대한산업공학회/한국경영과학회 2002 춘계공동학술대회
한국과학기술원(KAIST) 2002년 5월 3일~4일

k-NN based Pattern Selection for Support Vector Classifiers

Hyunjung Shin and Sungzoon Cho

Department of Industrial Engineering, Seoul National University,
San 56-1, Shillim-Dong, Kwanak-Gu, 151-742, Seoul, Korea
email : {hjshin72, zoon}@snu.ac.kr

Abstract

we propose a *k*-nearest neighbors(*k*-NN) based pattern selection method. The method tries to select the patterns that are near the decision boundary and that are correctly labeled. The simulations over synthetic data sets showed promising results: (1) By converting a non-separable problem to a separable one, the search for an optimal error tolerance parameter became unnecessary. (2) SVM training time decreased by two orders of magnitude without any loss of accuracy. (3) The redundant SVs were substantially reduced.

1 Introduction

The support vector machine(SVM) methodology introduced in [2], is receiving increasing attention in recent years due to its clear-cut theory and practical performance [3][8]. When M training patterns $(\boldsymbol{x}_i, y_i)$, $i=1,\cdots,M$, $\boldsymbol{x}_i \in R^n$, $y_i \in \{-1, 1\}$ are given and the classes are linearly separable, SVM finds the decision boundary $\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b = 0$ that separates the two classes by the largest margin: Find $\boldsymbol{w} \in R^n$, $b \in R$ by optimizing following quadratic programming problem.

$$\begin{aligned} \text{min.} \quad & \Theta(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s. t.} \quad & y_i \cdot (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 \\ & i = 1, \cdots, M. \end{aligned} \tag{1}$$

But, when the two classes are not linearly separable, the constraints in Eq. (1) can not be satisfied. To get over this difficulty, nonnegative slack variable $\xi's$ are introduced in the constraints and a penalty term is added to the objective function as in Eq. (2).

$$\begin{aligned} \text{min.} \quad & \Theta(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \\ \text{s. t.} \quad & y_i \cdot (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i \\ & i = 1, \cdots, M. \end{aligned} \tag{2}$$

In SVM, nonlinear decision boundary can be built through implicit projection onto kernel(feature) space. Commonly used kernel functions are shown in Eq. (3): RBF, polynomial, tansig.

$$\begin{aligned} K_{RBF}(\boldsymbol{x}, \boldsymbol{x}') &= \exp\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2s^2}\right) \\ K_{POLY}(\boldsymbol{x}, \boldsymbol{x}') &= (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^p \\ K_{TANSIG} &= (\rho \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \delta) \end{aligned} \tag{3}$$

And SVM decision function is Eq. (4).

$$f(x) = \text{sign}\left(\sum_{i \in SVs} \alpha_i \cdot K(\boldsymbol{x}_i, \boldsymbol{x})\right) \tag{4}$$

where $\alpha_i's$ are the Lagrange multipliers in dual formulation of Eq. (2). The idea of SVM is simple and easily understandable compared to other machine learning algorithms such as neural networks, decision tree, etc. Besides empirical results in many real world situations show that SVMs practically perform better.

However, difficulty arises when a large set of training patterns is given. The time and memory requirements to solve the quadratic programming increase almost exponentially since the number of training patterns equals to the number of constraints in Eq. (2). Given that "critical" patterns in SVMs are

only a few near the decision boundary, most patterns except them could be considered of no use or even harmful.

One way to circumvent this difficulty is to select only the critical patterns. There have been various methods reported in the literature. In [4], the Mahalanobis distance was used between class core and each pattern to find the boundary patterns. In [6], they implement RBF classifiers, somewhat like SVMs, by selecting patterns near the decision boundary. They propose 1-nearest neighbor method in opposite class after class-wise clustering. But this method presumes that the training set should be clean. In [7], the clean patterns near the decision boundary are selected based on the bias and variance of outputs of a network ensemble. This approach is successful in selecting intended and relevant patterns, though it requires additional time for training a network ensemble. A pattern selection approach, specifically designed for SVMs, is proposed in [1]. They conduct k-means clustering on the entire training set first. Then only the centroids are selected for a homogeneous composition(same class label) while all patterns are selected for a mixed composition. Their approach seems to be successful but a difficulty still remains: how to determine k, the number of clusters.

In this paper, we propose a k-nearest neighbors (k-NN) based method. The idea is to select those patterns with a correct class label near the decision boundary. It is simple and computationally efficient. These are pertinent properties as a preprocessor.

In section 2, the proposed method is introduced with its motives and algorithm in detail. In section 3, simulations over synthetic data sets are shown. In section 4, a conclusion as well as a future research work is given.

2 Pattern Selection Algorithm

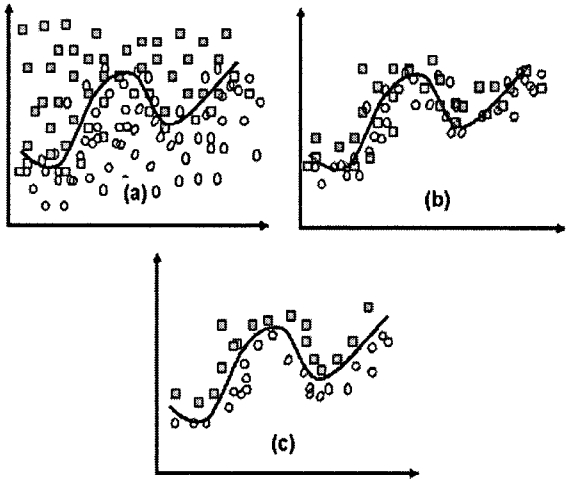
The proposed method tries to select the patterns that are located near the boundary and are correctly labeled. In order to do that, two quantitative measures are introduced, **"proximity"** and **"correctness"**.

First, we introduce proximity. A pattern near the decision boundary tends to have neighbors with mixed class labels. Thus, the entropy of k-nearest neighbors' class labels can estimate the proximity. We select those patterns with "positive" proximity values.

Among them, we want to choose only those with a correct class label. We define correctness as k-NN's voting probability to the pattern's correct class label.

We select only those patterns whose correctness is larger than a threshold, set to 1/J (J is the number of classes) in our experiments.

The effect is as follows: among those patterns near the boundary, the pattern whose class label is same as its neighbors' major class label is regarded as a correct pattern near the decision boundary. On the other hand, the pattern whose class label disagrees with its neighbors' major class label is discarded. Fig. 1 shows the conceptual procedure of the proposed approach. By proximity, patterns near the decision boundary are first selected (a→b). Then by correctness only the clean patterns are selected among them (b→c). Fig. 2 presents the algorithm.



[Fig. 1] Conceptual procedure to select the "clean" patterns near the decision boundary.

① *Find the* k *nearest neighbors for pattern* x.

② *For* x, *calculate the voting probabilities of* k *nearest neighbors over* J *classes.*

$$P_j(x)=\frac{\sum_{i=1}^{k} 1 \text{ if } F_i(x)=j)}{k},$$
$$i=1,\cdots,k,\ j=1,\cdots,J.$$

where $F_i(x)$ *is the label of the* i*th nearest neighbors of* x, $F_i(\cdot)\in\{1,\cdots,J\}$. $F_0(x)$ *is defined as the label of* x *itself.*

③ *Calculate* x*'s proximity to the decision boundary.*

$$\textbf{proximity}\ (x)=\sum_{j=1}^{J} P_j(x)\log_{|J|}\frac{1}{P_j(x)}.$$

In all calculations, $0\log 0$ *is defined to be* 0.

④ *Calculate* x*'s correctness.*

$$\textbf{correctness}\ (x)=P_j^*(x) \text{ where } j^*=F_0(x).$$

⑤ *Apply ①~④ to all* x_s *in the training set.*

⑥ *Select the patterns satisfying the following conditions.*

$$\textbf{proximity}(\cdot)>0 \text{ and } \textbf{correctness}(\cdot)\geq\frac{1}{j}$$

[Fig. 2] Pattern selection algorithm.

One example on 3-class classification when j=3, M=4, k=6 is depicted in Fig. 3. In the example x_1, x_3 and x_4 patterns are selected.

	J*	F_1	F_2	F_3	F_4	F_5	F_6	P_1	P_2	P_3	P_j*	$\sum_j P_j(x_i)\log_M \frac{1}{P_j(x_i)}$
x_1	1	1	1	2	3	1	1	4/6	1/6	1/6	4/6	0.7897
x_2	1	1	1	1	1	1	1	6/6	0/6	0/6	6/6	0
x_3	2	1	1	2	2	3	3	2/6	2/6	2/6	2/6	1
x_4	3	3	3	2	2	3	1	1/6	2/6	3/6	3/6	0.9227

[Fig. 3] Example on the proposed algorithm.

3 Results

The proposed method was tested on two artificial binary classification problems. All simulations were performed on a PENTIUM PC using the Gunn's SVM MATLAB code [5].

The first problem is a continuous XOR problem. From the four gaussian distributions, a total of 600 training patterns were generated. Because of an overlap between the distributions, there are about 10% innate noise patterns, i.e., having an incorrect class label near the decision boundary.

PROBLEM (A):

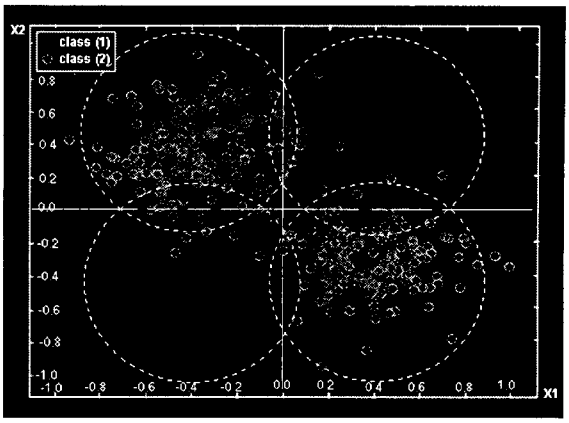
$$class(1) = \{(x_1, x_2) \mid N(C,\ 0.5^2 I) \quad where\ C = (1,1)\ or\ (-1,-1)\},$$
$$class(2) = \{(x_1, x_2) \mid N(C,\ 0.5^2 I) \quad where\ C = (-1,1)\ or\ (1,-1)\}.$$

In the second problem, patterns were generated from the two-dimensional uniform distribution, and then class labels were determined by a decision function. In this problem, four different gaussian noises were added on purpose along the decision boundary, i.e., $N(a, b^2)$ where a = point on the decision boundary and b = gaussian width parameter(0.1, 0.3, 0.8, 1.0). Among 500 training patterns, 20% were incorrectly labeled. Fig. 4 shows both problems after normalization ranged from -1 to 1: Normalization is essential for the better performance of finding the nearest neighbors and of adapting to SVM kernels.

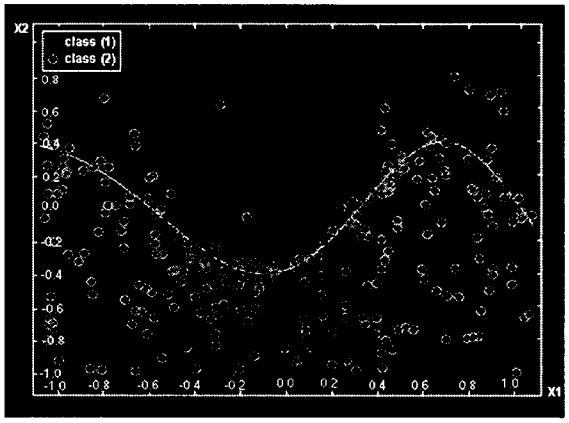
PROBLEM (B):

$$class(1) = \{(x_1, x_2) \mid x_2 > \sin(3x_1 + 0.8)^2\},$$
$$class(2) = \{(x_1, x_2) \mid x_2 \le \sin(3x_1 + 0.8)^2\},$$
$$where\ \ 0 \le x_1 \le 1\ \ and\ \ -2.5 \le x_2 \le 2.5.$$

Notice that in both problems classes cannot be easily separated. And PROBLEM(A) has a sparser density of patterns near the decision boundary than PROBLEM(B).



(a) PROBLEM(A)

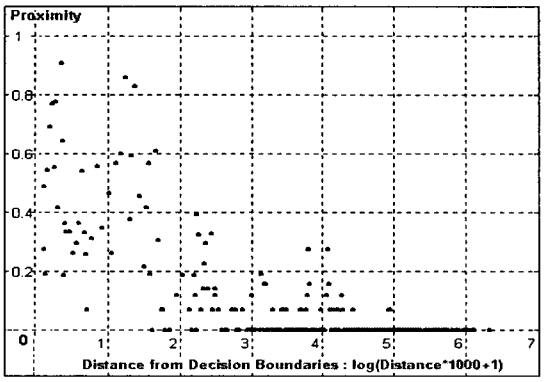


(b) PROBLEM(B)

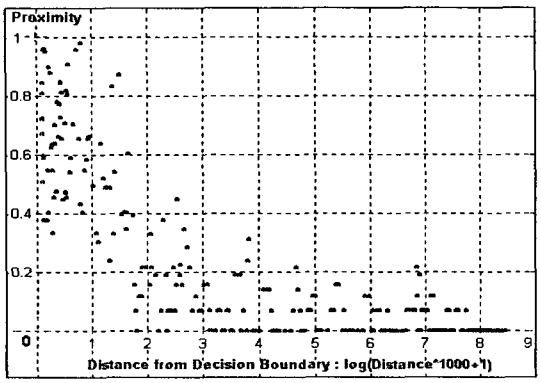
[Fig. 4] Two artificial problems.

The value of k was empirically set as 6 for PROBLEM(A) and 9 for PROBLEM(B). Some other values of k were tried, but such trials did not affect significantly to the SVM performance.

Fig. 5 illustrates the relationship between the proposed "proximity" and the distance from the decision boundary. The patterns near the decision boundary were assigned higher proximity values than the remote ones. It means that proximity offers an appropriate measure for finding the nearby patterns to the decision boundary. Among the patterns with non-zero proximity value, noise patterns were discarded by means of "correctness".



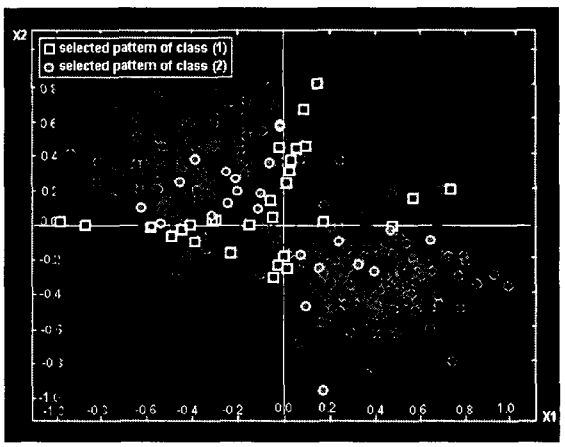
(a) PROBLEM(A)



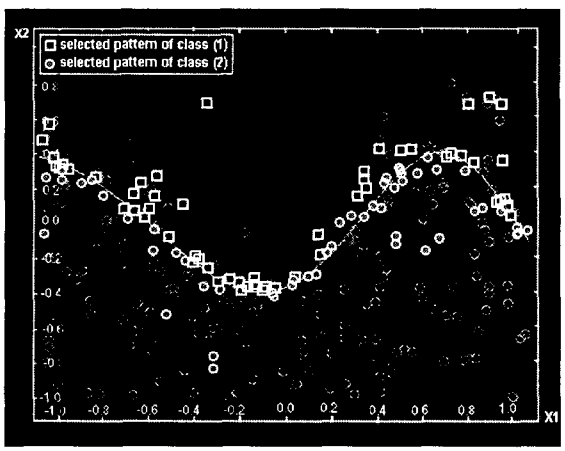
(b) PROBLEM(B)

[Fig. 5] Proximity and the distance from the decision boundary.

In Fig. 6, the selected patterns shown in white contours are scattered against original ones. From both plots, the proposed method seems to extract relevant patterns from redundant ones for SVMs. Only 56(9.3%) patterns were selected for PROBLEM(A) and 109(21.8%) for PROBLEM(B). The difference in reduction ratio is due to the difference in densities near the decision boundary.



(a) PROBLEM(A)



(b) PROBLEM(B)

[Fig. 6] Selected patterns against the original ones.

Table 1 and 2 show SVM adaptation results of CASE(A) and CASE(B), respectively. The upper block of each table is the result for all patterns and the lower block is for the selected patterns. For each problem, 300 test patterns were generated from a statistically identical distribution to its training set. Five RBF kernels with different width parameters(s = 0.25, 0.5, 1, 2, 3) and five polynomial kernels with different order parameters(p = 1, 2, 3, 4, 5) were adopted. For SVMs trained with all patterns, various values were tried for the tolerance parameter (C=0.1, 1, 5, 10, 20, 100, 1000, ∞), since the problem under consideration are non-separable. Whereas for SVMs trained with the selected subset of patterns, C was fixed with ∞ since the proposed selection method removed "incorrectly labeled" patterns through converting a non-separable problem into a separable one.

Performance was compared in terms of the number of support vectors, the execution time and the accuracy(test error). First, The average number of SVs used in the proposed method is just 27.00 for PROBLEM(A) and 86.20 for PROBLEM(B). These are comparable with 297.31 for PROBLEM(A) and 300.21 for PROBLEM(B) when all patterns were used for SVM training. In our method, the uppermost number of SVs are bounded by the number of the selected patterns. If the generalization performance is not improved, there is no reason to project input patterns onto too high dimensional feature space even though all calculations are achieved implicitly.

Second, the average execution time needed to train with all patterns were 748.78(sec) and 430.97(sec), respectively. Whereas, the training time with selected patterns were just 1.89(sec) and 5.67(sec), even including the time elapsed due to the proposed selection procedure. In the worst case, when one doesn't know the data noise level(it happens almost always), one might set the tolerance parameter C=∞. In that case, one should endure 2772.97(sec) and 1626.10(sec) to take the results on such simple artificial problems. Therefore, the proposed method is quite noticeable again at that point.

Finally, for accuracy(test error), SVMs with selected patterns do not degrade their original performances in both problems: on average, PROBLEM(A): 42.53(for all patterns) vs. 41.10 (for the selected patterns) and PROBLEM(B): 51.93 vs. 43.60.

Fig. 7 and 8 shows the decision boundaries and margins of SVMs both with all patterns and with the selected patterns. Kernel parameter was fixed at the value which was best performed with all patterns for comparison purpose(for the selected patterns, it is not the best one). For PROBLEM(A), RBF with s=1 and for PROBLEM(B), polynomial with p=3 are shown(they are marked with outlined circles in the tables). From the figures, it is easily seen that the

[Table 1] Simulation results of PROBLEM(A)

with ALL patterns (# 600) , Tested on 300 patterns

CASE (A)		RBF					POLY				
		s=0.25	s=0.5	s=1.00	s=2.00	s=3.00	p=1	p=2	p=3	p=4	p=5
C=0.1	Exe. Time	506.58	506.74	496.36	470.72	470.49	469.56	491.31	501.41	504.88	518.28
	# of SVs	312	344	549	600	600	600	431	339	292	261
	# of Test Error	34	35	29	76	103	155	27	35	30	31
C=1	Exe. Time	504.99	486.89	459.57	449.89	437.26	469.61	472.80	480.16	491.86	496.47
	# of SVs	194	203	322	557	600	598	262	216	193	185
	# of Test Error	30	36	33	32	103	102	30	32	34	34
C=5	Exe. Time	467.03	456.70	447.97	430.56	431.83	445.39	453.52	470.38	466.87	480.82
	# of SVs	171	172	228	396	552	597	201	180	172	168
	# of Test Error	31	30	32	31	29	95	31	33	32	36
C=10	Exe. Time	511.57	464.56	442.26	424.46	418.37	435.39	453.19	453.85	460.22	465.55
	# of SVs	165	166	203	340	486	596	186	173	168	165
	# of Test Error	30	34	30	36	30	95	29	31	35	36
C=20	Exe. Time	480.32	456.21	444.78	427.10	419.24	443.30	444.62	448.74	452.53	464.94
	# of SVs	164	162	186	292	414	596	180	169	164	159
	# of Test Error	31	35	33	31	30	95	30	30	35	35
C=100	Exe. Time	467.36	467.14	454.83	453.19	435.01	455.77	462.31	455.28	470.87	476.04
	# of SVs	152	159	167	215	291	596	170	163	159	158
	# of Test Error	30	33	29	32	31	95	30	30	35	37
C=1000	Exe. Time	490.21	478.67	475.88	473.79	460.99	471.09	473.02	485.76	489.50	496.20
	# of SVs	150	156	160	174	197	596	167	161	156	157
	# of Test Error	32	31	34	31	30	95	31	29	37	38
C=INF	Exe. Time	2107.76	2697.61	3612.06	2122.32	2455.94	1686.49	2385.74	5511.33	2732.71	1917.78
	# of SVs	375	238	268	599	286	600	305	289	292	600
	# of Test Error	55	36	32	29	33	147	29	31	36	32

with SELECTED patterns (# 56) , Tested on 300 patterns

C=INF	Exe. Time (+0.88)	1.04	1.09	0.98	0.88	0.82	0.99	1.04	1.04	1.15	1.09
	# of SVs	14	11	9	50	56	56	46	9	9	10
	# of Test Error	29	30	30	29	30	145	29	30	30	29

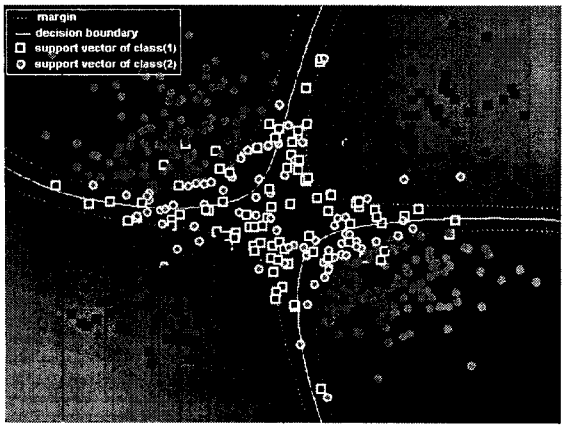
[Table 2] Simulation results of PROBLEM(B)

with ALL patterns (# 500) , Tested on 300 patterns

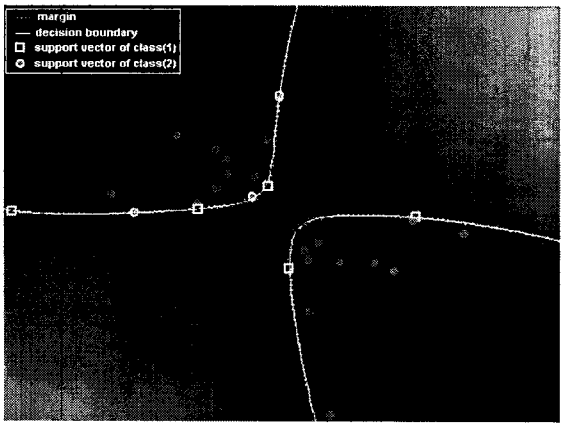
CASE (B)		RBF					POLY				
		s=0.25	s=0.5	s=1.00	s=2.00	s=3.00	p=1	p=2	p=3	p=4	p=5
C=0.1	Exe. Time	280.34	282.10	280.12	278.19	277.32	281.33	281.33	284.30	288.42	290.23
	# of SVs	406	354	367	420	473	345	327	316	306	302
	# of Test Error	49	55	60	62	61	62	63	62	50	48
C=1	Exe. Time	267.71	266.12	260.95	261.72	258.15	259.14	264.36	268.31	272.82	273.04
	# of SVs	281	276	303	325	343	307	297	283	277	271
	# of Test Error	41	49	58	60	62	61	59	52	42	41
C=5	Exe. Time	257.98	252.50	248.05	244.96	243.67	246.01	250.62	265.56	256.78	262.49
	# of SVs	261	254	285	302	311	303	293	272	261	256
	# of Test Error	40	42	49	61	63	61	60	47	44	43
C=10	Exe. Time	254.58	252.22	244.75	240.08	241.67	243.81	244.25	261.78	253.92	254.65
	# of SVs	256	252	278	299	306	303	293	270	257	254
	# of Test Error	44	40	50	57	62	61	60	47	42	43
C=20	Exe. Time	252.54	251.95	249.25	242.28	239.47	243.76	248.53	257.71	255.78	254.80
	# of SVs	248	246	272	295	300	301	293	269	255	253
	# of Test Error	46	43	47	57	61	61	60	48	42	43
C=100	Exe. Time	257.65	252.68	250.85	247.86	253.70	248.38	254.03	263.37	267.76	255.68
	# of SVs	236	243	260	288	294	301	293	266	250	252
	# of Test Error	50	43	40	55	59	61	60	41	42	43
C=1000	Exe. Time	269.03	264.90	264.64	265.29	260.13	258.04	265.84	275.39	279.62	268.92
	# of SVs	234	245	249	273	288	301	293	266	250	251
	# of Test Error	54	40	42	45	57	61	60	45	43	44
C=INF	Exe. Time	1217.43	1328.59	1298.38	1112.24	2397.06	1440.25	2294.98	954.55	2490.32	1727.19
	# of SVs	199	268	334	500	368	463	434	500	372	369
	# of Test Error	92	71	46	39	41	63	60	48	44	44

with SELECTED patterns (# 109) , Tested on 300 patterns

C=INF	Exe. Time (+0.66)	5.16	4.94	4.28	3.95	4.63	5.49	5.71	7.25	3.85	4.66
	# of SVs	21	24	109	109	100	109	108	64	109	109
	# of Test Error	39	40	40	40	41	68	45	43	40	40

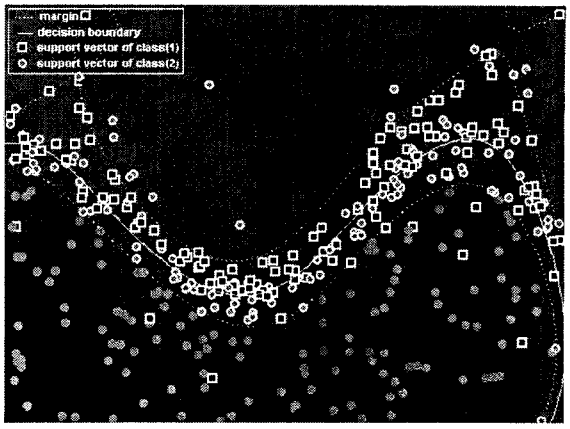


(a) With all patterns

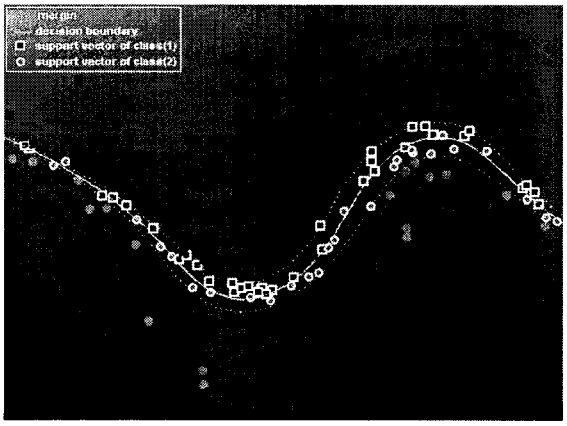


(b) With the selected patterns

[Fig. 7] Decision boundary, margin and SVs : PROBLEM (A) with polynomial kernel(p=3).



(a) With all patterns



(b) With the selected patterns

[Fig. 8] Decision boundary, margin and SVs : PROBLEM (B) with RBF kernel(s=1).

decision boundaries formed using the selected patterns are almost same as those with all patterns. Margins in (b) figures are much narrower than those of original margins in (a) since noise pattern elimination enabled us to set C=∞. And also the number of support vectors are remarkably smaller in (b) figures.

Conclusion

We proposed a pattern selection method as a filtering procedure for SVM training. By utilizing *k*-nearest neighbor method, the patterns with a correct class label near the decision boundary were selected.

The proposed method produced encouraging results as follows. First, the search for an optimal tolerance parameter C is not necessary anymore. Second, SVM execution time decreased two orders of magnitude without any loss of accuracy. Third, by reducing the redundant SVs, the projection onto too high dimensional feature space can be avoided.

Our method can be extended to multi-classification problems without any correction. In this paper, we just demonstrated the proposed method over synthetic data but it is currently applied to over real world problems. Finally, we found that the misclassification error rate of *k*-NN with a large *k* seemed to be similar to the noise level which we imposed on the data on purpose. Hence, this approach could be utilized to predict the data noise level ahead of time.

Acknowledgements

This research was supported by Brain Science and Engineering Research Program sponsored by Korean Ministry of Science and Technology and by the Brain Korea 21 Project.

References

[1] Almeida, M.B., Braga, A. and Braga J.P.(2000). SVM-KM: speeding SVMs learning with a priori cluster selection and k-means, *Proc. Of the 6th Brazilian Symposium on Neural Networks*}, pp. 162-167

[2] Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers, *In D. Haussler, Proc. Of the 5th Annual ACM workshop on Computaional Learning Theory*, Pittsborgh, PA:ACM press

[3] Burges, C.J.C., (1998). A Tutorial on Support Vector Machines for Pattern Recognition, *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167

[4] Foody, G.M., (1999). The Significance of Border Training Patterns in Classification by a Feedforward Neural Network Using Back Propagation Learning, *International Journal of Remote Sensing*, vol. 20, no. 18, pp. 3549-3562

[5] Gunn, S., (1998). Support Vector Machines for Classification and Regression, *ISIS Technical Report*

[6] Lyhyaoui, A., Martinez, M., Mora, I., Vazquez, M., Sancho, J. and Figueiras-Vaidal, A.R., (1999). Sample Selection Via Clustering to Construct Support Vector-Like Classifiers, *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1474-1481

[7] Shin, H. J. and Cho, S. Z., (2002). Pattern Selection Using the Bias and Variance of Ensemble, *Journal of the Korean Institute of Industrial Engineers*, (to appear 2002.03)

[8] Vapnik, V., (1999). *The Nature of Statistical Learning Theory,* Springer. 2nd eds