STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, December 2026, pp 5413–5437.
Published online in International Academic Press (www.IAPress.org)

SABSVM: Speed-Accuracy Boosted SVM with Compact Basis Set from Analytic Mappers via Kreĭn-Hilbert Subspace Compression

Rezaul Karim*, Zakir Hossain, Ali Ahmed Ave, Amit Kumar Kundu

Department of Electrical and Electronic Engineering, Uttara University, Dhaka, Bangladesh

Abstract MKL with SVM learns a combination of kernels that are individually associated with a RKHS and the resultant kernel induces a corresponding synthesized rich RKHS that makes the machine ready to handle data from different and heterogeneous subspaces of the input space sources. However, while providing many and highly flexible base kernels with SVM oriented MKL algorithm may overfit data and boost the test cost by generating numerous Support Vectors (SVs), usually, best performing MKL involve complex optimizations like SDP, SOCP, SILP turning these methods unviable in training phase by the surge of combined kernel number with memory exhaustion. In this circumstance, moving towards extreme challenge to achieve the top in both classification accuracy and speed simultaneously, in our novel SABSVM algorithm, we avoid overfitting, reduce training and test load by taking efficient combination of a universal Positive Definite (PD) kernel and a learning-theory supported optimal analytic mapper that resultantly induces a combined RKHS-RKKS generated rich space as the base of our SVM based detector. Then finding its subspace compression gives further compact subspace approximation of the target function using a significantly much smaller subset of the main kernel-vector (SV) set. Experiments on benchmark and synthetic data exposes that our method poses classification accuracy similar to that of state-of-the-art classifiers while demanding substantially lower classification cost. It gives test classification accuracy almost as good as these higher complexity based top-accuracy giving SKL & MKL classifiers while competes easily with the reduced versions of such, which is clear from its lowest *Machine's Error-Cost Loss (MECL)* on benchmark data. Moreover, sparsity of our model increases with the increment of the training data-size indicating its higher efficiency on larger data.

Keywords Multiple Kernel Learning (MKL), Single Kernel Learning (SKL), Machine's Error-Cost Loss (MECL), SVM, LP, QP, RKHS, RKKS, PD kernel, Analytic mapper, Sparsity, Classification Accuracy, Overfitting.

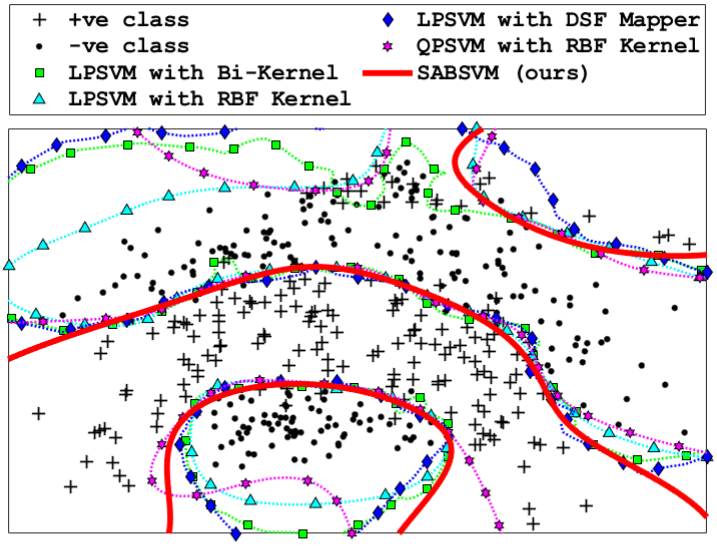
DOI: 10.19139/soic-2310-5070-4301

1. Introduction

Kernel based large margin methods [1, 2] are well established leading algorithms for classification problems in machine learning as they generally offer good performance by incorporating kernel mappings. But while some of these classifiers, for example, SVM [1],[3]–[10], which is a cutting-edge very powerful method by basing on solid mathematical foundation that uses structural risk minimization theory [3] and gives the global minima of a convex problem, its single-kernel learning (SKL) variant [6, 11] may be insufficient to model the complexity of the data by failing to exploit diverse information across feature spaces as real data can arise from different heterogeneous sources and be expressed through various representations. This emphasizes the importance of exploring machine learning methods for maximizing the margin in feature space by combining multiple kernels to get improved performance and adaptability [12, 13] in practical problems. In the domains of machine learning and data mining, such problems are usually recognized as "Multiple Kernel Learning" (MKL) [14]–[20] and has attracted significant concentration from researchers. But a common problem with this high-accuracy classical SKL SVM or the higher-accuracy foundational MKL is that they are generated or implemented by kernel execution, which is costly with

*Correspondence to: Rezaul Karim (Email: rezkar@uttarauniversity.edu.bd). Department of Electrical and Electronic Engineering, Uttara University. 77 Beribadh Road, Dhaka 1230, Bangladesh.

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 2026 International Academic Press



Machine	TrEr (%) (▼)	KCVs (▼)
SABSVM (ours)	8.75	18
LPSVM with Bi-Kernel	7.50	68
LPSVM with RBF Kernel	9.00	26
LPSVM with DSF Mapper	9.75	58
QPSVM with RBF Kernel	8.50	103

Figure 1. Decision boundaries on Banana data (400×2) from proposed SABSVM & some other SOTA machines. From the small metric values above, it can be traced that using the least number of (= 18) Kernel computing Vectors (KCVs) among all, the proposed SABSVM produces Training error (TrEr) of 8.75% very near to the other high accuracy machine, leading to the lowest (157.5) Machine's Error-Cost Loss (MECL, defined later) compared to others (234, 510, 565.5 and 875.5 respectively) as desired. Its decision boundary (red solid line) also looks quite smooth yet powerful.

respect to time or computation. To overcome this limitation, numerous algorithms for efficient SKL [21]–[36] or efficient MKL [37]–[46] are proposed where the main motivation is to improve the accuracy–cost tradeoff basing on acceptable balance among expressivity, optimization complexity, and interpretability. Moreover, in this direction of expressivity, non-PSD or indefinite kernel [47]–[49] are introduced both in SKL [50] and MKL [51] to increase modeling flexibility and efficiently handle domain-specific knowledge. However, this tension between accuracy and computational cost escalates as the dataset gets larger since the number of kernel-computing support vectors (SVs) generally increases when the number of training patterns grows [52] while it is also followed that in the top accuracy domain on complex datasets, SVM often generates a significantly larger number of support vectors to achieve only a modest gain in classification accuracy (Figure 2). Furthermore, the continuous raise in the size and complexity of data, driven by modern technological advancements, has escalated the demand for faster classification. Consequently, optimal minimization of the number of kernel-computing basis functions has become a key target for improving the efficiency of kernel-based classifiers.

In this regard, while the existing prominent or best performing approaches have addressed model compression or kernel sparsification through reduced set methods, budget constraints, sparse regularization, pruning, kernel approximation, cascading etc, most of them require either computationally expensive or complex (like hard or non-convex) optimization while none achieves extreme kernel-computing-vector reduction and MKL like higher accuracy simultaneously. This high ambition is specifically addressed by our novel two-stage compact basis selection algorithm using highly flexible analytic mappers and structured mathematical transformation. It exploits the complementary strengths of PD-non-PSD kernel expressiveness and kernel subspace compression to achieve

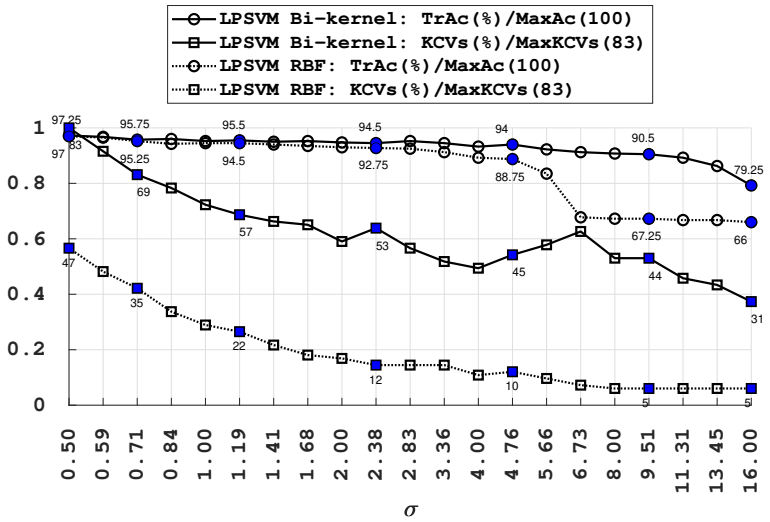


Figure 2. Training Accuracy & #KCV on Kernel Parameter, σ from LPSVM

the most sparse support vector selection and accept the *basis set* elements as only the extremely important patterns belonging to the very tiny fractional part of the training set efficiently so that the resultant discriminator can represent demanded complex functional relationships. A fast impression about our model's discrimination capability along with some other state-of-the-art machines can be achieved from Figure 1.

The rest of this paper is organized with the following sequence: Model's Feasibility Analysis is presented in section 2, Section 3 presents the relevant minimum basics whereas Section 4 explains the proposed method with a concise analysis. Section 5 describes the experimental details while Section 6 concludes by an open discussion and possible directions for future work.

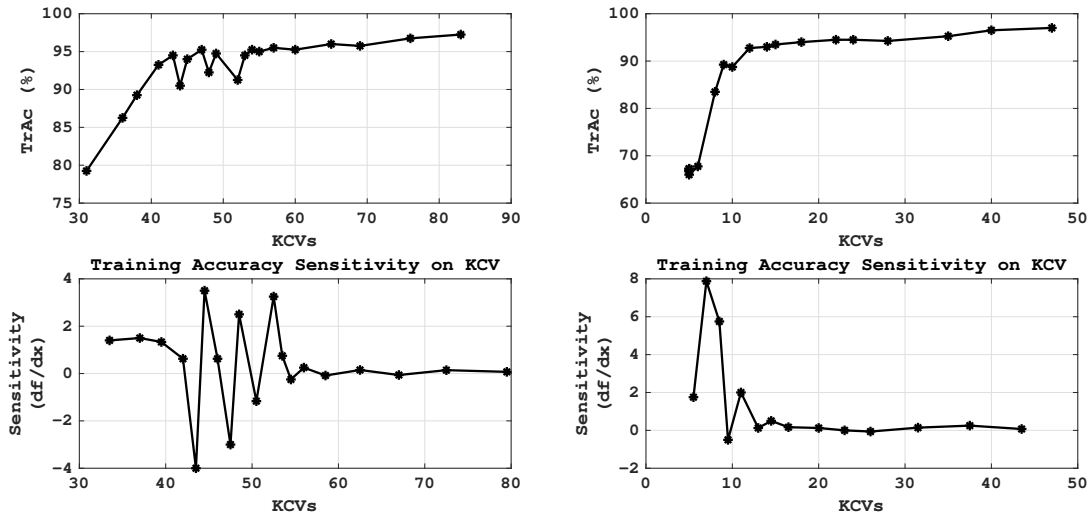
2. Model A Priori Feasibility Analysis

As in all circumstances, optimum accuracy from the machine is a mandatory objective, we start with a fundamental accuracy-sensitivity analysis to observe the feasibility of our speed-accuracy trade-off goal in the following.

Figure 2 shows two pairs of curves representing training accuracy and KCV of the high accuracy region depending on σ from Bi-kernel (RBF & L_1 Mapper) LPSVM and RBF based SKL LPSVM. First, for the Bi-kernel one in the top accuracy neighborhood (termed as the *Highly Interesting Region*, HIR), the accuracy falls very little even by drastic reduction of KCV density. This leads to the empirical hypothesis that if the accuracy function is considered as dependent on KCVs, its HIR has negligible sensitivity with respect to KCV, that is, $\frac{d(accuracy)}{d(KCV)} \approx 0$, which is also represented by its flat trajectory in Figure 3. This gives an expectation that if depending on the application, a very tiny amount of accuracy budget can be sacrificed in our kernel based machine, we can reduce number of KCVs significantly, where these KCVs are produced due to different rich functional spaces by the virtue of Bi-kernel (that may also be the reason for the training accuracy curve to be upright for long in Figure 2). That is, discarding a large amount of support vectors from the Bi-kernel based framework through an optimal process will still give a rich representer set that can model a machine with strong generalization capability.

On contrary, curves from RBF based SKL SVM show that although its accuracy stays below from Bi-kernel one and falls earlier than it in the high accuracy region, its accuracy is very close to the Bi-kernel one and behaves very similar to it considering KCV based sensitivity throughout the HIR. This tells that RBF based SKL SVM can also produce good results with its optimally designed reduced version and has quite the potential to be further boosted by following sequential alignment and tracing this highly compact and rich sub-set of representer vectors

from the Bi-kernel. This hierarchical events can dramatically reduce kernel computational cost of the machine and hence boost up its classification speed while preserving the high accuracy from the Bi-kernel base high-accuracy discriminator.



(a) LPSVM Bi-kernel (b) LPSVM RBF

Figure 3. Training Accuracy & Accuracy-Sensitivity on KCV. Very low and nearly flat sensitivity in the top accuracy region is interesting.

3. Related Basics

3.1. *Fundamental Concepts*

As our proposed method is mainly built upon the SVM framework, we will have a brief discussion about SVM with its small mathematical augmentation. Prior to this, we discuss some very fundamental concepts that are directly related to it.

3.2. *Kernel, Similarity Function, Mapper*

3.2.1. Kernel: Cover's theorem [53] stats that non-linear mapping of data into a higher-dimensional space increases its probability to be linearly separated. Accordingly, many machines employ the mappers following kernel method [54, 55] that replaces standard inner products with an optimal positive-definite function characterized by Mercer's Theorem [56].

Lemma 1 (Manifold Separability [53])**.** Let $\mathcal{X}$ be an input space of patterns. A non-linear mapping $\phi : \mathcal{X} \to \mathcal{H}$ into a higher-dimensional feature space $\mathcal{H}$ strictly increases the probability that a non-linearly separable dataset in $\mathcal{X}$ becomes linearly separable in $\mathcal{H}$.

Theorem 1 (Mercer's Theorem [54]–[56])**.** Let $K(x, y)$ be a continuous, positive semi-definite function on a compact set $\mathcal{X} \times \mathcal{X}$, satisfying $\sum_{i,j=1}^{d} c_i c_j K(x_i, x_j) \geq 0$ for any finite subset $\{x_i\}_{i=1}^{d} \subset \mathcal{X}$ and $\mathbf{c} \in \mathbb{R}^d$. Then, there exists a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that the kernel function implicitly evaluates the inner product in the Hilbert space $\mathcal{H}$:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{1}$$

3.2.2. Similarity Function: However, while kernel theory is well-designed and established, its practical implementation sometimes may seem unproductive by the requirement that similarity functions be positive (semi) definite (PSD), which imposes restrictive conditions that may not always be useful. In contrast, complex similarity functions that satisfy the learning framework of Balcan and Blum [57] and provide strong generalization need not be valid kernel functions. Consequently, learning with non-PSD similarities or indefinite kernels has attracted significant attention with its variants according to strength or so.

Definition 1. [57] A similarity function over X is any pairwise function $\mathcal{K} : X \times X \to [-1, 1]$. We say that $\mathcal{K}$ is a symmetric similarity function if $\mathcal{K}(x, x') = \mathcal{K}(x', x)$ for all x, x'.

Definition 2. [58] $\mathcal{K}$ is a strongly (ϵ, γ)-good similarity function for a learning problem P if at least a $1 - \epsilon$ probability mass of examples x satisfy:

$$\mathbb{E}_{x' \sim P}\big[\mathcal{K}(x, x') \mid \ell(x') = \ell(x)\big] \;\geq\; \mathbb{E}_{x' \sim P}\big[\mathcal{K}(x, x') \mid \ell(x') \neq \ell(x)\big] + \gamma.$$

A well defined similarity function that maps data to higher dimensional space will be noted here as *mapper*.

3.3. Support Vector Machine (SVM)

Having a solid mathematical foundation basing on Statistical Learning Theory (SLT) and convex optimization, SVM has achieved strong generalization property that guided it to lead the most prominent and popular class among machine learning methods. The problem that led to its original formulation exists in multiple forms as the capacity control [59], bias variance trade-off [60], overfitting [61] with the core concept remaining invariant [8].

3.3.1. SVM with SLT: Consider the binary pattern recognition problem as follows: for a given set of sample points, $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{S} \times \{\pm 1\}$, we are to model a *discriminator* $f : \mathcal{S} \to \{\pm 1\}$, with $\mathcal{S}$ denoting a dataset of components from where the patterns, x_i are taken, usually known as the *domain* and y_i are termed as the *labels* of these patterns' classes. A properly trained machine should have good generalization capability that gives a well structured decision function giving possibly small value of the expected test error or the actual risk

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

for $dP(x, y) = p(x, y) \, dx \, dy$ provided that the joint density $p(x, y)$ is well-defined.

Now, incorporating the given training data, the computed mean training error or "empirical risk", denoted by $R_{\text{emp}}[f]$ is expressed as

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

VC theory [4] of SLT establishes an *upper bound* on the above actual risk or test error depending on this empirical risk and the capacity of the function class in the following way for some $0 \leq \eta \leq 1$ and $m > h$, with probability $1 - \eta$:

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

where h is the VC dimension (or "model complexity"), which is a non-negative integer providing an estimate of the aforementioned capacity that controls the main characteristic of a machine. Right-hand part of equation (4) is termed as the "risk bound" whereas its second term is known as the "VC confidence" or "model complexity penalty" and minimization of these bounds directs to the paradigm of *Structural Risk Minimization* (SRM) [62]. SVM, by applying this SRM [63] principle, formulating a convex optimization problem, applies a sequence of models with increasing complexity. Figure 4 illustrates how the total model error varies with the model-complexity-index as expressed by equation (4). Basing on this SRM principle, we discuss about SVM model using optimization in the next.

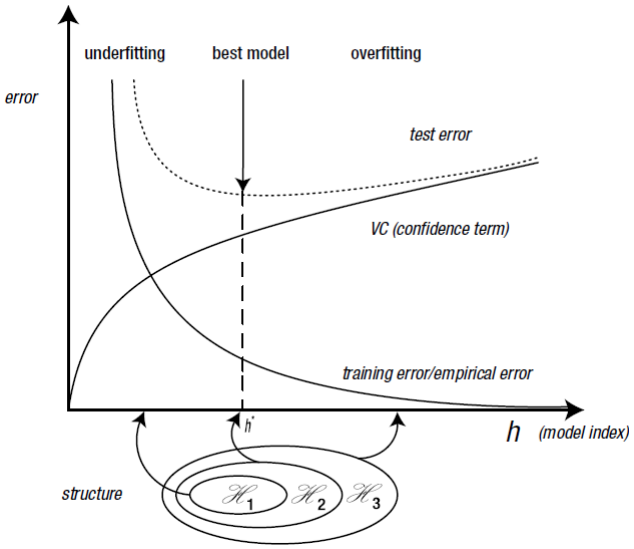


Figure 4. Error Trends On Model Complexity [64].

3.3.2. SVM with Mathematical Programming: SVM is primarily introduced through Quadratic Programming (QP), which we call here QPSVM. On contrary, since the L_1-norm tends to produce sparser solutions than the L_2-norm [65], Vapnik modelled another SVM formulation based on Linear Programming (LP), referred to here as the LPSVM. This section introduces SVM and its two principal formulations: QPSVM & LPSVM while further details can be found in [3].

QPSVM: This corresponds to the classical SVM formulation, which uses SRM concept through cost function, implementing training error penalization and margin maximization between two classes (equivalently, managing bias-variance trade-off) that aims to find the separating hyperplane where margin is the shortest distance from the separating hyperplane to the closest pattern(s) of a class. For our binary classification problem, SVM constructs a classifier using a decision function, $f(x) = w \cdot \phi(x) + b$, which assigns the class label according to $class(x) = sgn(f(x))$ with $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ is a kernel operation. The primal optimization formulation of this QPSVM is given by:

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

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

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

where the auxiliary variables $\zeta_i > 0$ account for patterns that lie outside their respective class margins (here referred to as "margin-outward-deviated patterns"), and $C > 0$ is a data-dependent regularization parameter that controls the trade-off between margin maximization and classification error, in line with the bias–variance trade-off in statistical learning. The respective dual problem is given by:

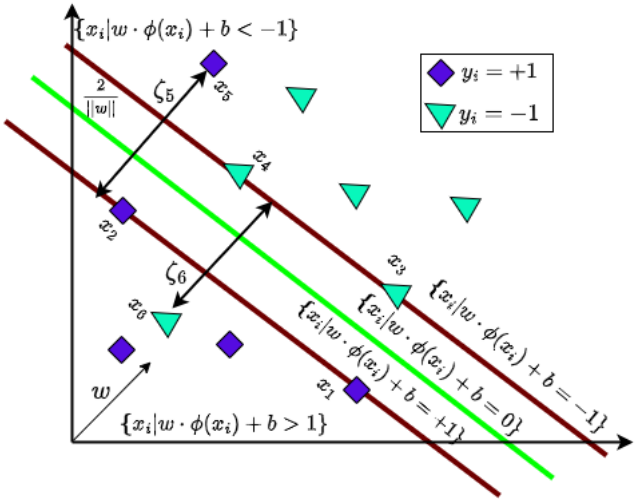


Figure 5. A visual representation of the standard SVM[21]

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

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

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

where the patterns having $\alpha_i > 0$ are the support vectors (SVs) and these α are utilized to compute the bias, b as well as the weight vector w as $w = \sum \alpha_i y_i \phi(x_i)$ where $\|w\|^2$ is inversely proportional to margin. A schematic illustration of this margin-based SVM is shown in Figure 5.

LPSVM: Vapnik developed this method to form a separating hyperplane by effectively incorporating an indirect L_1 norm in objective function, facilitating even further sparsity while managing accuracy similar to QPSVM, which applies the L_2 norm. Consequently, LPSVM becomes similar to QPSVM but is computationally less expensive for pattern classification.

If w denotes the weight variable of the LPSVM hyperplane, its decision function will be expressed as : $f(x) = w\cdot\phi(x) + b$ directing to $class(x) = sgn(f(x))$. Primal formulation of this LPSVM is given below:

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

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

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

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

In this formulation, the auxiliary variables $\xi_i > 0$ are introduced to accommodate the unity-outward-deviated training patterns (i.e., the patterns from training data giving $ClassLabel(x_i)\cdot f(x_i)$ <1). These slack variables are also optimization variables for the problem (11)-(14) that are jointly determined with βs, b, subject to the chosen

hyperparameters. For clarity, the training samples, (x_j) of this LPSVM, associated with nonzero coefficients, $\beta_j > 0$ are referred to as the Expansion Vectors (EVs). Although EVs play a role analogous to Support Vectors (SVs) in QPSVM, they differ in their analytical interpretation. The optimal βs are subsequently used to compute the weight vector as $w = \sum \beta_j y_j \phi(x_j)$. The number of EVs identified by LPSVM algorithm is typically much smaller than the total number of training samples, demonstrating the higher sparsity of the LPSVM model. Our experimental results, along with [66], shows that LPSVM achieves classification performance very similar to QPSVM with lower inference cost due to the deployment of fewer KCVs. Its LP formulation further reduces computational complexity for certain stochastic optimization settings on top of giving another advantage that the kernel function, $K(\cdot,\cdot)$, is not required to satisfy Mercer's conditions [3], providing greater flexibility in kernel selection.

3.3.3. SVM With Multiple Kernels: While the choice of kernel in SVM methods critically impacts performance, MKL learns an optimal kernel as a convex combination of base kernels:

$$K_\mu(x,x') = \sum_{m=1}^{M} \mu_m K_m(x,x'), \quad \mu_m \geq 0, \quad \sum_{m=1}^{M} \mu_m = 1.$$

where the objective is to jointly optimize classifier parameters and kernel weights μ for better adaptivity and interpretability, hence for significantly improving the classification performance, especially in complex datasets.

QPSVM With Multiple Kernels: Here we discuss about the standard MKL-SVM, that is QP-Based Multiple Kernel SVM with Optimized Convex Kernel Coefficients and Shared dual variable, α

Primal Formulation: Given a training set of data $\{(x_i, y_i)\}_{i=1}^N$, $y_i \in \{-1,+1\}$, SVM primal problem with multiple kernel K_μ is [67, 68]

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|_{\mathcal{H}_\mu}^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(\langle w, \phi_\mu(x_i)\rangle + b) \geq 1-\xi_i, \quad \xi_i \geq 0, \quad i = 1,\ldots,N, \end{aligned}$$

where ϕ_μ is the feature map for K_μ.

Since

$$K_\mu = \sum_{m=1}^{M} \mu_m K_m,$$

the RKHS norm decomposes as

$$\|w\|_{\mathcal{H}_\mu}^2 = \sum_{m=1}^{M} \frac{1}{\mu_m}\|w_m\|_{\mathcal{H}_m}^2,$$

with $w = \sum_m w_m$, $w_m \in \mathcal{H}_m$.

Thus the primal MKL problem becomes

$$\begin{aligned} \min_{\substack{w_m \in \mathcal{H}_m, b, \xi, \\ \mu \in \Delta}} \quad & \frac{1}{2}\sum_{m=1}^{M}\frac{1}{\mu_m}\|w_m\|_{\mathcal{H}_m}^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i\left(\sum_{m=1}^{M}\langle w_m, \phi_m(x_i)\rangle + b\right) \geq 1-\xi_i, \quad \xi_i \geq 0, \\ & \mu_m \geq 0, \quad \sum_{m=1}^{M}\mu_m = 1, \end{aligned}$$

where Δ is the simplex.

Now using the representer theorem and the shared dual variable for all base kernels α, we can write

$$w_m = \sum_{i=1}^{N} \alpha_i y_i \phi_m(x_i),$$

with $\alpha \in \mathbb{R}^N$ shared across kernels.

Define kernel matrices K_m with entries $[K_m]_{ij} = K_m(x_i, x_j)$ and

$$K_\mu = \sum_{m=1}^{M} \mu_m K_m.$$

Dual Formulation: The dual problem becomes

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K_\mu(x_i, x_j) \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0. \end{aligned}$$

where the dual variable α and the co-efficient variable μ can be found by implementing some efficient optimization algorithms.

LPSVM With Multiple Kernels: As multiple kernel SVM demands more kernel computation than single kernel based one and the LPSVM produces less number of SVs than QPSVM, we now study multi kernel based LPSVM for saving computational load while having optimal classification performance.

Vapnik's LP-Based Multiple Kernel SVM with Convex Kernel Coefficients μ and Shared Expansion Variable β: With the convex combination of base kernels with expansion coefficients μ_i as QPSVM, in this variant [15, 69, 70], the decision function becomes:

$$f(x) = \sum_{i=1}^{N} \beta_i y_i \left(\sum_{m=1}^{M} \mu_m K_m(x_i, x) \right) + b \tag{15}$$

This formulation is inspired by Vapnik's LP-based SVM but generalized to allow **learned convex combinations** of multiple kernels, while keeping the primal optimization formulation and shared support vector coefficients.

Primal Optimization Problem:

$$\min_{\beta, \mu, \xi} \quad \sum_{i=1}^{N} \beta_i + C \sum_{i=1}^{N} \xi_i + \lambda \Omega(\mu) \tag{16}$$

$$\text{s.t.} \quad y_i \Big[\sum_{j=1}^{N} \beta_j y_j \left(\sum_{m=1}^{M} \mu_m K_m(x_j, x_i) \right) + b \Big] \ge 1 - \xi_i, \quad \forall i \tag{17}$$

$$\beta_i \ge 0, \ \xi_i \ge 0, \ \forall i \tag{18}$$

$$\mu_m \ge 0, \ \sum_{m=1}^{M} \mu_m = 1 \tag{19}$$

where $\Omega(\mu)$ is convex regularizer on kernel weights and $\lambda > 0$.
Holding representer theorem, this convex problem ensures the solution to lie in the kernel span and leverages shared support vectors across all kernels assisting controlled sparsity in both support vectors and kernel weights depending on regularizers. However, handling with such problem may not go easy.

4. Our Method: Speed-Accuracy Boosted SVM (SABSVM)

In this section we describe our method by dividing it into few main parts as below.

4.1. Optimized Composite Kernel SVM

4.1.1. Kernel Functional Set up: While in many applications, finding kernel measure by combining kernels enhances generalization performance of a machine, a full MKL is computationally expensive even if it is with the sparser LP based SVM. Hence, a pragmatic alternative is to use an optimized convex combination of two kernels by using $M = 2$ in (19):

$$K(x, x') = \mu_1 \cdot K_1(x, x') + \mu_2 \cdot K_2(x, x')$$

where K_1, K_2 are two different kernels with $\mu_1, \mu_2 \in [0, 1] \mid \mu_1 + \mu_2 = 1$ that controls the kernel mixture and is selected via optimization.
This efficient approach avoids joint kernel-SVM optimization (as (16) - (19)) and works well with linear programming SVM (LPSVM), promoting sparsity and scalability [3] as optimum of these μ parameters could be found in a comparatively easier way, for example, by cross-validation process. Thus the problem becomes as

$$\min_{\boldsymbol{\beta}, \boldsymbol{\xi}} \sum_{i=1}^{N} \beta_i + C \sum_{i=1}^{N} \xi_i \tag{20}$$

$$\text{s.t. } y_i \left(\sum_{j=1}^{N} \beta_j y_j K(x_j, x_i) + b \right) \geq 1 - \xi_i \quad \forall i \tag{21}$$

$$\beta_i \geq 0, \quad \xi_i \geq 0 \tag{22}$$

where: $K(x_j, x_i) = \mu_1 K_1(x_j, x_i) + \mu_2 K_2(x_j, x_i)$ with $\mu_1, \mu_2 \in [0, 1] \mid \mu_1 + \mu_2 = 1$.
This is still a linear program with shared β, and very efficient to solve.

4.1.2. Productive Selection of Kernels: This is one of the most crucial parts in kernel induced learning. To obtain better classification performance and improved generalization ability of the SVM, it is vital to use a kernel or similarity function that captures different aspects of similarity between data points. While the most reputed, Gaussian (RBF) kernel effectively measures local similarity based on Euclidean distance, it may not fully exploit variations along individual feature dimensions, especially in high-dimensional data [50, 71, 72]. To address this, it is better to incorporate a complementary component based on the Manhattan distance, which emphasizes feature-wise differences and adds robustness (with reducing linear dependence of kernels) to the kernel while being computationally less expensive[73], less noise-sensitive[74], less feature-scale sensitive[75]. One more flexibility with this similarity measuring component is that having the 1st kernel, RBF as a positive definite kernel, we do not need to be strict on this to be a valid positive (semi) definite kernel [57] while we are using LP based SVM that can handle well a non-PD kernel[47]. Moreover, Haasdonk showed that SVMs with indefinite kernels remain optimal under convex hull constraints in pseudo-Euclidean spaces, rather than margin maximization in Hilbert spaces [72]. So, we select here the L_1 mapper non-PSD kernel, DSF[50], which has these discussed properties and has also shown higher classification performance on different datasets while both of these similarity mappers could

be regularized by tuning their parameters. Thus, the mathematical expression of our base kernels are

$$K_1(x_j \cdot x_i) = \exp(-\frac{\|x_j - x_i\|^2}{2\sigma_1^2}); \;\; K_2(x_j \cdot x_i) = \frac{1}{\frac{\sum_{q=1}^{d} |x_{jq} - x_{iq}|}{\sigma_2} + 1} \tag{23}$$

and hence our composite kernel takes the following form:

$$K(x_j \cdot x_i) = \mu_1 K_1(x_j \cdot x_i) + \mu_2 K_2(x_j \cdot x_i) = \mu_1 \exp(-\frac{\|x_j - x_i\|^2}{2\sigma_1^2}) + \mu_2 \frac{1}{\frac{\sum_{q=1}^{d} |x_{jq} - x_{iq}|}{\sigma_2} + 1} \tag{24}$$

By combining these two functions, we obtain a new similarity mapper that balances local similarity and feature-wise robustness, resulting in improved generalization and classification performance.

4.2. Functional Space Transformation

In 4.1 above we have modeled a highly efficient composite kernel SVM. However, further efficiency is really necessary in many real-time applications. So, basing on it, we design a detector that is developed by transforming the functional approximation space into its optimum by evolving in two stages and achieves much higher efficiency, which is described next and led by some most related definitions and description of symbols.

Definition 3 (RKHS [76])**.** A Hilbert space $\mathcal{H}_K$ of functions $f : \mathcal{X} \to \mathbb{R}$ is an RKHS if there exists a PSD kernel K such that

$$f(x) = \langle f, K(x, \cdot) \rangle_{\mathcal{H}_K}$$

Definition 4 (RKKS [49])**.** A Kreĭn space $\mathcal{K}$ of functions $f : \mathcal{X} \to \mathbb{R}$ is called a Reproducing Kernel Kreĭn Space (RKKS) if there exists a symmetric kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (not necessarily positive semidefinite) such that

$$f(x) = [f, K(x, \cdot)]_{\mathcal{K}}, \qquad \forall f \in \mathcal{K}, \; \forall x \in \mathcal{X},$$

where $[\cdot, \cdot]_{\mathcal{K}}$ denotes the indefinite inner product of the Kreĭn space $\mathcal{K}$.

RKKS generalizes RKHS to handle indefinite kernels, enabling learning in pseudo-Euclidean spaces [9].

Definition 5 (Space of Convex Combination of Kernels)**.** Given kernels K_1 (PSD) and K_2 (possibly non-PSD), define

$$K_\mu = \mu K_1 + (1 - \mu) K_2, \quad \mu \in [0, 1].$$

If K_μ is indefinite, it corresponds to a RKKS.

Explanation of Symbols:

- $\mathcal{H}_1$: RKHS associated with positive-definite kernel, K_1.
- $\mathcal{H}_2$: Reproducing Kernel Kreĭn Space (RKKS) associated with the non-PSD similarity kernel, $K_2 \not\succeq 0$.
- $\mathcal{F}_{\text{sim}}$: data-induced similarity space, $\mathcal{H}_1 \oplus \mathcal{H}_2$, which may not always be a valid RKHS.
- f: first-stage discriminant or function in the empirical similarity space, $\mathcal{F}_{\text{sim}}$.
- $C_{\mathcal{S}_1}$: Subspace compression operator mapping from $\mathcal{F}_{\text{sim}}$ into RKHS subspace $\mathcal{H}_{1,S_1} \subset \mathcal{H}_1$.
- $\tilde{f}$: second-stage discriminant (compressed function)
- f^*: ideal or target discriminant function (e.g., Bayes-optimal decision function).

Formal Setup: With the given training set, $\{(x_i, y_i)\}_{i=1}^N \subset \mathcal{X} \times \{-1, 1\}$, we consider:

- A valid PD kernel K_1 (RBF), inducing RKHS $\mathcal{H}_1$.
- A strong similarity function (L_1 mapper) $K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is symmetric and bounded but not necessarily PSD.

Two-Stage LPSVM For Higher Efficiency: In 4.1 above, we have designed a composite-kernel SVM for higher classification accuracy with lower classification cost. Here, we introduce one more stage to further reduce the number of KCVs of the final machine by filtering while keeping possibly the same generalization ability. As we are entering into a two-staged algorithm, we will have to find suitable values of much more parameters. Thus to reduce model complexity and being inspired by the fact that usually uniformly-weighted MKL performs better than sparse MKL[77], we start by setting $\mu_1 = \mu_2 = 0.5$ in 4.1 and complete the following stages.

- **Stage 1 (Hybrid Similarity Approximation):**
  Train an LP based SVM on full training set using a composite kernel generated by a convex combination of a positive definite (RBF) kernel K_1 and an L_1 mapper (non-PSD kernel), K_2 to form a high-capacity decision function in a rich space and extract support vectors as the discriminative basis points of the learned function. That is, here we use the following convex mixture (composite) kernel:

$$K_\mu(x, x') := 0.5K_1(x, x') + 0.5K_2(x, x')$$

The matrix $[K_\mu(x_i, x_j)]$ may not be PSD due to the contribution from K_2 but in practice the LP based SVM formulation still yields effective solutions due to the linear nature of its constraints and formulation.
We use a soft-margin LPSVM with this composite kernel K_μ where the optimization problem is:

$$\begin{aligned} \min_{\beta_i, b, \xi_i} \quad & \sum_{i=1}^{N} \beta_i + C \sum_{i=1}^{N} \xi_i \\ \text{subject to:} \quad & y_i \left(\sum_{j=1}^{N} \beta_j y_j K_\mu(x_j, x_i) + b \right) \geq 1 - \xi_i \\ & \beta_i \geq 0, \quad \xi_i \geq 0; \ i = 1, ..., N \end{aligned}$$

Theorem 2 (Conditional Representer Theorem [6, 49])**.**
(a) If K_μ is PSD, the LPSVM solution is

$$f(x) = \sum_{i=1}^{N} \alpha_i K_\mu(x_i, x) + b.$$

(b) If K_μ is indefinite, the solution exists in the corresponding RKKS and can be expressed as a finite linear combination of kernel evaluations, but the RKHS norm does not exist.

Proof Sketch. (a) Classical representer theorem applies in RKHS. (b) By RKKS theory, every solution of a regularized functional in a Kreĭn space has a kernel expansion [49]. □

Whether the kernel K_μ is definite or indefinite, from this theorem, in both cases, the solution lies in the span of the kernel evaluations:

$$f(x) = \sum_{i=1}^{N} \beta_i y_i K_\mu(x_i, x) + b$$

and hence the decision function learned in Stage 1 becomes

$$f(x) = \sum_{i=1}^{N} \beta_i y_i \left(0.5K_1(x_i, x) + 0.5K_2(x_i, x)\right) + b \tag{25}$$

While this function may not reside in a strict RKHS (but reside in RKKS) due to the non-PSD nature of K_2, it can be interpreted as a *generalized approximation* in a semi-inner-product space, or more practically, as a data-adaptive similarity embedding enabling tight approximation of target function, f^*. We define the induced approximation space as:

$$\mathcal{A}_1 := \text{span}\{K_1(x_i, \cdot), K_2(x_i, \cdot)\}_{i=1}^{N}$$

This is a valid functional approximation space in which we construct discriminants using empirical similarity structure

- **Stage 2 (Kreĭn-Hilbert subspace compression):**
  For compactness and generalization with sparsity, train a full LP based SVM with a single PD kernel (K_1 (RBF)) using only the data points that are found as support vectors in the first stage.

Let $\mathcal{S}_1 = \{x_i : \beta_i > 0\}$ be the obtained support vectors after solving Stage 1. We now train a new soft-margin LPSVM using only these SVs and a positive definite kernel, K_1 (RBF) to form a compact and efficient model by solving the following optimization problem

$$\begin{aligned} \min_{\lambda_i, b', \zeta_i} \quad & \sum_{i \in Ind(\mathcal{S}_1)} \lambda_i + C' \sum_{i \in Ind(\mathcal{S}_1)} \zeta_i \\ \text{subject to:} \quad & y_i \left(\sum_{j \in Ind(\mathcal{S}_1)} \lambda_j y_j K_1(x_j, x_i) + b' \right) \geq 1 - \zeta_i \\ & \lambda_j \geq 0, \quad \zeta_i \geq 0 \end{aligned}$$

and construct the decision function:

$$\tilde{f}(x) = \sum_{i \in Ind(\mathcal{S}_1)} \lambda_i y_i K_1(x_i, x) + b'$$

This defines a compact subspace:

$$\mathcal{A}_2 := \text{span}\{K_1(x_i, \cdot)\}_{x_i \in \mathcal{S}_1} \subset \mathcal{H}_1$$

and $\tilde{f} \in \mathcal{A}_2$ is a sparse RKHS-based approximation that retains essential structure learned via the richer first-stage similarity function.

Thus to summarize the **Resultant Functional Structure Transfer**, we can say that although the first-stage function $f \in \mathcal{F}_{\text{sim}}$ or $\mathcal{H}_1 \oplus \mathcal{H}_2$, which may not always be an RKHS in the strict sense (since $K_2 \not\succeq 0$) but in RKKS, second stage can be regarded as a *subspace compression* or *structure transfer* from a data-induced similarity space (or RKKS) into the more stable and analyzable RKHS subspace, $\mathcal{H}_{1,\mathcal{S}_1} \subset \mathcal{H}_1$, where this mapping is done by the following compression operator

$$C_{\mathcal{S}_1} : \mathcal{F}_{\text{sim}} \to \mathcal{H}_{1,S_1}, \text{with } \tilde{f} = C_{\mathcal{S}_1}(f)$$

and a complete proof of the compression is given below:

Theorem 3 (Functional Subspace Compression)**.** The operator $\mathcal{C}_{S_1} : \mathcal{F}_{\text{sim}} \to \mathcal{H}_{1,\mathcal{S}_1}$ induces a strict dimensionality reduction and capacity compression on the hypothesis space.

Proof
From (25), we have the decision function of Stage 1, $f \in \mathcal{F}_{\text{sim}}$ as:

$$f(x) = \sum_{i=1}^{N} \beta_i y_i \left(0.5K_1(x_i, x) + 0.5K_2(x_i, x)\right) + b$$

whose functional expansion depends on $2N$ independent basis functions in the composite space $\mathcal{F}_{\text{sim}} = \mathcal{H}_1 \oplus \mathcal{H}_2$, that means, $\dim(\text{span}(f)) = 2N$. Then by solving the L_1-regularized LP here, we get a sparse support vector set $\mathcal{S}_1 = \{x_i : \beta_i > 0\}$ having cardinality $|\mathcal{S}_1| = M \leq N$. Now, the compression operator $\mathcal{C}_{\mathcal{S}_1}$ projects f onto

$\mathcal{H}_{1,\mathcal{S}_1} = \text{span}\{K_1(x_j,\cdot) \mid x_j \in \mathcal{S}_1\}$ and generates the compressed function:

$$\tilde{f}(x) = \mathcal{C}_{\mathcal{S}_1}(f)(x) = \sum_{j \in Ind(\mathcal{S}_1)} \lambda_j y_j K_1(x_j, x) + b'$$

The resulting function space poses a capacity that satisfies :

$$\dim(\mathcal{H}_{1,\mathcal{S}_1}) = \text{rank}(\mathbf{K}_{1,\mathcal{S}_1}) \leq M \leq N < 2N = \dim(\mathcal{F}_{\text{sim}})$$

where $\mathbf{K}_{1,\mathcal{S}_1} \in \mathbb{R}^{M\times M}$ represents the kernel matrix restricted to $\mathcal{S}_1$. Thus, $\mathcal{C}_{\mathcal{S}_1}$ induces a strict compression in the capacity of the hypothesis space. □

By this functional compression into a lower-dimensional compact subspace $\mathcal{H}_{1,\mathcal{S}_1}$, the overall mapping yields a sparse functional approximation of the discriminator as: $\tilde{f} \approx f \approx f^*$ having the interesting properties like improved sparsity and interpretability, well-defined RKHS norm, $\|\tilde{f}\|_{\mathcal{H}_1}$, generalization guarantees, and retained discriminative power (due to basis-set filtering). Proof of the well-posed property of this norm is given below:

Theorem 4 (Well-Posedness of Norm $\|\tilde{f}\|_{\mathcal{H}_1}$)**.** The function norm $\|\tilde{f}\|_{\mathcal{H}_1}$ in the compressed subspace $\mathcal{H}_{1,\mathcal{S}_1}$ is well-posed, finite, and strictly obeys the axioms of a Hilbert space norm.

Proof
From its formation, $K_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric positive-definite kernel. By Mercer's Theorem, K_1 uniquely results in a RKHS $\mathcal{H}_1$ endowed with an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_1}$ that satisfies the following reproducing property:

$$\langle K_1(x,\cdot), K_1(x',\cdot)\rangle_{\mathcal{H}_1} = K_1(x,x'), \quad \forall x, x' \in \mathcal{X}$$

Now, as $\mathcal{S}_1 \subset \mathcal{X}$ is a finite set with M points, any function $\tilde{f} \in \mathcal{H}_{1,\mathcal{S}_1}$ is a finite linear combination $\tilde{f} = \sum_{j\in Ind(\mathcal{S}_1)} c_j K_1(x_j,\cdot)$, where $c_j = \lambda_j y_j$ and the squared norm is computed explicitly via bilinearity:

$$\|\tilde{f}\|^2_{\mathcal{H}_1} = \left\langle \sum_{j\in Ind(\mathcal{S}_1)} c_j K_1(x_j,\cdot), \sum_{l\in Ind(\mathcal{S}_1)} c_l K_1(x_l,\cdot) \right\rangle_{\mathcal{H}_1} = \sum_{j\in Ind(\mathcal{S}_1)} \sum_{l\in Ind(\mathcal{S}_1)} c_j c_l K_1(x_j,x_l) = \mathbf{c}^T \mathbf{K}_{1,\mathcal{S}_1} \mathbf{c}$$

Because K_1 is positive-definite, the matrix $\mathbf{K}_{1,S_1} \in \mathbb{R}^{M\times M}$ satisfies $\mathbf{c}^T\mathbf{K}_{1,S_1}\mathbf{c} \geq 0$ for all $\mathbf{c} \in \mathbb{R}^M$, with equality holding iff $\mathbf{c} = \mathbf{0}$.

In contrast to the indefinite inner product in the initial Kreïn space $\mathcal{F}_{\text{sim}}$ (where $K_2 \not\geq 0$ permits negative or non-existent norms), $\|\tilde{f}\|_{\mathcal{H}_1}$ satisfies:

1. **Non-negativity:** $\|\tilde{f}\|_{\mathcal{H}_1} \geq 0$.
2. **Definiteness:** $\|\tilde{f}\|_{\mathcal{H}_1} = 0 \iff \tilde{f} = 0$.
3. **Triangle Inequality:** $\|\tilde{f} + \tilde{g}\|_{\mathcal{H}_1} \leq \|\tilde{f}\|_{\mathcal{H}_1} + \|\tilde{g}\|_{\mathcal{H}_1}, \quad \forall \tilde{f}, \tilde{g} \in \mathcal{H}_{1,S_1}$.

Hence, the compressed function norm $\|\tilde{f}\|_{\mathcal{H}_1}$ is mathematically well-posed and strictly bounded. □

The whole SABSVM method is described concisely in Algorithm 1.

5. Experimental Evaluation

5.1. Set-Up

This section presents the experimental setup used to evaluate the performance of the proposed SABSVM classifier. Experiments are conducted on ten benchmark binary labeled datasets obtained from Gunnar Rätsch's benchmark collection [78]. The datasets include banana, breast-cancer, diabetis, flare-solar, heart, ringnorm, thyroid, titanic,

Algorithm 1 SABSVM: Speed Accuracy Boosted SVM

Input: A training set $\{(x_i, y_i)\}_{i=1}^{N}$
Output: A discriminator $f_{SABSVM}(\cdot)$

1: **Select two pairs of hyperparameters** $(C^*, C_V^*, \sigma^*, \sigma_V^*) = \arg \max_{(C,C_V),(\sigma,\sigma_V)} CV_{Acc}(C, C_V, \sigma, \sigma_V)$, $CV_{Acc} = Validation\ Accuracy$; $(\mu_1 = 0.5 \mid \mu_1 + \mu_2 = 1)$

2: **Solve LP using composite mapper:**

$$\begin{aligned} \min_{\beta,\xi,b} \quad & \sum_{i=1}^{N} \beta_i + C^* \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & y_i \left(\sum_{j=1}^{N} \beta_j y_j \phi(x_j) \cdot \phi(x_i) + b \right) \geq 1 - \xi_i \\ & \beta_j \geq 0; \quad j = 1, 2, \ldots, N \\ & \xi_i \geq 0; \quad i = 1, 2, \ldots, N \end{aligned}$$

$$where,\ \phi(x_j) \cdot \phi(x_i) = \mu_1[\exp(-\frac{\|x_j - x_i\|^2}{2(\sigma^*)^2})] + \mu_2[\frac{1}{\frac{\sum_{q=1}^{d} |x_{jq} - x_{iq}|}{\sigma^*} + 1}];\ \mathbf{x}_i \in \mathbb{R}^d$$

3: **Extract:** $(\mathcal{S}_1 = \{(x_j, y_j) \mid \beta_j > 0)\};\ |\mathcal{S}_1| = M \leq N$

4: **Solve LP**:

$$\begin{aligned} \min_{\lambda,\zeta,b_V} \quad & \sum_{l=1}^{M} \lambda_l + {C_V}^* \sum_{l=1}^{M} \zeta_l \\ \text{s.t.} \quad & y_i \left(\sum_{l=1}^{M} \lambda_l y_l \phi(x_l) \cdot \phi(x_i) + b_V \right) \geq 1 - \zeta_i \\ & \lambda_l \geq 0; \quad l = 1, 2, \ldots, M \\ & \zeta_i \geq 0; \quad i = 1, 2, \ldots, M \end{aligned}$$

$$where,\ \phi(x_l) \cdot \phi(x_i) = \exp(-\frac{\|x_l - x_i\|^2}{2({\sigma_V}^*)^2});\ \{(x_l, y_l)\}_{l=1}^{M} = \mathcal{S}_1$$

5: **Extract KCV:** $\mathcal{S}_2 = \{(x_m, y_m) \mid \lambda_m > 0\}_{m=1}^{P}$ & ${b_V}^*;\ \mathcal{S}_2 \subseteq \mathcal{S}_1$

6: $w_{\text{SABSVM}} \leftarrow \sum_{m=1}^{P} \lambda_m y_m \phi(x_m)$; $b_{\text{SABSVM}} \leftarrow {b_V}^*$

7: **Return;** $f_{SABSVM}(\cdot) = w_{SABSVM} \cdot \phi(\cdot) + b_{SABSVM}$

Table 1. The statistics of benchmark datasets [78] used in our experiments.

dataset	banana	breast-cancer	diabetis	flare-solar	heart
# dimensions	2	9	8	9	13
# training patterns	400	200	468	666	170
# test patterns	4,900	77	300	400	100
dataset	ringnorm	thyroid	titanic	twonorm	waveform
# dimensions	20	5	3	20	21
# training patterns	400	140	150	400	400
# test patterns	7,000	75	2,051	7,000	4,600

twonorm, and waveform. Each dataset contains 100 predefined train-test splits with varying numbers of training and testing samples, as summarized in Table 1. All reported results correspond to the average performance over these 100 splits, and the best result in each table is highlighted in bold. The proposed SABSVM is compared with several representative baseline and state-of-the-art (SOTA) classifiers from the categories of Single Kernel Learning (SKL), Sparse Single Kernel Learning (SSKL), Multiple Kernel Learning (MKL), and Sparse Multiple Kernel Learning (SMKL).

Single Kernel Learning (SKL)

- **LPSVM** [3]: Vapnik's Linear Programming Support Vector Machine implemented separately with two kernel functions, namely the Gaussian Radial Basis Function (RBF) and the Distance Similarity Function (DSF) [50].
- **QPSVM** [4]: Standard Quadratic Programming Support Vector Machine using the RBF kernel.

Sparse Single Kernel Learning (SSKL)

- **RLPSVM** [34]: A sparse LP-based SVM reducing the number of KCVs and lowering the prediction cost.
- **SpSVM-2** [28]: A SVM based sparse single-kernel learning algorithm designed to improve classification efficiency through greedy optimization.
- **SOSVM** [35]: A SVM oriented sparse single-kernel learning algorithm modeled for efficient classification through sequential QP and LP.

Multiple Kernel Learning (MKL)

- **NGMKL3** [18]: A neural-network-based multiple kernel learning framework that combines multiple kernels through neural processing units.
- **LapMKL** [19]: A multiple kernel learning algorithm based on graph Laplacian regularization.
- **2LMKL**$^{\inf}$ [20]: Two-Layer Multiple Kernel Learning method to learn further kernelized machines by combining multiple kernels in a multi-layer structure.

Sparse Multiple Kernel Learning (SMKL)

- **SMKC2 (4%)** [46]: A sparse MKL algorithm that employs gradient-based optimization under a predefined sparsity constraint based on the ratio between the number of support vectors obtained by SK-SVM and the number of expansion vectors used in Sparse MKL.
- **MLLKM (Gaussian), MLLKM (Square), MLLKM (Component Gaussian), MLLKM (Component Square), and LLSVM** [79]: A family of sparse multiple kernel learning approaches that utilize linear, nonlinear, and locally linear kernel combinations with different formulations.

Hyperparameters are selected using a modified five-fold cross-validation procedure. In the first stage, 4 fold of the training data are used to train the weighted Bi-kernel LPSVM for each candidate pair of regularization and kernel parameters C, σ. The resulting Kernel Computing Vectors (KCVs) are then used as the training data for the single kernel LPSVM in the second-stage, where another set of hyperparameters C_V, σ_V is evaluated. The discriminator produced by the second stage is regarded as the decision function of SABSVM which is subsequently used to classify the remaining fold. This procedure is repeated for all five folds, and the hyperparameter combination that achieves the highest average 4D cross-validation accuracy is selected as the optimal parameter set $C^*, \sigma^*, C_V^*, \sigma_V^*$. The search ranges for the hyperparameters are taken as, $\{C, \sigma, C_V, \sigma_V\} \in \{(2^{-2}, 2^{-1}, \ldots, 2^8), (2^{-2}, 2^{-1}, \ldots, 2^7), (2^{-1}, 2^0, \ldots, 2^6), (2^0, 2^1 \ldots, 2^8)\}$. All are implemented in MATLAB2020a where *linprog* is used to solve the optimization problem under the same experimental environment. The results are presented next.

5.2. Proposed Novel Metric

Machine's Error-Cost Loss (MECL): To have a machine's real time productivity, its high capability is mandatory on both classification accuracy and speed simultaneously. Hence, a high-performing but heavy cost-effective computing model is to be designed. To check such, we introduce a new metric, *MECL* to observe the machine's ability to give possibly least error with minimum kernel execution during prediction. This estimates its prediction error coupled computational load and is mathematically defined (for kernel executed machine) as below:

$$MECL = KEPPP \times Test\ Error\ Rates = \#KCVs \times \#Base\ Kernels \times Test\ Error\ Rates \tag{26}$$

where KEPPP = Kernel Execution Per Pattern Prediction.
Smaller MECL (with a reasonable TeEr rate) is always expected, as it indicates that the classifier achieves lower prediction error while requiring fewer KCVs during inference.

5.3. Result Analysis

5.3.1. Comparison of Results

SABSVM with SKL & SSKL: While comparing our method with SKL, it is seen from Table 2, SABSVM offers an overall average test error of 16.74%, which is slightly higher than LPSVM-RBF (15.48%) and QPSVM-RBF (15.71%) while using about 77% fewer KCVs than LPSVM-RBF & about 95% fewer KCVs than QPSVM-RBF which directly leads to boost up the prediction speed. Further, it reduces MECL by 88.5% & 96.7% compared to LPSVM-RBF and QPSVM-RBF respectively. A separate comparison is made for LPSVM-DSF shows a modest increase in test error yields a substantial reduction in prediction cost. However, SABSVM requires only 10.31 KCVs, compared with 96.16 for LPSVM-DSF which reduces MECL from 1604.43 to 111.05, corresponding to 93% lower computational cost. This sacrifices of a small amount of classification accuracy and dramatically reducing computational cost clearly demonstrates the achievement of an optimum balance between prediction accuracy and testing cost.

Table 2. Comparison of TeEr(%), KCV and MECL among SABSVM and SKL Machines

	SABSVM (ours)			**LPSVM RBF**			**QPSVM RBF**			**LPSVM DSF**		
dataset	TeEr(%)	#KCV	MECL	TeEr(%)	#KCV	MECL	TeEr(%)	#KCV	MECL	TeEr(%)	#KCV	MECL
banana	14.67	15.37	225.50	10.75	**15.08**	**162.11**	**10.61**	102.26	1084.98	*NR	*NR	*NR
breast-cancer	27.30	**3.60**	**98.28**	26.05	18.89	492.08	28.53	200.00	5706.00	**25.47**	27.10	690.24
diabetis	**23.35**	15.93	371.97	23.40	**12.58**	**294.37**	23.28	263.22	6127.76	*NR	*NR	*NR
flare-solar	36.67	**5.99**	**219.67**	32.56	251.66	8194.05	32.41	609.98	19769.45	**32.23**	157.38	5072.36
heart	**16.27**	**4.09**	**66.54**	17.44	21.94	382.63	16.60	68.23	1132.62	*NR	*NR	*NR
ringnorm	**1.74**	**15.83**	**27.57**	1.73	15.99	27.66	2.23	77.00	171.71	*NR	*NR	*NR
thyroid	5.15	**7.51**	**38.65**	5.09	8.97	45.66	5.20	43.51	226.25	**3.55**	77.09	273.67
titanic	26.64	**1.76**	**46.88**	22.91	83.91	1922.38	**22.68**	148.50	3367.98	*NR	*NR	*NR
twonorm	3.63	**24.15**	**87.59**	3.71	32.00	118.72	**2.42**	299.18	724.02	3.10	123.05	381.45
waveform	11.95	**15.64**	**186.83**	**11.18**	20.81	232.66	13.15	228.33	3002.54	*NR	*NR	*NR
Overall Average	16.74	**10.99**	**136.95**	**15.48**	48.18	1187.23	15.71	204.02	4131.33			
Average w.r.t reported data in LPSVM-DSF	18.19	**10.31**	**111.05**	16.85	77.88	2212.63	17.14	288.17	6606.43	**16.09**	96.16	1604.43

*NR: Not Reported

Table 3 compares the proposed SABSVM with Sparse Single Kernel Learning (SSKL) machines. In terms of computational efficiency, SABSVM achieves the lowest overall average MECL of 136.95, representing an 82% reduction compared with RLPSVM and a 12% reduction compared with SpSVM-2 while offering very close test error rate. Specifically, SABSVM requires only about one-third of the kernel computing vectors (KCVs) needed by

RLPSVM, while its average KCV differs from the training-costly SpSVM-2 by only 0.41, indicating that SABSVM is almost as sparse as SpSVM-2. Furthermore, when compared with SOSVM, SABSVM uses 41% fewer KCVs, resulting in 51% lower MECL. Overall, SABSVM consistently achieves one of the lowest MECL values across nearly all ten benchmark datasets while maintaining competitive classification performance.

Table 3. Comparison of TeEr(%), KCV and MECL among SABSVM and SSKL Machines

	SABSVM (ours)			**RLPSVM**			**SpSVM-2**			**SOSVM**		
dataset	TeEr(%)	#KCV	MECL	TeEr(%)	#KCV	MECL	TeEr(%)	#KCV	MECL	TeEr(%)	#KCV	MECL
banana	14.67	15.37	225.50	11.38	**13.47**	**153.29**	**10.87**	17.30	188.05	10.91	15.23	166.16
breast-cancer	**27.30**	**3.60**	**98.28**	28.21	15.46	436.13	29.22	12.10	353.56	*NR	*NR	*NR
diabetis	**23.35**	15.93	371.97	23.67	**6.69**	**158.35**	23.47	13.80	323.89	23.43	12.56	294.28
flare-solar	36.67	**5.99**	**219.67**	**32.44**	140.76	4566.25	33.90	8.40	284.76	*NR	*NR	*NR
heart	16.27	**4.09**	**66.54**	15.51	8.07	125.17	**15.50**	4.30	66.65	15.55	10.60	164.83
ringnorm	**1.74**	15.83	27.57	1.73	15.99	27.66	1.97	**12.90**	**25.41**	1.81	40.26	72.87
thyroid	**5.15**	7.51	**38.65**	5.09	8.97	45.66	5.47	10.60	57.98	7.77	**4.34**	**33.72**
titanic	26.64	**1.76**	**46.88**	22.91	82.22	1883.66	**22.68**	3.30	74.84	23.34	48.48	1131.52
twonorm	3.63	24.15	87.59	3.70	19.27	71.30	**2.96**	**8.70**	**25.75**	3.38	18.50	62.53
waveform	11.95	15.64	186.83	**10.44**	20.64	215.48	10.66	**14.40**	**153.50**	11.02	21.06	232.08
Overall Average	16.74	10.99	**136.95**	**15.51**	33.15	768.29	15.67	**10.58**	155.44			
Average w.r.t reported data in SOSVM	12.92	12.53	131.44	11.80	21.91	335.07	**11.70**	**10.66**	**114.51**	12.15	21.38	269.75

SABSVM with MKL & SMKL: Tables 4–6 compare the proposed SABSVM with several state-of-the-art MKL and SMKL methods. Since some competing MKL methods do not report the exact KCVs, their computational efficiency is analyzed based on theoretical formulations of *Classification Cost (CC)* of any machine involving *Kernel Execution Per Pattern Prediction (KEPPP)*, with $KEPPP = \#BaseKernel \times \#KCV$ that refers the number of kernel execution required to classify a single pattern. As SABSVM employs only one kernel, it $CC \propto KEPPP = \#KCV$.Moreover, $\#KCV \ll \#Training_Pattern$, for every benchmark dataset indicates that SABSVM depends only on a very small subset of training samples. Unlike SABSVM, NGMKL3 use every training samples in prediction ($\#KCV = \#Training_Pattern$) and additional layer-processing increases computational cost. Since the number of KCVs are not reported in LapMKL and 2LMKL$^{\text{inf}}$, and each prediction in 2LMKL$^{\text{inf}}$ requires an additional kernel evaluation, these methods inherently incur higher computational complexity compared to SABSVM. So, SABSVM provides only one kernel evaluation per selected KCV, avoids any layer-processing overhead, yields a highly sparse decision function, and requires substantially fewer kernel evaluations.

Table 5 shows that SABSVM reduces the average MECL by 55% compared with SMKC2 (304.01 to 136.95) and achieves the smaller CC indicating a faster prediction capability. In Table 6, Heart and Diabetis datasets are selected because complete statistics are available only for these two datasets. As per this table, SABSVM consistently outperforms other machines in all aspects. For Heart dataset, SABSVM achieves the lowest test error rate while reducing upto 87% in CC and 86% in MECL with respect to other SMKL Machines. Similarly, on the Diabetes dataset, SABSVM again attains the lowest test error and reduces CC and MECL upto 34% and 37%, respectively. These results demonstrate that SABSVM effectively combines superior classification performance with substantially lower prediction cost.

5.3.2. SABSVM on Large Data: To check the scalability of SABSVM on large data, we try to observe its continuous and monotonic sparsity in parallel with the increment of datasize. For this purpose, #training patterns of the *banana* dataset was gradually increased from x to $12x$, where $x = 400$ and the sparsity is measured as, $\text{Sparsity}(x_\text{Machine}) = \frac{\#\text{training patterns}}{\#\text{KCV}(x_\text{Machine})}$, whereas a higher sparsity indicates that a comparatively fewer KCVs are required. From Figure 6a, it is clear that sparsity of QPSVM remains almost unchanged with the increment of data size but it increases slowly in case of LPSVM after a little zigzag in the beginning while it rises very fast

Table 4. Comparison of TeEr(%) & CC among SABSVM and MKL Machines

	dataset	banana	breast-cancer	diabetis	flare-solar	heart	ringnorm	thyroid	titanic	twonorm	waveform
Machines	#training patterns	400	200	468	666	170	400	140	150	400	400
SABSVM (ours)	TeEr(%)	14.67	27.30	**23.35**	36.67	16.27	1.74	5.15	26.64	3.63	11.95
	#Base Kernel	1									
	#KCV	15.37	3.60	15.93	5.99	4.09	15.83	7.51	1.76	24.15	15.64
	Classification Cost (CC)	$CC \propto KEPPP$ $where,\ KEPPP = \#Base\ Kernel \times \#KCV,$ $and\ \#KCV \ll \#training\ pattern$									
NGMKL3	TeEr(%)	11.06	26.04	24.29	35.79	*NR	1.57	**4.17**	22.41	**2.9**	**9.55**
	#Base Kernel	*NR									
	#KCV	$\#KCV = \#training\ pattern$									
	Classification Cost (CC)	$CC \propto KEPPP + Layer\ Processing\ Cost$ $where, KEPPP = \#Base\ Kernel \times \#KCV$									
LapMKL	TeEr(%)	12.86	30.5	29.35	35.5	**14.15**	1.77	4.6	**20.08**	*NR	14.12
	#Base Kernel	*NR									
	#KCV	*NR									
	Classification Cost (CC)	$CC \propto KEPPP$ $KEPPP = \#Base\ Kernel \times \#KCV$									
2LMKL^{inf}	TeEr(%)	**9.8**	**3.1**	23.4	**34.7**	16.4	**1.5**	5.2	22.2	*NR	9.6
	#Base Kernel	*NR									
	#KCV	*NR									
	Classification Cost (CC)	$CC \propto KEPPP$ $KEPPP = (\#Base\ Kernel + 1) \times \#KCV$									

*NR: Not Reported

Table 5. Comparison of TeEr(%), CC and MECL between SABSVM and SMKC2 (4%)

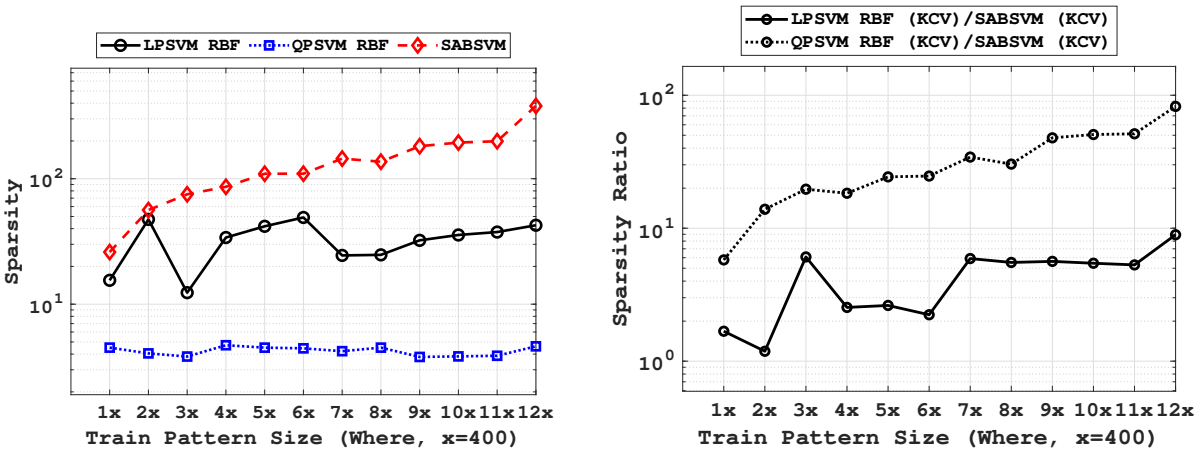
dataset	**SABSVM (ours)**			**SMKC2 (4%)**		
	TeEr(%)	CC=KEPPP	MECL	TeEr(%)	CC=KEPPP	MECL
banana	14.7	**15.4**	225.50	**12.2**	**15.4**	**187.88**
breast-cancer	27.3	**3.6**	**98.28**	**26**	11.5	299.00
diabetis	**23.4**	15.9	371.97	23.7	**9.3**	**220.41**
flare-solar	36.7	**6.0**	**219.67**	**33.2**	59.4	1972.08
heart	16.3	**4.1**	**66.54**	**15.9**	6.1	96.99
ringnorm	1.7	15.8	27.57	**1.4**	**6**	**8.40**
thyroid	**5.1**	7.5	38.65	9.5	**1**	**9.50**
titanic	26.6	**1.8**	**46.88**	**24.2**	4.6	111.32
twonorm	3.6	24.1	87.59	**2.9**	**3.9**	**11.31**
waveform	11.9	15.6	186.83	**10.1**	**12.2**	**123.22**
Overall Average	16.7	**11.0**	**136.95**	**15.9**	**12.9**	304.01

continuously in case SABSVM, which is more clear by observing the ratios of sparsity among them in Figure 6b. This gives a very optimistic message about higher efficiency of SABSVM on larger data.

5.3.3. Visual Evaluation: To have a visual comparison, decision boundaries have been plotted on 2D datasets: benchmark banana (Figure 1), and two synthetic [80], namely two-spiral (Figure 7a) and half-kernel (Fgiure 7b), are used to evaluate the discriminative capability of SABSVM and other machines. From the figures, it can seen that

Table 6. Comparison of TeEr(%), CC and MECL of Heart and Diabetis dataset among SABSVM and SMKL Machines

	dataset					
	Heart			Diabetis		
Machine	TeEr(%)	CC=KEPPP	MECL	TeEr(%)	CC=KEPPP	MECL
MLLKM (Gaussian)	18.40	26.00	478.40	24.70	24.00	592.80
MLLKM (Square)	18.60	27.10	504.06	25.10	28.10	705.31
MLLKM (Component Gaussian)	17.80	27.30	485.94	24.00	29.60	710.40
MLLKM (Component Square)	18.90	26.80	506.52	27.40	38.60	1057.64
LLSVM	19.60	32.00	627.20	25.80	32.00	825.60
SABSVM (ours)	**16.27**	**4.09**	**66.54**	**23.35**	**15.93**	**371.97**



(a) Data Size Vs Sparsity

(b) Data Size Vs Sparsity Ratio

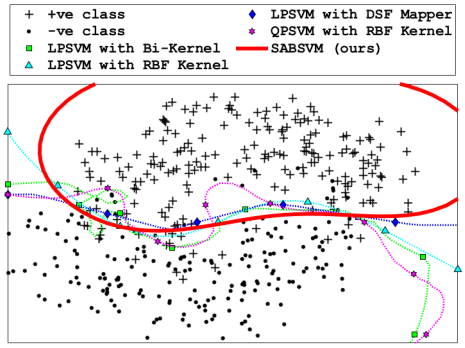
Figure 6. Sparsity with variational training pattern size

the proposed SABSVM gives training accuracy similar to other high accuracy complex machines while demanding much smaller number of KCVs.

6. Conclusion & Future Work

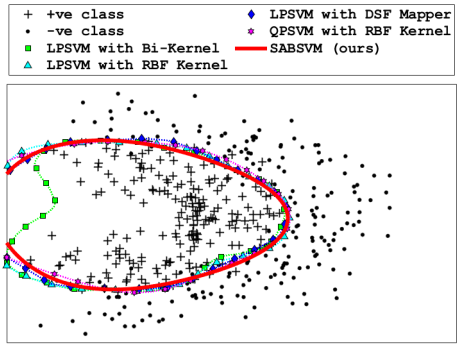
We have proposed a novel two-stage heavily compressed basis selection framework, which performs a kernel subspace compression using SVM, balancing expressiveness, sparsity, and efficiency that preserves the accuracy of a high quality MKL classifier while drastically reducing the number of support vector outperforming conventional single-stage SVMs and some MKL approaches.

The first stage is exclusively used to identify a compact and informative set of candidate basis vectors that capture the geometric structure of the decision boundary. Here, we combine a strong analytic mapper[50, 57] with a universal kernel following a linear combination of linearly independent and heterogeneous kernels (PD with non-PSD) within multi-kernel frameworks to generate a richer kernel space for structural representations, expressivity. This construction captures both smooth and domain-specific structural relationships, producing an informative and diverse support vector set that form a compact, expressive basis spanning the decision geometry learned by the multi-kernel model. It is also enriched with the support of LP that can efficiently handle non-PSD kernels [47] avoiding non-convexity introduced by indefinite similarity matrices and offers sparse, efficient and accurate SVM learning.



Machine	TrEr (%) (▼)	KCVs (▼)
SABSVM (ours)	9.75	5
LPSVM with Bi-kernel	7.75	30
LPSVM with RBF Kernel	10.00	6
LPSVM with DSF Kernel	10.25	8
QPSVM with RBF Kernel	8.00	122

(a) Two Spiral



Machine	TrEr (%) (▼)	KCVs (▼)
SABSVM (ours)	7.75	7
LPSVM with Bi-kernel	7.25	65
LPSVM with RBF Kernel	8.00	22
LPSVM with DSF Kernel	8.50	10
QPSVM with RBF Kernel	7.50	77

(b) Half Kernel

Figure 7. Decision boundaries on Synthetic data (400×2) from proposed SABSVM & some other SOTA machines

The second stage performs functional structure transfer mechanism between kernel spaces. Here, we treat the collected SV set in stage-1 as a compact training basis and train an RBF based LP SVM to learn more expressive and flexible decision boundaries, which generalize well across different patterns in the data. This projection onto a sparse RKHS basis acts as an information-preserving compression mechanism, yielding a substantially smaller set of effective support vectors while representing complex relationships and maintains almost the same classification accuracy.

From SLT's perspective, this method provides a bias-variance trade-off by slightly increasing the probability of Bias error due to changing kernel and restricted subspace while decreasing the probability of Variance due to lower model capacity and fewer parameters while the reduced variance from a smaller hypothesis space can optimally outweigh the approximation bias incurred. On the other hand, the whole method is modelled as convex as the reduced SVM retraining is a surrogate convex optimization problem because it approximates the original well-posed convex and complex MKL-SVM optimization by solving another simpler, well-posed convex problem on a restricted set of variables (the support vectors) and fixed kernel, trading some optimality for computational efficiency and tractability. Further, unlike many other prominent and high performing methods, we do not require any complex training by solving SDP, QCQP, SOCP, SILP.

Generally, MKL leads to a richer approximation space than SKL SVMs, especially when combining diverse kernels. However, the actual richness depends on the choice and weighting of kernels during learning and our method keeps MKL partly for higher accuracy while leaves it partly for cost reduction that leads us to take the responsibility to fill the gap in the literature: achieving significant reductions in SV count without compromising accuracy, where we move towards to attack the *extremely challenged task: increasing both classification accuracy and speed simultaneously*.

Moreover, to check a machine's practical productivity very closely, we have defined a new term *MECL* that can be directly measured and we have evaluated it on different machines.

Comparing with **accuracy focused machines**, SABSVM poses classification accuracy similar to high accuracy giving SKL machines and a nearly similar to the top accuracy giving highly complex MKL machines while using much lower number of KCVs than SKL machines. This gets further impressive when it pops up that the heavy cheap SABSVM gives *lower test error rate* than the highly complex and costly LapMKL in 4 out of 9 benchmark data sets as well as the mean of these 9. Probably, *overfitting* from LapMKL be a reason for this. Further, considering the joint error-cost loss, SABSVM gives significantly lower valued MECL compared to the complex SKL machines. But while it becomes analytically highly probable that SABSVM will give tremendously lower valued MECL from

the highly complex MKL machines, we cannot provide an exact numerical report here as the number of KCVs used by these cannot be found.

On contrary, comparing with **efficiency targeted machines**, apart from 5 LL(locally linear) sparse MKLs, SABSVM produces classification accuracy similar to the sparse SKL as well as sparse MKL machines while using similar KCVs except 33.1% of RLPSVM. This becomes really interesting when it reveals that SABSVM gives *lower test error rate* than all of the 5 sparse LL MKL machines in both of their 2 reported data sets (heart, diabetis). Additionally, regarding the joint error-cost loss, SABSVM gives considerably lower valued MECL compared to all sparse SKL as well as sparse MKL classifiers except the training-costly but test-efficient SpSVM-2. With this machine, the train-test efficient SABSVM has a neck to neck competition in classification accuracy, number of KCV while SABSVM gives a little smaller MECL value.

Besides the above interesting results, SABSVM showed increasing sparsity with respect to the increment of data size, which indicates its higher efficiency and usefulness on large data. Its plotted class-boundaries on 2D data also look quite smooth and promising.

Thus, on top of its higher accuracy, the compressed model exhibits large improvements in inference speed and memory usage, meaning both of its speed and accuracy are boosted (hence, SABSVM) enabling its deployment in resource-constrained settings. Moreover, as the method wraps around only the standard LP solvers and post-processes their outputs, it is practical for many real world applications.

However, with the above many advantages of the method, its one implicit limitation could be that the balance between approximation quality and model simplicity depends on how well the single kernel RKHS overlaps with the MKL approximation space and how informative the selected SVs are.

Following ideas are so far noted for future work directions: i)integrating adaptive kernel selection, ii)joint and composite body training of extreme states, iii) introducing error probability bounded learning.

REFERENCES

1. C. Cortes, and V. Vapnik, *Support-vector networks*, Machine Learning, vol. 20, no. 3, pp. 273–297, 1995. DOI: 10.1007/BF00994018
2. B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A Training Algorithm for Optimal Margin Classifiers*, in COLT, 1992.
3. V. N. Vapnik, *Statistical Learning Theory*, Wiley, 1998.
4. V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, NY, USA, 1995. DOI: 10.1007/978-1-4757-3264-1
5. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, *A comprehensive survey on support vector machine classification: Applications, challenges and trends*, Neurocomputing, vol. 408, pp. 189–215, 2020.
6. B. Schölkopf, and A. J. Smola, *Learning with Kernels*, MIT Press, 2002.
7. N. Cristianini, and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, UK, 2000. DOI: 10.1017/CBO9780511801389
8. C. J. C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121–167, 1998. DOI: 10.1023/A:1009715923555
9. 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. DOI: 10.3390/math12243935
10. L. Belkhiri-Brahmi, B. Brahmi, and M. O. Bib, *An Adaptive Support Method for Training SVM Problems*, Statistics, Optimization & Information Computing, vol. 16, no. 2, pp. 1447–1465, 2026. DOI: 10.19139/soic-2310-5070-2682
11. B. Sidaoui, and H. Abdelmoumene, *Kernel Functions for Support Vector Machines: A Survey*, in Proceedings of the 11th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1757–1762, 2025. DOI: 10.1109/CoDIT66093.2025.11321616
12. S. Poliyama, N. Achmad, and S. N. Abdussamad, *Comparison of Multiple Kernel Learning and Single Kernel Support Vector Machine for Public Opinion Classification*, Journal of Mathematics, Computations and Statistics, vol. 9, no. 1, pp. 48–62, 2026.
13. W. Xue, B. Xu, H. Wang, X. Zhu, J. Qin, G. Zhou, P. Yu, S. Li, and Y. Jin, *Development and validation of a machine learning model based on multiple kernel for predicting the recurrence risk of Budd-Chiari syndrome*, Frontiers in Physiology, vol. 16, p. 1589469, 2025.
14. M. Gönen, and E. Alpaydin, *Multiple Kernel Learning Algorithms: A Survey*, Journal of Machine Learning Research, vol. 12, no. 64, pp. 2211–2268, 2011.
15. A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, *SimpleMKL*, Journal of Machine Learning Research, pp. 2491–2521, 2008.

16. F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, *Multiple kernel learning, conic duality, and the SMO algorithm*, in Proceedings of the Twenty-First International Conference on Machine Learning (ICML), ICML '04, p. 6, 2004. DOI: 10.1145/1015330.1015424
17. S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, *Large Scale Multiple Kernel Learning*, Journal of Machine Learning Research, vol. 7, no. 57, pp. 1531–1565, 2006.
18. A. N. Ghanizadeh, K. Ghiasi-Shirazi, and R. Monsefi et al., *Neural Generalization of Multiple Kernel Learning*, Neural Processing Letters, vol. 56, p. 12, 2024. DOI: 10.1007/s11063-024-11516-0
19. Z. Hao, G. Yuan, X. Yang, and Z. Chen, *A primal method for multiple kernel learning*, Neural Computing and Applications, vol. 23, no. 3, pp. 975–987, 2013.
20. J. Zhuang, I. W. Tsang, and S. C. Hoi, *Two-layer multiple kernel learning*, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 909–917, 2011.
21. R. Karim, Z. Hossain, A. K. Kundu, and A. A. Ave, *EBSVM: A Sparsification Method From LP-Based SVM via Stochastic Functional ξ-Analysis*, Informatica, vol. 49, no. 28, 2025.
22. S. Zhou, *Sparse SVM for sufficient data reduction*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 9, pp. 5560–5571, 2021.
23. H. Wang, W. Li, Y. Shao, and Z. Hongwei, *Sparse and robust alternating direction method of multipliers for large-scale classification learning*, Neurocomputing, p. 130893, 2025.
24. H. Wang, G. Li, and Z. Wang, *Fast SVM classifier for large-scale classification problems*, Information Sciences, vol. 642, p. 119136, 2023.
25. S. Qu, R. De Leone, and M. Huang, *Two Novel Sparse Models for Support Vector Machines*, Symmetry, vol. 17, no. 11, p. 2004, 2025.
26. C. J. C. Burges, *Simplified support vector decision rules*, in ICML, pp. 71–77, 1996.
27. S. S. Keerthi, S. K. Shevade, and C. Bhattacharyya, *Improvements to Platt's SMO algorithm for SVM classifier design*, Neural Computation, vol. 13, pp. 637–649, 2002.
28. 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.
29. C. K. I. Williams, and M. Seeger, *Using the Nyström Method to Speed Up Kernel Machines*, NeurIPS, 2001.
30. A. Rahimi, and B. Recht, *Random Features for Large-Scale Kernel Machines*, NeurIPS, 2007.
31. S. Fine, and K. Scheinberg, *Efficient SVM Training Using Low-Rank Kernel Approximations*, Journal of Machine Learning Research, 2001.
32. 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, Lecture Notes in Computer Science, vol. 4713, pp. 395–404, Springer, 2007. DOI: 10.1007/978-3-540-74936-3_40
33. D. Cuadrillero Moles, *Kernel Feature Selection via Random Fourier Features*, arXiv, 2025.
34. R. Karim, and A. K. Kundu, *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.
35. 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 (IJACSA), vol. 9, no. 2, 2018. DOI: 10.14569/IJACSA.2018.090244
36. A. N. Alkhateeb, A. M. Algboory, Z. A. Hadied, O. E. Al-Saqal, and Z. Y. Algamal, *Enhancing Surface Water Potability Assessment through a ν-SVM Hybrid Statistical Learning Model*, Statistics, Optimization & Information Computing, vol. 15, no. 6, pp. 5481–5494, 2026. DOI: 10.19139/soic-2310-5070-3929
37. J. Huang, L. Luo, and X. Wang, *An Efficient Multiple Empirical Kernel Learning Algorithm with Data Distribution Estimation*, Electronics, vol. 14, no. 9, p. 1879, 2025.
38. A. J. Pinar, J. Rice, L. Hu, D. T. Anderson, and T. C. Havens, *Efficient multiple kernel classification using feature and decision level fusion*, IEEE Transactions on Fuzzy Systems, vol. 25, no. 6, pp. 1403–1416, 2016.
39. T. Suzuki, and R. Tomioka, *SpicyMKL: a fast algorithm for multiple kernel learning with thousands of kernels*, Machine Learning, vol. 85, no. 1, pp. 77–108, 2011.
40. Qi Fan and Zhe Wang and Daqi Gao and Dongdong Li, *MPEKDyL: Efficient multi-partial empirical kernel dynamic learning*, Knowledge-Based Systems, vol. 77, pp. 40–55, 2015.
41. C.-Y. Yeh, W.-P. Su, and S.-J. Lee, *An efficient multiple-kernel learning for pattern classification*, Expert Systems with Applications, vol. 40, no. 9, pp. 3491–3499, 2013.
42. C. Zhu, and D. Gao, *Improved multi-kernel classification machine with Nyström approximation technique*, Pattern Recognition, vol. 48, no. 4, pp. 1490–1509, 2015.
43. Z. Wang, C. Zhu, Z. Niu, D. Gao, and X. Feng, *Multi-kernel classification machine with reduced complexity*, Knowledge-Based Systems, vol. 65, pp. 83–95, 2014.
44. Q. Fan, Z. Wang, H. Zha, and S. Chen, *MREKLM: A fast multiple empirical kernel learning machine*, Pattern Recognition, vol. 61, pp. 197–209, 2017. DOI: 10.1016/j.patcog.2016.07.027
45. F. Aiolli, and M. Donini, *EasyMKL: A scalable multiple kernel learning algorithm*, Neurocomputing, vol. 169, pp. 215–224, 2015. DOI: 10.1016/j.neucom.2014.11.078
46. M. Hu, Y. Chen, and J. T. Y. Kwok, *Building sparse multiple-kernel SVM classifiers*, IEEE Transactions on Neural Networks, vol. 20, no. 5, pp. 827–839, 2009.
47. I. Alabdulmohsin, X. Gao, and X. Zhang, *Support Vector Machines with Indefinite Kernels*, in Proceedings of the Sixth Asian Conference on Machine Learning, pp. 32–47, 2015.

48. Jimenez Castaño, Cristian and Alvarez-Meza, A. and Cardenas-Pena, David and Orozco-Gutierrez, Alvaro and Erazo, Josiah, *Kreĭn Twin SVMs for Imbalanced Classification*, Pattern Recognition Letters, vol. 182, pp. 39–45, 2024. DOI: https://doi.org/10.1016/j.patrec.2024.03.017
49. C. S. Ong, X. Mary, S. Canu, and A. J. Smola, *Learning with non-positive kernels*, in Proceedings of the Twenty-First International Conference on Machine Learning (ICML), p. 81, 2004.
50. 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. DOI: 10.31449/inf.v47i8.4767
51. H. Xue et al., *Multiple indefinite kernel learning for feature selection*, Knowledge-Based Systems, vol. 191, p. 105272, 2020. DOI: 10.1016/j.knosys.2019.105272
52. I. Steinwart, *Sparseness of Support Vector Machines—Some Asymptotically Sharp Bounds*, in Advances in Neural Information Processing Systems 16, pp. 1069–1076, 2004.
53. 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. DOI: 10.1109/PGEC.1965.264137
54. 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.
55. 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. DOI: 10.3390/math12243935
56. 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, vol. 209, pp. 415–446, 1909. DOI: 10.1098/rsta.1909.0016
57. M.-F. Balcan, A. Blum, and N. Srebro, *A theory of learning with similarity functions*, Machine Learning, vol. 72, no. 1, pp. 89–112, 2008. DOI: 10.1007/s10994-008-5059-5
58. M.-F. Balcan, A. Blum, and N. Srebro, *Improved guarantees for learning via similarity functions*, Carnegie Mellon University, 2008.
59. 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.
60. S. Geman, E. Bienenstock, and R. Doursat, *Neural Networks and the Bias/Variance Dilemma*, Neural Computation, vol. 4, no. 1, pp. 1–58, 1992. DOI: 10.1162/neco.1992.4.1.1
61. D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 6th ed., John Wiley & Sons, 2021.
62. V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer, 1979.
63. V. and A. Chervonenkis, *A Note on One Class of Perceptrons*, Automation and Remote Control, vol. 25, 1964.
64. M. Awad, and R. Khanna, *Support Vector Machines for Classification*, in Efficient Learning Machines, Apress, Berkeley, CA, pp. 39–66, 2015. DOI: 10.1007/978-1-4302-5990-9_3
65. S. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441
66. 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. DOI: 10.1109/icmla.2009.52
67. Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, *Simple and efficient multiple kernel learning*, in Proceedings of the 27th International Conference on Machine Learning (ICML), pp. 1175–1182, 2010.
68. M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, *Lp-norm multiple kernel learning*, Journal of Machine Learning Research, vol. 12, pp. 953–997, 2011. DOI: 10.5555/1953048.2021054
69. F. Liu, L. Zhou, S. Chunhua, and J. Yin, *Multiple Kernel Learning in the Primal for Multi2011modal Alzheimer2019s Disease Classification*, IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 3, pp. 984–990, 2014. DOI: 10.1109/JBHI.2013.2285378
70. A. Argyriou, C. A. Micchelli, and M. Pontil, *Learning Convex Combinations of Continuously Parameterized Basic Kernels*, in Proceedings of the 18th Annual Conference on Learning Theory (COLT 2005), pp. 338–352, Springer, 2005. DOI: 10.1007/11503415_23
71. C. C. Aggarwal, A. Hinneburg, and D. A. Keim, *On the surprising behavior of distance metrics in high dimensional space*, in International Conference on Database Theory, Springer, pp. 420–434, 2001. DOI: 10.1007/3-540-44503-x_27
72. B. Haasdonk, *Feature space interpretation of SVMs with non-positive definite kernels*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 4, pp. 482–492, 2005. DOI: 10.1109/TPAMI.2005.69
73. D. R. Wilson, and T. R. Martinez, *Improved heterogeneous distance functions*, Journal of Artificial Intelligence Research, vol. 6, pp. 1–34, 1997.
74. V. Prasath et al., *Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier–A Review*, arXiv preprint arXiv:1708.04321, 2017. DOI: 10.1089/big.2018.0175
75. A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah, *A comparison study on similarity and dissimilarity measures in clustering continuous data*, PloS One, vol. 10, no. 12, p. e0144059, 2015.

DOI: 10.1371/journal.pone.0144059
76. N. Aronszajn, *Theory of Reproducing Kernels*, Transactions of the American Mathematical Society, vol. 68, pp. 337–404, 1950. DOI: 10.2307/1990404
77. R. Tomioka, and T. Suzuki, *Sparsity-accuracy trade-off in MKL*, arXiv: Machine Learning, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:88518305
78. G. Rätsch, *Benchmark datasets*, Re-packaged and available on GitHub. [Online]. Available: https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets.
79. D. Picard, *Multiple locally linear kernel machines*, arXiv preprint arXiv:2401.09629, 2024.
80. J. Kools, *6 functions for generating artificial datasets*, 2013. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generatingartificial-datasets.