COPYRIGHT NOTICE

This is the author's version of the work. The definitive version was published in *IEEE Transactions on Neural Networks* (TNN), 21(2):351-357, February 2010.

DOI: `http://dx.doi.org/10.1109/TNN.2009.2039227`.

[4] W. Y. Wang, M. L. Chan, T. T. Lee, and C. H. Liu, "Adaptive fuzzy control for strict-feedback canonical nonlinear systems with H_∞ tracking performance," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 30, no. 6, pp. 878–885, Dec. 2000.
[5] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable Adaptive Neural Network Control*. Boston, MA: Kluwer, 2001.
[6] C. F. Hsu, C. M. Lin, and T. T. Lee, "Wavelet adaptive backstepping control for a class of nonlinear systems," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1175–1183, Sep. 2006.
[7] D. Wang and J. Huang, "Neural network-based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict-feedback form," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 195–202, Jan. 2005.
[8] H. Du, H. Shao, and P. Yao, "Adaptive neural network control for a class of low-triangular-structured nonlinear systems," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 509–514, Mar. 2006.
[9] S. S. Ge, F. Hong, and T. H. Lee, "Adaptive neural control of nonlinear time-delay systems with unknown virtual control coefficients," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 34, no. 1, pp. 499–516, Feb. 2004.
[10] W. Chen, "Adaptive NN control for discrete-time pure-feedback systems with unknown control direction under amplitude and rate actuator constrains," *ISA Trans.*, vol. 48, no. 3, pp. 304–311, Jul. 2009.
[11] W. Chen and J. Li, "Decentralized output-feedback neural control for systems with unknown interconnections," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 38, no. 1, pp. 258–266, Feb. 2008.
[12] S. Hara, Y. Yamamoto, T. Omata, and M. Nakano, "Repetitive control system: A new type servo system for peiodic exogenous signals," *IEEE Trans. Autom. Control*, vol. AC-33, no. 7, pp. 258–266, Jul. 1988.
[13] Y. P. Tian and X. Yu, "Robust learning control for a class of nonlinear systems with periodic and aperiodic uncertainties," *Automatica*, vol. 39, no. 11, pp. 1957–1966, Nov. 2003.
[14] J. X. Xu, "A new periodic adaptive control approach for time-varying parameters with known periodicity," *IEEE Trans. Autom. Control*, vol. 49, no. 4, pp. 579–583, Apr. 2004.
[15] Z. Ding, "Adaptive estimation and rejection of unknown sinusoidal disturbances in a class of non-minimum phases nonlinear systems," *Inst. Electr. Eng. Proc.—Control Theory Appl.*, vol. 153, no. 4, pp. 379–386, Aug. 2006.
[16] M. Tomizuka, "Dealing with periodic disturbances in controls of mechanical systems," *Annu. Rev. Control*, vol. 32, no. 2, pp. 193–199, Dec. 2008.
[17] W. E. Dixon, E. Zergeroglu, D. M. Dawson, and B. T. Costic, "Repetitive learning control: A Lyapunov-based approach," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 32, no. 4, pp. 538–545, Aug. 2002.
[18] S. Liuzzo, R. Marino, and P. Tomei, "Adaptive learning control of nonlinear systems by output error feedback," *IEEE Trans. Autom. Control*, vol. 52, no. 7, pp. 1232–1248, Jul. 2007.
[19] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–398, Mar. 1996.
[20] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. London, U.K.: Tayor & Francis, 1999.

Scaling Up Support Vector Machines Using Nearest Neighbor Condensation

Fabrizio Angiulli and Annabella Astorino

***Abstract*—In this brief, we describe the FCNN-SVM classifier, which combines the support vector machine (SVM) approach and the fast nearest neighbor condensation classification rule (FCNN) in order to make SVMs practical on large collections of data. As a main contribution, it is experimentally shown that, on very large and multidimensional data sets, the FCNN-SVM is one or two orders of magnitude faster than SVM, and that the number of support vectors (SVs) is more than halved with respect to SVM. Thus, a drastic reduction of both training and testing time is achieved by using the FCNN-SVM. This result is obtained at the expense of a little loss of accuracy. The FCNN-SVM is proposed as a viable alternative to the standard SVM in applications where a fast response time is a fundamental requirement.**

***Index Terms*—Classification, large data sets, training-set condensation, nearest neighbor rule, support vector machines (SVMs).**

I. Introduction

The objective of pattern classification is to find a rule, based on external observation, to assign an object to exactly one among several classes. Many algorithms have been indeed devised for automatically distinguishing among different samples on the basis of their patterns. A well-known supervised classification technique is the support vector machine (SVM) algorithm (see [5], [15], [16], and [18]). For many applications, the SVM method, alone or in combination with other methods, yields superior performance with respect to other machine learning options. In general, SVMs work very well in practice, have a small number of tunable parameters in comparison with neural networks, and tend towards global solutions.

Training an SVM is equivalent to solving a convex quadratic programming (QP) problem characterized by a dense, positive–semidefinite matrix with a number of rows equal to the number of training data points. Thus, it is a real challenge to deal with practical applications where the data set is made up of thousands of points. In fact, just computing the matrix for the QP problem is expensive and one may not be able to store it. Moreover, the computing time for solving the QP problem is $O(N^3)$, where N denotes the number of training examples, while the space complexity is $O(N^2)$; these may become prohibitive on large training sets. More in general, applying SVMs on large training sets requires a drastic increase of memory, training time, and prediction time. Indeed, prediction time is proportional to the number of support vectors (SVs), which still increases with the number of data points.

Many algorithms and implementation techniques have been developed for efficiently training SVMs for massive data sets. Among them we mention decomposition techniques and data reduction techniques. Decomposition techniques speed up the SVM training by dividing the original QP problem into smaller pieces, thereby reducing the size of

Manuscript received August 11, 2009; revised November 16, 2009; accepted December 09, 2009. First published January 12, 2010; current version published February 05, 2010.

F. Angiulli is with the Department of Electronic, Computer Science and Systems Engineering, University of Calabria, Rende (CS) 87036, Italy (e-mail: f.angiulli@deis.unical.it).

A. Astorino is with the Institute of High Performance Networking and Computing of the National Research Council, Rende (CS) 87036, Italy (e-mail: astorino@icar.cnr.it).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2039227

each QP problem. They are among the first proposals to speedup SVMs. Well-known decomposition techniques are the "chunking" algorithm [3], Osuna's decomposition algorithm [13], and the sequential minimal optimization (SMO) [14]. A potential disadvantage of these techniques is that they may need a long training time because they usually require to execute many passages over the data set to reach a reasonable level of convergence. These techniques are at present natively supported by standard SVM tools, as LIBSVM [4].

Different data reduction techniques have been proposed for making the training data of manageable size. The principal idea of this approach is that a subsampling of the training set may speed up a classifier. This can be achieved by randomly removing training points or by selecting some significant samples and ignoring others that can be absorbed, or represented, by those selected. Among the data reduction approaches for SVM proposed in the literature there are the reduced support vector machine (RSVM) [11] and its related variants [8], [10], the cross-training algorithm [2], and the clustering-based SVM (CB-SVM) algorithm [20].

In this brief, we introduce a data reduction method for SVMs, called FCNN-SVM, with the goal of making them scalable to very large data sets. The FCNN-SVM couples the SVM algorithm with the fast nearest neighbor condensation algorithm [1]. Differently from other methods, the FCNN-SVM is based on selection criteria which are guided by the decision boundary rather than by sampling and clustering like techniques, and it can be applied to any kind of data and kernel function.

As a main contribution, it will be experimentally shown that the FCNN-SVM scales well on large and high-dimensional data sets and that it is one or two orders of magnitude faster than the SVM. Moreover, the FCNN-SVM noticeably reduces the size of the model, since, in most cases, the number of SVs is more than halved with respect to the SVM. Thus, a drastic reduction of both training and testing times is achieved by using the FCNN-SVM. As far as test accuracy is concerned, the FCNN-SVM exhibits a little loss of accuracy.

The rest of this work is organized as follows. Section II introduces the FCNN-SVM classifier. Section III experimentally compares the FCNN-SVM and the standard SVM. Finally, Section IV summarizes contributions and concludes the brief.

II. The FCNN-SVM Rule

The FCNN-SVM combines two well-known classification rules, namely, the SVM and the nearest neighbor condensation. As for the definition of SVM and for the nearest neighbor condensation rule the reader is referred to [5], [15], [16], and [18], and to [1], [6], [7], [9], [17], and [19], respectively.

The strong point of SVMs is that they permit to build a classifier which maximizes generalization performances. Nevertheless, their training is costly. The SVs are critical points near the boundary between the two classes, which determine an optimal separating hyperplane. It must be noted that removal of training points that are not SVs has no effect on the hyperplane selection. If one could approximate the right SVs before the training process, then both computing time and memory usage of the SVM would be reduced, and an optimal separating hyperplane could be found by training the SVM on a relatively smaller number of selected points.

From the point of view of the generalization properties, the nearest neighbor rule is not as good as SVM. Nonetheless, a training set which is a consistent subset for the nearest neighbor rule can be found quite efficiently by using the FCNN algorithm [1]. The FCNN method selects a subset of the overall data set having the robust property of correctly classifying the rest of the data set and being mostly composed by points close to the decision boundary. Hence, using the FCNN to reduce the amount of data on which the SVM has to be trained is expected to provide a method with a profitable tradeoff between execution time and prediction accuracy.

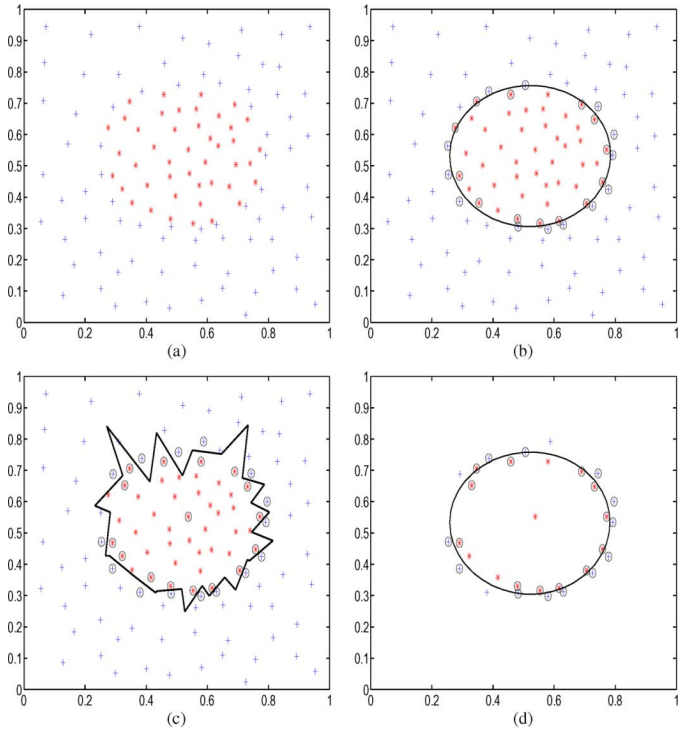

Fig. 1. Example of the FCNN-SVM training.

Given a training set T, the FCNN-SVM classifier is built in two steps.

1) Compute a training set consistent subset S of T for the nearest neighbor rule through the FCNN algorithm.
2) Train an SVM on the consistent subset S.

After selecting a subset of training points having the property of approximating the decision boundary (step 1), a separating hyperplane can be obtained by training an SVM on this subset (step 2).

Fig. 1 shows an example of FCNN-SVM training on a 2-D data set consisting of 121 points in the plane. Fig. 1(a) reports the original training set, composed by two well-separated classes. In Fig. 1(b), the SVs (circled points) are highlighted together with the decision boundary of an SVM trained with parameter $C = 500$ and using a radial basis function (RBF) kernel with $\gamma = 1$. Say SVM^{full} is the above defined classifier (using 26 SVs). Fig. 1(c) shows the FCNN subset (circled points) and the decision boundary of the FCNN rule trained on the original data set. The FCNN subset is composed of 31 points. Note that these points are very close to the decision boundary and most of them coincide with the SVs of the SVM^{full} classifier. Finally, Fig. 1(d) shows the SVs (the 24 circled points) and the decision boundary of the SVM trained on the FCNN subset with parameter $C = 500$ and using an RBF kernel with $\gamma = 1$ that are the SVs and the decision boundary of the FCNN-SVM rule.

Note that while the decision boundary of the FCNN classifier is rather crisp, the decision boundary of the FCNN-SVM classifier is smooth and more accurate. The decision boundary of the FCNN-SVM classifier is practically identical to the decision boundary of the SVM classifier, but it has been obtained at a reduced computational cost, since the training has been accomplished on just the 31 points returned by the FCNN algorithm instead of the 121 points of the overall input training set. Moreover, the number of SVs of the FCNN-SVM classifier is slightly smaller than the number of SV of the standard SVM.

There are different variants of the basic FCNN algorithm. In the following, we make use of the FCNN2 variant (see [1]), which is the fastest one. As far as the temporal cost of the FCNN-SVM method is

TABLE I
EXPERIMENTS ON SMALL DATA SETS

BUPA Liver						
Kernel	Par.	C	SVM		FCNN-SVM	
			Acc.	SVs	Acc.	SVs
Linear		0.1	67.7	240.9	62.8	151.6
		1	72.4	205.4	72.4	139.3
		100	**72.7**	192.7	**73.3**	134.1
RBF	0.01	0.1	58.3	256.0	55.4	153.8
		1	58.3	256.3	55.4	153.9
		100	71.9	199.1	69.2	137.6
	0.1	0.1	58.3	256.5	55.4	153.8
		1	71.0	229.7	67.5	149.9
		100	72.4	178.2	70.1	130.4
	1	0.1	69.8	253.4	56.9	154.6
		1	71.2	198.1	71.8	142.4
		100	66.6	167.8	64.2	130.0
Poly	2	0.1	58.3	256.6	58.3	153.0
		1	67.4	245.0	62.4	153.1
		100	70.9	176.9	69.8	131.1
	3	0.1	58.3	257.4	58.3	153.0
		1	61.8	257.2	58.3	153.2
		100	72.7	188.4	69.2	137.6

Cleveland			
SVM		FCNN-SVM	
Acc.	SVs	Acc.	SVs
84.5	110.2	**84.5**	73.5
85.5	92.8	84.1	61.8
84.5	91.0	82.1	56.9
72.1	152.2	72.1	81.5
85.9	143.4	72.4	81.2
84.5	99.6	83.2	65.4
72.4	154.9	72.4	82.79
84.2	123.1	82.5	80.8
80.1	107.3	71.1	70.2
72.1	231.3	72.1	87.6
76.2	231.0	73.8	87.69
78.2	225.5	76.5	87.4
72.1	155.7	72.1	82.5
79.1	153.3	71.7	82.6
73.7	102.6	71.1	65.8
72.1	158.6	72.1	84.1
78.7	158.6	72.1	84.1
75.4	111.2	67.1	75.1

Pima			
SVM		FCNN-SVM	
Acc.	SVs	Acc.	SVs
77.2	434.5	74.5	298.0
76.7	372.8	76.0	270.4
76.4	359.8	76.4	263.1
65.1	483.0	65.1	298.2
66.4	483.8	65.1	298.9
77.2	367.5	76.7	265.6
65.2	484.2	65.1	298.7
77.3	412.3	76.2	290.0
76.6	347.5	75.9	259.8
75.1	475.3	66.7	300.6
75.6	387.6	76.4	282.0
71.2	347.6	68.1	263.5
65.1	484.9	65.1	299.5
77.7	462.9	70.5	299.5
76.7	349.2	75.5	258.0
65.1	485.1	65.1	300.3
69.1	485.3	65.1	300.6
77.4	366.6	**77.5**	267.7

Ionosphere						
Kernel	Par.	C	SVM		FCNN-SVM	
			Acc.	SVs	Acc.	SVs
Linear		0.1	87.4	122.9	36.6	44.7
		1	87.7	88.8	72.3	41.8
		100	86.6	62.0	76.6	32.3
RBF	0.01	0.1	64.3	227.9	35.7	44.8
		1	89.4	169.8	35.7	45.1
		100	93.1	66.4	84.3	37.7
	0.1	0.1	93.4	223.4	35.7	43.2
		1	93.7	109.4	64.3	43.2
		100	**94.3**	66.3	81.4	34.2
	1	0.1	64.3	240.7	35.7	62.3
		1	92.0	211.8	61.7	62.3
		100	93.1	208.2	70.0	62.3
Poly	2	0.01	64.3	228.1	35.7	48.8
		1	89.1	213.1	35.7	49.3
		100	91.1	83.2	**89.1**	47.0
	3	0.01	64.3	227.8	35.7	53.1
		1	64.6	228.3	35.7	54.0
		100	91.4	126.3	79.4	56.0

Letter			
SVM		FCNN-SVM	
Acc.	SVs	Acc.	SVs
99.1	713.2	95.9	85.6
99.1	507.2	97.3	83.4
99.1	460.7	98.4	75.6
95.9	1365.5	95.9	84.6
99.1	1158.8	95.9	84.2
99.3	445.8	98.8	84.1
99.1	1128.4	95.9	92.4
99.3	560.4	95.9	87.5
99.8	241.6	99.3	86.6
99.4	692.7	95.9	125.4
99.7	385.5	99.2	126.2
99.9	241.4	**99.7**	108.1
95.9	1365.6	95.9	86.4
99.2	872.2	95.9	86.1
99.5	334.3	99.3	86.8
95.9	1366.7	95.9	78.3
98.9	1161.6	95.9	85.9
99.5	377.0	95.9	87.3

Image			
SVM		FCNN-SVM	
Acc.	SVs	Acc.	SVs
96.0	277.8	94.7	139.0
96.3	244.7	95.0	117.2
96.1	210.6	94.7	109.5
53.3	1793.5	14.3	205.2
86.7	1336.6	19.0	203.4
94.6	444.5	92.5	178.8
86.7	1392.9	15.6	207.5
92.5	783.7	45.7	204.3
96.7	305.7	**95.0**	161.6
92.1	1021.4	16.0	208.3
95.6	560.8	74.8	202.3
96.9	406.7	91.2	181.0
96.4	192.9	94.0	121.9
96.5	182.4	92.0	116.8
96.0	174.7	91.6	112.1
95.0	187.1	92.3	122.8
95.4	184.3	93.3	121.3
96.1	183.8	92.9	125.3

TABLE II
FURTHER EXPERIMENTS ON SMALL DATA SETS

Data set	Kernel Type	Kernel Param.	C	SVM		FCNN-SVM	
				Acc.	SVs	Acc.	SVs
BUPA Liver	Linear	–	10	73.3	194.3	72.7	134.7
Cleveland	Linear	–	0.7	85.9	94.2	85.2	62.2
Pima	Linear	–	70	77.6	360.4	77.1	260.4
Ionosphere	RBF	0.08	9	95.1	75.6	92.0	45.7
Letter	RBF	1	10	99.9	252.8	99.7	113.5
Image	RBF	0.1	100	96.7	306.0	95.0	162.0
Average				88.1	213.9	87.0	129.8

concerned, let N be the number of examples in the input training set T, and let n be the size of the training set consistent subset S, then the FCNN algorithm has cost $O(Nn)$.[1] Since the cost of the SVM is cubic in the size of the input training set, the temporal cost of the FCNN-SVM is $O(Nn+n^3)$, which is at most quadratic in the input training set size and cubic in the consistent subset size. The value n depends on the characteristics of the data set, but usually it holds that $n \ll N$, with n corresponding to a few percent of the whole data set size N [1].

[1]The spatial cost of the FCNN algorithm is also $O(Nn)$.

III. EXPERIMENTAL RESULTS

In this section, several experiments involving the FCNN-SVM are described. The performance of the FCNN-SVM rule[2] has been compared with that of LIBSVM [4].

Experiments are organized as follows. We present first the experiments on small data sets, in order to compare data reduction and classification accuracy (Section III-A). Of course, the advantages in terms of temporal cost of the FCNN-SVM rule can be appreciated only when very large data sets have to be processed, which are accounted for in Section III-B. Finally, in Section III-C, we consider some families of synthetical data sets in order to point out both advantageous and critical scenarios for the FCNN-SVM method.

A. Small Data Sets

We applied both FCNN-SVM and SVM on some test problems drawn from the classification literature that are BUPA Liver, Cleveland Heart, Pima Indians, Ionosphere, Letter Recognition, and Image Segmentation, all from the University of California at Irvine (UCI) Machine Learning Repository [12].

[2]The Euclidean distance was employed by the FCNN as distance measure.

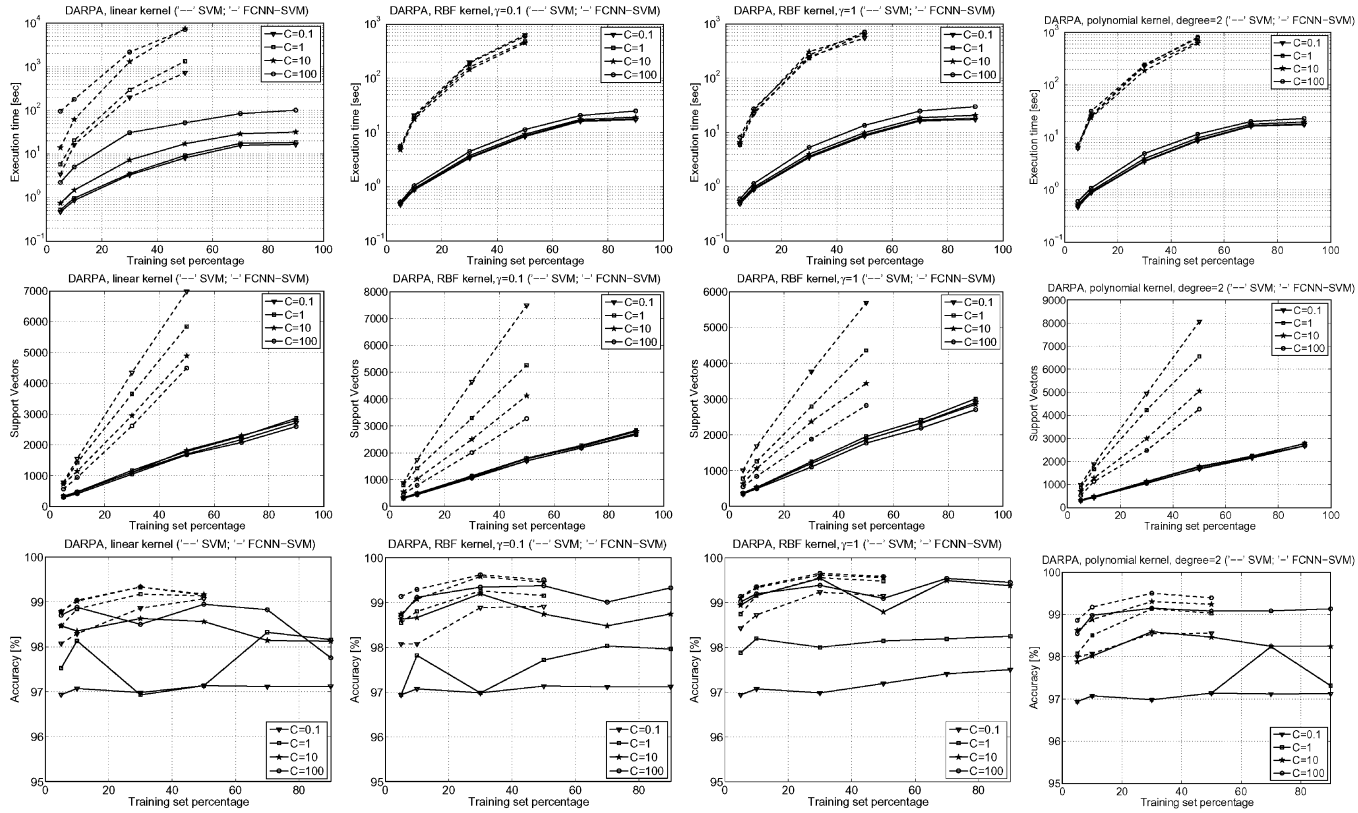

Fig. 2. Experiments on the DARPA 1998 data set.

To make the choice of the free parameters, we considered the following combinations of values on a finite grid:

Penalty C	0.1 1 10 100
Kernel Type	Linear RBF Poly
RBF-parameter	0.01 0.1 1
Poly-parameter	2 3

Classification accuracy (Acc.), measured through tenfold cross validation, and number of SVs are reported in Table I (values for $C = 10$ are not reported in the table due to space limitations). It can be observed that the optimal parameters for the SVM may in general be different from the optimal ones for the FCNN-SVM.

Moreover, for each data set, we have considered a thinner grid of parameter values in the intervals where the SVM method showed a regularly near-optimal behavior in terms of testing classification accuracy and number of SVs; Table II shows the results of these experiments.

As far as the classification accuracy is concerned, the FCNN-SVM exhibits a loss of accuracy with respect to the SVM, and this can be explained as a consequence of the great reduction of number of SVs. Table I shows that the FCNN-SVM is more sensitive to the selection of parameters, since its classification accuracy presents a greater variability with respect to that of the SVM. However, if we select the optimal values of accuracy, either the accuracies of the two methods are comparable or the FCNN-SVM presents a small loss.

B. Scaling Analysis on Very Large Data Sets

In this section, the scaling analysis of the FCNN-SVM and of the SVM on three very large data sets,[3] that are DARPA 1998, Forest Cover, and Checkerboard, is accomplished.

DARPA 1998 is obtained from the Defense Advanced Research Projects Agency 1998 intrusion detection evaluation data[4] and it is composed of 458 301 real vectors each having 23 features, one of which stating whether the vector is associated with a network attack. Forest Cover[5] consists of 581 012 instances each having 54 attributes. Instances are partitioned into seven classes. Checkerboard is a synthetically generated data set composed of 1 000 000 points into the unit square. A 4×4 checkerboard partitions the points into two classes associated with white and black cells of the board.

For each data set, we extracted six random samples of increasing size to form six training sets. Among the remaining objects, we also picked a random sample to form the associated test set.

In particular, on the DARPA 1998 and Forest Cover data sets we considered the same grid of parameters used for the small data sets (Figs. 2 and 3 report some of these experiments; the missing ones showed a very similar behavior). Since the SVM was very slow on these two data sets, it was not executed on the largest samples. On the Checkerboard, we considered only one single combination, which is the RBF kernel with $\gamma = 100$ and C set to 500, since by using these parameters, the SVM performs remarkably well in terms of classification accuracy.

Fig. 2 shows the results of the experiments on the DARPA 1998 data set. The first row of the figure reports the execution times. As an example, when the linear kernel with $C = 10$ was used on half of the data set, the FCNN-SVM terminated in about 17 s, while the SVM employed about 7500 s (a difference of more than two orders of magnitude). Likewise, for all the other kernels, the difference of execution time remained of about the same order. The second row reports the SVs. We observe that the reduction achieved by the FCNN-SVM is noticeable. In almost all cases the number of SVs of the FCNN-SVM is less

[3]Experiments were performed on an Intel Core 2 Duo (single core) 1.83-GHz-based machine with 1 GB of main memory and the Windows operating system.

[4]http://www.ll.mit.edu/IST/ideval/index.html

[5]http://kdd.ics.uci.edu/databases/covertype/covertype.html

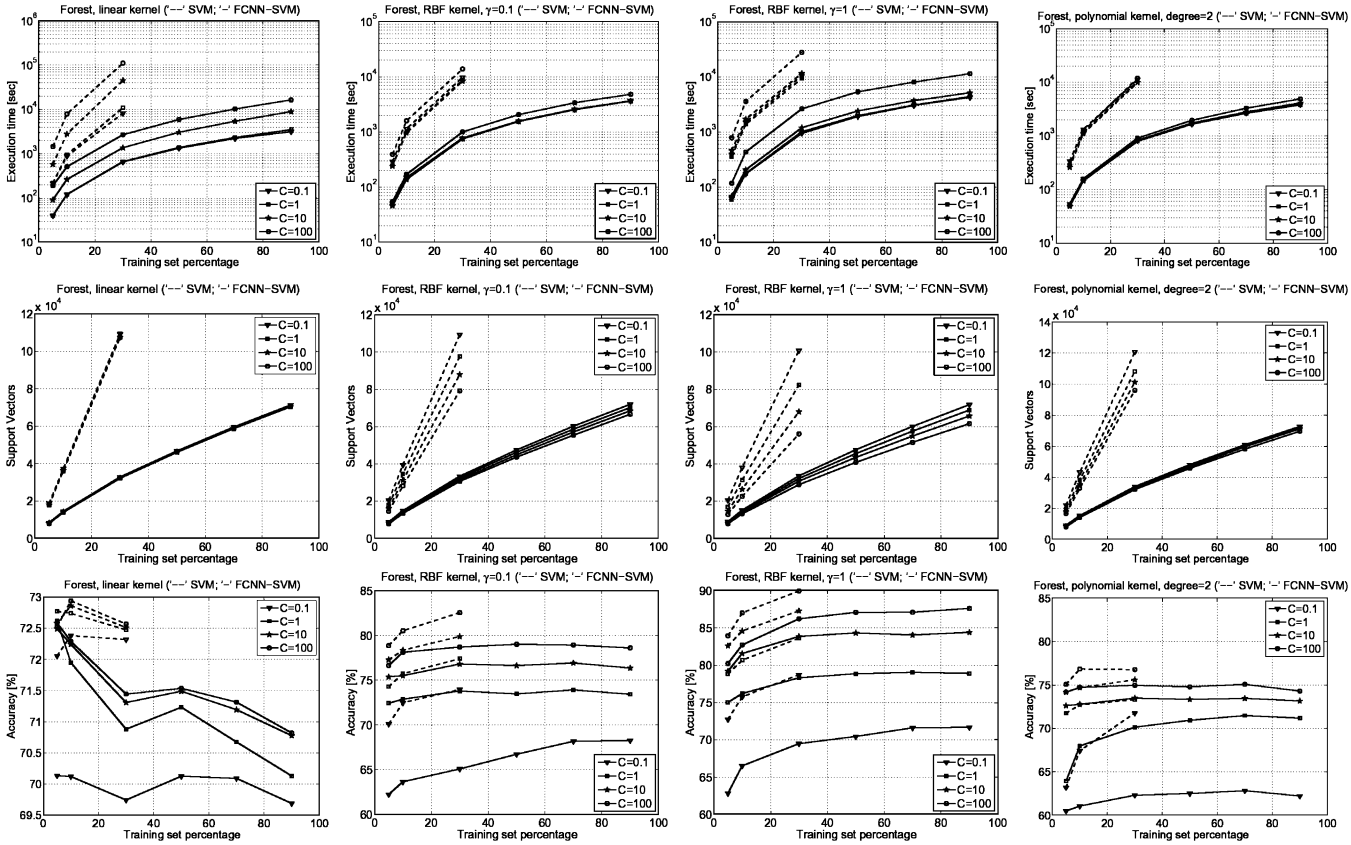

Fig. 3. Experiments on the Forest Cover data set.

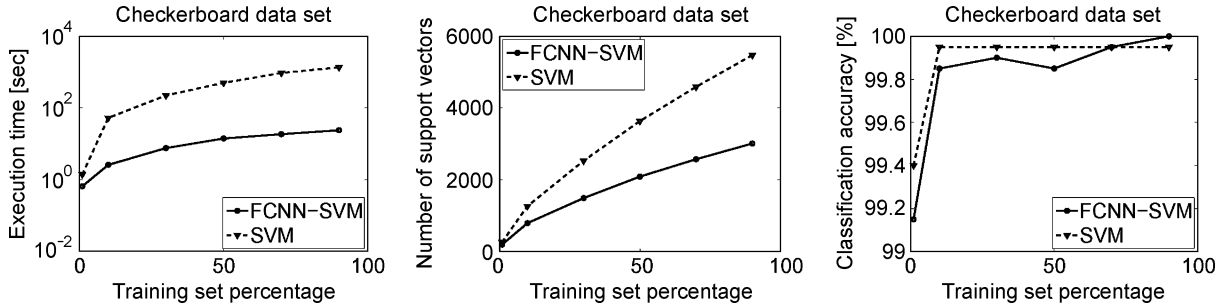

Fig. 4. Experiments on the *Checkerboard* data set.

than half that of the SVM; e.g., by using a polynomial kernel of degree 2 and $C = 0.1$, the number of SVs of the FCNN-SVM is about 20% of that of the SVM. The third row shows the classification accuracy. It is clear that for small values of the parameter C, the FCNN-SVM presents a perceptible loss of accuracy, which, in any case, is within 2%. Interestingly, for greater values of C (e.g., $C = 100$) the accuracies of FCNN-SVM and SVM are comparable.

Fig. 3 shows the results of the experiments on the Forest Cover data set. For this data set, considerations similar to those drawn for the DARPA 1998 data set hold. For example, as for the execution time, by using the linear kernel with $C = 100$ on the 30% sample, the FCNN-SVM terminated in about 2700 s, while the SVM required about 111 000 s.

Fig. 4 shows the results of the experiments on the Checkerboard data set.

While the SVM required about 1350 s, the FCNN-SVM terminated after about 24 s. The FCNN-SVM in this case is about 60 times faster than the SVM. Interestingly, on the largest sample, the FCNN-SVM computed 3012 SVs, while the SVM computed 5474 SVs (55%), without any loss in test accuracy. Indeed, the FCNN-SVM reached the 100% accuracy, while the SVM stabilized on the 99.95% accuracy.

It is interesting to point out that, as far as the execution time and the number of SVs are concerned, as the data set size increases, the performances of the FCNN-SVM improve with respect to those of the SVM. Consider the SVM *relative execution time* (also called *speedup* of the FCNN-SVM) that is the ratio $t_{\mathrm{SVM}}/t_{\mathrm{FCNN-SVM}}$ between the execution time t_{SVM} of the SVM and the execution time $t_{\mathrm{FCNN-SVM}}$ of the FCNN-SVM. We note that, in all the experiments, this ratio increases with the data set size. As an example, the following table reports some SVM relative execution times:

	1%	5%	10%	30%	50%	90%
DARPA	6.91	43.18	35.79	70.51	137.66	–
Forest	2.37	7.68	15.48	41.31	–	–
Checkerboard	2.14	–	20.39	29.26	35.57	56.65

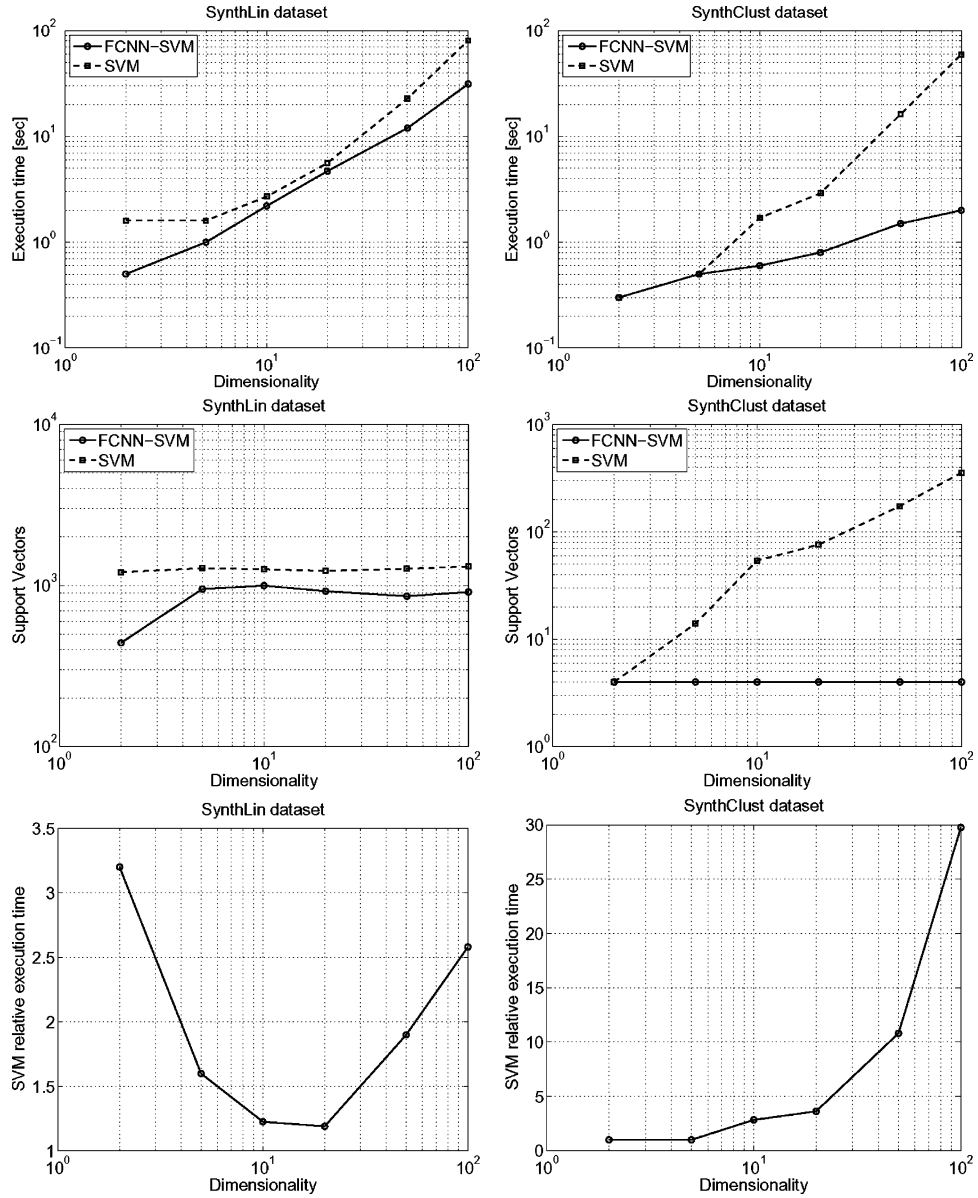

Fig. 5. Experiments on synthetic data sets.

For DARPA 1998 and Forest Cover, the parameters considered in the table above are the linear kernel and C equals 100. From this table, we expect that the larger the data set, the greater the SVM relative execution time. We point out that the above behavior is common to all the other combinations of parameters. Moreover, the FCNN-SVM relative number of SVs% decreases with the data set size. As an example, the following table reports the FCNN-SVM relative number of SVs on the same combination of the parameters above considered:

	1%	5%	10%	30%	50%	90%
DARPA	59%	52%	44%	40%	37%	–
Forest	60%	45%	39%	30%	–	–
Checkerboard	74%	–	63%	59%	57%	55%

Also in this case, the above behavior is common to all the other combinations of parameters.

C. Syntethic Data Sets

Experiments presented in this section are designed to model both advantageous and critical scenarios for the FCNN-SVM method. We considered two families, called *SynthLin* and *SynthClust*, of synthetically generated data sets. The two families differentiate for the data class distribution, while the data sets within the same family differentiate for the dimensionality of the feature space.

The structure of the data set is intentionally simple in order to make immediately intelligible the behavior of the methods. The data sets of the *SynthLin* family are composed of 10 000 points uniformly distributed in the hypercube $[0, 1]^d$, where d is the dimensionality of the feature space ($d \in [2, 100]$). The points are partitioned into two classes by means of a random generated separating hyperplane (classes are almost balanced). Moreover, the labels of 100 randomly selected points (1% of the data set) are flipped in order to simulate the presence of noise. Thus, the classes of these data sets are almost linearly separable, but the points in the data sets "fill" the whole space. The data sets of the *SynthClust* family are composed of 100 000 points distributed in four

well-separated Gaussian clusters in the hypercube $[0, 1]^d$. The clusters are partitioned into two classes in a way that makes them not linearly separable.

Fig. 5 shows the experimental results on the two aforementioned families. As for the *SynthLin* data sets, we used the linear kernel and set the parameter C to 10, since this value guaranteed accuracy close to 99%. As for the *SynthClust* data sets (second row of the figure), we used an RBF kernel with $\gamma = 10$ and set C to 10, since these values guaranteed accuracy close to 100%.

The *SynthLin* data set (first column of Fig. 5) represents a scenario for which the SVM is very appropriate, since classes are almost linearly separable. As for the FCNN-SVM, it must be said that on this kind of data, it does not provide appreciable advantages. Indeed, since the points are uniformly distributed in the unit hypercube and since the class boundary is represented by a hyperplane, as the dimensionality of the space increases, the number of points of one class "facing" the opposite class increases (note the total number of points is held fixed with the dimensionality). Since these are the points contributing to form the FCNN subset, the size of this subset increases as well. As a consequence, the execution time of the FCNN-SVM is comparable to that of the SVM. Moreover, the FCNN-SVM slightly reduces the number of SVs.

The *SynthClust* data set (second column of Fig. 5) represents a common scenario in real-life data, where multidimensional objects are clustered in homogeneous subpopulations. It represents a scenario in which using the FCNN-SVM is very profitable. In fact, it can be observed that in this case the SVM relative execution time worsens as the dimensionality increases, and that the number of SVs selected by the SVM is far greater than that selected by the FCNN-SVM. This can be explained, since the FCNN-SVM works only on the reduced subset selected by the FCNN.

Summarizing, experiments presented in this section serve the purpose of discussing two prototypical scenarios in which using the FCNN-SVM either does not pay much or is very profitable. In general, we can say that the FCNN-SVM does not exhibit large time improvements over the SVM when the number of points contributing to form the FCNN decision boundary is comparable to the size of the training set. This situation could happen, but it is not the standard. In fact, multidimensional real-life data sets are often clustered or even present low intrinsic dimensionality.

Finally, the above experiments show also the scalability behavior of the method with respect to the data dimensionality (while those reported in the preceding section concern scalability with respect to the data set size). Clearly, the absolute execution time should increase with the dimensionality, due to the major cost to compute the distance function. Nonetheless, this does not mean that the speedup of the FCNN-SVM has to worsen. As a matter of fact, for one of the two families shown above (*SynthClust*), the speedup is increasing.

IV. Conclusion

In this brief, we introduced the FCNN-SVM classifier, which greatly reduces the training time and the number of SVs with only a little loss of accuracy with respect to the standard SVM.

References

[1] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1450–1464, Nov. 2007.

[2] G. H. Bakur, L. Bottou, and J. Weston, "Breaking SVM complexity with cross-training," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press, 2004.

[3] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.

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

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[6] W. Gates, "The reduced nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 3, pp. 431–433, May 1972.

[7] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968.

[8] L.-R. Jen and Y.-J. Lee, "Clustering model selection for reduced support vector machines," in *Proc. Int. Conf. Intell. Data Eng. Autom. Learn.*, 2004, pp. 714–719.

[9] B. Karaçali and H. Krim, "Fast minimization of structural risk by nearest neighbor rule," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 127–134, Jan. 2002.

[10] Y.-J. Lee, H.-Y. Lo, and S.-Y. Huang, "Incremental reduced support vector machine," in *Proc. Int. Conf. Inf. Cybern. Syst.*, 2003.

[11] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001.

[12] P. M. Murphy and D. W. Aha, UCI Repository of Machine Learning Databases, Univ. California Irvine, Irvine, CA, 1992 [Online]. Available: www.ics.uci.edu/~mlearn/MLRepository.html

[13] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, 1997, pp. 276–285.

[14] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999.

[15] B. Schölkopf, "Advances in kernel methods: Support vector learning," in *Advances in Neural Information Processing System*, C. J. C. Burges and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999.

[16] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[17] G. Toussaint, "Proximity graphs for nearest neighbor decision rules: Recent progress," in *Proc. Symp. Comput. Statist.*, Montreal, QC, Canada, 2002, pp. 17–20.

[18] V. Vapnik, *The Nature of the Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[19] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 257–286, 2000.

[20] H. Yu, J. Yang, and J. Han, "Classifying large data sets using SVMs with hierarchical clusters," in *Proc. Int. Conf. Knowl. Disc. Data Mining*, 2003, pp. 306–315.