2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

An ordinal kernel trick for a computationally efficient support vector machine

Yara Rizk Nicholas Mitri Mariette Awad

***Abstract*—A principled approach to machine learning (ML) problems because of its mathematical foundations in statistical learning theory, support vector machines (SVM), a non-parametric method, require all the data to be available during the training phase.. However, once the model parameters are identified, SVM relies only, for future prediction, on a subset of these training instances, called support vectors (SV). The SVM model is mathematically written as a weighted sum of these SV whose number, rather than the dimensionality of the input space, defines SVM's complexity. Since the final number of these SV can be up to half the size of the training dataset, SVM becomes challenged to run on energy aware computing platforms. We propose in this work Knee-Cut SVM (KCSVM) and Knee-Cut Ordinal Optimization inspired SVM (KCOOSVM) that use a soft trick of ordered kernel values and uniform subsampling to reduce SVM's prediction computational complexity while maintaining an acceptable impact on its generalization capability. When tested on several databases from UCI, KCSVM and KCOOSVM produced promising results, comparable to similar published algorithms.**

***Keywords*—SVM; sparse decision rules; ordinal optimization; real time testing; supervised and binary classification**

I. Introduction

A sparse kernel and maximum margin machine learning (ML) approach, support vector machine (SVM) has found its way into a myriad of applications [1] since proposed by Vapnik [2]. Deeply rooted in statistical learning theory, SVM writes the optimal separating hyper plane that is equidistant from the classes as a weighted sum of a subset of the training set, referred to as support vectors (SV). Found after an optimization step involving an objective function regularized by an error term and a constraint using the Lagrangian relaxation, the final number of SV, which can be up to half the size of the training dataset, is data dependent and varies based on the data complexity, which is captured by the data dimensionality and the class separability, making SVM computationally too expensive real time prediction on platforms that are power challenged.

Motivated to develop an energy aware SVM model to be deployed on resource challenged computing terminals such as mobile and handheld devices, we propose in this paper, Knee-Cut SVM (KCSVM) and Knee-Cut Ordinal Optimization inspired SVM (KCOOSVM), two novel algorithms that attempt, using soft computing concepts and the kernel trick, to reduce the time and resources needed by SVM to perform online prediction while minimizing the additional overhead needed during supervised training without a significant loss in accuracy. KCSVM extracts boundary vectors in kernel space by retaining the vectors that have a distance less than a threshold, automatically computed and problem specific. Unlike [3], KCSVM and KCOOSVM use the kernel value as a distance measure between two vectors in kernel space and compute only the distances between vectors of different classes to minimize computations. A uniform sampling of the ordered kernel space allows the selection of the training set for KCOOSVM, which proved to be comparable in accuracy to existing methods while preserving a repeatable reduced model over a few databases from UCI [4]. Although these algorithms can be applied as a preprocessing block to any SVM training algorithm, most of the experiments were performed using libSVM's [5] implementation which is considered by many the most popular and widely used implementation [6-7].

The remainder of this paper is such that a literature review of relevant published work is presented in Section 2. KCSVM and KCOOSVM are detailed, after briefly discussing the basic concepts of SVM in Section 3. Section 4 presents our experimental results and Section 5 concludes the paper with a summary of the obtained results.

II. Literature Review

Many researchers have investigated computational improvements for this optimal and robust classifier to obtain a sparser decision rule.

Burges addressed SVM's slow online prediction by reducing the complexity of the decision rule using a post processing algorithm which finds an approximation to this complex decision rule, represented by a computed reduced set of vectors with predefined cardinality [8]. Although promising, Burges' algorithm is not easy to implement and does not provide control over the resulting prediction accuracy [9]. Instead, [9] uses SV regression machines to approximate the hyper plane, obtained by training a standard SVM, by a subset of these SVs.

Ref. [10] solves SVM's optimization problem in the primal or dual formulation using a cutting planes based algorithm which results in efficient training and sparser decision rules.

Ref. [11] retains the linearly independent SVs, produced by the SMO solver, using the row reduced echelon form. Although acceptable for polynomial kernels and RBF with large sigma values, the cardinality of the removed SV is kernel dependent. Ref. [12] iteratively replaces two nearest SV belonging to the same class by a constructed SV.

Ref. [13]'s cross-training SVM divides the training database in n equal sets, trains n SVM independently, and

Yara Rizk, Nicholas Mitri and Mariette Awad are with the Electrical Engineering Department, American University of Beirut, Beirut, Lebanon (e-mail: {yar01, ngm04, mariette.awad}@ aub.edu.lb).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

retains training points whose average margin, defined as the average sum of the n SVM models' predicted label for this point, is between 0 and 1, to train the final SVM model. Inspired by cross-training SVM, [14] proposed separable case approximation (SCA) algorithm which approximates SVM's decision rule by running a hard margin SVM on the separable SVs in kernel space.

Reduced SVM (RSVM) randomly selects a subset of the training data to solve the SSVM optimization problem [15]. Averaging results over multiple runs, RSVM resulted in a lower prediction accuracy which decreased further as the reduction in SV set cardinality increased. Ref. [16] attempted to improve on RSVM by training on margin vectors, identified by computing the self and the mutual center distances in feature space and eliminating statistically insignificant points based on the center distance ratio of these points.

Ref. [3] identified boundary vectors using k-nearest neighbor (kNN); the distance between each vector to all other vectors in input or kernel space is computed and the vectors that have among their k nearest neighbors a vector of opposing class are retained.

Ref. [17] clustered the trained data using k-means and trained on the cluster heads. The reduction is controlled by the maximum number of allowable clusters. LMSVM clusters the training set and retains the clusters with a high heterogeneity score [18]. Ref. [19] reduced the training set by clustering the training set using k-means, identifying the cluster heads and crisp clusters (clusters with points from the same class), then modifying clusters to eliminate points that are less likely to affect the decision plane.

III. KCSVM AND KCOOSVM

A. SVM Overview

SVM is used in classification problems to find an optimal hyper plane separating two classes. This optimal solution is obtained by solving the optimization problem in primal form as shown in (1). The nomenclature, adopted in the equations hereafter, is summarized in Table I.

$$\min_{\mathbf{w},\xi} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{i=1}^{N}\xi_i \quad (1)$$

$$\text{Subject to} \quad y_i(\mathbf{w}^{\mathrm{T}}\varphi(\boldsymbol{x}_i) + w_0) \geq 1 - \xi_{\mathrm{i}}$$

$$\xi_i \geq 0, \forall i$$

$\varphi(x_l)$ is such that $K(x_1, x_2) = \varphi(x_1).\varphi(x_2)$.

The solution should satisfy the Karush-Kuhn-Tucker (KKT) conditions [20-21], stated below; boldface letters represent vectors.

1. $\boldsymbol{w} = \Sigma_{\mathrm{i}=1}^{N}\lambda_i y_i \boldsymbol{x}_i$
2. $\Sigma_{i=1}^{N}\lambda_i y_i = 0$
3. $C - \mu_i - \lambda_i = 0 \; i = 1, 2, \ldots, N$
4. $\lambda_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 + \xi_i] = 0 \; i = 1, 2, \ldots, N$
5. $\mu_i\xi_i = 0 \; i = 1, 2, \ldots, N$
6. $\mu_i, \xi_i > 0 \; i = 1, 2, \ldots, N$

Since the dual formulation of this problem, shown in (2), is more efficient to solve, it is adopted in most implementations.

$$\min_{\boldsymbol{\lambda}} \frac{1}{2}\boldsymbol{\lambda}^T K\,\boldsymbol{\lambda} - \mathbf{1}^T\boldsymbol{\lambda} \quad (2)$$

$$\text{s.t } 0 \leq \boldsymbol{\lambda} \leq C$$

$$\boldsymbol{y}^T\boldsymbol{\lambda} = 0$$

For linearly non-separable classification problems, kernels are used to project the input data into a higher dimension space where the database would be at least pseudo linearly separable. The widely used Gaussian Radial Basis Function (RBF) [22-23] uses (3) to project the input vector to the kernel space, and results in a positive semi-definite kernel matrix. The sigmoid kernel, another kernel that does not necessarily produce a positive semi-definite kernel matrix, is computed using (4) and useful for many problems. These kernel values belong to the range [-1, 1]. Since the kernel values are used as distance measures between two data points, negative values are not suitable. Therefore, the absolute value of the kernel matrix is taken to produce positive distances. Kernel parameters such as σ^2, β and γ are obtained by performing a grid search and choosing the values that produce the best classification accuracy.

$$K(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{\sigma^2}\right) \quad (3)$$

$$K(x_i, x_j) = \tanh(\beta x_i^T x_j + \gamma) \quad (4)$$

Solving the optimization task becomes more computationally expensive as the size of the data set increases, since the KKT conditions need to be checked for all points in the training set. The number of SV used to describe the separating hyper plane can also become large. Therefore, we propose KCSVM and KCOOSVM to reduce the number of SV by reducing the database size used to train SVM without significant sacrifices to the classification accuracy. Since the boundary points, the vectors closest to the separating plane, are the ones that will mostly affect the hyper plane parameters, all non-boundary data points could be eliminated without significant change in the final decision rule, as shown in Fig. 1. This fact is evident in the formulation of the optimization problem in (1). The optimization problem finds the hyper plane which is farthest from either class. Therefore, if the points on the periphery of one class that are closest to the opposite class were shifted, the hyper plane would shift as well. On the other hand, if non-boundary points were shifted, the hyper plane would not be greatly affected. Boundary points are determined by measuring their distance to data points of opposite class. If the points are far enough from the points of the other class, they are removed from the training set. After evaluating the distances of the data points to all points in opposing classes, the reduced set is formed of those points that are close enough to the boundary between the two classes. A pruning procedure, discussed next, was adopted to reduce the size of the data set.

TABLE I NOMENCLATURE

N: data set cardinality
N_1: class 1 set cardinality
N_2: class 2 set cardinality
$K(x_i, x_j)$: kernel value for instances x_i and x_j
w, w_0: hyper plane parameter vector
x_i: data point belonging to $R^f, i = 1, \ldots, N$
y_i: label for data point x_i
ξ_i: slack variable associated with each data point

C: box constraint
σ^2: variance
β and γ: kernel parameter
φ: basis function that maps input to kernel space
λ and μ: Lagrange multiplier vector
$Cutoff$: point on the cumulative curve below which points are retained, $Cutoff \in \{1, \dots, N_1 N_2\}$
$pullback$: distance from saturation point to cutoff point, $pullback \in (0,1)$
SP: saturation point
$SPThreshold$: saturation point threshold, $SPThreshold \in (0,1)$
$StepSize$: subsampling rate, $StepSize \in \{1, \dots, N_1 N_2\}$
D_{red}, C_{red}: reduced data matrix and class vector

TABLE II KCSVM WORKFLOW

1. Given an input database defined by the data matrix $D_{N \times f}$ and the class vector $C_{N \times 1}$
2. Sort the database based on the class vector, using counting sort
3. Compute the partial kernel matrix K, i.e. inter-class kernel entries, using (5)
4. Reshape K from $N_1 \times N_2$ matrix to K' a $N_1 N_2 \times 1$ vector
5. Sort K' in descending order to obtain the vector in (6)
6. Compute the cumulative distance values, using (7)
7. Find the cutoff point based on the slope, using (8) and (9)
8. Select the boundary points, using (10) and (11)
9. Return the reduced database D_{red}, C_{red}

B. KCSVM for RBF Kernel

The size of the input database is reduced based on the workflow in Table II. First, the database is sorted based on the label of each instance to easily construct the kernel matrix. The inter-class kernel matrix values are only computed to identify the boundary points; the intra-class values are discarded. The kernel values are sorted in descending order since closer points have larger kernel values based on (3). Then, the sorted values are accumulated, using (7), to produce the plot in Fig. 2. At the saturation point of the curve, the points become too far from each other and are not considered boundary points. Therefore, retaining all the points corresponding to the kernel values below the saturation point is a reasonable approach to identify the boundary points. The saturation point is detected based on the slope of the curve at that point. When the slope at a point on the curve drops below a predefined percentage, called saturation point threshold, of the initial slope, that point is considered the saturation point, as computed in (8). To control the reduction in training set size, the pullback is defined as the distance from the saturation point where the curve will be cut off, based on (7). As this value decreases, the cutoff point moves away from the saturation point and the achievable reduction increases, as shown in Fig. 2. Therefore, all instances, corresponding to the kernel values below the cutoff point, form the reduced training set.

Although this method was only investigated for a Gaussian kernel, it could be extended to other kernels with slight modifications to some of the steps in the algorithm. For example, when using a sigmoid kernel, small values correspond to boundary points. Therefore, the matrix values should be sorted in ascending order. The absolute value of the kernel values is used because negative distances do not have any significance. Its saturation point can be determined based on the slope of the curve and the points below the saturation point are retained. Similar modifications can be made for other kernels depending on the properties of these kernels.

C. KCOOSVM: Ordinal Optimization Inspired KCSVM

Selecting the boundary points only to train the classifier could result in over-fitting since the resulting classifier is more susceptible to outliers. Therefore, we made use of concepts in ordinal optimization to minimize over-fitting in KCSVM by injecting non-boundary points into the reduced training set. First, a brief introduction of the ordinal optimization (OO) theory will be presented.

OO or soft optimization makes hard, large scale problems solvable by reducing the complexity of the required computations. OO is dependent on two basic concepts [24]. The first declares that finding the "order" of the solution is much easier, faster and more robust to noise than finding the "value" of the solution. The second states that finding "good enough" solutions is less expensive than finding the "best" solution available. Based on OO's second concept, a modification to the KCSVM algorithm is performed.

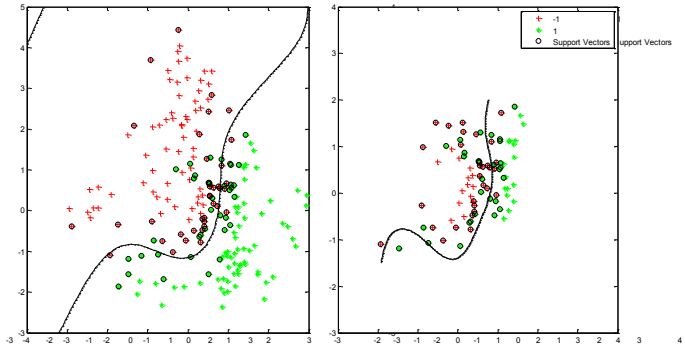

Fig. 1. (Left) Hyper plane after training SVM on original data (Right) Hyper plane after training data on reduced set produced by KCSVM using RBF kernel, after 55% reduction in training set size

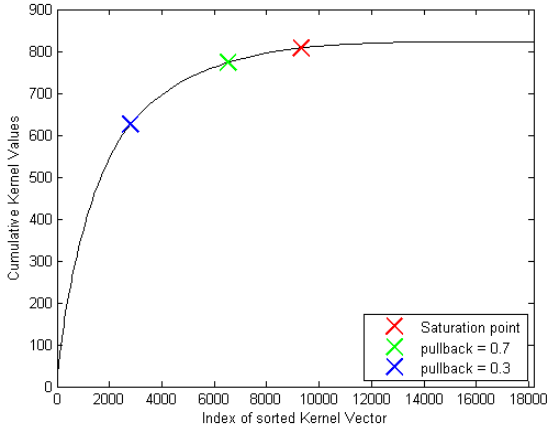

Fig. 2. Illustrative cumulative distances plot for an RBF kernel

$$K(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{\sigma^2}\right) \forall i, j \in \{1, \dots, N\} \; such \; that \; y_i \neq y_j \quad (5)$$

$$K' = \{(K'_1, \dots, K'_j, \dots, K'_{N_1N_2}) | K'_i \geq K'_j \; \forall i < j\} \quad (6)$$

$$CumDist(k) = \sum_{k=1}^{N_1N_2} \sum_{l=1}^{k} K'(l) \quad (7)$$

$$Cutoff = pullback \times SP \quad (8)$$

$$SP = \left\{ j \left| \begin{array}{c} \text{atan}(K(j), K(j+1)) \leq SPThreshold \times \text{atan}(K(1), K(2)) \\ 1 \leq j \leq N_1N_2 \\ j \leq i \; \forall \; i \in \{1, \dots, N_1N_2\} \; such \; that \; \text{atan}(K(i), K(i+1)) \leq SPThreshold \times \text{atan}(K(1), K(2)) \end{array} \right. \right\} \quad (9)$$

$$D_{red} = \{x_i | \exists j \neq i \in \{1, \dots, N\} \; such \; that \; K(x_i, x_j) \in K'(1{:}\, Cutoff) \; \forall i \in \{1, \dots, N\}\} \quad (10)$$

$$C_{red} = \{y_i | \exists j \neq i \in \{1, \dots, N\} \; such \; that \; K(x_i, x_j) \in K'(1{:}\, Cutoff) \; \forall i \in \{1, \dots, N\}\} \quad (11)$$

$$D_{red} = \left\{ x_i \left| \exists i \in \{1, \dots, N\} such \; that \; K(x_i, x_j) \in K'(l \times stepSize), l \in \left\{1, \dots, \frac{N_1N_2}{stepSize}\right\} \right. \right\} \quad (12)$$

$$C_{red} = \left\{ y_i \left| \exists i \in \{1, \dots, N\} such \; that \; K(x_i, x_j) \in K'(l \times stepSize), l \in \left\{1, \dots, \frac{N_1N_2}{stepSize}\right\} \right. \right\} \quad (13)$$

Since the distance values between the data points for KCSVM are sorted, choosing the point at index x or $x \pm \epsilon$ should not make much of a difference; it should keep us within the set of "good enough" solutions. Therefore, instead of selecting all the points on the boundary, the algorithm can select some points that are on the boundary in addition to some points farther away from the boundary, preventing the model from over fitting. One method to achieve this is uniformly sampling the curve in Fig. 2. Consequently, a smaller problem is obtained and a "good enough" solution would be achieved with less computational effort than the "best" solution. The resulting hyper plane, shown in Fig. 3, is a closer approximation of the hyper plane generated when training on the whole data set as shown in Fig. 1(Left).

As mentioned earlier, selecting boundary points only might result in over-fitting. However, for sparsely distributed points and/or datasets that show severe imbalance close to the hyper plane location, uniformly subsampling might result in worse performance because the separating hyper plane's original shape might be lost due to the point selection strategy. Therefore, instead of simply adopting one algorithm regardless of the characteristics of the database in the hyper plane's vicinity, making use of these characteristics in deciding which algorithm to adopt would be advantageous. One method to gain some insight into the density of the points' distribution around the boundary would be to compute the initial slope of the cumulative distances plot, which contains the closest points between the classes and to check for the imbalance ratio of the database. If the curve has a large initial slope, many points are clustered on the boundary in kernel space and very few points are far from each other. At the other extreme, if the slope is small, the points' distribution is sparse, i.e. points are far from each other. Based on this observation, the workflow, in Table III, is proposed. After computing and sorting the kernel matrix, and computing the cumulative distances, the slope at 10% of the cumulative kernel values, the original imbalance ratio of the complete database and the imbalance ratio at 10% are used to suggest adopting either KCSVM or KCOOSVM. If this slope is less than a predefined threshold t_1 and the imbalance ratio is greater than a predefined threshold t_2, the reduced set is selected based on KCSVM instead of KCOOSVM.

IV. Experimental Results

We used MATLAB 2011a (64-bit) [25] on a PC equipped with an Intel Core 2 Extreme dual processor at 2.67 GHz with 4GB of RAM to assess our proposed methods.

A. Database Description

Six databases, shown in Table IV, were chosen from the UCI Machine Learning Repository [4] to assess the performance of the proposed reduction algorithms and compare them to other published methods.

B. Experimental Setup

A 5-fold cross validation was used to validate the results of the various models. Based on Fig. 4, four sets were used in training and the fifth in testing. The same partitioning was used for all algorithms to make the results comparable.

The SVM's parameter values, the box constraint, C, and the RBF kernel parameter σ, used in our experiments and reported in Table IV, were obtained by performing a grid search on the full database and choosing the values that produced the highest classification accuracy. Algorithm specific parameters were swept to vary reduction rates, as shown in Table V.

TABLE III KCOOSVM WORKFLOW

1. Given an input database defined by the data matrix $D_{N \times f}$ and the class vector $C_{N \times 1}$
2. Sort the database based on the class vector, using counting sort
3. Compute the partial kernel matrix, i.e. inter-class kernel entries, using (5)
4. Reshape K from $N_1 \times N_2$ matrix to K' a $N_1N_2 \times 1$ vector
5. Sort K' in descending order to obtain the vector in (6)
6. Compute the cumulative distance values, using (7)
7. Compute the slope and imbalance ratio at 10% of cumulative distance values
8. If slope < t_1 and imbalance ratio > t_2
   a. Use KCSVM point selection method, using (10) and (11)
9. Else
   a. Use KCOOSVM point selection method, using (12) and (13)
10. Return the reduced database D_{red}, C_{red}

TABLE IV DATABASE DETAILS AND SVM PARAMETERS FOR OUR RESULTS

Database	Number of attributes	Number of points	Points in Class 0 / %	Points in Class 1 / %	Published Classification Accuracy (%)	C	σ
Spambase	57	4601	2788 / 60.6	1813 / 39.4	91-93 [26]	256	0.015625
Musk (Version 2)	166	7074	5850 / 83	1224 /17	97 [27]	32	0.015625
Breast Cancer Wisconsin Diagnostic (BCWD)	30	569	357 / 62.7	212 / 37.3	98.98 [28]	1	0.0039
Connectionist Bench	60	208	111 / 53.4	97 / 46.6	99.12 [29]	40	0.25
Ionosphere	34	351	126 / 35.9	225 / 64.1	94.7 [30]	3	0.25
SPECTF Heart	44	267	212 / 79.4	55 / 20.6	94 [31]	32	0.015625

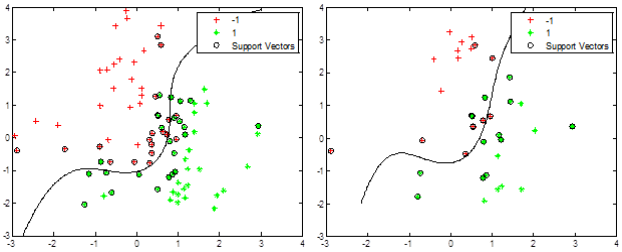

Fig. 3. KCOOSVM (Left) reduction = 56% (Right) reduction = 81%

TABLE V ALGORITHM PARAMETERS

Algorithm	Parameter	Range	Increment
RSVM [15]	Reduction rate	[0.1, 1]	0.1
kNN SVM [3]	k	[3, 11]	1
KMSVM [17]	Reduction rate	[0.1, 1]	0.2
LMSVM [18]	Threshold	{0.2,0.3,0.5,0.7,0.9,1}	
	Cluster count	60	
KCSVM	Pullback	[0.1, 1]	0.1
KCOOSVM	Sampling step size	[0.1, 1]	0.1

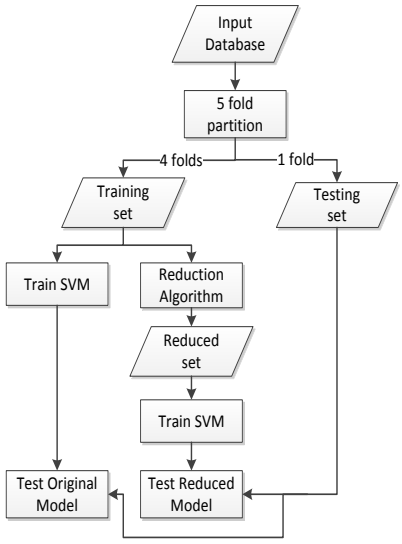

Fig. 4. Cross validation technique

TABLE VI TRAINING RESULTS FOR MUSK (VERSION 2) DATABASE

	Set Reduction (%)	Preprocessing time (seconds)	Training time (seconds)
RSVM	0.00	0.00	3.44
	10.00	0.01	2.80
	30.00	0.01	1.67
	50.01	0.00	0.85
	70.01	0.00	0.32
	90.01	0.00	0.04
kNN SVM	0.00	0.00	3.44
	64.13	4.83	0.48
	72.43	4.79	0.28
	75.86	4.84	0.23
	79.50	4.86	0.17
	83.56	4.83	0.11
	87.72	4.76	0.07
KMSVM	0.00	0.00	3.42
	10.03	393.50	2.76
	30.03	401.68	1.62
	50.01	382.65	0.87
	70.02	290.09	0.31
	90.02	135.24	0.04
LMSVM	0.00	0.00	3.88
	15.05	4.77	2.90
	46.04	4.93	1.23
	65.38	5.01	0.51
	74.70	4.97	0.28
	88.30	4.88	0.07
KCSVM	0.00	0.00	7.00
	0.40	1.91	5.00
	0.94	2.14	5.47
	1.71	1.66	4.47
	4.06	1.95	4.98
	7.40	2.42	6.36
	10.80	2.29	5.72
KCOOSVM	0.00	0.00	5.29
	3.15	1.44	4.40
	7.00	1.62	4.63
	11.60	1.61	4.21
	16.12	1.13	2.88
	20.39	1.57	3.45
	24.20	1.58	3.10
	27.87	1.58	2.78
	31.32	1.19	2.83

C. *Preprocessing and Training Time Analysis*

Table VI presents the preprocessing and training time required on the Musk (Version 2) at different set reduction values. SVM was trained on the reduced sets using libSVM with the pre-computed kernel option. KCSVM and KCOOSVM's preprocessing time increased as the database size increased; it is independent of the required size reduction but the training time decreased as the training set size decreased. Similar methods published in literature, specifically RSVM, kNN SVM, KMSVM and LMSVM, were implemented in Matlab to compare their results to those of the proposed algorithms. KMSVM was implemented using Matlab's k-means algorithm implementation.

Compared to these algorithms, KCSVM and KCOOSVM needed less preprocessing time than all other methods with the exception of RSVM.

D. *Prediction Accuracy vs. Support Vectors Analysis*

Although the reduction in training set size decreased the SV set cardinality, improving online prediction computation and memory requirements, KCSVM experienced a reduction in

prediction accuracy. Fig. 5 plots the prediction accuracy as a function of the percentage of data points which were SV for multiple algorithms and databases.

On the Breast Cancer database, prediction accuracy dropped from 91% to 63% for a reduction of 96% in training set size whereas the SV set decreased from 70 to 60% of the training data, as shown in Fig. 5. SPECTF Heart database did not experience a reduction in accuracy when the training set was reduced by 43%. Furthermore, a 1% increase in prediction accuracy was witnessed at 24% training set size reduction which can be attributed to eliminating noisy data points that might have affected the shape of the separating hyper plane. The number of SV went from 40% to approximately 27% of the data points. The reduction in SV was not significant, as the reduction in training points increased; as more non boundary points were discarded, the remaining points were mostly SV. This also explains the fact that the reduction in accuracy was negligible on this database, as discussed in Section 4.3. Therefore, KCSVM selected quality points that influenced the separating hyper plane and eliminated points that were irrelevant, resulting in minimal prediction accuracy loss for an increase in computational and memory savings.

KCSVM achieved comparable results to other methods in literature for the Ionosphere, Musk (Version 2) and Spambase databases. It outperformed the other methods on SPECTF Heart but fell short on Breast Cancer Wisconsin Diagnostic and Connectionist Bench databases. In general, KCSVM was over-fitting the training data since almost all the retained data points were boundary points and some information about the general distribution of the data points was lost.

KCOOSVM did not over fit the training data since it selected points that were on the boundary in addition to points that were not. In general, a reduction in training set size resulted in a decrease in SV set cardinality, accompanied by a slight dip in prediction accuracy. For example, a 77% set size reduction led to a 5% decrease in SV accompanied by a 3% loss in prediction accuracy for the Ionosphere database. However, KCOOSVM did not fare well on the SPECTF Heart database, as previously mentioned. As shown in Fig. 5, KCOOSVM achieved comparable reduction in SV and corresponding prediction accuracy to other methods in the literature but resulted in better prediction accuracy for the Breast Cancer Wisconsin Diagnostic database.

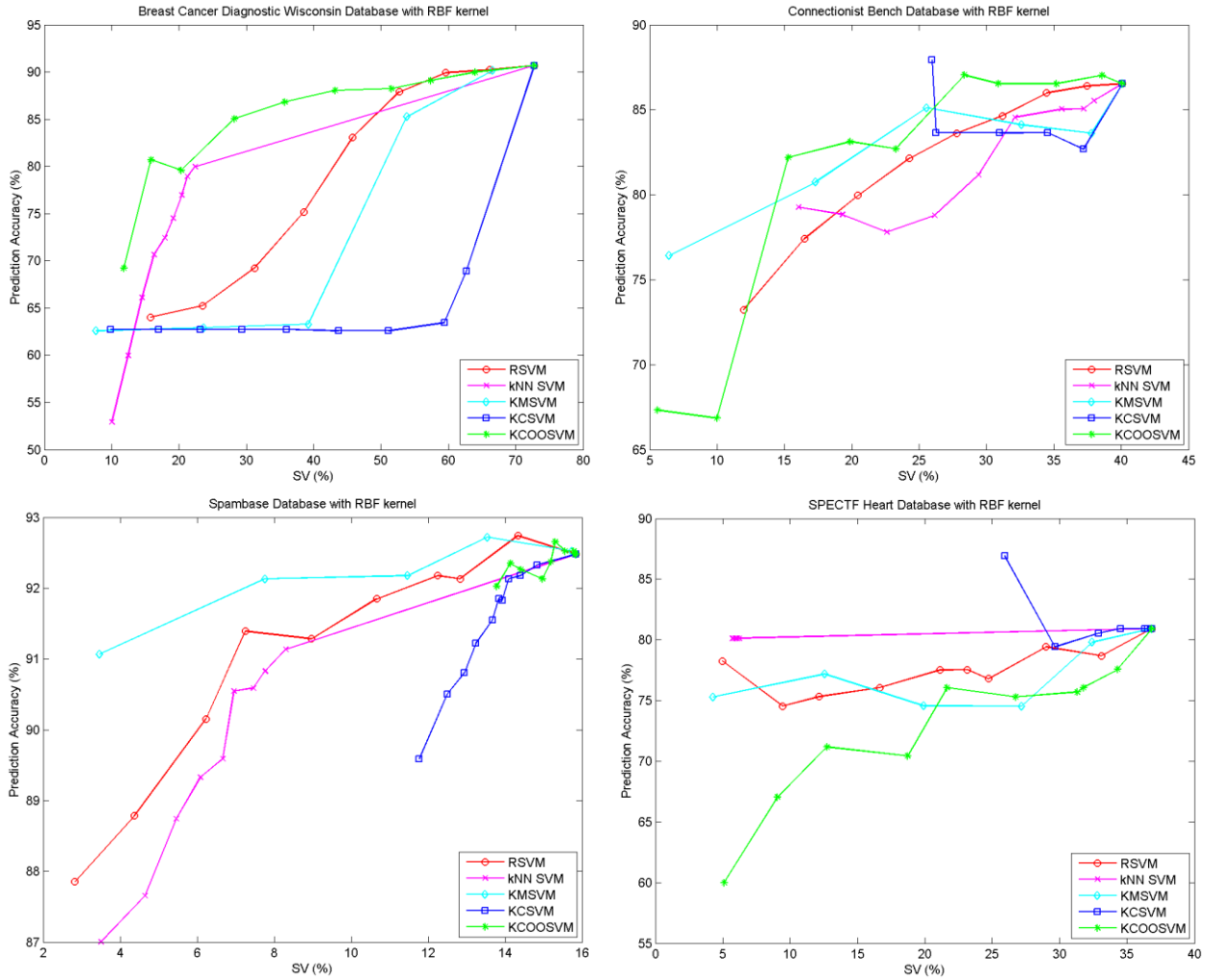

Fig. 5. Prediction accuracy reduction as a function of SV set size when using an RBF kernel for each database

E. KCSVM vs. KCOOSVM

Fig. 6 shows the cumulative distances curves of each of the databases. The initial slopes, the original imbalance ratio of the complete databases and the imbalance ratios at several cutoff points are summarized in Tables VII and VIII, respectively. As the cumulative distances curve cutoff point increases, a wider buffer zone around the boundary, containing more points, is considered. The slope, measured for various buffer zone widths, decreases as more points were included until it reached the saturation point. Databases that had a sharp initial slope, such as the Breast Cancer Wisconsin Diagnostic and Ionosphere databases, exhibited better performance when KCOOSVM was used.

Similarly, the imbalance ratio was measured at several cutoff points; as the cutoff point moved closer to the saturation point, points farther way from the separating hyper plane were included. The class distribution did not vary greatly at different cutoff points.

Combining the class distribution with the initial slope, databases that did better when KCSVM was used had a small slope and an unbalanced distribution as suggested by our proposed workflow. To get a better idea of the behavior of the database at the boundary, a narrower buffer zone, at 10%, should be considered.

To numerically distinguish between slow vs. sharp rising slopes and balanced vs. unbalanced datasets, empirical values for the thresholds defined in Section 2.3 were set to: $t_1 \approx 2$ and $t_2 \approx 60$ with the initial slope and imbalance ratio both computed at 10%.

F. Repeatability Analysis

At first glance, RSVM is an attractive method to select a reduced set because it takes a few tens of microseconds of preprocessing when other methods, including our KCOOSVM, take in the order of a few hundreds of milliseconds, 1000 times slower than RSVM. However, the model generated by RSVM should be averaged over multiple runs to obtain a representative subset of the database and hence good online performance. As the training set size reduction increases, RSVM's performance worsens. On the other hand, our method is more systematic since it does not include any randomness and needs to be run only once to get a good approximation of the original separating hyper plane.

Fig. 7 displays the testing accuracy of the individual runs of RSVM, the average accuracy of RSVM after 100 runs, the average accuracy of RSVM up to the given run and the accuracy of KCOOSVM for a training set size reduction of 80% on the Connectionist Bench database. Clearly, RSVM is very jumpy, with a maximum prediction accuracy differential of 17%. Although RSVM had higher accuracies than KCOOSVM in some runs, the average accuracy after 100 runs is less than KCOOSVM's accuracy. Computing the average of RSVM up to a specific run shows that RSVM is also affected by how many times training is performed. KCOOSVM results in better accuracy than RSVM for slightly more preprocessing cost.

TABLE VII INITIAL SLOPE OF CUMULATIVE DISTANCES CURVE

Database	At 10%	At 15%	At 20%
Spambase	2.25	2.20	2.14
Musk (Version 2)	11.00	9.95	8.90
BCWD	689.00	689.00	689.00
Connectionist Bench	1.80	1.71	1.65
Ionosphere	14.00	12.91	11.82
SPECTF Heart	1.70	1.66	1.62

TABLE VIII IMBALANCE RATIO OF EACH DATABASE

Database	Original (%)	At 10% (%)	At 15% (%)	At 20% (%)
Spambase	60.6	62.69	62.23	60.94
Musk (Version 2)	83.0	82.27	82.27	81.59
BCWD	62.7	66.67	60.00	55.56
Connectionist Bench	53.4	52.14	50.74	50.34
Ionosphere	64.1	80.82	79.00	79.67
SPECTF Heart	79.4	63.21	64.52	67.88

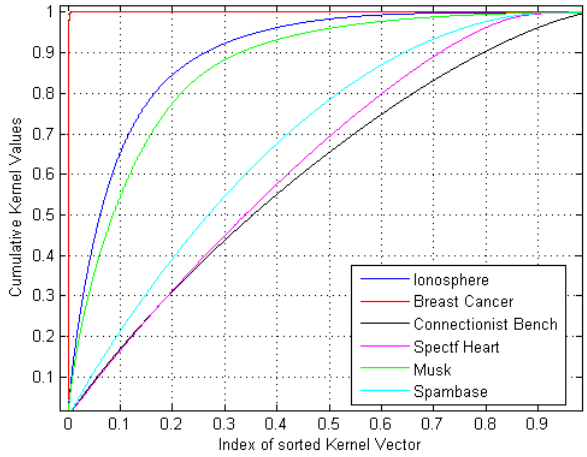

Fig. 6. Cumulative distance plot for each database

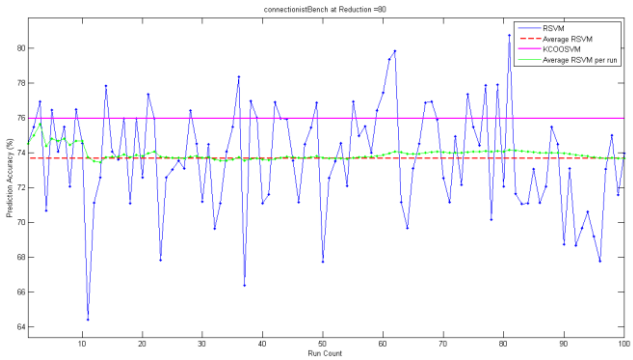

Fig. 7. Repeatability analysis of RSVM for training set size reduction of 80% on Connectionist Bench

V. CONCLUSION

In this paper, two algorithms, KCSVM and KCOOSVM applicable to RBF kernel, were presented that attempt to reduce online computation and memory requirements by reducing the training set size, which results in less SV. KCSVM extracts boundary points by viewing kernel values between data points of unequal classes as distance measures between these points. Points from different classes with small distance values are more likely on the boundary and will most likely be SV. Since selecting the boundary points only resulted in over-fitting, KCOOSVM was introduced, inspired

by concepts in ordinal optimization, which uniformly subsampled the cumulative distance curve.

Experimental results on six databases from the UCI repository showed promising results for both methods. Both resulted in SV count reduction with minimal impact on the prediction accuracy based on the associated training set reduction. Compared to several algorithms in the literature, KCSVM and KCOOSVM outperformed some of these algorithms on some databases, but produced comparable results on other databases. Furthermore, although RSVM required the least preprocessing overhead, it has to be repeated multiple times to ensure a proper representation of the database which renders its preprocessing just as costly as KCOSVM, with lower average prediction accuracy.

ACKNOWLEDGMENT

This work was partially funded by the University Research Board at the American University of Beirut and partially by MER, a partnership between Intel Corporation and King Abdul-Aziz City for Science and Technology (KACST) to conduct and promote research in the Middle East.

REFERENCES

[1] M. Awad and Y. Motai. “Dynamic classification for video stream using support vector machine.” *Journal of Applied Soft Computing*, vol. 8 (4), pp. 1314-1325, 2008.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1999.

[3] L. Zhang, N. Ye, W. Zhou and L. Jiao. "Support vectors pre-extracting for support vector machine based on K nearest neighbour method," in *Int. Conf. on Information and Automation,*, 2008, pp. 1353-1358.

[4] K. Bache and M. Lichman (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science [Online]. Available: http://archive.ics.uci.edu/ml

[5] C. C. Chang and C. J. Lin. “LIBSVM: a library for support vector machines.” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27, 2011.

[6] T. Cheng, Y. Wang, and S. H. Bryant. “FSelector: a Ruby gem for feature selection.” *Bioinformatics*, vol. 28(21), pp. 2851-2852, 2012.

[7] D. Wang, T. Li, J. Sun, D. Li, W. Xiong, W. Wang, and S. Tang. “Shape string: a new feature for prediction of DNA-binding residues.” *Biochimie*, 2012.

[8] C. J. Burges. “Simplified support vector decision rules,” in *International Conference on Machine Learning*, 1996, pp. 71-77.

[9] E. Osuna and F. Girosi. “Reducing the run-time complexity of support vector machines,” in *Proc. of the Int. Conf. on Pattern Recognition*, 1998.

[10] T. Joachims and C. N. J. Yu. “Sparse kernel SVMs via cutting-plane training.” *Machine Learning*, vol. 76 (2-3), pp. 179-193, 2009.

[11] T. Downs, K. E. Gates and A. Masters. "Exact simplification of support vector solutions." *The Journal of Machine Learning Research,* vol. 2, pp. 293-297, 2002.

[12] D. D. Nguyen and T. B. Ho. "A bottom-up method for simplifying support vector solutions." *IEEE Transactions on Neural Networks,* vol. 17, pp. 792-796, 2006.

[13] G. Bakır, L. Bottou and J. Weston. “Breaking SVM complexity with cross training.” *Advances in neural information processing systems*, vol. 17, pp. 81-88, 2005.

[14] D. Geebelen, J. A. Suykens, and J. Vandewalle. “Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation.” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23 (4), pp. 682-688, 2012.

[15] Y. J. Lee and O. L. Mangasarian. "RSVM: Reduced support vector machines," in *Proc. of the First SIAM Int. Conf. on Data Mining,* 2001, pp. 5-7.

[16] B. Kong and H. Wang. "Reduced support vector machine based on margin vectors," in *2010 Int. Conf. on Computational Intelligence and Software Engineering (CiSE),* 2010, pp. 1-4.

[17] J. Wang, X. Wu and C. Zhang. "Support vector machines based on K-means clustering for real-time business intelligence systems." *Int. Journal of Business Intelligence and Data Mining,* vol. 1, pp. 54-64, 2005.

[18] Y. Rizk, N. Mitri, and M. Awad. “A Local Mixture Based SVM for an Efficient Supervised Binary Classification,” in *Int. Joint Conf. on Neural Networks*, Dallas, TX, 2013.

[19] R. Koggalage and S. Halgamuge. “Reducing the number of training samples for fast support vector machine classification.” *Neural Information Processing-Letters and Reviews*, vol. 2(3), pp. 57-65, 2004.

[20] H. W. Kuhn and A. W. Tucker. “Nonlinear programming,” in *Proc. of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, 1951, pp. 481–492.

[21] H. W. Kuhn. “Nonlinear programming: A historical view,” In *Nonlinear Programming*, R. W. Cottle and C. E. Lemke, Ed. SIAM-AMS Proceedings, 1976, pp. 1–26.

[22] S. Theodoridis and K. Koutroumbas. (1999). Pattern recognition.

[23] M. Fauvel, J. Chanussot, and J. A. Benediktsson. “Evaluation of kernels for multiclass classification of hyperspectral remote sensing data,” in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006, pp. II-II.

[24] Y. Ho, Q. Zhao, and J. Q. Shan. *Ordinal Optimization: Soft Computing for Hard Problems*. Springer, 2007.

[25] Mathworks (2013). *Matlab [Online]*. Available*: www.mathworks.com*

[26] C. Dimitrakakis and S. Bengio. “Online adaptive policies for ensemble classifiers.” *Neurocomputing*, vol. 64, pp. 211-221, 2005.

[27] Z. H. Zhou, "Multi-instance learning: A survey," *AI Lab, Department of Computer Science and Technology, Nanjing University, Tech. Rep,* 2004.

[28] C. A. Pena-Reyes and M. Sipper. “Fuzzy CoCo: A cooperative coevolutionary approach to fuzzy modeling.” *IEEE Transactions on Fuzzy Systems*, vol. 9 (5), pp. 727-73, 2001.

[29] Z. Chelly, A. Smiti, and Z. Elouedi. “COID-FDCM: the fuzzy maintained dendritic cell classification method.” *Artificial Intelligence and Soft Computing*, pp. 233-241, 2012.

[30] G. Fung, M. Dundar, J. Bi, and B. Rao. “A fast iterative algorithm for fisher discriminant using heterogeneous kernels,” in *Proc. of the 21st Int. Conf. on Machine Learning*, 2004, pp. 40.

[31] R. Asadi, N. Mustapha, N. Sulaiman, and N. Shiri. “New supervised multi-layer feed forward neural network model to accelerate classification with high accuracy.” *European Journal of Scientific Research*, vol. 33, pp. 163-178, 2009.