Efficiency and Performance Analysis of a Sparse and Powerful Second Order SVM Based on LP and QP

Rezaul Karim, Amit Kumar Kundu
Department of Electrical and Electronic Engineering
Uttara University
Dhaka, Bangladesh

Abstract—**Productivity analysis is done on the new algorithm "Second Order Support Vector Machine (SOSVM)", which could be thought as an offshoot of the popular SVM and based on its conventional QP version as well as the LP one. Our main goal is to produce a machine which is: 1) sparse & efficient; 2) powerful (kernel based) but not overfitted; 3) easily realizable. Experiments on benchmark data shows that to classify a new pattern, the proposed machine, SOSVM requires samples up to as little as 2.7% of original data set or 4.8% of conventional QP SVM or 48.3% of Vapnik's LP SVM, which is already sparse. Despite this heavy test cost reduction, its classification accuracy is very similar to the most powerful QP SVM while being very simple to be produced. Moreover, two new terms called "Generalization Failure Rate (GFR)" and "Machine-Accuracy-Cost (MAC)" are defined to measure generalization-deficiency and accuracy-cost of a detector, respectively and used to compare such among different machines. Results show that our machine possesses GFR up to as little as 1.4% of the QP SVM or 1.5% of Vapnik's LP SVM and MAC up to as little as 2.6% of the QP SVM or 35.9% of the Vapnik's sparse LP SVM. Finally, having only two types of parameters to tune, this machine is straight forward and cheaper to be produced compared to the most popular & state-of-the-art machines in this direction. These collectively fulfill the three key goals that the machine is built for.**

Keywords—Generalization failure rate; Kernel machine; LP; QP; machine accuracy cost; Second Order Support Vector Machine; sparse

I. Introduction

Run time optimization of classifiers is a crucial issue for fast data classification. A prominent example is from Viola and Jones [1] on face detection based on a cascade of boosted weak classifiers. This framework is not efficiently applicable to kernel based classifiers like support vector machines (SVMs) [2], for instance, because boosting based on such strong classifiers as components is less effective. In many applications, the flexibility of such kernel machines is a real advantage. While SVM based classifiers play the leading role in pattern classification with highest accuracy, one of its key properties is that the learned classifier can be expressed in terms of only a subset of the training patterns, known as support vectors (SVs). But as the computational load of using such a classifier to classify a pattern is proportional to the number of SVs, SV sparsity is extremely important for large datasets. This is especially the case when the training is done once on powerful computers that can handle large data but the prediction is needed to be done multiple times possibly on a small low-powered devices in real time. This motivates to design kernel based classifiers maintaining the trade-off between accuracy and sparsity. Consequently, this problem has come to the center of main attention in research recently.

In this paper, we have proposed a new sparse algorithm "Second Order SVM (SOSVM)" and carried out experimental studies on it as well as standard QP SVM [2] and Vapnik's LP SVM [2] to analyze their performance & efficiency on the basis of computational cost and generalization ability. For simplicity in discussion, only Gaussian kernel is applied throughout the whole work. Standard machine learning benchmark data is used for experiment.

In Section II related works are discussed, in Section III we re-describe SVMs, in Section IV we explain our approach whereas Section V is for experiments and we use Section VI for conclusion and discussion including future work.

II. Related Works

Related work can be approximately, but not disconnectedly, classified

- into approaches [3]–[11] to the design of Reduced SVMs (RSVMs) that demand less computational loads than standard SVM for classifying a pattern;
- into approaches [7], [12]–[16] that exploit SVMs as components of a detector with structured architecture for classification
- into approaches [17]–[20] that develop SVM related cheaper classifiers, which are different from usual RSVMs;
- into approaches [21]–[31] that investigate ensemble-detector by boosting weak classifiers;
- into approaches [32]–[40] that improve one or more of the three variables cost, efficiency, & accuracy of a detector by applying different techniques on different hypothetical single classifiers using one of them or combining more of them considering un/balanced data.

Regarding the first class of approaches, RSVMs demand only a fraction of kernel evaluations to classify a pattern. Wavelet approximations of these latter vectors have also been investigated in [6] for an efficient evaluation of the arguments to which the kernel function is applied. However, while [4], [9], [11] proposed some smart iterative algorithms for reduced

SVMs with impressive results, [9] reported a memory run out from [4], [11] in case of their implementations on large dataset whereas [9] has a considerable practical variation with heavy parameter selection from its defined approach. The second class of approaches, in contrast to the first one, is focusing on structured SVM-based classification for pattern detection. Heisele et al. [13] studied a hierarchy of linear SVMs with a single nonlinear SVM at the end. Thresholds were tuned for optimizing classification performance and speed, followed by feature selection. Romdhani et al. [7] proposed a single chain of SVMs that is optimized also by threshold tuning, and by approximating a fully nonlinear SVM that has to be computed beforehand, whereas a decision tree with linear SVMs is suggested in [12]. Sahbi and Geman [14] presented a tree-structured hierarchy of SVMs that is optimized by reduced set technique in [7] and threshold selection, and operates on application specific partitioning of the space of patterns following different poses. Huo Chen [15] talked about numerical strategies for optimal cascade and checked three heuristics on synthetic data using binary SVM on each stage of a cascade. However, the third class of approaches, being a bit correlated to the second one as originated from the SVM principle, has reasonable discrepancy from that as well from the structural point of view. Maji et al. [17], [20], [36] showed that SVMs using histogram as well as additive kernels are faster and outperform linear SVM. Ladicky - Torr [18] proposed a novel locally linear SVM classifier with smooth decision boundary and bounded curvature while suggesting a trade-off the number of anchor points against the expressivity of the classifier in order to avoid overfitting and speed problem. Xu et al. [19] introduced a post-processing algorithm that compresses the learned SVM by further training on the SVs with adding few extra training parameters. Enthusiastically, the fourth class of approaches has a bit similarity to the second one from the construction principle as they both use a cascade like approach. Xiao et al. [21] used an idea named "Dynamic Cascade" as Face detector that is trained on large data set by dividing them into subsets and hence working on them while using "Bayesian Stump" as weak learners for boosting. Luo H. [22] designed optimization for cascaded classifier that finds the optimum thresholds of different stages for a fixed set up. Saberian et al. [23] introduced a mathematical model for a cascaded detector relating classification and complexity. Chen et al. [24] proposed an algorithm for a cascaded detector considering operational cost, accuracy, and feature extraction cost. Chen et al. [25] presented a general cascade framework that unifies detection learning and alignment for face detection. Li Zhang [26] offered a fast cascaded object detector having fewer stages and using logistic regression as weak learner, which emphasize on training efficiency. Raykar et al. [27] proposed a soft cascade where classifiers accept/reject patterns following probability distributions induced by the earlier stages' classifiers. Considering a fixed order of different stages in the cascade, they tried to find a trade off between accuracy and feature acquisition cost. Visentini et al. [28] devised an algorithm that dynamically builds a cascade of classifiers to speed-up the Online Boosting technique. The cascade explicitly considers the computational cost of the involved features to maintain real-time performance while its classifiers are automatically in tune balancing speed and accuracy. Saberian et al. [29] suggested a cascade boosting algorithm, fast cascade boosting (FCBoost) that minimizes Lagrangian risk while considering speed and accuracy. They introduced the concept of "neutral predictors" that robotically determines the cascade configuration such as number of cascade stage and number of weak learners in each stage. Xu et al. [30] offered a tree of classifiers to balance the test cost and accuracy while Xu et al. [31] analyzed the trade-off problem considering one more variable, feature orientation cost. At last, interestingly, the fifth class of approaches is quite diverge. Fu et al. [32] discussed a problem of combining linear SVMs to classify non-linear data set and claimed experimental results showing that their method can achieve the efficiency of LSVMs in the prediction phase while providing a classification performance comparable to nonlinear SVMs. Cheng- Jhan [33] proposed a pedestrian detector by cascading AdaBoost and SVM classifiers in different stages. A classifier for digit recognition was proposed by Maji et al. [34] that poses reduced operational cost with improved features. It also claimed to have the best result in all three aspects like accuracy, train-cost, and test-cost while using histogram-gradient features and intersection kernel SVM. Gu - Han [35] designed a Clustered Support Vector Machine (CSVM), by weighted combination of linear SVMs (LSVM) trained on the clustered subsets of the training data to separate the data locally. These combined LSVMs are regularized globally to leverage the inter cluster information and avoid over-fitting in each cluster. They derive a data-dependent generalization error bound for CSVM, which explains the advantage of CSVM over linear SVM. Sharma et al. [37] offered an approach for learning non-linear SVM at reduced computational cost in the test phase and empirically analyzed the tradeoff between encoder and classifier complexity and strength. Osadchy et al. [38] proposed a so called hybrid classifier to tackle the problem with data set having high asymmetry as the large portion of the pattern space belongs to the negative class; their kernel hybrid classifier is for further efficiency than SVM while having similar accuracy [39]. Vedaldi et al. [40] offered a three-stage classifier combining linear, quasi-linear, and non-linear kernel SVMs. They showed that increasing the non-linearity of the kernels increases their discriminative power at the cost of an increased computational complexity. Nevertheless, their three stage cascade to overcome the complexity cost has resulted in quite a 'heavy' algorithm in both training and testing.

III. Support Vector Machine (SVM)

Support Vector Machines (SVMs) is a state-of-the-art and popular machine learning technique that has been confirmed as a very powerful tool for Supervised Classification. In this part, we re-describe SVM with its two main variants; one is the standard & most common method using the quadratic programming (QP), we call it QPSVM, while the other one is the Vapnik's linear programming SVM, we call it VLPSVM. We also make a mild comparison between these two.

A. Quadratic Programming SVM (QPSVM)

Here, we briefly review the basic learning algorithm of the QP based Support Vector Machine (SVM) using margin maximization between two classes, which consists in finding the separating hyperplane that is furthest from the closest object; a detailed introduction could be found in [2].

Consider a binary classification problem of dataset where a set of training patterns $\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^d$ and

$y_i \in \{-1, 1\}$ is given. As the objective of the SVM algorithm is to find the optimal separating hyperplane that skillfully separates these patterns into two classes, it offers a classifier using a decision function (for the input pattern x) of 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 function and the parameters w and b are found from a series of calculations starting from the following QP problem:

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

$$s.t. \ \ y_i\big(w \cdot \phi(x_i) + b\big) \ \geq \ 1 - \zeta_i \qquad (2)$$

$$\zeta_i \ \geq \ 0; \ i = 1, 2, ..., N \qquad (3)$$

Where the set of constraints (2) implies that the decision function should classify correctly all patterns from the given training set up to some tolerable errors, the slack variables $\zeta_i > 0$ hold for margin-outward-deviated patterns (that is, patterns staying outwards from their class-margins) and $C > 0$ is a parameter of the classifier that controls the trade off between two main goals of the objective function in (1): one is to optimally maximize the margin between the two classes and another is to minimize the number of misclassifcations on the training patterns.

Common practice to realize a solution for this problem is to solve its dual problem, developed by introducing a Lagrangian and the Lagrangian of the problem form (1)-(3) is

$$L_P(w, b, \zeta, \alpha, \gamma) = \frac{1}{2}\|w\|^2 \ + C\sum_{i=1}^{N}\zeta_i$$

$$-\sum_{i=1}^{N}\alpha_i\Big(y_i\big(w \cdot \phi(x_i) + b\big) \ - 1 + \zeta_i\Big) \ - \sum_{i=1}^{N}\gamma_i\zeta_i \qquad (4)$$

$$\alpha_i, \gamma_i \ \geq \ 0; \ i = 1, 2, ..., N \qquad (5)$$

where α_i are Lagrange multipliers and we get the corresponding dual problem as

$$\max_{\alpha} f_D(\alpha) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) + \sum_{i=1}^{N}\alpha_i \qquad (6)$$

$$s.t \ \ \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad (7)$$

$$0 \leq \alpha_i \leq C; \ \ \ i = 1, 2, ..., N \qquad (8)$$

This is also a QP problem and optimum values of its variable α are used to find the primal variables w, b as $w = \sum \alpha_i y_i \phi(x_i)$ and $b = y_s - w \cdot \phi(x_s)$ where s is an index of any pattern for which $0 < \alpha_s < C$. One of the KKT conditions for the problem (1)-(3) is $\alpha_i\Big(y_i\big(w \cdot \phi(x_i) + b\big) \ - 1 + \zeta_i\Big) = 0$ from which for $\alpha_i \neq 0$, we get $y_i\big(w \cdot \phi(x_i) + b\big) \ - 1 + \zeta_i = 0$. These patterns having $\alpha_i > 0$ are support vectors (SVs), which are usually far less (depending on the data set) in number compared to the total training set size that proves QPSVM to be sparse. From these SVs, $\alpha_i < C$ (are those having $\zeta_i = 0$) patterns lie on the margin of own class whereas $\alpha_i = C$ (are those having $\zeta_i > 0$) patterns stay outwards from their corresponding margins. Interestingly, the constraint (7) that is $\sum_{i=1}^{N} \alpha_i y_i = 0$ ensures that in this QPSVM, SVs set must have members from both classes. In QPSVM model, SVs are the only training patterns that contribute in designing an optimal classifier.

B. Vapnik's LP SVM (VLPSVM)

Here, we concisely go through the linear programming approach proposed by Vapnik to find a separating hyperplane that is very similar to that of the QPSVM one but demands comparatively less computation to classify a pattern. More elaborate could be found in [2].

Inferring that the classifier has the same form of kernel expansion using the SVs in the QPSVM, Vapnik used a linear objective function to minimize the sum of all the coefficients used in the kernel expansion. Each coefficient is associated with its corresponding KCV (kernel computing vector) in the expansion. For better clarification, we name these vectors of this machine as "Expansion Vector (EV)", which is similar to SVs in QPSVM.

Considering that the decision function preserves exactly the same form of kernel expansion as the QPSVM and the error constraints of the QPSVM also remain almost the same, Vapnik proposed this VLPSVM focusing at minimizing the number of kernel computation by reducing EVs of the separating hyperplane that has the weight vector w_V of the decision function $f_V(x) = w_V \cdot \phi(x) + b_V$ leading to $class(x) = sgn(f_V(x))$. For this purpose, he formed a linear objective function using the coefficients of the EVs directly and coupling the error penalty on top of the error constraints as below:

$$\min_{\lambda,\xi,b_V} \ \ \sum_{i=1}^{N}\lambda_i \ + \ C_V\sum_{i=1}^{N}\xi_i \qquad (9)$$

$$s.t. \ \ y_i\Big(\sum_{j=1}^{N}\lambda_j y_j \phi(x_j) \cdot \phi(x_i) + b_V\Big) \ \geq \ 1 - \xi_i \qquad (10)$$

$$\lambda_j \ \geq \ 0; \ j = 1, 2, ..., N \qquad (11)$$

$$\xi_i \ \geq \ 0; \ i = 1, 2, ..., N \qquad (12)$$

Alike the QPSVM, the set of constraints (10) implies that the decision function should classify correctly all patterns from the given training set up to some tolerable errors, the slack variables $\xi_i > 0$ hold for absolute-unity-outward-deviated patterns (that is, training patterns having $ClassLabel(x_i) \cdot f_v(x_i)$ <1) and $C_V > 0$ is a parameter of the classifier that controls the trade off between data learning and overfitting.

For this machine, the bias term, b_V of the decision function is an optimization variable of the main problem ((9)-(12)) and found along with other optimization variables λ and ξ. The optimum values of λ are used to find the weight vector w_V as $w_V = \sum \lambda_j y_j \phi(x_j)$. Training patterns, x_j with coefficients $\lambda_j > 0$ are the EVs, which are usually very much smaller (depending on the data set) in number compared to the total training set size showing VLPSVM to be sparser. Unlike QPSVM, here we have no constraints that forces the machine to have KCV from both classes.

C. Comparison between VLPSVM & QPSVM

Unlike QPSVM, we directly implement the primal to be optimized in case of VLPSVM and get the optimal value of the bias along with the optimal co-efficient set of the expansion vector. Interestingly, in case of this machine, values of the absolute-unity-outward-deviation parameters ξ_i are also found as a subset of the optimum variable by solving the LP.

Although QPSVM & VLPSVM do not have the same constraint set fully, part of the constraints they use are almost the same. For example, they use nearly the same error constraint & the non-negative lower bounds of the variables (KCV-expansion co-efficients α, λ, margin/absolute-unity outward deviation ζ, ξ).

Both machines are heavily influenced by the error penalty parameter (on top of kernel parameter) where the QPSVM introduces this parameter, C through the constraint but the VLPSVM uses this parameter, C_V by directly optimizing the primal cost function that includes it, which gives it a chance to have further significance.

For the QPSVM, maximizing the margin and minimizing the error are the two basic modules in its mathematical modelling while finding the best trade off between minimizing the error and minimizing the sum of the KCV coefficients is the main theme of VLPSVM. To do so, in its mathematical formulation, VLPSVM directly puts the sum of the non-negative coefficients of KCVs in the objective function to be minimized that gives a sparser solution. Amazingly, with this direct involvement of non negative coefficients of KCVs (by a non-negative summation) in the cost function, VLPSVM very closely replicates the QPSVM. However, by so far, compared to the QPSVM, VLPSVM misses many of the interesting properties that make QPSVM academically richer such as a concrete and validated theoretical base with the efficient dual transformation that couples the kernel functions in the simplest and productive fashion. Still, our experiments on benchmark data as well as other reports [41] show that considering classification performance with generalization, VLPSVM is quite competent like QPSVM while being more efficient proving LPSVM to be empirically richer and more productive. Apparently, it appears to be paradoxical to the basic principle of SVM that a machine with less number of KCVs poses similar generalization performance to that of a machine with more KVCs, but the key point here is that the KCVs in VLPSVM do not have exactly the same topoloical and geometric interpretation as that in QPSVM despite the fact that they are being extensively called by the same name (SVs) in many literature. Further in the same path, unlike QPSVM (due to its constraint $\sum_{i=1}^{N} \alpha_i y_i = 0$) there is no condition in VLPSVM that the KCV set contains training patterns from both classes, which may help it to reduce the number of KCVs in the decision function. Furthermore, as L_1 norm is usually more intending to sparser solution compared to L_2norm [42], by formulating an indirect L_1 norm in VLPSVM cost function, it leads to further sparsity compared to QPSVM that uses the L_2 norm for such. Another concern related to the larger number of KCV of the QPSVM (compared to VLPSVM) is its KKT condition $\alpha_i \Big(y_i \big(w \cdot \phi(x_i) + b \big) \ - 1 + \zeta_i \Big) = 0$ that forces all the training patterns staying on the class-margin of own class or outwards to be KCV. That means, some sorts of training patterns must be included in the KCVs set in the QPSVM and this becomes specially serious in case of large and noisy datasets as they contain such overlapping and non-separable examples with a big portion. On contrary, VLPSVM has no such condition, which gives it flexibility to chose KCVs from more scattered pattern space following the demand of the stochastic and topological property of data-patterns leading to pick up few but crucial patterns that are perfect to be KCVs for a very sparse but powerful classifier with strong generalization capability.

IV. Proposed Method: SOSVM

While a powerful classifier is essential to handle with the difficulties from large and noisy data, controlling the classifier-complexity is also important to achieve better generalization. Additionally, considering both cost and accuracy, the best classifier is the most sparse one, having the highest generalization capability, posing least test error. To serve this purpose, we try a novel algorithm by applying both QP and LP in a structured sequence.

As we discussed earlier that although both QPSVM and LPSVM are sparse, VLPSVM produces sparser solution than QPSVM while posing very similar accuracy. Still, this sparsity form VLPSVM is not sufficient for large data set. So, we look for a machine that is even further sparse and faster but more generalized and powerful aiming at real-time classification on very large and complex data.

We know that the sparseness of SVMs heavily depends on the noise and complexity of the data. When the data set is very noisy, a good generalized QPSVM may get more outliers, which will be included into the SVs set in addition to the patterns that are just on the margins. So, number of SVs will soar while generalization capability of the machine will also rise and this SVs set is one of the best representative sets of the whole data. Moreover, as the SVs set from QPSVM are sufficient to represent the discrimination between the classes, we consider only this SVs patterns (who also mostly stay around the discrimination boundary using margin maximization concept) for next manipulation in order to produce our efficient classifier by further sparsification without losing generalization ability. We then run LP (in VLPSVM fashion) on this SVs set as this LP will impose a co-efficient vector carrying weights of these SVs patterns to minimize the objective function while maintaining classification accuracy and generalization potential. Hence, this weight vector will have updated co-efficient values (from the QPSVM SVs-coefficients) being further (2nd time) sparse and will promote to an extensive computational reduction by enabling much smaller number of KCVs (after throwing a large part of the SVs) to be involved in the final decision function.

This gives us twofold benefits: one, training set is condensed by a pure filtration picking only the significant patterns that are already bases of a theoretically solid and powerful classifier and are sole representer of the data. By this, we also abstain from the computation of novel representatives of SVs as this relies upon complex optimization problems that are susceptible to initialization, step sizes, etc. Second, we take advantage from the sparser, and flexible pattern selecting capability of VLPSVM for KCV from a scattered and random

Our Algorithm: SOSVM

```
1: Input: A training set $(x_i, y_i)_{i=1}^N$
2: Output: A discriminator $f_{SOSVM}(\cdot)$
3: Select two of the best pairs of $(Penalty\ parameter, Kernel\ parameter) \equiv (C,\sigma),(C_V,\sigma_V)$ for two stages
4: Run QPSVM on the training set solving the following following problem:
   $$\min_{\alpha} f(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi(x_i)\cdot\phi(x_j) - \sum_{i=1}^{N}\alpha_i$$
   $$s.t \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$
   $$0 \le \alpha_i \le C; \quad i = 1,2,...,N$$
5: Extract SVs with their labels $(x_l, y_l)_{l=1}^M$ from the QPSVM using $\alpha_i > 0$
6: Run LP in VLPSVM fashion on these SVs patterns solving the following following problem:
   $$\min_{\lambda,\xi,b_V} \quad \sum_{l=1}^{M}\lambda_l \ + \ C_V\sum_{i=1}^{M}\xi_l$$
   $$s.t. \quad y_i\Big(\sum_{l=1}^{M}\lambda_l y_l \phi(x_l)\cdot\phi(x_i) + b_V\Big) \ \ge \ 1 - \xi_i$$
   $$\lambda_l \ \ge \ 0; \ l = 1,2,...,M$$
   $$\xi_i \ \ge \ 0; \ i = 1,2,...,M$$
7: Extract EVs with their labels $(x_m, y_m)_{m=1}^P$ using $\lambda_m > 0$ and the bias $b_V$ from the LP
8: $w_{SOSVM} \leftarrow \sum_{m=1}^{P}\lambda_m y_m \phi(x_m)$ ;   $b_{SOSVM} \leftarrow b_V$
9: Return $f_{SOSVM}(\cdot) = w_{SOSVM}\cdot\phi(\cdot) + b_{SOSVM}$
```

pattern space. Additionally, the sequential training of two inductive submachines where the second one is denser being truncated and a function of the first one leads to a resultant final one higher ordered function. So, our discriminator virtually plays the functional role of a second ordered decision function, which echos the name "Second Order SVM (SOSVM)" of our algorithm while this higher ordered nature better handles the random and nonlinear behavior of the data.

Patterns having the non-zero co-efficient values from final output are the KCVs of our SOSVM and we call them "Machine Vectors(MVs)" whereas the bias of this SOSVM comes from the solution of the final optimization problem; these components construct the SOSVM using corresponding patterns and labels. It is worth mentioning here that while the optimizations in the two stages are done in sequence, their supporting parameters are found simultaneously by a joint search using a modified cross validation technique, which is in accordance with the overall system as a single machine.

Fig. 1 shows the decision boundary with number of Kernel computing vectors (KCVs), and training error rates from QPSVM, VLPSVM, and SOSVM on Banana data set from machine learning benchmark. Fig. 1(a) shows the decision boundary for QPSVM with $C = 4096$ and $\sigma = 2$, Fig. 1(b) shows the decision boundary for VLPSVM with $C_V = 4$ and $\sigma_V = 1$ and the decision boundary for SOSVM with $C = 8, \sigma = 0.125, C_V = 4$ and $\sigma_V = 1$ is shown in Fig. 1(c). Banana is a two dimensional data with 400 training patterns from which QPSVM uses 94 KCVs, VLPSVM uses 14 whereas our SOSVM uses only 11 while posing training error rates 6.75%, 8.75%, and 8.75%, respectively. Therefore, to classify a single pattern, SOSVM demands only 11.7% kernel execution of QPSVM (which is sparse) and 78.6% that of the VLPSVM (which is sparser) while offering very similar accuracy!

V. Experiments and Results

A. Key Terms to Analyze Machine's Perfection

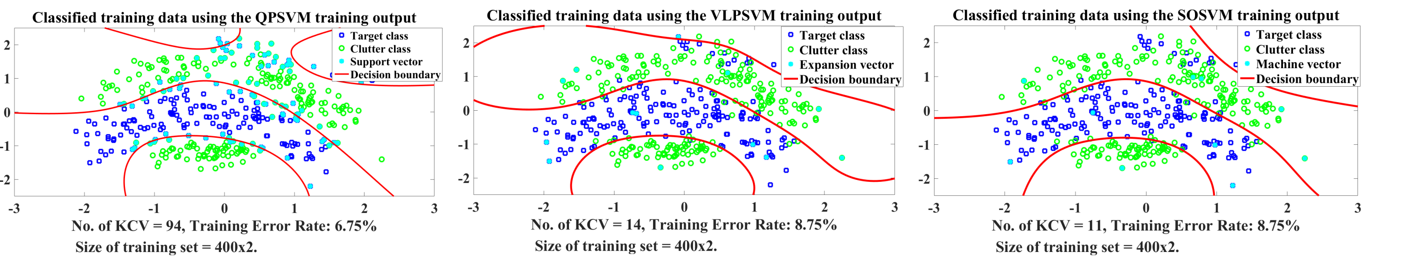
So far, there is a convention to find the test error rate of a classifier to realize its generalization-quality. In fact, it tells about the classifier's performance on test data, which is a must to know but may not give complete info about machine's bridging capability between the training and test data. So, we define a novel term called "Generalization Failure Rate (GFR)" that includes machine's performances on the training set, test set and their difference. To evaluate the classifier's deficiency, GFR is based on the two coupled info: 1) How much the classifier intends to overfit; and 2) How bad it performs on the test set.

Further, to terminate the confusion between the usefulness of an expensive machine with highest accuracy and a cheaper machine with acceptable accuracy, another new term "Machine-Accuracy-Cost(MAC)", expressing cost per accuracy is defined.

1) Generalization Failure Rate (GFR): The main goal of a classification algorithm is to discover a discriminating function basing on the training set (input patterns and the corresponding labels) that will generalize well by classifying the novel patterns with the least possible errors. However, to make it real time applicable, the secondary objective is that the classifier should be as sparse as possible, that is, in case of basis-vector based machine, it should have as few basis as possible. But this basis-set (hence its size) radically influences the machine's generalization performance.

If this basis-set lead towards a very simple model, it fails to learn the data-complexity and thus poses poor performance on both the training and test set by underfitting.

On contrary, if this basis-set lead towards a very complex model, it learns the irrelevant detail and noise in the training dataset (and weakening the general model) leading to the decrement of the training error by overfitting and increment of the test error with generalization-failure. Thus measuring the generalization quality of a classifier is really indispensable. However, although both overfitting and underfitting can lead to model's performance failure, the most frequent problem in machine learning is overfitting. So, we start by focusing on it and define a term called "Overfitting Tendencey(OT)" as the difference between Test Error Rate and Train Error Rate per Train Error Rate; mathematically, $OT = \frac{TestErrorRate - TrainErrorRate}{TrainErrorRate}$. Hence, OT gets higher for a higher value of $\frac{TestErrorRate}{TrainErrorRate}$, which increases for lower Train Error and higher Test Error. Further, to include the loss done by underfitting, we divide this OT by Test Accuracy and define the term as the GFR that measures the overall Generalization deficiency of the model, that is $GFR = \frac{OT}{TestAccuracy}$. It is quite clear that GFR gets higher either by increasing in OT or by decreasing the Test Accuracy or by both. It is to note that the term "GFR" is defined here on the assumption that $TestErrorRate, TrainErrorRate \in (0, 100)\%$ and $TestErrorRate > TrainErrorRate$, which is the usual case.



(a) (b) (c)

Fig. 1. Decision boundaries, number of Kernel Computing Vector (KCV), & training error rates from (a) QPSVM, (b) VLPSVM, & (c) SOSVM on Banana data.

2) Machine-Accuracy-Cost (MAC): In almost all cases, it is desired to have an efficient machine, that is a machine with less computational cost but having high performance. Therefore, we need to build a machine demanding less kernel execution (to classify a test pattern), hence with less number of Kernel Computing Vectors (Support Vectors or Basis Vectors or Expansion Vectors or Machine Vectors, however it is called by different authors for different machines, we, here, call those "Kernel Computing Vectors(KCVs)" that involve kernel evaluation) and high test accuracy. To measure the achievement of such property by a machine, we define a term "Machine-Accuracy-Cost(MAC)" as $MAC = \frac{Number\ of\ KCVs(\#KCVs)}{Test\ Accuracy(TeA)}$. So, a machine with the highest number of KCVs which increases for lower Train Error and hig lowest test accuracy will have the maximum MAC(which is never desired), whereas a machine with the lowest number of KCVs giving the highest test accuracy will have the minimum MAC (which is always desired).

B. Experimental Set-up and Results

In this section, the experimental results obtained from the proposed method are presented to show the efficiency of the SOSVM and to compare with QPSVM and VLPSVM. The experiments were performed on six benchmark machine learning datasets [43] namely Banana, Diabetics, Heart, Thyroid, Titanic and Twonorm as listed in Table I. For all three machines Gaussian kernel is used throughout the whole experiment. In case of QPSVM and VLPSVM, the penalty parameters C, C_V and the kernel parameters σ, σ_V are chosen based on the lowest crossvalidation error rate for each dataset using five-fold crossvalidation scheme. This two cases are experimented with $C \in \{2^{-2}, 2^0, 2^2, ..., 2^{12}\}$ and $\sigma \in \{2^{-2}, 2^0, ..., 2^6\}$. For SOSVM, there are C, σ in the first stage and C_V, σ_V in the second stage. To figure out the best C, σ, C_V and σ_V, a modified scheme of five-fold crossvalidation is implemented. In this scheme, 4 folds of randomly chosen training data are used to feed in the first stage for a particular value of C and σ. The KCVs from the first stage are used to feed as training set with a particular value of $C_V$and σ_V in the second stage. The returned KCVs of the second stage VLPSVM are treated as the overall KCVs of SOSVM and this classifier is used to test the remaining one fold of training data. The parameters C, σ, C_V, σ_V with the lowest crossvalidation error rate are chosen as the best ones. This approach is experimented with $C \in \{2^{-2}, 2^0, 2^2, ..., 2^{12}\}, \sigma \in \{2^{-2}, 2^0, ..., 2^6\}, C_V \in C \times \{2^{-2}, 2^{-1}, 2^0, ..., 2^5\}$ and $\sigma_V \in \sigma \times \{2^{-2}, 2^{-1}, 2^0, ..., 2^5\}$.

To evaluate the quality of the result obtained by the proposed SOSVM, it is compared with the results obtained by QPSVM and VLPSVM in Table I. Table I presents the number of training and testing patterns of different dataset along with the average number of KCVs and average test error rate. From this Table, it is observed that the proposed method (SOSVM) results in a lower number of KCV than QPSVM in all cases and also lower than VLPSVM in most cases. In case of QPSVM the average number of KCVs for all the datasets is 154.15 (SD 8.19) whereas for VLPSVM it is 29.08 (SD 8.16). But in case of our proposed SOSVM the average number of KCVs is 18.29 (SD 8.34) which is $1/9^{th}$ of QPSVM and $2/3^{rd}$ of VLPSVM. Therefore, it is clear that in most cases SOSVM results in a reduction in the number of KCVs and in some cases substantial reduction.

Moreover, in order to show how well our machine is capable of generalizing than the traditional machines, the performance of the proposed method along with the traditional QPSVM and VLPSVM is compared in terms of Generalization Failure Rate (GFR). In Table II, the GFRs of traditional QPSVM, VLPSVM and the proposed SOSVM are listed for various dataset. Also, the ratios of GFRs of the proposed SOSVM with respect to traditional machines are listed to show how well SOSVM performs to generalize the training data. It can be seen from the table that the average GFR value of our machine as small as 30% of the most powerful classifier like the QP SVM and 2% of the LP SVM.

Finally, in order to judge the proposed in terms of Machine Accuracy Cost (MAC), the MAC values of the proposed method along with the traditional QPSVM and VLPSVM is listed Table III. Also, the ratios of MACs of the proposed SOSVM with respect to traditional machines are listed to show how well our machine minimizes the cost. It can be seen from the table that the average MAC value of our machine as small as 16% of the most the most powerful classifier like the QPSVM and 80% of the sparse LP SVM.

VI. Conclusion and Future Work

We have developed a fast but powerful classifier by using sequential optimization that is supported with simultaneous parameter search. We have also defined two new terms "GFR" and "MAC" that can be directly and easily measured to verify a detector's perfection. Our classifier is very much straight forward using least effort to train. Compared to the state-of-the-art sparse classifiers, it is more efficient, hence, posing average MAC value as small as 16% of the standard QP

TABLE I. No. of KCVs (Kernel Computing Vectors), which Increases for Lower Train Error and High Test Error Rate on Benchmark Data for Different Machines

Dataset	No. of train pattern	No. of test pattern	Data Dimension	QPSVM mean SVs (SD)	QPSVM mean TeER (SD)	VLPSVM mean EVs (SD)	VLPSVM mean TeER (SD)	SOSVM mean MVs (SD)	SOSVM mean TeER (SD)	$\frac{\#MVs}{\#SVs}$	$\frac{\#MVs}{\#EVs}$
BANANA	400	4900	2	102.26 (14.09)	10.61 (0.53)	15.08 (1.30)	10.75 (0.51)	15.23 (1.75)	10.91 (0.52)	0.1489	1.0099
DIABETIS	468	300	8	263.22 (15.63)	23.28 (1.70)	12.58 (1.94)	23.40 (1.77)	12.56 (1.93)	23.43 (1.78)	**0.0477**	0.9984
HEART	170	100	13	68.23 (6.01)	16.60 (3.05)	21.94 (2.61)	17.44 (3.49)	10.60 (9.47)	15.55 (3.20)	0.1554	**0.4831**
THYROID	140	75	5	43.51 (3.10)	5.20 (2.08)	8.97 (1.59)	5.09 (2.11)	4.34 (1.27)	7.77 (8.14)	0.0997	0.4838
TITANIC	150	2051	3	148.50 (3.33)	22.69 (0.86)	83.91 (36.52)	22.91 (0.60)	48.48 (34.06)	23.34 (1.39)	0.3265	0.5778
TWONORM	400	7000	20	299.18 (7.00)	2.42 (0.14)	32.00 (5.03)	3.71 (0.55)	18.50 (1.55)	3.38 (0.38)	0.0618	0.5781
Average	288	2404.33	8.50	154.15 (8.19)	13.47 (1.39)	29.08 (8.16)	13.88 (1.50)	18.29 (8.34)	14.06 (2.57)	–	–

TABLE II. Generalization Failure Rate (GFR) for Different Machines

Dataset	QPSVM	VLPSVM	SOSVM	$\frac{SOSVM}{QPSVM}$	$\frac{SOSVM}{VLPSVM}$
BANANA	0.0031	0.0026	0.0019	0.6265	0.7618
DIABETIS	0.0010	0.0014	0.0014	1.3083	1.0100
HEART	0.0039	0.0060	0.0012	0.3073	0.2018
THYROID	0.0918	0.0109	0.0013	**0.0142**	0.1194
TITANIC	0.0017	0.0015	0.0010	0.6021	0.6975
TWONORM	0.0017	1.7005	0.0246	14.2239	**0.0145**
Average	0.0172	0.2871	0.0052	–	–

TABLE III. Machine Accuracy Cost (MAC) for Different Machines

Dataset	QPSVM	VLPSVM	SOSVM	$\frac{SOSVM}{QPSVM}$	$\frac{SOSVM}{VLPSVM}$
BANANA	1.1439	0.1690	0.1520	0.1329	0.8996
DIABETIS	3.4311	0.1642	0.0876	**0.0255**	0.5337
HEART	0.8181	0.2657	0.0955	0.1168	**0.3594**
THYROID	0.4590	0.0945	0.0945	0.2059	1.0000
TITANIC	1.9208	1.0885	1.0666	0.5553	0.9799
TWONORM	3.0661	0.3323	0.2001	0.0653	0.6021
Average	1.8065	0.3524	0.2827	–	–

SVM, 80% of the sparse LP SVM by Vapnik. Moreover, being optimally complex and powerful, its overfitting tendency is really low, which leads it to offer average GFR value as small as 30% of the most powerful classifier like the standard QP SVM and 2% of Vapnik's LP SVM.

Due to this exceptionally good performance, one question pops up about how it manages to perform better than Vapnik's LP SVM (VLPSVM) or the standard QP SVM (QPSVM). We do not have any theoretical explanation for it now (and left for future work) but one plausible explanation for it could be that by the second layered training from the sequential combination of the two sub-machines generated by QP and LP (using corresponding parameters by a joint and simultaneous search) respectively, we get a machine vectors set being second ordered filtered and scaled (hence learned) having stochastic and topological properties complex and sophisticated than Support Vectors (in case of QPSVM) or Expansion Vectors (in case of VLPSVM) while working in the similar method for the discriminator. Thus, our algorithm produces a hybrid and unconventional hyperplane, based on a compact second ordered representer set coupled with corresponding co-efficent vector and bias that collectively adopts the statistical and geometric properties of training data very skillfully and generalization is boosted.

Anyway, while our classifier has consistently over performed state of the art complex and sparse classifiers with respect to computational cost and accuracy, we are considering some further manipulation with it where parts are given below:

1) An extension from two-stages including further stages.

2) A deep theoretical analysis relating the data characteristics, components of the sub-machines as well as their sequential behavior and pattern-space sharing including low GFR and MAC would be interesting.

At last, an efficient and accurate classifier like our SOSVM is very much essential. For example, our classifier is indispensable in real life, where one may have more time and resources to train but very less to test.

References

[1] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[2] V. Vapnik, *Statistical learning theory*. Wiley, 1998.

[3] F. Vojtěch and V. Hlaváč, "Greedy algorithm for a training set reduction in the kernel methods," in *Proc. of the International Conference on Computer Analysis of Images and Patterns*. Springer, 2003, pp. 426–433.

[4] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *Journal of Machine Learning Research, (JMLR)*, vol. 7, no. Jul, pp. 1493–1515, 2006.

[5] E. Osuna and F. Girosi, "Reducing the run-time complexity of support vector machines," in *Proc. of the International Conference on Pattern Recognition (ICPR)*, 1998, pp. 1–10.

[6] M. Rätsch, S. Romdhani, G. Teschke, and T. Vetter, "Over-complete wavelet approximation of a support vector machine for efficient classification," in *Proc. of the Joint Pattern Recognition Symposium*. Springer, 2005, pp. 351–360.

[7] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake, "Efficient face detection by a cascaded support–vector machine expansion," in *Proc. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 460, no. 2051, 2004, pp. 3283–3297.

[8] M. Wu, B. Schölkopf, and G. Bakır, "A direct method for building sparse kernel learning algorithms," *Journal of Machine Learning Research (JMLR)*, vol. 7, no. Apr, pp. 603–624, 2006.

[9] A. Cotter, S. Shalev-Shwartz, and N. Srebro, "Learning optimally sparse support vector machines," in *Proc. of the 30th International Conference on Machine Learning (ICML)*, 2013, pp. 266–274.

[10] Y. J. Lee and O. L. Mangasarian, "RSVM: reduced support vector machines," in *Proc. of the First SIAM International Conference on Data Mining (SDM)*, 2001, pp. 1–17.

[11] T. Joachims and C. N. J. Yu, "Sparse kernel svms via cutting-plane training," in *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML)*, 2009, p. 8.

[12] K. Z. Arreola, J. Fehr, and H. Burkhardt, "Fast support vector machine classification using linear SVMs," in *Proc. of the 18th International Conference on Pattern Recognition (ICPR)*, vol. 3, 2006, pp. 366–369.

[13] B. Heisele, T. Serre, S. Prentice, and T. Poggio, "Hierarchical classification and feature reduction for fast face detection with support vector machines," *Pattern Recognition*, vol. 36, no. 9, pp. 2007–2017, 2003.

[14] H. Sahbi and D. Geman, "A hierarchy of support vector machines for pattern detection," *Journal of Machine Learning Research (JMLR)*, vol. 7, no. Oct, pp. 2087–2123, 2006.

[15] X. Huo and J. Chen, "Building a cascade detector and applications in automatic target detection," *Applied Optics: Information Processing*, vol. 43, no. 2, pp. 1–47, 2003.

[16] R. Karim, M. Bergtholdt, J. H. Kappes, and C. Schnörr, "Greedy-based design of sparse two-stage svms for fast classification," in *Proc. of the 29th DAGM Symposium on Pattern Recognition*, 2007, pp. 395–404.

[17] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel svms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 66–77, 2013.

[18] L. Ladicky and P. H. S. Torr, "Locally linear support vector machines," in *Proc. of the 28th International Conference on Machine Learning, (ICML)*, 2011, pp. 985–992.

[19] Z. E. Xu, J. R. Gardner, S. Tyree, and K. Q. Weinberger, "Compressed support vector machines," *arXiv preprint arXiv:1501.06478*, 2015.

[20] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[21] R. Xiao, H. Zhu, H. Sun, and X. Tang, "Dynamic cascades for face detection," in *Proc. of the IEEE 11th International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[22] H. Luo, "Optimization design of cascaded classifiers," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 480–485.

[23] M. J. Saberian and N. Vasconcelos, "Boosting classifier cascades," in *Proc. of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 2010, pp. 2047–2055.

[24] M. Chen, Z. E. Xu, K. Q. Weinberger, O. Chapelle, and D. Kedem, "Classifier cascade for minimizing feature evaluation cost," in *Proc. of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 218–226.

[25] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun, "Joint cascade face detection and alignment," in *Proc. of the 13th European Conference on Computer Vision (ECCV)*, 2014, pp. 109–122.

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

[27] V. C. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: tradeoff between accuracy and cost," in *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 853–860.

[28] I. Visentini, L. Snidaro, and G. L. Foresti, "Cascaded online boosting," *J. Real-Time Image Processing*, vol. 5, no. 4, pp. 245–257, 2010.

[29] M. J. Saberian and N. Vasconcelos, "Boosting algorithms for detector cascade learning," *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 2569–2605, 2014.

[30] Z. E. Xu, M. J. Kusner, K. Q. Weinberger, and M. Chen, "Cost-sensitive tree of classifiers," in *International Conference on Machine Learning*, 2013, pp. 133–141.

[31] Z. E. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle, "Classifier cascades and trees for minimizing feature evaluation cost," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2113–2144, 2014.

[32] Z. Fu, A. R. Kelly, and J. Zhou, "Mixing linear svms for nonlinear classification," *IEEE Trans. Neural Networks*, vol. 21, no. 12, pp. 1963–1975, 2010.

[33] W. C. Cheng and D. M. Jhan, "A cascade classifier using adaboost algorithm and support vector machine for pedestrian detection," in *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, 2011, pp. 1430–1435.

[34] S. Maji and J. Malik, "Fast and accurate digit classification," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-159*, 2009.

[35] Q. Gu and J. Han, "Clustered support vector machines," in *Proc. of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013, pp. 307–315.

[36] S. Maji and A. C. Berg, "Max-margin additive classifiers for detection," in *Proc. of the IEEE 12th International Conference on Computer Vision (ICCV)*, 2009, pp. 40–47.

[37] G. Sharma, F. Jurie, and P. Perez, "Learning non-linear SVM in input space for image classification," Ph.D. dissertation, GREYC CNRS UMR 6072, Universite de Caen, 2014.

[38] M. Osadchy, D. Keren, and B. F. Specktor, "Hybrid classifiers for object classification with a rich background," in *Proc. of the 12th European Conference on Computer Vision (ECCV)*, 2012, pp. 284–297.

[39] M. Osadchy, D. Keren, and D. Raviv, "Recognition using hybrid classifiers," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 38, no. 4, pp. 759–771, 2016.

[40] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Proc. of the IEEE 12th International Conference on Computer Vision (ICCV)*, 2009, pp. 606–613.

[41] 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.

[42] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[43] G. Rätsch. Benchmark data sets. [Online]. Available: http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm