting as an indicator of this capacity control by proposing a trade-off between the empirical error and this VC dimensions, h, equivalently, the hypothesis of space complexity. Having a form of convex optimization, SRM applies a set of sequential models with an ascending order of complexity. Figure 2 briefly displays how the total model error changes depending on the model-complexity-index of a machine that can be analyzed from the relation (4). For low-complex models (similar to high bias but low variance in relation (1) during a stochastic machine learning), the error gets higher as simpler models are unable to learn all of the data-complexity leading to an underfitting situation. On contrary, for higher model indices (similar to high variance but low bias in relation (1) during the learning of a stochastic machine), the structure tries to adapt its learning model with higher concentration to the very specific training data leading to an overfitting condition, which reduces the amount of training error with increasing the model complexity but at the cost of a worsening performance on the test data. Thus the optimal model index, h^* lies between these higher and lower values of this model index where the model gets down to its lowest value.

2.2.2 Kernels with SVM

In the very simple pattern recognition problems, SVM utilizes a linear separating hyperplane to form a classifier with the largest margin. To achieve this, the learning problem for SVM will be formulated as a constrained nonlinear programming problem where the objective function are quadratic and the constraints are linear, that means, it will

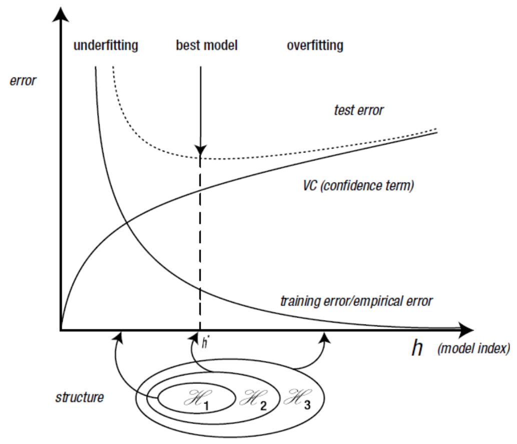


Figure 2: Error trends on model complexity [48]

be needed to solve a QP(quadratic programming) problem. On the other hand, when a dataset cannot be separated linearly in the original input space, the soft-margin SVM struggles to manage a reliable separating hyperplane that both minimizes misclassified data points and generalizes effectively. But following Cover's theorem [49], data points get more probable to be in the form for being linearly separable after they are nonlinearly mapped into a higher-dimensional space. In this regard, SVM uses Kernel method, which was first introduced in [50] [51] and implicitly performs a nonlinear transformation of the input training data from its existing space to a much higher-dimensional feature space (or kernel space) by replacing the inner product by an optimum positive definite function (or kernel function, strongly related to Mercer's Theorem [52]), where data will be linearly separable (figure 3 on page 166) rather than solving a high-order separating hypersurface in the input space and this is accomplished before solving the learning problem through a convex optimization. This approach is preferred because a nonlinear separating boundary in the input space escalates computational demands during the optimization phase. Nonlinear kernel functions lessen the curse of dimensionality [53] productively. However, selecting and setting the right kernel function are vital for SVM optimality.

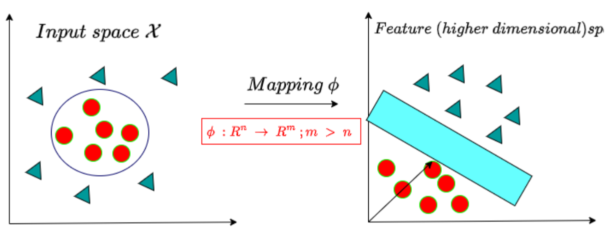

Input space $\mathcal{X}$
Mapping ϕ
$\phi : R^n \rightarrow R^m ; m > n$
Feature (higher dimensional) sp

Figure 3: Non-linear transformation of data through kernel methods: mapping input space to a much higher dimensional feature space, realizing linear separation of data-classes in the feature space, which is not linear in input space.

Mercer's theorem for kernels

Mercer's Theorem is a fundamental concept from Functional Analysis to provide the theoretical basis for using kernels in SVM, which can be described in the following way. Let $K(x, y)$ be a continuous function defined on a compact set $X \times X$. Mercer's Theorem states that if $K(x, y)$ is *positive semi-definite* (i.e., for all d-dimensional vectors **v**, the matrix $[K(x_i, x_j)]$ is positive semi-definite), then there exists a *feature space* and a *mapping* $\phi(x)$ such that:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Explaining in another way, the function $K(x, y)$ can be expressed as the inner product of two vectors in some high-dimensional (even possibly, infinite-dimensional) feature space (equivalently, RKHS) where some key concepts are described below:

Key concepts

– **Positive Semi-Definiteness**: A function $K(x, y)$ is positive semi-definite if, for any finite set of points $\{x_1, x_2, \ldots, x_d\}$, the matrix formed by evaluating $K(x_i, x_j)$ for all pairs (x_i, x_j) is positive semi-definite. This means that all eigenvalues of the matrix are non-negative.

– **Reproducing Kernel Hilbert Space (RKHS)** [54–57]: The feature space $\mathcal{H}$ associated with the kernel function $K(x, y)$ is called the RKHS. The kernel K defines an inner product in this space, and the function $\phi(x)$ maps the original input space to this RKHS.

– **Significance for Machine Learning**: Mercer's Theorem justifies the use of kernel functions like the *Gaussian (RBF) kernel*, *polynomial kernel*, and others in machine learning. The theorem ensures that these kernels correspond to valid inner products in high-dimensional feature spaces, enabling the use of linear methods in these spaces.

Most commonly used kernels are: **Polynomial kernel:** $K(x, y) = (x^T y + c)^d$ & **Gaussian (RBF) kernel:** $K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sum^2}\right)$

For further kernels:

It is possible to construct more complex kernel functions by combining simpler kernels.

– *Non-negative Linear Combination of Mercer kernels also produces a Mercer kernel :* If $K_1(x, y), K_2(x, y), \ldots, K_n(x, y)$ are Mercer kernels and $\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0$, then $K(x, y) = \sum_{i=1}^{n} \alpha_i K_i(x, y)$ is also a Mercer kernel.

– *Product of Mercer kernels result in a Mercer kernel :* If $K_1(x, y)$ and $K_2(x, y)$ are Mercer kernels, then $K(x, y) = K_1(x, y) \cdot K_2(x, y)$ is also a Mercer kernel.

2.2.3 SVM With mathematical programming

Being mathematically solid and hence powerful, SVM has become one of the most popular supervised machine learning method for classification where the problem is framed as estimating a decision boundary under uncertainty. Due to its high power and minimal classification error despite sparsity, it has gained high popularity. Initially, it was introduced through Quadratic Programming (QP), known as QPSVM. However, as the L_1-norm is more likely to produce sparser solutions than L_2-norm [58], Vapnik modelled an alternative SVM model using Linear Programming (LP) to design the objective function, called VLPSVM. This part addresses SVM and its two core forms (QPSVM and VLPSVM), with further particulars provided in [59].

Quadratic programming (QP) based SVM

This is the standard SVM, a leading algorithm in the field of machine learning. QPSVM uses structural risk minimization concept from statistical learning theory implementing margin maximization between two classes that aims to find the data depending separating hyperplane, which is as far as possible from the nearest data point. Let us consider a binary classification problem using a set of given training patterns $\{(x_i, y_i)\}_{i=1}^{N}$; $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. To divide these patterns into two classes effectively, SVM determines a classifier based on a decision function having the form $f(x) = w \cdot \phi(x) + b$, leading to $class(x) = sgn(f(x))$, where $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ is a kernel operation. The primal domain formulation of this QPSVM is as follows:

$$\min_{w,b,\zeta} f_P(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\zeta_i \tag{5}$$

$$s.t.\ \ y_i\big(w \cdot \phi(x_i) + b\big) \ \geq\ 1 - \zeta_i \tag{6}$$

$$\zeta_i \ \geq\ 0;\ i = 1, 2, ..., N \tag{7}$$

where the auxiliary variables $\zeta_i > 0$ work mainly for patterns that remain outside their respective class-margins (or we call it here, ”margin-outward-deviated patterns”) with $C > 0$ is a data-influenced parameter to regularize the classifier following bias-variance tradeoff concept in stochastic learning. The corresponding dual problem becomes:

$$\max_{\alpha} f_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) \tag{8}$$

$$s.t\ \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{9}$$

$$0 \leq \alpha_i \leq C;\ \ i = 1, 2, ..., N \tag{10}$$

which is also a QP problem and the patterns for which $\alpha_i > 0$ are the support vectors (SVs). They can also be called as the stochastic decision points as they act as the bases of the final decision function through their coefficients α, which are used to determine the bias term, b and the weight variable w as $w = \sum \alpha_i y_i \phi(x_i)$. An illustrative depiction of this margin based SVM is provided in figure 4 on page 167.

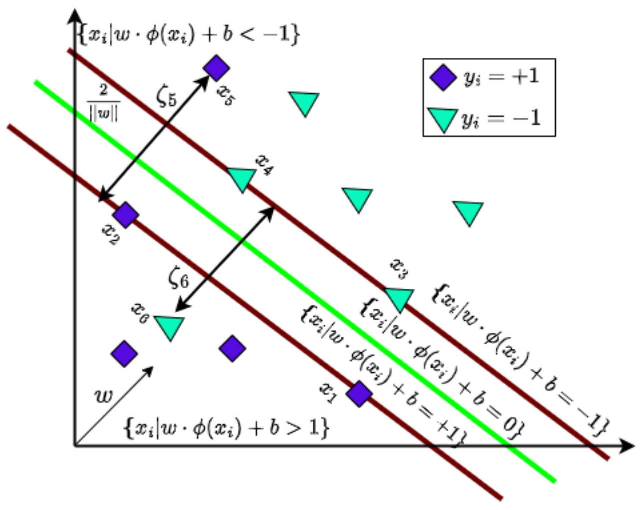

$\{x_i|w \cdot \phi(x_i) + b < -1\}$
$\frac{2}{\|w\|}$
ζ_5
x_5
$y_i = +1$
$y_i = -1$
x_4
x_2
ζ_6
x_6
x_3
$\{x_i|w \cdot \phi(x_i) + b = -1\}$
$\{x_i|w \cdot \phi(x_i) + b = 0\}$
$\{x_i|w \cdot \phi(x_i) + b = +1\}$
x_1
w
$\{x_i|w \cdot \phi(x_i) + b > 1\}$

Figure 4: A visual representation of the standard SVM is shown, where patterns for the positive and negative classes are depicted by squares and triangles, respectively. The green straight line represents the optimal separating hyperplane, and the brown lines mark the margins for the classes. Patterns that lie outside their respective classes are referred to as outward-deviated patterns, where the depth of this deviation is represented by a non-negative variable, ζ. The goal of this QP-based SVM is to maximize the margin while minimizing the training error as much as possible. The green line is also known as the decision boundary from the training output.

Linear programming (LP) based SVM

Vapnik proposed an LP model to find a separating hyperplane, which we name here VLPSVM and effectively incorporates an indirect L_1 norm while modeling the cost function, promoting even more sparsity while maintaining accuracy comparable to QPSVM that uses the L_2 norm. As a result, VLPSVM shares similarities with QPSVM but demands less computational effort for classifying patterns. If w represents the weight vector of the hyperplane for VLPSVM, its decision function is given by: $f(x) = w \cdot \phi(x) + b$ leading to $class(x) = sgn(f(x))$. The primal domain formulation of this VLPSVM model is as follows:

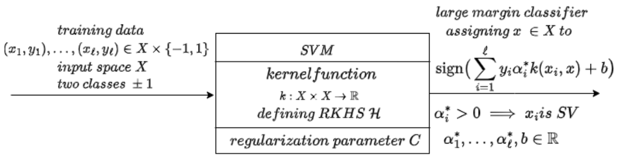

training data
$(x_1, y_1), \ldots, (x_\ell, y_\ell) \in X \times \{-1, 1\}$
input space X
two classes ± 1
SVM
kernel function
$k : X \times X \to \mathbb{R}$
defining RKHS $\mathcal{H}$
regularization parameter C
large margin classifier
assigning $x \in X$ to
$\mathrm{sign}\left(\sum_{i=1}^{\ell} y_i \alpha_i^* k(x_i, x) + b\right)$
$\alpha_i^* > 0 \implies x_i$ is SV
$\alpha_1^*, \ldots, \alpha_\ell^*, b \in \mathbb{R}$

Figure 5: Classification with a kernel based SVM. The learning method is completely expressed by the used kernel function k along with the penalty parameter C. Having training data, the machine produces the coefficients α for the decision function [61].

$$\min_{\beta,\zeta,b} \sum_{i=1}^{N} \beta_i + C \sum_{i=1}^{N} \xi_i \tag{11}$$

$$s.t.\ y_i\Big(\sum_{j=1}^{N} \beta_j y_j \phi(x_j) \cdot \phi(x_i) + b\Big) \geq 1 - \xi_i \tag{12}$$

$$\beta_j \geq 0;\ j = 1, 2, ..., N \tag{13}$$

$$\xi_i \geq 0;\ i = 1, 2, ..., N \tag{14}$$

In this case, the auxiliary variables $\xi_i > 0$ are assigned to handle with the unity-outward-deviated patterns (i.e., the training patterns that give $ClassLabel(x_i) \cdot f(x_i)$ <1) and these ξ are also optimization variables for the problem (11)-(14) that are found with the other variable βs, b depending on the data-dependent stochastic parameters, kernel parameter and penalty parameter. To avoid confusion and for sharp clarification, here, training examples (x_j) of this VLPSVM, for which, coefficients $\beta_j > 0$ are labeled as Expansion Vectors (EVs), which are quite similar to the support vectors (SVs) in QPSVM while being different from both topological and physical points. These optimum value of βs are used to determine $w = \sum \beta_j y_j \phi(x_j)$. The number of EVs found from the VLPSVM algorithm are usually very much smaller compared to the total number of training patterns proving further sparsity of VLPSVM model. Our experiments on benchmark data as well as other works [60] show that VLPSVM has nearly the same performance as QPSVM despite of being more efficient due to less kernel execution in classification stage showing VLPSVM to be more productive for many cases, specially, when classification cost, speed, resources, etc are under constraints. Further, this LPSVM simplifies the optimization problem by using linear programming, making it computationally efficient for certain stochastic processes and for this model, the kernel, $K(.,.)$ does not need to satisfy Mercer's conditions [59]. A graphical representation of this (kernel based) SVM is given in figure 5 on page 168.

3 Proposed method with stochastic functional ξ- analysis

3.1 Research gap, motivation & our direction

Despite of being the most mathematically sound machine learning algorithm by using the famous statistical learning theory, maximum margin concept, and mathematical programming, while working on probabilistic data, implementing total risk minimization principle, SVM does not directly control the model complexity but it automatically determines it by selecting the type and number of stochastic decision points (support vectors (SVs)) depending on the data-dependent parameters. These SVs use kernel operations that highly boosts the expressive capability of the decision function while preserving the underlying linearity that ensures the learning to remain tractable. However, on top of the mounted cost by excessive kernel computations from a large number of SVs, this increased flexibility raises the risk of overfitting since the selection of separating hyperplane turns to be increasingly ill-posed because of the number of degrees of freedom [62]. This becomes more serious in case of large and complex data with noise. Thus managing the machine from getting excessive number of SVs is necessary regarding both cost and accuracy.
Moreover, as the standard SVM algorithm concentrates on error performance indirectly through cost-function and constraints using data-dependent parameters, these parameters play a very crucial role in the algorithm, specially, finding the best trade-off between model-complexity and generalization. However, as there is no exact and analytic expression for these stochastic parameters, we have to find their values using cross-validation method that does not give the exact best and unique values of theses parameters. On top of this, it is also followed that to improve a very little amount of classification accuracy on complex dataset, SVM generates a relatively larger number of support vectors (represented in figure 6). Additionally, because of the continuous raise in the heavy size and complexity of data in parallel with modern technological development, further speed up of the classification process is being demanded continuously, which stipulates one to work on reducing the number of SVs as much as possible.
Thus, while the existing methods for SVM cost reduction are either 1) complex method that is very expensive and not so convenient to implement or 2) bear uncertainty to get as being non-convex or 3) cannot gain sparsity and accuracy simultaneously upto that level, a mathematical and computational analysis based suitable operation on the first-step solution from SVM algorithm may also be productive and useful. One of the simplest but heavily productive approach in this direction could be to reduce the support vectors by discarding the least significant SVs, misleading SVs, or such kernel computing vectors by sacrificing an insignificant amount of classification accuracy (even in case it becomes demanded, the worst scenario). But while discarding

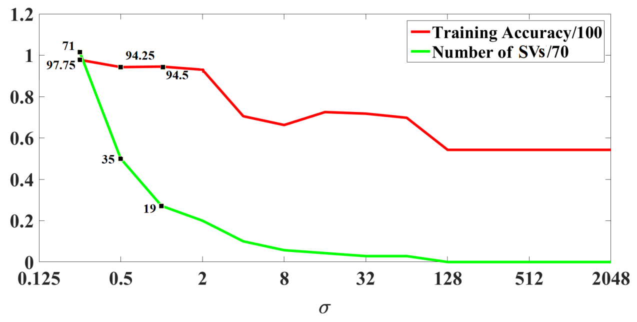


Figure 6: The number of SVs fluctuating with training accuracy. The plots are normalized by dividing them by their respective maximum values, bringing within 0 and 1.

SVs without considering their contribution can create negative impact on decision-making, in practice, training data may contain noise or outliers that stay far from their main cluster and as the SVM decision boundary relies on SVs, the presence of outliers in the SVs set can make the decision function loaded with more misleading info and shape the boundary less smooth, or may even cause overfitting, especially with complex datasets. Hence, a method to reduce the number of support vectors (as well as the machine-misleading noise due to the noise in the data) after analyzing the outlier's right-leading or misleading on the decision function is rewarding with respect to both cost and accuracy. In this direction, we start with the basis of Statistical Learning Theory where we try to further reduce the Right Hand Side of (4) by reducing both of its 1st term and 2nd term simultaneously that eventually lowers the upper bound on the actual risk, leading towards a machine having better generalization performance than before while also reducing the classification cost.
This is achieved by using some mathematical and computational analysis with the stochastic parameter based values of the discriminator function, auxiliary variable, and coefficient of the stochastic decision points found from the first step SVM solution to select the insignificant SVs, misleading and noisy SVs and discard from them as many as possible while preserving the machine's efficiency and performance.
Experimental results demonstrate that our proposed method significantly reduces classification costs while posing classification accuracy analogous to the standard SVM, particularly with large and complex training data that contains many outliers.

3.2 Deterministic mapping of data

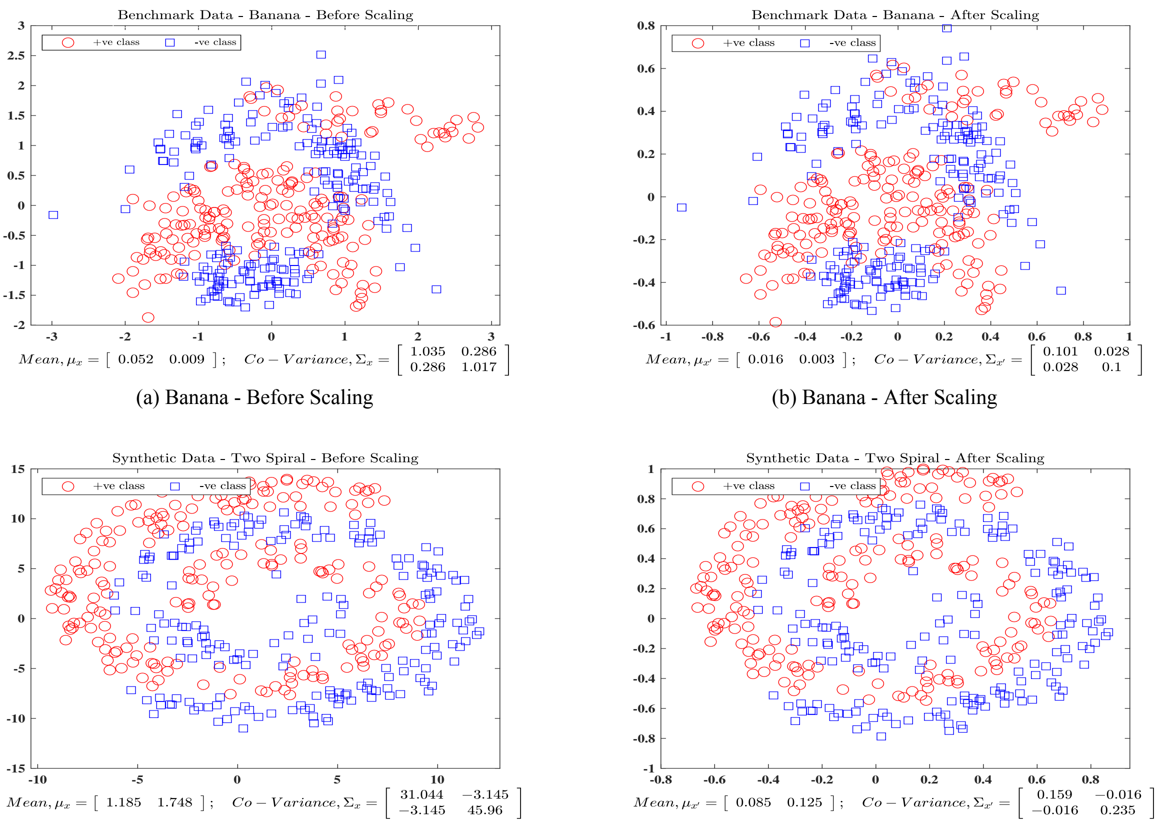
As the numerical values of raw data varies widely, scaling it may help to avoid numerical instability by dropping data redundancy and irregularity that is useful for the objective function to work properly and to improve the convergence of the optimization algorithm. This also gets further motivated from the reporting about data preprocessing for higher accuracy by Alaa et al. [63] and about data scaling to improve the condition number of the matrix in the optimization problem [64]. However, while doing this, we must try for such a scaling that may not hamper the crucial properties of data. For example, we can try for a careful scaling of data that may allow a bit change in its topological spanning without noticeably losing its main stochastic property. But we also know that the mean of a scaled dataset is scaled by the scaling factor in each dimension and the elements of the co-variance matrix of the scaled dataset is scaled by the square of the scaling factor (proof is given in the later part of this section). Hence, we decide first to pick the maximum absolute of all the feature values and then divide each entry by this, which maps each entry of the column to lie in [-1,1]. This kind of deterministic transformation would help to evade data leakage in the stage of model testing but may turn out to be sensitive to the outliers that are very common in the real-world problem. Hence, by scaling we bring feature values ≤ 1 and then throw the machine vectors for which corresponding stochastic auxiliary variables ξ give values ≥ 1 by following some numerical analysis based on the stochastic functional analysis described next. A pictorial view of before and after scaling of a benchmark and a synthetic data set is given in figure 7 on page 170.

Statistical impact on mean and variance by scaling a multidimensional dataset

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of n data points, where each data point $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id})$ is a vector in $\mathbb{R}^d$ (a d-dimensional space). If We scale this dataset by multiplying with a constant factor $\lambda > 0$, then each of the data points will be multiplied by λ and the new dataset becomes: $\mathbf{x}'_i = \lambda \cdot \mathbf{x}_i = (\lambda x_{i1}, \lambda x_{i2}, \ldots, \lambda x_{id})$. So, the scaled dataset $\mathbf{X}'$ contains the scaled vectors $\mathbf{x}'_i \in \mathbb{R}^d$ Now, we know that, in case of the original (unscaled) data, the mean vector of $\mathbf{X}$ be $\boldsymbol{\mu}_\mathbf{X} = \mathbb{E}[\mathbf{X}]$ and the covariance matrix of $\mathbf{X}$ be:

$$\sum\nolimits_\mathbf{X} = \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X})(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X})^\top]$$

Then in case of the scaled data, the mean vector of $\mathbf{X}'$ be $\boldsymbol{\mu}_{\mathbf{X}'} = \mathbb{E}[\mathbf{X}'] = \mathbb{E}[\lambda\mathbf{X}] = \lambda\mathbb{E}[\mathbf{X}] = \lambda\boldsymbol{\mu}_\mathbf{X}$ and the covariance matrix of $\mathbf{X}'$ be: $\sum_{\mathbf{X}'} = \mathbb{E}[(\mathbf{X}' - \boldsymbol{\mu}_{\mathbf{X}'})(\mathbf{X}' - \boldsymbol{\mu}_{\mathbf{X}'})^\top] = \mathbb{E}[(\lambda\mathbf{X} - \lambda\boldsymbol{\mu}_\mathbf{X})(\lambda\mathbf{X} - \lambda\boldsymbol{\mu}_\mathbf{X})^\top] = \lambda^2\mathbb{E}[(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X})(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X})^\top] = \lambda^2\sum_\mathbf{X}$. That is, the covariance matrix of the scaled random vector is scaled by λ^2. But the *variance* of a single dimension of a dataset is simply the diagonal element of the covariance matrix. Then in case of the original dataset $\mathbf{X}$, Var$(x_j) = \sum_{X,jj}$ and for the scaled dataset $\mathbf{X}'$, Var$(x'_j) = \sum_{X',jj} = \lambda^2 \cdot \sum_{X,jj}$ Thus, the variance in each dimension is scaled by a factor of λ^2. We scale the dataset by dividing it by the maximum value of the elements of $\mathbf{X}$, that is in our case, if $\rho = max(max(abs(\mathbf{X}))) \geq 1$ then $\lambda = 1/\rho$ and the new dataset $\mathbf{x}'_i = \lambda \cdot \mathbf{x}_i = \frac{1}{\rho} \cdot \mathbf{x}_i = \frac{1}{max(max(abs(\mathbf{X})))} \cdot \mathbf{x}_i \leq \mathbf{x}_i$ with $\mu_{X'} = \lambda \cdot \mu_X = \frac{1}{\rho} \cdot \mu_X = \frac{1}{max(max(abs(\mathbf{X})))} \cdot \mu_X \leq \mu_X$ and $\sum_{X'} = \lambda^2 \cdot \sum_X = \frac{1}{\rho^2} \cdot \sum_X = \frac{1}{[max(max(abs(\mathbf{X})))]^2} \cdot \sum_X \leq$



(a) Banana - Before Scaling

(b) Banana - After Scaling

(c) Two Spiral - Before Scaling

(d) Two Spiral - After Scaling

Figure 7: Visualization of the effect on 2D data by our way of scaling. It can be clearly observed that after data scaling, in case of Benchmark Banana dataset [65] the mean is scaled by $\frac{1}{3.25}$ and Co-variance is scaled by $(\frac{1}{3.25})^2$ whereas for the Synthetic Two-spiral dataset [66] the mean is scaled by $\frac{1}{13.98}$ and Co-variance is scaled by $(\frac{1}{13.98})^2$. Note that the topological spanning of the datasets are reduced by contracting its spatial scattering but the stochastic property of the datasets are not harmed noticeably.

$\sum_X$.

3.3 Analyzing stochastic function

In case of learning from probabilistic data, we need to model a data dependent stochastic parameter based discriminator function with the capacity to master the training data well to build optimum generalization ability that can classify novel patterns with least error. For this, it is needed to model the problem set using error-related constraints. Thus in case of both LPSVM and QPSVM, while the decision function are expressed as $f(x) = w.\phi(x) + b$, their error-associated constraints from (6) & (12) can be represented by the relation, $y_i f(x_i) \geq 1 - \xi_i \Rightarrow \xi_i \geq 1 - y_i f_(x_i)$.
Now, considering any wrongly classified training example x_i, if $y_i f(x_i) = -\delta_i$ with $\delta_i > 0$, then $\xi_i \geq 1 + \delta_i$. Thus, for M wrongly classified training examples, $\sum_{i=1}^{M} \xi_i \geq M + \sum_{i=1}^{M} \delta_i$. Hence, suppressing $\sum_i \xi_i$ through regularizer will curb $M + \sum_{i=1}^{M} \delta_i$, not only just M, which is the total training error. It is possible that suppressing $M + \sum_{i=1}^{M} \delta_i$ could take place during the following three conditions: i) Minimizing M faster than $\sum_{i=1}^{M} \delta_i$, which eventually decrease the training error rate, ii) Minimizing M comparatively slower than $\sum_{i=1}^{M} \delta_i$, which may cause unexpected phenomena on training performance, iii) Minimizing M and $\sum_{i=1}^{M} \delta_i$ with the same rate and preference, which may not lower the training error rate expectedly. Thus, throwing P ($P \leq M \leq N$) outlying patterns apparently decreases P training errors while also lowers P kernel executions in classification stage when these example patterns are also elements of the final classifier. Nevertheless, how this outlier throwing impacts the classification of other patterns is an important issue to address. We examine it through a careful analysis of the contribution as well as confusion caused by them in the accurate classification of other patterns, which we refer to as ξ-analysis, and it is explained in the following sections.
Further, the discriminators from both VLPSVM and QPSVM can be rewritten using the following common expression by introducing a shared term, ”Kernel Computing Vector (KCV)”, for Support Vector(SV), Expansion Vector(EV), or such other kernel operating vectors: $f(p) = \sum_{l \in \text{KCV-set}} \beta_l y_l K(x_l, p) + b$, for p representing any arbi-

trary pattern with $K(x_l, p) = \phi(x_l) \cdot \phi(p)$ is a kernel function (for example, in case of RBF: $\phi(x_l) \cdot \phi(p) = e^{-\frac{(x_l-p)^2}{2\Sigma^2}}$), which is the leading contributor to computational expenses with all $\beta_l > 0$ involved in the aggregation of time during the classification stage. Following this, we aim to identify a subset from this kernel operating x_l patterns that have $\beta_l > 0$ and discard those patterns that do not contribute meaningfully to decision-making because of being outliers. This is the main focus of our *stochastic ξ-analysis*, which is discussed in the next section.

3.4 Numerical analysis of stochastic auxiliary variable, ξ

A model should ideally take the data's inherent signal and exclude any noise or erratic patterns, which is not a simple task as data is often far from perfect like containing outliers, irrelevant variables, etc, which are related to the stochastic property of data. This is efficiently handled by introducing the auxiliary variable, ξ in the mathematical model. In case of LPSVM, its value comes from the solution of an optimization problem using probabilistic data dependent stochastic parameters. This variable quantifies the degree of violation of the margin constraint for the patterns, which can be interpreted as a measure of uncertainty or noise in the data and following its values, patterns' class-position, patterns' role as well as actual significance in the final discriminator can be analyzed.

3.4.1 Patterns' class-position following ξ

As we have described before that an "outlier" pattern is the one that lies beyond the limits of its class boundary. Thus, a pattern is considered "inlier" if it stays within its own class boundary. Hence, for any arbitrary training pattern x_i having the class-label y_i with the final decision function-value $f(x_i)$ is considered as an inlier when $y_i = sgn\big(f(x_i)\big) \Rightarrow 1 = y_i sgn\big(f(x_i)\big) \Rightarrow y_i f(x_i) > 0$ and is considered as an outlier if $y_i = -sgn\big(f(x_i)\big) \Rightarrow -1 = y_i sgn\big(f(x_i)\big) \Rightarrow y_i f(x_i) < 0$. Expressing the SVM-error-constraints in (6) or in (12) in a common single form utilizing the stochastic auxiliary variable ξ as $y_i f(x_i) \geq 1 - \xi_i \Rightarrow \xi_i \geq 1 - y_i f(x_i)$, we can write for outlier $\xi_i > 1$ while in case of inlier $0 \leq \xi < 1$ with the next two scenarios: i) inlier with $y_i f(x_i) < 1 \Rightarrow 1 - y_i f(x_i) > 0 \Rightarrow \xi_i > 0$ ii) inlier with $y_i f(x_i) > 1 \Rightarrow 1 - y_i f(x_i) < 0 \Rightarrow \xi_i = 0$ as ξ_i cannot be negative.
From the above discussed points, it is evident that patterns with $\xi > 1$ belong to different classes rather than their original classes. Now, we examine their significance by evaluating the roles they play in the discriminating function in case of both LPSVM and QPSVM.

3.4.2 Roles of the $\xi > 1$ patterns in the discriminating function

Role while working with QPSVM: QPSVM has a KKT condition as $\alpha_i\Big(y_i\big(w \cdot \phi(x_i) + b\big) - 1 + \zeta_i\Big) = 0$. Using $\alpha_i \neq 0$ & $\alpha_i\Big(y_i\big(w \cdot \phi(x_i) + b\big) - 1 + \xi_i\Big) = 0$ leads to $y_i\big(w \cdot \phi(x_i) + b\big) - 1 + \xi_i = 0 \Rightarrow y_i\big(w \cdot \phi(x_i) + b\big) = 1 - \xi_i$. From this, it can be found that QPSVM gives three types of KCV patterns (that is, patterns having $\alpha_i \neq 0$) considering the values of ξ: i) $\xi = 0$ ii) $0 < \xi \leq 1$ iii) $\xi > 1$. But after a little calculation with another KKT condition, it can be seen that if $\xi > 0$ for any pattern, its corresponding co-efficient value, α becomes equal to the penalty parameter, C that is the upper limit of α values. Thus, discarding patterns where $\xi > 1$ not only ignores a KKT condition but also removes crucial elements of the decision function that have the largest coefficient values, potentially impacting the decision-making process significantly. Additionally, removing patterns having $\alpha = C$ could significantly disrupt the condition $\sum_{i=1}^{N} \alpha_i y_i = 0$, which ensures the balance between the support vector patterns and their associated coefficients, similar to a mechanical system, thereby impacting the optimal margin and the final decision boundary. Therefore, discarding any pattern from this machine is not in our option.

Role while working with VLPSVM: In this case, unlike QPSVM, values of this stochastic auxiliary variables ξ are found by solving an optimization problem directly using its primal form where no direct involvement among KCV-coefficients, KKT conditions, and ξ are seen. Moreover, there is no other constraint-based restrictions involving the KCV coefficients. This provides an opportunity to work further on this machine. Experience with VLPSVM shows that patterns with $\xi > 1$ have relatively small corresponding coefficients β in the final decision function. This aligns with the two main parts of its objective function: $\min_{\beta,\xi,b} \sum_{i=1}^{N} \beta_i + C \sum_{i=1}^{N} \xi_i$. Nevertheless, these coefficient values do not fully vanish, meaning the corresponding patterns continue to be part of the decision function, with each of them requiring kernel evaluation for classification, even though they may not have a considerable constructive impact on the final decision. This brings the arguments about whether they should be included in the final discriminator, considering the trade-off between classification accuracy and computational cost. To address this issue, we examine their contribution as well as the confusion they create in accurate classification, as explained below:

3.4.3 Contribution and confusion from the patterns having $\xi > 1$ while working with VLPSVM

We have seen earlier that these patterns consistently stay beyond the boundaries of their classes, produce comparatively larger kernel values while in operation with patterns from the opposite classes, leading to confusion.

On contrary, they generate lower kernel values while in operation with patterns from their same classes, which produces weaker contribution in the final discriminating function. However, there may be confusion from the inlier KCVs by very weakly attracting patterns from the other classes, we ignore this here as they are negligible in this circumstance and efficiently handled by the complete machine.

Now, the final discriminating function that decides about any arbitrary training pattern x_a that has class-label y_a is $f(x_a) = \sum_{l \in KCV-set} \beta_l y_l K(x_l, x_a) + b$. Considering $m, n : m \cup n = \{l\}, m \cap n = \{\}, y_m = -y_n$, then x_m, x_n are KCVs. Now as $\beta_m, \beta_n > 0$ & $K(x_m, x_a), K(x_n, x_a) > 0$, if $\frac{\beta_m y_m K(x_m, x_a)}{y_a} < 0 \Rightarrow \frac{y_m}{y_a} < 0 \Rightarrow x_a$ and x_m do not belong to the same class and x_m pulls x_a outwards from x_a's own class (i,e., towards, the opposite class of x_a) and if $\frac{\beta_n y_n K(x_n, x_a)}{y_a} > 0 \Rightarrow \frac{y_n}{y_a} > 0 \Rightarrow x_a$ and x_n are from the same class with x_n pulls x_a towards the x_a's home class. However, from these KCV patterns, those with $\xi > 1$ remain within the boundaries of their opposite classes, produce comparatively larger kernel values while considering samples from the opposite classes, which misleads the discriminating function. Simultaneously, they generate lower kernel values while considering patterns from their same classes, poorly guiding the discriminating function. Thus, to enhance both classification accuracy and speed, we decide to discard these $\xi > 1$ KCV patterns after evaluating their level of confusion (which pushes training patterns away from their respective classes) as well as their contribution (which pulls training patterns towards their own classes) [22], taking into account all training patterns through the following mathematical relations:
Contribution of a $\xi > 1$ KCV, x_O with label y_O on all training patterns, $Contr(x_O) = \frac{1}{Card\{i\}} \sum_{\{i: y_i \cdot y_O = 1\}} \beta_{x_O} K(x_O, x_i)$, whereas Confusion of that KCV, x_O on all training patterns, $Conf(x_O) = \frac{1}{Card\{i\}} \sum_{\{i: y_i \cdot y_O = -1\}} \beta_{x_O} K(x_O, x_i)$.

Extensive experiments conducted on the benchmark datasets using the mathematical formulations above reveal that, in almost all cases, $\xi > 1$ KCV patterns pose confusion more than contribution value. This agrees with our decision to remove $\xi > 1$ KCV patterns from the final discriminating function. Figure 8 on page 173 provides an example of a contribution-confusion sketch.

3.4.4 Contribution, confusion as a function of data mapping

We have discussed about our data mapping and its motivation earlier, here we discuss and check our hypothesis that compact mapping reduces the cardinality of the representer set, that is, information space with respect to the representer set gets denser. It is likely that the KCVs identified after data mapping may form a subset of the KCVs derived from the original non-scaled data or may contain part of it and the KCVs obtained after data mapping should carry more information. To check this, we have computed contribution to confusion ratio (CCR) for the intersecting basis vector sets from before and after mapping. For each of common vectors $KCV_b = B \cap B'$, CCR and CCR' for pre and post mapped basis pattern x_b, x'_b with label y_b, y'_b can be described as follows:

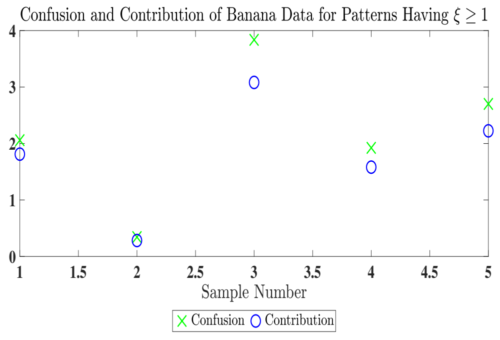
$$CCR = \frac{\frac{1}{\text{card}\{i\}} \sum_{\{i: y_i \cdot y_b = 1\}} \beta_{x_b} K(x_b, x_i)}{\frac{1}{\text{card}\{i\}} \sum_{\{i: y_i \cdot y_b = -1\}} \beta_{x_b} K(x_b, x_i)}$$

$$CCR' = \frac{\frac{1}{\text{card}\{i\}} \sum_{\{i: y_i \cdot y'_b = 1\}} \beta_{x'_b} K(x'_b, x_i)}{\frac{1}{\text{card}\{i\}} \sum_{\{i: y_i \cdot y'_b = -1\}} \beta_{x'_b} K(x'_b, x_i)}$$

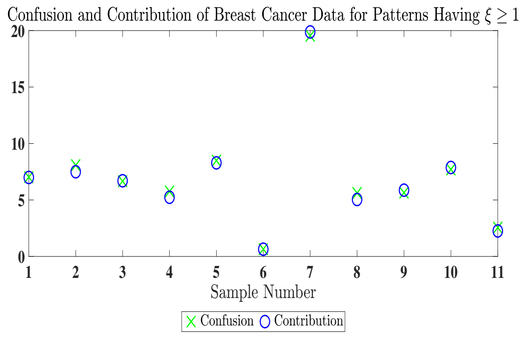
Where, $(x_b, y_b) \in B$, and $(x'_b, y'_b) \in B'$
Our findings show that after data mapping, KCV_b set have a higher contribution-to-confusion ratio, such that $CCR' > CCR$, as illustrated in figure 9 on page 173. This observation supports the idea that the contribution of the reduced basis set becomes significantly higher than the confusion, thereby highlighting the effectiveness of our data mapping.

3.4.5 Updating bias

Weight vector of our sparser machine w_{EBSVM} is found by taking KCV set, S_R as a subset of the KCV set, S from the full scale LPSVM solution so that $S_R \subseteq S$ and if R and M be the elements of S_R and S respectively, then $R \leq M$. Now, if w be the weight vector of this full scale LP SVM machine, it is very most likely that $w_{EBSVM} = \sum_{j=1}^{R} \beta_j y_j \phi(x_j) \neq w = \sum_{p=1}^{M} \beta_l y_l \phi(x_l)$ and thus for any pattern x, $w_{EBSVM} \cdot \phi(x) + b \neq w \cdot \phi(x) + b$. Hence, it is important to provide an optimally updated bias for our proposed EBSVM.
Now, as $R \leq M$, it is normal that model complexity of EBSVM is less than the model complexity of the full scale LPSVM, $\|w_{EBSVM}\|^2 \leq \|w\|^2$ and hence margin of EBSVM is larger than the margin of full scale LPSVM. In another way, as $R \leq M$, 2nd term or the VC-confidence term of the risk bound (4) of EBSVM becomes smaller than that of the full scale LPSVM. But this reduction in the VC-confidence term due to lesser model-complexity generally raises the 1st term, $R_{emp}[f]$ of the risk bound (4). Hence, we need to focus to reduce this part. We can do it by imitating the discriminator function (value) of a complex machine and here we select the full scale LP SVM one as it is the origin from where EBSVM comes after modification. We do it in the following way:
Let the variable bias value of EBSVM is b_R, whose optimum value would be the final value of our EBSVM. Now if δ be the total squared difference between the function values from the full scale LPSVM and variable bias based EBSVM using all training patterns, then we can write,



(a) Banana



(b) Breast Cancer

Figure 8: Confusion & contribution values for the Patterns having $\xi \geq 1$ from the training set

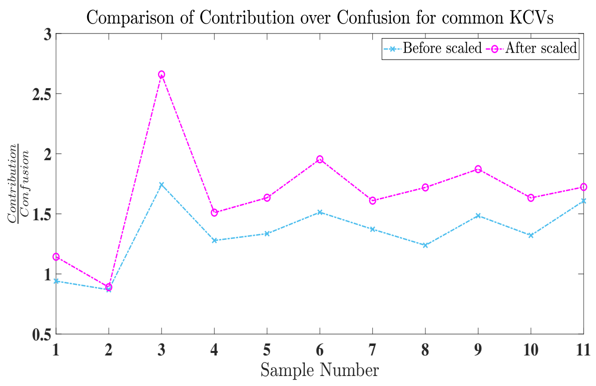


Figure 9: Confusion and contribution ratio for common KCVs from before and after mapping of data

$$\delta = \sum_{i=1}^{N}\left[\left(\sum_{p=1}^{M}\beta_l y_l \phi(x_l)\cdot\phi(x_i) + b\right) - \left(w_{EBSVM}\cdot\phi(x_i) + b_R\right)\right]^2 \quad (15)$$

At the minimum of δ, we should get

$$\frac{\partial\delta}{\partial b_R} = 0 \quad (16)$$

$$\Rightarrow \sum_{i=1}^{N}\left[\left(\sum_{p=1}^{M}\beta_p y_p \phi(x_p)\cdot\phi(x_i) + b\right) - \left(w_{EBSVM}\cdot\phi(x_i) + b_R\right)\right] = 0$$

$$\Rightarrow b_R^* = \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{p=1}^{M}\beta_p y_p \phi(x_p)\cdot\phi(x_i) + b - w_{EBSVM}\cdot\phi(x_i)\right] \quad (17)$$

Algorithm 1: Proposed EBSVM (Efficiency Boosted SVM)

```
1: Input: A set of training data $\{(x_i, y_i)\}_{i=1}^{N}$
2: Output: A discriminator $f_{EBSVM}(\cdot)$
3: Scale the training set efficiently
4: Select the best values of the data dependent parameters: $(Penalty-parameter, Kernel-parameter) \equiv (C, \sum)$
5: Run LP based SVM (VLPSVM) using training data and solve the following optimization problem as below:
   $$\min_{\beta,\xi,b} \sum_{l=1}^{N}\beta_l + C\sum_{l=1}^{N}\xi_l$$
   $$s.t.\ y_i\left(\sum_{l=1}^{N}\beta_l y_l \phi(x_l)\cdot\phi(x_i) + b\right) \geq 1 - \xi_i$$
   $$\beta_l \geq 0;\ l = 1, 2, ..., N; \xi_i \geq 0;\ i = 1, 2, ..., N$$
6: Obtain the KCVs along with their labels $\{(x_p, y_p)\}_{p=1}^{M}$ and the bias term, $b$ derived from the VLPSVM applying $\beta_l > 0$
7: Find $\{\beta_p|\xi_p > 1\}$ and discard the corresponding patterns belong to the KCVs set and form a new smaller set of KCVs (RKCVs), $\{x_p|\xi_p \leq 1\ with\ \beta_p > 0\}$
8: Retrieve the RKCVs with their labels $\{(x_j, y_j)\}_{j=1}^{R}$
9: $w_{EBSVM} \leftarrow \sum_{j=1}^{R}\beta_j y_j \phi(x_j)$
10: $b_{EBSVM} \leftarrow argmin_{b_R}\sum_{i=1}^{N}\left[\left(\sum_{p=1}^{M}\beta_p y_p \phi(x_p)\cdot\phi(x_i) + b\right) - \left(w_{EBSVM}\cdot\phi(x_i) + b_R\right)\right]^2$
11: Return $f_{EBSVM}(\cdot) = w_{EBSVM}\cdot\phi(\cdot) + b_{EBSVM}$
```

This b_R^* is the value of the bias term of the proposed, EBSVM. Algorithm 1 on page 173 outlines the steps of the proposed EBSVM whereas figure 10, 11, 12, 13 show the decision boundaries using both the synthetic and benchmark datasets, from QPSVM, VLPSVM as well as the proposed EBSVM, along with their corresponding KCV counts and training error rate.

3.5 Analyzing noise-sensitivity of the method

To assess the noise sensitivity of our model, we perturbed the input synthetic training data [66] by introducing controlled uniform additive noise with varying levels and observed the behavior of our machine and other two most prominent machines, the standard QP based SVM(QPSVM) and the LP based SVM (LPSVM). To perform a sound comparison among them with solid analysis, we first state the variables with their functional relationships explicitly as the following:

Independent variable: noise, ν.
Dependent variable: Training Accuracy Rate, $A = A(\nu)$; Computational Cost ($\equiv number\ of\ KCV$), $CC = CC(\nu)$.
Different values of these dependent variables depending on the independent variable (noise) are plotted in figure 14 on page 176.

We first follow the sensitivity of the dependent variables with respect to the independent variable and then we measure an the overall effects throughout the whole noise input of different levels.

3.5.1 Sensitivity

We know that the sensitivity of a dependent variable with respect to (its depending) independent variable refers to how much the dependent variable changes in response to a change in the independent variable. Mathematically, this sensitivity is quite often expressed by the the rate of change of the dependent variable with respect to the independent variable. That is, if the dependent variable, F depends on the independent variable, u then sensitivity of F with respect to u is

$$\text{Sensitivity} = \lim_{\Delta u \to 0} \frac{F(u+\Delta u) - F(u)}{\Delta u} = \frac{dF}{du} \tag{18}$$

Thus sensitivity of our two noise dependent variables can be expressed as,

$$\begin{gathered}\text{Training Accuracy Rate Sensitivity,}\\ = \lim_{\Delta\nu \to 0} \frac{A(\nu+\Delta\nu) - A(\nu)}{\Delta\nu} = \frac{dA}{d\nu}\end{gathered} \tag{19}$$

and

$$\begin{gathered}\text{(Kernel) Computational Cost Sensitivity,}\\ = \lim_{\Delta\nu \to 0} \frac{CC(\nu+\Delta\nu) - CC(\nu)}{\Delta\nu} = \frac{dCC}{d\nu}\end{gathered} \tag{20}$$

We have calculated them using the Forward Difference method of Numerical differentiation as below

$$F'(\nu) \approx \frac{F(\nu+h) - F(\nu)}{h} \tag{21}$$

Sensitivity (that is gradient) of these two variables at different noise levels are counted and hence plotted in figure 15 on page 176. From the figure it is clear that in case of Accuracy sensitivity, all three machine behave very much similarly, by being flat and steady upto 55% of the controlled full noise and then a mixed reaction of rise & fall for higher noise levels. But considering Cost sensitivity curve, while QPSVM shows heavy zigzag, our model and LPSVM give almost flat and steady curves showing they are not much effected by the noise. The most likely reason for this in case of EBSVM is the bias updating on top of the mechanism of our machine by stochastic ξ functional analysis, which smartly interfaces with noises.

3.5.2 Graphical analysis of noise induced output

To evaluate the overall performances of the machines throughout the whole noise introduced operation, we use the concept of Epigraph & Hypograph as given below:

Epigraph and hypograph

Let $F : \mathbb{R}^n \to \mathbb{R}$ be a real-valued function.

Epigraph: The **epigraph** of F, denoted by epi(F), is defined as:

$$\text{epi}(F) = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} \mid F(x) \le t\}$$

Geometrically, this is the set of points on or above the graph of the function.

Area of an epigraph: To compute a finite area, we bound it by a constant $M \ge F(x)$ over the interval $[a, b]$. The area between this bound and the function is:

$$Ar_{\text{epi}} = \int_a^b (M - F(x))\, dx \tag{22}$$

Hypograph: The **hypograph** of F, denoted by hypo(F), is defined as:

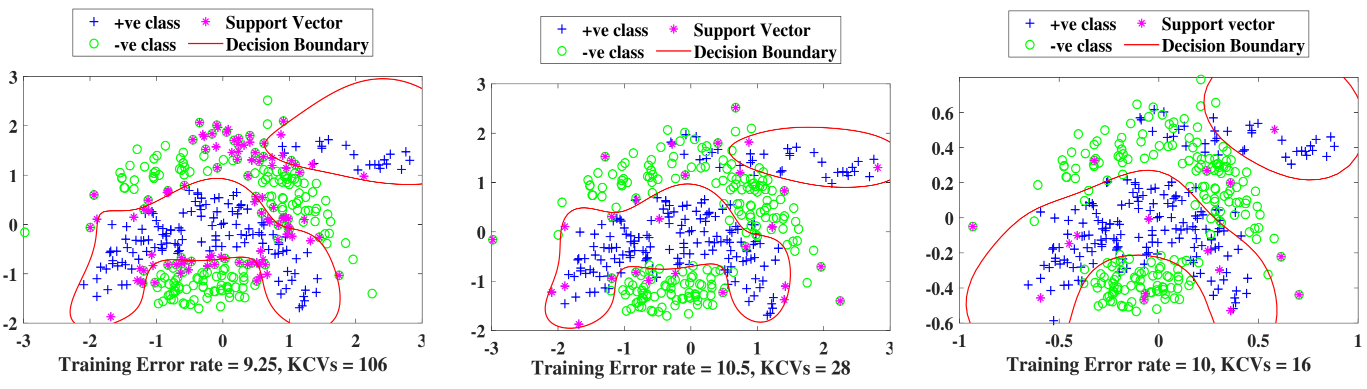
$$\text{hypo}(F) = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} \mid F(x) \ge t\}$$

This is the set of points on or below the graph of the function.

Area of a hypograph: Let $F(x)$ be a continuous real-valued function on an interval $[a, b]$. The area of the hypograph is the area under the graph of the function from a to b, given by:

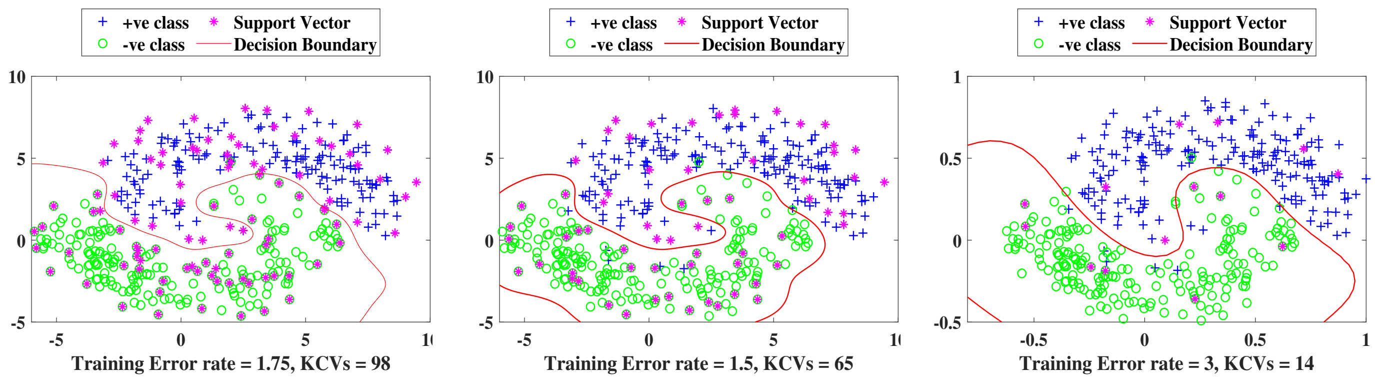
$$Ar_{\text{hypo}} = \int_a^b F(x)\, dx \tag{23}$$

To compute the area in (22) & (23) we have used the following Simpson's Rule to perform numerical integration



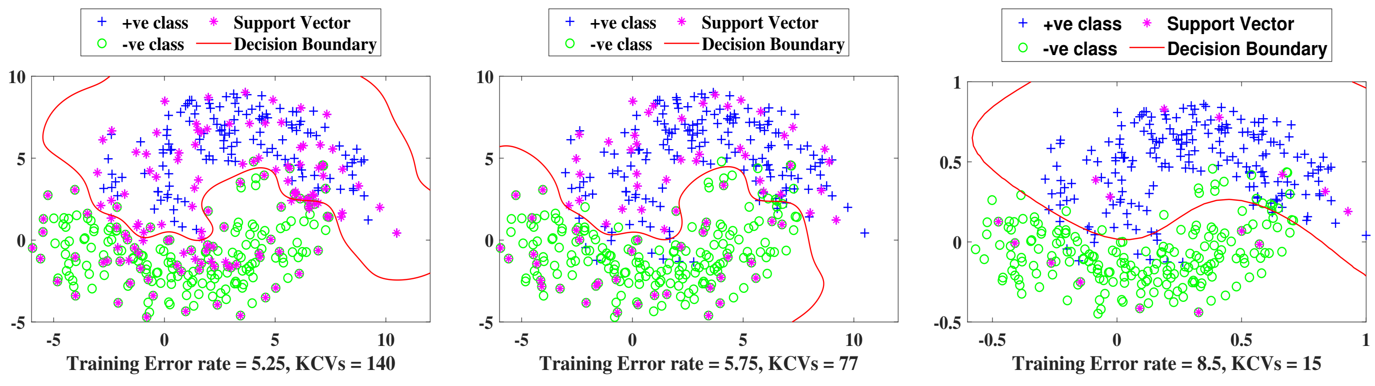
(a) QPSVM (b) VLPSVM (c) Proposed EBSVM

Figure 10: Pattern Classification Boundaries on Banana data along with their respective KCVs counts and training error rates for different machines having the size of training set 400x2.



(a) QPSVM (b) VLPSVM (c) Proposed EBSVM

Figure 11: Pattern Classification Boundaries on synthetic Two spiral data (with noise level 3.3) along with their respective KCVs counts and training error rates for different machines having the size of training set 400x2.

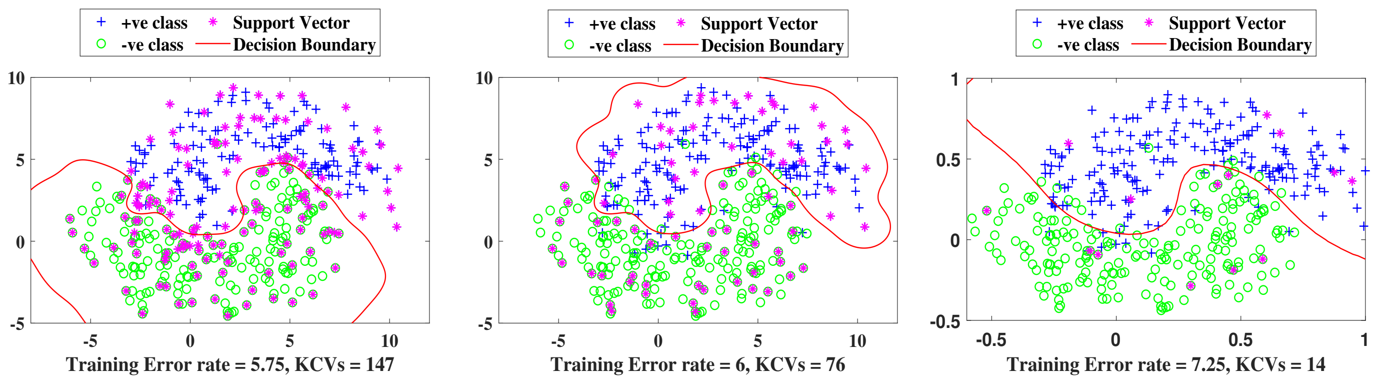


(a) QPSVM (b) VLPSVM (c) Proposed EBSVM

Figure 12: Pattern classification boundaries on synthetic two spiral data (with noise level 4.3) along with their respective KCVs counts and training error rates for different machines having the size of training set 400x2.

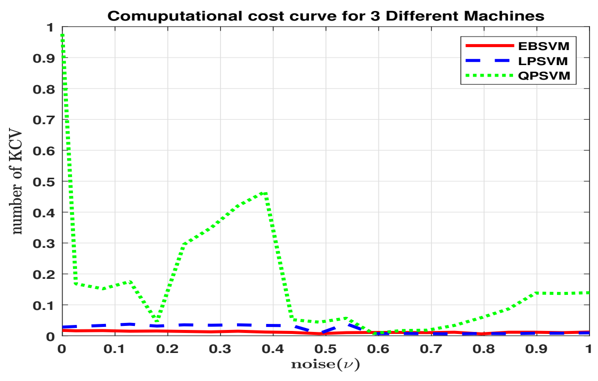
$$\int_a^b F(x)dx \approx \frac{\Delta x}{3}[F(x_0) + 4\sum_{i=1}^{n/2} F(x_{2i-1}) + 2\sum_{i=1}^{n/2-1} F(x_{2i}) + F(x_n)] \tag{24}$$

As we need to maximize Accuracy, we find the area of $hypograph(A(\nu))$, which is the higher the better hav-

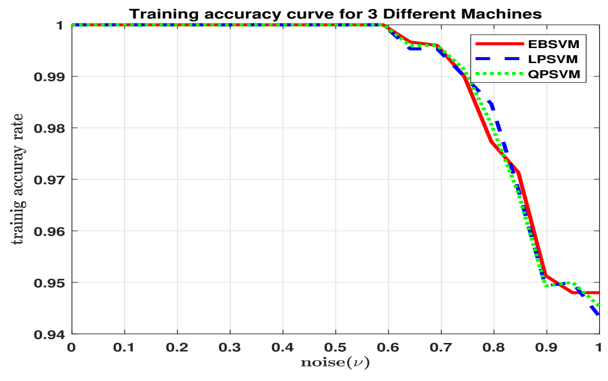


(a) QPSVM (b) VLPSVM (c) Proposed EBSVM

Figure 13: Pattern classification boundaries on synthetic two spiral data (with noise level 4.6) along with their respective KCVs counts and training error rates for different machines having the size of training set 400x2.

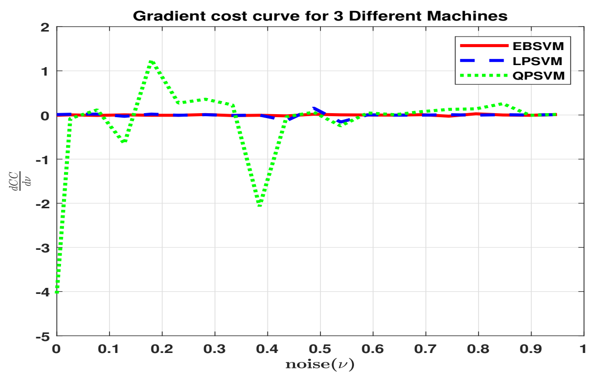


(a) Computational cost

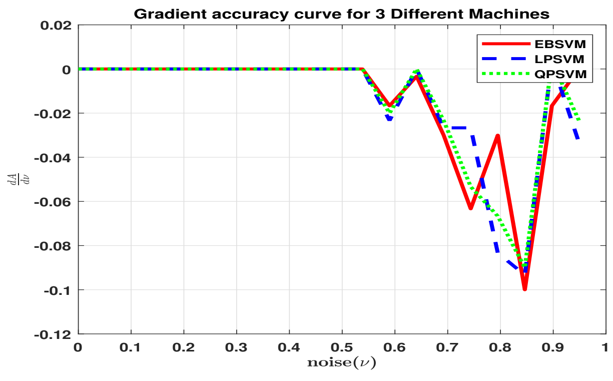


(b) Training accuracy rate

Figure 14: Comparison of (a) computational cost and (b) training accuracy rate with respect to noise levels for EBSVM, LPSVM, QPSVM, respectively.



(a) Computational cost



(b) Training accuracy rate

Figure 15: Sensitivity measurement of (a) computational cost and (b) training accuracy rate with respect to noise levels for EBSVM, LPSVM, QPSVM, respectively.

ing the maximum limit equal to 1, whereas as we want to minimize the Computational Cost, we find the area of $hypograph(CC(\nu))$, which is the smaller the better while having the maximum limit equal to 1. Further, the area between $CC = 1$ (meaning 100% computational cost, that is using all of the training patterns to build up the classifier) & $epigraph(CC(\nu))$ is the Sparsity, which is the higher the better having the maximum limit equal to 1, that is, $area(epigraph(CC(\nu))) - area(epigraph(CC = 1)) = 1 - area(hypograph(CC(\nu))) = Sparsity$.

4 Experimental set-up and results

4.1 Set-up method

In this section, the effectiveness of the suggested EBSVM algorithm is demonstrated by a comparison of its experimental outcomes with those of many state-of-the machines, including QPSVM [45], VLPSVM [59], EESVM [1] and SpSVM-2 [2]. The outcomes are broken down into two dimensions: the algorithm's classification cost and it's performance. Ten publicly available benchmark datasets — Heart, Twonorm, Titanic, Thyroid, Breast Cancer, Banana, Flare Solar, German, and Waveform are used to evaluate the algorithm's performance, as referenced in [65]. Each of these datasets includes 100 realizations of both training and test sets. Scaling is executed on each dataset by dividing with the maximum among the absolute values of the features, as discussed in the Data Scaling section. All studies are conducted using a Gaussian kernel on these scaled datasets. A varied collection of $C \in \{2^0, 2^2, ..., 2^{20}\}$ and $\sum \in \{2^{-2}, 2^0, 2^2, ..., 2^8\}$ is employed in fivefold cross-validation, wherein each C value is normalized by the scaling factor of the respective dataset, thereby calibrating the computation of C for each dataset pattern. Fivefold cross-validation is employed to determine the optimal parameters, extracting the best values of C and $\sum$ for each dataset. The initial training set among 100 is utilized in cross-validation whereas output results (values of different key terms/variables) are collected by taking mean over the 100 example-sets given in the benchmark datasets. The mean number of KCV values and the test error rates for QPSVM, VLPSVM, SpSVM-2, EESVM, and the proposed approach are presented in Table 1 on page 178. The proposed EBSVM achieves similar accuracy with all of these SOTA machines while reducing the KCV count dramatically compared to the standard SVM (QPSVM, which is sparse itself), significantly compared to the sparser VLPSVM while considerably compared to the further sparser EESVM and SpSVM-2. Along with data-description, details of number of KCVs, Error rates from different machines and the proposed one are described in Table 1. All of the experiments were done using MATLAB.

Numerical operations and outputs during noise-sensitivity experiment

We normalize the Training Accuracy axis (in percentage) by 100 as 100% accuracy is the best whereas the noise axis is normalized by the highest noise level. Thus the total area of region from this normalized Noise-Accuracy window is equal to 1. Moreover, as SVMs are sparse by using only a small subset of training points (support vectors) to determine its final model while many of the other machines, for example, k-Nearest Neighbors (k-NN), Kernel Density Estimation (KDE), etc are not sparse as they involve all of the training patterns to build the final machine. Thus we normalize the Computation Cost (i.e., number of KCV) axis by total number of training patterns. Hence the total area of the region from this normalized Noise-Cost window is equal to 1.

To calculate the overall Training Accuracy Rate, A, we use (23) & (24) replacing F by A and using $a = 0, b = 1$. On contrary, to calculate the overall Computational Cost, CC, we use (23) & (24) replacing F by CC and using $a = 0, b = 1$ while to compute the overall sparsity, we use $M = 1$ in (22). All calculated results are show in Table 2 on page 179. From the Table 2 it is seen that while our method, EBSVM gives very similar overall Training Accuracy Rate with respect to the prominent QPSVM and LPSVM, it offers the lowest normalized computational cost region 0.012, being nearly half of LPSVM and 1/13th of QPSVM. In terms of sparsity, while all of these 3 machines generate heavy normalized sparsity region, proposed EBSVM shows the highest sparsity as 99% , whereas the sparser LPSVM gives 98% and the sparse-famous QPSVM gives 84%.

4.2 Result analysis

4.2.1 Error-accuracy in stochastic classification

For any arbitrary pattern x having class label y, our discriminator models the decision function in the form: $f(x) = w^T\phi(x) + b$ with the predicted class label as $Classified\ Class\ Label(x) = y' = sgn(f(x))$. Thus a novel pattern is classified correctly if its decision function value has the same polarity as its actual class label, i.e., $f(x) > 0$ if it is from positive class and $f(x) < 0$ if it is from negative class and consequently, during this detection (to accept positive patterns and reject negative patterns) by classifications of patterns, two types of errors are to be considered first. They are:

False positive error: This is the error when a Negative pattern is considered as a Positive pattern by mistake. We denote this rate or probability as $P_0(x = 1)$. Mathematically, we can define it as,

$$\begin{aligned}
&P_0(x = 1) \\
&= \frac{Number\ of\ Falsely\ accepted\ Negative\ Pattern}{Total\ number\ of\ Input\ Negative\ Pattern} \\
&\Rightarrow P(y' = 1|y = -1) \\
&= \frac{Number\ of\ Falsely\ accepted\ Negative\ Pattern}{Total\ number\ of\ Input\ Negative\ Pattern} \\
&\Rightarrow P(y' = 1|y = -1) \\
&= \frac{\frac{Number\ of\ Falsely\ accepted\ Negative\ Pattern}{Total\ number\ of\ Input\ Pattern}}{\frac{Total\ number\ of\ Input\ Negative\ Pattern}{Total\ number\ of\ Input\ Pattern}} \\
&\Rightarrow P(y' = 1|y = -1) = \frac{P(y'=1, y=-1)}{P(y=-1)}
\end{aligned} \tag{25}$$

Where $P(y' = 1|y = -1)$ is the conditional probability that the predicted class label y' is positive but the actual class label y is negative, $P(y' = 1, y = -1)$ is the joint probability distribution of this predicted and actual class

Table 1: Number of KCVs as well as test error rates for various state-of-the art machines and our proposed method (EBSVM)

Name of Dataset, (#Training Examples, #Testing Examples, #Dimension)	QPSVM [45] (KCVs, TestErR)	VLPSVM [59] (KCVs, TestErR)	EESVM [1] (KCVs, TestErR)	SpSVM-2 [2] (KCVs, TestErR)	EBSVM (KCVs, TestErR)
Banana (400, 4900, 2)	(102.26, 10.61)	(15.08, 10.75)	(13.62, 11.53)	(17.30, 10.87)	(16.64, 10.65)
Breast Cancer (200, 77, 9)	(200, 28.53)	(18.89, 26.05)	*NR	(12.10, 29.22)	(**8.01**, 27.56)
Diabetes (468, 300, 8)	(263.22, 23.28)	(12.58, 23.40)	*NR	(13.80, 23.47)	(**8.81**, 23.66)
Flare Solar (666, 400, 9)	(609.98, 32.56)	(251.66, 32.41)	*NR	(8.40, 33.90)	(**3.87**, 32.32)
German (700, 300, 20)	(438.75, 23.70)	(26.47, 24)	(27.61, 24.16)	(14.00, 24.90)	(19.35, 24.28)
Heart (170, 100, 13)	(68.23, 16.60)	(21.94, 17.44)	(7.45, 15.33)	(4.30, 15.50)	(7.42, **15.23**)
Ringnorm (400, 7000, 20)	(77.00, 2.23)	(15.99, 1.73)	(15.83, 2.54)	(12.9, 1.97)	(15.84, 1.86)
Splice (150, 2051, 3)	(943.65, 12.34)	(292.15, 12.39)	*NR	*NR	(**283.25**, 12.39)
Titanic (150, 2051, 3)	(148.50, 22.68)	(83.91, 22.91)	(39.19, 23.18)	(3.30, 22.68)	(5.70, 23.12)
Waveform (400, 4600, 21)	(228.33, 13.15)	(20.81, 11.18)	(10.67, 12.72)	(14.40, 10.66)	(13.01, 10.81)
Overall Average	(307.99, 18.55)	(75.95, 18.24)	(- , -)	(- , -)	(**38.19**, 18.37)
Average (1) w.r.t SpSVM-2	(237.36, 19.24)	(51.93, 18.89)	(- , -)	(11.17, 19.24)	(**10.96**, 19.03)
Average (2) w.r.t EESVM	(177.18, 14.83)	(30.7, 14.67)	(19.06, 14.91)	(- , -)	(**12.99**, 14.55)

*NR: Not Reported

Table 1 provides a detailed comparison of Kernel Computing Vectors (KCVs) and test error rates (TestErR) across various SVM models including QPSVM, VLPSVM, EESVM, SpSVM-2, and the proposed EBSVM. The results highlight that EBSVM achieves significant computational gain while maintaining strong classification accuracy. Notably, in the breast cancer, flare solar, heart, and splice datasets, EBSVM demonstrates remarkable efficiency, significantly reducing KCVs compared to other models while maintaining competitive error rates. Looking at the overall average, excluding datasets where values were not reported, EBSVM reduces KCVs by 87.6% compared to QPSVM and 49.7% compared to VLPSVM, with just a minimal 0.7% and 0.71% increase in test error rates, respectively. When compared to Average (1) i.e., w.r.t SpSVM-2, EBSVM requires 1.2% fewer KCVs and achieves a 0.21% lower error rate. Against EESVM of Average (2), EBSVM reduces KCVs by 31.5% while also achieving a 2.4% lower test error rate. These findings confirm that EBSVM is a highly efficient approach, striking the right balance between computational cost and classification accuracy, making it a strongly desired classifier in the filed of modern machine learning.

labels, $P(y = -1)$ is the marginal probability distribution of negative class labels. This can also be thought as $1 - Negative\ Rejection\ Rate$. *Negative Rejection rate* is often termed as *power*.

False negative error: This is the error when a Positive pattern is considered as a Negative pattern by mistake. Sometimes,it is also called as ”missed detection”. We denote this rate or probability as $P_1(x = 0)$. Mathematically, we can define it as,

$$P_1(x=0)$$

$$= \frac{Number\ of\ Rejected\ Positive\ Pattern}{Total\ number\ of\ Input\ Positive\ Pattern}$$
$$\Rightarrow P(y' = -1|y = 1)$$

$$= \frac{Number\ of\ Rejected\ Positive\ Pattern}{Total\ number\ of\ Input\ Positive\ Pattern}$$

$$\Rightarrow P(y' = -1|y = 1)$$

$$= \frac{\frac{Number\ of\ Rejected\ Positive\ Pattern}{Total\ number\ of\ Input\ Pattern}}{\frac{Total\ number\ of\ Input\ Positive\ Pattern}{Total\ number\ of\ Input\ Pattern}}$$

$$\Rightarrow P(y' = -1|y = 1) = \frac{P(y' = -1, y = 1)}{P(y = 1)} \tag{26}$$

Where $P(y' = -1|y = 1)$ is the conditional probability that the predicted class label y' is negative but the actual class label y is positive, $P(y' = -1, y = 1)$ is the joint probability distribution of this predicted and actual class labels, $P(y = 1)$ is the marginal probability distribution of positive class labels.

Using the above two terms, we can also find the total error probability during each detection as below

$$\begin{aligned}
P(Error) &= P(Classified\ Class\ Label(x) \\
&\qquad \neq Actual\ Class\ label(x)) \\
&= P(y' \neq y) = P(y' = 1, y = -1) \\
&\quad + P(y' = -1, y = 1) \\
&= P(y' = 1|y = -1)P(y = -1) \\
&\quad + P(y' = -1|y = 1)P(y = 1) \\
&= \frac{Number\ of\ accepted\ Negative\ Pattern}{Total\ number\ of\ Input\ Negative\ Pattern} \\
&\quad \frac{Total\ Number\ of\ Negative\ Pattern}{Total\ number\ of\ Input\ Pattern} \\
&\quad + \frac{Number\ of\ Rejected\ Positive\ Pattern}{Total\ number\ of\ Input\ Positive\ Pattern} \\
&\quad \frac{Total\ Number\ of\ Positive\ Pattern}{Total\ number\ of\ Input\ Pattern} \\
&= \frac{\text{Number of accepted Negative Pattern} \atop \text{+Number of Rejected Positive Pattern}}{\text{Total number of Input Pattern}}
\end{aligned} \tag{27}$$

and accordingly, the total accuracy probability during each detection is

$$P(Accuracy) = 1 - P(Error)$$

However, in our evaluation, we measure the error rates and accuracy in percentages (%) as

$$Error\ Rate = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i \neq y_i'] \times 100$$

where N is the total number of classified patterns and **1** is the indicator function.

Table 2: Comparison of machines based on computational cost and training accuracy rate

	EBSVM	**LPSVM**	**QPSVM**
Computational Cost (CC)	0.01209	0.02204	0.1573
Sparsity	0.98791	0.97796	0.8427
Training Accuracy Rate(A)	0.9904	0.9903	0.9903

4.2.2 Machine accuracy cost (MAC)

A powerful yet cost-effective computing model is always really preferred. Considering this the objective is to design a discriminator that minimizes kernel evaluations during testing while preserving high accuracy. To strike a balance between an expensive machine with superior accuracy and a highly cost-effective system with similar efficiency, the concept of MAC [21] is introduced for kernel-based models, defined as follows:

$$MAC = \frac{Number\ of\ KCVs}{Test\ Accuracy} \tag{28}$$

A lower MAC value is always desirable, as it signifies that the machine attains optimal test accuracy with minimal kernel computations (KCVs). The values of MAC for different systems are provided in Table 3 in page 180.

4.2.3 GFR (Generalization failure rate)

We know that the optimal stochastic machine is designed by properly tuned from the bias-variance tradeoff and gives least generalization error ensuring that the model is not overfitting to the training data and can perform well on novel, unseen data. We have also tried to activate this on our model basing on some suitable mathematical formulations. However, as it is further sparsified from a sparser machine, its bridging capability between the training and new unseen data becomes more into interest. In this regard, we have used the term GFR [21] to assess the generalization deficiency of our machine as below, which is defined considering two points simultaneously: i) how badly the machine overfits, and ii) how poor it performs on test patterns:

$$\begin{aligned} &Overfitting\ Tendency,\ OT \\ &= \frac{Test\ Error\ Rate - Train\ Error\ Rate}{Train\ Error\ Rate} \end{aligned} \tag{29}$$

$$GFR = \frac{OT}{Test\ \ Accuracy} \tag{30}$$

Why less GFR or good generalization performance from our proposed EBSVM

As we have seen that $w = \sum \beta_j y_j \phi(x_j)$, for LP SVM, throwing KCVs with $\beta_j > 0$ gives less $||w||$, hence higher margin and reduced VC confidence, 2nd part of the RHS of the risk bound (4) becomes smaller. This usually increases the 1st part of the RHS of the risk bound (4), $R_{emp}[f]$ due to the reduced model complexity. But in our algorithm, we generate the bias value of our final discriminator by imitating the function value of the complex and non-reduced discriminator, which leads us to realize the final discriminator generated from an optimally further larger margin and sparser class based hypothesis where the hyperplane is shifted towards the complex model reducing the value of $R_{emp}[f]$ as well. Thus the overall value of the right hand side of (4) gets lower and we manage to get the machine having reduced upper bound of the actual risk that gives less generalization error. The values of GFR for different machines are provided in Table 4 on page 180.

5 Conclusion and future work

In this paper, we have discussed one of the most mathematically solid statistical machine learning approaches, Support Vector Machine (SVM) that gives decision under uncertainty depending on the given probabilistic data and their dependent stochastic parameters. We have gone through briefly using its two main mathematical bases, statistical theory and mathematical programming. Basing on these concepts, we have gone a bit further to produce a very efficient detector having both high classification accuracy and speed that is realized from further sparsification of Vapnik's LPSVM, which is indeed sparser itself. This proposed machine requires kernel computation upto as small as 0.6% of the sparse QPSVM, 1.5% of the sparser VLPSVM, 14.5% of the sparser EESVM, and 46.1% of the sparser SpSVM-2 by Keerthi et al [2], whereas in average 12.4% of the sparse QPSVM, 50.3% of the sparser VLPSVM, 68.1% of the sparser EESVM, and 98.1% of the sparser SpSVM-2 though produces very similar classification accuracy despite of being very straight forward by demanding very little training-effort. It also poses the least MAC (Machine Accuracy Cost, a term to assess the kernel load with respect to accuracy) value considering these machines while giving

Table 3: MAC (machine accuracy cost) table

Dataset	QPSVM	VLPSVM	EESVM	SpSVM-2	EBSVM
Banana	1.144	0.169	0.154	0.194	0.186
Breast Cancer	2.798	0.255	-	0.171	**0.111**
Diabetes	3.431	0.164	-	0.180	**0.116**
Flare Solar	9.025	3.732	-	0.127	**0.057**
German	5.750	0.348	0.364	0.186	0.256
Heart	0.818	0.266	0.088	0.051	0.088
Ringnorm	0.788	0.163	0.162	0.132	0.162
Splice	10.765	3.335	-	-	**3.233**
Titanic	1.921	1.088	0.510	0.043	0.074
Waveform	2.629	0.234	0.122	0.161	0.148

Table 3 presents the MAC values for the different machines, including the proposed algorithm. According to the results, the proposed algorithm completely outperforms all other classifiers in terms of MAC values for the Flare Solar, Diabetes, Breast Cancer, and Splice datasets while giving much smaller MAC values compared to the QPSVM, VLPSVM for all datasets. However, in case of other six datasets, it gives very similar MAC values compared to the other two further sparser machines, EESVM and SpSVM-2.This shows that compared to the reported classification algorithms, the suggested model achieves a more cost-efficient accuracy.

Table 4: Generalization failure rate (GFR) for different machines

Dataset	QPSVM	VLPSVM	EESVM	EBSVM
Banana	5.97×10^{-3}	6.14×10^{-3}	1.8×10^{-3}	2.18×10^{-3}
Breast Cancer	1.82×10^{-3}	2.89×10^{-3}	-	$\mathbf{1.23 \times 10^{-3}}$
Diabetes	1.36×10^{-3}	1.39×10^{-3}	-	$\mathbf{1.18 \times 10^{-3}}$
Flare Solar	-7.67×10^{-5}	-9.5×10^{-5}	-	-9.06×10^{-5}
German	1.44×10^{-3}	7.18×10^{-3}	9×10^{-4}	$\mathbf{8.7 \times 10^{-4}}$
Heart	1.27×10^{-3}	1.94×10^{-3}	1×10^{-3}	$\mathbf{8.98 \times 10^{-4}}$
Ringnorm	1.22×10^{-1}	1.03×10^{-2}	1.62×10^{-2}	2.21×10^{-2}
Splice	4.11×10^{-1}	9.42×10^{-0}	-	9.42×10^{-0}
Titanic	1.01×10^{-3}	1.49×10^{-3}	1×10^{-3}	1.59×10^{-3}
Waveform	1.1×10^{-2}	4.25×10^{-3}	1.8×10^{-3}	2.38×10^{-3}

Table 4 presents the GFR values of various algorithms, including the proposed EBSVM, alongside QPSVM, VLPSVM, and EESVM (SpSVM-2 is not reported here due to the lack of information). The results indicate that for the Breast Cancer, Diabetes, German, and Heart datasets, the proposed machine achieves lower GFR values compared to all other machines. However, for the remaining datasets, the proposed machine generally performs better in most cases.

least GFR(Generalization failure rate) values compared to these SOTA machines as well, which proves its high generalization quality too.

Moreover, like other two most popular machines, QPSVM and LPSVM, our model's Training Accuracy shows almost no sensitivity for low and mid noise levels but sensitive towards higher noise levels. However, interestingly, while (Kernel) Computational cost by QPSVM shows sensitive throughout the whole noise levels, LPSVM and proposed EBSVM pose negligible sensitivity in this scenario. Further, while unlike many other prominent machine learning algorithms QPSVM and LPSVM offer heavy sparsity (here, 84% and 98% respectively), our method gives the most (here, 99%) sparsity proving its unique performance in case of noisy data.

It is seen from the experiment that after our compact deterministic mapping of probabilistic data, contribution increases relatively higher compared to the confusion from the same KCVs. This supports our idea that by our mapping, topological spanning of the data shrinks without harming its stochastic nature countably and a basis set with comparatively less number of patterns becomes sufficient to represent the dataset. Hence relatively fewer number of KCVs can efficiently generate the discriminator machine having optimum generalization capability. It is also seen that in case of the outlier KCVs, contribution gets smaller than confusion and hence these misleading (playing undesired role in the final discriminator) or least significant outlier KCVs become potential to be thrown away from the basis set of the final discriminator for further sparsification of the sparser machine, which simultaneously helps to maintain high classification accuracy by reducing the probability of over-fitting and boost up the classification speed. Our method recovers the small scale gap that exists from SVM model which are:

i) It focuses on minimizing the error-depth instead of emphasizing the classification-accuracy directly.

ii) There is no complete certainty to find the best values of the regularization parameters from their range of continuous and real values through cross-validation process where a small deviation from the best values may lead to the solution that selects more patterns to be Support Vectors which drags the model towards over-fitting while increasing computational cost as well.

iii) Having a highly deviated and noise-contaminated heavy-outlier may change the SVM solution dramatically from the training set without having it. While some state-of-the-art machines pose the risk of decreasing classification accuracy by sparsification from throwing some significant basis vectors, our model does not do so as its sparsification is done by following a smart stochastic mathematical analysis. By throwing outlier KCVs, our model reduces machine-misleading noise, gives a hypothesis from a class with less VC dimension, larger margin with reduced model complexity and VC confidence term in (4). Further, updating the bias term of our machine optimally, we have also managed to reduce the empirical risk from our model simultaneously. These jointly lowers the upper bound on the actual risk from our model that eventually produces a machine with excellent generalization performance, which is experimentally demonstrated from the GFR (Generalization Failure Rate) and Test Error Rate of our machine on benchmark data. However, a possible limitation of this method is that it may not show this high efficiency(sparsification) in case of datasets having very few outliers or noise.

In case of a highly complex and large dataset, a well generalized SVM is likely to get more outliers that are also included into the SVs set making the machine more complex and computationally expensive, which is efficiently tackled by our stochastic ξ analysis to make this machine optimally sparse and complex while in case of the dataset is less complex without having many outliers, our method will produce sparser solution with less number of KCVs naturally. This certainly proves the significance of our model.

Finally, while a classifier with high accuracy and speed like this proposed EBSVM is really indispensable to satisfy constraints in real-time, in future, we will try to make the KCV set κ-sparse, where κ will be a user defined variable.

References

[1] A. K. Kundu, R. Karim, and A. A. Ave, "Escalating svm-efficiency by sequential qp lp and slack variable analysis," *Journal of Computer Science*, vol. 19, no. 10, pp. 1253–1262, 2023. [Online]. Available: https://thescipub.com/abstract/jcssp.2023.1253.1262

[2] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *Journal of Machine Learning Research*, vol. 7, pp. 1493–1515, 2006. [Online]. Available: https://www.jmlr.org/papers/v7/keerthi06a.html

[3] I. Steinwart, "Sparseness of support vector machines—some asymptotically sharp bounds," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds., 2004, pp. 1069–1076. [Online]. Available: https://proceedings.neurips.cc/paper/2003/file/4c8c76b39d294759a9000cbda3a6571a-Paper.pdf

[4] T. Downs, K. E. Gates, and A. Masters, "Exact simplification of support vector solutions," *Journal of Machine Learning Research*, vol. 2, pp. 293–297, 2001. [Online]. Available: https://dl.acm.org/doi/10.5555/944790.944814

[5] T. Joachims and C.-N. J. Yu, "Sparse kernel svms via cutting-plane training," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, vol. 5781. Springer, 2009, pp. 47–62. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-04180-8_8

[6] A. Cotter, S. Shalev-Shwartz, and N. Srebro, "Learning optimally sparse support vector machines," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 266–274. [Online]. Available: https://proceedings.mlr.press/v28/cotter13.html

[7] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake, "Efficient face detection by a cascaded support–vector machine expansion," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 460, no. 2051, pp. 3283–3297, 2004. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.2004.1333

[8] M. Wu, B. Schölkopf, and G. Bakır, "A direct method for building sparse kernel learning algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 603–624, 2006. [Online]. Available: https://jmlr.org/papers/volume7/wu06a/wu06a.pdf

[9] M. Rätsch, S. Romdhani, G. Teschke, and T. Vetter, "Over-complete wavelet approximation of a support vector machine for efficient classification," in *Pattern Recognition: 27th DAGM Symposium*, ser. Lecture Notes in Computer Science, vol. 3663. Springer, 2005, pp. 351–360. [Online]. Available: https://link.springer.com/chapter/10.1007/11550518_44

[10] B. Heisele, T. Serre, S. Prentice, and T. Poggio, "Hierarchical classification and feature re-

duction for fast face detection with support vector machines," *Pattern Recognition*, vol. 36, no. 9, pp. 2007–2017, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0031320303000621

[11] R. Karim, M. Bergtholdt, J. H. Kappes, and C. Schnörr, "Greedy-based design of sparse two-stage svms for fast classification," in *Pattern Recognition: 29th DAGM Symposium*, ser. Lecture Notes in Computer Science, vol. 4713. Springer, 2007, pp. 395–404. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-74936-3_40

[12] K. Z. Arreola, J. Fehr, and H. Burkhardt, "Fast support vector machine classification using linear svms," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, 2006, pp. 366–369.

[13] H. Sahbi and D. Geman, "A hierarchy of support vector machines for pattern detection," *Journal of Machine Learning Research*, vol. 7, pp. 2087–2123, 2006. [Online]. Available: https://jmlr.org/papers/v7/sahbi06a.html

[14] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel svms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 66–77, 2013. [Online]. Available: https://acberg.com/papers/mbm2012pami.pdf

[15] Ľubor Ladický and P. H. S. Torr, "Locally linear support vector machines," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 985–992. [Online]. Available: https://icml.cc/Conferences/2011/papers/508_icmlpaper.pdf

[16] Z. Xu, J. Gardner, S. Tyree, and K. Weinberger, "Compressed support vector machines," *arXiv preprint arXiv:1501.06478*, 2015. [Online]. Available: https://arxiv.org/abs/1501.06478

[17] J. Li and Y. Zhang, "Learning surf cascade for fast and accurate object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3468–3475.

[18] V. C. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: Tradeoff between accuracy and cost," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. ACM, 2010, pp. 853–860. [Online]. Available: https://dl.acm.org/doi/10.1145/1835804.1835912

[19] R. Karim, A. K. Kundu, and A. A. Ave, "Sequential mathematical programming with ζ-analysis for hesvm," in *Mathematics and Computer Science: Contemporary Developments*. BP International, 2024, vol. 7, pp. 144–169. [Online]. Available: https://stm.bookpi.org/MCSCD-V7/article/view/16162

[20] Z. Fu, A. Robles-Kelly, and J. Zhou, "Mixing linear svms for nonlinear classification," *IEEE Transactions on Neural Networks*, vol. 21, no. 12, pp. 1963–1975, 2010. [Online]. Available: https://doi.org/10.1109/TNN.2010.2080319

[21] R. Karim and A. K. Kundu, "Efficiency and performance analysis of a sparse and powerful second order svm based on lp and qp," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 311–318, 2018. [Online]. Available: https://thesai.org/Publications/ViewPaper?Volume=9&Issue=2&Code=IJACSA&SerialNo=44

[22] ——, "Computational analysis to reduce classification cost keeping high accuracy of the sparser lpsvm," *International Journal of Machine Learning and Computing*, vol. 9, no. 6, pp. 728–733, 2019. [Online]. Available: https://www.ijmlc.org/vol9/865-L0294.pdf

[23] V. Franc and V. Hlaváč, "Greedy algorithm for a training set reduction in the kernel methods," in *Computer Analysis of Images and Patterns (CAIP)*, ser. Lecture Notes in Computer Science, vol. 2756. Springer, 2003, pp. 426–433. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-45179-2_53

[24] Q. Gu and J. Han, "Clustered support vector machines," in *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013, pp. 307–315. [Online]. Available: https://proceedings.mlr.press/v31/gu13b.html

[25] R. Karim, M. Hasan, A. K. Kundu, and A. A. Ave, "Lp svm with a novel similarity function outperforms powerful lp-qp-kernel-svm considering efficient classification," *Informatica*, vol. 47, no. 8, pp. 1–12, 2023. [Online]. Available: https://www.informatica.si/index.php/informatica/article/view/4767

[26] S. Zhou, "Sparse SVM for Sufficient Data Reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4945–4958, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9415153

[27] D. Geebelen, J. A. K. Suykens, and J. Vandewalle, "Reducing the number of support vectors of svm classifiers using the smoothed separable case approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 4, pp. 682–688, 2012.

[28] H. Wang, Y. Shao, S. Zhou, C. Zhang, and N. Xiu, "Support Vector Machine Classifier via L0/1 Soft-Margin Loss," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 7253–7265, 2022.

[29] J. Liu, L.-W. Huang, Y.-H. Shao, W.-J. Chen, and C.-N. Li, "Nonlinear kernel support vector machine with 0-1 soft margin loss," *arXiv preprint arXiv:2203.00399*, 2022. [Online]. Available: https://arxiv.org/abs/2203.00399

[30] M. Wu, Z. Yang, and J. Ye, "Nonlinear Kernel-Free Quadratic Hyper-Surface Support Vector Machine with 0-1 Loss Function," *arXiv preprint arXiv:2404.10559*, 2024. [Online]. Available: https://arxiv.org/abs/2404.10559

[31] M. Akhtar, M. Tanveer, and M. Arshad, "RoBoSS: A robust, bounded, sparse, and smooth loss function for supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[32] R. Lin, Y. Yao, and Y. Liu, "Kernel support vector machine classifiers with ℓ_0-norm hinge loss," *Neurocomputing*, vol. 589, p. 127669, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0925231224004405

[33] H. Wang and Y. Shao, "Sparse and robust svm classifier for large scale classification," *Applied Intelligence*, vol. 53, no. 16, pp. 19 647–19 671, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s10489-023-04511-w

[34] ——, "Fast truncated huber loss svm for large scale classification," *Knowledge-Based Systems*, vol. 260, p. 110074, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122011674

[35] H. Wang, Z. Zhu, and Y. Shao, "Fast support vector machine with low-computational complexity for large-scale classification," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, pp. 4151–4163, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268772587

[36] H. Wang, Y. Shao, and S. Zhou, "Fast svm classifier for large-scale classification problems," *Information Sciences*, vol. 625, p. 119136, 2023.

[37] L. K. Hansen, "Stochastic linear learning: Exact test and training error averages," *Neural Networks*, vol. 6, no. 3, pp. 393–396, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/089360809390006I

[38] R. Karim, "Theoretical analysis for exact results in stochastic linear learning," Master's thesis, Citeseer, 2004.

[39] V. Kecman, "Support vector machines—an introduction," in *Support Vector Machines: Theory and Applications*, ser. Studies in Fuzziness and Soft Computing, L. Wang, Ed. Springer, 2005, vol. 177, pp. 1–47. [Online]. Available: https://link.springer.com/chapter/10.1007/10984697_1

[40] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992. [Online]. Available: https://doi.org/10.1162/neco.1992.4.1.1

[41] I. Guyon, V. N. Vapnik, B. E. Boser, L. Bottou, and S. A. Solla, "Structural risk minimization for character recognition," *Advances in Neural Information Processing Systems*, pp. 471–479, 1992. [Online]. Available: https://proceedings.neurips.cc/paper/1991/file/10a7cdd970fe135cf4f7bb55c0e3b59f-Paper.pdf

[42] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 6th ed. John Wiley & Sons, 2021.

[43] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998. [Online]. Available: https://doi.org/10.1023/A:1009715923555

[44] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995. [Online]. Available: https://dl.acm.org/doi/10.5555/525960

[45] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995. [Online]. Available: https://doi.org/10.1007/978-1-4757-2440-0

[46] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*. Springer, 1979.

[47] V. Vapnik and A. Chervonenkis, "A note on one class of perceptrons," *Automation and Remote Control*, vol. 25, 1964.

[48] M. Awad and R. Khanna, "Support vector machines for classification," in *Efficient Learning Machines*. Berkeley, CA: Apress, 2015, pp. 39–66. [Online]. Available: https://doi.org/10.1007/978-1-4302-5990-9_3

[49] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 3, pp. 326–334, 1965. [Online]. Available: https://doi.org/10.1109/PGEC.1965.264137

[50] M. Aizerman, E. Braverman, and L. Rozonoer, “Theoretical foundations of the potential function method in pattern recognition learning,” *Automat. Remote Control*, vol. 25, pp. 821–837, 1964. [Online]. Available: https://cs.uwaterloo.ca/ y328yu/classics/kernel.pdf

[51] K.-L. Du, B. Jiang, J. Lu, J. Hua, and M. N. S. Swamy, “Exploring kernel machines and support vector machines: Principles, techniques, and future directions,” *Mathematics*, vol. 12, no. 24, p. 3935, 2024. [Online]. Available: https://doi.org/10.3390/math12243935

[52] J. Mercer, “Functions of positive and negative type, and their connection with the theory of integral equations,” *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 209, pp. 415–446, 1909. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsta.1909.0016

[53] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[54] N. Aronszajn, “Theory of reproducing kernels,” *Transactions of the American Mathematical Society*, vol. 68, pp. 337–404, 1950.

[55] F. Girosi, “An equivalence between sparse approximation and support vector machines,” Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Tech. Rep. AIM-1606, 1997. [Online]. Available: https://dspace.mit.edu/handle/1721.1/7289

[56] N. E. Heckman, “The theory and application of penalized least squares methods or reproducing kernel hilbert spaces made easy,” Department of Statistics, University of British Columbia, Tech. Rep. Technical Report 216, 1997. [Online]. Available: https://www.stat.ubc.ca/technical-reports-archive/doc/216.pdf

[57] G. Wahba, *Spline Models for Observational Data*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia: Society for Industrial and Applied Mathematics, 1990, vol. 59. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611970128

[58] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: https://doi.org/10.1017/CBO9780511804441

[59] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998. [Online]. Available: https://www.wiley.com/en-us/Statistical+Learning+Theory-p-9780471030034

[60] A. Nefedov, J. Ye, C. Kulikowski, I. Muchnik, and K. Morgan, “Experimental study of support vector machines based on linear and quadratic optimization criteria,” *DIMACS Technical Report 2009-18*, 2009. [Online]. Available: https://doi.org/10.1109/icmla.2009.52

[61] T. Suttorp and C. Igel, “Multi-objective optimization of support vector machines,” pp. 199–220, 2006. [Online]. Available: https://doi.org/10.1007/11399346_9

[62] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge: Cambridge University Press, 2000. [Online]. Available: https://doi.org/10.1017/CBO9780511801389

[63] A. Tharwat, A. E. Hassanien, and B. E. Elnaghi, “A ba-based algorithm for parameter optimization of support vector machine,” *Pattern Recognition Letters*, vol. 93, pp. 13–22, 2016.

[64] S. R. Gunn, “Support vector machines for classification and regression,” School of Electronics and Computer Science, University of Southampton, Tech. Rep. ISIS Technical Report, 1998. [Online]. Available: https://eprints.soton.ac.uk/256459/

[65] G. Rätsch. Benchmark datasets. Re-packaged and available on GitHub. [Online]. Available: https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets

[66] J. Kools, “6 functions for generating artificial datasets,” 2013. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets