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
{hjshin72, zoon}@snu.ac.kr

Abstract. SVMs tend to take a very long time to train with a large data set. If "redundant" patterns are identified and deleted in pre-processing, the training time could be reduced significantly. 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]. 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. 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.

H. Yin et al. (Eds.): IDEAL 2002, LNCS 2412, pp. 469–474, 2002.

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 Proposed Algorithm for Pattern Selection

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 $\frac{1}{J}$ (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.

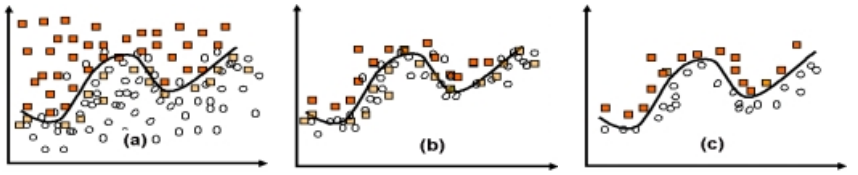

Fig. 1. Conceptual procedure to select the "clean" patterns near the decision boundary.

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.

1. *Find the k nearest neighbors for pattern $\boldsymbol{x}$.*
2. *For $\boldsymbol{x}$, calculate the voting probabilities of k nearest neighbors over J classes.*
   $P_j(\boldsymbol{x}) = \frac{\sum_{i=1}^{k} 1 \quad \text{if} \quad F_i(\boldsymbol{x})=j}{k}, \quad i = 1, \ldots, k, \quad j = 1, \ldots, J.$

   where $F_i(\boldsymbol{x})$ is the label of the ith nearest neighbor of $\boldsymbol{x}$, $F_i(\cdot) \in \{1, \ldots, J\}$. $F_0(\boldsymbol{x})$ is defined as the label of $\boldsymbol{x}$ itself.
3. *Calculate $\boldsymbol{x}$'s proximity to the decision boundary.*
   proximity $(\boldsymbol{x}) = \sum_{j=1}^{J} P_j(\boldsymbol{x}) \log_{|J|} \frac{1}{P_j(\boldsymbol{x})}$.
   In all calculations, $0 log 0$ is defined to be 0.
4. *Calculate $\boldsymbol{x}$'s correctness.*
   correctness $(\boldsymbol{x}) = P_j^*(\boldsymbol{x})$ where $j^* = F_0(\boldsymbol{x})$.
5. *Apply 1 through 4 to all $\boldsymbol{x}_s$ in the training set.*
6. *Select the patterns satisfying the following conditions.*
   proximity $(\cdot) > 0$ and **correctness** $(\cdot) \geq \frac{1}{J}$.

Fig. 2. Pattern selection algorithm

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)=$\{(x_1, x_2) | N\left(C, 0.5^2 I\right)$ where $C = (1, 1)$ or $(-1, -1)\}$,
class(2)=$\{(x_1, x_2) | N\left(C, 0.5^2 I\right)$ where $C = (-1, 1)$or$(1, -1)\}$,
where $-3 \leq x_1 \leq 3$ and $-3 \leq x_2 \leq 3$.

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.

PROBLEM(B) : class(1)=$\{(x_1, x_2) | x_2 > \sin(3x_1 + 0.8)^2\}$,
class(2)=$\{(x_1, x_2) | x_2 \leq \sin(3x_1 + 0.8)^2\}$,
where $0 \leq x_1 \leq 1$ and $-2.5 \leq x_2 \leq 2.5$.

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. 3 shows both problems (above) and selected patterns (below) 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. 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 9.3% patterns were selected for PROBLEM(A) and 21.8% for PROBLEM(B). The different reduction ratio is due to the difference in densities near the decision boundary.

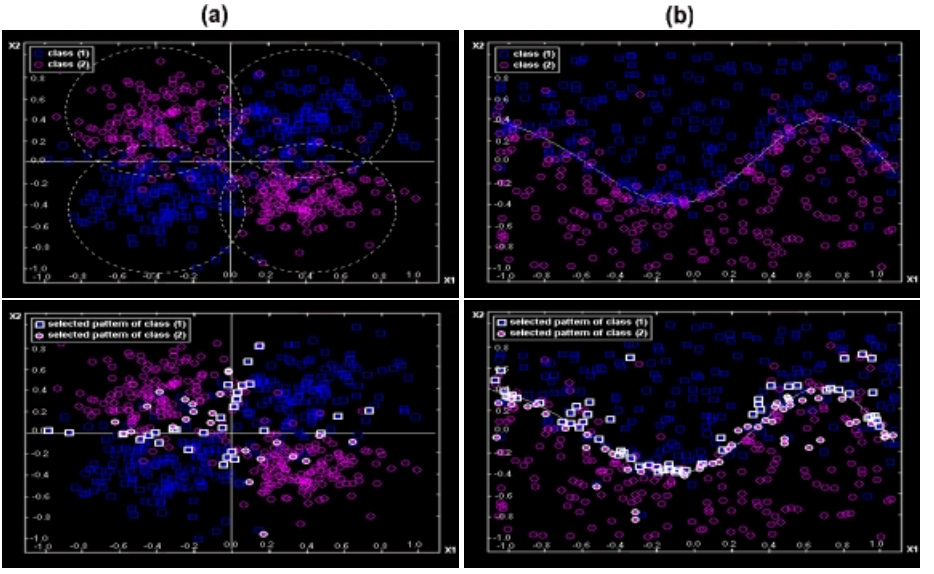

Fig. 3. Original patterns (above) and selected ones (below): (a) PROB(A), (b) PROB(B).

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 ($C = 0.1, 1, 5, 10, 20, 100, 1000, \infty$), 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.

Average performance was compared in terms of the execution time, the number of support vectors and the test error in Table 1. First two columns are ob-

Table 1. SVM Experimental Results

Avg.	PROB(A) ALL ($\forall C$)	PROB(A) ALL ($C = \infty$)	PROB(A) SELECTED ($C = \infty$)	PROB(B) ALL ($\forall C$)	PROB(B) ALL ($C = \infty$)	PROB(B) SELECTED ($C = \infty$)
CPU-time(sec)	748.78	2772.97	1.89 (=0.88+1.01)	430.97	1626.10	5.67 (=0.66+5.01)
No. of patterns	600	600	56	500	500	109
No. of SVs	297.31	385.21	27.00	300.21	380.70	86.20
Test Err.(%)	14.18	15.33	13.70	17.31	18.27	14.53

tained when all training patterns were used. The first column is for all C values we experimented and the second one is for $C = \infty$. The last column is the average results for $C = \infty$ when the selected training patterns were used. The training time with only selected patterns were just 1.89(sec) and 5.67(sec) even including the time taken to execute the proposed selection procedure. For PROBLEM(A), the pattern selection took 0.88(sec) and training SVMs took 1.01(sec). For PROBLEM(B), 0.66(sec) and 5.01(sec) respectively. In the worst case, when one doesn't know the data noise level(it happens almost always), one might set the tolerance parameter $C = \infty$ like in second column. In that case, one should endure 2772.97(sec) and 1626.10(sec) to take the results on such simple artificial problems. Second, 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. Finally, for accuracy, SVMs with selected patterns do not degrade their original performances in both problems. Fig. 4 shows the decision boundaries and margins of SVMs both with all patterns and with the selected patterns. Kernel parameter was

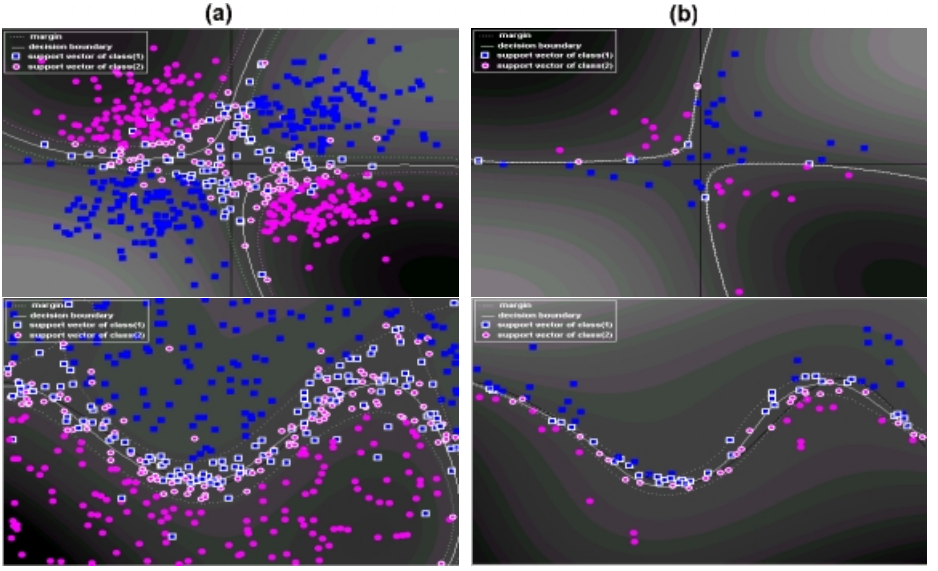

Fig. 4. Decision boundary, margin and SVs, (a) with all patterns and (b) with the selected patterns: the upper figures are for **PROB(A)** with polynomial kernel($p = 3$) and the lower figures are **PROB(B)** with RBF kernel($s = 1$).

fixed at the value which was best performed with all patterns for comparison purpose(for the selected patterns, it is not the best one). From the figures, it is easily seen that the 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 $C = \infty$. And also the number of support vectors are remarkably smaller in **(b)** figures.

4 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)
8. Vapnik, V., (1999). *The Nature of Statistical Learning Theory*, Springer. 2nd eds