A Local Mixture Based SVM for an Efficient Supervised Binary Classification

Yara Rizk, Nicholas Mitri, Mariette Awad
Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
{yar01, ngm04, mariette.awad}@aub.edu.lb

Abstract—Despite support vector machines’ (SVM) robustness and optimality, SVM do not scale well computationally. Suffering from slow training convergence on large datasets, SVM online testing time can be suboptimal because SVM write the classifier hyper-plane model as a sum of support vectors that could total as much as half the datasets. Motivated to speed up SVM real time testing by reducing the number of SV, we introduce in this paper a novel local mixture based SVM (LMSVM) approach that exploits the increased separability provided by the kernel trick, while introducing a one-time computational expense. LMSVM applies kernel k-means clustering to the data in kernel space before pruning unwanted clusters based on a mixture measure for label heterogeneity. LMSVM's computational complexity and classification accuracy on four databases from UCI show promising results and motivate follow on research.

Keywords—SVM; k-means clustering; real time testing, supervised and binary classification

I. INTRODUCTION

Deeply rooted in the principle of structural risk minimization, Support Vector Machines (SVM) was first proposed by Boser, Guyon and Vapnik [1-2] in their work on statistical learning theory. Known for their robustness, good generalization ability, and unique global optimum solution, SVM have found their way into a myriad of classification and regression tasks in various pattern recognition applications [3]. It is with larger datasets, though, that SVM fail to efficiently deliver; especially in the nonlinear classification case. Large datasets impose great computational time and storage requirements [4], rendering them, in some cases, slower than neural network, already known for their slower convergence [5].

A survey related to SVM and its variants reveals a dichotomy of speedup strategies. The first class of techniques applies to the training phase of the SVM algorithm which incurs the heftier computational expense in its search for the optimal separator. The intent of these algorithms is to reduce the cardinality of the data set. The second class of techniques optimizes the testing cycle.

The goal of this paper is to present a method to reduce the prediction time and decrease the number of support vectors (SV) without a significant reduction in accuracy or a significant training overhead cost. SV count is data dependent and could reach up to half the data set size. To that end, we propose a local mixture based SVM (LMSVM), which applies a pre-training heuristic clustering followed by a pruning technique to render the solution space sparser. The pruning technique is based on a local mixture metric. LMSVM’s performance was evaluated by comparing its accuracy and computation time to the original SVM algorithm. Time reduction of up to 45% in testing was achieved with an accuracy drop of 1.6% for the Spambase data set.

With the proliferation of power conscious mobile devices and ubiquitous computing being more pushed from the cloud to these terminals, LMSVM can be used in many applications where computational resources are limited and real time prediction is necessary. For example, online prediction on mobile devices would greatly benefit from the reduced computations required to perform a prediction.

In the remainder of the paper, we present in Section II a literature review of several SVM speed up strategies and clustering algorithms. The related work is organized as subset representation modifications, role approximation modifications, and finally applications of R SVM. Section III details the proposed method while Section IV presents the experimental results. Finally, Section V concludes the paper with follow on research directions.

II. RELATED WORK

Competing with “divide and conquer” methods such as Platt’s SMO [6], reduced SVM (RSVM) as proposed by Lee and Mangasarian [4] reduce the number of SV which are the essential subset of data samples to the minimization task, and thus cut down the computational time and storage requirements with no significant compromise in performance. Since the classification time of a new pattern is proportional to the number of SV and the SVM minimization task is strongly coupled with the dataset cardinality, RSVM [7] lends itself to applications that involve large datasets and demand fast training and reliable online performance; possibly real-time [8-11].

The original formulation of RSVM [4] has given rise to several modifications. While the former proposed a random selection of the dataset extracted to form the reduced set, Songfeng et al. [12] focused on the representational quality of their reduced set opting for an unsupervised clustering approach to obtain a reduced set made up of cluster centers for the positive and negative datasets which are used as SV
to train RSVM. Their algorithm performed better than RSVM. Using libSVM [13], Wang et al. [14] attempted to reduce the number of SV by clustering the data set, for real time business intelligent applications. The data set is clustered using k-means algorithm and SVM is trained on the cluster centers. In the same spirit, Kong et al. [7] proposed a margin based set reduction with the ability to deal with data imbalance and outliers. Using measures of self and mutual distance coupled with percentiles, the method in [7] was shown to outperform traditional RSVM and CSVM in terms of accuracies on three data sets from the UCI repository [15] but with higher computational time than that of RSVM [7]. On the other hand, [16] proposed using k-nearest neighbors (kNN) to identify boundary points that will subsequently form the reduced set used in training SVM. Boundary points are those points whose k nearest neighbors include points of opposing class. For linearly non-separable data sets, this method is applied in the kernel space. Results showed a decrease in SV by 34%, a decrease in training time by 65% and a decrease in testing accuracy by 0.82% on a large scale non-separable problem using RBF kernel.

While previous methods followed a strategy to inject a representative quality into the elements of the reduced set, Downs et al. [17] proposed a reduction scheme that cancels out SV based on their linear dependency on others. Testing on four data sets from [15] using different kernels showed a reduction range of 0% to 92%, which means that the strict compliance with exact performance incurred inconsistent reductions and a non-unique solution. Expanding on Downs’ work, Li et al. [18] suggested an iterative merging technique of close SV pairs to approximate the performance of both SVM and RSVM. Their method yielded, for every pair of proximal SV, a new feature vector approximately representative of the pair’s role in the solution. The proposed method was implemented on seven UCI datasets, and when compared with SVM and Downs’ method showed a significant reduction in SV and an accuracy drop up to 2.5%. A similar approach was developed by Nguyen et al. [19] based on the weighted average of proximal SV. In [20], Jiang et al. used SV coalitions instead of pairs and implemented a similar iterative procedure to obtain the weight of the new representative SV. Experiments were conducted using the Power Quality Disturbance Recognition Database and results outperformed [17] in average computational time (3.5sec vs. 12.3sec) while consistently achieving higher reduction rates and almost identical accuracies.

Several well-known clustering algorithms have been published to cluster data sets in input and kernel space. K-means algorithm was first introduced in 1955; it is one of the most popular clustering algorithms due to its simplicity, efficiency and empirical success [21]. This algorithm partitions the data set into clusters based on minimized the square distance between data points and the centroids of the clusters, the empirical mean of the cluster. The original formulation of the k-means clustering algorithm performed the clustering in the input space. It was later extended to the kernel space [22]. Some variants include k-medoids [23] and k-medians [24] that define medoids and medians as cluster heads, respectively. Fuzzy c-means, a soft clustering method, allows data points to belong to several clusters instead of one [25] and later reformulated to allow clustering in kernel space [26]. Self-organizing map (SOM) partitions data sets into clusters based on unsupervised neural networks [27] which was extended to the kernel space in [28]. Although SOM exhibits low error and fast convergence, it suffers in returning an adequate solution when the data set distribution is not Gaussian or spherical in shape [28]. Furthermore, several clustering algorithms for online applications have been developed. One such algorithm is the self-adaptive kernel machine (SAKM) algorithm, a computationally efficient method that groups non-stationary data into changing clusters in the kernel space but implementing initialization, adaptation, fusion, and elimination stages [29].

III. LMSVM

Given our aim at speeding up the classifier’s prediction phase with minimal impact on classification accuracy, we present in what follows a preprocessing strategy that effectively reduces the size of the data set. The rationale behind our approach is rooted in the observation that SVM is a sparse technique and only SV contribute to the classifier’s model parameter computation. Consequently, any data point lying outside of the “SV pool” can be considered redundant and is discarded. This is done by clustering the data set and applying a merit measure that decides whether a cluster should be preserved or not, as opposed to representing each cluster by its cluster head [12, 14]. We also propose a cluster bias measure to quantify the heterogeneity of a cluster with respect to the two classes. With the understanding that SV lie around the hyper-plane that separates classes, it becomes clear that heterogeneous clusters should be preserved for the best performance approximation. Heterogeneous clusters are usually found around the boundary of both classes. Hence, boundary points can be identified by the heterogeneous cluster measure instead of kNN [16] or self and mutual distance measures [7]. Therefore, LMSVM looks to preserve clusters with heterogeneity scores greater than a threshold. As this threshold is increased, greater reduction is achieved.

Fig. 1 shows the process workflow involved where the kernel matrix is computed from the input data set before the data points are clustered and pruned based on a calculated cluster bias measure. Finally, the reduced database is plugged into the SVM solver. A grid search was performed to find suitable parameters for both k-means clustering and SVM as detailed in Section IV.

A. Kernel Space Clustering

LMSVM clusters the data in the kernel space because the likelihood of achieving a pseudo-optimal separation of classes is increased due to the kernel mapping into a higher dimensional space. Clustering in the kernel space will make it easier to extract the portion of the data set closest to the optimal separating hyper-plane where SV candidates are most likely to exist.

Since the data used in LMSVM does not change with time and the clustering is performed once, SAKM would not be a suitable choice. K-means was chosen due to its popularity and simplicity. In what follows, we propose the
kernel space equivalent of the traditional k-means clustering algorithm that was outlined for the input space in [22]. The following nomenclature is adopted:

- \( \varphi \): basis function that maps input to kernel space \( K(\cdot, \cdot) \)
- \( K \): kernel matrix
- \( \alpha \): hyper-plane parameter vector
- \( \omega \): hyper-plane offset
- \( x_i \): data point belonging to \( \mathbb{R}^m, l = 1, \ldots, N \)
- \( X = \{ x_1, x_2, x_3, \ldots, x_N \} \): the data set with data points \( x_i \in \mathbb{R}^m \)
- \( y_i \): label for \( x_i \) belonging to \{-1,1\}
- \( \xi_i \): slack variable associated with each data point
- \( N_i \): support vector set cardinality
- \( s_p \): support vector, \( p = 1, \ldots, N_i \)
- \( \lambda_i \): Lagrange multiplier for each \( s_p \)
- \( C \): regularization parameter
- \( r \): predefined radius
- \( c_i \): cluster \( i \)
- \( O_j \): centroid of \( c_i \)
- \( N_i \): cardinality of \( c_i \)

**Cluster_Num**: number of clusters

**b**: cluster measure

**T**: threshold on cluster bias

The clustering algorithm executes the following steps:

1. \( c_1 = \{ \varphi(x_1) \}, O_1 = \varphi(x_1), \) Cluster_Num = 1, \( Z = \varphi(X) - \varphi(x_1) \)
2. If \( Z = \varphi \), then STOP
3. For a sample \( x_i \) \( X \), choose the cluster center \( O_j \) closest to \( \varphi(x_i) \) from the existing cluster centers
4. If the Euclidian distance \( d(\varphi(x_i), O_j) < r \), add \( \varphi(x_i) \) to \( c_j \) and update the cluster center \( O_j = \frac{\sum \varphi(x_k)}{N_j} \) \( \) (1)
5. If \( d(\varphi(x_i), O_j) > r \), increment Cluster_Num and create a new cluster with center \( \varphi(x_i) \)
6. \( Z = Z - \{ \varphi(x_i) \} \), go to step 2

**B. Bias Measure and Reduction**

With reduction as our main goal, the next step is the use of heuristics to prune clusters that are unlikely to contribute to the hyper-plane parameter calculation i.e. those that do not contain any SV candidates. With the proper kernel choice, the mapping of the data into the kernel space should allow us to construct a linear separator with SV located in its near vicinity. Since SVM is a sparse technique and SV retain sole control over the hyper parameters, we are motivated to discard those clusters that are homogeneous i.e. whose elements belong to the same class. Theoretically, those clusters are considered redundant and dispensable and are unlikely to contain SV. Practically, that is not always the case. In the case of overlapping classes, such clusters may still contain SV. Nevertheless, since our main goal is computational savings, a compromise must be made between exact performance and fast performance. Therefore, we will emphasize speed and not only discard homogeneous clusters but unbalanced ones as well. For that purpose, we introduce the bias measure shown in (2) to model the classes as opposing or balancing forces within the cluster. It is easy to see that for balanced clusters, \( b_k = 0 \), for homogeneous clusters \( b_k = 1 \), and for heterogeneous clusters \( 0 \leq b_k < 1 \).

\[
b_k = \frac{\sum_{i \in C_k} \varphi(y_i)}{N_k}
\] (2)

To reduce the dataset, we will resort to pruning all clusters whose bias measures exceed \( T \). The higher the threshold, the less severe the reduction and the closer the performance will be to the original classifier, both in accuracy and time.

**C. Kernel-Based SVM**

The traditional SVM algorithm performs the optimization task in (3).

\[
\min_{\omega, \xi} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{N} \xi_i
\]
\[\text{s.t.} \quad y_i (\omega^T \varphi(x_i) + \alpha) \geq 1 - \xi_i
\]
\[\xi_i \geq 0, \forall i \]

\( \varphi(\cdot) \) is dictated by the kernel function \( K(x, x') = \varphi(x) \cdot \varphi(x') \). Solving the SVM algorithm using a variety of optimization methods yields the SV set \( \mathcal{SV} = \{ s_1, s_2, \ldots, s_{N_S} \} \) and their corresponding Lagrange multipliers.

The decision rule for the standard SVM is

\[
f(x) = \text{sign}(\sum_{i} \lambda_i y_i K(x, s_i) + a)
\]

and

\[
\omega = \sum_{\mathcal{SV}} \lambda_i y_i \varphi(s_i)
\] (5)

The Wolfe dual of (3) is computationally more convenient and is solved for the Lagrange multipliers \( \lambda_i \) using the kernel matrix \( K \):

\[
\min_{\lambda} \frac{1}{2} \lambda^T K \lambda - 1^T \lambda
\]
\[\text{s.t.} \quad 0 \leq \lambda \leq C
\]
\[y^T \lambda = 0
\] (6)

Fig. 2 shows the result of applying the proposed method to a 2D toy example with data generated by a bivariate Gaussian distribution. The plot on the left shows the separating hyper-plane of the classifier, trained on the original data set. To the right of Fig. 2, the reduced data set and the corresponding optimal separator as computed using LMSVM on the original set with a threshold \( T=1 \) are displayed. It is easy to see that the proposed method preserves as many SV as possible and thus converges to a separator that approximates that of the original set fairly well.

**IV. EXPERIMENTAL RESULTS**

We used MATLAB 2011a (64-bit) and LIBSVM [13, 30-32] for their fast convergence and variety of integrated tools on a PC equipped with an Intel Core 2 Extreme dual processor at 2.67 GHz with 4GB of RAM to assess LMSVM.

![LMSVM Workflow](image)

**Fig. 1. LMSVM Workflow**
A. Experimental Setup

Four databases were chosen from the UCI Machine Learning Repository [15] to validate LMSVM. Some statistics on the Spambase [33], Musk (version 2) [34], Statlog (Shuttle) [35] and SPECTF Heart [36] data sets are included in Table I. The Statlog Shuttle data set [35] contains 58,000 instances with a class distribution of 80 vs. 20%. However, in the results reported below, only 10,000 instances were used, due to some memory limitations of the machine being used for validation. Fig. 3 displays the overlap of some features between the classes of these databases.

Since the choice of the kernel and number of clusters are problem specific, we chose the radial basis function (RBF) due to its popularity and its accordance with the data sets chosen. The sigma parameter of the RBF (SIG), the cluster number (k), and the regularization term C were obtained using a traditional grid search. The percentage reduction in data set size was used as a merit function to calculate the best (SIG,k) configuration for the clustering module.

Cross fold validation was used to compare the performance of SVM on the input and reduced data set. The data set was divided into 5 folds. SVM was trained on a set formed of 4 folds of the input data set and tested on the remaining fold. This training set was given to LMSVM, reduced to form the reduced set and used to train SVM. The resulting reduced model was tested on the remaining fold. Therefore, both models were tested on the same data and can be compared. The workflow is illustrated in Fig. 4.

B. LMSVM Accuracy and Computational Analysis

The computational times associated with LMSVM as well as relevant information about SV counts and prediction accuracy is shown in Tables II and III. The value of the threshold variable used to filter the data set was varied to obtain several set reduction percentages. The lower the threshold, the more SV are discarded by removing proportionally biased clusters. Clustering is repeated at each run. As expected, a threshold of 1 discards only homogeneous clusters and performs very similarly to the classifier trained using the full data set. This is seen most clearly in the algorithm’s preservation of most of the SV. As the threshold is decreased, we lose that approximation as more and more SV are pruned.

Tables II and III show that as the reduction percentage increases, the training time decreases. However, to achieve this, overhead preprocessing should be considered part of the training time. Therefore, the current LMSVM implementation does not speed up the overall training procedure since the training time before reduction is less than the training time reported in the cluster + filter time column shown in Table III. While this is problematic at face value, it is a one-time expense incurred offline during the training phase. In the light of the available strategies of parallelizing the clustering process, which were not exploited in this work, this computational expense would be effectively reduced multifold.

Despite the overall longer training time for LMSVM, the significant speedup achieved at the prediction level makes
LMSVM desirable for scenarios where training is done offline and prediction is expected to happen in pseudo-real time manner. For example, prediction time decreased by 45% for 67.94% reduction accompanied by 1.6% drop in accuracy for Spambase; an acceptable compromise in applications where the speed of prediction is very important. Similarly, improvements were obtained for the Musk dataset where an 89% size reduction led to a 77% test time reduction while maintaining testing accuracy above 90%. SPECTF Heart data set experienced a reduction of 25% in prediction time at the cost of 2.27% test accuracy reduction when the training set size was reduced by 53.27%.

The reduction observed in prediction time is due to the decreased number of SV, which affects directly the number of operations needed for prediction, and hence the power consumption needed to predict the class of a new instance. The reduction in SV count is clearly accompanied by a decrease in accuracy. However, this decrease is acceptable in most cases. For example, approximately 46% reduction in SV count resulted in an accuracy drop of 1.6% for the Spambase data set. For the Musk data set, a 72% reduction in SV count resulted in a drop of 8% in accuracy. Reducing the SV count by 4.79% for the SPECTF Heart database resulted in a 2.72% decrease in test accuracy.

Reducing the data set size also resulted in a decrease in the prediction accuracy of the classifier. For example, eliminating 89.31% of the Musk database instances still resulted in accuracy above 90%. Removing 88% of the Spambase instances resulted in a 6% drop in accuracy. However, some cases exhibited a slight (less than 1%) increase in accuracy for certain threshold values. This could be justified by the fact that pruning SV can result in shrinking the overlap between classes and would translate into a higher accuracy. Furthermore, some noise instances and outliers might have been eliminated, increasing the accuracy of the classifier. The Statlog dataset exhibited substantial degradation in performance for a small decrease in threshold. This is due to the nature of the database which had its instances distributed into a small number of clusters. Hence, a large size reduction percentage was achieved for a small value of the threshold.

C. LMSVM vs. Published Work

As presented in Section II, several publications have tackled the problem of speeding up the prediction phase of SVM classifiers. The algorithms of several of these publications were implemented, tested and their results were compared them to LMSVM. Fig. 5 displays line graphs comparing the results of RSVM [4], KMSVM [14] and kNN SVM [16] to LMSVM on Spambase [33] and SPECTF Heart [36] data sets. All four algorithms used libSVM’s solver.

Comparing the prediction accuracy for a given percentage of SV, LMSVM has a slight advantage over the other methods, for Spambase, when the SV percentage is greater than 12%. However, as the SV decrease, KMSVM gains a slight advantage. For the more difficult SPECTF Heart database, kNN SVM performed best while RSVM exhibited more jumpy behavior and LMSVM’s testing accuracy decreased steadily, except for a slight glitch around a SV percentage of 20%, outperforming KMSVM. Examining the testing accuracy as a function of the training data set reduction, LMSVM exhibits better accuracy for a given reduction value. A slight dip in performance for reduction above 80% on Spambase but LMSVM was consistently better than the other methods on SPECTF Heart. However, LMSVM had generally more SV than the other methods for a given training set reduction value.

RSVM’s degraded performance, which was only evident in SPECTF Heart, can be attributed to the more complex nature of this database. Selecting points randomly did not faithfully represent the original data distribution and resulted in bad generalization; which was not the case for well-behaved databases. KMSVM also experienced difficulty with this database. kNN SVM and LMSVM fared better on this dataset since they extracted the boundary points which usually have the most influence on the separating hyperplane.

Comparing LMSVM accuracy for these databases with published results speaks in favor of LMSVM especially for offline training, limited memory and power consumption scenarios. Ref. [37] reported accuracies between 86.6 and 88.7% on the Spambase data set while [38] achieved better results with approximately 91 to 93% accuracy. For a 33.6% reduction, LMSVM was able to achieve comparable results (93.262%) on Spambase. Published results on the Musk (Version 2) data set indicated accuracies of 86.6% [39], 90.3% [40], 91% [41] and 97% [42]. LMSVM achieved 97.922% accuracy for a 56.088% reduction. The Statlog (shuttle) database had a testing accuracy between 95.17 and 97.922% accuracy for a 56.088% reduction. The Statlog dataset had a testing accuracy between 95.17 and 99.99% [43] whereas LMSVM produced 98.68% accuracy after reducing the number of instances by 37%. Finally, published results on the SPECTF Heart data set reports accuracies of 77% [44-45] and 81% [46]. LMSVM achieved 80% testing accuracy for a reduction of 46%.

![Fig. 4. LMSVM validation technique](image)

```plaintext
Input Database

5 fold partition

Training set

Test Original Model

Test Reduced Model

Reduced set

Trained SVM

LMSVM

Reduced

Test Reduced Model

Trained SVM
```
D. Repeatability Analysis

Since the k-means algorithm uses random seeding and hence, might not be consistent across runs, we tested LMSVM’s repeatability. To test the robustness of this reduction scheme in the face of varying clusters resulting from different seeds, LMSVM was run several times for the given set of SVM parameters, shown in Table I, and partition of the datasets using random seeds, while fixing the set reduction to 40-42%. K-means produced different clusters in each run. Table IV includes a few sample runs and average results of the 20 runs for each of the four databases. As shown in this table, the testing accuracy of LMSVM did not have a large standard deviation. Based on all 20 runs, the largest difference in accuracy between training on the whole dataset and the reduced set was 1.15% for Spambase, 1.24% for Musk, 2.18% for Statlog and 2.64% for SPECTF Heart.

V. CONCLUSION

In this paper, we presented LMSVM, a novel approach to speeding up prediction time for SVM and reducing the number of SV needed in a classification task. Exploiting the structure of the kernel space, the sparsity of SVM and the influence of SV over the optimal separation hyper-plane, we propose removing samples that are unlikely to contribute much to the final hyper-plane directivity. Coupling a bias measure with a threshold, significant time saving can be made. Experimental testing showed that the number of SV was significantly reduced without a detrimental impact in accuracy, resulting in less computational power and memory requirements. Therefore, improvement in performance was achieved in the prediction phase making LMSVM suitable for applications where online prediction and limited computational resources are important. The focus of future work is to reduce the expensive clustering stage in the training phase and to investigate a scheme to update the reduced system with the possibility of modifying the bias measure to incorporate more information such as inter-cluster distances.
### TABLE III. Computational Requirements

<table>
<thead>
<tr>
<th>Database</th>
<th>Threshold</th>
<th>Set Reduction (%)</th>
<th>Cluster + Filter time (seconds)</th>
<th>Training time (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spambase [33]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>520.51</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>14.181</td>
<td>2.504</td>
<td>443.47</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>33.605</td>
<td>2.343</td>
<td>310.34</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>39.120</td>
<td>2.38</td>
<td>262.61</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>67.943</td>
<td>2.872</td>
<td>77.91</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>88.128</td>
<td>2.321</td>
<td>31.93</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>90.220</td>
<td>2.331</td>
<td>12.20</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1365.81</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>11.380</td>
<td>6.157</td>
<td>1097.45</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>56.088</td>
<td>6.161</td>
<td>443.61</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>77.823</td>
<td>6.165</td>
<td>189.23</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>85.262</td>
<td>6.119</td>
<td>49.35</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>89.309</td>
<td>6.168</td>
<td>28.94</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>93.797</td>
<td>6.19</td>
<td>6.26</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>43.98</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>37.088</td>
<td>11.198</td>
<td>30.33</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>82.300</td>
<td>12.052</td>
<td>2.29</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>93.200</td>
<td>12.105</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>99.863</td>
<td>12.003</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>56.67</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>45.794</td>
<td>0.008</td>
<td>32.20</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>45.794</td>
<td>0.009</td>
<td>33.68</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>53.271</td>
<td>0.009</td>
<td>26.71</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>82.710</td>
<td>0.010</td>
<td>10.27</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>86.916</td>
<td>0.010</td>
<td>4.54</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>93.925</td>
<td>0.009</td>
<td>2.78</td>
</tr>
<tr>
<td>Musk (Version 2) [34]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1365.81</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>11.380</td>
<td>6.157</td>
<td>1097.45</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>56.088</td>
<td>6.161</td>
<td>443.61</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>77.823</td>
<td>6.165</td>
<td>189.23</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>85.262</td>
<td>6.119</td>
<td>49.35</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>89.309</td>
<td>6.168</td>
<td>28.94</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>93.797</td>
<td>6.19</td>
<td>6.26</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>43.98</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>37.088</td>
<td>11.198</td>
<td>30.33</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>82.300</td>
<td>12.052</td>
<td>2.29</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>93.200</td>
<td>12.105</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>99.863</td>
<td>12.003</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>56.67</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>45.794</td>
<td>0.008</td>
<td>32.20</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>45.794</td>
<td>0.009</td>
<td>33.68</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>53.271</td>
<td>0.009</td>
<td>26.71</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>82.710</td>
<td>0.010</td>
<td>10.27</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>86.916</td>
<td>0.010</td>
<td>4.54</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>93.925</td>
<td>0.009</td>
<td>2.78</td>
</tr>
</tbody>
</table>

### TABLE IV. Repeatability Results

<table>
<thead>
<tr>
<th>Database</th>
<th>Mean over 20 runs</th>
<th>Seed</th>
<th>Set Reduction (%)</th>
<th>Reduction Accuracy (%)</th>
<th>Original Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spambase [33]</td>
<td>1973972471</td>
<td>141.452</td>
<td>92.728</td>
<td>93.436</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1951418661</td>
<td>1.509</td>
<td>0.211</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1959983081</td>
<td>42.224</td>
<td>99.032</td>
<td>99.632</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1947860015</td>
<td>1.628</td>
<td>0.214</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1276913411</td>
<td>43.280</td>
<td>98.697</td>
<td>99.120</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1788613688</td>
<td>1.605</td>
<td>0.575</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2304602519</td>
<td>41.893</td>
<td>81.190</td>
<td>82.411</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1212464214</td>
<td>2.751</td>
<td>0.981</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ACKNOWLEDGMENT

This work was partly supported 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 and the University Research Board at the American University of Beirut in Lebanon.

### REFERENCES


