Step Function Approximation for Support Vector Reduction

Amin Allahyar[1] and Hadi Sadoghi Yazdi[1,2]

1-Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
2- Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad
E-mail: Amin.Allahyar@stu-mail.um.ac.ir, H-Sadoghi@um.ac.ir

Abstract

Since introduction, the Support Vector Machines (SVM) has been popularly used in machine learning and data mining tasks due to their strong mathematical background and promising result. Nevertheless, they are noticeably slow in the prediction stage. The speed is influenced by number of support vectors determined in the training phase. Motivated by this fact, several studies are done to reduce the number of support vectors. The reduction should consider the degeneration of learning quality and preserve it at much as possible. Most previous methodologies either reduce the training set or apply a post-processing step to reduce the number of support vectors. In this paper, we proposed a new SVM cost function called Step Regularized Support Vector Machine (SRSVM), which is a standard SVM with extra constrained to reduce the number of support vectors, which can be defined by user. Experimental results are done to evaluate the efficiency and speed of proposed algorithm. SRSVM are also compared to other related SVM algorithms. The comparisons showed that the proposed method is effective in reducing number of support vectors while preserving the high performance of the classifier.

Keywords: *Classification; Support Vector Machine; Reduce Complexity; Number of Support Vectors*

1. Introduction

C Machine learning is divided into two main field of research including Classification and Clustering. A classification algorithm tries to build a predictor so that it can predict the correct labels for each data point based on the previously seen set of training data points whose category memberships are known. An example of such task would be predicting the corresponding category of given email and assigning it into "spam" or "non-spam" classes. This can be done by analyzing the previous emails with known category and extracting some useful quantized features. Regularly this part is out of machine learning scope and will be done by the expert in that specific domain (*e.g.*, the users of Email provider in this example). Next, these extracted features will be used to train the corresponding predictor.

Since the last three decades, many different types of classifiers are proposed in the literatures. Some of them are including: Artificial Neural Network (ANN) [1], Decision Tree (DT) [2], Naïve Bayes (NB) [3], K-Nearest-Neighbors (KNN) [4] and Support Vector Machines (SVM) [5]. Each of these classifiers has specific properties, which make them valuable for variety of problem. For example, the ANN classifiers are capable of learning the predictor without explicit specification of functional or distributional form for the underlying model [6]. Decision Trees are known for their ability to solve the classification problem by divide-and-conquer approach that will greatly reduce the computational cost in the training and testing phase [7]. On the other hand, along with these capabilities, each category suffers

from set of drawbacks. For example although KNN classifiers are very simple and effective in classification, their performance are greatly influenced by selection of k value [4]. Similarly, the Support Vector Machines (SVM) classifiers are known for their high degree of generalization, which is backed by strong mathematical foundation [8]. This important property has made the SVM a very popular classifier in the machine learning studies and real world applications. Although, these classifiers provide a high degree of performance, they are slow in the test phase [9]. This is because after training phase, the closest data points to the separator hyperplane called Support Vectors (SV) need to be utilized to determine the discriminator function.

Many different methodologies have been proposed to reduce the number of support vectors without loss of much generalization performance. However, these previous methods either provide a sparse predictor function using an estimation to the solution of standard SVM classifier or learn this predictor on the nested subsets of the training set [10]. Recently, the Sparse Support Vector Classifier (SSVC) is also proposed which is based on approximating the zero norm with an Expectation–Maximization (EM) algorithm from a two-level hierarchical Bayes model [10]. Although it was a powerful model, utilizing the EM algorithm to optimize the cost function may cause the algorithm to get stuck in local minima. In this paper, inspired by SSVC, we propose a modified SVM called SRSVM to approximate the zero norm. In the contrary to SSVC, the SRSVM cost function is formulated as a constrained nonlinear multivariable function, which can be globally solved. Another remarkable property of SRSVM is that it provides an option for user to determine his desired number of support vector. This is crucial in environments with specific amount of memory or computation resource.

The rest of this paper is organized as follows. In the preliminary section, the SVM formulation and its properties are discussed. Further, its computational dependency on the number of support vectors is demonstrated. We present a review of previous works aimed to provide a sparse discriminator function in Section 3. Section 4 is dedicated to SRSVM algorithm and its corresponding discussions. The experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2. Preliminary

First, we define our notations along with the assumptions made in this paper to provide a proper perspective for our contribution. Scripted letters such as $\mathcal{X}$ represent sets. Capital letters like X and W are matrixes, while bold lower case letters like $\boldsymbol{x}$ and $\boldsymbol{u}$ show column vectors. Lower case letters indicate scalars *e.g.*, n and d. In this paper, we represent functions with a bar sign above their name *e.g.*, $\overline{Ne}(.)$. Similar to popular notation, we use subscripts to index elements of a matrix or vector. For example, x_i is i-th element of vector $\boldsymbol{x}$ and $\boldsymbol{w}_j$ is the j-th column vector in matrix W. We will use a subscript below the vector norm notation to show its class. For example, $\|.\|_0$ indicate the zero norm while $\|.\|_1$ will be a l_1 norm. For simple representation, the l_2 norm will be denoted by $\|.\|$ *e.g.*, $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$. There are l training samples where the i-th data point denoted by $\boldsymbol{x}_i$ is from d dimensional space e.g. $\boldsymbol{x}_i \in \mathbb{R}^{d\times 1}$ with its corresponding label y_i. We assume two class in the dataset *e.g.*, $y_i \in \{+1, -1\}$. Therefore, the training dataset $\mathcal{X} \in \mathbb{R}^{d\times l}$ is divided into two sets represented by $\mathcal{X}_+$ and $\mathcal{X}_-$ respectively. Next, we will review the SVM formulation and its properties.

2.1. Support Vector Machines and their Computational Cost

The SVM classifier tries to find a linear-hyper plane to discriminate between data points with negative and positive labels [5]. A hyperplane definition can be formulized as follows:

$$\overline{f}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$$

Where $\boldsymbol{w}$ is the vector containing hyperplane weights and b is the bias. For any point lies on this hyperplane we have $\overline{f}(\boldsymbol{x}) = 0$. The goal is to find optimal values for $\boldsymbol{w}^*$ and b such that the following relation holds.

$$\overline{f}(\boldsymbol{x}_i) = \boldsymbol{w}^{*T}\boldsymbol{x}_i + b^* > 0 \quad \text{if } y_i = +1$$
$$\overline{f}(\boldsymbol{x}_i) = \boldsymbol{w}^{*T}\boldsymbol{x}_i + b^* < 0 \quad \text{if } y_i = -1$$

These conditions can be rewritten as (1):

(1)

$$y_i\left(\boldsymbol{w}^{*T}\boldsymbol{x}_i + b^*\right) > 0, \quad \forall(\boldsymbol{x}_i, y_i) \in \mathcal{X}$$

Assuming that such linear function exists to correctly classifies every point in $\mathcal{X}$, (1) can be reformulated to construct a *Margin* between two classes.

$$y_i\left(\boldsymbol{w}^{*T}\boldsymbol{x}_i + b^*\right) \geq 1, \quad \forall(\boldsymbol{x}_i, y_i) \in \mathcal{X}$$

The distance from the hyperplane to a vector $\boldsymbol{x}_i$ can be calculated as $\frac{\overline{f}(\boldsymbol{x}_i)}{\|\boldsymbol{w}\|}$. Therefore, the margin can be formulized as (2) [11].

(2)

$$Margin = \frac{1}{\|\boldsymbol{w}\|}$$

Maximizing (2) is equal to minimizing $\|\boldsymbol{w}\|$. Thus a constrained optimization problem, which is called *Primal* problem, can be defined as (3) [12].

(3)

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \overline{J}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t. \quad & y_i\left(\boldsymbol{w}^{*T}\boldsymbol{x}_i + b^*\right) \geq 1, \quad \forall(\boldsymbol{x}_i, y_i) \in \mathcal{X} \\ & \forall i \in \{1, \dots, l\} \end{aligned}$$

Where the $\frac{1}{2}$ in the cost function is used for mathematical convenience. In case of non-linear separable data points, we need to add *slack* variables ξ_i as (4).

(4)

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & \overline{J}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + c\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & y_i\left(\boldsymbol{w}^{*T}\boldsymbol{x}_i + b^*\right) + \xi_i \geq 1, \quad \forall(\boldsymbol{x}_i, y_i) \in \mathcal{X} \\ & \xi_i \geq 0 \\ & \forall i \in \{1, \dots, l\} \end{aligned}$$

Wshere ξ_i are the slack variables. The vector $\boldsymbol{\xi}$ contains these variables *e.g.*, $\boldsymbol{\xi} = [\xi_1 \quad \xi_2 \quad \dots \quad \xi_l]^T$ and vector $\mathbf{1}$ is a column vector with its elements equal to 1. The

parameter c is a trade-off constant between the margin and training set error. The kernel trick can be easily incorporated to SVM cost function as follows.

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & \bar{J}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + c\mathbf{1}^T\boldsymbol{\xi} \\ s.t. \quad & y_i\left(\boldsymbol{w}^{*T}\overline{\varphi}(\boldsymbol{x}_i) + b^*\right) + \xi_i \geq 1, \quad \forall(\boldsymbol{x}_i, y_i) \in \mathcal{X} \\ & \xi_i \geq 0 \\ & \forall i \in \{1, \dots, l\} \end{aligned}$$

Where $\overline{\varphi}(.): \boldsymbol{\mathcal{X}} \to \mathcal{H}$ is a non-linear function, which transforms the input space to the higher dimensional space to provide a better class separation. The *Dual* problem can be formulized as (5).

(5)

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \overline{D}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{\alpha}^T H\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ s.t. \quad & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq c \\ & \forall i \in \{1, \dots, l\} \end{aligned}$$

Where $\boldsymbol{y} = [y_1 \quad y_2 \quad \dots \quad y_l]^T$ and $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_l]$. The matrix H is symmetric with its element defined as follows.

$$H_{ij} = y_i y_j \overline{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$H_{ij} = y_i y_j \overline{\varphi}(\boldsymbol{x}_i)^T \overline{\varphi}(\boldsymbol{x}_j)$$

Finally, after solving (5) the discriminator function will take the form (6).

(6)

$$\overline{f}(\boldsymbol{x}) = \sum_{i=1}^{l} y_i \alpha_i^* \overline{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) + b^*$$

Where α_i^* and b^* is the optimal solution of (5) cost function.

As described, the coefficients α_i are the Lagrange multipliers. Regularly, many α_i with low values will appear after solving (5) [10]. The data points with corresponding coefficient $\alpha_i \neq 0$ are called *Support Vectors* (SV). Geometrically, these data points lay on the margin boundaries. In the other word, the data points of each class with smallest distance to the discriminating hyperplane are support vectors. According to (6), the computational cost for predicting the label for each test data point is directly related to the number of support vectors. Fortunately, there may be a situation where some support vectors can be deleted without loss of much accuracy or generality. An example of such cases are represented in Figure 1.

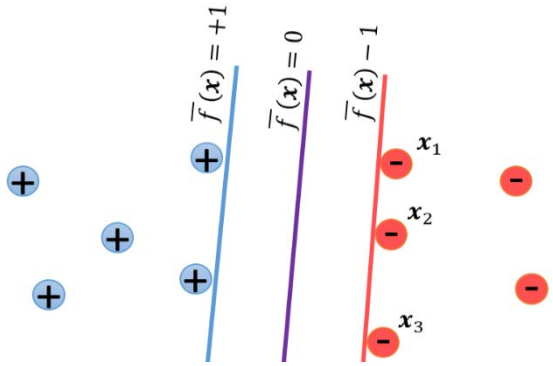


Figure 1. Demonstration of Redundant Support Vectors. In this Case, Data Point x_1 or x_2 or x_3 can be Removed without Loss of Accuracy or Generality

The number of these redundancies greatly increases when the SVM classifier is trained on a non-separable training set. Therefore, many studies are done to reduce the number of support vectors in the final result of SVM classifier. In the next section, we will review the recent proposed algorithm to find redundant support vectors.

3. Related Work

Burges method can be considered as one of the early works in support vector reduction [13]. His algorithm computes an estimation of discriminator function with a reduced set of vectors. For the discussion on the reduced set selection and reduced set construction, please refer to Schölkopf study [14]. The Burges method suffers from two main issues. First, it has a high computation cost and second it may be stuck at local minima. An improved version of Burges algorithm is introduced by Nguyen *et al.*, [15]. Their reduction algorithm iteratively selects two nearest support vectors from same class and replaces them by a newly constructed vector. In 2002, Downs et al. proposed an algorithm to detect the support vectors with linear dependency properties [16]. These unnecessary support vectors can be deleted from solution without any degeneration of accuracy or generalization. Instead of finding the important support vectors, Li and Zhang proposed an iterative procedure to train an SVM classifier with the most important training points [17]. In their procedure, a preprocessing step is utilized to detect the outliers to increase the convergence speed of the iterative algorithm. This process is repeated until the SVM classifier reaches a minimum acceptable accuracy or the number of support vectors stabilizes for a several consecutive iterations.

Lee and Mangasarian proposed Reduced Support Vector Machines (RSVM) algorithm [18]. In RSVM, a random subset of the training data points is selected. Next, an SVM classifier is trained using this set and their optimal set of α_i is determined using the l_1 regularizer $\|\boldsymbol{\alpha}\|_1$. As the training set is chosen randomly, this method requires many more training data points compared to standard SVM in order to achieve a degree of accuracy close to the standard SVM classifier trained over whole training dataset. The RSVM model is theoretically explored by Huang and Lee from the viewpoint of robust design in model building [19]. Their investigation showed that the uniform random selection of a reduced set in RSVM is the optimal robust selection scheme in terms of three main criteria including minimum model variation, maximal model bias and the minimal test power [19]. Keerthi *et al.*, suggested a righteous modification to RSVM, which was basically a greedy approach to iteratively select the subset of the training data point to form the representation [20]. Similar

to Li model, the Keerthi's iterative method can be stopped when the accuracy of trained SVM reaches to a certain amount of complexity.

Brank *et al.*, studied the possibility of using linear SVM for feature selection problem [21]. They showed that how this method and other feature selection algorithms could be used to develop a tradeoff between the training dataset size and the sparsity of the document representation for the fixed amount of system resource. Li *et al.*, improved this idea and proposed an adaptive greedy method called Feature Vector Selection (FVS) to choose appropriate feature vectors from the solution's support vectors based on vector correlation principle [22]. Their method can determine the structure of the feature space by resembling a basis of the support vector solutions. Therefore, the statistical information in the trained SVM classifier is maintained. Similar to our algorithm, in FVS, the number of the support vectors can be selected adaptively according to resource specification.

In 2010, Huang *et al.*, proposed Sparse Support Vector Classification (SSVC) [10]. Unlike the previous methods that find an approximation solution of SVM classifier or train it on the nested subsets of the training set, SSVC imports the l_0 norm regularization term of support vectors into the primal optimization problem and iteratively compute the enhanced cost function on the training set until it converges to a highly sparse solution. The SSVC is essentially an extension of Sparse Probit Classifier (SPC) proposed by Figueiredo [23, 24]. Although the SSVC is a powerful model, it utilizes an Expectation Maximization algorithm to find the optimal solution. Therefore, it may stuck to local minima and need to be executed with several starting point to reach a good enough solution. Inspired by SSVC, we propose SRSVM by modifying the cost function of standard SVM classifier. The SRSVM cost function is a constrained nonlinear multivariable function, which can be globally solved. On the other hand, similar to FVS, the SRSVM provides an option for user to determine his desired number of support vector. Thus, the trade-off between generalization and complexity of the SRSVM model can be directly controlled by user.

4. Proposed Algorithm

In this section, we present SRSVM to reduce the number of support vectors in the final solution of SVM classifier. This is done by reformulating the standard SVM cost function to a constrained nonlinear multivariable function, which enforce the sparseness of SVM model. Following the general goal, we would like to find the solution of SVM cost function where number of support vectors be an small as possible without loss of much generality or accuracy. This goal can be formulated as (7).

$$\min_{\boldsymbol{\alpha}} \quad \|\boldsymbol{\alpha}\|_0 \qquad s.t. \quad \left\{ \begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & \overline{D}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{\alpha}^T H \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \\ s.t. & \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \\ & 0 \le \alpha_i \le c \\ & \forall i \in \{1, \dots, l\} \end{array} \right\} \tag{7}$$

However, using $\|\boldsymbol{\alpha}\|_0$ makes (7) a non-convex function and therefore the analytic solution cannot be found. Instead, the number of Lagrange multipliers that is not equal to zero will be controlled by adding a new constraint to the problem. The l_0 norm of a vector is equal to sum of a step function applied to each elements of that vector. In the other word, we have (8).

(8)

$$\|\boldsymbol{\alpha}\|_0 = \sum_{i=1}^{l} \overline{step}(\alpha_i)$$

Where the sign function is defined as

$$\overline{step}(q) = \begin{cases} +1 & q > 0 \\ 0 & q \leq 0 \end{cases}$$

As discussed, we would like to provide a parameter for user to control the tradeoff between generalization and complexity of the model. Therefore, the goal is to find the solution of standard SVM classifier where (9) hold.

(9)

$$\|\boldsymbol{\alpha}\|_0 = \sum_{i=1}^{l} \overline{step}(\alpha_i) \leq k$$

Where k is the tradeoff control parameter defined by user. However, this constrain is still a non-convex function. We suggest using an approximation of l_0 norm. To approximate l_0 norm we propose $g(x)$ which can be defined as follows.

$$g(x) = \frac{1}{1 + e^{-\eta x}}$$

This function can provide a close approximation to the step function. In addition, $g(x)$ is a convex function while effectively stimulate the behavior of sign function. This fact is demonstrated in Figure 2. Using

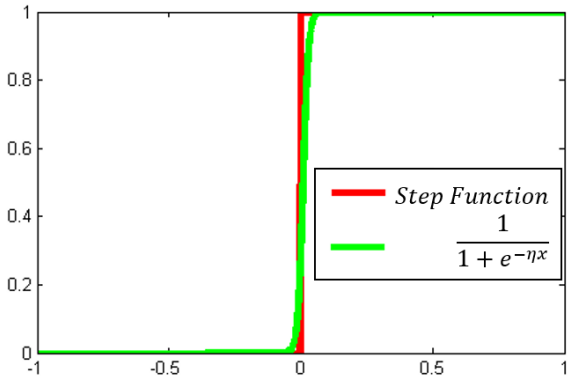


Figure 2. Approximating the Step Function (red) with $g(x)$ Function (green)

The modified constrain can be defined as follows.

$$\|\boldsymbol{\alpha}\|_0 \approx \sum_{i=1}^{N} g(\alpha_i) \leq k$$

(10)

$$\|\boldsymbol{\alpha}\|_0 \approx \sum_{i=1}^{N} \frac{1}{1 + e^{-\eta\alpha_i}} \leq k$$

The constrain $0 \leq \alpha_i \leq c$ in (7) implies that every coefficient α_i to be positive. Therefore using a large enough value for η, the $\|\boldsymbol{\alpha}\|_0$ can be closely approximated by $g(x)$ function. We will use $\eta = 10^2$ in our experiments. By applying the above modification, (7) can be reformulated as (11).

(11)

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \overline{D}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{\alpha}^T H \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \\ s.t. \quad & \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \\ & \sum_{i=1}^{N} \frac{1}{1 + e^{-\eta \alpha_i}} \leq k \\ & 0 \leq \alpha_i \leq c \\ & \forall i \in \{1, \dots, l\} \end{aligned}$$

This cost function is in form of constrained nonlinear multivariable function and therefore its globally optimal solution can be found. There is also several toolbox to solve such functions including: fmincon (Matlab), NLopt (C++) and bfgsmin (Octave).

5. Experimental Result

Although SRSVM algorithm is studied theoretically in previous section, but it needs to be experimentally examined to represent its effectiveness. In this section, we discuss about the experiments that have been done to investigate the efficiency of SRSVM algorithm and report their corresponding results.

5.1. Setup

We will compare the SRSVM with the closest algorithms in support vector reduction. These rival algorithms that are discussed in the related work section are including: Nguyen's algorithm (NSVM), Downs's algorithm (DSVM), Lee and Mangasarian's iterative model (RSVM) and Lee's procedure (FVS) and Huang's methodology (SSVC). In our experiments, we will measure the accuracy of final solution and the speed of training such model. To reach a fair comparison, this measurement will be done based on the number of support vectors provided by each algorithm. In the other word, for algorithms with a k parameter for enforcing the exact number of support vectors *e.g.*, SRSVM and FVS, the k will be set to a specific value. For other approaches *e.g.*, SSVC and DSVM, we will adjust the regularizer parameter until the exact number of support vectors is returned by each algorithm. In all experiments, we will use a constant range for acceptable number of support vectors. This range starts from $k = 5$ with increase of 5 in each step until $k = 60$. Therefor we will examine each algorithm twelve times *i.e.*, $\frac{60}{5} = 12$. The accuracy in larger number of support vectors did not examined because with larger number of support vectors, the accuracy would be close to the standard SVM.

We have used 10-Folds cross validation in the experiments. The whole dataset is divided into 10 partitions. One partition is the test set and the other nine parts are considered as training sets. Next, the number of support vectors k will change and the accuracy of each specific algorithm is measured by 10-Folds cross validation using new value for k. The measurement is repeated 5 times with randomly selected set of labeled data points in the training set. This process is done for each folds, therefore the experiment will be repeated 50 times (5 × number of folds). We will name the results with the number of support vector k.

For example, the result indicated with 5 indicates that the algorithms are forced to provide only 5 support vectors. In all experiments, each dataset is normalized to have zero mean and unit variance. For input datasets, we used synthetic and UCI datasets. Further to examine the effectiveness of SRSVM in real world problems, it is applied to English-Persian Sentence-Aligning problem [25]. Finally, as the SVM classifier is only capable of discriminating two classes, we need a method to convert the datasets with higher number of classes into two-class dataset. To do this, we merged two classes with smallest number of data points together and constructed a new class. This process is repeated until two classes remain.

5.2. Performance Measure

For performance measure, we used F-Measure. F-Measure is a very popular measurement in machine learning experiments. It can be defined as follows.

$$\overline{fm}(cm) = 2 \times \frac{\overline{r}(cm) \times \overline{p}(cm)}{\overline{r}(cm) + \overline{p}(cm)}$$

Where cm is the confusion matrix that is calculated from ground truth and predicted labels. $\overline{r}(.)$ and $\overline{p}(.)$ are precision and recall respectively which can be calculated as follows.

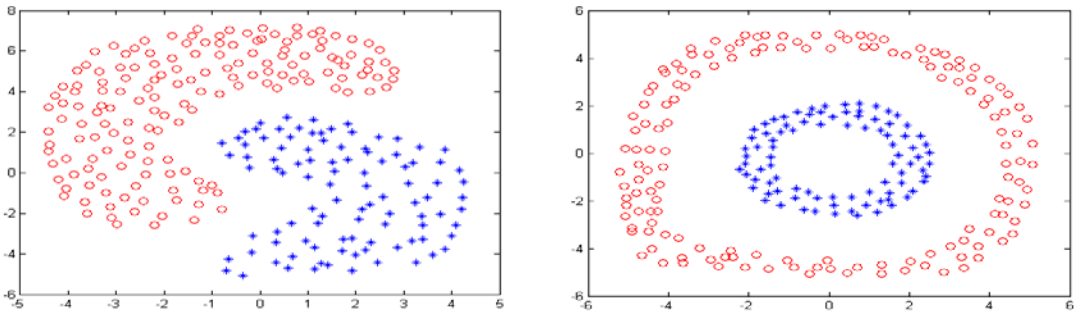
$$\overline{r}(cm) = \frac{tp}{tp + fn}$$

$$\overline{p}(cm) = \frac{tp}{tp + fp}$$

In this formulation tp, fp and fn are true positive, false positive and false negative respectively.

5.3. Datasets

For input dataset, three main types of data are considered in the experiments. The first category is the synthetic datasets including two moons, four clusters and two rings. These datasets are represented in Figure 5. In this figure, the data points belong to each cluster are depicted with different colors and markers.



a b

Figure 5. Three Synthetic Datasets used for Comparisons. a) Two Moons, b) Two Rings

The second category is UCI repository dataset. We used a variety of datasets with low, medium and high amount of sample sizes and features. Table 1 represents the selected UCI datasets. In the last column of this table, we represented how several classes is merged to form a two-class dataset. The last category is a real world dataset, which is prepared by Web

Technology Lab (WT-LAB) in Ferdowsi University of Mashhad. This dataset consist of 3235 aligned bilingual English–Persian sentences with 26 dimensions and 3 main classes of 1-2, 2-1 and 2-3. We have merged 2-1 and 2-3 classes to reach a two-class dataset.

Table 1. The Collection of UCI Datasets Selected for Comparison

Name	Instances	Cluster	Attribute	Merged Classes
Soybean	47	4	35	{1, 2} {3, 4}
Iris	150	3	4	{1, 2} {3}
Wine	178	3	13	{1, 3} {2}
Sonar	208	2	60	{1} {2}
WDBC	569	2	30	{1} {2}
Scale	625	3	4	{1, 2} {3}
Vehicle	846	4	18	{1, 4} {2, 3}
Vowel	990	11	10	{1, 2, 3, 4, 7, 10, 11} {5, 6, 8, 9}
Waveform	5000	3	21	{1, 2} {3}

5.4. Experiment on Synthetic Dataset

In this experiment, we will examine the effect of reducing the number of support vectors with SRSVM compared to standard SVM. Here the number of allowed support vector is set $k = 30$. Result of such experiment is given in Figure 6.

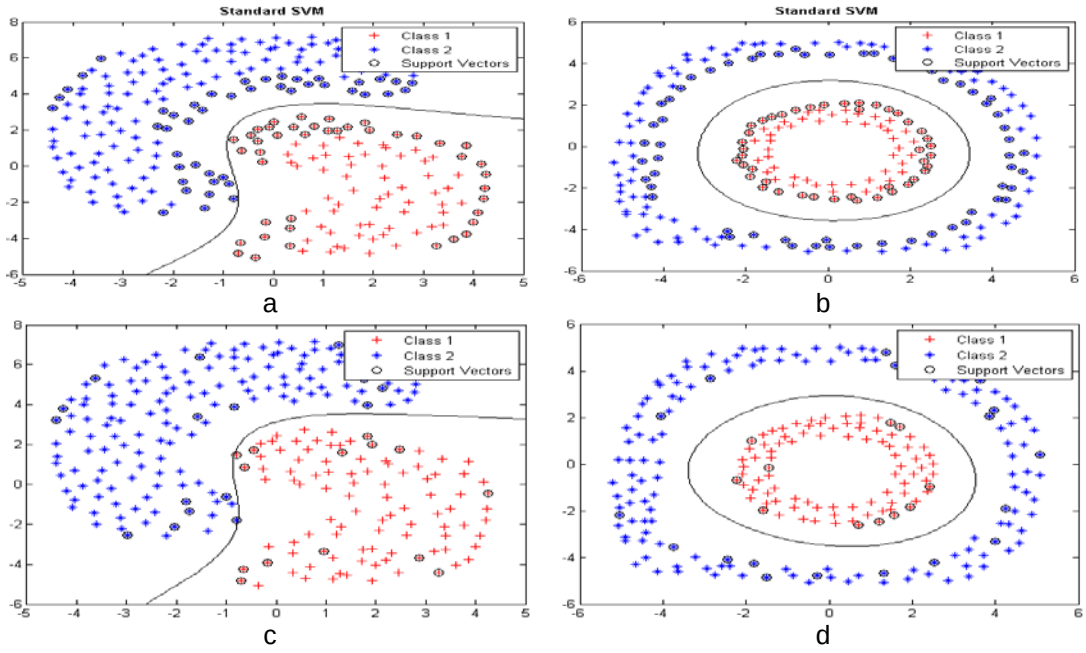


Figure 6. Result of Running Standard SVM and SRSVM in Two Synthetic Datasets. a, b are Result of SVM and c, d are Result of SRSVM

In the next experiment we will measure the result of considering different values for k. As described, the number of allowed support vector is started from 5. In the next iteration, k value increases by 5. These iterations are repeated until k reaches 30. The accuracy of each selected algorithm with 10-Fold cross validation is measured in each iteration. We did not continue this procedure because the experimental result for all algorithm reached 100%

accuracy and investigation with larger number of support vectors would return 100% accuracy. Figure 7 shows the F-Measure result of running selected algorithms with specific value for *k* on the synthetic datasets.

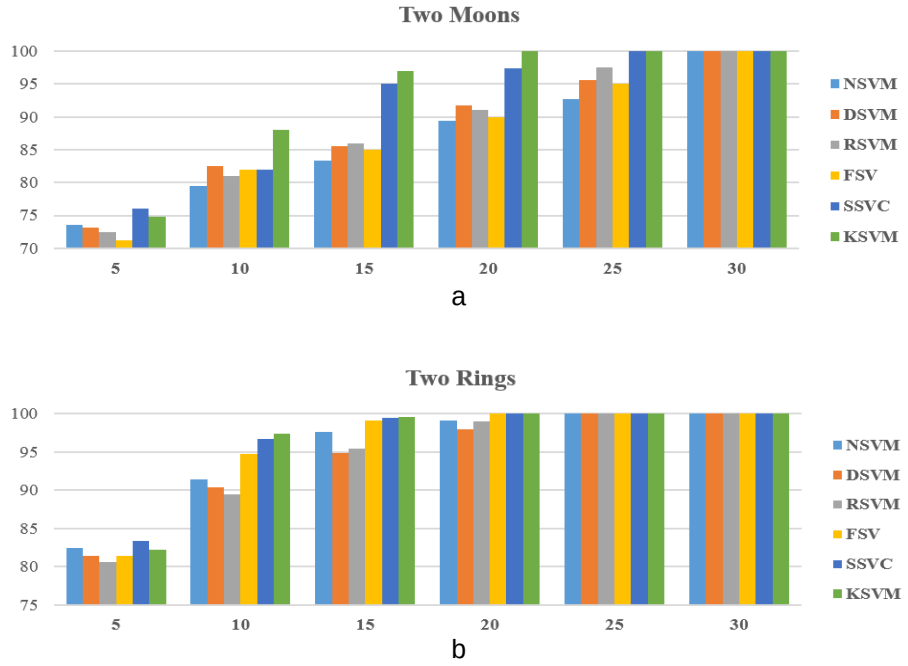
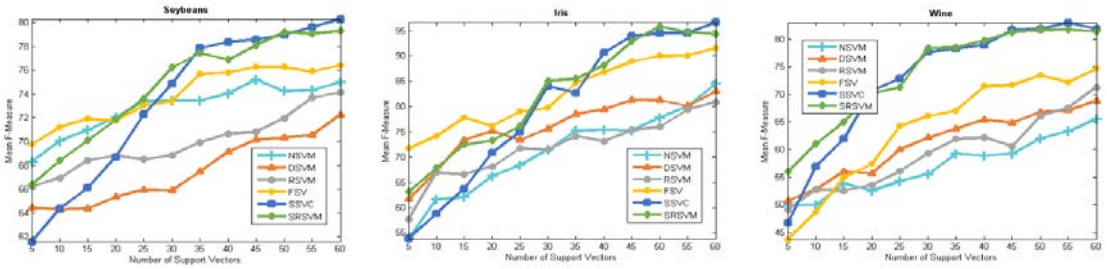


Figure 7. Result of Running each Selected Algorithm with Different Amounts of Allowed Support Vectors. a) Two Moons, b) Two Rings

According to these experiments, SRSVM provides promising results. However, the effectiveness of these algorithms in simple datasets cannot be a dependable measure. In the next section, we will measure these algorithms with UCI repository datasets.

5.5. Experiment on UCI Dataset

In this part, we examine our algorithm with UCI repository [1] datasets. Each set of experiments for specific number of support vectors is repeated 5 times with 10-folds cross validation. The average F-Measure is reported in Figure 8.



[1] http://archive.ics.uci.edu/ml/

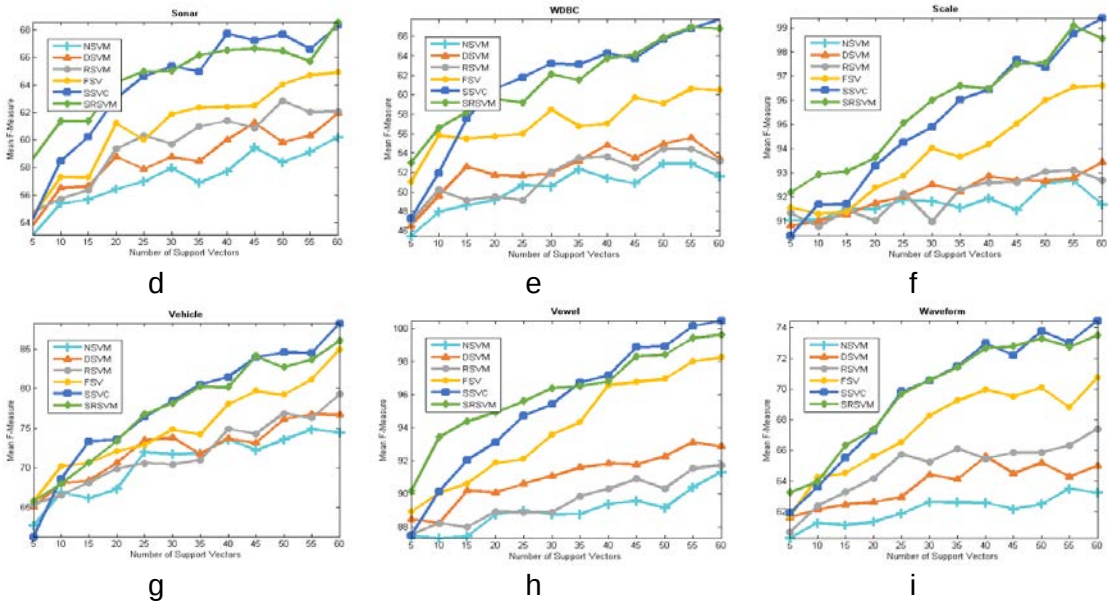


Figure 8. Result of Running each Selected Algorithm 5 Times with 10-folds Cross Validation with Different Amounts of Allowed Support Vectors on UCI Repository Datasets. a) Soybean, b) Iris, c) Wine, d) Sonar, e) WDBC, f) Scale, g) Vehicle, h) Vowel, i) Waveform

These results show that the FSV, SSVC and SRSVM algorithms can provide a more suitable support vectors in compared to other algorithms (*e.g.*, NSVM, DSVM and RSVM). Interestingly, the SRSVM provides a higher F-Measure with some exceptions.

6. Conclusion

In this paper, we introduced a novel support vector reduction method called SRSVM. We claimed that the zero norm can be approximated with exponential function. Next we incorporated this function into the constrains of SVM dual form. The resulting cost function is in form of nonlinear multivariate function and its global solution can be easily found. We showed that SRSVM could be considered as an efficient method to reduce the number of support vector in an SVM classifier. Experimental results on the synthetic, UCI, datasets along with FEP-the sentence-aligning corpus-showed the superior performance of the proposed support vector reduction method. In the future, we will investigate other forms of constrains to better approximation of zero norm. We also intend to investigate the performance and efficiency of our method on the other real-world applications.

References

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain", Psychological review, vol. 65, **(1958)**, pp. 386.
[2] J. R. Quinlan, "Induction of decision trees", Machine learning, vol. 1, **(1986)**, pp. 81-106.
[3] D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval", Machine Learning: ECML-98, **(1998)**, pp. 4-15.
[4] K. Fukunaga, Introduction to statistical pattern recognition: Academic Pr, **(1990)**.
[5] V. Vapnik, The nature of statistical learning theory: springer, **(1999)**.
[6] G. P. Zhang, "Neural networks for classification: a survey", Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 30, **(2000)**, pp. 451-462.

[7] J. R. Quinlan, "Generating production rules from decision trees", Proceedings of the Tenth International Joint conference on Artificial intelligence, **(1987)**, pp. 304-307.
[8] L. Wang, "Support Vector Machines: theory and applications", Springer, vol. 177, **(2005)**.
[9] N. Cristianini and J. Shawe-Taylor, "An introduction to support vector machines and other kernel-based learning methods", Cambridge university press, **(2000)**.
[10] K. Huang, D. Zheng, J. Sun, Y. Hotta, K. Fujimoto and S. Naoi, "Sparse learning for support vector classification", Pattern Recognition Letters, vol. 31, **(2010)**, pp. 1944-1951.
[11] C. J. Burges, "A tutorial on support vector machines for pattern recognition", Data mining and knowledge discovery, vol. 2, **(1998)**, pp. 121-167.
[12] H. Yu and S. Kim, "SVM tutorial: Classification, regression, and ranking", Handbook of Natural Computing, **(2009)**.
[13] C. J. Burges, "Simplified support vector decision rules", Machine learning-international workshop then conference, **(1996)**, pp. 71-77.
[14] B. Scholkopf, S. Mika, C. J. Burges, P. Knirsch, K.-R. Muller and G. Ratsch, "Input space versus feature space in kernel-based methods", Neural Networks, IEEE Transactions on, vol. 10, **(1999)**, pp. 1000-1017.
[15] D. Nguyen and T. Ho, "An efficient method for simplifying support vector machines", Proceedings of the 22nd international conference on Machine learning, **(2005)**, pp. 617-624.
[16] T. Downs, K. E. Gates and A. Masters, "Exact simplification of support vector solutions", The Journal of Machine Learning Research, vol. 2, **(2002)**, pp. 293-297.
[17] Y. Li, W. Zhang and C. Lin, "Simplify support vector machines by iterative learning", Neural Information Processing: Letters and Reviews, vol. 10, **(2006)**, pp. 11-17.
[18] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines", Proceedings of the first SIAM international conference on data mining, **(2001)**, pp. 5-7.
[19] Y.-J. Lee and S.-Y. Huang, "Reduced support vector machines: A statistical theory", Neural Networks, IEEE Transactions on, vol. 18, **(2007)**, pp. 1-13.
[20] S. S. Keerthi, O. Chapelle, D. DeCoste, and P. Bennett, "Building support vector machines with reduced classifier complexity", Journal of Machine Learning Research, vol. 7, **(2006)**, pp. 1493-1515.
[21] J. Brank, M. Grobelnik, N. Milic-Frayling and D. Mladenic, "Feature selection using linear support vector machines," Proceedings of the 3rd International Conference on Data Mining Methods and Databases for Engineering, **(2002)**.
[22] Q. Li, L. Jiao and Y. Hao, "Adaptive simplification of solution for support vector machine", Pattern Recognition, vol. 40, **(2007)**, pp. 972-980.
[23] M. A. Figueiredo, "Adaptive sparseness using Jeffreys prior", Adv. NIPS, **(2001)**, pp. 697-704.
[24] M. A. Figueiredo, "Adaptive sparseness for supervised learning", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 25, **(2003)**, pp. 1150-1159.
[25] A. Toosi, M. Kahani and R. Saeedi, "A New Approach to English-Persian sentence alignment", presented at the 17th Annual Int. CSI Computer Conference (CSICC'2012), Tehran, Iran, (in Persian), **(2012)** March 07-09.

Author

Hadi Sadoghi Yazdi is currently an Associate Professor of Computer Science and Engineering at Ferdowsi University of Mashhad (FUM). He received his B.S. degree in Electrical Engineering from FUM in 1994, and received his M.S. and Ph.D. degrees in Electrical Engineering from Tarbiat Modares University in 1996 and 2005, respectively. Dr. Sadoghi Yazdi has received several awards including Outstanding Faculty Award and Best System Design Award in 2007. His research interests are in the areas of Pattern Recognition, Machine Learning, Machine Vision, Signal Processing, Data Mining and Optimization.