

공학박사학위논문

Efficient Pattern Selection for Support Vector
Classifiers and its CRM Application

Support Vector Classifiers 를 위한 효율적 패턴 선택 및 CRM 응용

2005년 2월

서울대학교 대학원

산업공학과 데이터마이닝 전공

신 현 정

공학박사학위논문

Efficient Pattern Selection for Support Vector
Classifiers and its CRM Application

Support Vector Classifiers 를 위한 효율적 패턴 선택 및 CRM 응용

2005년

서울대학교 대학원

산업공학과 데이터마이닝 전공

신 현 정

학위논문 원문 제공 서비스에 대한 동의서

본인은 본인의 연구결과인 학위논문이 앞으로 우리나라의 학문발전에 조금이나마 기여 할 수 있도록, 서울대학교 중앙도서관을 통한 “학위논문 원문제공 서비스”에서 다음과 같은 방법 및 조건 하에 논문을 제공함에 동의합니다.

1. 인터넷을 통한 온라인 서비스와 보존을 위하여 저작물의 내용을 변경하지 않는 범위에서 복제를 허용함.
2. 저작물을 이미지DB(PDF)로 구축하여 인터넷을 포함한 정보통신망에서 공개하여 논문 일부 또는 전부의 복제·배포 및 전송에 동의함.
3. 해당 저작물의 저작권을 타인에게 양도하거나 또는 출판 허락을 하였을 경우 1개월 이내에 서울대학교 중앙도서관에 알림.
4. 배포, 전송된 학위논문은 이용자가 다시 복제 및 전송할 수 없으며 이용자가 연구 목적이 아닌 상업적 용도로 사용하는 것을 금함에 동의함.

논문제목: (영문) Efficient Pattern Selection for Support Vector Classifiers and its CRM Application

(국문) Support Vector Classifiers 를 위한 효율적 패턴 선택 및 CRM 응용

학 과: 산업공학과

학 번: 2000-30378

연락처: 서울시 동작구 사당2동 우성 APT 302-705호 02-595-1444

제출일: 2005년 1월 28일

저작자: 신 현 정 (인)

Abstract

Support Vector Machine (SVM) employs Structural Risk Minimization (SRM) principle to generalize better than conventional machine learning methods employing the traditional Empirical Risk Minimization (ERM) principle. However, training SVM requires large memory and long cpu time when training set is large. One way to circumvent this computational burden is to select some of training patterns in advance which contain most information given to learning. One of the merits of SVM theory distinguishable from other learning algorithms is that it is clear that which patterns are of importance to training. Those are called support vectors (SVs), distributed near the decision boundary, and fully and succinctly define the classification task at hand. Furthermore, on the same training set, the SVMs trained with different kernel functions, i.e., RBF, polynomial, and sigmoid, have selected almost identical subset as support vectors. Therefore, it is worth finding such *would-be* support vectors prior to SVM training. In the thesis, we propose *neighborhood property based pattern selection algorithm* (NPPS) which selects the patterns near the decision boundary based on the neighborhood properties. We utilizes k nearest neighbors to look around the pattern's periphery. The *first* neighborhood property is that "a pattern located near the decision boundary tends to have more heterogeneous neighbors in their class-membership." The *second* neighborhood property dictates that "an overlap or a noisy pattern tends to belong to a different class from its neighbors." The first one is used for identifying those patterns located near the decision boundary. The second one is used for removing the patterns located on the wrong side of the decision boundary. These properties are first implemented as a *naive* form with time complexity $O(M^2)$ where M is the number of given training patterns. Then, to accelerate the pattern selection procedure we utilize another property. The *third* neighborhood property is that "the neighbors of a pattern located near the decision boundary tend to be located near

the decision boundary as well.” The third one skips calculation of unnecessary distances between patterns. This advanced form of algorithm, *fast* NPPS, has a practical time complexity of $O(vM)$ where v is the number of patterns located in the overlap region. The number of patterns located in the overlap region v is closely related to determine a parameter of NPPS, the number of neighbors k , accordingly we provide a heuristic method to set the value of k . Then, we prove invariance of the neighborhood relation under the input to feature space mapping, which assures that the patterns selected by NPPS in input space are likely to be located near decision boundary in feature space in where SVs are defined. NPPS is demonstrated on synthetic as well as real-world problems. The results show that NPPS reduces SVM training time up to almost two orders of magnitude with virtually no loss of accuracy. In addition, to exemplify a practical usage of the algorithm, we applies NPPS to marketing domain problem - response modeling. This experiment is rather a domain-oriented application than a simple algorithm-oriented application. We diagnose the domain specific difficulties which can arise in practice when applying SVM to response modeling, then propose how to alleviate and solve those difficulties: informative sampling, different costs for different classes, and use of distance to decision boundary. Again, SVM with NPPS achieved the accuracies and uplifts comparable to those of SVM with 80–100% random samples, while the computational cost was comparable to those of SVM with 10–20% random samples.

Keywords: Data Mining, Machine Learning, Support Vector Machines (SVM), Pattern Selection, Customer Relationship Management (CRM), Response Modeling

Student Number: 2000–30378

Preface

The thesis consists of a summary report and a collection of six research papers written during the period 2002–2004, and elsewhere published.

December 2004.

HyunJung Shin

Papers included in the thesis

- [A] HyunJung Shin and Sungzoon Cho, (2002). “Pattern Selection for Support Vector Classifiers,” *Proc. of the 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Lecture Notes in Computer Science (LNCS 2412)*, Manchester, UK, pp. 469–474.
- [B] HyunJung Shin and Sungzoon Cho, (2003). “Fast Pattern Selection for Support Vector Classifiers,” *Proc. of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Artificial Intelligence (LNAI 2637)*, Seoul, Korea, pp. 376–387.
- [C] HyunJung Shin and Sungzoon Cho, (2003). “Fast Pattern Selection Algorithm for Support Vector Classifiers: Time Complexity Analysis,” *Proc. of the 4th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Lecture Notes in Computer Science (LNCS 2690)*, Hong Kong, China, pp. 1008–1015.
- [D] HyunJung Shin and Sungzoon Cho, (2003). “How Many Neighbors To Consider in Pattern Pre-selection for Support Vector Classifiers?,” *Proc. of INNS-IEEE International Joint Conference on Neural Networks (IJCNN)*, Portland, Oregon, U.S.A., pp. 565–570.
- [E] HyunJung Shin and Sungzoon Cho, (2004). “Invariance of Neighborhood Relation under Input Space to Feature Space Mapping,” *Pattern Recognition Letters* (to appear—currently appears online on ScienceDirect <http://www.sciencedirect.com/>).
- [F] HyunJung Shin and Sungzoon Cho, (2004). “Response Modeling with Support Vector Machines,” *Data Mining and Knowledge Discovery* (in revision). A preliminary shorter version of the paper, “How to Deal with

Large Dataset, Class Imbalance and Binary Output in SVM based Response Model,” won the *Best Paper Award* in the 2003 Korean Data Mining Society Conference, *Proc. of the Korean Data Mining Society Conference*, Seoul, Korea, pp. 93–107.

Contents

Abstract	i
Preface	iii
Papers included in the thesis	v
1 Introduction	1
2 Literature Review	5
3 Support Vector Machines and Critical Training Patterns	9
4 Naive Pattern Selection Algorithm	13
4.1 <i>Naive</i> NPPS	13
4.2 Limitation of <i>Naive</i> NPPS	18
5 Fast Pattern Selection Algorithm	21

5.1	<i>Fast</i> NPPS	21
5.2	Time Complexity Analysis	22
6	How to Determine the Number of Neighbors	33
6.1	Overlap Region and Overlap Set	34
6.2	Pattern Set with Positive Nearest Entropy	35
6.3	Estimation of the Size of Overlap Set	37
6.4	Procedure to Determine the Number of Neighbors	38
7	Invariance of Neighborhood Relation under Input Space to Feature Space Mapping	41
7.1	Proofs on Validity of Pattern Selection in Input Space	42
7.2	Remark	46
8	Applications with Small to Medium Scale Data sets	49
8.1	Time Complexity of <i>Fast</i> NPPS	51
8.2	Procedure to Determine k	51
8.3	Training Time Reduction	59
8.4	Comparison to A Similar Previous Approach	64
8.5	Real World Problems	66
9	Applications with a Large Scale Data set – Response Modeling For Customer Relationship Management	71
9.1	Introduction	72
9.2	Related Work	73

CONTENTS

ix

9.3	Methods to Cope with Practical Difficulties	77
9.4	Experiments	80
9.5	Summary	90
10	Conclusion and Discussion	93
	Acknowledgements	105

List of Figures

1.1	Pattern selection: a large training set shown in (a) is condensed to a small training set (b) which is composed of only potential support vectors.	3
3.1	SVM classification problem: Through a mapping function $\Phi(\cdot)$, the class patterns are linearly separated in a feature space. The patterns determining both margin hyperplanes are outlined. The decision boundary is the half-way hyperplane between margins. .	10
3.2	Three categories of training patterns	12
4.1	Neighbors_Entropy and Neighbors_Match procedures	14
4.2	An example of selected patterns from three class ($J = 3$) classification problem by setting $k = 6$ and $\beta = 0.5$: the numbers, 1, 2 and 3 stand for the patterns (\vec{x}^i s)' class labels. Six patterns out of 29 are marked by dotted circles to show their function values of LabelProbability, Neighbors_Entropy, Neighbors_Match in table 4.1.	16
4.3	<i>Naive</i> NPPS	17

5.1	A toy example for <i>fast</i> NPPS: the procedure starts with randomly sampled patterns from the initial stage (a), and gets at the final stage (d). “Outlined” circles or squares are the patterns to be expanded (to find its neighbors or to evaluate its neighbors). Among them the selected patterns are marked as “black solid” while expanded but not selected ones as “double outlined”. Neighbor searching is represented as dotted arrows.	23
5.2	<i>Fast</i> NPPS	24
5.3	Theoretical relationship between the computation time and v : the computation time of <i>fast</i> NPPS is linearly proportional to v when M is fixed.	31
6.1	Two class classification problem where circles belong to class 1 while squares belong to class 2. The area enclosed by the two dotted lines comprise the overlap area.	34
6.2	Effect of k on \mathbf{B} : solid dots and squares belong to \mathbf{B}	36
6.3	Schematic probability densities of overlap region.	38
6.4	Procedure to determine the value of k	39
8.1	Two uniform distributions’ overlap: the dark gray area is the overlap region which contains v patterns. The degree of overlap or the number of patterns in the overlap region, v , is set to every decile of training set size, M . (a) and (b) depicts when $v = 0.3M$ and $v = 0.8M$, respectively.	52
8.2	Actual computation time for various values of v	52
8.3	Actual percentage of expanded/selected patterns for various values of v	53
8.4	Continuous XOR with four Gaussian distributions: Selected patterns are outlined. Patterns are normalized ranging from -1 to 1.	55
8.5	Continuous XOR: Number of patterns and SVM test error rate .	55

8.6	Patterns and SVM decision boundaries of Continuous XOR problem ($C = 20$, $\sigma = 0.5$): decision boundary is depicted as a solid line and the margins are defined by the dotted lines in both sides of it. Support vectors are outlined.	56
8.7	Sine Function: Selected patterns are outlined. Patterns are normalized ranging from -1 to 1.	58
8.8	Sine function: Number of patterns and SVM test error rate . . .	58
8.9	Patterns and SVM decision boundaries of Sine Function problem ($C = 20$, $\sigma = 0.5$): decision boundary is depicted as a solid line and the margins are defined by the dotted lines in both sides of it. Support vectors are outlined.	59
8.10	Distance from decision boundary and Neighbors_Entropy: The closer to the decision boundary a pattern is, the higher value of Neighbors_Entropy it has. The selected patterns are depicted as solid circles against outlined ones.	60
8.11	SVM test error comparison: In each cell, the upper number indicates test error rate for ALL and the lower for SELECTED.	63
8.12	4x4 Checkerboard problem: Selected patterns, shown as outlined circles or squares, are scattered against the original ones	65
8.13	Patterns and SVM decision boundaries of 4x4 Checkerboard problem ($C = 20$, $\sigma = 0.25$): decision boundary is depicted as a solid line and the margins are defined by the dotted lines in both sides of it. Support vectors are outlined.	66
8.14	Number of patterns: \hat{v} , b_k , and SVs	67
8.15	SVM CV error rates	68
8.16	MNIST Result Comparison	70
9.1	NPPS and random sampling select different subsets: outlined circles and squares are the patterns belonging to class 1 (non-respondents' group) and class 2 (respondents' group), respectively. Black solid circles and squares are the selected patterns. . .	78

9.2	Accuracies: accuracy of R*-SVM is depicted as a solid circle while that of S-SVM is represented as a dotted reference line.	86
9.3	Lift chart of cumulative average response rate: R*-SVMs are depicted dotted lines but among them R10-SVM is represented as a dash-dot line. S-SVM is represented as a solid-dot line.	87
9.4	Top-Decile response rate and Weighted-Decile response rate: R*-SVM is depicted as a bar while S-SVM is represented as a dotted reference line.	88
9.5	Lift chart of cumulative average profit: R*-SVMs are depicted dotted lines but among them R10-SVM is represented as a dash-dot line. S-SVM is represented as a solid-dot line.	89
9.6	Top-Decile profit and Weighted-Decile profit: R*-SVM is depicted as a bar while S-SVM is represented as a dotted reference line.	89
9.7	How well S-SVM performed relative to R*-SVMs	91

List of Tables

4.1	A Toy Example of Selected Patterns by Selecting Condition . . .	17
4.2	Total Time Complexity of <i>Naive</i> NPPS: “DT” and “ST” stand for distance computation time and search(query) time, respectively.	18
5.1	Notation	25
8.1	Overall Results	50
8.2	Estimation of v for various overlap degrees	53
8.3	Continuous XOR: SVM Result of All Patterns (600)	61
8.4	Continuous XOR: SVM Result of Selected Patterns (179) . . .	61
8.5	Sine Function: SVM Result of All Patterns (500)	62
8.6	Sine Function: SVM Result of Selected Patterns (264)	62
8.7	Best Result Comparison	63
8.8	Execution Time Comparison	64
8.9	Result Comparison	66

8.10 SVM Training Results: ALL vs. SELECTED 68

8.11 MNIST Result Comparison 69

9.1 Input Variables 81

9.2 SVM models: the number of patterns selected from NPPS slightly varies with the given set of each fold, thus it is represented as an average over the five reduced training sets. 82

9.3 Confusion Matrix: FP, FN, TP and TN means false positive, false negative, true positive, and true negative in due order where TP and TN are the correct classification. 83

9.4 Computational Efficiency of SVM response models 90

Introduction

Support Vector Machine (SVM) has been spotlighted in the machine learning community thanks to its theoretical soundness and practical performance. First, it is quite satisfying from a theoretical point of view. SVM can be characterized by three statements (Vapnik, 1999). SVM attempts to position a decision boundary so that the *margin* between the two classes is maximized. The major parameters of SVM are taken from the training patterns. Non-linear SVM is based on the use of kernels to deal with high dimensional feature space without directly working in it. Conventional neural networks tend to overfit the training dataset, resulting in poor generalization since parameter selection is based on Empirical Risk Minimization (ERM) principle which minimizes the error on the training set. On the contrary, the SVM formulation embodies the Structural Risk Minimization (SRM) principle which minimizes the error on the training set with the lowest *capacity*. The difference allows SVM to generalize better, which is the goal in statistical learning. Theoretically, SVM includes a large class of neural networks (including radial basis functions networks), yet it is simple enough to be analyzed mathematically. Second, SVM achieved great success in practical applications as diverse as face detection and recognition, handwritten character and digit recognition, text detection and categorization, etc. (Dumais, 1998; Heisele *et al.*, 2000; Moghaddam and Yang, 2000; Osuna *et al.*, 1997). In particular, *Dumais (1998)* and *Joachims (1998)* applied a number of learning methods to text categorization, such as SVMs, nearest neighbor classifiers, probabilistic Bayesian models, decision trees, and neural networks.

Among them, SVMs achieved most substantial improvements over the currently best performing methods and they behaved robustly over a variety of different learning tasks. Aside from the aforementioned research efforts, *Byun and Lee (2002)* gave a comprehensive up-to-date survey on SVM applications.

However, in SVM quadratic programming (QP) formulation, the dimension of kernel matrix ($M \times M$) is equal to the number of training patterns (M). A standard QP solver has time complexity of order $O(M^3)$: MINOS, CPLEX, LOQO, and MATLAB QP routines. In order to attack the large scale SVM QP problem, the decomposition methods or iterative methods have been suggested which break down the large QP problem into a series of smaller QP problems: Chunking, SMO, SVM^{light}, and SOR (Hearst *et al.*, 1997; Platt, 1999). The general time complexity of those methods is approximately $T \cdot O(Mq + q^3)$ where T is the number of iterations and q is the size of the working set. Needless to say, T increases as M increases.

One way to circumvent this computational burden is to select some of training patterns in advance which contain most information given to learning. One of the merits of SVM theory distinguishable from other learning algorithms is that it is clear that which patterns are of importance to training. Those are called support vectors (SVs), distributed near the decision boundary, and fully and succinctly define the classification task at hand (Cauwenberghs and Poggio, 2001; Pontil and Verri, 1998; Vapnik, 1999). Furthermore, on the same training set, the SVMs trained with different kernel functions, i.e., RBF, polynomial, and sigmoid, have selected almost identical subset as support vectors (Schölkopf *et al.*, 1995). Therefore, it is worth finding such *would-be* support vectors prior to SVM training (see Fig. 1.1). Related researches will be reviewed in the next chapter.

In this thesis, we propose *neighborhood property based pattern selection algorithm* (NPPS). The practical time complexity of NPPS is $O(vM)$ where v is the number of patterns in the overlap region around decision boundary. We utilize k nearest neighbors to look around the pattern's periphery. The *first* neighborhood property is that "a pattern located near the decision boundary tends to have more heterogeneous neighbors in their class-membership." The *second* neighborhood property dictates that "an overlap or a noisy pattern tends to belong to a different class from its neighbors." And the *third* neighborhood property is that "the neighbors of a pattern located near the decision boundary tend to be located near the decision boundary as well." The first one is used for identifying those patterns located near the decision boundary. The second

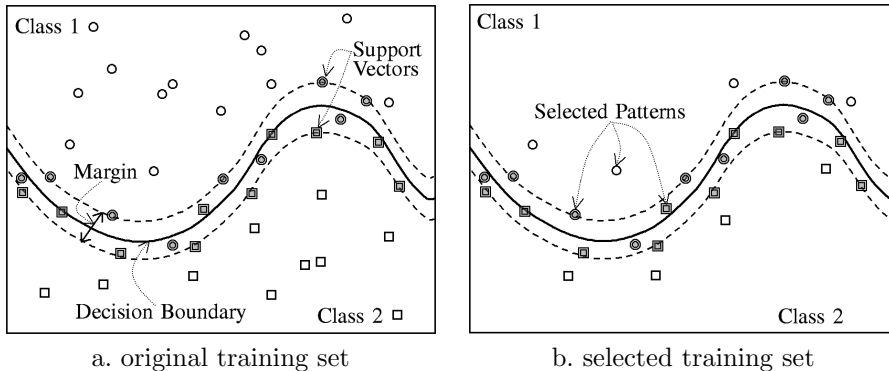


Figure 1.1: Pattern selection: a large training set shown in (a) is condensed to a small training set (b) which is composed of only potential support vectors.

one is used for removing the patterns located on the wrong side of the decision boundary. And the third one is used for skipping calculation of unnecessary distances between patterns, thus accelerating the pattern selection procedure. In short, NPPS uses only local neighbor information to identify those patterns likely to be located near decision boundary.

The thesis is organized as follows. Chapter 2 presents the literature review on pattern selection. Chapter 3 briefly explains the SVM theory, in particular, the patterns critically affecting the training. In chapter 4, we introduce a naive algorithm (*naive* NPPS). In chapter 5, *fast* NPPS is introduced which evaluates the patterns near the decision boundary only. This chapter is divided into the introduction of algorithm and the time complexity analysis. Chapter 6 provides a procedure of determining the number of neighbors, k , which is a parameter used in *naive* NPPS and *fast* NPPS as well. In chapter 7, we provide proofs on the invariance of the neighborhood relation under the input to feature space mapping. The proof assures that the patterns selected by NPPS in input space are likely to be located near decision boundary in feature space in where SVs are defined. In chapter 8, we show the experimental results involving synthetic problems and also real world problems. Chapter 9 exemplifies a practical usage of NPPS by applying to a real problem in marketing domain - response modeling in direct marketing. This chapter is rather a domain-oriented application than a simple algorithm-oriented application. The last chapter concludes the thesis with the limitations of our approach and the related future work.

Literature Review

To date, there have been several approaches on pattern selection for SVM. *Lyhyaoui et al. (1999)* implemented RBF classifiers which somewhat resemble SVMs, to make clear the difference between both methods. RBF classifiers were built based on the patterns near the decision boundary. To find them, they proposed to search 1-nearest neighbor in opposite class after class-wise clustering. But this method presumes that the training set should be clean. *Almeida et al. (2000)* conducted *k*-means clustering first on the entire training set regardless of patterns' class-membership. Those clusters which contain patterns from one class are called *homogeneous*, while those which don't are called *heterogeneous*. All the patterns from a homogeneous cluster are replaced by a single centroid pattern, while the patterns from a heterogeneous cluster are all selected. The drawback of this research is that it is not clear how to determine the number of clusters. *Koggalage and Halgamuge (2004)* also employed clustering to select the patterns from the training set. It is quite similar to *Almeida et al. (2000)*'s approach in that they conducted clustering on the entire training set first and chose the patterns which belong to the clusters having heterogeneous members. For a homogeneous cluster, on the contrary, the patterns along the rim of cluster were selected not the centroid. It is relatively a safer approach since even for homogeneous clusters there can exist the patterns near the decision boundary if the cluster's boundary is almost in contact with the decision boundary. On the other hand, it has a relative shortcoming as well in that the patterns far away from the decision boundary are also picked as long as they lie along the rim.

And further, it is still vague how to set the radius and how to define the width of the rim from it. *Zheng et al. (2003)* proposed to substitute clusters' centroids for random samples of *Lee and Mangasarian (2001)*'s reduced SVM (RSVM). RSVM is to choose random samples from the training set and regard them as support vectors. But all the training patterns are still used as constraints of SVM QP. In RSVM, it is not clear that how many random samples are required not to degrade the original accuracy of SVM. *Zheng et al. (2003)*'s idea on RSVM is based on that the centroids are more representative than random samples. In summary, clustering-based algorithms have a common weakness: the selected patterns are fully dependent on the clustering performance which could be unstable. A related performance comparison was dealt with in the research of *Liu and Nakagawa (2001)*. A bit different approach was done by *Sohn and Dagli (2001)*. In order to reduce the SVM training set as well as to eliminate noisy patterns (outliers), they utilized fuzzy class membership through k nearest neighbors. According to the value of fuzzy class membership, they could check a possibility of the pattern belonging to the class and eliminate the ones having a weak possibility. However, they seem to overlook the importance of the patterns near the decision boundary by treating them equal to the noisy patterns (outliers far from the decision boundary).

In some sense, pattern selection for SVM could be seen somewhat similar to SVM *active learning* or *query learning* since the latter also attempts to identify those critical patterns out of training set. However, there are substantial differences between pattern selection and active (or query) learning. Firstly, the primary motivation for active learning comes from the time or expense of obtaining labeled training patterns, not of training itself. These are likely to be applications of active learning. In some domains, for instance, such as industrial process modeling, a single training pattern may require several days and cost highly to generate. Or in other domains, such as email-filtering (or classifying), obtaining training patterns is not expensive, but may require the user to spend tedious hours to label them. On the other hand, in pattern selection we assume that training patterns, of course labeled, are given. Secondly, active learning is an incremental learning, not a batch learning. Active learning takes turns in training and making queries over the next training pattern. As it were, whenever a new pattern is added to the training set, it runs repeatedly the training algorithm and improves the learner with a newly introduced pattern. On the contrary, in pattern selection, identifying a subset occurs once prior to training, and training is also conducted once over it. Despite the differences, it is still a common concern of both of them to find the patterns near the decision boundary that influence SVM training significantly. In active learning domain, *Schohn and Cohn (2000)* attempted to select a training pattern which maximally narrows the existing margin by evaluating the proximity to the decision hyperplane. *Campbell et al (2000)*'s selecting strategy is to start by requesting the labels of a

random subset of training patterns and subsequently iteratively requesting the label of that patterns which is closest the current decision hyperplane. *Brinker (2003)* issued multiple runs of active learning. To reduce the training runs, he proposed to select batches of new training patterns instead of single one. A batch set of new patterns is selected to satisfy both “minimal distance” from the existing hyperplane and “maximal diversity” among them. Compared with a SVM trained on randomly selected patterns, those active learning heuristics provided significantly better generalization performance for a given number of training patterns.

Approaches to select the patterns near the decision boundary can be found in other learning algorithm. *Shin and Cho (2002)* selected the clean patterns near the decision boundary based on the bias and variance of outputs of a network ensemble. This approach is successful in selecting the intended and relevant patterns, though it consumes too much time in training a network ensemble. *Foody (1999)* showed the significance of decision boundary patterns in training neural networks. He used Mahalanobis distance from class core pattern to find the boundary patterns. A neural network trained with a set of decision boundary patterns may have a lower accuracy of learning but a significantly higher accuracy of generalization than the one trained with a set of class core patterns. To lessen the burden of the MLP training, *Choi and Rockett (2002)* used k NN classifier to estimate the posterior probabilities for pattern selection. But one major drawback is that it takes approximately $O(M^2)$ to estimate the posterior probabilities. The authors argued that the time spent for this prior procedure would not be problematic since conventionally the MLP training needs multiple trials to find hyper-parameters but the pattern selection runs only once. Another drawback is that it is not clear how to determine the value of k . *Hara and Nakayama (2000)* attempted to add extra correct patterns to the selected boundary pattern set in order to enhance the overlap region near the decision boundary, which is common in (Choi and Rockett, 2002). The rationale is that “overlap patterns” located on the “wrong” side of the decision boundary cause the MLP training to take a longer time. Since the derivatives of the back-propagated errors are evaluated at those patterns, the derivatives are very small if they are grouped in a narrow region on either side of the decision boundary. By means of adding extra correct patterns, the network training converged faster. Empirically, we also found a similar performance decline in terms of the SVM accuracy for non-separable classification cases. Our approach to enhancing the decision boundary pattern set is, however, to eliminate the overlap patterns and to retain the correct ones in the overlap region near the decision boundary. More researches related to training with the patterns near the decision boundary can be found in (Hara and Nakayama, 1998; Lee and Landgrebe, 1997; Leisch *et al.*, 1998).

Support Vector Machines and Critical Training Patterns

Support Vector Machines (SVMs) are a general class of statistical learning architecture that performs *structural risk minimization* on a nested set structure of separating hyperplanes (Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002; Vapnik, 1999). Consider a binary classification problem with M patterns $(\vec{x}_i, y_i), i = 1, \dots, M$ where $\vec{x}_i \in \mathfrak{R}^d$ and $y_i \in \{-1, 1\}$. Let us assume that patterns with $y_i = 1$ belong to class 1 while those with $y_i = -1$ belong to class 2. SVM training involves solving the following quadratic programming problem which yields the largest margin ($\frac{2}{\|\vec{w}\|}$) between classes,

$$\begin{aligned}
 \min \quad & \Theta(\vec{w}, \xi) = \frac{1}{2} \|\vec{w}\|^2 + C \sum_i^M \xi_i, \\
 \text{s. t.} \quad & y_i (\vec{w} \cdot \Phi(\vec{x}_i) + b) \geq 1 - \xi_i, \\
 & \xi_i \geq 0, \quad i = 1, \dots, M,
 \end{aligned} \tag{3.1}$$

where $\vec{w} \in \mathfrak{R}^d$, $b \in \mathfrak{R}$ (see Fig. 3.1). Eq.(3.1) is the most general SVM formulation allowing both non-separable and nonlinear cases. The ξ 's are nonnegative slack variables for a non-separable case, which play a role of allowing a certain level of misclassification. The $\Phi(\cdot)$ is a mapping function for a nonlinear case

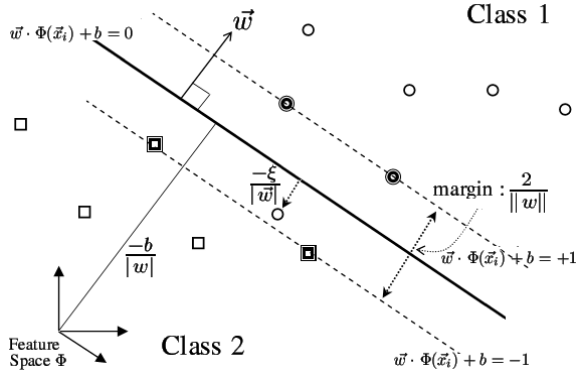


Figure 3.1: SVM classification problem: Through a mapping function $\Phi(\cdot)$, the class patterns are linearly separated in a feature space. The patterns determining both margin hyperplanes are outlined. The decision boundary is the half-way hyperplane between margins.

that projects patterns from the input space into a feature space. This nonlinear mapping is performed implicitly by employing a kernel function, $K(\vec{x}, \vec{x}')$, to avoid the costly calculation of inner products, $\Phi(\vec{x}) \cdot \Phi(\vec{x}')$. There are three typical kernel functions, RBF, polynomial, and tansig in due order,

$$\begin{aligned}
 K(\vec{x}, \vec{x}') &= \exp(-\|\vec{x} - \vec{x}'\|^2 / 2\sigma^2), \\
 K(\vec{x}, \vec{x}') &= (\vec{x} \cdot \vec{x}' + 1)^p, \\
 K(\vec{x}, \vec{x}') &= \tanh(\rho(\vec{x} \cdot \vec{x}') + \delta).
 \end{aligned} \tag{3.2}$$

The optimal solution of Eq.(3.1) yields a decision function of the following form,

$$\begin{aligned}
 f(\vec{x}) = \text{sign}(\vec{w} \cdot \Phi(\vec{x}) + b) &= \text{sign}\left(\sum_{i=1}^M y_i \alpha_i \Phi(\vec{x}_i) \cdot \Phi(\vec{x}) + b\right) \\
 &= \text{sign}\left(\sum_{i=1}^M y_i \alpha_i K(\vec{x}_i, \vec{x}) + b\right),
 \end{aligned} \tag{3.3}$$

where α_i s are nonnegative Lagrange multipliers associated with training patterns, respectively. The solutions, α_i s, are obtained from the dual problem of Eq.(3.1), which minimizes a convex quadratic objective function under con-

straints

$$\min_{0 \leq \alpha_i \leq C} W(\alpha_i, b) = \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j K(\vec{x}_i \cdot \vec{x}_j) - \sum_{i=1}^M \alpha_i + b \sum_{i=1}^M y_i \alpha_i.$$

The first-order conditions on $W(\alpha_i, b)$ are reduced to the Karush-Kuhn-Tucker (KKT) conditions,

$$\begin{aligned} \frac{\partial W(\alpha_i, b)}{\partial \alpha_i} &= \sum_{j=1}^M y_i y_j K(\vec{x}_i, \vec{x}_j) \alpha_j + y_i b - 1 = y_i \bar{f}(\vec{x}_i) - 1 = g_i, \\ \frac{\partial W(\alpha_i, b)}{\partial b} &= \sum_{j=1}^M y_j \alpha_j = 0, \end{aligned} \quad (3.4)$$

where $\bar{f}(\cdot)$ is the function inside the parentheses of *sign* in Eq.(3.3). The KKT complementarity condition, Eq. (3.4), partitions the training pattern set into three categories according to the corresponding α_i s.

- (a) $g_i > 0 \rightarrow \alpha_i = 0$: irrelevant patterns
- (b) $g_i = 0 \rightarrow 0 < \alpha_i < C$: margin support vectors
- (c) $g_i < 0 \rightarrow \alpha_i = C$: error support vectors

Fig. 3.2 illustrates those categories (Cauwenberghs and Poggio, 2001; Pontil and Verri, 1998). The patterns belonging to (a) are out of the margins, thus irrelevant to training, while the patterns belonging to (b) and (c) are critical ones directly affecting training. They are called support vectors (SVs). The patterns of (b) are strictly on the margin, hence called *margin SVs*. On the other hand, the patterns of (c) lie between two margins, hence called *error SVs* but are not necessarily misclassified. (Note that there is another type of SVs belonging to the category of error SVs. They are such patterns incorrectly labelled, and very *far from the decision boundary*. Since they hardly seem to be originated from natural class overlap, we regard them as outliers or noises which do not contribute the margin construction. For that reason, we will focus on the SVs residing around decision boundary not deep in the realm of the opposite class.)

Going back to Eq.(3.3), we can now see that the decision function is a linear combination of kernels on only those critical training patterns (denoted as SVs) because the patterns corresponding to $\alpha_i = 0$ have no influence on the decision

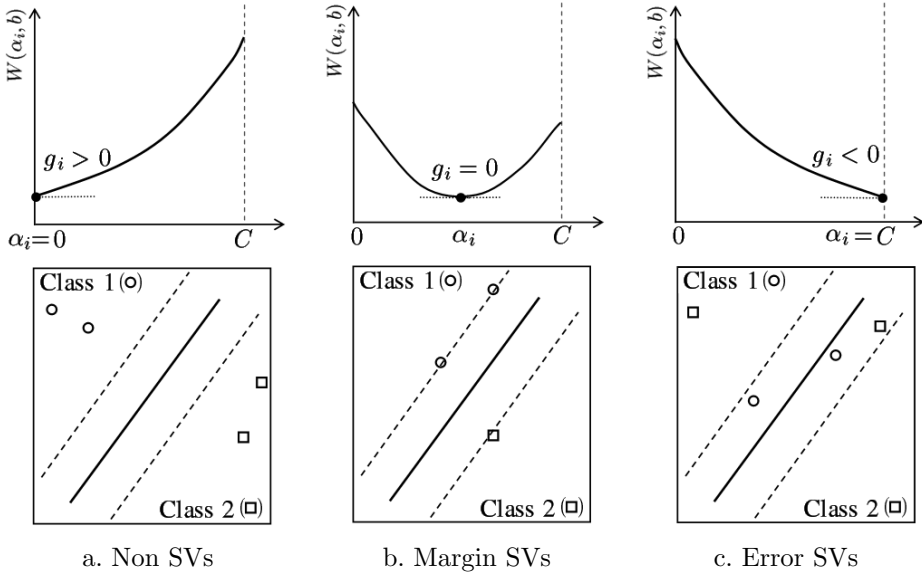


Figure 3.2: Three categories of training patterns

result.

$$f(\vec{x}) = \text{sign} \left(\sum_{i=1}^M y_i \alpha_i K(\vec{x}_i, \vec{x}) + b \right) = \text{sign} \left(\sum_{i \in \mathbf{SV}_s} y_i \alpha_i K(\vec{x}_i, \vec{x}) + b \right). \quad (3.5)$$

To sum up, it is clear in SVM theory which patterns are of importance to training. Those are distributed near the decision boundary, and fully and succinctly define the classification task at hand. And the SVMs trained with different kernel functions (RBF, polynomial, tansig) on the same training set have been founded to select almost identical subset as support vectors (Schölkopf *et al.*, 1995). Therefore, it is worth finding such would-be support vectors in advance.

Naive Pattern Selection Algorithm

In this chapter, we introduce *naive* NPPS and indicate its limitation.

4.1 *Naive* NPPS

The proposed idea is to select only those patterns located around decision boundary since they are the ones that contain most information. Obviously, the decision boundary is not known until a classifier is built. Thus, the algorithm utilizes neighborhood properties to infer the proximity of a pattern to the decision boundary. *The first neighborhood property* is that “a pattern located near the decision boundary tends to have more heterogeneous neighbors in their class-membership.” Thus, the proximity of pattern \vec{x} 's to the decision boundary is estimated by “*Neighbors_Entropy* (\vec{x}, k)”, which is defined as the entropy of the pattern \vec{x} 's k -nearest neighbors' class labels (see Fig. 4.1). In most cases, a pattern with a positive value of “*Neighbors_Entropy* (\vec{x}, k)” is close to the decision boundary, thus selected. Those patterns are likely to be SVs, which correspond to the margin SVs in Fig. 3.2.b or the error SVs in Fig. 3.2.c. Among the patterns selected, however, overlap patterns or noisy patterns are also present. Here, let us first define *overlap patterns* as the patterns that are

```

LabelProbability ( $\vec{x}, k$ ) {
  /* For  $\vec{x}$ , calculate the label probabilities of  $k\text{NN}(\vec{x})$  over  $J$  classes,
   $\{C_1, C_2, \dots, C_J\}$ , where  $k\text{NN}(\vec{x})$  is defined
  as the set of  $k$  nearest neighbors of  $\vec{x}$ . */
   $k_j = |\{\vec{x}' \in C_j | \vec{x}' \in k\text{NN}(\vec{x})\}|$ ,  $j = 1, \dots, J$ .

  return ( $P_j = \frac{k_j}{k}, \forall j$ ).
}

Neighbors_Entropy ( $\vec{x}, k$ ) {
  /* Calculate the neighbors-entropy of  $\vec{x}$  with its nearest neighbors' labels.
  In all calculations,  $\log_J \frac{1}{0}$  is defined to be 0. */

  Do LabelProbability( $\vec{x}, k$ ).
  return ( $\sum_{j=1}^J P_j \cdot \log_J \frac{1}{P_j}$ ).
}

Neighbors_Match ( $\vec{x}, k$ ) {
  /* Calculate the neighbors-match of  $\vec{x}$ .
   $j^*$  is defined as the label of  $\vec{x}$  itself.*/
   $j^* = \arg_j \{C_j | \vec{x} \in C_j, j = 1, \dots, J\}$ .
  Do LabelProbability( $\vec{x}, k$ ).
  return ( $P_{j^*}$ ).
}

```

Figure 4.1: Neighbors_Entropy and Neighbors_Match procedures

located in the *other* side of the decision boundary since the class distributions' overlap. Overlap region is a region in feature space occupied by the overlap patterns from either side of the decision boundary. Note that the overlap region contains not only the overlap patterns, but also the *close non-overlap* patterns which are located close to the decision boundary, yet in the *right* side of the decision boundary. On the contrary, genuine *noisy patterns* are also defined as the patterns that are located in the other side of the decision boundary, but far away from the decision boundary. Those are not adjacent to overlap patterns since they occur due to reasons other than class distribution overlap. Either overlap patterns or noisy patterns have to be identified and removed as much as possible since they are more likely to be the error SVs misclassified (between the margins or far from the margins, see Fig. 3.2.c).

With this end, we take *the second neighborhood property*- “an overlap or a noisy

pattern tends to belong to a different class from its neighbors.” If a pattern’s own label is different from the majority label of its neighbors, it is likely to be incorrectly labeled. The measure “*Neighbors_Match* (\vec{x}, k)” is defined as the ratio of \vec{x} ’s neighbors whose label matches that of \vec{x} (see Fig. 4.1). The patterns with a small *Neighbors_Match*(\vec{x}, k) value is likely to be the ones incorrectly labeled. Only the patterns satisfying [Selecting Condition],

$$\text{Neighbors_Entropy}(\vec{x}, k) > 0 \text{ and } \text{Neighbors_Match}(\vec{x}, k) \geq \beta \cdot \frac{1}{J}.$$

are selected where $0 < \beta \leq 1$. The larger value of β leads to the smaller number of selected patterns. Setting β with a value of 1 means that we select the patterns correctly classified by k NN, preliminary to the original classifier, SVM. Thus, some of the critical patterns near the decision boundary might be discarded. To conserve the original SVM accuracy, we lessen the prior influence of k NN by weighing down $\beta = 0.5$. By doing so, the overlap (genuine noise) patterns far away from the decision boundary can still be eliminated while the patterns near the decision boundary can be more preserved.

One simple example of the patterns selected according to Selecting Condition is depicted in Fig. 4.2 when $J = 3$ and $k = 6$. The numbers, 1, 2, and 3 in the figure, stand for the patterns(\vec{x}^i s)’ class labels. We show the function values of LabelProbability, Neighbrs_Entropy, Neighbors_Match in table 4.1 only for six patterns marked by dotted circles among them. Let us consider \vec{x}^1 in class 1 region first. It is remote from the decision boundary, thus, is surrounded by the neighbors which all belong to class 1 like \vec{x}^1 itself. Actually, \vec{x}^1 does not satisfy the Selecting Criteria since its Neighbors_Entropy value is zero. Meanwhile, \vec{x}^2 is a noise pattern which resides in the other class region. It is surrounded by the neighbors which all belong to the same class. Thus, it has zero Neighbors_Entropy value. However, it is different from the pattern \vec{x}^1 in that its neighbors are all belonging to class 2 while \vec{x}^2 itself belongs to class 1, which results in zero Neighbors_Match value. In any case, the remote patterns from the decision boundary like \vec{x}^1 and \vec{x}^2 are excluded from selection. On the other hand, \vec{x}^3 which is close to the decision boundary has various neighbors in their class membership. Two of its neighbors belong to class 1, another two belong to class 2, and the rest belong to class 3. Thus, its Neighbors_Entropy is 1, and the Neighbors_Match is 1/3. Therefore, the pattern \vec{x}^3 is selected because it satisfies the Selecting Criteria ($\beta = 1.0$). Similarly, among the six patterns in the table 4.1, \vec{x}^4 and \vec{x}^5 , are chosen. If we reduce the value of $\beta=1.0$ to 0.5, \vec{x}^6 is also added to the selected pattern set. See table 4.1 and note that the patterns

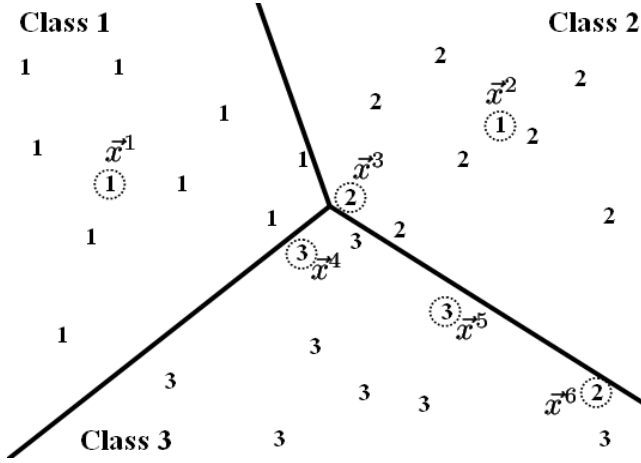


Figure 4.2: An example of selected patterns from three class ($J = 3$) classification problem by setting $k = 6$ and $\beta = 0.5$: the numbers, 1, 2 and 3 stand for the patterns (\vec{x}^i 's) class labels. Six patterns out of 29 are marked by dotted circles to show their function values of LabelProbability, Neighbors_Entropy, Neighbors_Match in table 4.1.

selected are near the decision boundary in Fig. 4.2.

The parameter β controls the selectivity. The larger value of β leads to the smaller number of selected patterns. We have empirically set β with a value of 0.5. Setting β with a value of 1 means that we select the patterns correctly classified by k NN, preliminary to the original classifier, SVM. In general, however, k NN classifier is not as good as a sophisticated classifier such as SVM. Thus, some of the critical patterns near the decision boundary might be discarded. To conserve the original SVM accuracy, we lessen the prior influence of k NN by weighing down $\beta = 0.5$. By doing so, the overlap (genuine noise) patterns far away from the decision boundary can still be eliminated while the patterns near the decision boundary can be more preserved. More discussion of the parameter β will be given in the last chapter.

Fig. 4.3 shows the algorithm, *naive* NPPS. For each of M patterns, we find the k nearest neighbors from the rest $M - 1$ patterns. If a pattern satisfies the Selecting Criteria, it is added to the selected pattern set \mathbf{S} . This procedure is iterated over all patterns in \mathbf{D} , then we get the final pattern set \mathbf{S} in the end.

Table 4.1: A Toy Example of Selected Patterns by Selecting Condition

j^* is the class label of \bar{x}^i 's own, and j^k is that of its k^{th} nearest neighbor.

	j^*	Neighbors Class Label						LabelProbability			Neighbors		Selected Patterns	
		j^1	j^2	j^3	j^4	j^5	j^6	P_1	P_2	P_3	Entropy	Match	$\beta = 1.0$	$\beta = 0.5$
\bar{x}^1	1	1	1	1	1	1	1	6/6	0/6	0/6	0	6/6	X	X
\bar{x}^2	1	2	2	2	2	2	2	0/6	6/6	0/6	0	0/6	X	X
\bar{x}^3	2	1	1	2	2	3	3	2/6	2/6	2/6	1	2/6	○	○
\bar{x}^4	3	3	3	2	2	3	1	1/6	2/6	3/6	0.9227	3/6	○	○
\bar{x}^5	3	3	3	3	3	3	2	0/6	1/6	5/6	0.4100	5/6	○	○
\bar{x}^6	2	3	3	3	3	3	2	0/6	1/6	5/6	0.4100	1/6	X	○

```
NaiveNPPS(D,  $k$ ) {
```

```
  /* Constitute the select patterns set S from the original training set D
  where  $\bar{x}^i \in \mathbf{D}$ ,  $i = 1, \dots, M$ . The number of classes  $J$  is given. */
```

```
  Initialize the selected pattern set S.
```

```
  S  $\leftarrow \emptyset$ .
```

```
  For ( $i \leftarrow 1$  :  $i \leq M$  :  $i++$ )
```

```
    Add  $\bar{x}^i$  satisfying the [Selecting Condition] to the selected pattern set S.
```

```
    S  $\leftarrow \mathbf{S} \cup \{\bar{x}^i \mid Neighbors\_Entropy(\bar{x}^i, k) > 0 \text{ and}$   

                       $Neighbors\_Match(\bar{x}^i, k) \geq \beta \cdot \frac{1}{j}, \bar{x}^i \in \mathbf{D}\}$ .
```

```
  end
```

```
  return S
```

```
}
```

Figure 4.3: Naive NPPS

Table 4.2: Total Time Complexity of *Naive* NPPS: “DT” and “ST” stand for distance computation time and search(query) time, respectively.

DT	$O(d \cdot (M - 1))$
ST	$\min\{O((M - 1) \cdot \log(M - 1)), O(k \cdot (M - 1))\}$
Total Time Complexity	$O(M \cdot (DT + ST)) \approx O(M^2)$

4.2 Limitation of *Naive* NPPS

In *naive* NPPS (Fig. 4.3), the k NNs of all patterns are evaluated. This algorithm is easy to implement and also runs in a reasonable amount of time as long as the size of training set, M , is relatively small. However, when the size of the training set is large, the computational cost increases in proportion to the size. Let us assume that the distance between any two points in d -dimensional space can be computed in $O(d)$. Then, finding the nearest neighbors for each pattern takes sum of distance computation time “DT” and search time “ST” (see table 4.2). The total time complexity of *naive* NPPS, therefore, is $O(M \cdot (DT + ST))$. Roughly speaking, it is $O(M^2)$ since in general, $d \ll M$, $k \ll M$ and $k \leq \log(M - 1)$ hold.

There is a considerable amount of literature on efficient nearest neighbor searching algorithms for large data sets of a high dimension. Most approaches focus on reducing DT or ST. To mitigate the complexity of the distance computation time DT, *Grother et al. (1997)* employed L_∞ in place of L_2 distance metric. But disadvantage of this metric is an increase in error although this may be traded off for speed by using a larger training set. See also (Short and Fukunaga, 1981) for various distance measures for the nearest neighbor methods. Another technique in (*Grother et al., 1997*) to lessen DT is early rejection of the patterns during distance computation. If the distance till the d' ($d' \ll d$) features of the d -dimensional pattern already exceeds the distance to the current k^{th} closest neighbor, the rest $d - d'$ features are ignored from the calculation. So the distance between any two patterns can be computed within $O(d')$ not $O(d)$. Feature ordering methods can afford better improvement for this partial distance calculation. Reducing the search time ST is quite closely related to those researches on data structure. Replacing linear search with a tree traversal yields benefits for reducing not only ST but also DT by avoiding redundant pattern searching (*Arya et al., 1998; Bentley, 1975; Friedman et al., 1977; Grother et al., 1997; Guttman, 1984*). Another scheme to reduce the ST is to improve the termination condition of searching procedure. *Masuyama et al. (1999)* used an enforced termination condition on the basis of a branch-and-bound algorithm.

Their approach reduced the amount of computation from a few percent to 30 percent. In addition to the techniques introduced above, many other approaches were suggested such as the “approximated” NN rule (Arya *et al.*, 1998; Indyk, 1998), the “condensed” NN rule (Hart, 1968), and the “reduced” NN rule (Gates, 1972). See more in this context (Berchtold *et al.*, 1997; Borodin *et al.*, 1999; Ferri *et al.*, 1999; Johnson *et al.*, 2000; Kleingberg, 1997; Tsaparas, 1999) and also therein.

One of the well-developed methods above can easily reduce (DT + ST) of time complexity of *naive* NPPS. This falls on the second M of the naive algorithm complexity $O(M \cdot M)$. On the contrary, our efforts focus on reducing the first M , namely, reducing the number of patterns which have to evaluate its neighbors.

Fast Pattern Selection Algorithm

In this chapter, we propose *fast* NPPS which evaluates the patterns near the decision boundary only. We also provide its time complexity analysis.

5.1 *Fast* NPPS

Naive NPPS evaluating k NNs for M patterns has time complexity of $O(M^2)$, so the pattern selection process itself can be time-consuming. To accelerate the pattern selection procedure, let us consider *the third neighborhood property*, “*the neighbors of a pattern located near the decision boundary tend to be located near the decision boundary as well.*” Assuming the property, one may compute only the neighbors’ label entropy for the patterns near the decision boundary instead of all the training patterns. *A pattern is expanded* or *a pattern’s neighbors are evaluated* when its Neighbors_Entropy is positive. With an initial set of randomly selected patterns, we evaluate only the neighbors of a pattern satisfying [Expanding Condition]

$$\text{Neighbors_Entropy}(\vec{x}, k) > 0$$

in the next step. This successive *neighbors only* evaluation of the *current* pattern set is repeated until all the patterns near the decision boundary are chosen and evaluated.

For better understanding, Fig. 5.1 presents a toy example. Let us assume $J = 2$, $k = 3$, and $\beta = 1$. At the initial stage shown in Fig. 5.1.a, three patterns \vec{x}^1 , \vec{x}^2 , and \vec{x}^3 are randomly selected. They are marked as outlined circles or squares. Neighbor searching is represented by dotted arrows. After its neighbors' label composition is evaluated, \vec{x}^1 is not expanded since it does not meet the Expanding Condition. On the other hand, \vec{x}^2 and \vec{x}^3 are expanded. \vec{x}^2 satisfies the Selecting Condition as well as the Expanding Condition. Thus, it is added to the selected pattern set. \vec{x}^3 , however, does satisfy the Expanding Condition only. The patterns like \vec{x}^1 are depicted as “gray” circles or squares, \vec{x}^2 as “black solid” and \vec{x}^3 as “double outlined” in the next stage. Now in Fig. 5.1.b, \vec{x}^4 , \vec{x}^5 , \vec{x}^6 and \vec{x}^7 , \vec{x}^8 , \vec{x}^9 which are the neighbors of \vec{x}^2 and \vec{x}^3 , respectively, are evaluated. Among them, \vec{x}^4 is excluded from expanding, while the rest are all selected and expanded in Fig. 5.1.c. Neighborhood relationship is more than somewhat mutual, hence expanding is conducted only once per pattern to avoid redundant evaluation. Therefore, only those patterns that were not evaluated before such as \vec{x}^{10} , \vec{x}^{11} , \vec{x}^{12} , and \vec{x}^{13} are evaluated as shown in Fig. 5.1.c. Similarly, \vec{x}^{10} and \vec{x}^{13} are not expanded, and \vec{x}^{11} and \vec{x}^{12} are added to the selected pattern set, and their neighbors \vec{x}^{14} and \vec{x}^{15} are evaluated in the last stage, Fig. 5.1.d.

Fast NPPS is shown in Fig. 5.2 using notations displayed in Table 5.1.

5.2 Time Complexity Analysis

In this section, we show that *fast* NPPS terminates within a finite number of steps, and that its time complexity is significantly smaller than that of *naive* NPPS.

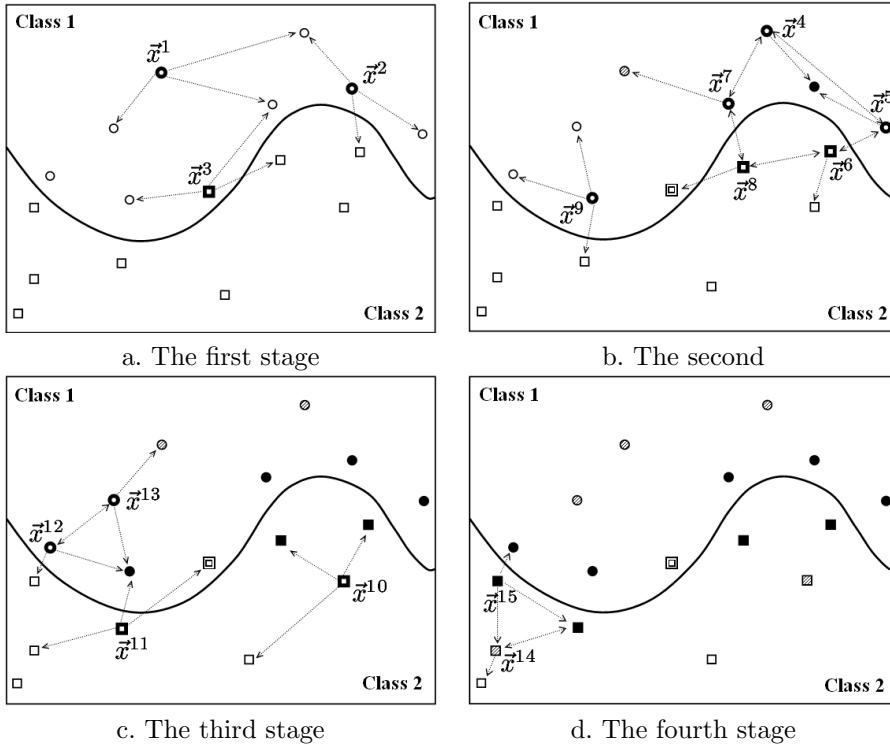


Figure 5.1: A toy example for *fast* NPPS: the procedure starts with randomly sampled patterns from the initial stage (a), and gets at the final stage (d). “Outlined” circles or squares are the patterns to be expanded (to find its neighbors or to evaluate its neighbors). Among them the selected patterns are marked as “black solid” while expanded but not selected ones as “double outlined”. Neighbor searching is represented as dotted arrows.

```

FastNPPS ( $\mathbf{D}$ ,  $k$ ) {
  [0] Initialize  $\mathbf{D}_e^0$  with randomly chosen patterns from  $\mathbf{D}$ .
      Constants  $k$  and  $J$  are given. Initialize  $i$  and various sets as follows:
       $i \leftarrow 0$ ,  $\mathbf{S}_o^0 \leftarrow \emptyset$ ,  $\mathbf{S}_x^0 \leftarrow \emptyset$ ,  $\mathbf{S}^0 \leftarrow \emptyset$ .

  while  $\mathbf{D}_e^i \neq \emptyset$  do
    [1] Choose  $\vec{x}$  satisfying [Expanding Condition].
         $\mathbf{D}_o^i \leftarrow \{\vec{x} \mid \text{Neighbors\_Entropy}(\vec{x}, k) > 0, \vec{x} \in \mathbf{D}_e^i\}$ .
         $\mathbf{D}_x^i \leftarrow \mathbf{D}_e^i - \mathbf{D}_o^i$ .

    [2] Select  $\vec{x}$  satisfying [Selecting Condition].
         $\mathbf{D}_s^i \leftarrow \{\vec{x} \mid \text{Neighbors\_Match}(\vec{x}, k) \geq \beta/J, \vec{x} \in \mathbf{D}_o^i\}$ .

    [3] Update the pattern sets.
         $\mathbf{S}_o^{i+1} \leftarrow \mathbf{S}_o^i \cup \mathbf{D}_o^i$  : the expanded,
         $\mathbf{S}_x^{i+1} \leftarrow \mathbf{S}_x^i \cup \mathbf{D}_x^i$  : the non-expanded,
         $\mathbf{S}^{i+1} \leftarrow \mathbf{S}^i \cup \mathbf{D}_s^i$  : the selected.

    [4] Compute the next evaluation set  $\mathbf{D}_e^{i+1}$ .
         $\mathbf{D}_e^{i+1} \leftarrow \bigcup_{\vec{x} \in \mathbf{D}_o^i} k\text{NN}(\vec{x}) - (\mathbf{S}_o^{i+1} \cup \mathbf{S}_x^{i+1})$ .

    [5]  $i \leftarrow i + 1$ .
  end
  return  $\mathbf{S}^i$ 
}

```

Figure 5.2: *Fast* NPPS

Table 5.1: Notation

Symbol	Meaning
\mathbf{D}	the original training set whose cardinality is M
\mathbf{D}_e^i	the evaluation set at i^{th} step
\mathbf{D}_o^i	a subset of \mathbf{D}_e^i , the set of patterns to be “expanded” from \mathbf{D}_e^i each element of which will compute its k nearest neighbors to constitute the next evaluation set, \mathbf{D}_e^{i+1}
\mathbf{D}_x^i	a subset of \mathbf{D}_e^i , the set of patterns “not to be expanded” from \mathbf{D}_e^i , or $\mathbf{D}_x^i = \mathbf{D}_e^i - \mathbf{D}_o^i$
\mathbf{D}_s^i	the set of “selected” patterns from \mathbf{D}_o^i at i^{th} step
\mathbf{S}_o^i	the accumulated set of expanded patterns, $\bigcup_{j=0}^{i-1} \mathbf{D}_o^j$
\mathbf{S}_x^i	the accumulated set of non-expanded patterns, $\bigcup_{j=0}^{i-1} \mathbf{D}_x^j$
\mathbf{S}^i	the accumulated set of selected patterns, $\bigcup_{j=0}^{i-1} \mathbf{D}_s^j$
	the last of which \mathbf{S}^N is the reduced training pattern set
$k\text{NN}(\vec{x})$	the set of k nearest neighbors of \vec{x}
\mathbf{B}	the set of patterns located in the “overlap” region characterized by $Neighbors_Entropy(\vec{x}, k) > 0$
\mathbf{B}^+	the set of k nearest neighbors of patterns belonging to \mathbf{B}

Lemma 5.1 *Different evaluation sets are disjoint:*

$$\mathbf{D}_e^i \cap \mathbf{D}_e^j = \emptyset, \forall i \neq j. \quad (5.1)$$

PROOF. Consider step [4] of the algorithm shown in Fig. 5.2,

$$\mathbf{D}_e^i = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^{i-1}} k\text{NN}(\vec{x}) \right) - (\mathbf{S}_o^i \cup \mathbf{S}_x^i). \quad (5.2)$$

Since \mathbf{S}_o^i and \mathbf{S}_x^i are defined as $\left(\bigcup_{j=0}^{i-1} \mathbf{D}_o^j \right)$ and $\left(\bigcup_{j=0}^{i-1} \mathbf{D}_x^j \right)$ respectively, their union results in

$$\mathbf{S}_o^i \cup \mathbf{S}_x^i = \bigcup_{j=0}^{i-1} (\mathbf{D}_o^j \cup \mathbf{D}_x^j) = \left(\bigcup_{j=0}^{i-1} \mathbf{D}_e^j \right). \quad (5.3)$$

By replacing $(\mathbf{S}_o^i \cup \mathbf{S}_x^i)$ in Eq.(5.2) with Eq.(5.3), we get

$$\mathbf{D}_e^i = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^{i-1}} k\text{NN}(\vec{x}) \right) - \left(\bigcup_{j=0}^{i-1} \mathbf{D}_e^j \right). \quad (5.4)$$

Eq.(5.4) clearly shows that \mathbf{D}_e^i does not share patterns with any of its earlier sets \mathbf{D}_e^j , $j = 0, \dots, i-1$.

Lemma 5.2 *The union of all \mathbf{D}_e^i 's is equivalent to the set of $k\text{NN}$'s of the union of all \mathbf{D}_o^i 's.*

$$\left(\bigcup_{i=1}^n \mathbf{D}_e^i \right) = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^0 \cup \mathbf{D}_o^1 \cup \dots \cup \mathbf{D}_o^{n-1}} k\text{NN}(\vec{x}) \right). \quad (5.5)$$

PROOF. From Eq.(5.4) in Lemma 5.1, we get

$$\bigcup_{i=1}^n \mathbf{D}_e^i = \bigcup_{i=1}^n \left(\bigcup_{\vec{x} \in \mathbf{D}_o^{i-1}} k\text{NN}(\vec{x}) \right) - \bigcup_{i=1}^n \left(\bigcup_{j=0}^{i-1} \mathbf{D}_e^j \right). \quad (5.6)$$

Since in general

$$\left(\bigcup_{\vec{x} \in \mathbf{A}_1} k\text{NN}(\vec{x}) \right) \cup \left(\bigcup_{\vec{x} \in \mathbf{A}_2} k\text{NN}(\vec{x}) \right) = \left(\bigcup_{\vec{x} \in \mathbf{A}_1 \cup \mathbf{A}_2} k\text{NN}(\vec{x}) \right) \quad (5.7)$$

holds, we get

$$\left(\bigcup_{i=1}^n \mathbf{D}_e^i \right) = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^0 \cup \mathbf{D}_o^1 \cup \dots \cup \mathbf{D}_o^{n-1}} k\text{NN}(\vec{x}) \right) - \left(\bigcup_{i=0}^{n-1} \mathbf{D}_e^i \right). \quad (5.8)$$

Eq.(5.8) can be rewritten in the form of

$$\left(\bigcup_{i=1}^n \mathbf{D}_e^i \right) = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^0 \cup \mathbf{D}_o^1 \cup \dots \cup \mathbf{D}_o^{n-1}} k\text{NN}(\vec{x}) \right) \cap \left(\bigcup_{i=0}^{n-1} \mathbf{D}_e^i \right)^C. \quad (5.9)$$

If we union $\left(\bigcup_{i=0}^{n-1} \mathbf{D}_e^i\right)$ to both sides of Eq.(5.9), then

$$\left(\bigcup_{i=0}^n \mathbf{D}_e^i\right) = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^0 \cup \mathbf{D}_o^1 \cup \dots \cup \mathbf{D}_o^{n-1}} k\text{NN}(\vec{x})\right) \cup \left(\bigcup_{i=0}^{n-1} \mathbf{D}_e^i\right) \quad (5.10)$$

results. Since $\mathbf{D}_e^i \subseteq \bigcup_{\vec{x} \in \mathbf{D}_o^{i-1}} k\text{NN}(\vec{x})$, $i = 1, \dots, n$, $\left(\bigcup_{i=1}^{n-1} \mathbf{D}_e^i\right)$, the last $n - 1$ components of the second factor of the right hand side may vanish. Then, we finally have

$$\left(\bigcup_{i=1}^n \mathbf{D}_e^i\right) \cup \mathbf{D}_e^0 = \left(\bigcup_{\vec{x} \in \mathbf{D}_o^0 \cup \mathbf{D}_o^1 \cup \dots \cup \mathbf{D}_o^{n-1}} k\text{NN}(\vec{x})\right) \cup \mathbf{D}_e^0. \quad (5.11)$$

If we consider only the relationship after the first iteration then \mathbf{D}_e^0 from both sides of Eq.(5.11) is not to be included. Now, the lemma is proved.

Lemma 5.3 *Every expanded set \mathbf{D}_o^i is a subset of \mathbf{B} , the set of patterns in the overlap region.*

$$\mathbf{D}_o^i \subseteq \mathbf{B}, \forall i \quad (5.12)$$

PROOF. Recall that in the proposed algorithm, \mathbf{D}_o^i is defined as

$$\mathbf{D}_o^i = \{\vec{x} \mid \text{Neighbors_Entropy}(\vec{x}, k) > 0, \vec{x} \in \mathbf{D}_e^i\}. \quad (5.13)$$

Compare it with the definition of \mathbf{B}

$$\mathbf{B} = \{\vec{x} \mid \text{Neighbors_Entropy}(\vec{x}, k) > 0, \vec{x} \in \mathbf{D}\}. \quad (5.14)$$

Since \mathbf{D}_e^i 's are subsets of \mathbf{D} , \mathbf{D}_o^i 's are subsets of \mathbf{B} .

Lemma 5.4 *Different expanded sets \mathbf{D}_o^i 's are disjoint.*

$$\mathbf{D}_o^i \cap \mathbf{D}_o^j = \emptyset, \forall i \neq j \quad (5.15)$$

PROOF. Every expanded set is a subset of the evaluation set by definition (see step[1] in *fast* NPPS, Fig. 5.2)

$$\mathbf{D}_o^i \subseteq \mathbf{D}_e^i, \forall i. \quad (5.16)$$

By Lemma 5.1, \mathbf{D}_e^i 's are disjoint from others for all i 's. Therefore, their respective subsets are disjoint, too.

Theorem 5.5 Termination of the Algorithm

If the while loop of the proposed algorithm exits after N iterations, then N is finite.

PROOF. We show that $N < \infty$. Inside the while-loop of the algorithm (Fig. 5.2), condition $\mathbf{D}_e^{i+1} \neq \emptyset$ holds. Therefore, $\mathbf{D}_o^i \neq \emptyset$, $i = 0, \dots, N - 1$. That means $n(\mathbf{D}_o^i) \geq 1$, $i = 0, \dots, N - 1$. Since \mathbf{S}_o^i is defined as $\bigcup_{j=0}^{i-1} \mathbf{D}_o^j$, and \mathbf{D}_o^j 's are disjoint (Lemma 5.4), we get

$$n(\mathbf{S}_o^i) = \sum_{j=0}^{i-1} n(\mathbf{D}_o^j). \quad (5.17)$$

Since $n(\mathbf{D}_o^i) \geq 1$, $i = 0, \dots, N - 1$, $n(\mathbf{S}_o^i)$ is monotonically increasing. In the meantime, the union of all the \mathbf{D}_o^j 's generated in the while loop is bounded by \mathbf{B} (Lemma 5.3). So, we obtain

$$\bigcup_{j=0}^{N-1} \mathbf{D}_o^j \subseteq \mathbf{B}. \quad (5.18)$$

Now, Lemma 5.4 leads us to

$$\sum_{j=0}^{N-1} n(\mathbf{D}_o^j) \leq n(\mathbf{B}). \quad (5.19)$$

Combination of Eq.(5.17) and Eq.(5.19) results in

$$n(\mathbf{S}_o^N) \leq n(\mathbf{B}). \quad (5.20)$$

Since $n(\mathbf{B}) \ll M$ and finite, $n(\mathbf{S}_o^N)$ is finite. Thus, N is finite.

Theorem 5.6 The Number of Pattern Evaluation

The number of patterns whose k NNs are evaluated is $(r \cdot n(\mathbf{B}^C) + n(\mathbf{B}^+))$, where \mathbf{B}^C is the complement set of \mathbf{B} or $\mathbf{D} - \mathbf{B}$, and r is the proportion of initial random sampling, ($0 < r < 1$).

PROOF. The number of patterns whose k NNs are evaluated is denoted as $\sum_{i=0}^N n(\mathbf{D}_e^i)$. Let us first consider cases from $i = 1$ to N . We have

$$\begin{aligned} \sum_{i=1}^N n(\mathbf{D}_e^i) &= n\left(\bigcup_{i=1}^N \mathbf{D}_e^i\right) \quad \text{by Lemma 5.1} \\ &= n\left(\bigcup_{\vec{x} \in \mathbf{D}_o^0 \cup \mathbf{D}_o^1 \cup \dots \cup \mathbf{D}_o^{N-1}} k\text{NN}(\vec{x})\right) \quad \text{by Lemma 5.2(5.21)} \\ &\leq n\left(\bigcup_{\vec{x} \in \mathbf{B}} k\text{NN}(\vec{x})\right) \quad \text{by Lemma 5.3} \\ &= n(\mathbf{B}^+). \end{aligned}$$

Let us include the case of $i = 0$.

$$\sum_{i=0}^N n(\mathbf{D}_e^i) \leq n(\mathbf{D}_e^0) + n(\mathbf{B}^+), \quad (5.22)$$

where $n(\mathbf{D}_e^0)$ is approximately $r \cdot n(\mathbf{D})$ because \mathbf{D}_e^0 is randomly chosen from \mathbf{D} . In the meantime, some patterns of \mathbf{D}_e^0 are already counted in $n(\mathbf{B}^+)$. The number of those pattern amounts to $n(\mathbf{D}_o^0)$ since $\mathbf{D}_o^0 = \{\vec{x} \mid \text{Neighbors_Entropy}(\vec{x}, k) > 0, \vec{x} \in \mathbf{D}_e^0\}$ and $\mathbf{D}_o^0 \subseteq \mathbf{B} \subseteq \mathbf{B}^+$. To get a tighter bound, therefore, we

exclude the duplicated count from Eq.(5.22):

$$\sum_{i=0}^N n(\mathbf{D}_e^i) \leq n(\mathbf{D}_e^0 - \mathbf{D}_o^0) + n(\mathbf{B}^+), \quad (5.23)$$

where $n(\mathbf{D}_e^0 - \mathbf{D}_o^0)$ denotes the number of the patterns which do not belong to \mathbf{B} . It amounts

$$\begin{aligned} n(\mathbf{D}_e^0 - \mathbf{D}_o^0) &= n(\mathbf{D}_e^0) - n(\mathbf{D}_o^0) \\ &\approx n(\mathbf{D}_e^0) - n(\mathbf{D}_e^0) \cdot \frac{n(\mathbf{B})}{n(\mathbf{D})} \\ &= r \cdot n(\mathbf{D}) - r \cdot n(\mathbf{D}) \cdot \frac{n(\mathbf{B})}{n(\mathbf{D})} \\ &= r \cdot n(\mathbf{B}^C), \end{aligned} \quad (5.24)$$

where $\mathbf{D}_o^0 \subseteq \mathbf{D}_e^0$. Thus, we get the following bound by Eq.(5.21) and Eq.(5.24):

$$\sum_{i=0}^N n(\mathbf{D}_e^i) \leq r \cdot n(\mathbf{B}^C) + n(\mathbf{B}^+). \quad (5.25)$$

The time complexity of *fast* NPPS is $(r \cdot b^C + b^+) \cdot M$ where $b^C = n(\mathbf{B}^C)$ and $b^+ = n(\mathbf{B}^+)$. Practically, b^C is almost as large as M , i.e., $b^C \approx M$. But the initial sampling ratio r is usually quite small, i.e., $r \ll 1$. Thus, the first term $r \cdot b^C M$ may be considered insignificant. In most real world problems, b^+ is just slightly larger than b . Thus, the second term $b^+ M$ can be approximated to bM . In short, $(r \cdot b^C + b^+)M$ can be simplified as bM , which is much smaller than M^2 since $b \ll M$. Actually, b is an increasing function of k by definition. But, provided that we get informed about the number of patterns in the overlap region, v , then we can find a right value of k which makes $b \approx v$. Since b approximates v , the complexity of *fast* NPPS is approximately $O(vM)$. The relationship between k , b , and v will be dealt with in the next chapter in detail.

Fig. 5.3 depicts the theoretical relationship between the computation time and v of the two algorithms. The computation time of *naive* NPPS $O(M^2)$ does not change as long as M is fixed. Meanwhile, that of *fast* NPPS is linearly proportional to v . Therefore, *fast* NPPS is always faster than *naive* NPPS except in the case of $v = M$.

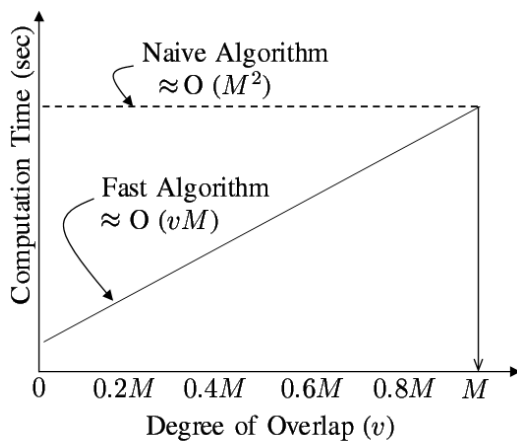


Figure 5.3: Theoretical relationship between the computation time and v : the computation time of *fast* NPPS is linearly proportional to v when M is fixed.

How to Determine the Number of Neighbors

In the previous chapter, we briefly mentioned on the effect of k , in particular, on the time complexity of the *fast* NPPS. But also in *naive* NPPS, the value of k is a parameter to be set. The value of k commonly affects both algorithms in the following manner. Too large a value of k results in too many patterns being selected. Hence no effect on the training pattern reduction is achieved. Too small a value of k , on the other hand, may degrade the SVM prediction accuracy since too many would-be support vectors are not selected. Therefore, we present a procedure for determining the suitable value of k here. We first identify a subset of the training pattern set \mathbf{D} that matches the overlap region \mathbf{R} as closely as possible. Definitions are provided for the classifier $f(\vec{x})$, training pattern set \mathbf{D} , overlap pattern set \mathbf{V} , overlap region \mathbf{R} , as well as k nearest neighbors' pattern set \mathbf{B}_k , the \mathbf{B} now specified by k . Second, some properties of \mathbf{B}_k as well as the proposed procedure are presented. Finally, an estimate of the cardinality of \mathbf{V} is presented. The value of k is determined from the estimated cardinality of \mathbf{V} .

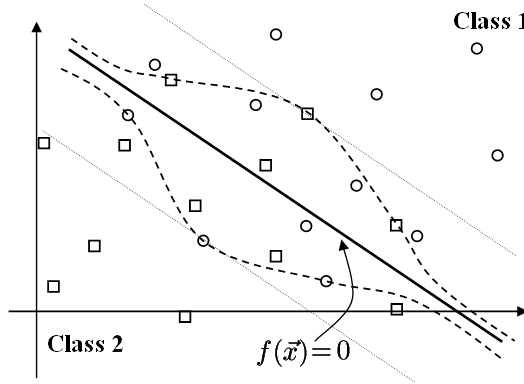


Figure 6.1: Two class classification problem where circles belong to class 1 while squares belong to class 2. The area enclosed by the two dotted lines comprise the overlap area.

6.1 Overlap Region and Overlap Set

Consider a two-class classification problem whose classes are C_1 and C_2 (see Fig. 6.1), with classifier $f(\vec{x})$ such that

$$f(\vec{x}) = \begin{cases} \vec{x} \rightarrow C_1 & \text{if } f(\vec{x}) > 0, \\ \vec{x} \rightarrow C_2 & \text{if } f(\vec{x}) < 0, \end{cases} \quad (6.1)$$

where $f(\vec{x}) = 0$ is its decision boundary. Let \mathbf{D} denote the set of training patterns. Let us define “overlap patterns” as the patterns that are located in the “other” side of the decision boundary since the class distributions overlap. For simplicity, genuine noisy patterns will be considered as overlap patterns. They are shown in Fig. 6.1 as squares located above $f(\vec{x}) = 0$ and circles located below $f(\vec{x}) = 0$. Let \mathbf{R} denote a hypothetical region where the overlap patterns reside, the area enclosed by the dotted lines in Fig. 6.1. Note that \mathbf{R} contains not only the overlap patterns, but also the “close non-overlap” patterns—those patterns that are located close to the decision boundary, yet in the “right” side of the decision boundary. Let \mathbf{V} denote the intersection of \mathbf{D} and \mathbf{R} , i. e. the subset of \mathbf{D} which comprises overlap patterns and close non-overlap patterns. There are six patterns in class 1 side and another six patterns in class 2 side in Fig. 6.1. The cardinality of \mathbf{V} is denoted as v .

6.2 Pattern Set with Positive Nearest Entropy

Now, let \mathbf{B}_k denote a subset of \mathbf{D} whose elements have positive k nearest neighbors' entropy values (see Fig. 4.1 and Table 5.1):

$$\mathbf{B}_k = \{\vec{x} \mid \text{Neighbors_Entropy}(\vec{x}, k) > 0, \vec{x} \in \mathbf{D}\}. \quad (6.2)$$

Let us consider how k affects \mathbf{B}_k and pattern selection based on it. Too large a value of k results in *excessive inclusion* of the training patterns. In other words, too many patterns are selected. If $k = M - 1$, then \mathbf{B}_k becomes \mathbf{D} . Suppose that pattern \vec{x} belongs to C_1 . Then its LabelProbability(\vec{x}, k) is

$$P_1 = \frac{m_1 - 1}{m_1 + m_2 - 1},$$

$$P_2 = \frac{m_2}{m_1 + m_2 - 1},$$

where m_j denotes the number of patterns belonging to C_j , ($j = 1, 2$). Thus, we have $P_j < 1$ for all j 's. If pattern \vec{x} belongs to C_2 , both P_1 and P_2 are less than 1. It should be noted that Neighbors_Entropy(\vec{x}, k) in Fig. 4.1 is always positive unless j exists such that $P_j = 1$. Therefore, all the patterns in training set \mathbf{D} have positive Neighbors_Entropy values, regardless of their location in the input space, thus become a member of \mathbf{B}_{M-1} . Every pattern from \mathbf{D} is selected for training (see Fig. 6.2.a). Too small a value of k , e. g. $k = 2$, on the other hand, results in *insufficient inclusion* of the patterns within the overlap region. Consider patterns \vec{x}^1 , \vec{x}^2 , and \vec{x}^3 in Fig. 6.2.b, lying within the overlap region. They all belong to overlap pattern set \mathbf{V} , but \vec{x}^2 and \vec{x}^3 do not belong to \mathbf{B}_2 while \vec{x}^1 does. First, \vec{x}^1 belongs to \mathbf{B}_2 since its two nearest neighbors \vec{x}^2 and \vec{x}^3 belong to different classes, which results in $P_1 = P_2 = 1/2$ and Neighbors_Entropy(\vec{x}^1, k) becomes 1. Second, the two nearest neighbors of \vec{x}^2 , \vec{x}^1 and \vec{x}^4 , both belong to class C_1 , which results in $P_1 = 1$, $P_2 = 0$, and Neighbors_Entropy(\vec{x}^2, k) is 0. So, \vec{x}^2 does not belong to \mathbf{B}_2 . Third, for the same reason, \vec{x}^3 does not belong to \mathbf{B}_2 , either. The patterns in the overlap region are critical to SVM training, since they are likely to be support vectors. Therefore, their exclusion could degrade the SVM prediction accuracy.

In short, \mathbf{B}_k larger than \mathbf{V} merely increases the SVM training time by introducing redundant training patterns. In contrast, \mathbf{B}_k smaller than \mathbf{V} could degrade the SVM accuracy. Therefore, our objective is to find the smallest \mathbf{B}_k that covers \mathbf{V} . The following property of \mathbf{B}_k results in a simple procedure.

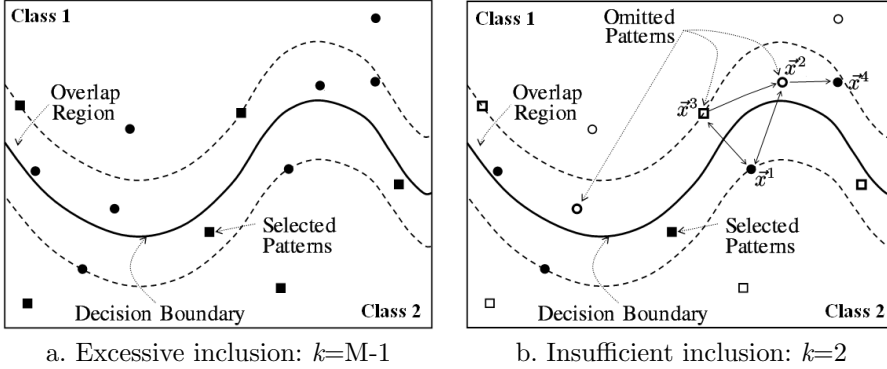


Figure 6.2: Effect of k on \mathbf{B} : solid dots and squares belong to \mathbf{B} .

Lemma 6.1 Every k^{th} entropy pattern set belongs to $(k+1)^{\text{th}}$ entropy pattern set:

$$\mathbf{B}_k \subseteq \mathbf{B}_{k+1} \text{ for } k = 2, \dots, M-2. \quad (6.3)$$

PROOF. Denote P_j^k as the probability that k_j out of k nearest neighbors belong to class C_j . If $\vec{x} \in \mathbf{B}_k$, then it means $\text{Neighbors_Entropy}(\vec{x}, k) > 0$. A positive Neighbors_Entropy is always accompanied with $P_j^k = \frac{k_j}{k} < 1, \forall j$. Therefore,

$$k_j < k, \forall j. \quad (6.4)$$

Adding 1 to both sides yields

$$(k_j + 1) < (k + 1), \forall j. \quad (6.5)$$

Suppose $(k+1)^{\text{th}}$ nearest neighbor belongs to C_{j^*} . Then, for j^* , $k_{j^*} + 1 < k + 1$ holds while for $j \neq j^*$, $k_j < k + 1$ holds. Therefore, both $P_{j^*}^{k+1} < 1$ and $P_j^{k+1} < 1, \forall j \neq j^*$. We have $\text{Neighbors_Entropy}(\vec{x}, k+1) > 0$ which indicates $\vec{x} \in \mathbf{B}_{k+1}$. From Lemma 6.1, it follows that b_k , the cardinality of \mathbf{B}_k , is an increasing function of k . Thus, optimal k , k^* , is computed as

$$k^* = \min\{k \mid b_k \geq v, k = 2, \dots, M-1\}. \quad (6.6)$$

6.3 Estimation of the Size of Overlap Set

Now, we need to estimate v . Let us assume that each training pattern is independently sampled from a training data distribution. Then, the probability that v patterns of M training patterns fall within region \mathbf{R} is given by the binomial law,

$$Pr(v) = \frac{M!}{v!(M-v)!} \left(P_{\mathbf{R}}(\vec{x}) \right)^v \left(1 - P_{\mathbf{R}}(\vec{x}) \right)^{M-v}, \quad (6.7)$$

where $P_{\mathbf{R}}(\vec{x})$ denotes the probability that a pattern \vec{x} lies in \mathbf{R} . We now can calculate v as

$$v = MP_{\mathbf{R}}(\vec{x}). \quad (6.8)$$

Here $P_{\mathbf{R}}(\vec{x})$ of \vec{x} can be described as

$$P_{\mathbf{R}}(\vec{x}) = \sum_{j=1}^2 P(\vec{x} \in \mathbf{R}, C_j), \quad (6.9)$$

where $P(\vec{x} \in \mathbf{R}, C_j)$ is the joint probability of \vec{x} belonging to class C_j and lying in \mathbf{R} . Fig. 6.3.a illustrates the schematic probability densities in the overlap region \mathbf{R} . Note that \mathbf{R} contains correct patterns as well as overlap patterns.

We divide region \mathbf{R} into \mathbf{R}_1 and \mathbf{R}_2 as follows (see also Fig. 6.3.b):

$$\begin{aligned} \mathbf{R}_1 &= \{\vec{x} \in \mathbf{R} \mid f(\vec{x}) \geq 0\}, \\ \mathbf{R}_2 &= \{\vec{x} \in \mathbf{R} \mid f(\vec{x}) < 0\}. \end{aligned} \quad (6.10)$$

Eq. (6.9) can be rewritten as

$$\begin{aligned} P_{\mathbf{R}}(\vec{x}) &= P(\vec{x} \in \mathbf{R}, C_1) + P(\vec{x} \in \mathbf{R}, C_2) \\ &= P(\vec{x} \in \mathbf{R}_1 \cup \mathbf{R}_2, C_1) + P(\vec{x} \in \mathbf{R}_1 \cup \mathbf{R}_2, C_2) \\ &= \left(P(\vec{x} \in \mathbf{R}_1, C_2) + P(\vec{x} \in \mathbf{R}_2, C_1) \right) \\ &\quad + \left(P(\vec{x} \in \mathbf{R}_1, C_1) + P(\vec{x} \in \mathbf{R}_2, C_2) \right). \end{aligned} \quad (6.11)$$

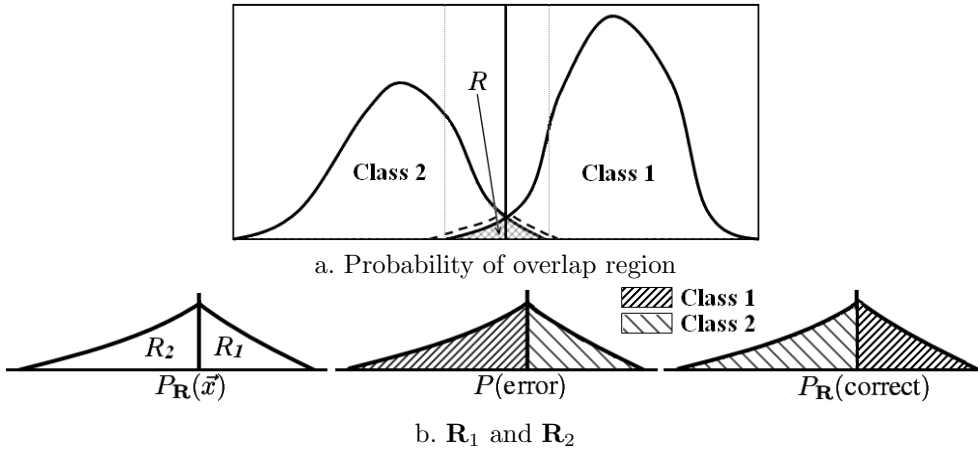


Figure 6.3: Schematic probability densities of overlap region.

The first parenthesis and second parenthesis denote the probabilities that the patterns located in \mathbf{R} are incorrectly and correctly classified, respectively. If \mathbf{R}_1 and \mathbf{R}_2 contain roughly the same number of correct and incorrect patterns, the probabilities of the two parentheses become identical. Since all the overlap patterns were included in \mathbf{R} , the first parenthesis actually refers to the misclassified error of classifier $f(\vec{x})$. Now, Eq. (6.11) can be simplified as

$$P_{\mathbf{R}}(\vec{x}) = 2P(\text{error}), \quad (6.12)$$

and Eq. (6.8) becomes

$$v = 2MP(\text{error}). \quad (6.13)$$

6.4 Procedure to Determine the Number of Neighbors

Now, the procedure for determining the optimal k value is shown in Fig. 6.4. Simplicity and computational efficiency are the reasons for using 1 -NN rule to estimate $P(\text{error})$.

- [1] Apply 1-NN rule over training set \mathbf{D} .
- [2] Estimate $P(\text{error})$ with $\hat{P}(\text{error})$, the training error rate of [1].
- [3] Calculate \hat{v} according to Eq. (6.13):
$$\hat{v} = 2M\hat{P}(\text{error}).$$
- [4] Find k^* according to Eq. (6.6):
$$k^* = \min\{k \mid b_k \geq \hat{v}, k = 2, \dots, M - 1\}.$$

Figure 6.4: Procedure to determine the value of k

Invariance of Neighborhood Relation under Input Space to Feature Space Mapping

NPPS described in the previous chapters operates in *input space* while decision boundary of SVM and support vectors are all defined in *feature space*. Since the mapping from input space to feature space is highly nonlinear and dimension expanding, distortion of neighborhood relation could occur. In other words, neighborhood relation in input space may not be preserved in feature space. If that is the case, local information in input space may not be correct in feature space, thus impairing the effectiveness of NPPS.

There are two approaches to solve this problem. The *first* involves running NPPS in feature space, and the *second* involves proving that the neighborhood relation is invariant under the input to feature space mapping. Let us consider the first approach. In order to compute the distance between two patterns, one has to have the optimal kernel function and hyper-parameters, which are usually found by trial-and-error accompanying with multiple trials of SVM training with all patterns. Obviously, that is not acceptable since the purpose of pattern selection is to avoid training SVM with all patterns. On the other hand, in the second approach, NPPS can be executed only once in input space since it does not involve searching for optimal kernel and hyper-parameters. Thus, we take the

second approach in this paper by showing that the neighborhood relation is invariant under the input to feature space mapping. Through the theoretical justification for the second approach, we can avoid the computational burden which could be incurred by taking the first approach.

The chapter is organized as follows. In section 7.1, we provide proofs on the invariance of the neighborhood relation under the input to feature space mapping through two typical kernel functions: RBF and polynomial. In section 7.2, we remark some issues on the proofs.

7.1 Proofs on Validity of Pattern Selection in Input Space

As described in previous chapters, NPPS operates in *input space* (I) using local information there. However, decision boundary of SVM and support vectors are all defined in *feature space* (Φ). Since the mapping $I \mapsto \Phi$ is highly nonlinear as well as dimension expanding, we have to ensure that neighborhood relation in input space be preserved in feature space. We now provide proofs on that the k nearest neighbors of a pattern in the input space I are also the k nearest neighbors of the pattern in the feature space Φ .

Definition 7.1 (k NN Invariance) Let $kNN_I(\vec{x})$ be the set of k nearest neighbors of a pattern \vec{x} in the input space I , and $kNN_\Phi(\vec{x})$ be that of the pattern $\Phi(\vec{x})$ in the feature space Φ . If both sets are identical

$$kNN_I(\vec{x}) = kNN_\Phi(\vec{x}), \quad \forall k > 0, \forall \vec{x}$$

the invariance of the k nearest neighbors (NN) holds.

Finding the nearest neighbors necessarily implies distance calculation. In terms of the squared Euclidean distance which is the most commonly used distance measure, the distance among patterns in the input space I is

$$||\vec{x} - \vec{y}||^2 = \vec{x} \cdot \vec{x} + \vec{y} \cdot \vec{y} - 2\vec{x} \cdot \vec{y}. \tag{7.1}$$

The distance in the feature space Φ is similarly drawn as

$$\|\Phi(\vec{x}) - \Phi(\vec{y})\|^2 = \Phi(\vec{x}) \cdot \Phi(\vec{x}) + \Phi(\vec{y}) \cdot \Phi(\vec{y}) - 2\Phi(\vec{x}) \cdot \Phi(\vec{y}) \quad (7.2)$$

where $\Phi(\cdot)$ is a mapping function from the input space to the feature space, $\Phi(\cdot) : I \mapsto \Phi$. One might obtain $\Phi(\vec{x})$ directly but the formula is extremely complicated. Thanks to the fact that the mapping $\Phi(\cdot)$ always appears within a form of inner product during SVM QP calculation, one thus uses *kernel trick* which substitutes the inner product to a kernel function, $\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x}, \vec{y})$. If this kernel trick is applied to Eq.(7.2), then the distance in the feature space becomes

$$\|\Phi(\vec{x}) - \Phi(\vec{y})\|^2 = K(\vec{x}, \vec{x}) + K(\vec{y}, \vec{y}) - 2K(\vec{x}, \vec{y}). \quad (7.3)$$

We will consider the following typical kernel functions:

$$\begin{aligned} \text{RBF: } K(\vec{x}, \vec{y}) &= \exp\left(-\frac{\|\vec{x} - \vec{y}\|^2}{2\sigma^2}\right), \\ \text{polynomial: } K(\vec{x}, \vec{y}) &= (\vec{x} \cdot \vec{y} + 1)^p. \end{aligned} \quad (7.4)$$

As long as the relative distance magnitude of the input space is preserved in the feature space Φ for all patterns, the composition of the k nearest neighbors of a pattern will be invariant. We now define *proximity invariance*.

Definition 7.2 (Proximity Invariance) For the patterns \vec{x}, \vec{y}_1 and \vec{y}_2 ($\vec{x} \neq \vec{y}_1$, $\vec{x} \neq \vec{y}_2$, and $\vec{y}_1 \neq \vec{y}_2$) in the input space I satisfying

$$\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2,$$

the invariance of proximity holds if they preserve their relative distances in the feature space Φ

$$\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 < \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2.$$

It is obvious that *kNN invariance* holds if *proximity invariance* holds. The following two theorems provide proofs on *kNN invariance* for the two kernels, RBF and polynomial, by inducing proximity invariance for each of them.

Theorem 7.3 (kNN Invariance for RBF Kernel) *kNN invariance holds when the mapping function $\Phi(\vec{x})$ is defined such that*

$$\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x}, \vec{y}) = \exp\left(-\frac{\|\vec{x} - \vec{y}\|^2}{2\sigma^2}\right).$$

PROOF. Let \vec{y}_1 and \vec{y}_2 be two distinct neighbors of \vec{x} with $\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2$, i.e., \vec{y}_1 is closer to \vec{x} than \vec{y}_2 . Suppose the invariance of proximity does not hold for mapping function $\Phi(\vec{x})$, i.e., $\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 \geq \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2$. Using Eq.(7.3), one can rewrite the inequality as

$$K(\vec{x}, \vec{x}) + K(\vec{y}_1, \vec{y}_1) - 2K(\vec{x}, \vec{y}_1) \geq K(\vec{x}, \vec{x}) + K(\vec{y}_2, \vec{y}_2) - 2K(\vec{x}, \vec{y}_2).$$

Since $K(\vec{a}, \vec{a})=1$ and $K(\vec{a}, \vec{b}) > 0 \quad \forall \vec{a}, \vec{b}$, the inequality is simplified as

$$K(\vec{x}, \vec{y}_1) \leq K(\vec{x}, \vec{y}_2).$$

Plugging the definition of RBF kernel, we obtain

$$\exp\left(-\frac{\|\vec{x} - \vec{y}_1\|^2}{2\sigma^2}\right) \leq \exp\left(-\frac{\|\vec{x} - \vec{y}_2\|^2}{2\sigma^2}\right),$$

which in turn can be simplified into

$$\|\vec{x} - \vec{y}_1\|^2 \geq \|\vec{x} - \vec{y}_2\|^2.$$

This is contradictory to our initial assumption that \vec{y}_1 is closer to \vec{x} than \vec{y}_2 . Thus the assumption that *the invariance of proximity does not hold* is not true. Therefore, kNN invariance holds for RBF kernel.

Theorem 7.4 (kNN Invariance for Polynomial Kernel) *kNN invariance holds when the mapping function $\Phi(\vec{x})$ is defined such that*

$$\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y} + 1)^p,$$

and training patterns are all norm vectors ($\|\cdot\| = 1$).

PROOF. Let \vec{y}_1 and \vec{y}_2 be two distinct neighbors of \vec{x} with $\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2$, i.e., \vec{y}_1 is closer to \vec{x} than \vec{y}_2 . Suppose the invariance of proximity does not hold for mapping function $\Phi(\vec{x})$, i.e., $\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 \geq \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2$. Using Eq.(7.3), one can rewrite the inequality as

$$K(\vec{x}, \vec{x}) + K(\vec{y}_1, \vec{y}_1) - 2K(\vec{x}, \vec{y}_1) \geq K(\vec{x}, \vec{x}) + K(\vec{y}_2, \vec{y}_2) - 2K(\vec{x}, \vec{y}_2).$$

Since $K(\vec{a}, \vec{a}) = 2^p$ from $\vec{a} \cdot \vec{a} = 1$, the inequality becomes

$$K(\vec{x}, \vec{y}_1) \leq K(\vec{x}, \vec{y}_2).$$

Plugging the definition of polynomial kernel, one obtains

$$(\vec{x} \cdot \vec{y}_1 + 1)^p \leq (\vec{x} \cdot \vec{y}_2 + 1)^p.$$

The polynomial degree p can be eliminated from both sides since norm vectors always satisfy $-1 < \vec{a} \cdot \vec{b} < 1$, $\forall \vec{a} \neq \vec{b}$, and $(\vec{a} \cdot \vec{b} + 1) > 0$. Therefore, we get

$$\vec{x} \cdot \vec{y}_1 \leq \vec{x} \cdot \vec{y}_2.$$

The inner product between the patterns can be represented

$$\|\vec{x}\| \|\vec{y}_1\| \cos \theta_1 \leq \|\vec{x}\| \|\vec{y}_2\| \cos \theta_2$$

where θ_1 and θ_2 denote the angles between \vec{x} and \vec{y}_1 , and between \vec{x} and \vec{y}_2 , respectively. Since $\|\vec{x}\| = \|\vec{y}_1\| = \|\vec{y}_2\| = 1$, finally one has

$$\cos \theta_1 \leq \cos \theta_2,$$

and hence $\theta_1 \geq \theta_2$. In other words,

$$\|\vec{x} - \vec{y}_1\|^2 \geq \|\vec{x} - \vec{y}_2\|^2,$$

which is contradictory to an initial assumption that \vec{y}_1 is closer to \vec{x} than \vec{y}_2 . Thus the assumption that *the invariance of proximity does not hold* is not true, which yields that k NN invariance holds for polynomial kernel.

Note that theorem 7.4 does not hold without the norm vector assumption ($\|\cdot\| = 1$). Consider three patterns, $\vec{x} = 1$, $\vec{y}_1 = 2$, and $\vec{y}_2 = -1$ in the input space (1-dim). They satisfy

$$\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2,$$

i.e., \vec{y}_1 is closer to \vec{x} than \vec{y}_2 . Let us consider $p = 2$ where $\Phi(a) \cdot \Phi(b) = K(\vec{a}, \vec{b}) = (\vec{a} \cdot \vec{b} + 1)^p$. The distance between \vec{x} and \vec{y}_1 in the feature space can be computed as

$$\begin{aligned} \|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 &= \|\Phi(1) - \Phi(2)\|^2 \\ &= K(1, 1) + K(2, 2) - 2K(1, 2) \\ &= (1 \cdot 1 + 1)^2 + (2 \cdot 2 + 1)^2 - 2(1 \cdot 2 + 1)^2 \\ &= 4 + 25 - 18 \\ &= 11. \end{aligned}$$

The distance between \vec{x} and \vec{y}_2 in the feature space can be similarly computed, resulting in 8. Thus, we have $\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 > \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2$, which violates the proximity invariance.

7.2 Remark

In case of polynomial kernel in section 7.1, we assumed that all patterns of training set are *norm vectors* ($\|\cdot\| = 1$), which could be somewhat restrictive. However, it should be noted that in some machine learning applications such as text mining, it is conventional to preprocess a training set by scaling $\vec{x}' = \vec{x}/\|\vec{x}\|$ so as to calculate a cosine distance $dist(\vec{x}, \vec{y}) = 1 - \vec{x}' \cdot \vec{y}'$.

Even though the proof should be similar in nature, we did not deal with sigmoid kernel here since it is a kernel with innate restrictions. Sigmoid kernel is a conditionally positive definite (CPD) kernel which satisfies Mercer's condition only for some values of the hyper-parameters ρ and δ , and only for norm vector (Schölkopf and Smola, 2002; Vapnik, 1999). Furthermore, sigmoid kernel has rarely been used since it has two hyper-parameters. It is a kernel devised for showing that SVM includes traditional MLP (multi-layer perceptron) neural network. A recent study of *Lin and Lin (2003)* on sigmoid kernel is worth reading. Even though sigmoid kernel seems to work well in a certain condition,

it is not better than RBF. As RBF has properties of being PD and having fewer parameters, there is no reason to use the sigmoid.

Applications with Small to Medium Scale Data sets

To confirm the efficiency of the proposed algorithm, experiments from different viewpoints were conducted section by section. In the first section, we show the empirical results of the time complexity analysis of *fast* NPPS, introduced in section 5.2. In the second section, using the three synthetic problems, we examined whether the proposed method to determine k in section 6 gives a reasonable estimation for v accompanied by SVM accuracies. The experiments in the third section emphasize how much time the proposed algorithms can reduce—not only for individual SVM training but also for SVM model selection. If we consider l candidate parameter combinations, then, the total time required to find out the best combination is l times the individual SVM training time. Individual SVM training time reduction, thus, leads to model selection time reduction. In the fourth section, we compared the proposed algorithm with a similar approach (Almeida *et al.*, 2000). In the last section, we present the results of three real world problems.

To preview table 8.1 would be helpful to perceive the overall empirical results before going into individual sections. ‘*’ stands for that the SVM training of corresponding problems was conducted with a standard QP solver, i.e., Gunn’s SVM MATLAB Toolbox. On the contrary, because of heavy memory burden and lengthy training time caused by large training set, others were trained with an iterative SVM solver known as one of the fastest solvers, i.e., OSU

SVM Classifier Toolbox (*Kernel-Machines.org*). All experiments were carried out on a Pentium 800 MHz PC with 256 MB memory. The table includes the results obtained from tests with three artificial datasets, and 5 real-world benchmarking datasets from (*UCI Repository*) and (*MNIST*). The results show that NPPS reduced SVM training time up to almost two orders of magnitude with virtually no loss of accuracy.

Table 8.1: Overall Results

	Num. of Training Patterns	Num. of Support Vectors	Execution Time (sec)	Test Error (%)
Continuous XOR *				
ALL	600	167	454.83	9.67
SELECTED	179	84	4.06 (=3.85 + 0.21)	9.67
Sine Function *				
ALL	500	250	267.76	13.33
SELECTED	264	136	8.96 (=8.79 + 0.17)	13.33
4x4 Checkerboard				
ALL	1000	172	3.81	4.03
SELECTED	275	148	0.41 (=0.09 + 0.32)	4.66
Pima Indian Diabetes				
ALL	615	330	203.91	29.90
SELECTED	311	216	28.00 (=27.86 + 0.14)	30.30
Wisconsin Breast Cancer				
ALL	546	87	2.14	6.80
SELECTED	96	41	0.13 (=0.03 + 0.10)	6.80
MNIST: 3-8				
ALL	11982	1253	477.25	0.50
SELECTED	4089	1024	147.73 (=49.84 + 97.89)	0.45
MNIST: 6-8				
ALL	11769	594	222.84	0.31
SELECTED	1135	421	58.96 (=14.69 + 44.27)	0.31
MNIST: 9-8				
ALL	11800	823	308.73	0.41
SELECTED	1997	631	86.23 (=26.61 + 59.62)	0.43

8.1 Time Complexity of *Fast* NPPS

We showed that *fast* NPPS runs in approximately $O(vM)$. In this section, an empirical time complexity is measured.

8.1.1 Uniform Distribution Problem

A total of M patterns, half from each class, were randomly generated from a pair of two-dimensional uniform distributions:

$$\begin{aligned} C_1 &= \left\{ \vec{x} \mid U \left(\left[\begin{array}{c} -1 \\ 0 - \frac{1}{2} \frac{v}{M} \end{array} \right] < \vec{x} < \left[\begin{array}{c} 1 \\ 1 - \frac{1}{2} \frac{v}{M} \end{array} \right] \right) \right\}, \\ C_2 &= \left\{ \vec{x} \mid U \left(\left[\begin{array}{c} -1 \\ -1 + \frac{1}{2} \frac{v}{M} \end{array} \right] < \vec{x} < \left[\begin{array}{c} 1 \\ 0 + \frac{1}{2} \frac{v}{M} \end{array} \right] \right) \right\}. \end{aligned} \quad (8.1)$$

We set v to every decile of M , i.e. $v = 0, 0.1M, 0.2M, \dots, 0.9M, M$. Fig. 8.1 shows the distributions of $v = 0.3M$ (Fig. 8.1.a) and $v = 0.8M$ (Fig. 8.1.b). The larger v values correspond to more overlap. We set out to see how v value determines the computation time of *fast* NPPS. In particular, it is of interest if the computation time is linearly proportional to v , as predicted in the analysis.

Fig. 8.2 shows the actual computation time for various values of v when (a) $M = 1,000$ and (b) $M = 10,000$, respectively. For both cases, it is clear that the computation time of *fast* NPPS is exactly proportional to v . On the other hand, that of *naive* NPPS is constant regardless of v . The reason why it is linearly proportional is because the number of evaluated (expanded) patterns increases proportional to v as shown in Fig. 8.3.

8.2 Procedure to Determine k

To show the validity of the procedure to determine k introduced in chapter 6, we ran experiments on three synthetic problems.

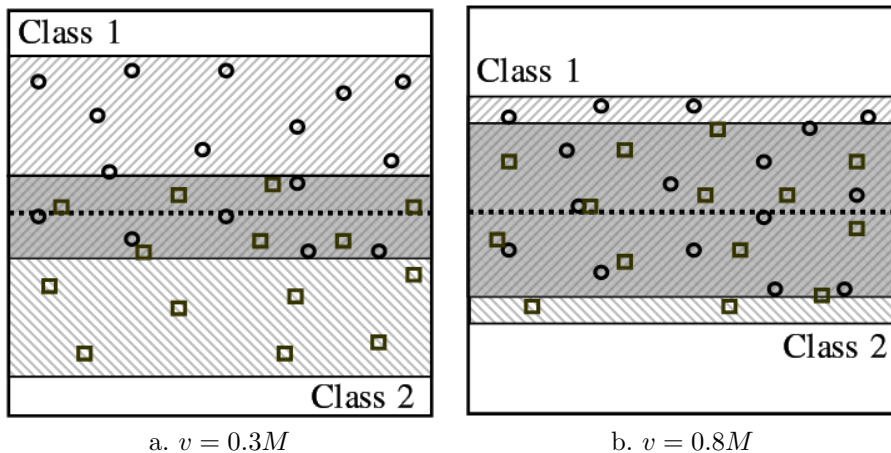


Figure 8.1: Two uniform distributions' overlap: the dark gray area is the overlap region which contains v patterns. The degree of overlap or the number of patterns in the overlap region, v , is set to every decile of training set size, M . (a) and (b) depicts when $v = 0.3M$ and $v = 0.8M$, respectively.

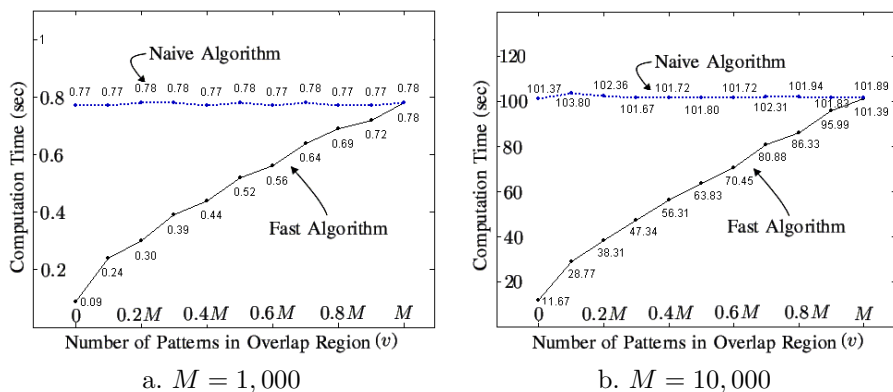
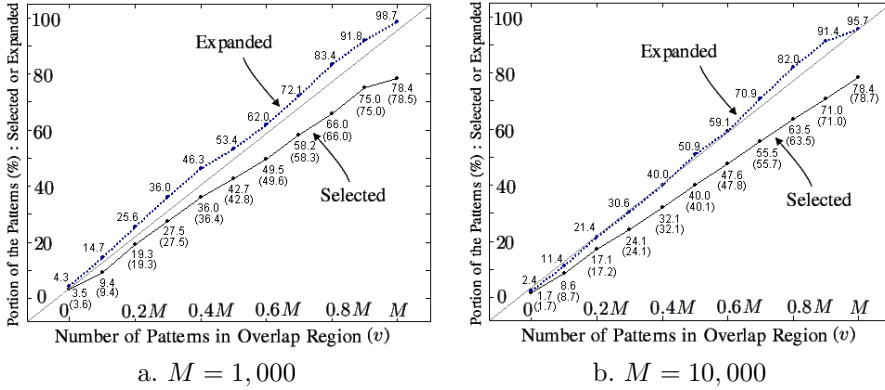


Figure 8.2: Actual computation time for various values of v

Figure 8.3: Actual percentage of expanded/selected patterns for various values of v Table 8.2: Estimation of v for various overlap degrees

v	100	200	300	400	500	600	700	800	900	1000
\hat{v}	112	202	334	414	512	602	706	806	882	942
b_k^*	115	209	339	429	518	606	708	816	906	972
k^*	5	4	5	4	5	5	5	5	5	5

8.2.1 Uniform Distribution Problem

First, we examined whether the proposed method gave a reasonably accurate estimation for v , using Uniform distributions' overlap with $M = 1,000$ (see Eq. (8.1)). 10 training pattern sets corresponding to 10 different numbers of overlap patterns were generated, i.e. $v = 100, 200, \dots, 900, 1000$ ($0.1M, 0.2M, \dots, 0.9M, M$). Table 8.2 provides the estimation results for various values of v . The second row shows the estimated values of \hat{v} . They are almost identical to the true values of v . The proposed method gave a reasonably accurate estimation of v . The last two rows show the smallest b_k larger than \hat{v} , and the corresponding value of k . Approximately, $k = 5$ seems to cover the overlap region regardless of the different degrees of overlap. The optimal value of k is likely to be dependent on the underlying distribution rather than the degree of the overlap itself.

8.2.2 Continuous XOR problem

The second experiment was on a Continuous XOR problem. The patterns of two classes were defined as follows (see also Fig. 8.4.a):

$$\begin{aligned} C_1 &= \left\{ \vec{x} \mid \vec{x} \in C_{1A} \cup C_{1B}, \begin{bmatrix} -3 \\ -3 \end{bmatrix} \leq \vec{x} \leq \begin{bmatrix} 3 \\ 3 \end{bmatrix} \right\}, \\ C_2 &= \left\{ \vec{x} \mid \vec{x} \in C_{2A} \cup C_{2B}, \begin{bmatrix} -3 \\ -3 \end{bmatrix} \leq \vec{x} \leq \begin{bmatrix} 3 \\ 3 \end{bmatrix} \right\} \end{aligned} \quad (8.2)$$

where C_{1A} , C_{1B} , C_{2A} and C_{2B} were

$$\begin{aligned} C_{1A} &= \left\{ \vec{x} \mid N \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \right) \right\}, \\ C_{1B} &= \left\{ \vec{x} \mid N \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \right) \right\}, \\ C_{2A} &= \left\{ \vec{x} \mid N \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \right) \right\}, \\ C_{2B} &= \left\{ \vec{x} \mid N \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \right) \right\}. \end{aligned} \quad (8.3)$$

A total of 600 training patterns, 300 from each class, were generated: There are about 33% training patterns in the overlap region ($v = 199$). A total of 1000 test patterns were generated from the statistically identical distributions to its training data distributions.

The proposed method estimated v as 208 ($\hat{v}=208$). And the value of k was set as 5 since b_5 was the minimum over 208 ($k^*=5$ and $b_{k^*}=217$). See Fig. 8.5.a. 179 patterns were selected after about 217 patterns were evaluated, when $k = 5$ (see Fig. 8.4.b). In order to test whether the selected pattern set taken from B_{k^*} would give rise to a reasonable SVM performance, we generated 29 selected pattern sets corresponding to $k = 2, \dots, 30$, and then we computed the SVM test error rates of the 29 sets. The 11.4% reference test error rate was obtained from the SVM trained with all 600 training patterns, among which 162 were picked as support vectors. We set the SVM error tolerance value at $C = 20$ and used the RBF kernel with width parameter $\sigma = 0.5$. SVM parameters, C and σ were defined in Eq. (3.1) and Eq. (3.2). The parameter values were fixed over all 30 SVMs. We used Gunn's SVM Toolbox which was built by standard QP routines in Matlab (*Kernel-Machines.org*).

Fig. 8.5.a shows that, when v is fixed, the increase of k leads to the increase of b_k . The number of selected patterns was slightly less than b_k , but it also

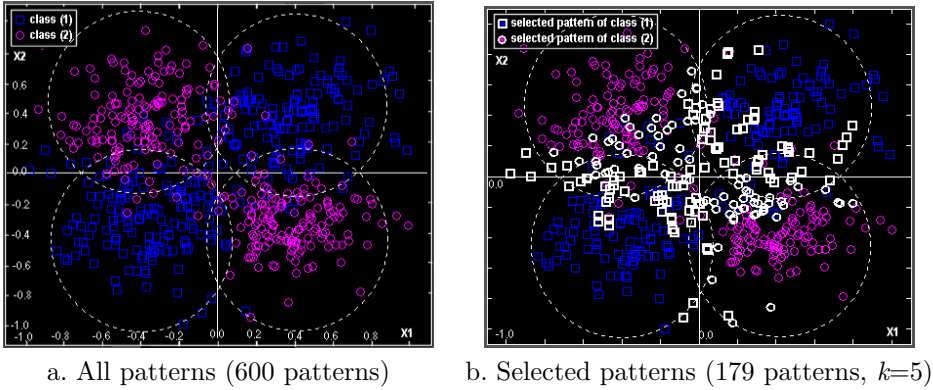


Figure 8.4: Continuous XOR with four Gaussian distributions: Selected patterns are outlined. Patterns are normalized ranging from -1 to 1.

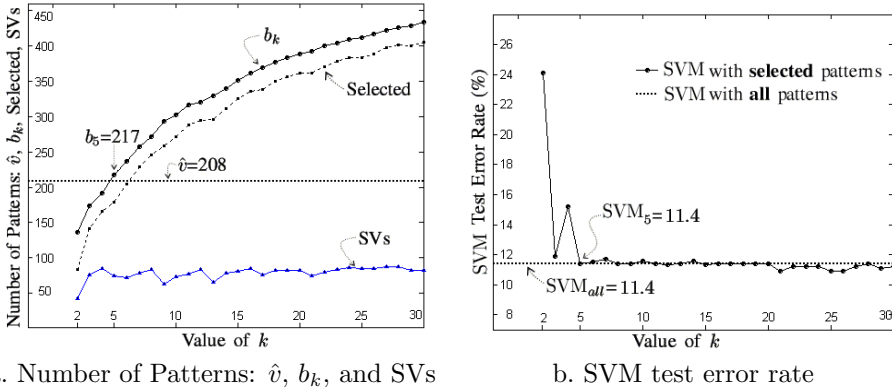


Figure 8.5: Continuous XOR: Number of patterns and SVM test error rate

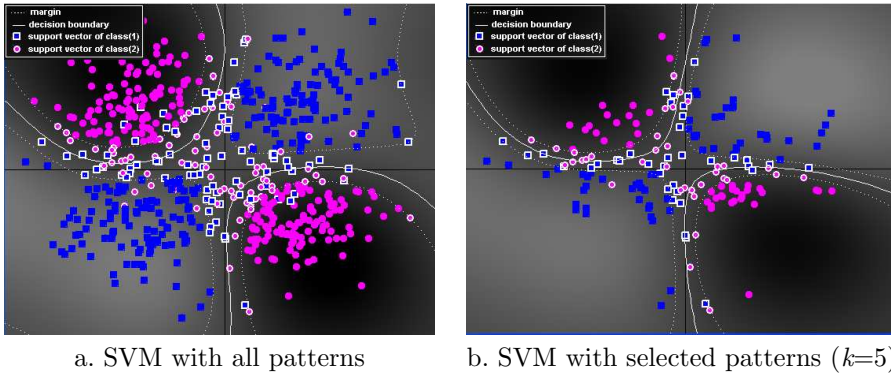


Figure 8.6: Patterns and SVM decision boundaries of Continuous XOR problem ($C = 20$, $\sigma = 0.5$): decision boundary is depicted as a solid line and the margins are defined by the dotted lines in both sides of it. Support vectors are outlined.

gradually increased almost parallel to the curve of b_k . The number of support vectors was also given at the bottom. For $k > 5$, it converged to about 78, which is only one half of the 162 SVs that were selected when trained with the full training pattern set. That is, only a subset of training patterns affected SVM training regardless of the number of training patterns. Meanwhile, only 78 SVs were adopted because “Neighbors_Match criterion” identified and removed those patterns that were suspected to be overlap patterns (see Fig. 4.1 and Fig. 5.2). Those overlap patterns were adopted as SVs in the original SVM by its error tolerance parameter, but they hardly contributed to margin constitution.

Fig. 8.5.b displays the test error rates for 30 different SVMs. The SVM performance was stabilized for k larger than 5 at which the test error rate was 11.4%. It is almost the same as the reference test error rate. The pattern sets larger than \mathbf{B}_5 did not lead to better SVM performance since they included the redundant patterns which did not contribute to SVM training. An evidence can be found from the number of support vectors in Fig. 8.5.a. From $k = 5$ to $k = 30$, the number of support vectors did not increase much from 75 to 83 while the number of the selected patterns increased by more than two times from 179 to 405.

Finally, Fig. 8.6 shows the decision boundaries and margins of the SVMs (a) with all patterns and (b) with the selected patterns when $k = 5$. Note that the two decision boundaries are quite similar. The selected patterns from \mathbf{B}_5 were sufficient to result in the same classification accuracy as the original SVM.

8.2.3 Sine Function Problem

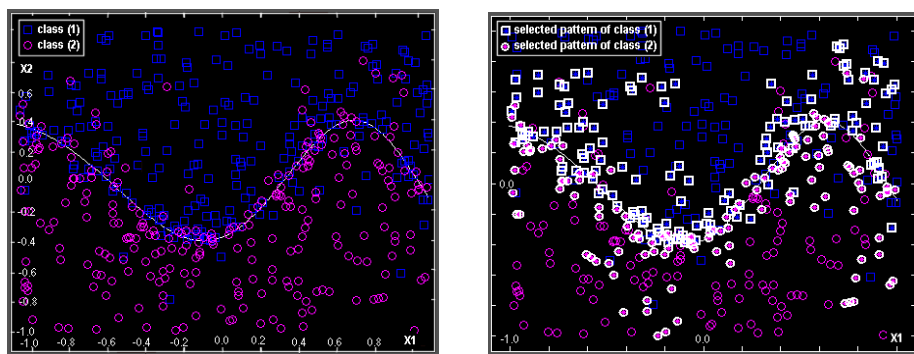
The third problem is a Sine Function problem. The input patterns were generated from a two-dimensional uniform distribution, and then the class labels were determined by whether the input is located above or below a sine decision function.

$$\begin{aligned} C_1 &= \left\{ \vec{x} \mid x_2 > \sin(3x_1 + 0.8)^2, \begin{bmatrix} 0 \\ -2.5 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 2.5 \end{bmatrix} \right\}, \\ C_2 &= \left\{ \vec{x} \mid x_2 \leq \sin(3x_1 + 0.8)^2, \begin{bmatrix} 0 \\ -2.5 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 2.5 \end{bmatrix} \right\}. \end{aligned} \quad (8.4)$$

To make the density near the decision boundary thicker, four different Gaussian noises were added along the decision boundary, i.e., $N(\vec{\mu}, s^2I)$ where $\vec{\mu}$ is an arbitrary point on the decision boundary and s is a Gaussian width parameter ($s = 0.1, 0.3, 0.8, 1.0$). A total of 500 training patterns were generated, 55% of them were located in the overlap region ($v = 276$). Fig. 8.7.a shows the problem. SVM parameters were $C = 20$ and $\sigma = 0.5$ (Eq. (3.1) and Eq. (3.2)), and 30 SVM test error rates ($SVM_k, k = 2, \dots, 30$ and SVM_{all}) were measured over a total of 1000 test patterns. The two graphs in Fig. 8.8 can be interpreted in the same manner as the Continuous XOR problem. The value of v was estimated as 296 ($\hat{v}=296$) and b_5 was the smallest among those b_k 's larger than \hat{v} . Thus, the value of k was set as 5 ($k^*=5$ and $b_{k^*}=302$). See Fig. 8.8.a. Among the patterns identified in \mathbf{B}_5 , 264 patterns were selected (Fig. 8.7.b). SVM test error rate, SVM_5 , was 15.0% (with 194 SVs), which was slightly better than $SVM_{all}= 15.2\%$ (with 255 SVs). As in the results of the Continuous XOR problem, for $k > 5$, the SVM test error rate and the number of SVs converged to 15.0% and 192, respectively. They are depicted in the Fig. 8.8.a. Fig. 8.9 shows that the decision boundaries and margins of the SVM_5 and SVM_{all} are quite similar.

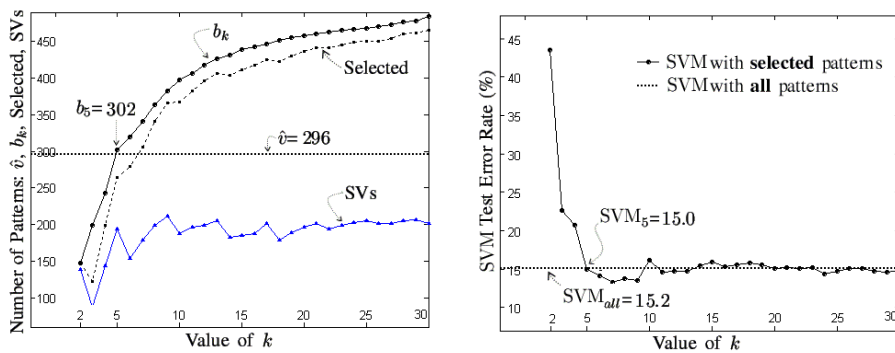
8.2.4 Discussion on Different Reduction Ratio

In the Sine Function problem, among the original training patterns, 264 (52.8%) were selected, which is greater than that of Continuous XOR, 179 (29.8%). The different reduction ratio is due to the difference in densities near the decision boundary. Fig. 8.10 presents the relationship between distance from decision boundary to a pattern and its value of Neighbors_Entropy. The closer to the decision boundary a pattern is, the higher value of Neighbors_Entropy it has. Both figures assure that Neighbors_Entropy, the measure we propose, is pertinent to



a. All patterns (500 patterns) b. Selected patterns (264 patterns, $k=5$)

Figure 8.7: Sine Function: Selected patterns are outlined. Patterns are normalized ranging from -1 to 1.



a. Number of Patterns: \hat{v} , b_k , and SVs b. SVM test error rate

Figure 8.8: Sine function: Number of patterns and SVM test error rate

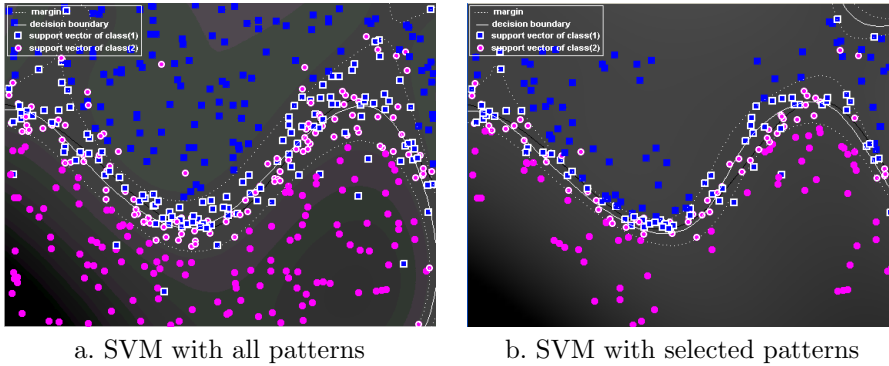


Figure 8.9: Patterns and SVM decision boundaries of Sine Function problem ($C = 20$, $\sigma = 0.5$): decision boundary is depicted as a solid line and the margins are defined by the dotted lines in both sides of it. Support vectors are outlined.

estimating the proximity to the decision boundary. Among the patterns with positive Neighbors_Entropy values, the ones which meet the Neighbors_Match criteria are selected. They are depicted as solid circles against outlined ones. Note that solid circles are distributed nearer the decision boundary.

8.3 Training Time Reduction

The main objective of our proposed algorithm is to reduce the number of training patterns to relieve lengthy SVM training time. The proposed pattern selection algorithm can alleviate individual SVM training time. The time reduction is more significant when we consider SVM optimal parameter searching process which is laborious and time-consuming but necessary. In this section, we present the experimental results from the viewpoints of individual SVM training time reduction and model selection time reduction.

For the Continuous XOR problem and the Sine Function problem (both were introduced in section 8.2), we trained 50 SVMs with all patterns and 50 SVMs with the selected patterns from candidate combination of hyper-parameters (σ , p , C). Five RBF kernels with different width parameters ($\sigma = 0.25, 0.5, 1, 2, 3$) and five polynomial kernels with different degree parameters ($p = 1, 2, 3, 4, 5$) were adopted. As for the misclassification error tolerance parameter, five lev-

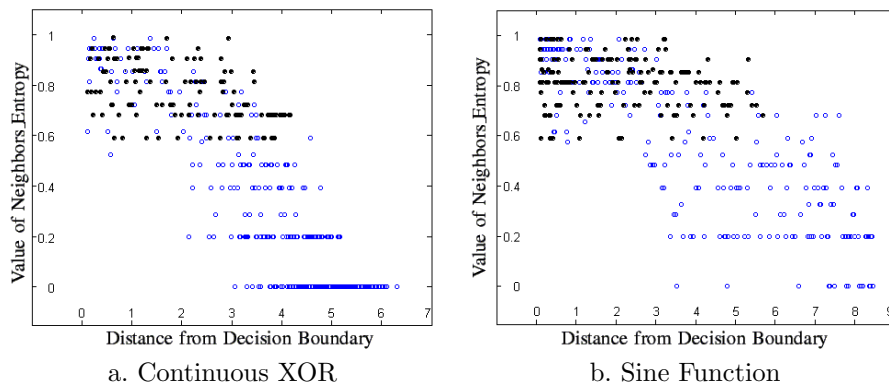


Figure 8.10: Distance from decision boundary and Neighbors_Entropy: The closer to the decision boundary a pattern is, the higher value of Neighbors_Entropy it has. The selected patterns are depicted as solid circles against outlined ones.

els ($C = 0.1, 1, 10, 100, 1000$) were used. Refer to those SVM hyperparameters from Eq. (3.1) and Eq. (3.2). Gunn’s SVM Matlab Toolbox was used for SVM training (*Kernel-Machines.org*). The selected pattern sets from *naive* NPPS and *fast* NPPS (with initial sampling ratio $r = 0.2$) were identical. SVM performances “with all patterns (ALL)” and “with the selected patterns (SELECTED)” were compared in terms of the test error rate, the execution time, and the number of support vectors.

The experimental results are shown in tables 8.3 through 8.6. Fig. 8.11 extracts SVM test error rates from table 8.3–8.6. In each cell, the upper number indicates the test error rate for ALL and lower for SELECTED. Gray cell means that SELECTED performed better than or equal to ALL, while white cell means that ALL performed better. An interesting fact is that SVM performance with SELECTED is more sensitive to parameter variation. This phenomenon is shown substantially with ill-posed parameter combination (white cells). For the parameters where SVM with ALL performed well, SVM with SELECTED also worked well.

Table 8.7 compares the best performance of each SVM. The corresponding parameter combination is marked “*” in Fig. 8.11. The SVM with the selected patterns was trained much faster thanks to fewer training patterns. Less important support vectors were eliminated in building the margins. Employing fewer SVs resulted in a smaller recall time. While the computational efficiency was achieved, the generalization ability was not sacrificed at all. Table 8.8 shows

Table 8.3: Continuous XOR: SVM Result of **All** Patterns (**600**)

Test Error (%)	RBF					Poly.				
Exe. Time (sec)										
Num. of SVs	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$C = 0.1$	11.33	11.67	9.67	25.33	34.33	51.67	10.67	11.67	10.00	10.33
	506.58	506.74	496.36	470.72	470.49	469.56	491.31	501.41	504.88	518.28
	312	344	549	600	600	600	431	339	292	261
$C = 1.0$	10.00	12.00	11.00	10.67	34.33	34.00	10.00	10.67	11.33	11.33
	504.99	488.89	459.57	449.89	437.26	469.61	472.80	480.16	491.86	496.47
	194	203	322	557	600	598	262	216	193	185
$C = 10$	10.00	11.33	10.00	12.00	10.00	31.67	9.67	10.33	11.67	12.00
	511.57	464.56	442.26	424.46	418.37	435.39	453.19	453.85	460.22	465.55
	165	166	203	340	486	596	186	173	168	165
$C = 100$	10.00	11.00	9.67	10.67	10.33	31.67	10.00	10.00	11.67	12.33
	467.36	467.14	454.83	453.19	435.01	455.77	462.31	455.28	470.87	476.04
	152	159	167	215	291	596	170	163	159	158
$C = 1000$	10.67	10.33	11.33	10.33	10.00	31.67	10.33	9.67	12.33	12.67
	490.21	478.67	475.88	473.79	460.99	471.09	473.02	485.76	489.50	496.20
	150	156	160	174	197	596	167	161	156	157

Table 8.4: Continuous XOR: SVM Result of **Selected** Patterns (**179**)

Test Error (%)	RBF					Poly.				
Exe. Time (sec)										
Num. of SVs	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$C = 0.1$	11.33	29.33	50.00	50.00	50.00	50.00	50.00	32.67	11.67	10.67
	5.11	4.95	4.34	4.39	4.94	8.85	4.39	4.28	4.28	4.29
	167	175	179	179	179	179	172	169	165	158
$C = 1.0$	10.33	12.00	31.33	50.00	50.00	50.00	15.67	12.33	11.33	11.00
	4.72	4.01	4.83	4.07	4.01	7.20	4.00	4.01	3.95	4.07
	99	128	165	168	179	179	159	133	118	110
$C = 10$	10.00	10.33	10.00	35.33	50.00	50.00	10.00	9.67	10.33	10.00
	3.84	3.79	3.68	3.63	3.79	6.70	3.74	3.79	3.84	3.90
	65	80	125	164	165	179	108	89	80	72
$C = 100$	9.00	10.33	9.67	10.33	22.67	50.00	9.00	10.00	10.33	10.67
	4.01	3.89	3.85	3.73	3.73	6.87	3.95	4.39	3.96	3.96
	59	59	84	133	162	179	79	69	62	59
$C = 1000$	11.00	10.67	10.33	10.33	10.00	50.00	9.00	10.00	11.33	11.67
	4.73	4.17	4.06	4.07	4.17	6.97	4.06	4.18	4.17	4.17
	49	56	64	91	119	179	70	58	55	54

Table 8.5: Sine Function: SVM Result of **All** Patterns (500)

Test Error (%)	RBF					Poly.				
Exe. Time (sec)										
Num. of SVs	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$C = 0.1$	16.33	18.33	20.00	20.67	20.33	20.67	21.00	20.67	16.67	16.00
	280.34	282.10	280.12	278.19	277.32	281.33	281.33	284.30	288.42	290.23
	406	354	367	420	473	345	327	316	306	302
$C = 1.0$	13.67	16.33	19.33	20.00	20.67	20.33	19.67	17.33	14.00	13.67
	267.71	266.12	260.95	261.72	258.15	259.14	264.36	268.31	272.82	273.04
	281	276	303	325	343	307	297	283	277	271
$C = 10$	14.67	13.33	16.67	19.00	20.67	20.33	20.00	15.67	14.00	14.33
	254.58	252.22	244.75	240.08	241.67	243.81	244.26	261.78	253.92	254.85
	256	252	278	299	306	303	293	270	257	254
$C = 100$	16.67	14.33	13.33	18.33	19.67	20.33	20.00	13.67	13.33	14.33
	257.65	252.88	250.85	247.88	253.70	248.38	254.03	263.37	267.76	255.68
	236	243	260	288	294	301	293	266	250	252
$C = 1000$	18.00	13.33	14.00	15.00	19.00	20.33	20.00	15.00	14.33	14.67
	269.03	264.90	264.64	265.29	260.13	258.04	265.84	275.39	279.62	268.92
	234	245	249	273	288	301	293	266	250	251

Table 8.6: Sine Function: SVM Result of **Selected** Patterns (264)

Test Error (%)	RBF					Poly.				
Exe. Time (sec)										
Num. of SVs	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$C = 0.1$	16.67	18.33	20.00	33.33	55.33	22.67	19.33	19.67	16.33	15.67
	11.76	8.19	10.44	9.34	8.85	22.36	9.34	8.41	8.19	8.68
	255	246	247	261	264	230	218	209	201	197
$C = 1.0$	13.33	15.67	19.33	20.33	22.33	22.00	19.33	16.67	14.00	13.67
	9.12	7.64	7.42	8.13	7.58	18.73	7.42	7.81	8.52	7.42
	172	187	204	216	228	202	196	187	175	170
$C = 10$	14.67	12.67	14.67	19.67	20.67	21.67	20.00	15.33	13.67	14.33
	6.71	8.51	6.48	6.65	6.81	17.47	7.36	7.91	8.25	7.64
	132	143	182	197	202	197	191	163	154	144
$C = 100$	15.33	13.33	13.67	18.67	19.33	21.67	18.67	13.67	13.33	14.33
	7.53	6.93	7.58	8.36	8.19	17.69	8.19	6.82	8.79	7.74
	120	128	158	190	192	196	190	152	136	131
$C = 1000$	18.00	13.33	14.33	14.67	17.67	20.33	18.67	12.33	14.33	14.00
	8.41	7.97	7.42	7.09	7.31	18.9	9.23	7.58	7.74	7.75
	112	124	135	174	189	196	190	151	130	124

Kernel \ C		Rbf (Width Parameter)					Polynomial (Degree Parameter)				
		0.25	0.5	1	2	3	1	2	3	4	5
0.1	11.33	11.67	9.67	25.33	34.33	51.67	10.67	11.67	10.00	10.33	
	11.33	29.33	50.00	50.00	50.00	50.00	50.00	32.67	11.67	10.67	
1.0	10.00	12.00	11.00	10.67	34.33	34.00	10.00	10.67	11.33	11.33	
	10.33	12.00	31.33	50.00	50.00	50.00	15.67	12.33	11.33	11.00	
10	10.00	11.33	10.00	12.00	10.00	31.67	9.67	10.33	11.67	12.00	
	10.00	10.33	10.00	35.33	50.00	50.00	10.00	9.67	10.33	10.00	
100	10.00	11.00	9.67	10.67	10.33	31.67	10.00	10.00	11.67	12.33	
	9.00	10.33	*	9.67	10.33	22.67	50.00	9.00	10.00	10.33	10.67
1000	10.67	10.33	11.33	10.33	10.00	31.67	10.33	9.67	12.33	12.67	
	11.00	10.67	10.33	10.33	10.00	50.00	9.00	10.00	11.33	11.67	

a. Continuous XOR problem

Kernel \ C		Rbf (Width Parameter)					Polynomial (Degree Parameter)				
		0.25	0.5	1	2	3	1	2	3	4	5
0.1	16.33	18.33	20.00	20.67	20.33	20.67	21.00	20.67	16.67	16.00	
	16.67	18.33	20.00	33.33	55.33	22.67	19.33	19.67	16.33	15.67	
1	13.67	16.33	19.33	20.00	20.67	20.33	19.67	17.33	14.00	13.67	
	13.33	15.67	19.33	20.33	22.33	22.00	19.33	16.67	14.00	13.67	
10	14.67	13.33	16.67	19.00	20.67	20.33	20.00	15.67	14.00	14.33	
	14.67	12.67	14.67	19.67	20.67	21.67	20.00	15.33	13.67	14.33	
100	16.67	14.33	13.33	18.33	19.67	20.33	20.00	13.67	13.33	14.33	
	15.33	13.33	13.67	18.67	19.33	21.67	18.67	13.67	*	13.33	14.33
1000	18.00	13.33	14.00	15.00	19.00	20.33	20.00	15.00	14.33	14.67	
	18.00	13.33	14.33	14.67	17.67	20.33	18.67	12.33	14.33	14.00	

b. Sine Function problem

Figure 8.11: SVM test error comparison: In each cell, the upper number indicates test error rate for ALL and the lower for SELECTED.

Table 8.7: Best Result Comparison

	Continuous XOR		Sine Function	
	ALL	SELECTED	ALL	SELECTED
Execution Time (sec)	454.83	3.85	267.76	8.79
Margin	0.0612	0.0442	0.1904	0.0874
Num. of Training Patterns	600	179	500	264
Num. of Support Vectors	167	84	250	136
Training Error (%)	10.50	13.89	19.00	19.32
Test Error (%)	9.67	9.67	13.33	13.33

Table 8.8: Execution Time Comparison

		Num. of patterns	Num. of SVs (avg.)	Pattern Selection CPU time	SVM Avg. CPU time	Model Selection CPU time
Continuous XOR	ALL	600	292.20	-	472.20	23610.16
	SELECTED	179	120.54	0.88 (Naive) 0.21 (Fast)	4.45	222.48
Sine Function	ALL	500	293.60	-	263.84	13191.90
	SELECTED	264	181.76	0.66 (Naive) 0.17 (Fast)	9.13	456.36

the execution time from Fig. 8.11. The average SVM training time of ALL was 472.20 (sec) for Continuous XOR problem and 263.84 (sec) for Sine Function problem. That of SELECTED was 4.45 (sec) and 9.13 (sec), respectively. Pattern selection itself took 0.88 (sec) and 0.66 (sec) by *naive* NPPS and 0.21 (sec) and 0.17 (sec) by *fast* NPPS, respectively. Therefore, by conducting pattern selection procedure just once, we were able to reduce SVM training time by one or two orders of magnitude, thus reduce the total time for model selection.

8.4 Comparison to A Similar Previous Approach

Almeida *et al.* (2000) proposed to select training patterns in order to reduce SVM training time in their previous study (Almeida *et al.*, 2000). Since their approach and goal are similar to ours, we could directly compare the results of the two methods based on the synthetic problem Almeida adopted.

8.4.1 4x4 Checkerboard Problem

fast NPPS was tested on a synthetic 4x4 Checkerboard problem. The training set consists of 1,000 patterns uniformly distributed in a 4x4 Checkerboard square, \vec{x} ranging from -1 to 1. Each square is occupied by patterns from a same class. See Fig. 8.12.a for overall distribution of the two class patterns. A

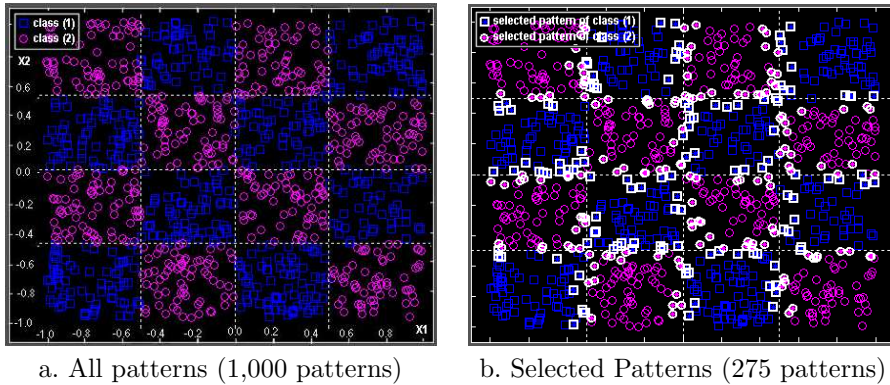


Figure 8.12: 4x4 Checkerboard problem: Selected patterns, shown as outlined circles or squares, are scattered against the original ones

total of 10,000 test patterns were also obtained from the same distribution. We trained a SVM with OSU SVM Classifier Matlab Toolbox, which is a hybrid algorithm of SMO and SVM^{light} since 1,000 training patterns were too large to fit into the memory for the standard Matlab QP solver such as Gunn’s (*Kernel-Machines.org*). SVM parameter values and the number of repetitions were set in the same way as in (Almeida *et al.*, 2000): RBF kernel with $\sigma = 0.25$, $C = 20$, and 100 times of repetition. The value of k was set as 4 by following the procedure in section 6.4. Fig. 8.12.b shows the 275 selected patterns by *fast* NPPS (27.5% of training set).

Fig. 8.13 shows a typical result from 100 repetitions, the decision boundaries, margins, and support vectors of the SVM “with all patterns (ALL)” and “with the selected patterns (SELECTED)”. The decision boundaries in both figures look quite similar, thus, generalization performance is similar.

Table 8.9 compares our approach to Almeida *et al.* (2000) on an average basis. First, the average number of the selected patterns in Almeida *et al.* (2000) was 497.15, but that of the proposed algorithm was 275.52. They selected about a half of the training patterns while we selected about a quarter. However, the number of support vectors selected by our approach was about the same as theirs. It means that the training set selected by the proposed method is more compact. Direct comparison of the execution time does not make sense because two experiments were performed in different computing environments. Thus, we do compare both algorithms in terms of the SVM training time reduction ratio. While Almeida *et al.* (2000)’s algorithm reduced 60% of training time, ours reduced 98%, almost two orders of magnitude: Almeida *et al.* (2000)’s

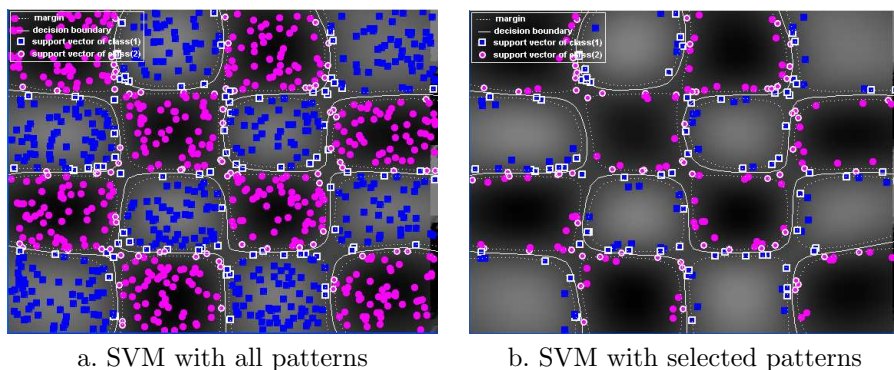


Figure 8.13: Patterns and SVM decision boundaries of 4x4 Checkerboard problem ($C = 20$, $\sigma = 0.25$): decision boundary is depicted as a solid line and the margins are defined by the dotted lines in both sides of it. Support vectors are outlined.

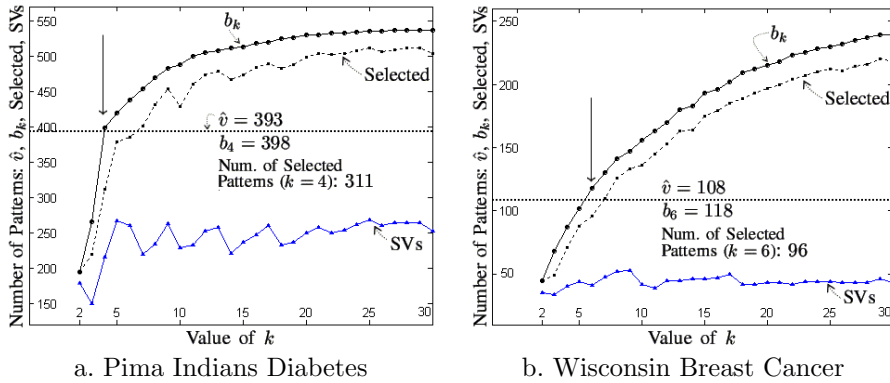
Table 8.9: Result Comparison

	Almeida <i>et al.</i> (2000)		Proposed Algorithm	
	ALL	SELECTED	ALL	SELECTED
Num. of Training Patterns	1000	497.15	1000	275.52
Num. of Support Vectors	172	159.69	172	148.20
Training Time Reduction Ratio	1	0.40 ($=\frac{13.00}{32.21}$)	1	0.02 ($=\frac{0.09}{3.81}$)
Pattern Selection Time (sec)	-	0.89	-	0.32
Test Error (%)	4.04	5.08	4.03	4.66

algorithm reduced 19 seconds out of 32 while ours reduced 3.72 seconds out of 3.81 seconds. The pattern selection process itself took less than a second, so it was marginal compared to the training time reduction. There was a slight increase in the test error, from 4% to 4.6%. However, it was less than 5% error achieved by (Almeida *et al.*, 2000).

8.5 Real World Problems

The proposed approach was applied to three real world datasets: Pima Indian Diabetes and Wisconsin Breast Cancer datasets from (*UCI Repository*) and

Figure 8.14: Number of patterns: \hat{v} , b_k , and SVs

MNIST dataset of handwritten digits from (*MNIST*). For all three datasets, OSU SVM Classifier Matlab Toolbox was used (*Kernel-Machines.org*).

8.5.1 UCI Repository Datasets

The experiments on Pima Indian Diabetes and Wisconsin Breast Cancer are presented first. To measure the performance, 5-fold cross validation (CV) was conducted on each dataset. According to Eq. (6.13), v values for the two datasets were estimated as 393 and 108 from $P(\text{error}) = 32.0\%$ and $P(\text{error}) = 9.9\%$, respectively. The value of k was determined by the procedure introduced in section 6.4 : $k = 4$ in Pima Indian Diabetes and $k = 6$ in Wisconsin Breast Cancer (see Fig. 8.14). We chose a quadratic polynomial SVM kernel and the error tolerance parameter $C = 100$ for Pima Indian Diabetes, and a cubic polynomial kernel and $C = 5$ for Wisconsin Breast Cancer. Fig. 8.15 depicts the average SVM CV error rates. In Pima Indian Diabetes, the SVM performance was stabilized at about 30.0% for k larger than 4, and in Wisconsin Breast Cancer, 6.7% for k larger than 6. The results are similar to those from the synthetic data experiments.

Table 8.10 compares the average execution times and the performances of SVM with all patterns vs. SVM with the selected patterns. In both datasets, the average SVM training time was reduced from 203.91 (sec) to 27.86 (sec), and from 2.14 (sec) to 0.03 (sec), respectively. But, the generalization accuracies were maintained almost same.

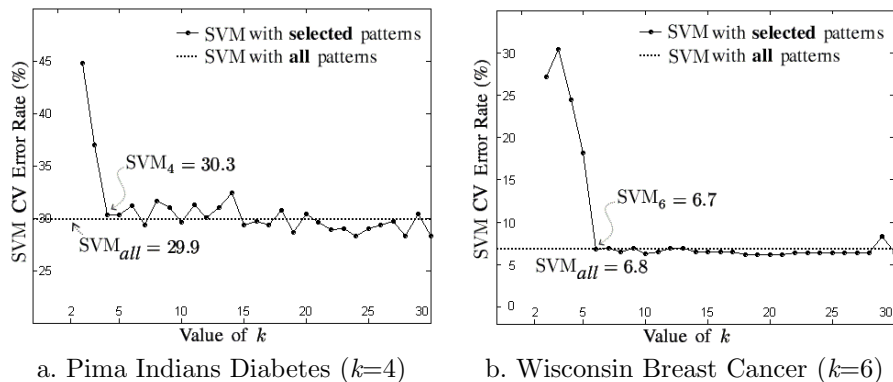


Figure 8.15: SVM CV error rates

Table 8.10: SVM Training Results: ALL vs. SELECTED

	Num. of Trn. patterns	Num. of SVs	Pattern Sel. time (sec)	SVM trn. time (sec)	SVM CV error (%)
Pima Indian Diabetes					
ALL	615	330	-	203.91	29.9
SELECTED ($k = 4$)	311	216	0.24	27.86	30.3
Wisconsin Breast Cancer					
ALL	546	87	-	2.14	6.8
SELECTED ($k = 6$)	96	41	0.10	0.03	6.7

Table 8.11: MNIST Result Comparison

	Paired Classes								
	0-8	1-8	2-8	3-8	4-8	5-8	6-8	7-8	9-8
ALL									
Num. Of Patterns	11774	12593	11809	11982	11693	11272	11769	12116	11800
Num. Of SVs	538	455	988	1253	576	1039	594	578	823
SVM Training Time (sec)	201.74	190.76	368.82	477.25	212.18	379.59	222.84	222.40	308.73
Test Error (%)	0.51	0.09	0.34	0.50	0.10	0.16	0.31	0.10	0.41
SELECTED									
Num. Of Patterns	1006	812	2589	4089	1138	3959	1135	999	1997
Num. Of SVs	364	253	828	1024	383	882	421	398	631
SVM Training Time (sec)	11.11	5.92	33.03	49.84	12.11	49.74	14.69	13.55	26.61
Pattern Sel. Time (sec)	46.11	43.33	78.81	97.89	48.13	93.42	44.27	40.14	59.62
Test Error (%)	0.41	0.95	0.34	0.45	0.15	0.21	0.31	0.18	0.43

8.5.2 MNIST Dataset

MNIST dataset consists of 60,000 patterns for training and 10,000 patterns for testing. All binary images are size-normalized and coded by gray-valued 28x28 pixels in the range between -1 and 1, therefore, input dimension is 784. Nine binary classifiers of MNIST were trained: class 8 is paired with each of the rest. A Fifth-order polynomial kernel ($p = 5$), C value of 10, and KKT(Karush-Kuhn-Tucker) tolerance of 0.02 were used as in (Platt, 1999), and the value of k was set to 50, more or less. The results for the training run both with all patterns (ALL) and with the selected patterns (SELECTED) are shown in Table 8.11. First, we compare the results with regard to the number of training patterns and also the number of support vectors. The proposed algorithm chose an average 16.75% of patterns from the original training sets. When all patterns were used, only 6.44% of the training patterns were used as support vectors. With selected patterns, 32.09% of its training patterns were support vectors. In terms of utilization, the pattern selection did well (see Fig. 8.16.a). Second, SVM training time was significantly reduced from 287.15 (sec) to 24.07 (sec) on average after the pattern selection procedure. Including the pattern selection time, the total time was, on average, 85.37 (sec). See Fig. 8.16.b. Note that SVM training is usually performed several times to find the optimal parameters. However, the pattern selection is performed only once. Now we finally compare the SVM test error rates between ALL and SELECTED. First, the average test error rate over nine classifiers was 0.28% for the former and 0.38% for the latter. The increase in average error of SELECTED was due to the high error rate of one classifier, i. e. 1–8 classifier. This exception may result from the unique

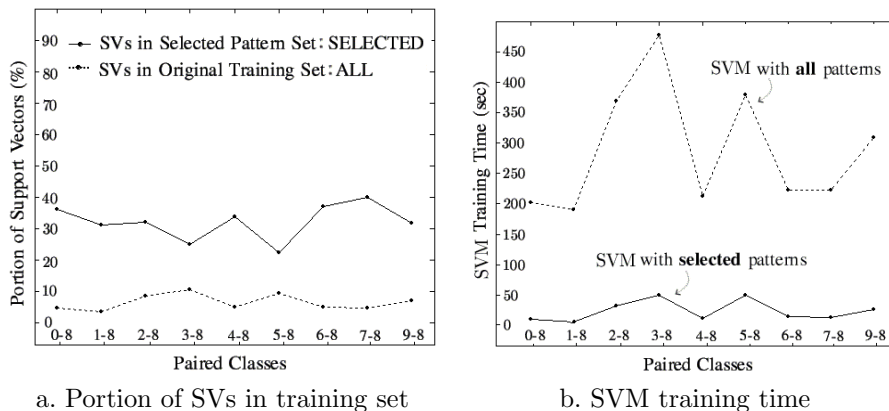


Figure 8.16: MNIST Result Comparison

characteristic of digit '1' images. Individual patterns belonging to '1' class are sparse vectors: only a few fields of a vector have significant gray-scaled value, and the rest of the fields have value of '0'. Therefore, a pattern in '1' class finds the most part of its neighbors in the same class not in the opposite class. It causes the set \mathbf{B} small, thus very few patterns are chosen (especially from '1' class) during the pattern selection procedure. This conjecture is accompanied by the number of selected patterns of '1' digit class. Only 95 patterns were selected from 6742 patterns in '1' digit class. In the meantime, the average test error rate for the rest was 0.30% for ALL and 0.31% for SELECTED.

Applications with a Large Scale Data set – Response Modeling For Customer Relationship Management

This chapter exemplifies a practical usage of the proposed method by applying to a real problem in marketing domain - response modeling in direct marketing. This chapter is rather a domain-oriented application than a simple algorithm-oriented application. We first diagnose the domain specific difficulties which can arise in practice when applying SVM to response modeling in direct marketing, such as large training data, class imbalance and binary SVM output. To alleviate or solve the addressed difficulties, we propose use of the proposed method, use of different costs for different classes, and use of distance to decision boundary. This chapter also provides various evaluation measures for response models in terms of accuracies, lift chart analysis and computational efficiency.

9.1 Introduction

Direct marketing is concerned with identifying likely buyers of certain products or services and promoting them to the potential buyers through various channels. A response model predicts a probability that a customer is going to respond to a promotion or offer. Using the model, one can identify a subset of customers who are more likely to respond than others. A more accurate response model will have more respondents and fewer non-respondents in the subset. By doing so, one can significantly reduce the overall marketing cost without sacrificing opportunities.

Various statistical and machine learning methods have been proposed for response modeling. These researches will be reviewed in section 9.2. Most recent is Support Vector Machine (SVM). However, there are some difficulties one would face when SVM is attempted to be applied to response modeling. First, SVM training can become computationally intractable. Generally, retailers keep huge amounts of customer data. Moreover, a new customer's record will be added on top of it on and on. Second, response modeling is likely to have a severe class imbalance problem since the customers' response rates are typically very low. Most of customers belong to the non-respondents' group (class 1), while only a few customers belong to the respondents' group (class 2). Under such a circumstance, most classifiers do not behave well, and neither does SVM. Third, one has to find a way to estimate scores or likelihoods from SVM. Given a limited amount of marketing expenses, a marketer wants to maximize the return or total revenue. Thus, one would like to know who is more likely to purchase than others. Response models compute each customer's likelihood or propensity to respond to a particular offer of a product or a service. These likelihood values or scores are then used to sort the customers in a descending order. Now, the marketer simply applies a cut-off value based on the marketing expenses and only those customers whose scores are larger than the value are identified. However, an SVM classifier returns a binary output, not a continuous output which can be interpreted as a score.

In this chapter, we address the obstacles mentioned above. For the intractability problem of SVM training, we use the proposed pattern selection algorithm that reduces the training set without accuracy loss. To alleviate the class imbalance problem, we propose to assign for different classes different misclassification costs which is proportional to the size of the counter class dataset. Finally, we propose to use the distance from a pattern to the decision hyperplane in the feature space for scores. In addition, we provide various measures for evaluating the response models in both accuracy and profit.

The remaining part of this chapter is organized as follows. Section 9.2 presents related works on various statistical or machine learning methods applied to direct marketing domain. Section 9.3 addresses the obstacles in applying SVM to response modeling. The section proposes ways to reduce the training set, to handle the class imbalance problem, and to obtain the customer scores from an SVM classifier. Section 9.4 provides the experimental results on a direct marketing dataset. The section includes the data set description, experimental design, and performance measurements. The last section summarizes the chapter.

9.2 Related Work

9.2.1 Response Modeling

Traditionally, statistical methods, mostly regression techniques, have been applied to response modeling. Most textbooks cover logistic regression as the de facto method due to its simplicity, explainability and availability (Hosmer and Lemeshow, 1989; Sen and Srivastava, 1990). *Malthouse (1999)* compared ridge regression with stepwise regression on the Direct Marketing Educational Foundation data set 2 (DMEF2) (*The Direct Marketing Association*). In his study, both methods were used for determining the moderate number of variables in response modeling. Empirically, he showed that ridge regression is a more stable and less risky method than dropping variables. In his recent report, a similar approach, which additively considered the dollars spent in response to an offer, was proposed (Malthouse, 2002). *Colombo and Jiang (1999)* proposed a simple Recency-Frequency-Monetary (RFM) stochastic model for ranking (or scoring) customers. The RFM stochastic model derived from the response distribution of the past was used to estimate the likelihood of future responses. A customer mailing list obtained from a tele-marketing company was used for comparing the performance of the stochastic model with that of regression and cross-tabulation model. They reported that the stochastic model provided a more insightful alternative to ranking customers.

Recently, machine learning methods have been proposed. They include decision trees and neural networks, etc. *Haughton and Oulabi (1997)* compared the response lifts of two mostly common decision tree algorithms: Classification

and Regression Tree (CART) and Chi-Square Automatic Interaction Detector (CHAID). Although the two models are different in their tree-generating mechanism, there was no significant difference in the response-lift perspective. *Ling and Li (1998)* compared a Naive Bayes response model and a C4.5 response model. Applying the ada-boost algorithm (Freund and Schapire, 1996) to each base model for better performance, they conducted experiments on three direct marketing problems such as loan product promotion, life insurance product campaign, and bonus program. All experiments were designed to discuss the difficulties which can arise during the response modeling process, such as class imbalance and justifiability of evaluating measures. *Coenen et al. (2000)* proposed to combine C5, a decision tree algorithm, and case-based reasoning (CBR). In this approach, the C5 based response modeling was conducted in the first step. Then, the respondents classified by the initial model were ranked by a CBR similarity measure. They improved the classification quality by accommodating a better ranking rather than the accuracy of the base response model itself. *Chiu (2002)* integrated genetic algorithm (GA) into a CBR based response model. For a better case identification accuracy, the fittest weighting values on the cases were searched by GA. On the application of an insurance product purchase dataset, the base response model, CBR, achieved a better classification accuracy. *Deichmann et al. (2002)* investigated the use of Multiple Adaptive Regression Splines (MARS) as a response model. MARS is an advanced decision tree technique enabling piecewise linear regression. The MARS response model outperformed the logistic regression model on the DMEF2 (*The Direct Marketing Association*).

There have also been many reports on neural networks. *Moutinho et al. (1994)* predicted bank customers' responses using neural networks, and *Bounds and Ross (1997)* showed that neural network based response models improved the response rate from 1 or 2 % up to 95%. *Zahavi and Levin (1997a)* addressed unique merits and demerits of neural networks for response modeling. *Viaene (2001a)* proposed to select relevant variables for neural network based response models. *Ha et al. (2004)* proposed a response model using *bagging* neural networks. The experiments over a publicly available DMEF4 dataset (*The Direct Marketing Association*) showed that bagging neural networks give more improved and stabilized prediction accuracies than single neural networks and logistic regression. Performance comparison of the methods has been one of the controversial issues in direct marketing domain. *Zahavi and Levin (1997)* and *Suh et al. (1999)* found that neural network did not outperform other statistical methods. They suggested to combine the neural network response model and the statistical method. On the other hand, *Bentz and Merunkay (2000)* reported that neural networks outperformed multinomial logistic regression. *Potharst et al. (2001)* applied neural networks to direct mailing campaigns of a large Dutch charity organization. According to their results, the performance of neural net-

works surpassed that of CHAID or logistic regression.

Although SVM is applied to a wide variety of application domains, there have been only a couple of SVM application reports in response modeling. *Cheung et al. (2003)* used SVM for content-based recommender systems. Web retailers implement a content-based system to provide recommendations to a customer. The system automatically matches his/her interests with product-contents through web-pages, newsgroup messages, and new items. It is definitely a form of direct marketing that has emerged by virtue of recent advances in the world wide web, e-business, and on-line companies. They compared Naive Bayes, C4.5 and 1-nearest neighbor rule with SVM. The SVM yielded the best results among them. More specific SVM application to response modeling was attempted by *Viaene (2001b)*. They proposed a Least Square SVM (LS-SVM) based wrapper approach. *Wrapper* indicates an input variable selection procedure working together with a learning algorithm, and it is frequently compared with alternative procedure, *filter*, that performs variable selection independently from a learning algorithm. In their study, the input variable pool was composed of RFM and non-RFM variables from the customer dataset provided by a major Belgian mail-order company. Then, the wrapper approach was performed in a sequential backward fashion, guided by a best-first variable selection strategy. Their approach, a wrapper around the LS-SVM response model, could gain significant reduction of model complexity without degrading predictive performance.

9.2.2 Large Training Set, Class Imbalance, and Binary Output

Now, let us focus on the researches related to the difficulties we addressed in this chapter. For the literature review on the issue of large training set, refer to chapter 2.

9.2.2.1 Class Imbalance

Regarding class imbalance, many researchers have recognized this problem and suggested several methods: enlarging the small class dataset by random sampling, reducing the large class dataset by random sampling, and ignoring the small class dataset and using only the large class dataset to build a one-class recognizer. *Japkowicz (2000)* compared the three commonly used methods above on the degree of concept complexity using a standard neural network classifier.

All the methods generally improved the performance of the learning algorithm. In particular, the first two methods were very effective especially as the concept complexity increases while the last one was relatively less accurate. *Ling and Li (1998)* addressed the specificity of the class imbalance problem which resides in marketing datasets. They did not attempt to balance the imbalanced class ratio for better predictive accuracy. Instead, to circumvent the class imbalance problem, a marketing specific evaluation measure, *lift index*, was suggested. Lift index provides the customer's rank (score) by reflecting the confidence of classification result. They argued that even if all of the patterns are predicted as one class, as long as the learning algorithm produces suitable ranking of the patterns, the imbalanced class distribution in the training set would no longer be a problem. However, in their experiments all the best lift index were obtained when the sizes of the classes were equal. Thus, they recommended to reduce the large class dataset so that its size becomes equal to that of the small class. Alternatively, different misclassification costs can be incorporated into classes, which avoids direct artificial manipulation on the training set (*Lee et al., 2001*).

9.2.2.2 Getting Scores from Binary Output

Getting scores from a logistic regression model or a neural network model with sigmoidal output function is well known. The output gives a value of probability belonging to the class, that is ranged from 0 to 1. Thus the output value is used as a score for sorting the customers. *Ling and Li (1998)* made use of the ada-boost algorithm (*Freund and Schapire, 1996*), an ensemble approach, to get the customers' scores. Basically, ada-boost maintains a sampling probability distribution on the training set, and modifies the probability distribution after each classifier is built. The probability of patterns with an incorrect prediction by the previous classifier is increased. So these patterns will be sampled more likely in the next round of boosting, to be learnt correctly. A pattern's probability to be incorrectly predicted allowed a corresponding rank. Sometimes, scores could be directly estimated by regression model having continuous target value, i.e., the dollars spent or the amount of orders. To do that, however, one needs to diagnose the problems the target variable has and conduct suitable remedies to cure them. *Malthouse (2001)* built a regression model to estimate the dollars spent on DMEF4 (*The Direct Marketing Association*). There was a large number of extreme values and the distribution was highly skewed. The extreme values could have a large influence on estimate values under least squares. And the variance of target variable most likely increased with its mean (heteroscedasticity). Thus, he performed log transformation to alleviate skewness and heteroscedasticity, and used *winsorization* to exclude some extreme values of target. The predicted value of the dollars spent was used as a score in lift chart analysis. The lift result by means of regression based scoring

will be briefly compared with that by means of classification based scoring in section 9.4. Generally speaking, regression problem requires more information from input variables than classification problem does. In other words, binary classification is the simplest subproblem of regression. Producing good scores from marketing regression model is difficult at the present time. In addition, since SVM theory stemmed from classification context (Schölkopf, 1999), it is natural to get scores from an SVM classifier.

9.3 Methods to Cope with Practical Difficulties

9.3.1 Large Training Set

The most straightforward method to reduce a large training set is random sampling. In SVM, however, the patterns near the decision boundary are critical to learning. The training set reduced by random sampling may omit those, thus would lead to significantly poorer prediction results. Instead, it will be desirable to apply the method proposed in the thesis to handle this difficulty. Contrary to a usually employed “random sampling,” NPPS can be viewed as “informative or intelligent sampling.” Fig. 9.1 conceptually shows the difference between NPPS and random sampling in selecting a subset of the training data. NPPS selects the patterns in the region around the decision boundary, while random sampling selects those from the whole input space.

9.3.2 Class Imbalance

Usually there are many more non-respondents than respondents in training datasets. Thus, sub-sampling of non-respondent class data is the most widely used method to balance the datasets. However, random sampling allows “important” patterns near the decision boundary to be missed. Those patterns are likely to become support vectors. Loss of those patterns could result in a poor generalization performance of SVM. Thus, instead, we propose to employ different misclassification costs to different class errors in the objective function, which is naturally allowed in SVM. This approach is not only safer, but also more principled.

Let m_1 and m_2 denote the size of class 1 and class 2 data sets, respectively, with

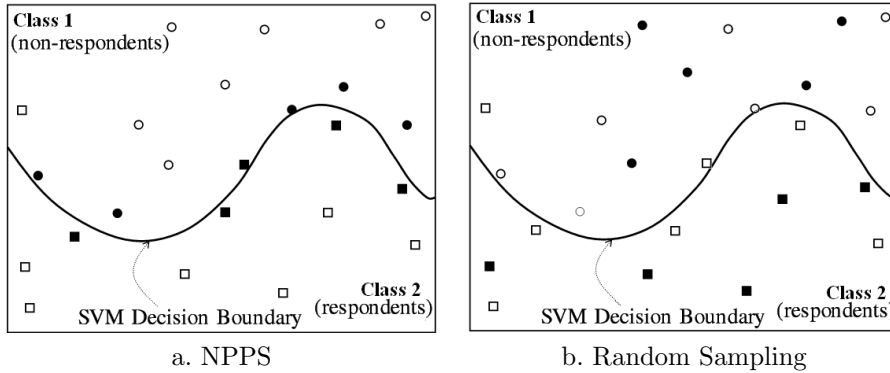


Figure 9.1: NPPS and random sampling select different subsets: outlined circles and squares are the patterns belonging to class 1 (non-respondents’ group) and class 2 (respondents’ group), respectively. Black solid circles and squares are the selected patterns.

$m_1 \gg m_2$ and $M = m_1 + m_2$. One way to alleviate data imbalance problem is to assign to a large class a smaller cost while assign to a small class a larger cost, which assures that a small class is not “neglected.” In response modeling, there are many more non-respondents than respondents, thus the size of non-respondents is m_1 while that of respondents is m_2 . One way to accomplish it is to define and assign C_1 and C_2 to each class as below

$$\Theta(\vec{w}, \xi) = \frac{1}{2} \|\vec{w}\|^2 + C_1 \sum_{i \in \text{nonrespondents}} \xi_i + C_2 \sum_{i \in \text{respondents}} \xi_i, \quad (9.1)$$

where C_1 and C_2 are defined respectively as

$$\begin{aligned} C_1 &= \frac{m_2}{M} \cdot C, \\ C_2 &= \frac{m_1}{M} \cdot C. \end{aligned} \quad (9.2)$$

In order to emphasize small respondent data set, a larger cost C_2 is assigned to its error term. Constant C is the original cost term used in Eq.(3.1).

9.3.3 Getting Scores from an SVM Classifier

The objective of response modeling is to compute the likelihood or propensity of each customer to respond to a particular offer so that the mailing response or profit is maximized. Lift chart is commonly used for this purpose, which sorts the customers by the descending order of their estimated value (score), and then the customers in the first several deciles are finally decided to be mailed. Although an SVM classifier returns a binary output (-1 or 1) as shown in Eq. (3.5), one can still estimate a score based on the *distance between a pattern and the decision boundary*. In other words, we assume that a pattern located further from the decision boundary has a higher probability of belonging to that class. The decision boundary hyperplane $\bar{f}(\vec{x})$ in a feature space Φ is represented as

$$\bar{f}(\vec{x}) = \sum_{i \in \mathbf{SV}_s} y_i \alpha_i \Phi(\vec{x}_i) \cdot \Phi(\vec{x}) + b = 0 \quad (9.3)$$

from Eq. (3.5). It should be noted that the decision boundary is a hyperplane in the feature space Φ even though it is a nonlinear hyper-surface in the input space. In the feature space, hence, the distance from a pattern $\Phi(\vec{x})$ to the decision boundary hyperplane $\bar{f}(\vec{x})$ can be calculated by

$$\text{dist}(\Phi(\vec{x}), \bar{f}(\vec{x})) = \frac{|\bar{f}(\vec{x})|}{\left| \sum_{i \in \mathbf{SV}_s} y_i \alpha_i \Phi(\vec{x}_i) \right|^2}. \quad (9.4)$$

The exact value of the distance is not possible to obtain since the actual mapping function $\Phi(\cdot)$ is not known. The feature space Φ is an infinite dimensional space, and multiple mapping functions $\Phi(\cdot)$ s can exist. In actual SVM training, a kernel function $K(\vec{x}, \vec{x}')$ replaces $\Phi(\vec{x}) \cdot \Phi(\vec{x}')$. Fortunately, however, one does not need to know the exact value of the distance. Instead, only a relative score or rank is all that is required. Since the denominator in Eq. (9.4) is common for all patterns, the signed function value in the numerator, $\bar{f}(\vec{x})$, can be used in computing ranks. The larger the value of $\bar{f}(\vec{x})$, the lower the rank of that particular customer's likelihood becomes.

9.4 Experiments

This section provides the empirical results of SVM based response modeling with the proposed approach. In particular, the performance evaluation measures pertinent to response modeling are also proposed and measured.

9.4.1 Dataset

In machine learning literature, so-called standard and public datasets are used. But, in response modeling, or in direct marketing for that matter, such datasets do not seem to exist. Many papers use a unique dataset which is not available for other researchers. The only exception seems to be datasets from the Direct Marketing Educational Foundation (DMEF) (*The Direct Marketing Association*). The DMEF makes marketing datasets available to researchers. Dataset DMEF4, was used in various researches (Ha *et al.*, 2004; Malthouse, 2001, 2002). It is concerned with an up-scale gift business that mails general and specialized catalogs to its customer base several times each year. The problem is to estimate how much each customer will spend during the test period, 09/1992–12/1992, based on the training period, 12/1971–06/1992. There are 101,532 patterns in the dataset, each of which represents the purchase history information of a customer. Each customer is described by 91 input variables. A subset of 17 input variables, some original and some derived, were employed just as in (Malthouse, 2001) (see table 9.1). The dataset has two target variables, TARGDOL (target mailing dollars) and TARGORD (target mailing orders). The former indicates the purchase dollar amount during the test period, and the latter indicates the number of orders during the test period. The TARGDOL or the TARGORD could be directly estimated by building a regression model. *Malthouse (2001)* built a regression model to estimate the value of TARGDOL. But due to the problems of regression (section 9.2), we formulated the problem into a classification one. A new target variable, RESPONSE, was defined as follows: 1 if TARGDOL (TARGORD) > 0, 0 otherwise. *Ha et al. (2004)* used the same derivation to fit a neural network classifier. Thus, all the customers were categorized into either a non-respondent (class 1) or a respondent (class 2). The response rate is 9.4%, which means the class distribution of the dataset is highly imbalanced.

Table 9.1: Input Variables

Variable	Formula	Description
<i>Original Variables</i>		
purseas		number of seasons with a purchase
falord		life-to-date (LTD) fall orders
ordtyr		number of orders this year
puryear		number of years with a purchase
sprord		LTD spring orders
<i>Derived Variables</i>		
recency		order days since 10/1992
tran38	$1/\text{recency}$	
tran51	$0 \leq \text{recency} < 90$	
tran52	$90 \leq \text{recency} < 180$	five dummy variables (tran51–55)
tran53	$180 \leq \text{recency} < 270$	having the value 1, if the condition
tran54	$270 \leq \text{recency} < 366$	is satisfied, otherwise the value 0
tran55	$366 \leq \text{recency} < 730$	
comb2	$\sum_{i=1}^{14} \text{prodgrp } i$	number of product groups purchased from this year
tran25	$1 / (1 + \text{lorditm})$	inverse of latest-season items
tran42	$\log(1 + \text{ordtyr} \times \text{falord})$	interaction between the number of orders
tran44	$\sqrt{\text{ordhist} \times \text{sprord}}$	interaction between LTD orders and LTD spring orders
tran46	$\sqrt{\text{comb2}}$	

Table 9.2: SVM models: the number of patterns selected from NPPS slightly varies with the given set of each fold, thus it is represented as an average over the five reduced training sets.

Model	Size of Training Data	Training Data
R05-SVM	4060	5% random samples
R10-SVM	8121	10% random samples
R20-SVM	16244	20% random samples
R40-SVM	32490	40% random samples
R60-SVM	48734	60% random samples
R80-SVM	64980	80% random samples
R100-SVM	81226	100% random samples
S-SVM	avg. 8871	the patterns selected by NPPS

9.4.2 SVM Models

To verify the effectiveness of NPPS, we considered seven SVMs trained with randomly selected patterns. They are denoted as R*-SVM where ‘*’ indicates the ratio of random samples drawn without replacement. S-SVM denotes the SVM trained with the patterns selected by NPPS (see table 9.2). Each model was trained and evaluated using five-fold cross-validation. The number of neighbors (k) of NPPS, was set to 4 according to guidelines suggested in Shin and Cho (2003c). All the SVM models in table 9.2 use the same hyper-parameter values to equalize their effects. The RBF kernel in Eq. (3.2) was used with parameter σ set to 0.5, and the misclassification tolerance parameter C in Eq. (3.1) set to 10. These parameter settings were determined through a trial-error approach over the combination of C and σ , ($\{0.1, 1, 10, 100, 1000\} \times \{0.25, 0.5, 1, 2, 3\}$), using ten fold cross-validation performance. The class imbalance problem addressed in section 9.3.2 appeared in all the eight datasets. The sets selected by random sampling showed the common class ratio of $m_1 : m_2 = 90.6\% : 9.4\%$. That is also the same ratio as the original training set since we conducted a stratified random sampling by the target variable. The training set reduced by NPPS, however, showed a different class ratio, $m_1 : m_2 = 65.5\% : 34.5\%$ ($= 5810 : 3061$) on average. Even though NPPS improved the ratio of the smaller class from 9.4% up to 34.5%, the imbalance problem still remained. Thus, the different misclassification costs, C_1 and C_2 were set on every dataset as they were defined in Eq. (9.2). C_1 and C_2 of R*-SVM were 0.94 ($= 0.094 \times 10$) and 9.06 ($= 0.906 \times 10$), respectively. On the other hands, those of S-SVM were 3.45 ($= 0.345 \times 10$) and 6.55 ($= 0.655 \times 10$).

Table 9.3: Confusion Matrix: FP, FN, TP and TN means false positive, false negative, true positive, and true negative in due order where TP and TN are the correct classification.

		<i>Classified</i>		
		class 1 (non-respondent)	class 2 (respondent)	
<i>Actual</i>	class 1 (non-respondent)	m_{11} (TN)	m_{12} (FP)	m_1
	class 2 (respondent)	m_{21} (FN)	m_{22} (TP)	m_2

9.4.3 Performance Measurements

The performances of the eight SVM response models were compared in terms of three criteria: accuracies, lift chart and computational efficiency.

9.4.3.1 Accuracies

The accuracy of a classifier can be described by a confusion matrix (see table 9.3). Let m_{ij} denote the number of patterns which were classified as class j but whose actual class label is class i . A most widely used accuracy measurement is an Average Correct-classification Rate (ACR) which is defined as

$$\text{Average Correct-classification Rate (ACR)} = \frac{TN+TP}{M} = \frac{m_{11}+m_{22}}{M}.$$

But, the average correct-classification rate can be misleading in an imbalanced dataset where the heavily-represented class is given more weight. Receiver Operating Characteristic (ROC) analysis is usually performed as well Provost and Fawcett (1997), which measures the classifier's accuracy over the whole range of thresholds in terms of Specificity (Sp) and Sensitivity (Se) SAS Institute (1998). They are defined as

$$\begin{aligned} \text{Specificity (Sp)} &= \frac{TN}{TN+FP} = \frac{m_{11}}{m_{11}+m_{12}} = \frac{m_{11}}{m_1}, \\ \text{Sensitivity (Se)} &= \frac{TP}{FN+TP} = \frac{m_{22}}{m_{21}+m_{22}} = \frac{m_{22}}{m_2}. \end{aligned}$$

Since we fixed the classification threshold at 0 in the SVM decision function Eq. (3.5), however, only one pair of Sp and Se per model was available. Thus,

here the ROC plot has the eight pairs of (1-Sp, Se) scattered for their comparison. Another accuracy measure, Balanced Correct-classification Rate (BCR), was defined so as to incorporate Sp and Se into one term. BCR enforces balance in the correct classification rate between two classes. It is defined as

$$\text{Balanced Correct-classification Rate (BCR)} = \text{Sp} \cdot \text{Se} = \left(\frac{m_{11}}{m_1}\right) \cdot \left(\frac{m_{22}}{m_2}\right).$$

9.4.3.2 Lift Chart Analysis of Response Rate and Profit

Once the test patterns were sorted in a descending order according to $\bar{f}(\vec{x})$, two kinds of lift charts were investigated. One is for *response rate*, and the other for *profit*. From the business point of view, the ultimate goal of direct mailing is to maximize the profit rather than the response rate itself Malthouse (1999). Thus we evaluated the eight competing SVM models from a profit aspect as well. For profit lift chart analysis, another target variable of DMEF4 dataset, *TARGDOL* (*target mailing dollar*), was associated with the rank of $\bar{f}(\vec{x})$, which indicates the purchase dollar amount during the test period. Two measurements were used in evaluating lift charts. One is the average response rate or profit in the top decile, “Top-Decile”. This measures how well two model identifies a small number of highly likely respondents. The other is “Weighted-Decile” defined as

$$\text{Weighted-Decile} = \frac{\{1.0 \times d_1 + 0.9 \times d_2 + 0.8 \times d_3 \dots + 0.1 \times d_{10}\}}{1.0 + 0.9 + 0.8 + \dots + 0.1},$$

where d_i , ($i = 1, \dots, 10$) is a cumulative average response rate or profit till i^{th} decile in the lift table. This measures how well the model identifies a larger number of likely respondents in a larger rollout. A similar evaluation by two measurements has been adopted in data mining competitions Ling and Li (1998).

9.4.3.3 Computational Efficiency

The evaluation was done in several measures: the number of training patterns, training time, and the number of support vectors, and recall time. The number of patterns directly influences the time complexity. The training time of

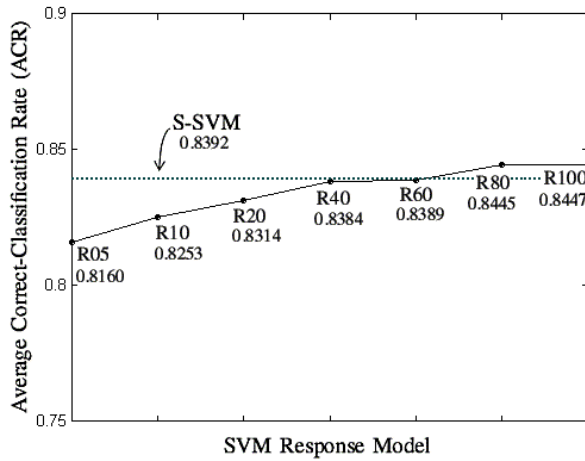
SVM increases in proportion to the cube of the number of training patterns (in case of standard QP solver). The recall time increases linearly to the number of support vectors. Training time is of important concern to a direct marketer who is in charge of SVM modeling with a huge amount of data, while recall time is critical when the model is deployed to work in a real-time application such as fraud detection. Although recall time is not a primary issue in response modeling, we measured it for potential use to another application.

9.4.4 Results

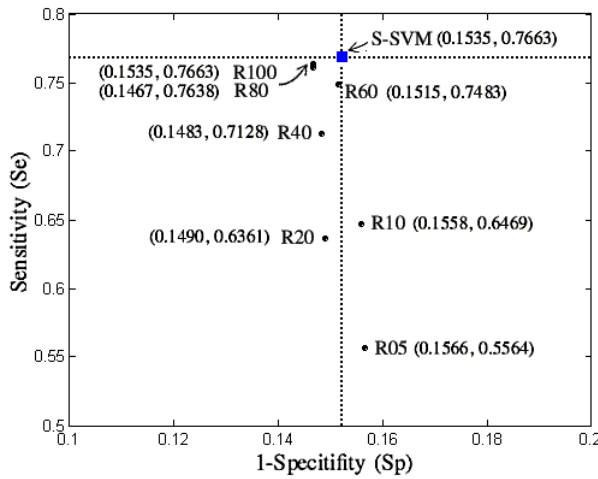
We now give the experimental results of the eight SVM response models in the order of accuracies, lift chart analysis, and computational efficiency.

Fig. 9.2 shows how the eight SVM response models performed in terms of ACR, ROC, and BCR. First, Fig. 9.2.a indicates a mean ACR over five-fold cross-validation of each SVM model. For the sake of convenience, R*-SVM is briefly denoted as 'R*' in the figure. Sampling more patterns results in higher ACR, but the increasing rate is not very high. From R05 to R100, only about 3.52% ($=\{0.8447 - 0.8160\}/0.8160 \times 100\%$) of accuracy was gained from 1,900% ($=\{100 - 5\}/5 \times 100\%$) data increase. The S-SVM achieved ACR in the range of those from R60–R80. However, we could not make good evaluation of the model comparison using ACR because of class imbalance. In Fig. 9.2.b, the eight pairs of (1-Sp, Se) were plotted in ROC chart. A point located upper left corresponds to a better performance. The ACR is effectively broken down into two classwise accuracies, Sp for non-respondents (class 1) and Se for respondents (class 2). The Sp's of the eight SVM models are similar, while the Se's show a significant differences. It should be noted that it is Se, accuracy for respondents' group, that is of greater importance to direct marketers, since their primary goal is to identify the respondents, not the non-respondents. S-SVM achieved a best Se, better than that of even R100-SVM. Fig. 9.2.c shows the BCRs of the eight SVM response models. BCR clearly distinguished the accuracies of the eight SVM models. Sampling more data results in a larger BCR also. The BCR of S-SVM is almost same as that of R100-SVM.

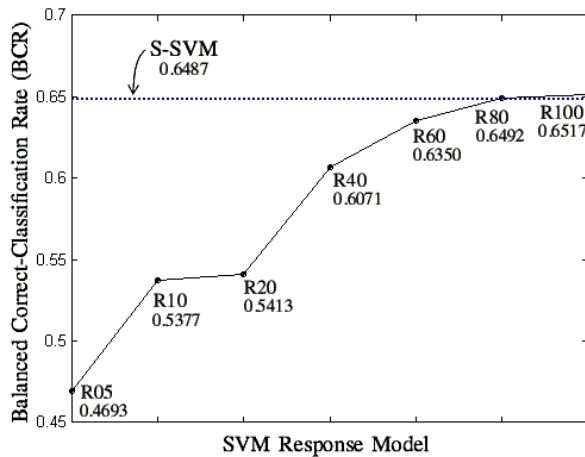
Fig. 9.3 illustrates the lift chart of the cumulative average response rate. The base average response rate of DMEF4 dataset was 9.4%, which is represented as a solid horizon at the bottom of the chart. Two observations can be made. First, all the SVM response models did better than the base response rate. Second, more training patterns lead to a better lift of the response rate. R100-SVM



a. ACR



b. ROC



c. BCR

Figure 9.2: Accuracies: accuracy of R*-SVM is depicted as a solid circle while that of S SVM is represented as a dotted reference line.

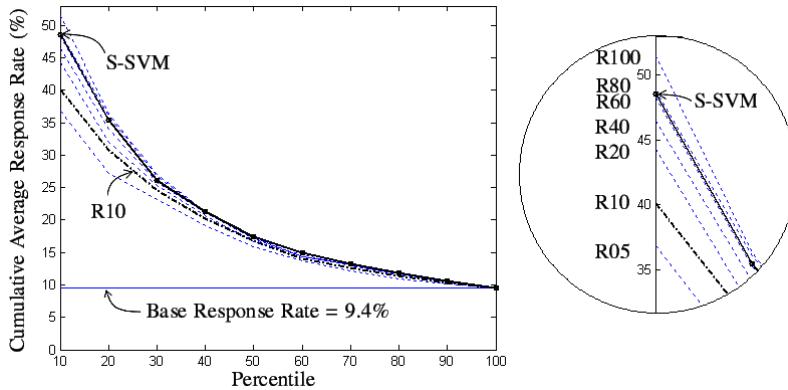


Figure 9.3: Lift chart of cumulative average response rate: R^* -SVMs are depicted dotted lines but among them R10-SVM is represented as a dash-dot line. S-SVM is represented as a solid-dot line.

showed the best performance while the R05-SVM showed the worst. Models trained with more patterns showed a steeper lift in the first several deciles. The lift curve of S-SVM was almost identical to that of R80-SVM. It is illuminating to compare the curve shape of S-SVM with that of R10-SVM represented as a dash-dot line. Although the two models had almost the same number of training patterns, they were significantly different in the lift performance. Fig. 9.4 shows the results of the lift measures described in section 9.4.3.2: Top-Decile and Weighted-Decile. From the top 10 percentile of customers, R100-SVM obtained 51.45% response rate (see Fig. 9.4.a). The Top-Decile response rate of S-SVM was 48.65%, which is almost equal to that of R80-SVM, 48.79%. Fig. 9.4.b shows the results of Weighted-Decile response rates. R100-SVM still did best, and S-SVM and R80-SVM came second. But the gap between the first and the second was not so big as in the Top-Decile response rate.

Now, Fig. 9.5 and Fig. 9.6 describe the lift chart results in terms of the profit. The average purchase dollar amount of DMEF4 was \$48 when averaged over the respondents' group, but \$4.5 when averaged over all customers. The horizontal line in the lift chart of Fig. 9.5 represents the \$4.5 base average profit. All the models did better than the base average profit and an SVM with more training patterns produced a higher profit in the first several deciles. But in terms of the profit lift, S-SVM showed a performance comparable to that of R100-SVM. It is also remarkable that the profit lifts of R100-SVM or S-SVM outperformed those of *Malthouse* who got the scores by fitting the problem as a regression one (Malthouse, 2001). For the cumulative average profit (dollars) of the second decile, *Malthouse*' regression model recorded \$12–\$15 while the SVM classifica-

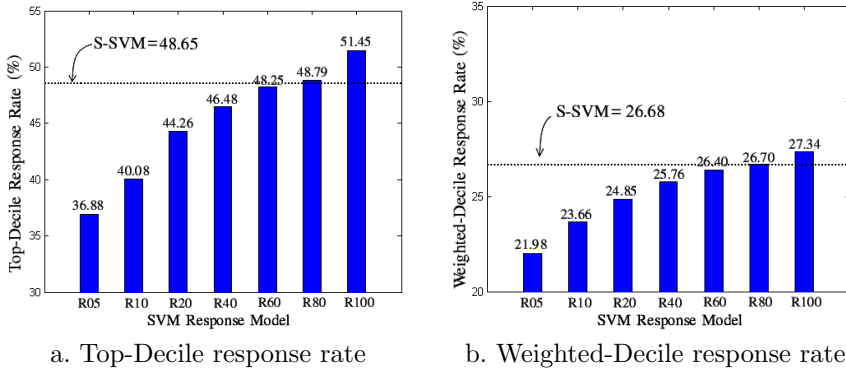


Figure 9.4: Top-Decile response rate and Weighted-Decile response rate: R*-SVM is depicted as a bar while S-SVM is represented as a dotted reference line.

tion model recorded \$17–\$18. Fig. 9.6 illustrates the Top-Decile profit and the Weighted-Decile profit. The Top-Decile profit and the Weighted-Decile profit of R100-SVM were \$23.78 and \$12.99, respectively, and those of R80-SVM were \$22.25 and \$12.56. S-SVM were \$23.53 in the Top-Decile profit and \$12.77 in the Weighted-Decile profit, which were slightly less than those of R100-SVM but more than those of R80-SVM.

Finally, table 9.4 shows the results of computational efficiency measures in columns: the number of training patterns, training time, the number of support vectors, its proportion to training patterns, and recall time. We used OSU SVM Classifier Matlab Toolbox, which is a hybrid algorithm of SMO and SVM^{light}, and is known as one of the fastest solvers Hearst *et al.* (1997). Training time increased proportionally to the number of training patterns with the peak of 4,820 (sec) for R100-SVM. On the other hand, S-SVM took only 68 (sec). The total time of S-SVM was 129 (sec), when the NPPS running time, 61 (sec), was included. Note that SVM training is usually performed several times to find a set of optimal parameters, but the pattern selection is performed only once. In the fourth column, the number of support vectors is represented. At most, half of the random sampling training patterns were support vectors while 74% of the NPPS selected training patterns were support vectors. The result confirms that the NPPS' selection of training patterns was more efficient. Recall time was proportional to the number of support vectors as shown in the last column. Overall, the computational efficiency of S-SVM was comparable to that of R10-SVM or R20-SVM.

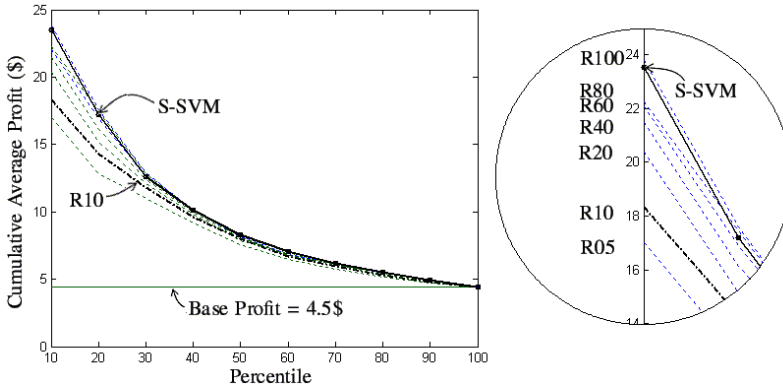


Figure 9.5: Lift chart of cumulative average profit: R*-SVMs are depicted dotted lines but among them R10-SVM is represented as a dash-dot line. S-SVM is represented as a solid-dot line.

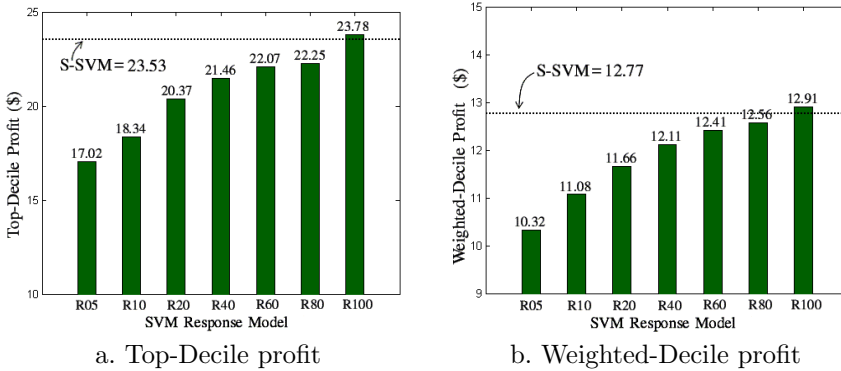


Figure 9.6: Top-Decile profit and Weighted-Decile profit: R*-SVM is depicted as a bar while S-SVM is represented as a dotted reference line.

Table 9.4: Computational Efficiency of SVM response models

	Num. of Training Patterns	Training Time (sec)	Num. of SVs (proportion)	Recall Time (sec)
R05	4,060	13.72	1,975 (48.65%)	17.22
R10	8,121	56.67	4,194 (51.64%)	31.39
R20	16,244	149.42	7,463 (45.94%)	56.17
R40	32,490	652.11	14,967 (46.07%)	112.08
R60	48,734	1,622.06	22,193 (45.54%)	166.11
R80	64,980	2,906.97	28,968 (44.58%)	237.31
R100	81,226	4,820.06	35,529 (43.74%)	381.31
S	8,871	68.29	6,624 (74.67%)	45.13

9.5 Summary

This chapter exemplifies a practical usage of the proposed method by applying to a real problem in marketing domain - response modeling in direct marketing. We diagnose the domain specific difficulties which can arise in practice when applying SVM to response modeling, then proposed how to alleviate and solve those difficulties: informative sampling, different costs for different classes, and use of distance to decision boundary. In the experiments, we showed that the proposed solutions worked quite well. In particular, several models were trained and evaluated in terms of accuracies, lift chart analysis and computational efficiency. The SVM trained with the patterns selected by proposed NPPS (S-SVM) were compared with the ones trained with random samples (R*-SVMs where ‘*’ indicates the sampling percentage). Fig. 9.7 summarizes the results in terms of various measures. The horizontal bars in the figure shows the performance of S-SVM relative to those of R*-SVMs in various measures. S-SVM achieved the accuracies and uplifts comparable to those of R80-SVM and R100-SVM with a computational cost comparable to those of R10-SVM and R20-SVM.

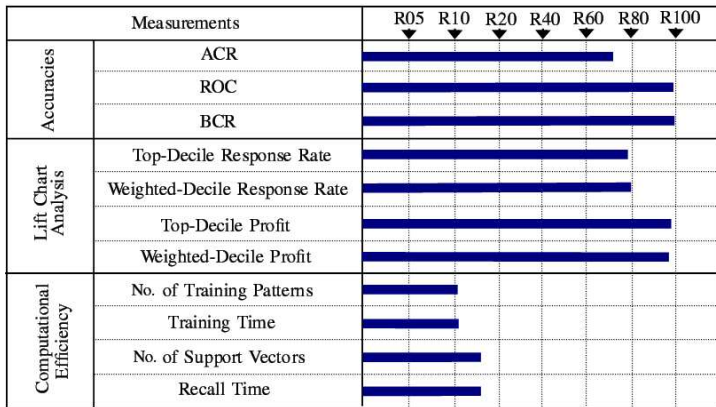


Figure 9.7: How well S-SVM performed relative to R*-SVMs

Conclusion and Discussion

In the thesis, we proposed to select the patterns near the decision boundary based on the neighborhood properties in order to relieve the computational burden in SVM training. We utilized k nearest neighbors to look around the pattern's periphery. The *first* neighborhood property is that "a pattern located near the decision boundary tends to have more heterogeneous neighbors in their class-membership." The *second* neighborhood property dictates that "an overlap or a noisy pattern tends to belong to a different class from its neighbors." The first one is used for identifying those patterns located near the decision boundary. The second one is used for removing the patterns located on the wrong side of the decision boundary. These properties were first implemented as a naive form with time complexity $O(M^2)$ where M is the number of given training patterns. Then, to accelerate the pattern selection procedure we utilized another property. The *third* neighborhood property is that "the neighbors of a pattern located near the decision boundary tend to be located near the decision boundary as well." The third one skips calculation of unnecessary distances between patterns. This advanced form of algorithm, *fast* NPPS, has a practical time complexity of $O(M^2)$ where v is the number of patterns located in the overlap region. The number of patterns located in the overlap region, v , is closely related to determine a parameter of NPPS, the number of neighbors, k , accordingly we provided a heuristic method to set the value of k . Then, we proved invariance of the neighborhood relation under the input to feature space mapping, which assures that the patterns selected by NPPS in input space are

likely to be located near decision boundary in feature space in where SVs are defined. Through experiments on the synthetic as well as the real-world problems, we empirically showed the efficiency of the proposed algorithms. SVMs with NPPS were trained much faster with fewer redundant support vectors, with little loss if any in test performance of most datasets. In addition, to exemplify a practical usage of the proposed method, we applied NPPS to marketing domain problem. SVM with NPPS achieved the accuracies and uplifts comparable to those of SVM with 80–100% random samples, while the computational cost was comparable to those of SVM with 10–20% random samples.

Here, we would like to address some limitations and future work.

1. *Fast* NPPS is geared up for efficient searching with initial random samples (see Fig. 5.2). A small proportion of random sampling, however, may cause a problem when the class distribution is multi-modal. A small lump of patterns rather isolated could be excluded from the searching unless an initial random sample hits one of them. A current remedy for the risk is to check the class distribution first. If the class distribution falls on a multi-modal case, then we recommend to use *naive* NPPS instead. Actually, we implemented *fast* NPPS to be able to easily switch to *naive* NPPS just by setting the sampling ratio to 1.
2. NPPS were developed under the assumption that the classes are overlapped (a non-separable case). Therefore, if one class is remote and clearly separable from the other, an empty set will be returned as a selected pattern set after the first iteration. And also the algorithms could not guarantee the original performance of SVM when the pattern is stored as a sparse vector. We encountered such a case in 1-8 SVM of MNIST data set. The patterns in ‘1’ digit class are known as sparse vectors. Moreover, the class is linearly separable from all other digit classes (Liu and Nakagawa, 2001; Platt, 1999). Thus it requires to see if the selected pattern set is too small or unbalanced. If so, training with all patterns is desirable.
3. To estimate the number of the patterns in the overlap region, we assumed that both sides of the decision boundary within that region contain roughly the same number of correct and incorrect patterns (in section 6.3). Therefore, further extension for more general cases needs to be pursued.
4. It is not easy to find the right value for β in Selecting Condition. Parameter β controls the selectivity by k NN classifier. The larger value of β leads to the smaller number of selected patterns. But, the performance of k NN classifier decreases as the dimensionality increases if k is fixed. Therefore, in low dimensional space, β can be set to a large value (i.e. $\beta = 1$), since k NN classifier performs well. But in high dimensional space, a large value

of β is somewhat risky. Empirically, $\beta = 1$ was not problematic up to about 20 dimensional space. The result coincides with the recommendation in (Mitchell, 1997).

5. An extension or a modification of the procedure of “determining the value of k ” is necessary to handle the high dimensionality. A higher dimensionality usually requires a larger k value. However, we have not yet considered this dimensionality-related factor in the procedure (see Eq. (6.13)). The proposed procedure gave a good estimate of k at a moderately high dimensional space of about 10 to 20 (i.e., Pima Indian Diabetes, Wisconsin Breast Cancer), but it did underestimate k at a fairly high dimensional space of 784 as in the case of MNIST datasets. The experiment involving MNIST datasets resulted in a k value less than 10 while a much better SVM performance was obtained with a k value larger than 50.
6. SVM solves a multi-class problem by divide-and-combine strategy which divides the multi-class problem into several binary sub-problems (e. g., one-versus-others or one-versus-one), and then, combines the outputs. This led us to apply our algorithm only to binary class problems in our current research. However, the proposed pattern selection algorithm may be extended without major correction to multi-class problems (see Fig. 4.1, Fig. 4.3 and Fig. 5.2).
7. The current version of NPPS works for classification problems only, thus is not applicable to regression problems. In a regression problem, the patterns located away from others, such as outliers, are less important to learning. Thus, a straightforward idea would be to use the mean (μ) and variance (Σ) of k nearest neighbors’ outputs. A pattern having a small value of Σ can be replaced by μ of its neighbors and itself. That is, $k + 1$ patterns can be replaced by one pattern. On the contrary, a pattern having a large value of Σ can be totally eliminated, and its neighbors will be used for the next pattern searching. A similar research was conducted in *Shin and Cho (2002)* based on ensemble neural network, but more extended study based on k nearest neighbors is still under consideration.

Bibliography

- 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.
- Arya, S., Mount, D. M., Netanyahu, N. S. and Silverman, R., 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, vol. 45, no. 6, pp. 891–923.
- Bentley, J. L., 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of ACM*, vol. 18, pp. 509–517.
- Berchtold, S., Böhm, C., Keim, D.A. and Kriegel, H.-P., 1997. A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space. *Proc. of Annual ACM Symposium on Principles Database Systems*, pp. 78–86.
- Bentz, Y. and Merunkay, D. 2000. Neural Networks and the Multinomial Logit for Brand Choice Modeling: a Hybrid Approach. *Journal of Forecasting*, vol. 19, no. 3, pp. 177–200.
- Borodin, A., Ostrovsky, R. and Rabani, Y., 1999. Lower Bound for High Dimensional Nearest Neighbor Search and Related Problems. *Proc. of the 31st ACM Symposium on Theory of Computing*, pp. 312–321.
- Bounds, D. and Ross, D. 1997. Forecasting Customer Response with Neural Networks. *Handbook of Neural Computation*, G6.2, pp. 1–7.
- Brinker, K. 2003. Incorporating Diversity in Active Learning with Support Vector Machines. *Proc. of the 20th International Conference on Machine Learning (ICML)*, pp. 59–66, AAAI Press.

- Byun, H. and Lee, S. 2002. Applications of Support Vector Machines for Pattern Recognition: A Survey. *International Workshop on Pattern Recognition with Support Vector Machines (SVM2002)*, *Lecture Notes in Computer Science (LNCS 2388)*, Niagara Falls, Canada, pp. 213–236.
- Campbell, C., Cristianini, N., Smola, 2000. Query learning with Large Margin Classifiers. *Proc. of the 17th International Conference on Machine Learning (ICML)*, pp. 111–118, Morgan Kaufmann.
- Cauwenberghs, G. and Poggio, T. 2001. Incremental and Decremental Support Vector Machine Learning. *Advances in Neural Information Processing Systems*, Cambridge MA: MIT Press, vol. 13, pp. 409–415.
- Cheung, K.-W., Kwok, J. K., Law, M. H. and Tsui, K.-C. 2003. Mining Customer Product Rating for Personalized Marketing. *Decision Support Systems*, vol. 35, pp. 231–243.
- Chiu, C. 2002. A Case-Based Customer Classification Approach for Direct Marketing. *Expert Systems with Applications*, vol. 22, pp. 163–168.
- Choi, S. H. and Rockett, P., 2002. The Training of Neural Classifiers with Condensed Dataset. *IEEE Transactions on Systems, Man, and Cybernetics-PART B: Cybernetics*, vol. 32, no. 2, pp. 202–207.
- Coenen, F., Swinnen, G., Vanhoof, K. and Wets, G. 2000. The Improvement of Response Modeling: Combining Rule-Induction and Case-Based Reasoning. *Expert Systems with Applications*, vol. 18, pp. 307–313.
- Colombo, R., and Jiang, W. 1999. A Stochastic RFM Model, *Journal of Interactive Marketing*, vol. 13, no. 3., pp. 1–12.
- Cristianini, N. and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines and other kernel-based Learning Methods*, Cambridge University Press.
- Deichmann, J., Eshghi, A., Haughton, D., Sayek, S. and Teebagy, N. 2002. Application of Multiple Adaptive Splines (MARS) in Direct Response Modeling. *Journal of Interactive Marketing*, vol. 16, no. 4., pp. 15–27.
- DeCoste D., Mazzoni, D. 2003. Fast Query-Optimized Kernel Machine Classification via Incremental Approximate Nearest Support Vectors. *Proc. of the 20th International Conference on Machine Learning (ICML)*, pp. 115–122, AAAI Press.
- Dumais, S. 1998. Using SVMs for Text Categorization, *IEEE Intelligent Systems*, pp. 21–23.

- Ferri, F. J., Albert, J. V. and Vidal, E., Considerations About Sample-Size Sensitivity of a Family of Edited Nearest-Neighbor Rules. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 29, no. 4, pp. 667–672.
- 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.
- Freund, Y. and Schapire, R. 1996. Experiments with a New Boosting Algorithm *Proc. of the Thirteenth International Conference on Machine Learning*, pp. 148–156.
- Friedman, J. H., Bentley, J. L., Finkel, R. A., 1977. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226.
- Gates, G. W., 1972. The Reduced Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, vol. IT-18, no. 3, pp. 431–433.
- Grother, P. J., Candela, G. T. and Blue, J. L, 1997. Fast Implementations of Nearest Neighbor Classifiers. *Pattern Recognition*, vol. 30, no. 3, pp. 459–465.
- Guttman, A., 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 47–57.
- Ha, K., Cho, S., and MacLachlan, D., 2004. Response Models Based on Bagging Neural Networks. (to appear).
- Hara, K. and Nakayama, K., 1998. Training Data Selection Method for Generalization by Multilayer Neural Networks. *IEICE Trans. Fund.*, vol. E81-A, no. 3, pp. 374–381.
- Hara, K. and Nakayama, K., 2000. A Training Method with Small Computation for Classification. *Proc. of the IEEE-INNS-ENNS International Joint Conference*, vol. 3, pp. 543–548.
- Hart, P. E., 1968. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, vol. IT-14, no. 3, pp. 515–516.
- Haughton, D. and Oulabi, S. 1997. Direct Marketing Modeling with CART and CHAID. *Journal of Direct Marketing*, vol. 11, no. 4, pp. 42–52.
- Hearst, M. A., Schölkopf, B., Dumais, S., Osuna, E., and Platt, J. 1997. Trends and Controversies - Support Vector Machines. *IEEE Intelligent Systems*, vol. 13, pp. 18–28.

- Heisele, B., Poggio, T., and Pontil, M. 2000. Face Detection in Still Gray Images. *Technical Report AI Memo 1687*, MIT AI Lab.
- Hosmer, D. W. and Lemeshow, S. 1989. *Applied Logistic Regression*, John Wiley & Sons, Inc.
- Indyk, P., 1998. On Approximate Nearest Neighbors in Non-Euclidean Spaces. *Proc. of IEEE Symposium on Foundations of Computer Science*, pp. 148–155.
- Japkowicz, N. 2000. Learning from Imbalanced Data Sets: A Comparison of Various Strategies. *In AAAI Workshop on Learning from Imbalanced Data Sets*, Menlo Park, CA, AAAI Press.
- Joachims, T. 1998. Text categorization with Support Vector Machines: Learning with Many Relevant Features. *Proc. of 10th European Conference on Machine Learning*, pp. 137–142.
- Johnson, M. H., Ladner, R. E. and Riskin, E. A., 2000. Fast Nearest Neighbor Search of Entropy-Constrained Vector Quantization. *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1435–1437.
- Kernel-Machines.org. <http://www.kernel-machines.org/>
- Kleingberg, J. M., 1997. Two Algorithms for Nearest-Neighbor Search in High Dimensions. *Proc. of 29th ACM symposium on Theory of Computing*, pp. 599–608.
- Koggalage, R. and Halgamuge, S., 2004. Reducing the Number of Training Samples for Fast Support Vector Machine Classification. *Neural Information Processing-Letters and Reviews*, vol. 2, no. 3, pp. 57–65.
- Lee, C. and Landgrebe, D. A., 1997. Decision Boundary Feature Extraction for Neural Networks. *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 75–83.
- Lee, K. K., Gunn, S. R., Harris, C. J, and Reed, P. A. S. 2001. Classification of Imbalanced Data with Transparent Kernels. *Proc. of INNS-IEEE International Joint Conference on Neural Networks*, pp. 2410–2415.
- Lee, Y. and Mangasarian O.L., 2001. RSVM: Reduced Support Vector Machines. *CD Proc. of the SIAM International Conference on Data Mining*, Chicago, April, SIAM, Philadelphia. (Available from <http://www.cs.wisc.edu/olvi/olvi.html>.)
- Leisch, F., Jain, L. C. and Hornik, K., 1998. Cross-Validation with Active Pattern Selection for Neural-Network Classifiers. *IEEE Transactions on Neural Networks*, vol. 9, pp. 35–41.

- Lin, H.-T. and Lin, C.-J., 2003. A study on sigmoid kernels for SVM and the training of non-PSD Kernels by SMO-type methods. *available at <http://www.csie.ntu.edu.tw/~cjlin/papers.html>*.
- Ling, C. X. and Li, C. 1998. Data Mining for Direct Marketing: Problems and Solutions. *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp. 73–79.
- Liu C. L., and Nakagawa M. 2001. Evaluation of Prototype Learning Algorithms for Nearest-Neighbor Classifier in Application to Handwritten Character Recognition. *Pattern Recognition*, vol. 34, pp. 601–615.
- 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.
- Malthouse, E. C. 1999. Ridge Regression and Direct Marketing Scoring Models. *Journal of Interactive Marketing*, vol. 13, no. 4, pp. 10–23.
- Malthouse, E. C. 2001. Assessing the Performance of Direct Marketing Models. *Journal of Interactive Marketing*, vol. 15, no. 1, pp. 49–62.
- Malthouse, E. C. 2002. Performance-Based Variable Selection for Scoring Models. *Journal of Interactive Marketing*, vol. 16, no. 4, pp. 10–23.
- Masuyama, N., Kudo, M., Toyama, J. and Shimbo, M., 1999. Termination Conditions for a Fast k -Nearest Neighbor Method. *Proc. of 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems*, Adelaide, Australia, pp. 443–446.
- Mitchell, T. M., *Machine Learning*, McGraw Hill, 1997. See also Lecture Slides on Chapter 8 for the Book at <http://www-2.cs.cmu.edu/~tom/mlbook-chapter-slides.html>.
- MNIST Dataset. <http://yann.lecun.com/exdb/mnist/>
- Moghaddam, B., and Yang, M. H. 2000. Gender Classification with Support Vector Machines. *Proc. of International Conference on Pattern Recognition*, Barcelona, Spain, 2000, and also appeared in *Proc. of 4th IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, pp. 306–311.
- Moutinho, L., Curry, B., Davies, F., and Rita, P. 1994. *Neural Network in Marketing*, New York: Routledge.
- Osuna, E., Freund, R., and Girosi, F. 1997. Training Support Vector Machines: An Application to Face Detection. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130–136.

- Platt, J. C. 1999. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. *Advances in Kernel Methods: Support Vector Machines*, MIT press, Cambridge, MA, pp. 185–208.
- Pontil, M. and Verri, A. 1998. Properties of Support Vector Machines. *Neural Computation*, vol. 10, pp. 955–974.
- Potharst, R., Kaymak, U., and Pijls, W. 2001. Neural Networks for Target Selection in Direct Marketing. *Provided by Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam in its series Discussion Paper with number 77*, <http://ideas.repec.org/s/dgr/eureri.html>.
- Provost, F. and Fawcett, T. 1997. Analysis and visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. *Proc. of 3rd International Conference on Knowledge Discovery and Data Mining*, AAAI press, pp. 43–48.
- SAS Institute Inc. 1998. *Enterprise Mining Premier*.
- Schohn, G. and Cohn, D., 2000. Less is More: Active Learning with Support Vector Machines. *Proc. of the 17th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 839–846.
- Sen, A. and Srivastava, M. 1990. *Regression Analysis: Theory, Method, and Applications*, (Springer Texts in Statistics), Springer-Verlag New York Inc.
- Schölkopf, B., Burges, D., and Vapnik, V. 1995. Extracting Support Data for a Given Task. *Proc. of 1st International Conference on Knowledge Discovery and Data Mining*, AAAI press, pp. 252–257.
- Schölkopf, B., Burges, C. J. C, and Smola, A. J. 1999. *Advances in Kernel Methods: Support Vector Learning*, MIT press, Cambridge, MA.
- Schölkopf, B., and Smola, A., J. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT press.
- Shin, H. J. and Cho, S. 2002. Pattern Selection Using the Bias and Variance of Ensemble. *Journal of the Korean Institute of Industrial Engineers*, vol. 28, No. 1, pp. 112–127.
- Shin, H. J. and Cho, S. 2002. Pattern Selection For Support Vector Classifiers. *The 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Lecture Notes in Computer Science (LNCS 2412)*, Manchester, UK, pp. 469–474.
- Shin, H. J. and Cho, S. 2003a. Fast Pattern Selection for Support Vector Classifiers. *Proc. of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Artificial Intelligence (LNAI 2637)*, Seoul, Korea, pp.376–387.

- Shin, H. J. and Cho, S. 2003b. Fast Pattern Selection Algorithm for Support Vector Classifiers: Time Complexity Analysis. *The 4th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Lecture Notes in Computer Science (LNCS 2690)*, Hong Kong, China, pp. 1008–1015.
- Shin, H. J. and Cho, S. 2003c. How Many Neighbors To Consider in Pattern Pre-selection for Support Vector Classifiers?. *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Portland, U.S.A., pp. 565–570.
- Short, R. D and Fukunaga, K., 1981. The Optimal Distance Measure for Nearest Neighbor Classification. *IEEE Transactions on Information and Theory*, vol. IT-27, no. 5, pp. 622–627.
- Sohn, S. and Dagli, C.H., 2001. Advantages of Using Fuzzy Class Memberships in Self-Organizing Map and Support Vector Machines. *CD Proc. of International Joint Conference on Neural Networks (IJCNN)*, Washington D.C. July.
- Suh, E. H., Noh, K.C., and Suh, C. K. 1999. Customer List Segmentation Using the Combined Response Model. *Expert Systems with Applications*, vol. 17, no. 2, pp. 89–97.
- The Direct Marketing Association (DMEF Dataset). <http://www.thedma.org/dmef/dmefdataset.shtml>
- Tsaparas, P., 1999. Nearest Neighbor Search in Multidimensional Spaces. *Qualifying Depth Oral Report, Technical Report 319-02, Dept. of Computer Science, University of Toronto*.
- UCI Repostory. <http://www.ics.uci.edu/~mllearn/>
- Vapnik, V. 1999. *The Nature of Statistical Learning Theory*, Springer, 2nd eds.
- Viaene, S., Baesens, B., Van den Poel, D., Dedene, G. and Vanthienen, J. 2001a. Wrapped Input Selection using Multilayer Perceptrons for Repeat-Purchase Modeling in Direct Marketing. *International Journal of Intelligent Systems in Accounting, Finance & Management*, vol. 10, pp. 115–126.
- Viaene, S., Baesens, B., Van Gestel, T., Suykens, J. A. K., Van den Poel, D., Vanthienen, J., De Moor, B. and Dedene, G. 2001b. Knowledge Discovery in a Direct Marketing Case using Least Squares Support Vector Machines. *International Journal of Intelligent Systems*, vol. 16, pp. 1023–1036.
- Zahavi, J., and Levin, N. 1997a. Issues and Problems in Applying Neural Computing to Target Marketing. *Journal of Direct Marketing*, vol. 11, no. 4, pp. 63–75.

- Zahavi, J., and Levin, N. 1997b. Applying Neural Computing to Target Marketing. *Journal of Direct Marketing*, vol. 11, no. 4, pp. 76–93.
- Zheng, S.-F., Lu, X.-F, Zheng, N.-N., and Xu, W.-P., 2003. Unsupervised Clustering based Reduced Support Vector Machines. *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, pp. 821–824.

초 록

Support Vector Machine (SVM) 알고리즘은 구조적 위험 최소화 (structural risk minimization, SRM)의 원리를 채택함으로써 경험적 위험 최소화 (empirical risk minimization, ERM)를 따르던 기존의 여타 기계학습 알고리즘들에 비해 월등한 일반화 성능 향상을 보인다. 그러나 학습 패턴 (데이터) 셋이 클 경우, SVM 학습은 상당량의 메모리와 긴 계산시간을 필요로 한다. 이러한 계산상의 한계점을 극복하기 위한 직접적인 방법은 학습에 앞서 중요한 패턴들만을 선택함으로써 주어진 학습 패턴 셋을 줄이는 것이다. 다른 기계학습 알고리즘들과 비교해 볼 때 SVM은 어떠한 패턴들이 학습에 유효한 영향을 주는지가 이론적으로 명확하다. 이러한 패턴들은 Support Vector (SV) 라고 불리우며 대부분 분류경계에 위치한다. 즉, 다른 패턴들 없이 SV 셋만으로도 주어진 문제를 온전히 정의할 수 있으며 해를 구하기에 충분하다. SV들은 또한, SVM에서 사용자가 선택해야만 하는 파라미터인 커널의 종류 즉, RBF, polynomial, sigmoid 등에 무관하게 동일하다. 따라서, 이러한 학습 셋의 일부가 미리 정의된다는 것은 불필요한 계산상의 부담을 줄이는 의미 있는 일이라 할 수 있다.

본 논문에서는 이러한 패턴들을 사전에 선택하는 전처리 알고리즘을 제안한다. 제안하는 알고리즘은 인접한 이웃 패턴들의 구성상의 특성에 따라서 해당 패턴이 분류경계에 있는지의 여부를 판단하고 선택여부를 결정한다. (이후 제안하는 알고리즘을 NPPS - neighborhood property based pattern selection 라 칭한다.) 우선적으로 NPPS는, 각 패턴 주변의 가장 가까운 k개의 이웃 패턴들을 조사하여, 그 이웃 패턴들의 클래스 구성이 이질적이면, 즉 k개의 패턴들 중 일부는 특정 클래스에 속하고 나머지는 다른 클래스에 속할 경우, 해당 패턴을 분류경계 패턴으로 선택한다. 이들 일차적으로 선택된 분류경계 패턴들 중 노이즈는 제거되어야 하는데 이를 위하여 두 번째 아이디어가 도입된다. 즉, 해당 패턴의 클래스가 그 주변 패턴 대다수가 속한 클래스와 다를 경우, 해당 패턴은 노이즈로 간주되어 제거된다. 이러한 두 단계의 아이디어가 NPPS의 근간을 이루며 분류경계 패턴들을 성공적으로

로 선택한다. 그러나 이 두 가지 아이디어만으로 구현된 NPPS는 전처리 알고리즘으로서 다소 길다고 볼 수 있는 시간상의 복잡도 $O(M^2)$ 을 갖는다. 이를 개선시키기 위해서 또 하나의 아이디어가 도입된다. 즉, 분류경계 패턴의 이웃 패턴도 역시 분류경계 패턴이 될 가능성이 크다는 특성을 이용하게 되는데, 이는 모든 패턴이 아닌 분류경계에 위치한 패턴들만을 효율적으로 탐색하게 한다. 이 개선된 알고리즘의 복잡도는 주어진 학습 데이터 셋의 수와 얼마나 많은 패턴들이 분류경계 부근에 있느냐에 비례한다.

본 논문에서는 앞서 언급한 NPPS 알고리즘 및 복잡도 분석을 수록한다. 또한 알고리즘 내에서 얼마나 많은 수의 이웃패턴들을 고려해야 하는지, 또 입력 공간상에서 정의된 이웃 패턴들이 SVM 알고리즘을 수행하기 위하여 사영된 고차원의 특성 공간상에서도 여전히 이웃관계를 유지하는지의 증명을 보인다. 제안하는 알고리즘은 인공데이터 및 표준화된 벤치마킹 데이터 셋에 대하여 실험 검증되었으며 그 결과 정확도의 손실없이 많게는 100 배까지 학습 수행시간을 단축시킨다.

본 논문에서는 또한 SVM 및 NPPS의 실질적인 사용용도 및 방법을 예시하기 위하여, 이를 마케팅 분야의 고객관계경영 (customer relationship management, CRM) 문제에 적용한다. 특히 고객반응모델링 (customer response modeling)을 하고자 할 때 당면하는 현실적인 어려움들을 즉, 방대한 고객 데이터 셋을 어떻게 줄여야 하는지, 반응고객수 대비 무반응고객수의 불균형을 어떻게 해결하여야 하는지, 반응고객 중에서도 어느 고객이 얼마나 더 잘 반응하는지를 어떻게 알아낼 수 있는지 등에 대한 해결방안 및 척도를 제시한다.

주요어 : 데이터마이닝 (data mining), 기계학습 (machine learning), Support Vector Machines (SVM), 패턴선택 (pattern selection), 고객관계경영 (Customer Relationship Management), 고객반응모델링 (response modeling)

학 번 : 2000-30378

Acknowledgements

Before anything, I would like to give warm thanks to my advisors, prof. Sungzoon Cho. He has encouraged me to do this PhD and inspired most of what is contained in the thesis. My gratitude also goes to all of my lab-mates. During my PhD, I have very much enjoyed the atmosphere of working together and having fun together with them.

I feel indebted beyond measure to my parents and sisters for their immense cheers. Last, but certainly most important, I would like to mention about the love and devotion of my dearly beloved husband, Kyoungwon Min. Without his help all this would not have been possible in the first place, thus I could not have had today. I can never thank him enough. To all my family I dedicate this thesis.