three scenarios: $y(0) = 0$, $y(1) = 1$ and $y(0) = a$. For $y(0) = 0$, we can obtain the ordering of $n - k$ smallest inputs. For $y(0) = 1$, we can obtain the ordering of k largest inputs. If the distribution of the inputs is given, we are able to estimate the average and the confidence level of the convergence time by Monte Carlo simulations. The advantage of our approach is that time-consuming simulations on the network dynamics are avoided. Our result provides us a guideline for the design of sampling duration (clock design) and the initial condition of DNN-kWTA networks.

For the case where the inputs are uniformly distributed, we have derived the mean and variance of the convergence time. Since nonuniformly distributed inputs can be converted into uniformly distributed inputs, we can say that the time behavior of DNN-kWTA networks for the uniform distribution is the same as that for other distributions.

ACKNOWLEDGMENT

The authors would like to thank J. Wang for sharing his unpublished manuscripts.

REFERENCES

[1] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(N) complexity," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 703–711.

[2] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

[3] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. Unit. States Amer.*, vol. 81, no. 10, pp. 3088–3092, May 1984.

[4] L. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.

[5] E. Majani, R. Erlanson, and Y. Abu-Mostafa, "On the k-winner-take-all network," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 634–642.

[6] G. Dempsey and E. McVey, "Circuit implementation of a peak detector neural network," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 40, no. 9, pp. 585–591, Sep. 1993.

[7] G. Seiler and J. Nossek, "Winner-take-all cellular neural networks," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 40, no. 3, pp. 184–190, Mar. 1993.

[8] L. Andrew, "Improving the robustness of winner-take-all cellular neural networks," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 43, no. 4, pp. 329–334, Apr. 1996.

[9] T. M. Kown and M. Zervakis, "k-WTA networks and their applications," *Multidimen. Syst. Signal Process.*, vol. 6, pp. 333–346, Oct. 1995.

[10] J. Narkiewicz and W. Burleson, "Rank-order filtering algorithms: A comparison of VLSI implementations," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3. Chicago, IL, May 1993, pp. 1941–1944.

[11] C. Marinov, B. Calvert, R. Costea, and V. Bucata, "Time evaluation for analog kWTA processors," in *Proc. 4th Eur. Congr. Comput. Methods Appl. Sci. Eng.*, Jul. 2004, pp. 1–7.

[12] J. Sum, C.-S. Leung, P. Tam, G. Young, W. Kan, and L.-W. Chan, "Analysis for a class of winner-take-all model," *IEEE Trans. Neural Netw.*, vol. 10, no. 1, pp. 64–71, Jan. 1999.

[13] X. Hu and J. Wang, "An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2022–2031, Dec. 2008.

[14] J. Wang and Z. Guo, "Parameric sensitivity and scalability of k-winners-take-all networks with a single state variable and infinity-gain activation functions," in *Advances in Neural Networks* (Lecture Notes in Computer Science), L. Zhang, B. L. Lu, and J. Kwok, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 77–85.

[15] J. Wang, "Analysis and design of a k-winners-take-all model with a single state variable and the heaviside step activation function," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1496–1506, Sep. 2010.

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

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

***Abstract*— In this brief, 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 the 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 the smoothed SCA (SSCA), that additionally upper-bounds the weight vector of the pruned solution and, for the 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*— Run time complexity, sparse approximation, support vector machine classifier.**

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] and [6], can be considerably higher because of the large number of support vectors.

Several methods have been proposed for pruning the solutions of SVMs. In [5] and [7], an algorithm is proposed that

Manuscript received March 26, 2010; revised January 25, 2012; accepted January 25, 2012. Date of publication February 21, 2012; date of current version March 6, 2012. This work was supported in part by the 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 Ph.D./Post-Doctoral, and Fellow Grants; Flemish Government: FWO: Ph.D./Post-Doctoral 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: Ph.D. 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 (259166) Contract Research: AMINAL Other: Helmholtz: viCERP ACCM.

The authors are with the Department of Electrical Engineering (ESAT-SCD-SISTA), Katholieke Universiteit Leuven, Leuven B-3001, Belgium (e-mail: dries.geebelen@esat.kuleuven.be; johan.suykens@esat.kuleuven.be; joos.vandewalle@esat.kuleuven.be).

Digital Object Identifier 10.1109/TNNLS.2012.2186314

2162–237X/$31.00 © 2012 IEEE

approximates the weight vector such that the distance to the original weight vector is minimized. Downs *et al*. [8] presents 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] presents an effective greedy method SVM that uses a basis selection criterion which is directly related to the training cost. In fixed-size (FS) least square (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 brief, an algorithm, called the 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 the smoothed SCA (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 a polynomial (Pol) kernel.

The remainder of this brief is organized as follows. In Section II, the separable and nonseparable cases are briefly reviewed. Section III explains how approximating an SVM classifier can be formulated as a classification problem, and presents two algorithms, namely, 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. SVM 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:

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

where C is a positive real constant. The hyperparameter C determines the tradeoff between the margin and the training error. In the nonseparable 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 are not tolerated. The dual, after applying the kernel trick, becomes

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

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 RBF kernel ($K(x, x_k) = \exp(-\|x - x_k\|_2^2 / \sigma^2)$) and the Pol kernel ($K(x, x_k) = (x^T x_k + \tau)^d$ with $\tau \geq 0$). 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 nonseparable case, the support vectors are characterized by

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

This means the support vectors are misclassified examples ($y_k[w^T \varphi(x_k) + b] \leq 0$) or examples close to the decision boundary in the feature space ($0 < y_k[w^T \varphi(x_k) + b] \leq 1$). 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. SCA

A. Approximating the Decision Boundary

This section 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 data point according to the original and the approximating classifier, respectively. The first approximation error measure we propose equals the probability that a data point is classified differently by $\hat{y}(x)$ and $\hat{y}'(x)$

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

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

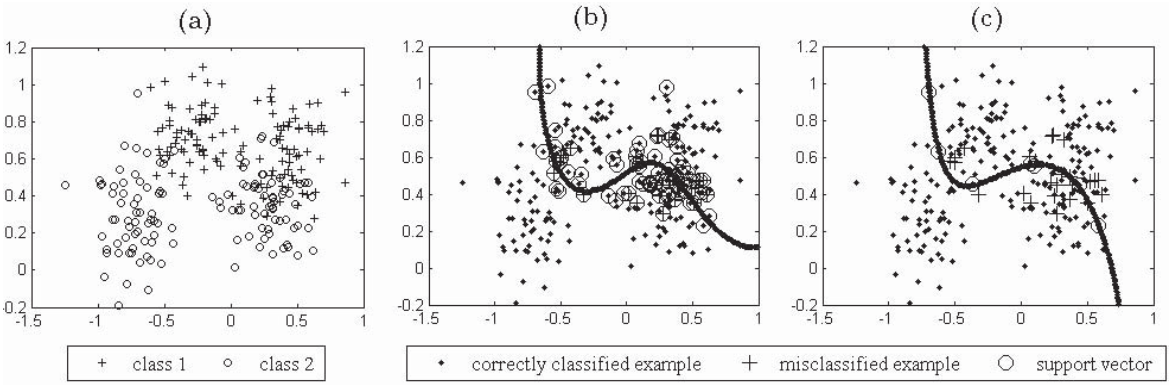


Fig. 1. Ripley dataset containing 250 training data points and 1000 test data points. (a) Training examples of the Ripley dataset with corresponding labels. (b) Nonpruned solution using Vapnik's SVM classifier for the nonseparable case with $\sigma = 1$ and $C = 100$. (c) Pruned solution using SCA with removal of misclassified examples.

Algorithm 1 SCA

1) Find the nonpruned solution using Vapnik's SVM for the nonseparable case.
2) Modify the training set such that it becomes separable by applying rules a) and b) to misclassified examples $(y_k[w^T\varphi(x_k) + b] \leq 0)$.
   a) Flip their labels.
   b) Remove them.
3) Given the modified training set, find the SVM solution for the separable case.

modified training set. If it is only important to classify correctly classified data points similarly, then the approximation error becomes

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

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 nonpruned solution separates it. This separability holds only 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. SCA

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

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 nongeneric 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 cross-training 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].

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

	SV	$\text{ACC}_{\text{TR}}(\%)$	$\text{ACC}_{\text{TEST}}(\%)$
Nonpruned	78	89.6	89.7
Remove	7	90.0	89.0
Flip	8	89.6	89.0

The SCA 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.

1) 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 sequential minimal optimization algorithm needs to find the solution.
2) We want to avoid overfitting (in the sense that training data points can be classified similarly by the pruned and nonpruned classifiers while the unseen data points are classified differently) by imposing smoothness.
3) Ideally, upper-bounding $\|w'\|_2$ decreases the number of support vectors.

To conclude this section we apply SCA to the Ripley dataset using an SVM classifier with an RBF kernel. The resulting classifier is visualized in Fig. 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. Fig. 1 shows that the support vectors of the pruned solution lie close to the decision boundary of the nonpruned solution.

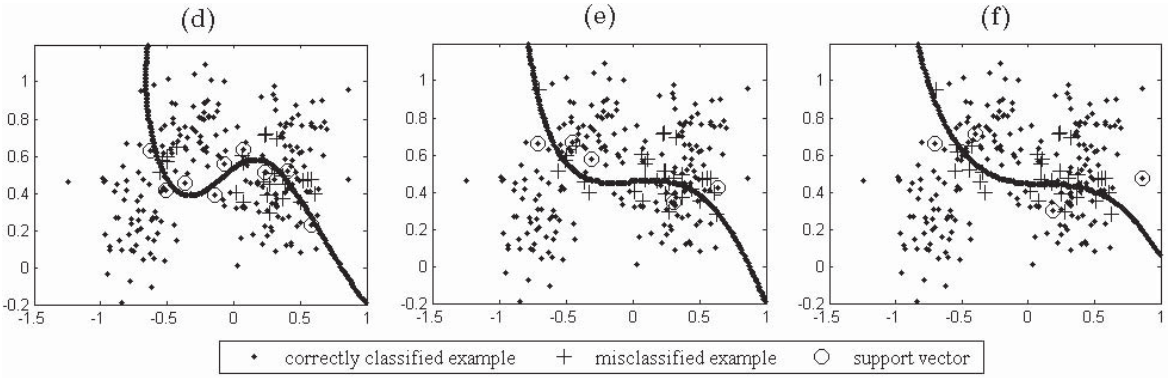


Fig. 2. SSCA with removal of misclassified examples applied to the Ripley dataset with $\sigma = 1$ and $C = 100$. Parameter D differs per figure. (d) $D = 0.3$. (e) $D = 1.3$. (f) $D = 2.0$.

Algorithm 2 SSCA

1) Find the nonpruned solution using Vapnik's SVM for the nonseparable case.
2) Modify the training set such that it becomes separable by applying rules a) and b) to misclassified examples ($y_k[w^T\varphi(x_k) + b] \leq 0$).
   a) Flip their labels.
   b) 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.

C. SSCA

The SSCA algorithm upper bounds $\|w'\|_2$ as shown in Algorithm 2.

SSCA makes a tradeoff 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:

1) all training examples at a distance higher than $D/\|w'\|_2$ from the separating hyperplane in the feature space are classified similarly by $\hat{y}$ and $\hat{y}'$;
2) $\|w'\|_2$ is upper bounded by $\|w\|_2/D$ 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', which is 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 an RBF or 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.

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, ACC_{TR} IS THE TRAINING SET ACCURACY, ACC_{TEST} IS THE TEST SET ACCURACY, AND $\|w'\|_2$ IS THE 2-NORM OF THE WEIGHT VECTOR

D	SV	ACC_{TR}(%)	ACC_{TEST}(%)	$\|w'\|_2$
Nonpruned	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

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

D. Nonseparable Case Approximation (NSCA)

The most straightforward way of upper-bounding $\|w'\|_2$ is using the SVM for the nonseparable case to solve the approximation problem. This implies an additional hyperparameter (C'). The results of applying 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, which 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

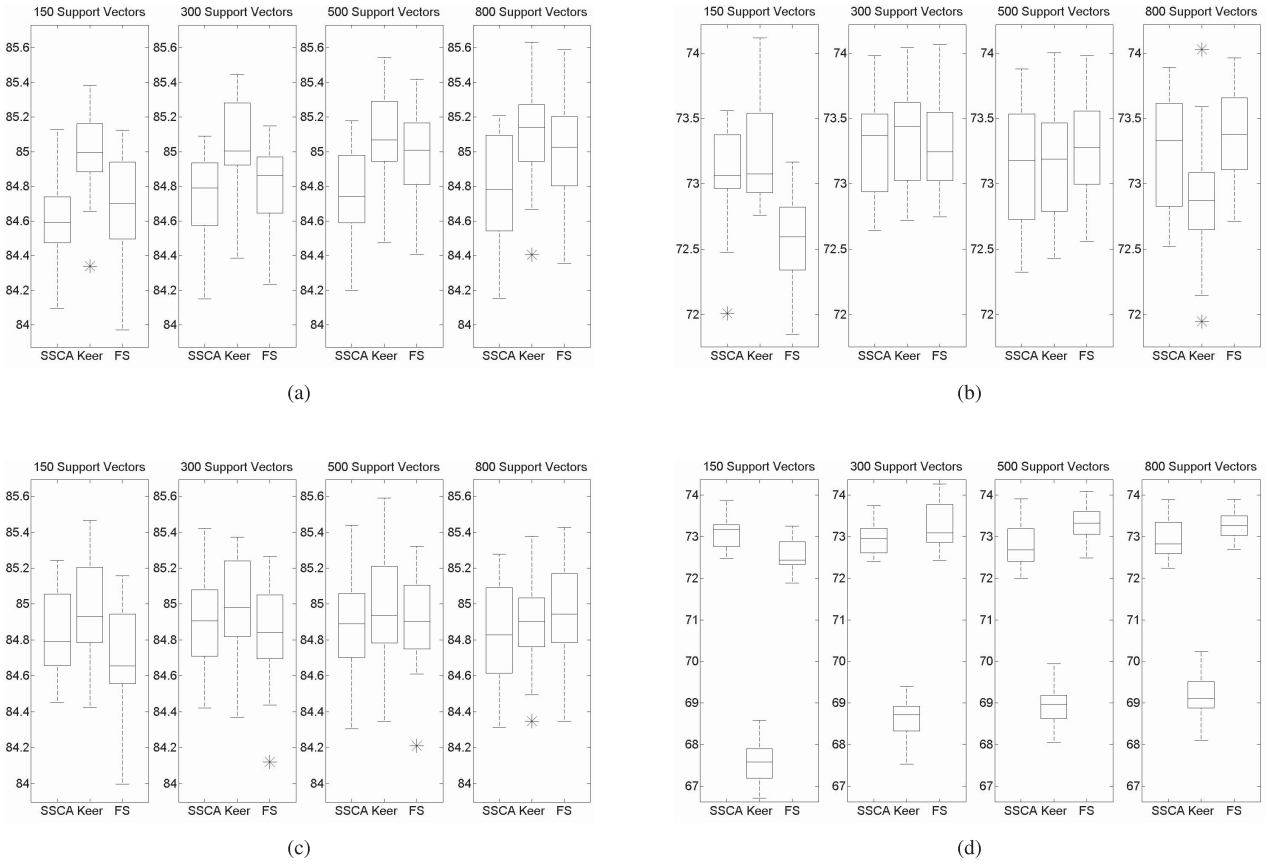


Fig. 3. SSCA, Keerthi's method (Keer), and FS LS-SVM reducing the number of support vectors of SVM classifiers using a RBF kernel ($K(x, x_k) = \exp(-\|x - x_k\|_2^2/\sigma^2)$) or Pol kernel ($K(x, x_k) = (x^T x_k + \tau)^d$ with $\tau \geq 0$). The bandwidth of the kernel used for the selection of the basis functions in FS LS-SVM is chosen according to the improved Silverman's normal rule of thumb (ISNR). For each randomization, the data points that belong to the training set are randomly selected. The remaining data points belong to the test set. $\text{ACC}_{\text{TEST,npr}}$ is the average test set accuracy of the nonpruned SVM solution, and SV_{npr} is the average number of support vectors of the nonpruned SVM solution. In the SSCA method, we always remove misclassified data points instead of flipping their labels. The hyperparameters are optimized using 10-fold cross-validation. For FS LS-SVM, the hyperparameters depend on the number of basis functions and differ from the hyperparameters of the nonpruned 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 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered as outliers, and the outliers are plotted individually. (a) *adu(rbf)*; randomizations: 20; hyperparameters: $\sigma_{\text{svm}} = 8$, $C_{\text{svm}} = 5$, $\sigma_{\text{kee}} = 8$, $C_{\text{kee}} = 3$; $\text{ACC}_{\text{TEST,npr}} = 84.71\%$; and $\text{SV}_{\text{npr}} = 11\,604$. (b) *pro(rbf)*; randomizations: 20; hyperparameters: $\sigma_{\text{svm}} = 7$, $C_{\text{svm}} = 20$, $\sigma_{\text{kee}} = 20$, $C_{\text{kee}} = 200$; $\text{ACC}_{\text{TEST,npr}} = 74.42\%$; and $\text{SV}_{\text{npr}} = 8116$. (c) *adu(pol)*; randomizations: 20; hyperparameters: $d_{\text{svm}} = 3$, $\tau_{\text{svm}} = 0.1$, $C_{\text{svm}} = 500$, $d_{\text{kee}} = 2$, $\tau_{\text{kee}} = 0.9$, $C_{\text{kee}} = 0.01$; $\text{ACC}_{\text{TEST,npr}} = 84.75\%$; and $\text{SV}_{\text{npr}} = 11\,583$. (d) *pro(pol)*; randomizations: 20; hyperparameters: $d_{\text{svm}} = 3$, $\tau_{\text{svm}} = 1.1$, $C_{\text{svm}} = 500$, $d_{\text{kee}} = 3$, $\tau_{\text{kee}} = 0.9$, $C_{\text{kee}} = 2000$; $\text{ACC}_{\text{TEST,npr}} = 74.37\%$; and $\text{SV}_{\text{npr}} = 8002$.

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, ACC_{TR} IS THE TRAINING SET ACCURACY, ACC_{TEST} IS THE TEST SET ACCURACY, AND $\|w'\|_2$ IS THE 2-NORM OF THE WEIGHT VECTOR

C'	SV	$\text{ACC}_{\text{TR}}(\%)$	$\text{ACC}_{\text{TEST}}(\%)$	$\|w'\|_2$
Nonpruned	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. N_{TEST} STANDS FOR THE NUMBER OF TEST DATA POINTS, N_{TR} FOR NUMBER OF THE TRAINING DATA POINTS, n_{INP} FOR THE NUMBER OF INPUT DIMENSIONS, AND n_{CLA} FOR THE NUMBER OF CLASSES

	N_{TEST}	N_{TR}	n_{INP}	n_{CLA}
adu	12222	30000	123	2
pro	5922	11844	357	3

subset. The decision functions of these SVMs are then used to discard two types of training examples: namely, those that are confidently recognized, and those that 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, such as Keerthi's method [9], incrementally choose basis functions according to a criterion that is directly related to the training cost function. In most

TABLE V
COMPARISON OF SSCA WITH REMOVAL OF MISCLASSIFIED EXAMPLES AND SSCA WITH FLIPPING OF LABELS OF MISCLASSIFIED EXAMPLES WHEN USING A 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 MISCLASSIFIED EXAMPLES. FLIP(*adu*), REM(*pro*), AND FLIP(*pro*) ARE DEFINED SIMILARLY. THE NONPRUNED 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 (FS LS-SVM). IN CASE OF SSCA, THE TIME REQUIRED TO FIND THE PRUNED SOLUTION GIVEN THE NONPRUNED SOLUTION IS IN PARENTHESES AND THE TIME REQUIRED TO FIND THE NONPRUNED 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)
Nonpruned 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)

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. FS LS-SVM

In FS 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. FS 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 ISNR [13].

V. EXPERIMENTAL RESULTS

We will compare SSCA with Keerthi's method [9] and FS LS-SVM [10]. The datasets used are Adult (*adu*) and Protein (*pro*). The dataset *adu* has been obtained from the UCI benchmark repository [14], and *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 three 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 nonpruned and 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 does not cause a significantly better tradeoff 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. Fig. 3 shows that SSCA drastically reduces the number of support vectors. On *adu*, Keerthi's method has mostly the best tradeoff between test set accuracy and the number of support vectors. On *pro(pol)*, SSCA has the better tradeoff 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 FS 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 FS 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 brief, we succeeded in drastically reducing the number of support vectors of SVM classifiers using the SSCA. Instead of approximating the underlying weight vector, we directly approximate the way the data points are classified. Compared to the SCA, SSCA has the following benefits: 1) it ensures numerical stability; 2) reduces the time to find the pruned solution; 3) avoids overfitting during the approximation phase; and 4) 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 nonpruned solution.

ACKNOWLEDGMENT

The authors would like 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," in *Proc. 17th Neural Inf. Process. Syst. Conf.*, vol. 17. 2005, pp. 81–88.

[2] P. Devijver and J. Kittler, *Pattern Recogniton: A Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[3] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[4] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

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

[6] E. Osuna and F. Girosi, "Reducing the run-time complexity of support vector machines," in *Proc. Int. Conf. Pattern Recognit.*, Brisbane, Australia, Aug. 1998, pp. 1–10.

[7] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. Müller, 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, Sep. 1999.

[8] T. Downs, K. E. Gates, and A. Masters, "Exact simplification of support vector solutions," *J. Mach. Learn. Res.*, vol. 2, pp. 293–297, Dec. 2001.

[9] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *J. Mach. Learn. Res.*, vol. 7, pp. 1493–1515, Jul. 2006.

[10] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 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," *Comput. Stat. Data Anal.*, 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. Royal Soc. London*, vol. 209, nos. 441–458, pp. 415–446, 1909.

[13] N. L. Hjort and D. Pollard, "Better rules of thumb for choosing bandwith in density estimation," Dept. Math., Univ. Oslo, Oslo, Norway, Res. Rep., 1996.

[14] C. L. Blake and C. J. Merz. (1998). *UCI Repository of Machine Learning Databases*. Dept. Inf. Comput. Sci., Univ. California, Irvine [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[15] C.-C. Chang and C.-J. Lin. (2001). *LIBSVM: A Library for Support Vector Machines* [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm

[16] J.-Y. Wang, "Application of support vector machines in bioinformatics," M.S. thesis, Dept. Comput. Sci. Inf. Eng., National Taiwan University, Taipei, Taiwan, 2002.