Sparse Solutions for Single Class SVMs: A Bi-Criterion Approach

Santanu Das * Nikunj C. Oza†

Abstract

In this paper we propose an innovative learning algorithm - a variation of One-class ν Support Vector Machines (SVMs) learning algorithm to produce sparser solutions with much reduced computational complexities. The proposed technique returns an approximate solution, nearly as good as the solution set obtained by the classical approach, by minimizing the original risk function along with a regularization term. We introduce a bi-criterion optimization that helps guide the search towards the optimal set in much reduced time. The outcome of the proposed learning technique was compared with the benchmark one-class Support Vector machines algorithm which more often leads to solutions with redundant support vectors. Through out the analysis, the problem size for both optimization routines was kept consistent. We have tested the proposed algorithm on a variety of data sources under different conditions to demonstrate the effectiveness. In all cases the proposed algorithm closely preserves the accuracy of standard one-class ν SVMs while reducing both training time and test time by several factors.

Keywords: Anomaly Detection, Optimization, Sparse, Scalability, Aeronautics

1 Introduction

Many problems in areas of interest to NASA, such as aviation safety and Earth science, have benefited and will continue to benefit from the use of data-driven methods for anomaly detection. For example, in aviation safety, many airlines have very large datasets representing the operation of their fleets of commercial aircraft. Most of this data represent normal operations of the aircraft—finding examples of anomalous operation is comparable to the proverbial problem of finding a needle in a haystack. An algorithm to find anomalies in such a large dataset clearly needs to be fast and scalable. The algorithm also must be accurate, which requires leveraging as many properties of the dataset as possible. In particular, data from commercial aircraft contain continuous sequences, representing sensor data such as airspeed and altitude, as well as discrete sequences, such as sequences of pilot switch presses. An algorithm that learns from such sequences will tend to outperform typical machine learning algorithms that assume that data collected at every instant in time is independent from data collected at every other instant in time.

We addressed the accuracy issue in [7], where we devised a Multiple Kernel Learning (MKL) version of one-class SVMs containing one kernel over discrete sequences and one kernel over continuous sequences. We chose one-class SVMs as the basis of our developments because of its strong performance as reported by other researchers, its guarantee of optimality given a particular training set, and the flexibility of kernel methods to utilize a variety of different types of features both in single kernel and multiple kernel methods. In [7], we demonstrated our algorithm's effectiveness at finding anomalies within commercial aircraft data. However, the running time of one-class SVMs is higher than for other algorithms that we use because of the need to solve an optimization problem.

In this paper, we address the speed and scalability issue discussed above. We do this through a bi-criterion formulation of one-class SVM—that is, we add a criterion to the objective function that biases the algorithm toward a sparser solution, which we demonstrate theoretically and experimentally. We show that our learning algorithm often has much lower training time than the classical one-class SVM learning algorithm. In some cases, our learning algorithm's run time is higher, but we demonstrate that, in all these cases, our algorithm's time to generate a classification (normal or anomalous) for a new data point is much lower. In spite of this, our algorithm's performance is nearly the same as that of the classical one-class SVM in terms of how it classifies new data. We achieved all these results without requiring any changes to the format of the data or any changes to the rest of the algorithm, such as the optimization problem solver, thereby making the algorithm easy to implement.

In the following section we provide some background research to speed up and scale Support Vector Machines. This will be followed by our motivation and contributions. In Section 3, we describe the optimization problem of original one-class support vector ma-

*UARC, UCSC, NASA Ames Researchh Center, Moffett Field, CA 94035, Santanu.Das-1@nasa.gov.

†NASA Ames Researchh Center, Moffett Field, CA 94035, Nikunj.C.Oza@nasa.gov.

chines model which is the underlying algorithm of our work. Subsequently, Section 4 discusses the bi-criterion optimization which is the heart of this paper, followed by some details on the solver. Experimental evidence of performance of the proposed technique is given in Section 5. Finally we conclude the paper with a discussion in Section 6.

2 Background and Motivation

Kernel based methods like one-class support vector machines have a significant disadvantage in addressing scalability to large number of training points. With increasing training points, the training time and the memory requirements drastically increase and at the same time the prediction time which is proportional to the number of representative support vectors also increases. The number of representative support vectors also holds a proportional relationship with the number of training points. There have been several efforts to over come training and testing time scaling issues either by building an online algorithm, a parallel batch algorithm, or a sophisticated scheme to select more informative training samples. A lot of researchers reported satisfactory contributions in multiple areas like data preprocessing, data compression, kernel modification [12] etc., while others have investigated more in the areas of optimization and solver development. Each of these tasks individually plays an important role in building the model. In [3] Burges and Schölkopf proposed "reduced set" method in order to improve on classification speed and "virtual support vectors" method to improve on accuracy, however at the cost of some increased training time. Some papers talk about how to improve the performance of kernel based methods in general. Most of these literatures examine techniques for efficient matrix factorization, low rank approximation, etc. Schwaighofer and Tresp [17] conducted a comprehensive study on using some of these approaches to scale Gaussian process regression technique on large data sets. Asharaf et al. addresses the scalability problem of SVMs using cluster based training [1, 14] where some selected samples representing the cluster abstractions of the entire training data are used to build the model without compromising the generalized performance. However the outcome of cluster based training will typically depend on the performance of the clustering algorithm. Liang Lie-quan and Liang Ying-hong [11] used a mode sensitive procedure called "mean shift" algorithm for clustering purpose. Another popular technique is chucking algorithms [18] which solves a smaller QP problem formed by samples corresponding to nonzero Lagrange multipliers. A vast amount of papers discuss iterative training of support vector machines (e.g. [19, 5]). There are separate examples of on-going research [13, 15, 8] looking for effective and efficient solvers that can handle large data sets and improve scalability of machine learning methods which may require solving optimization problems.

The scope of our current effort is intentionally restricted to scaling up the batch version of classical one-class SVMs formulation [16] without having to change the optimization problem solver. We assume that the entire data set can fit into memory but we plan to extend our algorithm in the future to run online or in parallel. Moreover we pose the additional restriction of not training or building the model iteratively to reach certain objective [6].

The work most closely related to this one is the "simple decomposition method" idea presented in [20]. The key idea in [20] is to avoid the burden of general linear constraints from the optimization and convert it to a simple bound-constrained problem. In our formulation we do not get rid of any linear constraints. Instead we take advantage of the relationship between the set of linear constraints and the bound information of the design variables. To the best of our knowledge, none of the existing literature discusses formulating this non-trivial regularized approximation from prior knowledge of constraints in the optimization problem that leads to a sparse one-class SVMs. Our main contributions in this paper are:

- We propose an optimization problem with an additional meaningful criterion. The proposed formulation is acceptable and still equivalent to the classical SVM problem in terms of generalization error. The proposed formulation is very simple and can easily be implemented.
- We provide reasoning on why the proposed algorithm produces sparser solutions which in return improves the testing time by several factors.
- The proposed algorithm is several orders of magnitude faster than existing learning method and at the same time it retains the accuracy of the benchmark algorithm. We provide theoretical explanations for this.
- We demonstrate the capability of the algorithm in handling simulated data sets with varying sparsity and real life data from airlines industry by measuring the performance of the proposed technique using different metrics, such as frequency, accuracy, sensitivity, ranking,and run time.
- We provide some useful insights regarding the effectiveness of proposed technique based on the experimental and simulation study.

3 Preliminaries on Single Class Support Vector Machines

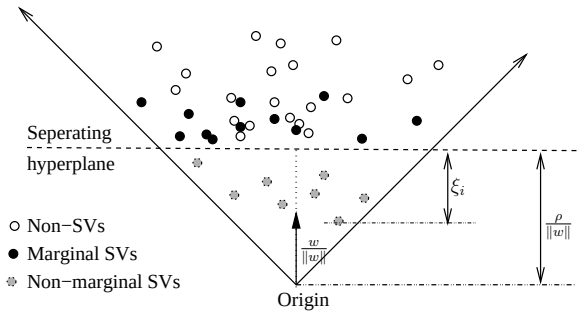

Figure 1: This figure illustrates the geometric interpretation of optimal hyperplane for one class Support Vector Machines.The empty circles, solid circles and the dotted circles represent non-support vectors, bounded support vectors and unbounded support vectors respectively.

Schölkopf [16] introduced one-class SVMs as an unique member of the SVMs family. As the name suggests, one-class SVMs is a unsupervised learning method which is trained on a single class and used for estimating the density of the target support objects. In standard one-class SVMs problem, we are given a set of labeled training data $\mathcal{D} = \{(\vec{x}_i, y_i)\}_{i=1}^{n}$ in the input space $\mathbf{R}$, where $\vec{x}_i \in \mathbf{R}^d$ and the corresponding labels $y_i \in \{+1\}$. The key idea is to construct a hyperplane that can separate outliers from the rest of the training examples, as shown in Fig. 1. At the end, we wish to develop a decision rule from the seen samples, so that when a new point comes in, we will be able to assign a class level depending on whether the model has seen this point or not. Since a $N-1$ dimensional hyperplane can exist in the N-dimensional feature space, the primary task is to find the optimal separating hyperplane that can maximize the margin between the training examples and the origin, which is the lone representative of the second class with negative label. This can be achieved by solving an optimization problem that leads to a set of training points, termed "Support Vectors" (SVs) which are the representatives of the decision boundary.

Let us define a function ϕ that can be used to map variables from the input space to the feature space $\mathcal{F}$, i.e. $\phi : \mathbf{R}^d \rightarrow \mathcal{F}$. In feature space the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ property holds, where $\mathbf{x}_i := \phi(x_i)$. While evaluating the dot product in the feature space, the explicit calculation using mapped feature ϕ can be avoided by simply evaluating the kernel function i.e. $k(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle$. However in order to do so, the chosen inner-product kernel must satisfy Mercer's theorem [4]. We will see an example of a normalized Longest Common Subsequence (nLCS) based kernel function later where we discuss our experimental studies.

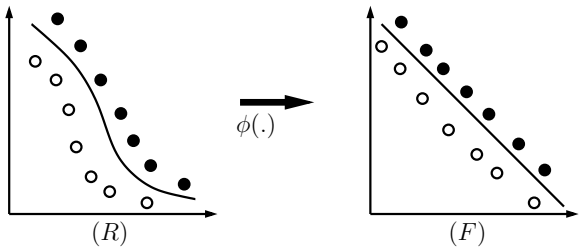

Figure 2: In this figure we provide the illustration of higher dimensional mapping for linear separation fields. It shows that even if the patterns are nonlinearly separable in input space, it is possible to map them in higher dimensional feature space where they may be linearly separable. Here $\phi(.)$ is the mapping function.

3.1 Derivation of the Optimization Problem:

In order to construct the optimal hyperplane we solve the following primal problem (Eqn. 3.1). The expression in Eqn. 3.1 simply means, "maximize the margin between the origin and the hyperplane (Fig. 1) for a nonseparable problem [16] in the feature space". The primal problem is represented as

$$\begin{aligned} \text{minimize} \quad & P(\mathbf{w}, \rho, \xi_i) = \frac{1}{2} w w^T + \frac{1}{\nu \ell} \sum_{i=1}^{\ell} \xi_i - \rho \\ \text{subject to} \quad & (\mathbf{w}.\phi(x_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \nu \in [0, 1] \end{aligned} \tag{3.1}$$

where ν is an user specified parameter that defines the upper bound on the training error, and also the lower bound on the fraction of training examples which are support vectors, ξ is the non-zero slack variable, ρ is the offset, $\phi(x_i)$ represents the transformed image of x_i in the Euclidean space and $i \in [\ell]$. The position of the optimal margin relative to the origin is represented by ρ, which in fact is the margin of separation between positive and negative class.

Using Lagrangian and some simple manipulations, the constrained primal problem (Eqn. 3.1) is converted to a dual problem [4],

$$\text{minimize} \quad Q = \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k\left(x_i, x_j\right)$$

(3.2)

$$\text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{\nu\ell}, 1 - \sum_i \alpha_i = 0, \quad \nu \in [0,1]$$

It is not difficult to show that $\rho = \sum_i \alpha_i k\left(x_i, x_j\right)$ for the solution w and pattern x_i corresponding to $0 < \alpha_i < 1$ while setting $\xi_i = 0$.

Weights to training points are Lagrangian multipliers ($\vec{\alpha}$) that ranges between 0 and 1. There exist at least $\nu\ell$ non-zero Lagrangian multipliers. Support Vectors (SVs) are training points $\{x_i : i \in [\ell], \alpha_i > 0\}$ with non-zero weights. Non-margin or bounded SVs are the ones with $\{x_i : i \in [\ell], \alpha_i = 1\}$ and margin or unbounded SVs are those with $\{x_i : i \in [\ell], 0 < \alpha_i < 1\}$.

Once $\vec{\alpha}$ is known, SVMs compute the decision function,

$$f(\vec{x_j}) = sign(\sum_{i\in\mathcal{I}_m}\alpha_i k(\vec{x}_i, \vec{x_j}) + \sum_{i\in\mathcal{I}_{nm}} k(\vec{x}_i, \vec{x_j}) - \rho)$$

(3.3)

where $\mathcal{I}_0 = \{i \ : \ \alpha_i = 0\}$, $\mathcal{I}_m = \{i \ : \ 0 < \alpha_i < 1\}$ and $\mathcal{I}_{nm} = \{i \ : \ \alpha_i = 1\}$ are the sets of indices of Lagrangian multipliers corresponding to non-SVs, marginal and non-marginal support vectors respectively. The pseudo-code of one-class SVMs algorithm is shown in Algorithm 1. Given a test point x_j , if $f(\vec{x_j}) < 0$, then x_j is predicted to be an outlier, whereas if $f(\vec{x_j}) \geq 0$, then x_j is predicted to be normal.

Algorithm 1 Single Class SVMs Algorithm

1: Input Vector: $X = \{x_1, x_2....x_m, z\}$, $X \in \mathcal{R}^d$.
2: Map Features: $K(\phi(x_i), \phi(x_j)))$.
3: Solve Eqn. 3.2 to obtain α corresponding to Support Vectors (SVs).
4: Calculate bias, $\rho = \sum_{k=1}^{Ns} \alpha_k K(\Phi(\vec{x})\Phi(\vec{x_k}))$.
5: Calculate score, $f(\vec{z}) = \sum_{k=1}^{Ns} \alpha_k K(\Phi(\vec{x_k})\Phi(\vec{z}))$.
6: **if** $f(\vec{z}) > \rho$ **then**
7: return 1
8: **else**
9: return 0
10: **end if**

4 The Multi-criterion Optimization

The multi-criterion optimization problem has several fascinating applications that compromise Economics, Engineering, Mathematics etc. Given a set of criteria $q(x) = \sum_i \lambda_i f_i(x)$ and a set of feasible points $\Omega \in \mathbf{R}^n$, the key idea is to find the optimal point $x \in \Omega$, for which $q(x) \leq q(z), \forall z$ from the feasible set. This can be expressed as,

$$\begin{aligned} &\min_{x\in\mathbf{R}^n} \quad q(x) \\ \text{subjected to} \quad &c_i = 0, i \in \varepsilon \\ &c_i \geq 0, i \in I \end{aligned} \tag{4.4}$$

where $c_i = 0$, $i \in \varepsilon$ are equality constraints and $c_i \geq 0$, $i \in I$ are inequality constraints. There are methods [10] that also find multiple solutions that cover the full set of possible trade-offs between the various objective functions. The selection of these criteria are typically based on the knowledge of optimal design or control variables, summary statistical, model assumptions, target objectives like smoothing, de-noising etc. A detailed description of techniques that take care of the trade-off between multiple criteria can be obtained in [2].

4.0.1 Bi-criterion Formulation: The Main Idea

To make the dual formulation more effective, we take into account the structure of the linear constraints and their dependencies on the variable bounds. We do this approximation by incorporating a second-order penalty function, keeping in mind the description of support vectors and the properties of the associated Lagrangian multipliers/weights. The bi-criterion formulation of one-class SVM takes the form of,

$$\min_{\alpha\in\Re^n} \quad Q = \frac{1}{2}\alpha^T K\alpha - \lambda(\frac{1}{2\nu\ell}\vec{1} - \alpha)^T(\frac{1}{2\nu\ell}\vec{1} - \alpha)$$

$$\text{subject to} \quad 0 \leq \alpha \leq \frac{1}{\nu\ell}\vec{1}, \vec{1}^T\alpha = 1, \quad \nu \in [0,1] \tag{4.5}$$

where α is the vector of Lagrangian multipliers and K is the similarity matrix. The motivation behind the additional penalty term is that the bi-criterion formulation seeks the values of the design variables closest to the extreme (upper or lower) bounds of the design variable while simultaneously minimizing the first term. Only training points with non-negative weights are considered as support vectors. It is very intuitive that the equality constraints are satisfied with the least number of design variables only when the weights corresponding to those variables tend to be close to the maximum possible value (i.e. $\alpha_i = \frac{1}{\nu\ell}$). Hence by solving the above problem we expect to obtain a sparse solution. In the following sections we will see that the quadratic penalty function is compatible with the method of direction search and plays a significant role to reach the optimal solution using less computations.

PROPOSITION 4.1. *Bi-criterion formulation (Eqn. 4.5) of SVMs is convex.*

Proof. Solving this optimization problem means that we need to minimize two convex criterion on a defined set:

- The Hessian of the objective function Q(in Eqn. 3.2) of classical One-class SVMs problem is given by $\nabla_x^2 Q(x) = K$, where $K \in S_+^n$ is a symmetric kernel matrix. Since we make sure that the defined kernel matrix is positive definite or positive semi-definite, it implies that the objective function is either strictly convex or convex.

- Since the controlled criterion takes the form of a squared Euclidean norm $h = (\frac{1}{2\nu\ell}\vec{1} - \alpha)^T(\frac{1}{2\nu\ell}\vec{1} - \alpha)$, h is strictly convex.

- Given $0 \leq \alpha_i \leq \frac{1}{\nu\ell}$, the constraint in Eqn. 3.2 defines convex set as $\sum_i \alpha_i$ is convex.

Here we will briefly discuss the nature of solutions that bi-criterion formulation may yield. With the control parameter $\lambda = 0$ (Eqn. 4.5), we would get the classical solution. However with a non-zero control parameter, (say $\lambda = 1$), the quadratic term leads to sparser solutions. Suppose we are given ℓ training samples and model parameter $\nu \in [0, 1]$, and define $p = \nu\ell$. The upper bound of the constraint (Eqn. 3.2) is $\frac{1}{p}$. The second order term of the objective function attains its maxima at 0 and $\frac{1}{p}$ and therefore, the solution will tend to push the α's toward the extreme values in the range. Since $\sum_i \alpha_i = 1$ and α_i can attend a maximum value of $\frac{1}{p}$, we can, without the loss of generality, decompose the previous expression as, $\sum_{i=1}^{N} \alpha_i = \sum_{i=1}^{p} \alpha_i + \sum_{i=p+1}^{N} \alpha_i = p\frac{1}{p} + 0 = 1$. Hence the solution is a set of p training inputs with maximum weights i.e, $\alpha_1 = \alpha_2 = \alpha_3 = \cdots = \alpha_p = \frac{1}{p}$. If p is not an integer, it is rounded to the nearest integer value (say $\hat{p}$) and the above process is repeated. This results in $\hat{p} - 1$ design variables attaining the upper bound and thus forcing the remaining ones to take any values from the range defined by $0 \leq \alpha_i \leq \frac{1}{p}$ such that $\sum_i \alpha_{i=1}^{\hat{p}} = 1$ is satisfied. Therefore with a λ which is large enough, the optimization is pushed toward a solution that is more sparse than the classical solution.

4.1 Active Set: The Quadratic solver "Active set" algorithm [9, 15] is very popular in solving QP problems with constraints, especially when the positive semi-definite matrix K is dense in nature. Equation 4.5 can be rewritten as,

$$\begin{aligned} &\min_{\alpha \in \Re^n} \quad Q = \frac{1}{2}\alpha^T \hat{K} \alpha + C^T \alpha \\ &\text{subject to} \quad 0 \leq \alpha \leq \frac{1}{\nu\ell}\vec{e}, \vec{e}^T\alpha = b, \quad \nu \in [0, 1] \end{aligned} \tag{4.6}$$

where $\vec{e} = \vec{1}$, $b = 1$, $\hat{K} = K - 2\lambda I$, $C = \frac{\lambda}{\nu\ell}\vec{e}$. The optimization problem defined above is a quadratic programming problem with a linear set of constraints and we would like to solve this problem in a finite number of steps using "Active set" algorithm. In active set algorithm, the first step is to compute a feasible start point which satisfies both the bounds and the equality constraints. Given a feasible start point α_0, the task is to iteratively minimize the objective function. However this requires us to find the suitable direction of search and a non-negative step size.

Definition At any α, the active set $\mathcal{A}(\alpha)$ consists of free variable indices from the equality constraints together with the indices of variables which are temporarily fixed on their upper/lower bounds.

4.1.1 Reduction of Problem Size At any k^{th} iteration, suppose we have some α_k. We would like to create a partitioning of the active set. If "X" refer to entities corresponding to design variables whose values are temporarily fixed and the complement set of variables, termed as free variables, are denoted by "R", we can create a partitioning of current points α_k i.e. $\alpha_k = [\alpha_k^R \alpha_k^X]$ and $n = [n^X n^R]$ where n is the cardinality of the design variable. Similarly we can also define the partitions is $A = [A^R A^X]$ and $C = [C^R C^X]$. We can also define,

$$\hat{K} = \begin{bmatrix} \hat{K}^{R,R} & \hat{K}^{R,X} \\ \hat{K}^{X,R} & \hat{K}^{X,X} \end{bmatrix}.$$

where $\hat{K}^{X,R} = (\hat{K}^{R,X})^T$. At iteration k, we can define a working set $\mathcal{W}_k$ which is constructed by t equality constraints only. Temporarily discard all the fixed variables so that we end up with $n = n^R$ and $t = m_\ell$, where ℓ denotes the total number of equality constraints. The direction of search is computed by solving the following reduced problem,

$$\begin{aligned} &\min_{\alpha \in \mathbf{R}^n} Q = \frac{1}{2}{\alpha_k}^{R^T} \hat{K}^{R,R} \alpha_k^R + {C^R}^T \alpha_k^R \\ &\text{subjected to} \quad 0 \leq \alpha_k^R \leq \frac{1}{\nu\ell}\vec{e}, {A^R}^T \alpha_k^R = b \\ &\qquad\qquad -{A^X}^T \alpha_k^X, \nu \in [0, 1] \end{aligned} \tag{4.7}$$

Once the reduced problem is formed, the next task is to check if $Q(\alpha^R)$ is minimized for the given α_k^R and

$\mathcal{W}_k$. If $Q({\alpha_k}^R)$ is not minimized, we need to compute the direction and the step size such that $Q({\alpha_{k+1}}^R) \leq Q({\alpha_k}^R)$. The pseudo code of the algorithm to compute the direction and the step size is shown in Algorithm 2. The bi-criterion formulation tends to push the α's toward the extreme values in the range. However the optmization prefers α_i to attend the maximum value of $\frac{1}{p}$ to maintain a finite step size in the suitable direction. As a consequence, the number of bounded variables quickly increases, thus resulting in a much smaller problem (Eqn. 4.7) to solve. The reduced QP problem (step-3, Algorithm 2) can be solved using elimination of variables or Lagrangian Methods.

Algorithm 2 Sub-problem of active set algorithm

1: Input: ${\alpha_k}^R, \hat{K}^{R,R}, C^R$. Let direction is denoted by ${d_k}^R = {\alpha_{k+1}}^R - {\alpha_k}^R$ and ${g_k} = {\alpha_k}^{R^T}\hat{K}^{R,R} + C^R$.
2: $Q({\alpha_{k+1}}^R) = Q({\alpha_k}^R + {d_k}^R) = \frac{1}{2}({\alpha_k}^R + {d_k}^R)^T\hat{K}^{R,R}({\alpha_k}^R + {d^R}_k) + C^{R^T}({\alpha_k}^R + {d^R}_k) = Q({\alpha_k}^R) + \frac{1}{2}{d_k}^{R^T}\hat{K}^{R,R}{d_k}^R + {g_k}^T{d_k}^R$.
3: Modified sub-problem
$min_d \quad \frac{1}{2}{d_k}^{R^T}\hat{K}^{R,R}{d_k}^R + {g_k}^T{d_k}^R$
subjected to, ${A^R}^T{d_k}^R = 0$
4: **if** ${d_k}^R \neq 0$ **then**
5: Calculating step size along the direction ${d_k}^R$
6: **if** ${\alpha_k}^R + {d_k}^R$ is feasible **then**
7: set ${\alpha_{k+1}}^R = {d_k}^R + {\alpha_k}^R$
8: **else**
9: set ${\alpha_{k+1}}^R = \gamma_k{d_k}^R + {\alpha_k}^R$, step size $\gamma_k \in [0, 1]$
10: **end if**
11: **else**
12: Check for KKT condition
$$\begin{bmatrix} \hat{K}^{R,R} & A^R \\ {A^R}^T & 0 \end{bmatrix} \begin{bmatrix} {d_k}^R \\ -\theta \end{bmatrix} = -\begin{bmatrix} {\alpha_k}^{R^T}\hat{K}^{R,R} + C^R \\ 0 \end{bmatrix}$$
13: **end if**

5 Experiments and Discussions

In this section we conduct computational experiments of bi-criterion SVMs and present some studies comparing bi-criterion and classical SVMs. In our analysis, we considered two very different data sets: one real-world FOQA (Flight Operations Quality Assurance) data and another simulated data set as benchmark applications. The aviation data is representative of one of the most complex engineering systems with very large size and dimensionality. Such a domain also poses a real challenge in identifying anomalies in high-dimensional, multivariate data sets containing discrete, categorical, and continuous features. Therefore it is an ideal platform to test the accuracy and scalability of anomaly detection algorithms. The simulation based study was proposed to conduct a proof- of-concept analysis that demonstrates the performance and effectiveness of the proposed bi-criterion algorithm under different test conditions. Both bi-criterion and classical one class SVMs algorithms were tested on Linux cluster that comprised of 16 slave nodes, each of which is a dual processor $1-U$ server containing two, quad-core Intel Xeon processors @ $2.66GHz$ totaling 128 cores and $128GB$ Ram ($1Gb/Core$). It is controlled by two master nodes and has $30Tb$ storage. Under each test condition, the design variable of the optimization from bi-criterion and classical one class SVMs were initialized with the same random set to preserve consistency.

5.1 Airlines Data: A Realistic Scenario The real world data set chosen for analysis is from a commercial airlines. The data is obtained from medium range narrow body passenger aircraft. In our current analysis we considered a total of 2048 flights, a small subset of which landed at the same airport. Each flight consists of 365 parameters acquired at 1 Hz. Our working data set consists of the decent portions of the flight from 10,000 ft to touch-down (average flight length of $10K$ samples) and has 104 discrete and 45 continuous parameters which were selected based on domain experts feedback. For continuous data, each parameter in the training and testing data are z-score normalized using the statistics of each parameter calculated across all training flights. The continuous and discrete data is converted to continuous and discrete sequences respectively. Once the sequences are generated the continuous and discrete kernel are separately computed pairwise across all possible flight combinations in the training set. For pairwise comparison we used longest common subsequence based similarity function (Eqn. 5.8).

$$K(\vec{x}_i, \vec{x}_j) = \frac{|LCS(\vec{x}_i, \vec{x}_j)|}{\sqrt{l_{\vec{x}_i} l_{\vec{x}_j}}}, \tag{5.8}$$

where $l_{\vec{x}}$ is the number of symbols in sequence $\vec{x}$. Given two sequences the common subsequences of sequences $\vec{x}_i$ and $\vec{x}_j$ is identified. The longest such subsequence of $\vec{x}_i$ and $\vec{x}_j$ is called the longest common subsequence (LCS) and is denoted by $LCS(\vec{x}_i, \vec{x}_j)$ and $|LCS(\vec{x}_i, \vec{x}_j)|$ is its length.

Once the kernels are generated, we combine them in a convex fashion. Algorithm 3 shows the operations to generate the kernel. For details see the original paper [7] where we demonstrated Multiple Kernel Anomaly Detection Algorithm (MKAD) algorithm that can detect if

the discrete pilot inputs combined with the observation vector are nominal or off nominal.

Algorithm 3 Pre-processing steps to generate a kernel

1: Continuous Input : $C = \{x_{1c}, x_{2c}....x_{mc}, z_c\}$, $C \in \mathcal{R}^d$, Discrete Sequence Input : $S = \{x_{1s}, x_{2s}....x_{ms}, z_s\}$, $S \in \mathcal{R}^{\widetilde{d}}$.
2: Generate Continuous Sequence: $\{x_{1q}, x_{2q}....x_{mq}, z_q\} = SAX\left(\{x_{1c}, x_{2c}....x_{mc}, z_c\}\right)$ [?].
3: Generate Continuous and Discrete Features: $\{\phi(x_{1q}), \phi(x_{2q}), \cdots \phi(x_{mq}), \phi(z_q)\}$ and $\{\phi(x_{1s}), \phi(x_{2s}), \cdots \phi(x_{ms}), \phi(z_s)\}$.
4: Combine kernel: $\beta_q K_q(\phi(x_{iq}), \phi(x_{jq})) + \beta_s K_s(\phi(x_{is}), \phi(x_{js}))$.

Active-set algorithm has been used to solve the quadratic problem. Through out this experiment, some of the user defined inputs for example, kernel matrix, initialization vector, ν parameter, stopping criteria, etc., were kept consistent for both the algorithms. From run to run, the design variables were randomly initialized with values between 0 and 1. However for any particular run both the algorithm started from the same initial point. In the first set of experiments, both models were built with training sizes varying from 200 samples up to 2000 sample points with $\nu = 0.05$ and the number of support vectors were recorded for each case. These results are unique and reproducible for the given data and parameter settings. Figure 3 shows that bi-criterion SVMs always produces fewer support vectors than the classical approach for different training sizes and the reduced set size is typically the lower bound of the number of total support vectors i.e. ν times the number of the training points.

Figure 4 compares the distribution of the weights (α_i for $i \in [l]$) corresponding to the support vectors for a case where we used 2000 sample points for training and set $\nu = 0.05$. For classical SVMs there are more instances where weights are scattered in-between the bounds. However for bi-criterion formulation we see all weights lie on the upper bound. Fig. 3 and Fig. 4 complement each other and show that our algorithm produces fewer support vectors by forcing weights toward the upper and lower bounds. In summary, with majority of the Lagrangian multipliers/weights on the upper bound, the model results in a much reduced set of non-zero weights.

Here we extend our observation from Fig. 3. We have seen that bi-criterion SVMs results sparser solution when compared to classical model. The analysis (Fig. 4) showed that classical solution consists of 331 non-zeros weights while bi-criterion SVMs produced 308 which is exactly $\nu\times$ 2048, the number of training samples. An initial investigation found that solutions from both these methods have a total of 304 support vectors in common and that jointly compromises approximately $97-98\%$ of the total weights (see the linear constraint in Eqn. 3.2) which is unity. To account for the remaining weight, bi-criterion SVMs proposes 4 unique SVs while the classical assigns 27 SVs which can very well be some source of redundancy. These results are summaries in table 1.

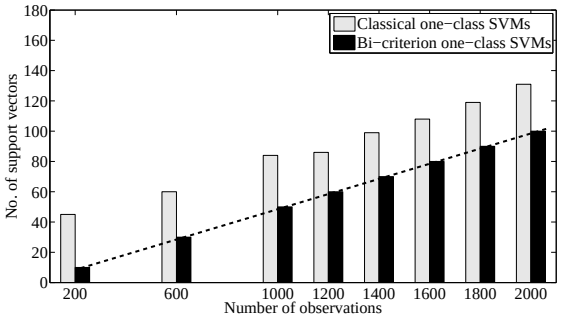

Figure 3: Figure comparing the number of support vectors obtained from the bi-criterion and classical SVMs technique for different training sizes over a single run. For each and every run, bi-criterion formulation converges with a sparser solution and thus outperformed classical SVMs formulation.

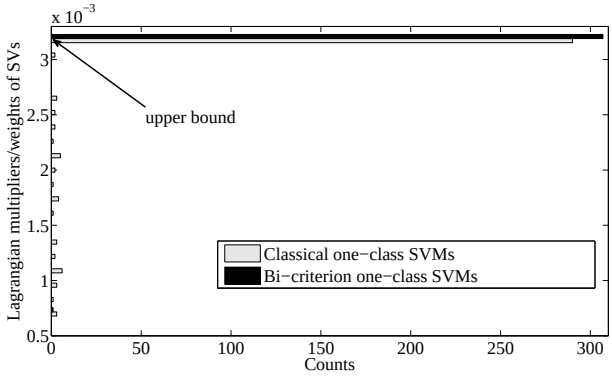

Figure 4: In this figure we compare the distribution of the weights of support vectors obtained from the classical and bi-criterion SVMs. We can observe that most design variables in bi-criterion formulation corresponds to the "upper bound" (i.e. $\frac{1}{\nu\ell}$, see Eqn. 3.2). For classical SVMs there are some instances where the design variables hold values between upper and lower bounds.

Table 1: Here we compare classical and bi-criterion SVMs to check the presence and the influence of redundancy. The analysis showed that 304 indices (of Support Vectors) are common in both solutions and they jointly compromises approximately $97-98\%$ of the total weights. To account for the remaining weight, classical SVMs uses approximately 7 times the number of unique SVs used by bi-criterion SVMs.

Algorithms	Overlapping index (influence)	Unique index (influence)
MKAD	304 (97.17%)	27 (2.83%)
Bi-criterion MKAD	304 (98.7%)	4 (1.3%)

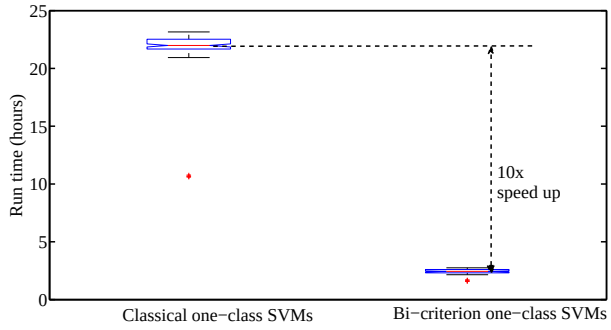

Figure 5: The figure shows the run time analysis for classical and bi-criterion SVMs under different initialization conditions. This experiment was repeated 100 times with random initializations and the running time were recorded. These are observed run times with random initializations. It is clear that bi-criterion formulation performs much better compared to classical SVMs.

In Fig. 5 we show the resulting training time (in hours) for the exact solution and bi-criterion formulation with 2000 sample points as training points and $\nu = 0.15$. In the box plot, we show the mean training time over 100 runs and their corresponding error bars. The mean rum times are 21.64 hours and 2.43 hours for classical and bi-criterion SVMs respectively. The standard deviations are 2.4 and 0.23 for the respective models. It can be observed that the proposed formulations consistently performs on average 10 times faster than the classical one-class SVM model for the given model parameter settings. This performance gain factor is expected to increase with increasing training set size. In a separate experiment, we repeated the same case with randomly initializations from the feasible region defined by the bound constraints. This led us to further gain in run time. This is because the optimization routine does not spend any time looking for a initialization set from the feasible region. However this observation is true for both the algorithms.

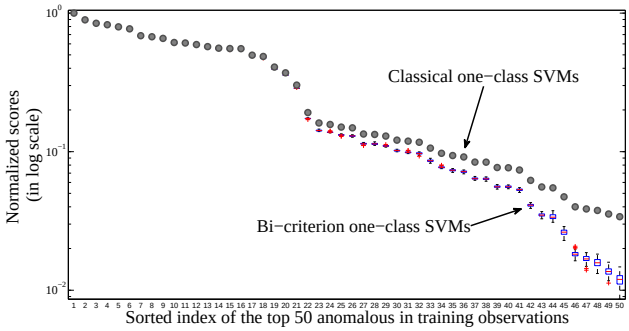

Figure 6: Normalized scores of the top 50 abnormal entries detected in the FOQA training set data. Both the scores were arranged in a descending order of the classical algorithm's score. This experiment was repeated 100 times with random initializations. This figure shows that the bi-criterion algorithm almost always orders data points the same way as the classical algorithm.

In this section we present some results on prediction performance. In this analysis, we asked both the models to predict the top 50 outliers from the training pool of 2048 and we compare their associated outlier scores and ranking. We sorted the outliers and thereafter normalized them to 1. In Fig. 6 we compare the mean score with associated error bars from multiple runs in log scale. This experiment was repeated 100 times with random initializations. Figure 6 clearly shows that bi-criterion SVMs correctly predicts and ranks the points in terms of their outlierness in a consistent fashion and the outcome is very comparable to observations from classical one-class SVMs.

We have conducted an initial study that describes the nature of the solution we obtain for varying λ. In the bi-criterion formulation, the value of the λ decides which criterion is weighted more. In Fig. 7, we plot the number of support vectors for a wide range of λ values. The case when we obtain maximum number of support vectors is for $\lambda = 0$ and we normalize the entire outcome using the maximum count. What we observed is, the number of support vectors drastically changes (7% change) as we start increasing λ from 0 but as λ becomes large enough (greater than 0.5) the influence of λ on the outcome diminishes and the

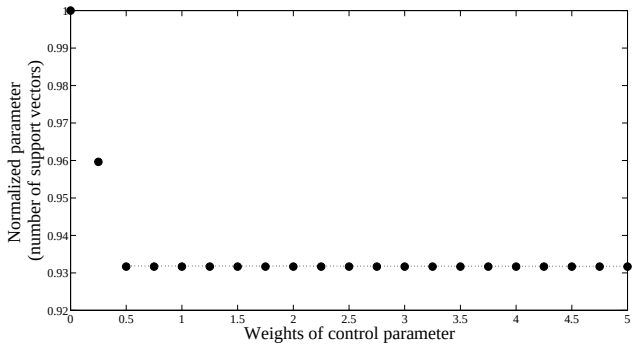

Figure 7: Figure demonstrating the influence of the control parameter λ on the performance of the bi-criterion SVMs algorithm. There is very small influence of the control parameter (λ) on the multi-criterion optimization outcome for $\lambda \geq \frac{1}{2}$.

sparsity of the solution is steady for the given data and parameter settings. We therefore set $\lambda = 1$ for all of our experiments..

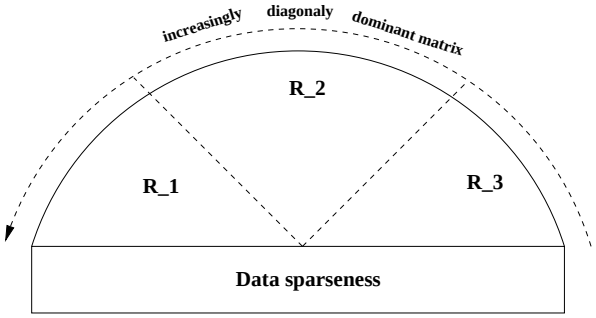

Figure 8: A cartoon that represents various levels of sparseness that can be observed in the kernel matrix $K(x_i, x_j)$. Region-1 (R_1) and Region-3 (R_3) represent extreme scenarios. When all entries of x and y are very different from each other and unique, the resultant similarity matrix is strictly diagonally dominant (R_1). With very similar x and y the K matrix will be very dense with very similar diagonal and off-diagonal elements (R_3). Region-2 (R_2) represents a case when the diagonal and off-diagonal elements are distributed over a certain range.

5.2 A Simulated Study: To test the robustness of the bi-criterion formulation, we developed a common test platform with a set of diverse test scenarios using synthetic data. Till now we have studied the influence of the quadratic penalty function (Eqn. 4.5) and the control parameter on the outcome. For problems of this nature, the property of the kernel matrix (K) in Eqn. 4.5 plays an important role. The implicit mapping into feature space based on different similarity functions and data sets are bound to conceal different types of density structures in the kernel matrix. The main optimization algorithm involves quadratic programming which learns on these kernel matrices. Here we intend to investigate the influence of varying kernel density on model performance and outcome. When the entries of the input data x_i and x_j are very different from each other and unique, the resultant similarity matrix is strictly diagonally dominant i.e. $|K(x_i, x_i)| > \sum_{i \neq j} |K(x_i, x_j)|, \forall i$. The other extreme scenario is when all the entries x and y are very similar in feature space and tightly clustered. The latter will result in a highly dense K matrix. In Fig. 8, we explain the above scenarios in a cartoon form. Region 1 (R_1) and Region 3 (R_3) represent the extremely sparse and highly dense cases, respectively. Region-2 (R_2) represents a case when the diagonal and off-diagonal elements are distributed over a certain range.

We will further illustrate the above scenarios of varying sparseness by using synthetic data set. This data is randomly generated from the normal distribution with user defined mean parameter μ and standard deviation parameter σ. The resultant kernel matrices we generate are symmetric and of size 2048. We force the diagonal elements to unity, as this is case for most similarity functions (e.g. nLCS function shown in Eqn. **??**) which vary between 0 and 1, where 1 represents the highest similarity or exact match. For each combination of μ and σ, we binned the elements of K into 50 equally spaced groups each of which represents a "values range" between 0 and 1. In returns we obtain the number of elements in each group. In the analysis, we conducted a total of 20 different cases where the density distribution of the matrix moves from one end of the "values range" to the other end. Figure 9 presents some of these examples. Subfigure **??**-(a) shows an example where all the elements of the kernel matrix, beside the diagonals, are of extremely small. This is a typical example of diagonally dominant matrix. Subfigure **??**-(c) is the other extreme case with very comparable diagonal and off-diagonal elements. Subfigure **??**-(b) and Subfigure **??**-(d) represent a simulated and a realistic (aviation data) scenario where the off-diagonal elements of the matrix K hold values from intermediate ranges. Under each test condition, we ran 10 experiments with random initializations and recorded the run time and number of support vectors for bi-criterion and Classical SVMs algorithm. To measure the effectiveness of the model, we

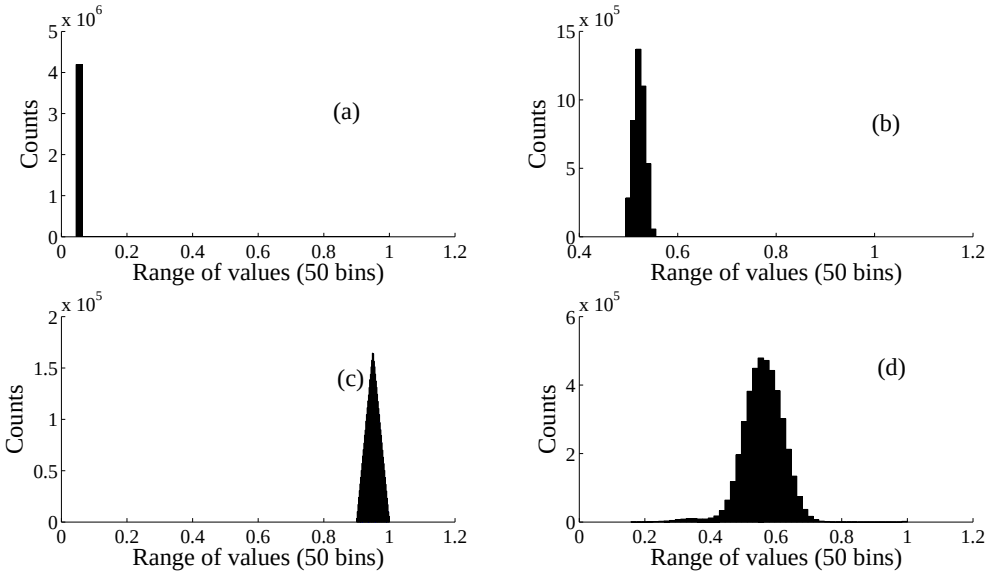

Figure 9: The above figure represents the histogram plots comparing the distributions of the elements of the kernel K under different test cases. Each element of K represents the similarity between two entities. The diagonal elements represent self similarities. Subfigure (a-c) was obtained from simulated data while subfigure (d) was obtained from a real airlines FOQA data consisting of 2048 flights with 149 continuous and discrete features. In subfigure (a) the kernel K is a diagonally dominant matrix while subfigure (c) represents the case where the diagonal and off-diagonal elements of the matrix K are very comparable. Subfigure (b) represents an intermediate scenario.

define the following two performance metrics.

Definition We defined degree of sparsity, a gain metric, that calculates the ratio of the counts of non-zero Lagrangian multipliers from bi-criterion to that of classical SVMs model. This metric explain how effective is the proposed model in testing phase. If degree of sparsity if high this simply implies that the solution is obtained with lesser support vectors.

Definition The execution time gain is a run time related gain metric, that calculates the ratio of the run time of bi-criterion to that of classical SVMs model. This metric shows how quick is the proposed optimization converges to a solution compared to the benchmark method.

Figure 10 summarizes the results using the performance metrics in a quadrant format for a better visual understanding. The two right hand quadrants always confirm a sparse solution. Similarly the upper two quadrants indicates seep up by some factors. Anywhere in the $+/+ve$ quadrant is the most desired operating region where under any circumstances the proposed solution is sparse and the execution time is less. A negative execution gain means that the classical solution converges faster. In Fig. 10, the execution time gain is intentionally plotted in log scale to obtain a better resolution in order to understand the differences in the performance space. As can be seen, bi-criterion formulation outperforms the benchmark algorithm considerably in most cases. On all occasions, bi-criterion SVMs always reported the lest number of support vectors (i.e. νN) but the solution of the classical method changes depending on the density structure of the kernel matrix. For instance, from diagonally dominant matrix (refer Fig. 8 region -1 and Fig. 9 -(a)) classical SVMs reports N support vectors which is equal to the number of training points. This is because all the training examples are so different and unique that all of them carries equal weightage to be a support vectors. However the convergence time of the classical approach was varying a lot. Under this scenario, there were several occasions where bi-criterion ran slower than the classical SVMs by a couple of factors. But majority of the gain was noticed during test phase where to evaluate a single test point the classical will have to do at least $\frac{1}{\nu}$ times more operations. On the other hand for highly dense kernel matrix with very similar diagonal and off-diagonal elements (refer Fig. 8 region -3 and Fig. 9

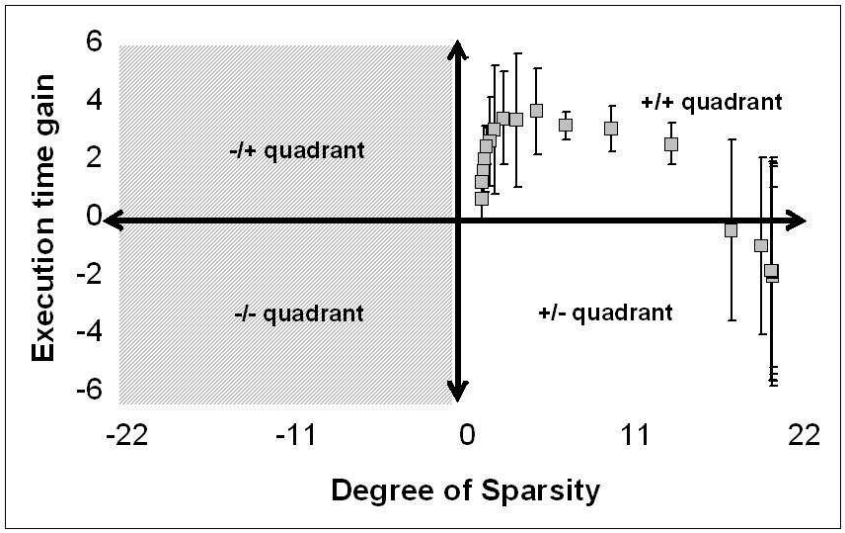

Figure 10: Performance comparison between bi-criterion and classical SVMs method. The execution time gain is in log scale.

-(c)), it takes extremely long time for the classical to converge but the solutions are very similar to those obtained by bi-criterion formulation. Under this scenario, the gain is in the run time while building the model. For any other cases the importance of regularization term was reflected. There are the test cases (refer Fig. 8 any combination of region -1, region -2 and region -3 and Fig. 9 -(b) and (d)) where the proposed algorithm outperforms the baseline in run-time by several factors and also results in a sparse solution. In reality, this is exactly one expects from an algorithm that can learn much faster and produce sparser solution so that the model can be used to test large volume of data in short time. Obviously, this sort of scaling will be very attractive for high-dimensional and dense data matrix, particularly when the detection accuracy is well preserved.

6 Conclusion

In this paper we devised a version of one-class SVMs with an addition to the objective function that leads to sparser solutions. We demonstrated that these solutions are nearly as accurate as the solutions from the classical one-class SVM algorithm but are obtained in much less time and/or can be used to classify new examples in much less time. We demonstrate that the reduced number of support vectors and the resulting reduction in running time that we obtain are not sensitive to λ which is the weight used to control the tradeoff between the two terms in our objective function. In combination with our earlier development of MKAD [7], we are able to identify anomalies in data from commercial aviation accurately and in a practical amount of time without losing any of the advantages of kernel methods such as global optimality for a given training set.

We plan to investigate further efficiency and scalability improvements by developing distributed and online versions of our algorithm. Because our algorithm only involved a simple change to the objective function and did not require any changes to the solver, we can utilize any other solvers used for SVMs. We plan to investigate how strong our efficiency improvements remain when using other solvers.

Acknowledgments

This work was supported through funding from the NASA Aeronautics Researchh Mission Directorate, Aviation Safety Program, Integrated Vehicle Health Management project. The authors thank Bryan Matthews for valuable discussions and suggestions.

References

[1] S. Asharaf, M. Narasimha Murty, and S.K. Shevade. Cluster based training for scaling non-linear support vector machines. *International Conference on Computing: Theory and Applications*, 0:304–308, 2007.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] Chris J.C. Burges and Bernhard Schlkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems 9*, pages 375–381. MIT Press, 1997.

[4] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[5] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning, 2000.

[6] Santanu Das, Kanishka Bhaduri, Nikunj C. Oza, and Ashok N. Srivastava. nu-anomica: A fast support vector based novelty detection technique. *Data Mining, IEEE International Conference on*, 0:101–109, 2009.

[7] Santanu Das, Bryan L. Matthews, Ashok N. Srivastava, and Nikunj C. Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 47–56, New York, NY, USA, 2010. ACM.

[8] Gary William Flake and Steve Lawrence. Efficient svm regression training with smo, 2001.

[9] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Trans. Math. Softw.*, 10(3):282–298, 1984.

[10] Yaochu Jin, editor. *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, 2006.

[11] Liang Lie-quan and Liang Ying-hong. Sample clustering for fast classification by using the mean shift procedure. In *ISECS '09: Proceedings of the 2009 Second International Symposium on Electronic Commerce and Security*, pages 179–183, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient ?

[13] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[14] Ya-Li Qi, Wei He, and Hou Shu. An optimized approach on reduced kernel matrix to clustersvm. pages 1446 –1449, aug. 2008.

[15] Katya Scheinberg. An efficient implementation of an active set method for svms. *J. Mach. Learn. Res.*, 7:2237–2257, 2006.

[16] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, 2001.

[17] Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate gaussian process regression. In *In*, page 953. MIT Press, 2002.

[18] Sren Sonnenburg, Gunnar Rtsch, Bernhard Schlkopf, and Gunnar Rtsch. Large scale multiple kernel learning. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:2006, 2006.

[19] S. V. N. Vishwanathan, Alexander J. Smola, and M. Narasimha Murty. Simplesvm.

[20] Chih wei Hsu and Chih-Jen Lin. A simple decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:291–314, 1999.