Reducing the Number of Support Vectors of SVM Classifiers using Smoothed Separable Case Approximation

Dries Geebelen, Johan A.K. Suykens, *Senior Member, IEEE,* and Joos Vandewalle, *Fellow, IEEE*

***Abstract*—In this letter we propose a new method to reduce the number of support vectors of Support Vector Machine (SVM) classifiers. We formulate the approximation of an SVM solution as a classification problem that is separable in the feature space. Due to the separability the hard-margin SVM can be used to solve it. This approach, which we call Separable Case Approximation (SCA), is very similar to the cross-training algorithm explained in [1] which is inspired by editing algorithms [2]. The norm of the weight vector achieved by SCA can, however, become arbitrarily large. For that reason we propose an algorithm, called Smoothed Separable Case Approximation (SSCA), that additionally upper bounds the weight vector of the pruned solution and that, for commonly used kernels, reduces the number of support vectors even more. The lower the chosen upper bound, the larger this extra reduction becomes. Upper bounding the weight vector is important because it ensures numerical stability, reduces the time to find the pruned solution and avoids overfitting during the approximation phase. On the examined datasets, SSCA drastically reduces the number of support vectors.**

***Index Terms*—Support vector machine classifier, sparse approximation, run time complexity.**

I. Introduction

CLASSIFYING data is a common task in machine learning. Given a set of input vectors along with corresponding class labels, the task is to find a function that defines the relation between input vectors and their class labels. Vapnik's support vector machine (SVM) classifier [3], [4] is a classifier that obtains good accuracy on real life datasets. Compared to other classification algorithms, the SVM classifier has an important drawback: the run time complexity, as explained in [5], [6], can be considerably higher due to the large number of support vectors.

Several methods have been proposed for pruning the solutions of SVMs. In Burges [5] and Schölkopf et al. [7], an algorithm is proposed that approximates the weight vector such that the distance to the original weight vector is minimized. Downs et al. [8] present an algorithm that eliminates support vectors linearly dependent on the other support vectors. Osuna and Girosi [6] used SVM itself as a regression tool to approximate the decision surface. Keerthi et al. [9] present an effective greedy method SVMs which uses a basis selection criterion that is directly related to the training cost. In fixed-size LS-SVM [10], [11] an active subset selection method based on quadratic Renyi entropy is used to select basis functions after which a training cost similar to that of LS-SVM is optimized. Bakir et al. [1] propose to selectively remove examples from the training set using probabilistic estimates related to editing algorithms.

In this paper, an algorithm, called Separable Case Approximation (SCA), is discussed that improves the sparsity of SVM classifiers. SCA is strongly related to the cross-training algorithm discussed in [1]. SCA first makes the training set separable by removing misclassified examples or by swapping their labels and then separates the dataset using Vapnik's SVM classifier for the separable case. Additionally, we propose an extension called Smoothed Separable Case Approximation (SSCA) that upper bounds the weight vector of the pruned solution. In many cases, SSCA results in even fewer support vectors than SCA when using a radial basis function (RBF) kernel or polynomial (Pol) kernel.

The remainder of this letter is organized as follows. In Section II the separable and non-separable cases are briefly reviewed. Section III explains how approximating an SVM classifier can be formulated as a classification problem and presents two algorithms, SCA and SSCA, to solve this classification problem. Existing pruning methods are reviewed in Section IV and their performances are compared to that of SSCA in Section V. Finally, Section VI makes some concluding remarks.

II. Support Vector Machine Classifier

Given a set of N training examples with input vectors $\{x_k \in \mathbb{R}^n\}_{k=1}^N$ and corresponding labels $\{y_k \in \{-1,1\}\}_{k=1}^N$, Vapnik's SVM classification algorithm builds a linear classifier in the feature space

$$\hat{y}(x) = \text{sign}(w^T\varphi(x) + b) \tag{1}$$

where the feature map $\varphi(x)$ maps the input data into a feature space. It solves the following primal problem

$$\min_{w,b,\xi} J_P(w) = \frac{1}{2}w^Tw + C\sum_{k=1}^N \xi_k$$

$$\text{such that } \begin{cases} y_k[w^T\varphi(x_k)+b] \geq 1-\xi_k, & k=1,...,N \\ \xi_k \geq 0, & k=1,...,N \end{cases} \tag{2}$$

D. Geebelen, J.A.K. Suykens and J. Vandewalle are with the Department of Electrical Engineering (ESAT-SCD-SISTA). Katholieke Universiteit Leuven, B-3001 Leuven (Heverlee), Belgium. Phone: +32-16-321709, fax: +32-16-321970, e-mail: dries.geebelen@esat.kuleuven.be.

Research supported by Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet , CoE EF/05/006 Optimization in Engineering (OPTEC) en PFV/10/002 (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC) IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011) ; IBBT EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC HIGHWIND (259 166) Contract Research: AMINAL Other: Helmholtz: viCERP ACCM. The scientific responsibility is assumed by its authors. JVDW is full professor at the Katholieke Universiteit Leuven, Belgium. JS is a professor at the Katholieke Universiteit Leuven, Belgium.

where C is a positive real constant. The hyperparameter C determines the trade-off between the margin and the training error. In the non-separable case ($C < \infty$), Vapnik's SVM classifier tolerates misclassifications in the training set due to the slack variables (ξ_k) in the problem formulation. In the separable case ($C = \infty$) all ξ_k equal 0 which implies misclassifications aren't tolerated. The dual after applying the kernel trick becomes

$$\max_{\alpha} J_D(\alpha) = -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l K(x_k, x_l)\alpha_k\alpha_l + \sum_{k=1}^{N}\alpha_k$$
$$\text{such that } \begin{cases} \sum_{k=1}^{N}\alpha_k y_k = 0 \\ 0 \le \alpha_k \le C \qquad k = 1, ..., N. \end{cases} \tag{3}$$

According to Mercer's theorem [12], any positive-definite kernel function can be expressed as

$$K(x, z) = \varphi(x)^T\varphi(z).$$

Due to the kernel trick, we can now work in huge dimensional feature spaces without having to perform calculations explicitly in this space. Popular nonlinear kernel functions are the radial basis function (RBF) kernel ($K(x, x_k) = \exp(-\|x - x_k\|_2^2/\sigma^2)$) and the polynomial (Pol) kernel ($K(x, x_k) = (x^T x_k + \tau)^d$ with $\tau \ge 0$). The predicted label (1) of a new data point becomes

$$\hat{y}(x) = \text{sign}\left(\sum_{k=1}^{N}\alpha_k y_k K(x, x_k) + b\right). \tag{4}$$

The run-time complexity scales linearly with the number of support vectors. For the non-separable case, the support vectors are characterized by

$$y_k[w^T\varphi(x_k) + b] \le 1.$$

This means the support vectors are misclassified examples ($y_k[w^T\varphi(x_k) + b] \le 0$) or examples close to the decision boundary in the feature space ($0 < y_k[w^T\varphi(x_k) + b] \le 1$). For the separable case, all support vectors satisfy

$$y_k[w^T\varphi(x_k) + b] = 1.$$

This means that all the support vectors are on the hyperplane where $w^T\varphi(x) + b = 1$ and the hyperplane where $w^T\varphi(x) + b = -1$.

III. Separable Case Approximation

A. Approximating the Decision Boundary

This Subsection explains the approximation criterion SCA and SSCA try to minimize and shows how minimizing this approximation error can be formulated as a classification problem. The goal of sparse approximation is to approximate the SVM classifier defined by w and b by a sparser SVM classifier defined by w' and b' such that the classification on unseen data is as similar as possible. We define $\hat{y}(x)$ and $\hat{y}'(x)$ as the label of a datapoint according to the original and the approximating classifier respectively. The first approximation error measure we propose equals the probability that a datapoint is classified differently by $\hat{y}(x)$ and $\hat{y}'(x)$:

$$E_1 = P\left(\hat{y}(x) \ne \hat{y}'(x)\right). \tag{5}$$

If we modify the training set by flipping the labels of training examples misclassified by $\hat{y}$, then the approximation problem according to (5) becomes a classification problem over the modified training set. If it is only important to classify correctly classified datapoints similarly, then the approximation error becomes

$$E_2 = P\left(\hat{y}(x) \ne \hat{y}'(x) | \hat{y}(x) = y(x)\right). \tag{6}$$

If we modify the training set by removing training examples misclassified by $\hat{y}(x)$, then the approximation problem according to (6) becomes a classification problem over the modified training set.

In both cases the modified training dataset is separable in the feature space because the non-pruned solution separates it. This separability only holds when the SVM classifier used to solve the approximation problem has the same kernel as the original classifier. We exploit this separability to gain sparsity.

B. Separable Case Approximation (SCA)

The SCA algorithm realizes sparsity by using the SVM classifier for the separable case to solve the approximation problem:

Algorithm 1 Separable Case Approximation (SCA)

```
1) Find the non-pruned solution using Vapnik's SVM for
   the non-separable case.
2) Modify the training set such that it becomes separable
   by applying rule 2a or rule 2b to misclassified examples
   ($y_k[w^T\varphi(x_k) + b] \le 0$)
     • rule 2a: Flip their labels.
     • rule 2b: Remove them.
3) Given the modified training set, find the SVM-solution
   for the separable case.
```

The final support vectors are training examples close to the original separating hyperplane. Vapnik's SVM classifier for the separable case will, in general, not cause more support vectors than $p+1$, where p is the dimensionality of the feature space. This upper bound exists because all support vectors lie on one of two parallel hyperplanes in the feature space and the degrees of freedom for choosing two parallel hyperplanes in a p dimensional space equals $p+1$. Under certain non-generic circumstances, such as linear dependency between p or less training data points, this upper bound can be exceeded. This algorithm is very similar to the cross-training algorithm [1]. It differs in the sense that for step 1 the crosstraining algorithm divides the training set in subsets, trains an SVM on each subset and then combines these different SVM classifiers into a global classifier. The flipping of labels is also not discussed in [1].

Separable case approximation has an important drawback: $\|w'\|_2$ can become arbitrarily large because the hyperplane where $w'^T\varphi(x)+b=1$ and the hyperplane where $w'^T\varphi(x)+b=-1$ can lie arbitrarily close to each other. We want to upper bound $\|w'\|_2$ for the following reasons:

- We want to remove the ill-conditioning of the approximation problem. This ill-conditioning makes the solution numerically unstable and significantly increases the time that the SMO (Sequential Minimal Optimization) algorithm needs to find the solution.
- We want to avoid overfitting (in the sense that training data points can be classified similarly by the pruned and non-pruned classifiers while unseen data points are classified differently) by imposing smoothness.
- Ideally, upper bounding $\|w'\|_2$ decreases the number of support vectors.

To conclude this Subsection we apply SCA to the Ripley dataset using an SVM classifier with RBF kernel. The resulting classifier is visualized in Figure 1 and the resulting accuracies are shown in Table I. The number of support vectors reduces drastically. As expected, the training set accuracy does not decrease after the pruning. Figure 1 shows that the support vectors of the pruned solution lie close to the decision boundary of the non-pruned solution.

C. Smoothed Separable Case Approximation (SSCA)

The SSCA algorithm upper bounds $\|w'\|_2$ and goes as follows:

Algorithm 2 Smoothed SCA (SSCA)

```
1) Find the non-pruned solution using Vapnik's SVM for
   the non-separable case.
2) Modify the training set such that it becomes separable
   by applying rule 2a or rule 2b to misclassified examples
   ($y_k[w^T\varphi(x_k)+b] \leq 0$)
     • rule 2a: Flip their labels.
     • rule 2b: Remove them.
   and additionally remove training examples for which
   $y_k[w^T\varphi(x_k)+b] < D$ with $D > 0$.
3) Given the modified training set, find the SVM-solution
   for the separable case.
```

SSCA makes a trade-off between the approximation accuracy on the training set and the norm of the weight vector. Increasing D, decreases the lower bound on the approximation accuracy of the training set and decreases the upper bound on $\|w'\|_2$ because:

- All training examples at a distance higher than $\frac{D}{\|w'\|_2}$ from the separating hyperplane in the feature space are classified similarly by $\hat{y}$ and $\hat{y'}$.
- $\|w'\|_2$ is upper bounded by $\frac{\|w\|_2}{D}$ as we explain next.

In the modified training set, all examples satisfy:

$$y_k\left[\left(\frac{w}{D}\right)^T\varphi(x_k)+\frac{b}{D}\right] \geq 1. \tag{7}$$

Because w', the weight vector of the pruned solution, is found according to the maximal margin criterion, its 2-norm is the smallest of all weight vectors satisfying

$$y_k[w^T\varphi(x_k)+b] \geq 1, \quad k=1,...,N$$

for the modified training set. Hence, the following holds

$$\|w'\|_2 \leq \frac{\|w\|_2}{D}. \tag{8}$$

Using SSCA, we can upper bound $\|w'\|_2$. In many cases, increasing D reduces the number of support vectors even more when using a radial basis function (RBF) or polynomial (Pol) kernel as empirically shown in Section V. The exact reason for this extra reduction is unclear. The best value for D can be found using (cross-)validation. The parameter D can be optimized independently from the other hyperparameters. The overhead generated to determine D is thus relatively small.

The results for the Ripley dataset, when removing misclassified examples, are shown in Table II and visualized in Figure 2. Increasing D, decreases the training set accuracy, decreases $\|w'\|_2$ and does not increase the number of support vectors.

D. Non-Separable Case Approximation (NSCA)

The most straightforward way of upper bounding $\|w'\|_2$ is using the SVM for the non-separable case to solve the approximation problem. This implies an additional hyperparameter (C'). The results of applying Non-Separable Case Approximation (NSCA) on the Ripley dataset are shown in Table III: decreasing C', decreases the training set accuracy, decreases $\|w'\|_2$ and increases the number of support vectors. The increase in the number of support vectors for upper bounding $\|w'\|_2$ is a very big disadvantage that makes us prefer SSCA over this algorithm.

IV. Other Pruning Methods

This Section briefly explains other pruning algorithms.

A. Cross-Training

Cross-Training begins with partitioning the training set randomly into subsets and training independent SVMs on each subset. The decision functions of these SVMs are then used to discard two types of training examples, namely those which are confidently recognized, and those with are misclassified. A final SVM is trained using the remaining examples. The training of the final SVM can be done using SCA. The innovation of our algorithm compared to Cross-Training is due to the smoothing used in SSCA.

B. Greedy Iterative Methods

Greedy iterative methods, like Keerthi's method [9], incrementally choose basis functions according to a criterion that is directly related to the training cost function. In most cases, the basis functions correspond to training data points. In Keerthi's method, the training cost function to minimize, given a chosen set of basis functions ($\mathcal{J}$), becomes

$$\min_{\beta_J} f(\beta_J) = \frac{1}{2}\beta_J K_{JJ}\beta_J + \frac{C}{2}\sum_{1}^{n}\max(0, 1-y_i o_i)$$

where J contains the indexes of the basis functions belonging to $\mathcal{J}$, $K_{ij} = k(x_i, x_j)$, $o_i = K_{i,J}\beta_j$ and K_{IJ} corresponds to the submatrix of the kernel matrix made of the rows indexed by I and columns indexed by J.

C. Fixed-Size LS-SVM

In fixed-size LS-SVM [10], the basis functions are first selected according to the quadratic Renyi entropy. Then, a cost function similar to that of LS-SVM is minimized in the primal space. Fixed-size LS-SVM is not an iterative method, the number of basis functions has to be chosen in advance or needs to be tuned. The bandwidth of the kernel used for the selection of the basis functions is, in our experiments, chosen according to the Improved Silverman's Normal Rule of Thumb (ISNR) [13].

V. Experimental Results

We will compare SSCA with Keerthi's method [9] and fixed-size LS-SVM [10]. The datasets used are Adult (*adu*) and Protein (*pro*). The dataset *adu* has been obtained from the UCI benchmark repository [14], *pro* has been obtained from LIBSVM [15] and is used in [16]. The properties of the datasets are shown in Table IV. The dataset *pro* contains 3 classes. We modify this into a two class problem by merging the classes labeled as '1' and '2'. For the pruned solution we approximate the separable case by taking a very large, but finite, value of C. This reduces the impact of the ill-conditioning. LIBSVM [15] is the software used to find both the non-pruned and the pruned solution. The experiments ran in Matlab on a Dell OptiPlex 780 with Windows 7 Professional, 4.0 GB RAM and a Intel(R) Core(TM)2 Quad Processor Q9550 (2.83 GHz).

In Table V we can see that flipping labels doesn't cause a significantly better trade-off between the test set accuracy and the number of support vectors compared to removing misclassified examples for *adu(rbf)* and *pro(rbf)*. In further experiments we will always apply SSCA with removal of misclassified labels. Figure 3 shows SSCA drastically reduces the number of support vectors. On *adu* Keerthi's method has mostly the best trade off between test set accuracy and the number of support vectors. On *pro(pol)* SSCA has the better trade off for 300 support vectors. The differences in performance can be partly caused by the differences in loss function. To compare the times required to find the solution, we have to take the time needed to tune the hyperparameters into account. In fixed-size LS-SVM the basis functions are chosen independently of the hyperparameters. This selection has to be done only once before tuning and the cost is thus not proportional to the number of tried hyperparameter values. Assuming that the number of values of hyperparameters to be tried exceeds 20, we can conclude from Table VI fixed-size LS-SVM is the fastest method followed by Keerthi's method. SSCA is almost as fast as SVM because the tuning of D can be done independently of the other hyperparameters.

VI. Conclusion

In this letter, we succeeded in drastically reducing the number of support vectors of SVM classifiers using Smoothed Separable Case Approximation (SSCA). Instead of approximating the underlying weight vector, we directly approximate the way the datapoints are classified. Compared to Separable Case Approximation (SCA), SSCA has the following benefits: it ensures numerical stability, reduces the time to find the pruned solution, avoids overfitting during the approximation phase and reduces the number of support vectors even more in certain cases. SSCA is slower than some other pruning algorithms, but almost as fast as finding the non-pruned solution.

Acknowledgements

The authors wish to thank an anonymous reviewer for the suggestion of flipping labels.

References

[1] G. H. Bakir, J. Weston, and L. Bottou, "Breaking SVM complexity with cross-training," *Advances in Neural Information Processing Systems,* vol. 17, pp. 81-88, 2005.

[2] P. Devijver and J. Kittler, *Pattern Recogniton, A statistical approach,* Prentice Hall, Englewood Cliffs, 1982.

[3] C. Cortes, V. Vapnik, "Support vector networks," *Mach. Learning,* vol 20, pp. 273-297, 1995.

[4] V. Vapnik, *Statistical learning theory,* New York: John Wiley & Sons, 1998.

[5] C.J.C. Burges, "Simplified support vector decision rules," *Proc. 13th Int. Conf. Mach. Learning,* pp. 71-77, 1996.

[6] E. Osuna and F. Girosi, "Reducing the run-time complexity of Support Vector Machines," *Int. Conf. Pattern Recognition*, Brisbane, Australia, August 16-20 1998.

[7] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K. Muller, G. Rätsch, and A.J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Netw.,* vol 10, no. 5, pp. 1000-1017, September 1999.

[8] T. Downs, K. E. Gates, and A. Masters, "Exact Simplification of Support Vector Solutions," *Journal of Machine Learning Research,* vol 2, pp. 293-297, 2001.

[9] S.S. Keerthi, O. Chapelle, and D. DeCoste, "Building Support Vector Machines with Reduced Classifier Complexity," *Journal of Machine Learning Research,* vol 7, pp. 1493-1515, 2006.

[10] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.

[11] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, and B. De Moor, "Optimized fixed-size least squares support vector machines for large data sets," *Computational Statistics and Data Analysis,* vol 54, no. 6, pp. 1484-1504, Jun. 2010.

[12] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London*, vol 209, pp. 415-446, 1909.

[13] N.L. Hjort, and D. Pollard, "Better rules of thumb for choosing bandwith in density estimation," *Statistical Research Report,* Department of Mathematics, University of Oslo, Norway, 1996.

[14] C. L. Blake and C. J. Merz (1998), UCI Repository of machine learning databases. Irvine, CA: University of California, Dept. of Information and Computer Science. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[15] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm. 2001.

[16] J.-Y. Wang. Application of support vector machines in bioinformatics. Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2002.

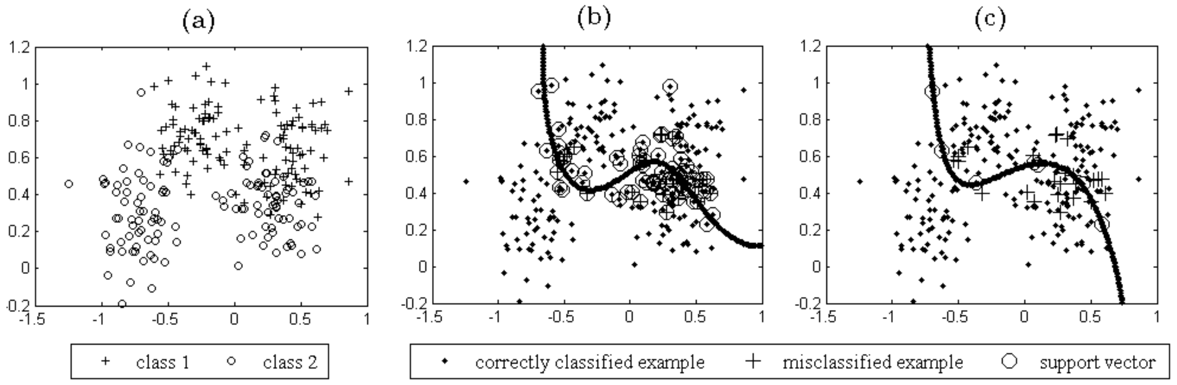


Fig. 1. The Ripley dataset contains 250 training data points and 1000 test data points. (a) Training examples of the Ripley dataset with corresponding labels; (b) non-pruned solution using Vapnik's SVM classifier for the non-separable case with $\sigma = 1$ and $C = 100$; (c) pruned solution using SCA with removal of misclassified examples.

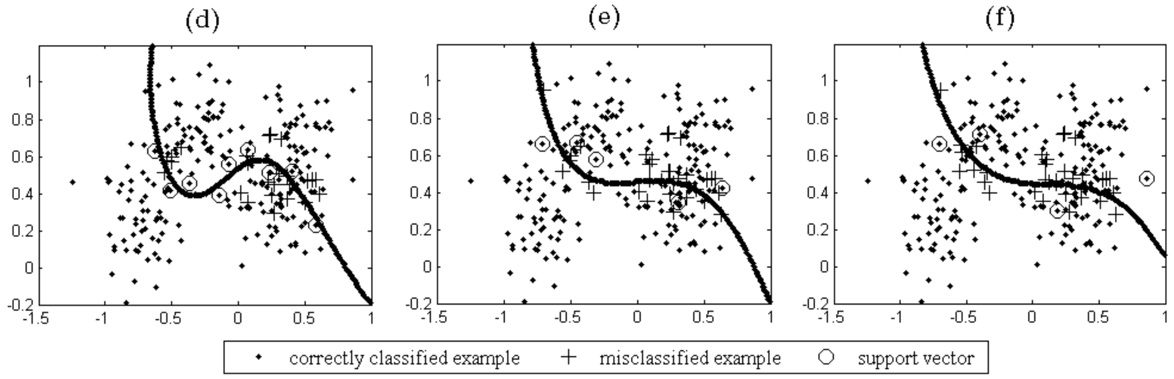


Fig. 2. SSCA with removal of misclassified examples applied to the Ripley dataset with $\sigma = 1$ and $C = 100$. Parameter D differs per figure: (d) $D = 0.3$; (e) $D = 1.3$; (f) $D = 2.0$.

TABLE I
SCA APPLIED TO THE RIPLEY DATASET. SV IS THE NUMBER OF SUPPORT VECTORS, $\mathrm{ACC_{TR}}$ IS THE TRAINING SET ACCURACY AND $\mathrm{ACC_{TEST}}$ IS THE TEST SET ACCURACY.

	SV	$\mathrm{ACC_{TR}}$(%)	$\mathrm{ACC_{TEST}}$(%)
non-pruned	78	89.6	89.7
remove	7	90.0	89.0
flip	8	89.6	89.0

TABLE II
SSCA WITH REMOVAL OF MISCLASSIFIED EXAMPLES APPLIED TO THE RIPLEY DATASET. D IS A PARAMETER OF THE SSCA ALGORITHM, SV IS THE NUMBER OF SUPPORT VECTORS, $\mathrm{ACC_{TR}}$ IS THE TRAINING SET ACCURACY, $\mathrm{ACC_{TEST}}$ IS THE TEST SET ACCURACY AND $\|w'\|_2$ IS THE 2-NORM OF THE WEIGHT VECTOR.

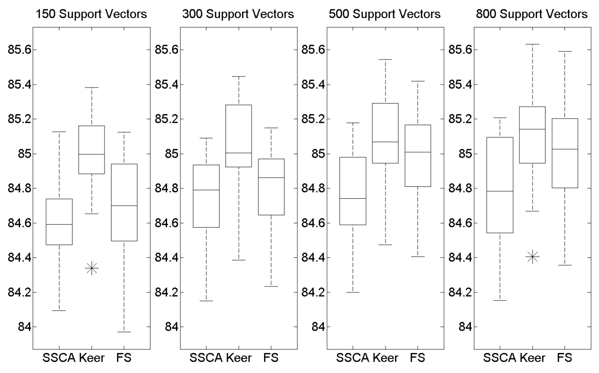
D	SV	$\mathrm{ACC_{TR}}$(%)	$\mathrm{ACC_{TEST}}$(%)	$\|w'\|_2$
non-pruned	78	89.6	89.7	25
0	7	90.0	89.0	236
0.3	9	89.6	89.9	47
0.9	8	88.8	90.1	21
1.3	5	87.2	90.2	11

TABLE III
NSCA WITH REMOVAL OF MISCLASSIFIED EXAMPLES APPLIED TO THE RIPLEY DATASET. C' IS A PARAMETER OF THE NSCA ALGORITHM, SV IS THE NUMBER OF SUPPORT VECTORS, $\mathrm{ACC_{TR}}$ IS THE TRAINING SET ACCURACY, $\mathrm{ACC_{TEST}}$ IS THE TEST SET ACCURACY AND $\|w'\|_2$ IS THE 2-NORM OF THE WEIGHT VECTOR.

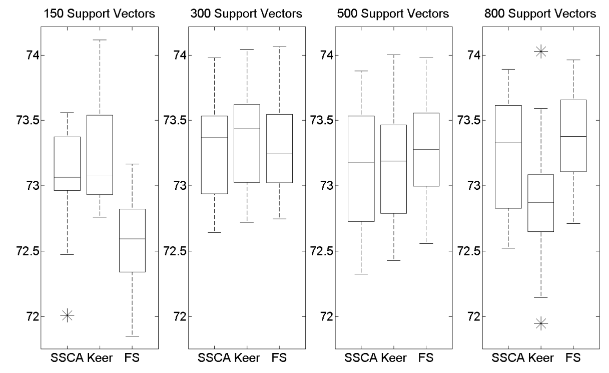
C'	SV	$\mathrm{ACC_{TR}}$(%)	$\mathrm{ACC_{TEST}}$(%)	$\|w'\|_2$
non-pruned	78	89.6	89.7	25
∞	7	90.0	89.0	236
1000	10	89.6	89.6	67
100	22	88.8	89.4	32
10	43	87.2	89.8	14

TABLE IV
PROPERTIES OF THE BENCHMARK DATASETS. $\mathrm{N_{TEST}}$ STANDS FOR THE NUMBER OF TEST DATA POINTS, $\mathrm{N_{TR}}$ FOR NUMBER OF THE TRAINING DATA POINTS, $\mathrm{N_{INP}}$ FOR THE NUMBER OF INPUT DIMENSIONS AND $\mathrm{N_{CLA}}$ FOR THE NUMBER OF CLASSES.

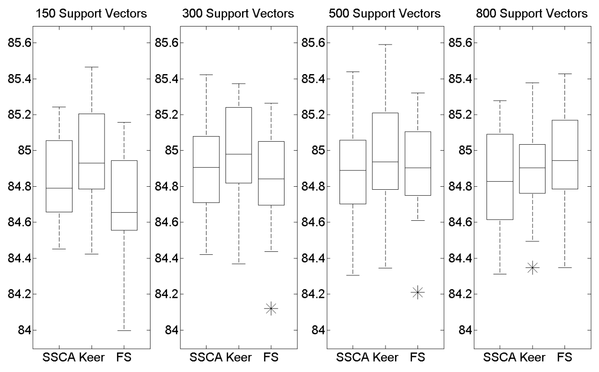
	$\mathrm{N_{test}}$	$\mathrm{N_{tr}}$	$\mathrm{n_{inp}}$	$\mathrm{n_{cla}}$
adu	12222	30000	123	2
pro	5922	11844	357	3



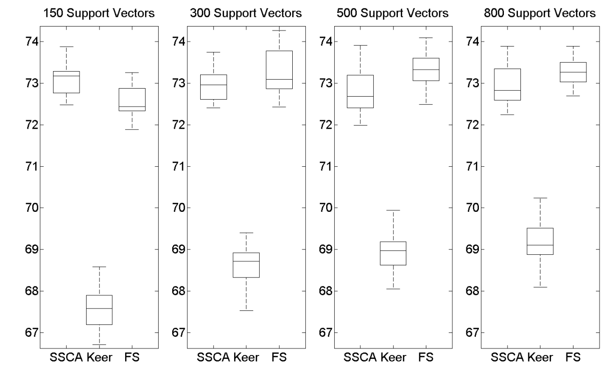
(a) *adu(rbf)*; Randomizations: 20; Hyperparameters:$\sigma_{\text{svm}} = 8$, $C_{\text{svm}} = 5$, $\sigma_{\text{kee}} = 8$, $C_{\text{kee}} = 3$; $\text{ACC}_{\text{TEST,npr}}$ = 84.71% ; SV_{npr} = 11604.



(b) *pro(rbf)*; Randomizations: 20; Hyperparameters: $\sigma_{\text{svm}} = 7$, $C_{\text{svm}} = 20$, $\sigma_{\text{kee}} = 20$, $C_{\text{kee}} = 200$; $\text{ACC}_{\text{TEST,npr}}$ = 74.42%; SV_{npr} = 8116.



(c) *adu(pol)*; Randomizations: 20; Hyperparameters: $d_{\text{svm}} = 3$, $\tau_{\text{svm}} = 0.1$, $C_{\text{svm}} = 500$,$d_{\text{kee}} = 2$, $\tau_{\text{kee}} = 0.9$, $C_{\text{kee}} = 0.01$; $\text{ACC}_{\text{TEST,npr}}$ = 84.75%; SV_{npr} = 11583.



(d) *pro(pol)*; Randomizations: 20; Hyperparameters: $d_{\text{svm}} = 3$, $\tau_{\text{svm}} = 1.1$, $C_{\text{svm}} = 500$,$d_{\text{kee}} = 3$, $\tau_{\text{kee}} = 0.9$, $C_{\text{kee}} = 2000$; $\text{ACC}_{\text{TEST,npr}}$ = 74.37%; SV_{npr} = 8002.

Fig. 3. SSCA, Keerthi's method (Keer) and Fixed-Size LS-SVM (FS) reduce the number of support vectors of SVM classifiers using a radial basis function (rbf) kernel ($K(x, x_k) = \exp(-\|x - x_k\|_2^2 / \sigma^2)$) or polynomial (pol) kernel ($K(x, x_k) = (x^T x_k + \tau)^d$ with $\tau \geq 0$). The bandwidth of the kernel used for the selection of the basis functions in Fixed-Size LS-SVM is chosen according to the Improved Silverman's Normal Rule of Thumb (ISNR). For each randomization the datapoints points that belong to the training set are randomly selected. The remaining datapoints belong to the test set. $\text{ACC}_{\text{TEST,npr}}$ is the average test set accuracy of the non-pruned SVM solution, SV_{npr} is the average number of support vectors of the non-pruned SVM solution. In the SSCA method, we always remove misclassified datapoints instead of flipping their labels. The hyperparameters are optimized doing 10-fold cross-validation. For fixed-size LS-SVM, the hyperparameters depend on the number of basis functions and differ from the hyperparameters of the non-pruned SVM solution. Hyperparameters of the SVM solution have subscript 'svm'. Hyperparameters of Keerthi's method have subscript 'kee'. Given a certain dataset, kernel and number of support vectors, the figures show one boxplot for each method: the central mark is the median, the edges of the box are the 25'th and 75'th percentiles, the whiskers extend to the most extreme datapoints not considered as outliers, and the outliers are plotted individually.

TABLE V

COMPARISON OF SSCA WITH REMOVAL OF MISSCLASSIFIED EXAMPLES AND SSCA WITH FLIPPING OF LABELS OF MISSCLASSIFIED EXAMPLES WHEN USING A RADIAL BASIS FUNCTION (RBF) KERNEL. D IS A PARAMETER OF THE SSCA ALGORITHM, REM(*adu*) IS THE TEST SET ACCURACY AND, IN PARENTHESES, THE NUMBER OF SUPPORT VECTORS OF SSCA APPLIED ON *adu* WHEN REMOVING MISSCLASSIFIED EXAMPLES. FLIP(*adu*), REM(*pro*) AND FLIP(*pro*) ARE DEFINED SIMILARLY. THE NON-PRUNED SOLUTION IS DENOTED AS 'NPR'.

D	rem(*adu*)	flip(*adu*)	rem(*pro*)	flip(*pro*)
npr	84.7	84.7	74.2	74.2
0	84.4(577)	84.4(771)	74.1(3678)	73.8(3782)
0.5	84.4(254)	84.4(271)	73.7(1740)	73.7(1741)
1	84.2(136)	84.2(140)	73.7(681)	73.7(682)
1.5	84.3(74)	84.3(76)	73.6(384)	73.6(387)
2	84.0(58)	84.0(58)	73.0(272)	73.1(273)

TABLE VI

TIME REQUIRED TO FIND THE SOLUTION CONTAINING 300 SUPPORT VECTORS FOR *adu(rbf)* AND 500 SUPPORT VECTORS FOR *pro(pol)*. THE COMPARED ALGORITHMS ARE SSCA, KEERTHI'S METHOD AND FIXED-SIZE LS-SVM (FS LS-SVM). IN CASE OF SSCA, THE TIME REQUIRED TO FIND THE PRUNED SOLUTION GIVEN THE NON-PRUNED SOLUTION IS IN PARENTHESES AND THE TIME REQUIRED TO FIND THE NON-PRUNED SOLUTION IS BEFORE THE PARENTHESES. IN CASE OF FS LS-SVM THE TIME REQUIRED TO FIND THE BASIS FUNCTIONS IS IN PARENTHESES AND THE TIME REQUIRED TO FIND THE SOLUTION, GIVEN THE BASIS FUNCTIONS, IS BEFORE THE PARENTHESES.

	adu(seconds)	*pro*(seconds)
non-pruned SVM	88.3	1112
SSCA	88.3(5.8)	1112(3.1)
Keerthi's method	23	78.5
FS LS-SVM (ISNR)	0.9(397)	1.4(483)