ขั้นตอนวิธีการเพื่อนบ้านใกล้ที่สุดของคู่กลุ่มตรงข้ามที่ใช้หลักการแบ่งและเอาชนะ สำหรับแก้ปัญหาการแบ่งกลุ่มและการถดถอย

นาวาอากาศเอก ธนพันธุ์ หร่ายเจริญ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547

ISBN 974-17-6418-9

A DIVIDE-AND-CONQUER APPROACH TO THE PAIRWISE OPPOSITE CLASS-NEAREST NEIGHBOR (POC-NN) ALGORITHM FOR CLASSIFICATION AND REGRESSION PROBLEMS

Group Captain Thanapant Raicharoen

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Department of Mathematics
Faculty of Science
Chulalongkorn University
Academic Year 2003
ISBN 974-17-6418-9

Thesis Title	A DIVIDE-AND-CONQUER APPROACH TO THE PAIRWISE OPPOSITE CLASS-NEAREST NEIGHBOR (POC-NN) ALGORITHM FOR CLASSIFICATION AND REGRESSION PROBLEMS
By	Group Captain Thanapant Raicharoen
Field of Study	Computer Science
Thesis Advisor	Professor Chidchanok Lursinsap, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctor's Degree

[illegible] .... Dean of the Faculty of Science

(Professor Piamsak Menasveta, Ph.D.)

THESIS COMMITTEE

[illegible] .... Chairman

(Associate Professor Peraphon Sophatsathit, Ph.D.)

[illegible] .... Thesis Advisor

(Professor Chidchanok Lursinsap, Ph.D.)

[illegible] .... Member

(Assistant Professor Krung Sinapiromsaran, Ph.D.)

Kosin. Chamnongthai .... Member

(Associate Professor Kosin Chamnongthai, Ph.D.)

[illegible] .... Member

(Assistant Professor Boonserm Kijsirikul, Ph.D.)

[illegible] .... Member

(Assistant Professor Thitipong Tanprasert, Ph.D.)

น.อ.ธนพันธุ์ หร่ายเจริญ : ขั้นตอนวิธีการเพื่อนบ้านใกล้ที่สุดของคู่กลุ่มตรงข้ามที่ใช้หลักการแบ่งและเอาชนะ สำหรับแก้ปัญหาการแบ่งกลุ่มและการถดถอย. (A DIVIDE-AND-CONQUER APPROACH TO THE PAIRWISE OPPOSITE CLASS-NEAREST NEIGHBOR (POC-NN) ALGORITHM FOR CLASSIFICATION AND REGRESSION PROBLEMS) อ. ที่ปรึกษา : ศาสตราจารย์ ดร. ชิดชนก เหลือสินทรัพย์, 67 หน้า. ISBN 974-17-6418-9.

วิทยานิพนธ์ฉบับนี้เสนอขั้นตอนวิธีการใหม่ที่อยู่บนพื้นฐานของหลักการแบ่งและเอาชนะ เพื่อเลือกเซตของต้นแบบที่มาจากข้อมูลที่ใช้ในการเรียนรู้ด้วยการใช้กฎเพื่อนบ้านใกล้ที่สุด วิธีการนี้มีจุดมุ่งหมายที่จะลดเวลาที่ใช้ในการคำนวณและเนื้อที่ที่ใช้ในการจัดเก็บรวมถึงความไวต่อลำดับของข้อมูลและสัญญาณรบกวนในข้อมูลที่ใช้สอน เซตของข้อมูลต้นแบบที่ถูกลดขนาดลงประกอบด้วยข้อมูลคู่ต้นแบบในกลุ่มตรงข้ามที่ใกล้กันซึ่งกลุ่มข้อมูลนี้จะอยู่ติดกับเส้นแบ่งเขต และเซตของต้นแบบนี้จะใช้ในการทดสอบแทนข้อมูลที่ใช้ในการเรียนรู้ทั้งหมด การที่ได้มาซึ่งข้อมูลคู่ต้นแบบของกลุ่มตรงข้ามที่ใกล้กันนั้นจะได้มาจากการแบ่งข้อมูลที่ใช้ในการเรียนรู้เป็นสองส่วนแบบทีละขั้นจนกระทั่งข้อมูลที่อยู่ในแต่ละส่วนได้ถูกแบ่งกลุ่มอย่างถูกต้อง การแบ่งข้อมูลนี้จะถูกกำหนดโดยเพื่อนบ้านใกล้ที่สุดของคู่กลุ่มตรงข้ามที่เป็นตัวกำหนดเส้นแบ่งกลุ่ม วิธีการนี้สามารถทำได้อย่างรวดเร็วและผลที่ได้จะไม่ขึ้นกับลำดับของข้อมูลที่ใช้ในการเรียนรู้ นอกจากนั้นจำนวนต้นแบบและความซับซ้อนของแบบจำลองสามารถลดลงได้โดยผู้ใช้เป็นผู้กำหนด วิธีการนี้นอกจากจะสามารถใช้ในการแก้ปัญหาการแบ่งกลุ่มแล้ว ยังสามารถใช้ในการแก้ปัญหาการถดถอยได้อีกด้วย ผลจากการทดลองได้แสดงให้เห็นอย่างมีนัยสำคัญว่า วิธีการนี้ให้ผลลัพธ์ทั้งในแง่ความถูกต้องในการทดสอบข้อมูลใหม่และในแง่ของจำนวนข้อมูลต้นแบบที่ลดน้อยลง รวมทั้งเวลาที่ใช้ในการเรียนรู้ดีกว่าวิธีการอื่นที่อาศัยกฎเพื่อนบ้านใกล้ที่สุดในอดีต

ภาควิชา	**คณิตศาสตร์**	ลายมือชื่อนิสิต......น.อ. [illegible]......
สาขาวิชา	**วิทยาการคอมพิวเตอร์**	ลายมือชื่ออาจารย์ที่ปรึกษา......[illegible]......
ปีการศึกษา	2547	

Abstract (English)

4473856423 : MAJOR COMPUTER SCIENCE
KEY WORD: CLASSIFICATION / REGRESSION / NEAREST NEIGHBOR RULE / PROTOTYPE SELECTION / MACHINE LEARNING.

GP.CAPT.THANAPANT RAICHAROEN: A DIVIDE-AND-CONQUER APPROACH TO THE PAIRWISE OPPOSITE CLASS-NEAREST NEIGHBOR (POC-NN) ALGORITHM FOR CLASSIFICATION AND REGRESSION PROBLEMS, THESIS ADVISOR: PROFESSOR CHIDCHANOK LURSINSAP, Ph.D., 67 pp. ISBN 974-17-6418-9.

This paper presents a new method based on divide-and-conquer approach to the selection of a set of prototypes from the training data by the nearest neighbor rule. The method aims at reducing computational time and memory space as well as sensitivity of the order and noise of the training data. A reduced prototype set contains Pairwise Opposite Class-Nearest Neighbor (POC-NN) prototypes, which are close to the decision boundary and used instead of the training patterns. POC-NN prototypes are obtained by recursively analysis and iterative separation of the training data into two regions until each region is correctly grouped and classified. The separability is determined by the POC-NN prototypes essential to define the locations of all separating hyperplanes. Our method is fast and order independent. The number of prototypes and the overfitting of the model can be reduced by the user. This method can be used to solve not only classification but also regression problems. The experimental results signify the effectiveness of this technique and its performance in both accuracy and prototype rate as well as in training time over those obtained by classical nearest neighbor techniques.

Acknowledgements

I am especially deeply grateful to my supervisor Professor Dr. Chidchanok Lursinsap who has provided me with a great deal of guidance in the area of machine learning and neural network. His valuable suggestions and comments made this work feasible. Moreover, he also gave me an opportunity to pursue a Ph.D. by employing me as his research assistant under the Thailand Research Fund.

I would like to thank Professor Dr. Frank Lin for discussing and applying our algorithm to the real world problem of the emergency flood forecasting methodologies on the Chaophraya River. I am grateful to Associate Professor Suchada Siripant for providing me with an excellent work environment in the Advanced Virtual and Intelligent Computing Center (AVIC) through all my Ph.D. student life.

I would like to thank Air Vice Marshal Payoongsak Ruayruen, Director of the Royal Thai Air Force Office of Information Technology, who always supports me throughout my work and study.

I would like to thank the Thailand Research Fund (the Royal Golden Jubilee Ph.D. Program) for funding source. This fund gave me a chance to pursue Ph.D. program and to have research collaboration in oversea.

I would like to thank my committee members for their corrections of my dissertation as well as providing valuable comments at the committee meetings.

I would like to thank all members of AVIC's group for their helps and discussions on my research. My special thanks are given to Miss Kingkarn Sookhanaphibarn for her help in the qualifying examination.

My special thanks go to my beloved parents for their everlasting supports and encouragements throughout my study. The success of my study has been paved by their love and vision. I especially thank my wife, Wing Commander Ubonrat Raicharoen, for her patience. My success would not be possible without her encouragement.

Table of Contents

List of Tables

List of Figures

Figure Page

Nomenclatures

d	=	The number of dimension
k	=	The number of Nearest Neighbors
m	=	The number of classes
n	=	The number of training patterns (samples)
p	=	The number of POC-NN prototypes
K	=	The number of Fold Cross Validations
S	=	The Set of training patterns (samples)
$S^{(i)}$	=	The Set of training patterns of class i
α	=	An acceptance interval
α_r	=	An acceptance interval-ratio
1-v-1	=	One-against-One
1-v-r	=	One-against-Rest
$\mathrm{Dist}(\cdot,\cdot)$	=	Distance metric
L_2-norm	=	Euclidean distance
$P(C_i)$	=	A priori probability of the classe i
P_e	=	The probability of error
σ	=	Standard deviation

Abbreviations

AR %	=	Accuracy Rate in Percent
CNN	=	Condensed Nearest Neighbor
GKA	=	Chidananda Gowda and Krishna Algorithm
k-NN	=	k-Nearest Neighbor
k-NCN	=	k-Nearest Centroid Neighbor
MNV	=	Mutual Neighborhood Value
MG	=	Mackey Glass
NN	=	Nearest Neighbor
N/A	=	Not Applicable/Available
OCR	=	Optical Character Recognition
POC-NN	=	Pairwise Opposite Class-Nearest Neighbor
POC-NN-SET	=	Set of Pairwise Opposite Class-Nearest Neighbor
PR %	=	Prototype Rate in Percent
PRD	=	Percent Root Mean Square Difference
Tomek	=	Tomek's Two Modification of the *CNN* Algorithm
RNN	=	Reduced Nearest Neighbor
SNN	=	Selective Nearest Neighbor

CHAPTER I

INTRODUCTION

The *Nearest Neighbor (NN)* rule is one of the most attractive *non-parametric decision rules* and *model-free methods/instance-based learning rules* for data classification and pattern recognition since no a priori knowledge is required concerning the underlying distributions of the data. Because the non-parametric decision rules are highly unstructured, they typically are not useful for understanding the nature of the relationship between the features and class outcome. However, as a *black-box* predictor, they can be very effective, and are often among the best performers in real problems. The *NN* technique can also be used in regression and works reasonably well for low-dimensional problems. However, with high-dimensional features, the bias-variance tradeoff does not work as favorably for nearest neighbor regression as it does for classification. Moreover, the original *NN* rule in general requires the computational load, both in time (finding the neighbor) and space (storing the entire training data set). Therefore, reducing storage requirements is important and still an ongoing research issue.

The remaining of this chapter will discuss problem and motivation, objective, scope and limitations, and contribution of the dissertation.

1.1 Problem and Motivation

The *Nearest Neighbor (NN)* rule was originally proposed by Cover and Hart [1, 2] in 1966 and has been shown to be very effective in many applications of pattern recognition. One reason for the use of this rule is its conceptual simplicity which is easy to implement. Moreover, under some continuity assumptions on the underlying distributions, the asymptotic error rate of this rule is at most twice Bayes' probability of errors [1, 3]. However, the *NN* rule suffers from various drawbacks. Firstly, it requires large memory space as the entire training data set has to be stored while each test pattern is being compared with every training pattern. Secondly, it requires large computational time to find the neighbors. Lastly, it is sensitive to noisy and/or outlier patterns.

To alleviate these drawbacks, two approaches have been introduced, namely, *prototype selection* and *prototype replacement* [4]. Both approaches aim at modifying an original training pattern (prototype) in order to reduce its size as well as to improve classification performance. One of the first and most popularly used techniques of prototype selection is the *Condensed Nearest Neighbor (CNN)* proposed by Hart [5]. The main goal of the condensing method is to obtain the reduced and *consistent set* of prototypes [5] to be used with the *1-NN* rule without error in the training set or with the *k-NN* rule without significantly degrading its performance. The condensing method proceeds by repeatedly selecting the prototypes whenever they cannot be correctly classified by the currently selected set. The whole process is iterated until there is no change in a complete pass through the initial training set. However, the method does not, in general, yield a minimal size of the consistent subset, and the final size as well as composition of the final condensed set may depend upon the order of data presentation. Since the development of the *CNN*, other methods were proposed successively, such as the *Reduced Nearest Neighbor (RNN)* rule proposed by Gates [6], an algorithm for a *Selective Nearest Neighbor (SNN)* decision rule was introduced by Ritter et al. [7], and Tomek presented the *Two Modifications of the CNN* by growing the condensed set using only patterns close to the decision boundary [8]. The way in which pairs of prototypes are selected makes the algorithm appropriate at preserving the original decision boundaries.

Chidananda Gowda and Krishna introduced the concept of *mutual* nearest neighbor neighborhood for selecting patterns close to the decision boundaries [9]. The position of a prototype in the ordered list of neighbors of its nearest neighbor from an opposite class is used as a way to measure the closeness to boundaries. Several theoretical results on *CNN* have been obtained in [10].

Another well known technique of prototype selection is the *editing* method [11] proposed by Wilson [12]. The main goal of the editing method is to improve the performance by discarding outliers and possible overlapping among classes rather than prototype reduction. However, the drawbacks of the editing method are that it still leaves too many prototypes in the edited set, and the complexity of computing the edited subset is very high. Therefore, Sanchez proposed the *k-Nearest Centroid Neighbors (k-NCN)* in order to identify and eliminate mislabeled, noisy and atypical training patterns [13]. Several editing experiments are carried out and comparative results are presented in [14]. The exploration and exploitation of the synergy among the *NN editing* and *condensing* methods in order to facilitate the use of *NN* techniques in real-world applications was studied in [15].

For the prototype replacement approach, one of the first methods, proposed by Chang [16], repeatedly merges the two nearest neighbors of the same class as long as this merger does not increase the error rate on the training set. One drawback of this method is that it may yield the prototypes that do not characterize the training set well in terms of generalization [17].

In order to obstruct the undesirable property of the order dependence of presentation data, several attempts, for example in [18, 19], were suggested to obtain selected prototypes that are less sensitive to this property. However, most improvement of *NN* methods cannot avoid overfitting for noisy and/or overlapping data, and do not consider any statistical properties of the training data.

This dissertation focuses on developing a new method for obtaining a set of selection prototypes from the training set for the nearest neighbor rule. Unlike all of the above methods, the proposed method is based on divide-and-conquer approach. That is the analogy to partition original training patterns into smaller regions by finding *POC-NN* prototypes for the regions, and then combine the *POC-NN* prototypes for the regions into a set of selection prototypes.

1.2 Objective

The objective is to develop a prototype selection method for the nearest neighbor rule, which enhances the performance in terms of accuracy rate and running time. This method can be used to solve classification and regression problems.

1.3 Scope and Limitations

In this dissertation, the scope of work is constrained as follows:

1. For classification problem, the experimental results will be compared with the results of the classical *NN* rule and some prototype selection methods, i.e. the *CNN*, the *Two Modifications of the CNN*, and the *Mutual NN* method. All experiments are based on data sets of the UCI Machine Learning Repository [20], the USPS data [21], and the Statlog [22].

2. For regression problem, the experimental results will be compared with the results of the nearest neighbor and the linear interpolation. All experiments are based on data sets of the sinc function, Mackey-Glass [23], Lorenz [24], Titanium [25], and Sunspot [26] data.

1.4 Contribution

This dissertation is proposed a new method based on divide-and-conquer approach to the selection of a set of prototypes from the training set for the nearest neighbor rule, which is order independent and fast. The number of prototypes and the overfitting of the model can be reduced by the user.

1.5 Research Methodology

1. Review and study the research papers that are related to the machine learning algorithms, especially the nearest neighbor rule.
2. Develop a new method for prototype selection based on the nearest neighbor rule.
3. Prepare data sets and set up the experiment.
4. Test against benchmark data set.
5. Compare with the other algorithms.
6. Analyze the results and summarize the outcome of study.

1.6 Organization of the dissertation

The dissertation is organized into seven chapters. Chapter II reviews the theoretical background and the main idea of the *NN* rule including the prototype selection methods related to the proposed method. Chapter III presents the proposed *POC-NN* method for classification problem, including its analysis. Chapter IV describes the experiment and evaluate the performance accordingly. Chapter V and Chapter VI present the *POC-NN* method for regression problem and the experimental results, respectively. Chapter VII concludes the research work and presents some directions for future work.

CHAPTER II

BACKGROUND AND LITERATURES REVIEWS

In this chapter, the theoretical background on the nearest neighbor rule for classification and regression problem is described. Literatures related to prototype selection methods are also reviewed. Three previous work on the prototype selection methods which are related to the proposed methods, i.e., the CNN, the Two Modifications of the CNN, and the mutual NN neighbor will be discussed.

2.1 Background

In this section, the theoretical background of the Nearest Neighbor Rules is reviewed.

2.1.1 The Nearest Neighbor Decision Rules

Decision Rules are employed in many areas such as pattern recognition and computer vision. They are used to determine class membership of an object based on some numerical measurements for that object. *Parametric decision rules* determine the membership classification based on a priori probabilities of occurrence of objects belonging to some class C_i. In many situations, however, these distributions are unknown or are difficult to describe and handle analytically. *Non-parametric decision rules*, such as the *Nearest Neighbor (NN) rule*, are attractive since no a priori knowledge of the distributions is required. These rules rely on the training set of objects with known class membership to make decisions on the membership of unknown objects.

Assuming there are m pattern classes, numbered 1,2,...,m. Let each pattern $\mathbf{x}_i$ be defined in a d-dimensional feature space and let there be n training patterns. Each training pattern is a pair ($\mathbf{x}_i$, y_i), $1 \leq i \leq n$, where $y_i \in \{1, 2, \ldots, m\}$ denotes the correct pattern class. Let $S = \{(\mathbf{x}_1, y_2), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ be the training set. Given an unknown pattern $\mathbf{u}$, $\mathbf{u}$ is in class y_j if

$$Dist(\mathbf{u}, \mathbf{x}_j) \leq Dist(\mathbf{u}, \mathbf{x}_i), 1 \leq i \leq n, \tag{2.1}$$

where Dist(·,·) is some d-dimensional distance metric (Euclidean distance metric is commonly used).

As a matter of fact, the preceding rule is more properly called the *1-NN* rule since it uses only one nearest neighbor. An obvious generalization of this is the k-NN rule, which takes the k nearest patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ and decides upon the pattern class that appears most frequently in the set $\{y_1, y_2, \ldots, y_k\}$.

A key feature of this decision rule is that it performs remarkably well despite is no explicit knowledge of the underlying distributions of the data used. Consider, for example, a two class problem and denote the *a priori* probabilities of the two classes by $P(C_1)$ and $P(C_2)$, a *posteriori* probabilities by $P(C_1|\mathbf{x})$ and $P(C_2|\mathbf{x})$, and the joint probability density function by

$$p(\mathbf{x}) = P(C_1)p(\mathbf{x}|C_1) + P(C_2)p(\mathbf{x}|C_2) \tag{2.2}$$

where $p(\mathbf{x}|C_i)$ is the class-condition probability density function given class $C_i, i = 1, 2$. In 1967 Cover and Hart [2] showed, under some continuity assumptions on the underlying distributions, that the asymptotic error rate of the *1-NN* rule, denoted by $P_e[1 - NN]$ was given by

$$P_e[1 - NN] = 2\mathbf{E}_x[P(C_1|\mathbf{x})P(C_2|\mathbf{x})] \tag{2.3}$$

where $\mathbf{E}_x$ denoted the expected value with respect to the joint probability density function *p(X)*. They also showed that $P_e[1 - NN]$ was bounded from above by twice the

Bayes' errors (the error of the best possible error). More precisely, and for the more general case of m pattern classes, the bounds proved by Cover and Hart [1, 3] are given by:

$$P_e \leq P_e[1 - NN] \leq P_e(2 - \frac{m}{m-1}P_e) \tag{2.4}$$

where P_e is the optimal Bayes probability of error.

2.1.2 The k-Nearest Neighbor for Classification

To demonstrate a k-Nearest Neighbor analysis, let's consider the task of classifying a new unknown pattern $\mathbf{x}_u$ (query point) among a number of known class label examples. This is shown in Figure 2.1 which depicts the patterns (instances) with the symbol plus "+" (Class $C_1 = 1$), the symbol star "$*$" (Class $C_2 = 2$), and the query point $\mathbf{x}_u$ inside a black circle. The task is to predict (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, it need to know whether the query point can be classified as a "+" or a "$*$" symbol.

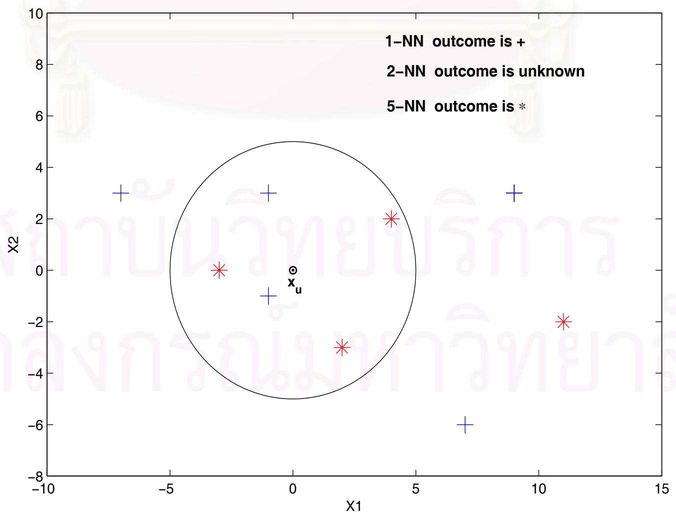


Figure 2.1: Demonstration of the *k-NN* Classification

To proceed, consider the outcome of *k-NN* based on 1-Nearest Neighbor. It is clear that in this case *k-NN* will predict the outcome of the query point with a plus (since the closest point carries a "+" sign). Now let increase the number of nearest neighbors to two, i.e., *2-NN*. This time *k-NN* will not be able to classify the outcome of the query point since the second closest point is a star, and so both the "+" and the "$*$" signs achieve the same score (i.e., win the same number of votes). The number of k should be odd in order to avoid a tie vote. If the number of nearest neighbors increases to five (*5-NN*), a nearest neighbor region, which is indicated by the circle shown in the Figure 2.1, is obtained. Since there are 2 "+" and 3 "$*$" signs in this circle, *k-NN* will assign a "$*$" sign to the outcome of the query point.

2.1.3 The k-Nearest Neighbor for Regression

In this section, the concept of k-Nearest Neighbors is generalized to include regression problems. Regression is the process of estimating a real-valued function based on a finite set of samples. The output of the system in regression problems is a random variable that takes on real value. The k-Nearest Neighbors technique can be viewed as a form of local risk minimization. In this method, the function estimates derive from taking a local average of the data. Locally is defined as the k data points that are nearest to the estimation point.

Consider the schematic shown in Figure 2.2, where a set of points ($\times$ Symbol) are drawn from the relationship between the independent variable x and the dependent variable y (sine curve). From the given set of training patterns (known as instances), the k-Nearest Neighbors method is used to predict the outcome of x (also known as unknown testing point or query point).

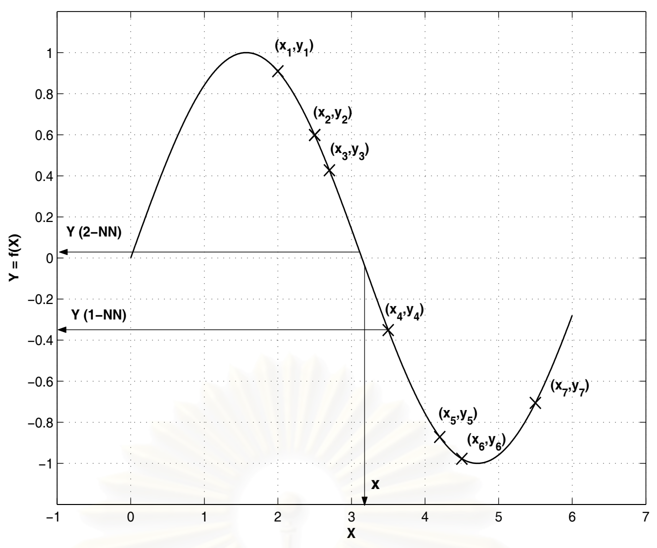


Figure 2.2: Demonstration of the *k-NN* for Regression

For *1-NN* method, the training set ($\times$ symbols) is search and located the one closest to the query point x. For this particular case, this happens to be x_4. The outcome of x_4 (i.e., y_4) is thus taken to be the answer to the query point x (i.e., y=y_4).

Next, let consider the *2-NN* method. In this case, we locate the first two closest points to x, which happen to be y_3 and y_4. Taking the average of their outcome, the solution for y can be obtained by y=$\frac{y_3+y_4}{2}$.

The above demonstration can be extended to any arbitrary number of nearest neighbors k. To summarize, in a k-NN method, the outcome y of the query point x is taken to be the average of the outcomes of its k-Nearest Neighbors.

2.2 Literature Review

This section reviews some popularly used prototype selection techniques for the condensed set using only patterns close to the decision boundary, which will be compared with the proposed technique.

2.2.1 The Condensed Nearest Neighbor (CNN) Rule

In 1968, Hart was the first to propose an algorithm for reducing the size of the stored data for the nearest neighbor decision rule called the *Condensed Nearest Neighbor (CNN) Rule* [5]. As a matter of fact, this is not a new decision rule since it still chooses the class of the nearest neighbor. Rather, the word condensed refers to a procedure for choosing a subset of the sample set or the training set S (set of samples or patterns with known class label). Its purpose is to reduce the size of the original data set S by elimination of certain samples without significantly affecting the performance of *NN* classification. Hart defined a *consistent* subset of the data as one that classified the remaining data correctly with the 1-Nearest Neighbor rule. Let S_{CNN} be a *consistent* subset of S, and initially be empty. The algorithm for constructing S_{CNN} proceeds as follows:

```
Function CNN (S: Dataset)

1. A sample (mostly used the first) pattern is copied from $S$ to $S_{CNN}$.
2. $S_{CNN}$ is used as the training set to classify each pattern of $S$, starting with the first
   sample pattern. This is repeated until one of the following two cases arises:
   2.1 every pattern in $S$ is correctly classified, in which case, the process terminates;
       Return $S_{CNN}$.
   2.2 one of the patterns in $S_{CNN}$ is classified incorrectly, in which case, go to 3.
3. Add the pattern from $S$ that was incorrectly classified to $S_{CNN}$.
4. Go to 2.
```

There are one things that must be noted about this algorithm. The number of nearest neighbors to be considered k is not mentioned. Hart [5] mentioned $k \neq 1$ as being a possibility for future research. However, this suggestion contains a fundamental flaw. The number of k should be the odd number in order to guarantee a unique solution.

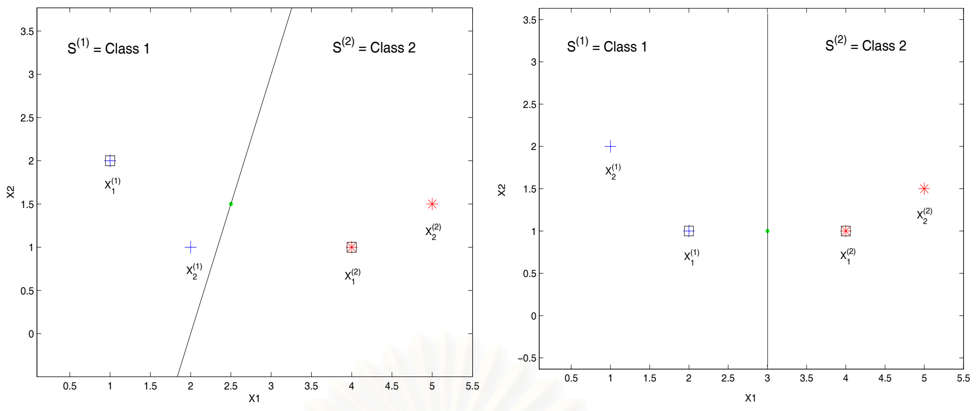


Figure 2.3: This example shows *CNN* is dependent on the order of presentation of data. $\mathbf{x}_1^{(1)}$ and $\mathbf{x}_2^{(1)}$ is the first and second pattern of class 1 ($S^{(1)}$), respectively.: (a) $\mathbf{x}_1^{(1)} = (1,2)'$; (b) $\mathbf{x}_1^{(1)} = (2,1)'$

It is clear that *CNN* has the following properties. First, S_{CNN} is a subset of the original set S, so that S_{CNN} is much smaller than S and thus computationally much better suited for *NN* classification. It requires less storage and computations. Second, S_{CNN} is a *consistent* subset of S which classifies (*1-NN* rule) all samples in S correctly, so that *NN* classification with S_{CNN} is very similar (although not necessarily identical) to *NN* classification with S. This is especially true when S is a good "representative" (which means that the number of samples and their distribution approximate the "true" underlying probability distribution by "good" relative frequency).

The disadvantages of the *CNN* are that, firstly, it is dependent on the order of presentation of S since it processes samples from S randomly by choosing the first sample pattern from S and copying to S_{CNN}. Secondly, S_{CNN} contains patterns which define a boundary on S_{CNN} but not on S (i.e., patterns not essential in S become boundary points in S_{CNN}). The examples of *CNN* when reordering training patterns show in Figure 2.3. The condensed prototypes are enclosed in square symbols "□". The ideal method of reduction of S would work essentially as *CNN* but would only use points close to the decision boundary to generate S_{CNN}.

2.2.2 The Tomek's Modification of the CNN

Since *CNN* may keep too many points that are not near the decision boundary, Tomek proposed the Two Modification of *CNN* [8] in which a preliminary pass of S is made to select an order-independent special subset of S that lies close to the decision boundary. After this pre-processing step his method proceeds in the same manner as *CNN* instead of moving to S_{CNN} data samples from the complete S. Only data points from the selected subset of S, called C, are used. Let S be a training set of n patterns composing of two subsets $S^{(1)}$ and $S^{(2)}$ whose sizes are $|S^{(1)}| = n^{(1)}$ and $|S^{(2)}| = n^{(2)}$. Both $S^{(1)}$ and $S^{(2)}$ are in different classes, namely, class 1 and 2, respectively, and $S^{(1)} \cap S^{(2)} = \varnothing$. It is assumed $\mathbf{x}_i^{(1)}$, $i = 1, \ldots, n^{(1)}$ and $\mathbf{x}_j^{(2)}$, $j = 1, \ldots, n^{(2)}$ are the patterns in S from class 1 and 2. The algorithm to pre-select the special subset of C proceeds as follows:

```
Function Tomek's Modification of CNN (S: Dataset)
1. Initialize: $C = \varnothing$.
2. For $1 \le i \le n^{(1)}$ Do
3.         For $1 \le j \le n^{(2)}$ Do
4.                 $\mathbf{z} = \frac{1}{2}(\mathbf{x}_i^{(1)} + \mathbf{x}_j^{(2)})$.
5.                 For $1 \le k \le n^{(1)}$ Do
6.                         If $\mathrm{Dist}(\mathbf{x}_k^{(1)}, \mathbf{z}) \le \mathrm{Dist}(\mathbf{x}_i^{(1)}, \mathbf{z})$ Then Goto 13.
7.                 End
8.                 For $1 \le l \le n^{(2)}$ Do
10.                        If $\mathrm{Dist}(\mathbf{x}_l^{(2)}, \mathbf{z}) \le \mathrm{Dist}(\mathbf{x}_i^{(1)}, \mathbf{z})$ Then Goto 13.
11.                End
12.                $C = C \cup \{\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(2)}\}$.
13.        End
14. End
15. Return $C$.
```

The algorithm to pre-select the C consists of keeping all pairs of points belong to different classes and the *diametral* sphere determined by $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_j^{(2)}$ does not contain any points of S in its interior. Such a pair are often called *Tomek links* [8]. Clearly, pair of points far from the decision boundary will tend to have other points in the interior of their diameter sphere. It is claimed [8] that the resulting C is a *consistent* set. However, Toussaint demonstrated a counter-example in [17]. Therefore, in the case of Tomek's algorithm the consistency is not guaranteed.

2.2.3 The CNN Rule using the concept of Mutual Nearest Neighborhood

Chidananda Gowda and Krishna have introduced the concept of mutual nearest neighborhood and a new similarity measure called the *Mutual Neighborhood Value (MNV)*. The *MNV* value between any two patterns of a set is the sum of conventional nearest neighbor ranks of these two patterns with respect to each other.

Let S be a training set of n patterns, $\mathbf{x}_i$, $1 \leq i \leq n$ be an element of S. Suppose $\mathbf{x}_j$ is the p^{th} nearest neighbor of $\mathbf{x}_i$, and $\mathbf{x}_i$ is the q^{th} nearest neighbor of $\mathbf{x}_j$. Then, the *MNV* between $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as $p+q$. That is, $MNV(\mathbf{x}_i, \mathbf{x}_j)=p+q$, where $p,q \in \{1, 2, \ldots n-1\}$, and 0 when $i=j$. The Gowda and Krishna's algorithm (GKA) for selecting a subset of patterns in a modified *CNN* is as follows:

```
Function Stage 1 of GKA(S: Dataset)
1. For $1 \leq i \leq n$ Do
2.     Find the nearest neighbor $\mathbf{y}_i$ from the opposite class of $\mathbf{x}_i$.
3.     Calculate $MNV(\mathbf{x}_i)$ by a number of patterns from the same class as $\mathbf{x}_i$ that
       lie closer to $\mathbf{y}_i$ than $\mathbf{x}_i$.
4. End
```

```
5. Order the $n$ patterns according to $MNV$ in ascending order.
   If the $MNV$'s of some of the patterns are identical
   Then order such patterns according to distances in ascending order.
6. The $CNN$ algorithm is applied to the ordered set $S_{ORD}$
   Return $S_{CNN}$ = CNN($S_{ORD}$).
```

When deletion of a pattern in the condensed subset produces no change in the classification of any member of the complete training set, the deleted pattern may be excluded from the condensed set [6]. This idea is used to make a further reduction in the number of patterns constituting the modified condensed set. The final contents of S_{RNN} constitute the second modified condensed set.

Procedure *Stage 2 of GKA(S_{CNN}: Dataset)*

```
1. $S_{RNN} = S_{CNN}$
2. Remove the first pattern from $S_{RNN}$.
3. Use $S_{RNN}$ to classify all the patterns in $S_{CNN}$:
   3.1 If all patterns are classified correctly
       Then Goto 4.
   3.2 If a pattern is classified incorrectly
       Then return the pattern that was removed and Goto 4.
4. If every patterns in $S_{RNN}$ has been removed once (and possibly replaced)
   Then Stop.
   Else Remove the next pattern and Goto 3.
```

CHAPTER III

THE METHODOLOGY OF POC-NN FOR CLASSIFICATION PROBLEM

This chapter gives details about the proposed method, the *Pairwise Opposite Class-Nearest Neighbor (POC-NN)* for prototype selection in order to solve classification problem. The idea is to isolate a subset of the training set that suffices for nearest neighbor predictions, and throw away the remaining data. Intuitively, it seems important to keep the training points (patterns) that are close to the decision boundaries and on the correct side of those boundaries, while some points far from the boundaries should be discarded. The bottom line of this idea is to find a *POC-NN* pattern for a given two-class training data set and then the remaining patterns can be discarded.

3.1 An Algorithm for Finding POC-NN Pattern

Let S be a training set of n patterns composed of two subsets $S^{(1)}$ and $S^{(2)}$ whose sizes are $|S^{(1)}| = n^{(1)}$ and $|S^{(2)}| = n^{(2)}$. Both $S^{(1)}$ and $S^{(2)}$ are in different classes, namely class 1 and class 2, respectively, and $S^{(1)} \cap S^{(2)} = \varnothing$. The algorithm to find a *POC-NN* pattern is given as follows:

Function *FINDING-POC-NN (S: Dataset)*

```
1. Let $S^{(1)}$ and $S^{(2)}$ be two training sets belong to classes 1 and 2,
   whose sizes are $n^{(1)}$ and $n^{(2)}$, respectively.
2. **If** $n^{(1)} >= n^{(2)}$
   **Then**
3.         $\mathbf{x}_m$ = mean of $S^{(1)}$.
3.1        Let $\mathbf{x}_{p2} \in S^{(2)}$ be the nearest pattern to $\mathbf{x}_m$.
3.2        Let $\mathbf{x}_{p1} \in S^{(1)}$ be the nearest pattern to $\mathbf{x}_{p2}$.
   **Else**
4.         $\mathbf{x}_m$ = mean of $S^{(2)}$.
4.1        Let $\mathbf{x}_{p1} \in S^{(1)}$ be the nearest pattern to $\mathbf{x}_m$.
4.2        Let $\mathbf{x}_{p2} \in S^{(2)}$ be the nearest pattern to $\mathbf{x}_{p1}$.
   **Endif**
5. **Return** ($\mathbf{x}_{p1}$,$\mathbf{x}_{p2}$) as a *POC-NN* pattern.
```

Figure 3.1(a) shows an example of how finding *POC-NN* algorithm works. There are two classes, 1 and 2. Each pattern in class 1 is denoted by the symbol "+" and each pattern in class 2 is denoted by the symbol "$*$". The mean of patterns ($\mathbf{x}_m$) in class 1 is denoted by the symbol "$\diamond$". The *POC-NN* prototypes ($\mathbf{x}_{p1}$,$\mathbf{x}_{p2}$) are enclosed in circle symbols "$\bigcirc$". This *POC-NN* prototype is performed during the training process and all considered patterns in all classes are considered as the training patterns. The identical *POC-NN* patterns as prototypes are always obtained independently from different reordering of the training patterns. Moreover, the *POC-NN* patterns have a desirable property similar to the *Support Vectors* [27, 28] which induce the optimal separating hyperplane. This is because if all the training vectors are linearly independent, the two closest patterns of opposite classes are support vectors. This proof can be found

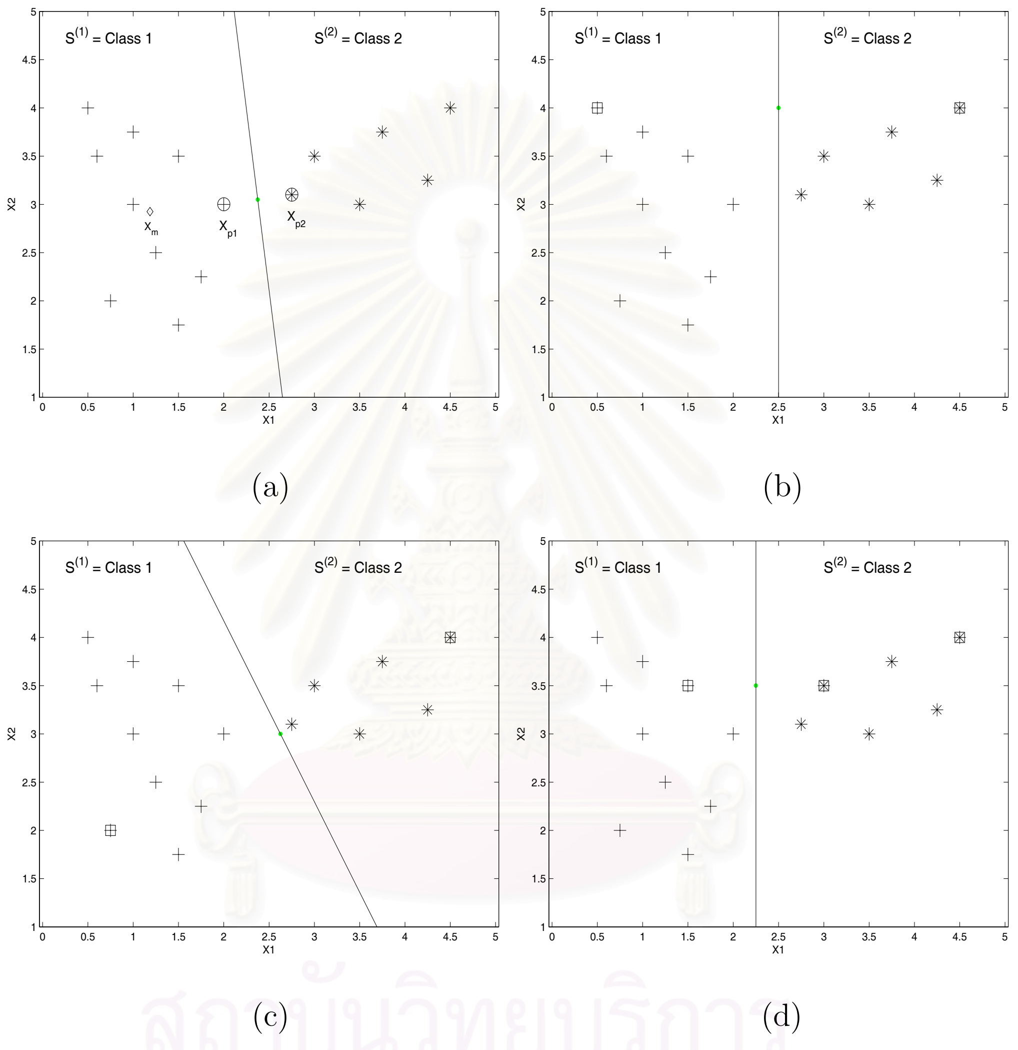


Figure 3.1: This example shows how to select the prototypes: (a) *POC-NN*; (b)-(d) CNN when reordering training patterns.

in [29]. However, the *POC-NN* algorithm is not guaranteed to find these support vectors. Figure 3.1(b) and (c) show the prototypes created by the *CNN* method [5] which does not guarantee the optimal solution. The condensed prototypes are enclosed in square symbols "□". The obtained prototypes strongly depend on the order of presentation of the training patterns. In this case, it depends on the first pattern of each class. Figure 3.1(d) also shows the prototypes by using the *CNN* method which creates the problems of redundant prototypes and also slow convergence.

Once *POC-NN* prototypes are found, a separating linear hyperplane is generated and orthogonally placed in the middle of the distance between these *POC-NN* prototypes. This hyperplane acts as a decision boundary (similar to a *Voronoi Diagram*). The boundary of each region is defined by the corresponding hyperplane generated by *POC-NN* prototype lying in each region. The following describes the detail of prototype selection by *POC-NN* algorithm.

3.2 The POC-NN Algorithm for Two-Class Classification Problem

In this section, the proposed algorithm for two-class classification problem is presented and in the later subsection, the proposed algorithm is extended to cover multi-class classification problems.

Prototype Selection by POC-NN Algorithm for Two-Class Classification

Let S be a training set of n patterns composing of two classes, and *POC-NN-SET* initially be an empty *POC-NN* prototypes set.

```
Function SELECTING-POC-NN (S: Dataset)
1. Find a POC-NN prototype in $S$ by using
   $(\mathbf{x}_{p1}, \mathbf{x}_{p2})$ = FINDING-POC-NN (S).
2. Determine the center point $\mathbf{c} = \frac{\mathbf{x}_{p1}+\mathbf{x}_{p2}}{2}$.
3. Create a separating hyperplane $H$: $\{\mathbf{x}|\mathbf{w}\cdot\mathbf{x}-b=0\}$,
   where $\mathbf{w} = \frac{\mathbf{x}_{p1}-\mathbf{x}_{p2}}{||\mathbf{x}_{p1}-\mathbf{x}_{p2}||}$ and $b = \mathbf{w}\cdot\mathbf{c}$.
4. Save $(\mathbf{x}_{p1}, \mathbf{x}_{p2})$ and corresponding $H$ into the POC-NN-SET.
5. Divide all patterns of $S$ into two regions, namely R1 and R2, where
   $R1 = \{\mathbf{x}_i \in S|\mathbf{w}\cdot\mathbf{x}_i - b \geq 0\}$, and
   $R2 = \{\mathbf{x}_i \in S|\mathbf{w}\cdot\mathbf{x_i} - b < 0\}$, $\forall i, i = 1, \ldots, n$.
6. Find any misclassification in both regions.
7. If any misclassification exists in region R1
   Then
8.       Consider all data in this region R1 as a new data set
         Call SELECTING-POC-NN (R1).
   Endif
9. If any misclassification exists in region R2
   Then
10.      Consider all data in this region R2 as a new data set
         Call SELECTING-POC-NN (R2).
   Endif
11.If no more misclassification exists
   Then
12.      Return POC-NN-SET as a set of selected prototypes.
         Stop.
   Endif
```

Figure 3.2 shows an example of how the algorithm works for non-linearly separable problems. The initial separating line (H_1) created by a *POC-NN* ($\mathbf{x}_{p_1}$, $\mathbf{x}_{p_2}$) is shown in Figure 3.2(a). However, this line still creates misclassified training patterns lying on the right side of this line. All the training patterns lying on this side are considered as a new data set. The second line (H_2) as shown in Figure 3.2(b) is introduced to resolve the previously misclassified patterns. After performing the algorithm, a set of selecting prototypes corresponding to the *POC-NN-SET* consisting of two pairs of $\{(\mathbf{x}_{p_1}, \mathbf{x}_{p_2})\}$ is obtained. The patterns from $S^{(1)}$ are separated into two regions (region 1 and region 2). Each region has a prototype ($\{\mathbf{x}_{p_1}\}$) representative class 1. The patterns from $S^{(2)}$ are not separated, however, as they enclose two prototypes ($\{\mathbf{x}_{p_2}\}$) representative class 2. The identical *POC-NN* patterns are always obtained and determine the optimal separating hyperplane, since the margin(distance) between two closest patterns of opposite classes is maximum for the local region which induces good generalization. Therefore, the proposed algorithm is too stringent to arrive at a "good" separating hyperplane.

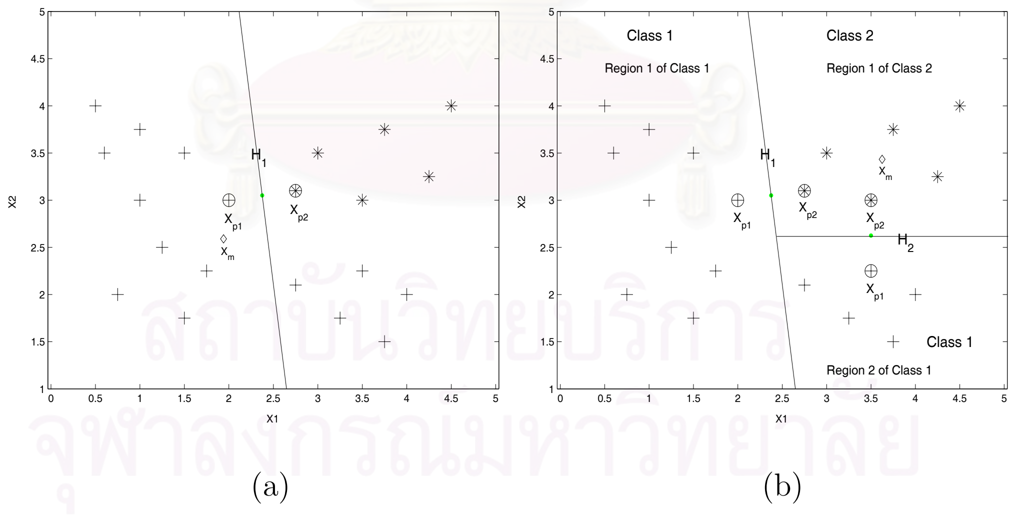


Figure 3.2: This example shows *POC-NN* separating hyperplanes: (a) initial separating line (H_1); (b) second separating line (H_2).

3.3 Reducing Complexity and Sensitivity to Noise

In order to reduce the complexity and sensitivity to noise as well as to avoid overfitting for overlapping data, the separating condition is relaxed. A separating hyperplane can be considered as a slab of width alpha (α), called *acceptance interval.* All patterns lying within the acceptance interval of the slab are considered as correctly classified patterns, and conversely considered as noisy and/or outlier patterns which can be ignored or discarded. The *acceptance interval* (α), is defined as follows:

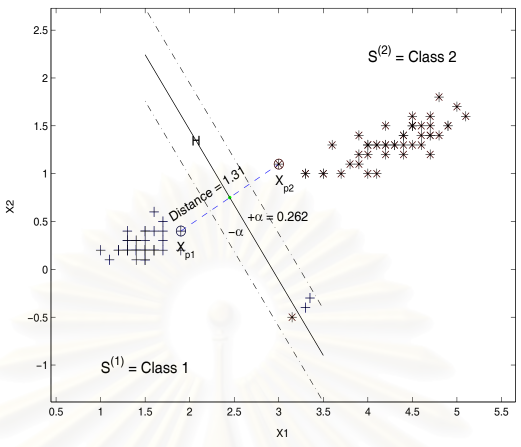
Suppose S be a training data set of n points with two classes and $\mathbf{x}_i \in S$.

Definition 3.1. *Let* $\{\mathbf{x}|\mathbf{w}\cdot\mathbf{x}-b=0\}$ *be a hyperplane (*H*).* α *is an acceptance interval if* $|\mathbf{w}\cdot\mathbf{x}_i-b|<\alpha$*, and* $\alpha>0$ *such that any* $\mathbf{x}_i$ *that falls inside* α *is assumed to be correctly classified.*

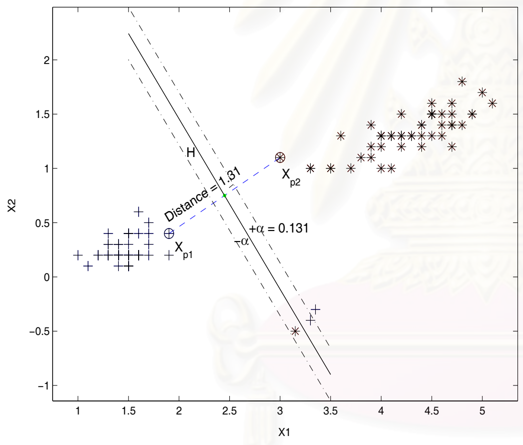
The acceptance interval is proportional to the distance d between two *POC-NN* prototypes, and is defined a priori to the training process. In other words, α is defined as $\alpha_r \times d$ for 0< α_r <0.5. This α_r is called α-ratio.

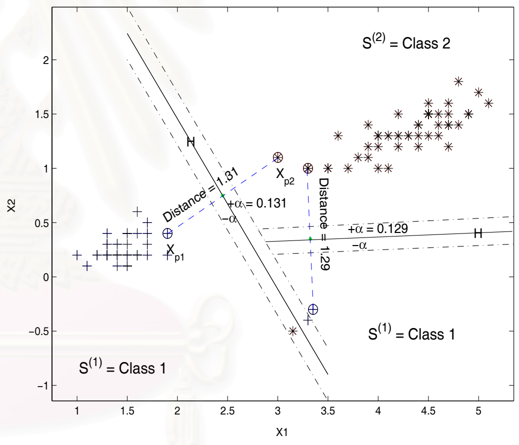
These α values for each hyperplane are not necessary the same values, but they depend on α-ratio and the distance between corresponding *POC-NN* prototypes.

Figure 3.3 shows an example of an acceptance interval α defined by giving the α-ratio 1:5 or 0.2 (a), and 1:10 or 0.1 (b) and (c). In Figure 3.3(a), the algorithm will stop after one iteration and generate two *POC-NN* prototypes. Each *POC-NN* is a prototype representative of each class, even though there are three misclassified patterns lying within the acceptance interval. They are considered as noisy and/or outlier patterns which are ignored. There are only two instead of three misclassified patterns as with the case in Figure 3.3(b). Figure 3.3(c) illustrates the case where one misclassified pattern is still present. The algorithm therefore continues and generates two additional *POC-NN* prototypes.



(a) α-ratio(α_r)=1:5(0.2)





(b) α-ratio(α_r)=1:10(0.1) (c) α-ratio(α_r)=1:10(0.1)

Figure 3.3: All points lying inside of the slab with radius alpha (α) are considered as noisy patterns. The ratio between an α value and the distance between corresponding *POC-NN* prototypes is pre-defined by the user. (a) α-Ratio is equal to 1:5(0.2). (b,c) α-Ratio is equal to 1:10(0.1).

3.4 Analysis of the POC-NN Algorithm

Some interesting consequences in selecting *POC-NN* prototype are worth discussing. Given a training set S of n patterns (n >1) in d dimensions composing of two subsets $S^{(1)}$ and $S^{(2)}$, whose sizes are $n^{(1)}$ and $n^{(2)}$. Both $S^{(1)}$ and $S^{(2)}$ are in different classes, namely, class 1 and class 2, respectively, and there never exist two patterns with different class label on the position ($S^{(1)} \cap S^{(2)}=\varnothing$).

Property 1. *The selecting POC-NN algorithm converges after p iterations, where p is the number of POC-NN patterns having the values between 1 and n-1.*

Proof. Let start the proof with $n^{(1)}$ and $n^{(2)}$ = l, so these two patterns are *POC-NN* pattern to each other and its hyperplane can be constructed to make them linearly separable. If one add a new pattern into this space, one always has two cases:

Case 1: A new pattern lies on the correct side, so all data are still linearly separable and correctly classified. The algorithm stops after one iteration.

Case 2: A new pattern lies on the wrong side. Here, one need to consider only the data lying on that side. A new pattern will become a new *POC-NN* pattern and a new hyperplane that makes these data linearly separable, is introduced. The algorithm will stop after two iterations.

If one gradually add a new pattern until there are n patterns of training set S, one still always have two cases as before. The algorithm will stop after n-1 iterations.

On the contrary, assuming that the algorithm does not converge. This would only be possible if no hyperplane could be constructed. But note that the hyperplane cannot be constructed if the nearest patterns of the two classes in data group cannot be found. It would only be possible if the data group has only the same class data and that is also the stopping condition of the algorithm. This is a contradiction and hence non-convergence does not occur. □

The consequence of this property yields that all training patterns are separated into regions of correctly classified classes after convergence. Therefore, a set of patterns in each region is a *consistent set.* However, the whole set of the selected *POC-NN* prototypes may not be a *consistent set.* This proof can be shown by giving a counterexample that the obtained set of prototypes generated by the proposed algorithm does not give 100 % accuracy on the training set by using *1-NN* rule. Similar to the Tomek's algorithm, the consistency is not guaranteed. A counterexample has been shown in [30]. However, there is no theoretical evidence on how the consistency of the condensed set relates to the generalization abilities. It may have an arbitrarily poor performance when applied to unseen patterns as shown by some experimental examples in Section 4.

Property 2. *The time complexity of the selecting POC-NN algorithm is* $O(dn^2p)$*.*

Proof. The upper bound of the time complexity is derived step by step in the following analysis. In step 1 of the selecting *POC-NN* algorithm, calculations for finding a *POC-NN* prototype take in worst case $O(dn^2)$, and $O(dn)$ in steps 5 and 6. The time complexity is bounded by $O(dn^2 + dn)$ after one iteration. If the algorithm does not converge, step 1 of the algorithm will continue with at most $(n-1)$ patterns. The time complexity is bounded by $O(d(n-1)^2 + d(n-1))$ after two iterations. From property 1, the algorithm converges after p iterations. Therefore, the total time complexity after p iterations is bounded by $O(dn^2 + dn) + O(d(n-1)^2 + d(n-1)) + \cdots + O(d1^2 + d1)$ which is equal to $O(dn^2p)$. $\square$

In comparison with *Condensing* and *Editing NN* algorithm, the complexity of computing the condensed subset and edited subset are $O(dn^3)$ and $O(d^3 n^{\lfloor d/2 \rfloor} \ln n)$, respectively [17, 3]. Both time complexities are higher than *POC-NN's.* In addition, the computational complexity of both Tomek's algorithm [8] and Gowda-Krishna's algorithm [9] which attempt to keep only prototypes close to the decision boundary is considerably higher than the original *CNN* algorithm. In case of Tomek's algorithm, only the step for finding a special subset, called *Gabriel Neighbors* [30], already takes $O(n^3)$ time. For large n such as that in OCR applications shown by USPS data [21] experimental example in Section 4, it is not feasible. Gowda-Krishna's algorithm takes longer time than Tomek's algorithm in some cases as shown in [9] since it requires a pre-process for finding *Mutual Neighborhood Value (MNV)* of each pattern to order the patterns according to *MNV*, and a post-process for reducing the number of condensed set of prototypes.

Property 3. *The selecting POC-NN algorithm is order independent.*

Proof. The training set can have $n!$ permutation orders of how to present the data. In steps 3 and 4 of finding a *POC-NN* pattern algorithm, the centroid or mean of a set of points (patterns) is calculated. The addition remains invariant to the order of the data elements. Thus, the mean operation is order independent. Finding the nearest point to the centroid is also order independent since the centroid and the distance between the centroid and all points from different classes are fixed values. The shortest distance is found and the corresponding point is selected as the representative prototype. If there is a tie between two or more points for the closest distance from the centroid, the point with the smallest feature mean value is selected. If this is the same, the point with the smaller value of feature 1 is selected. If this is still the same, the point with the smaller value of feature 2 is selected, and so on. Thus, the distance measured is order independent and the minimum of these distances remains the same no matter what the order of the presentation data is. The other steps in the algorithm are obviously order independent, thus the selecting *POC-NN* algorithm is order independent. $\square$

3.5 Algorithm for Multi-Class Classification Problem

In this section, the proposed algorithm is extended for multi-class classification problem by using a combination of many two-class classifiers into a multi-class classifier. In contrast to the standard approach to the m-class problem by a *'one-against-rest' (1-v-r)* like scheme, an alternative approach, so called *'one-against-one' (1-v-1)* scheme, with pairwise classification having $\frac{m(m-1)}{2}$ binary classifiers is used.

'one-against-one' (1-v-1) scheme

All m-class training data are classified into pairwise two-class training sets and perform the selecting *POC-NN* algorithm on each two-class training set. It is possible that the same pattern becomes a *POC-NN* prototype more than once as it has been selected by another combination of pairwise two-class training set. However, it is still the same representative prototype for its class.

Suppose having a training set S with m classes and there never exist two or more patterns with different class on the same position, such that $S = S^{(1)} \cup \ldots \cup S^{(m)}$ and $\varnothing = S^{(1)} \cap \ldots \cap S^{(m)}$.

1-v-1 POC-NN Algorithm for k-Class Classification

```
1. Initialize: $q = 0$.
2. For $1 \leq i \leq m$ Do
3.     For $i + 1 \leq j \leq m$ Do
4.         $q = q+1$.
5.         $S_q = S^{(i)} \cup S^{(j)}$.
6.         $POC\text{-}NN\text{-}SET_q = SELECTING\text{-}POC\text{-}NN\ (S_q)$.
7.     Endfor
8. Endfor
9. POC-NN-SET for m-Class $= \bigcup_{l=1}^{q} POC\text{-}NN\text{-}SET_l$.
```

Figure 3.4 shows an example of how *1-v-1 POC-NN* Algorithm for m-Class Classification works (m=3). Each pattern in classes 1, 2, and 3 is denoted by the symbol "+", "$*$", and "$\times$", respectively. Figure 3.4(a) shows the final prototype in *POC-NN-SET* illustrated by symbol "$\bigcirc$". The final prototype in *POC-NN-SET* consists of all prototypes in $POC\text{-}NN\text{-}SET_1$, $POC\text{-}NN\text{-}SET_2$, and $POC\text{-}NN\text{-}SET_3$ shown in Figure 3.4(b), (c), and (d), respectively.

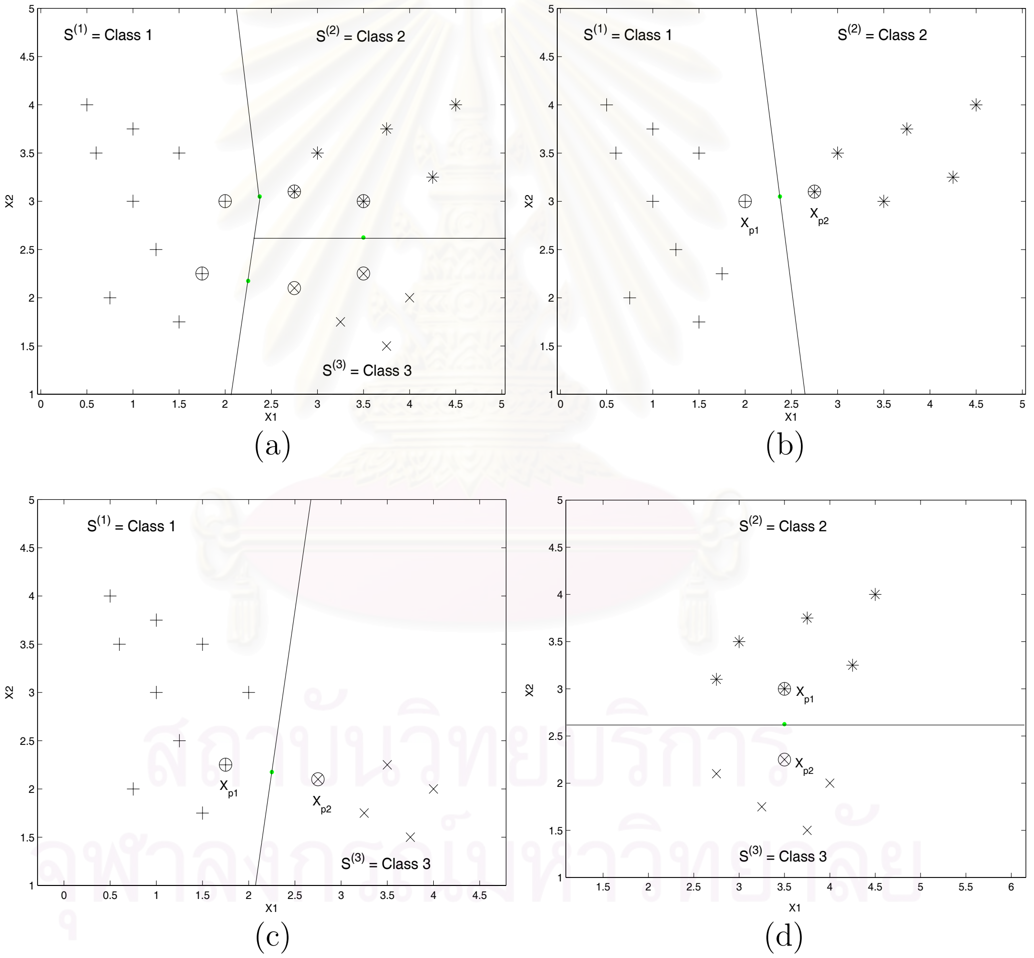


Figure 3.4: This example shows how *1-v-1 POC-NN* Algorithm for k-Class Classification works: (a) The final *POC-NN-SET* set; (b) $POC\text{-}NN\text{-}SET_1$ set (q=1); (c) $POC\text{-}NN\text{-}SET_2$ set (q=2); and (d) $POC\text{-}NN\text{-}SET_3$ set (q=3).

After performing the proposed algorithm for each two-class pair, *POC-NN* prototypes are obtained representative for each class. The time complexity for multi-class classification will increase to $O(\frac{m(m-1)}{2}(dn^2p))$, where m is the number of classes, d is the number of dimensions, n is the number of patterns with two classes, and p is the number of *POC-NN* prototypes.

CHAPTER IV

EXPERIMENTAL RESULTS FOR PATTERN CLASSIFICATION

This chapter presents the data sets and experimental results. Moreover, the performance in accuracy and prototype rate as well as in training time of the proposed algorithm is evaluated and compared with others *Nearest Neighbor* techniques.

4.1 Data sets

The proposed algorithm is tested and evaluated on a number of standard benchmark classification data sets, both artificial and real. These data sets are taken from UCI Repository of machine learning databases [20] except for the first and second ones which are the USPS data [21] and DNA from Statlog [22]. The properties of the data sets are given in Table 4.1. In the first four data sets, the training and test sets are separated. In the others part from the first four, the original data sets are separated into training and test sets (2:1 ratio) by using the following criteria. The first pattern belongs to the test set, the second and third patterns belong to the training set, the fourth pattern belongs to the test set, and the fifth and sixth patterns belong to the training set, and so on. Euclidean distance (L_2-norm) is used to measure dissimilarity and *1-NN* rule to test the results in all experiments.

Table 4.1: Properties of the data sets used

Data sets	No. of features	No. of classes	No. of trg. patterns	No. of test patterns
1. USPS	256	10	7291	2007
2. DNA	180	3	2000	1186
3. Sonar	60	2	104	104
4. Vowel	10	11	528	462
5. Cancer	9	2	466	233
6. Wine	9	7	118	60
7. Liver	6	2	230	115
8. Thyroid	5	3	143	72
9. Iris	4	3	100	50
10. Spiral	2	2	129	65

4.2 Experimental Results

4.2.1 Accuracy of Classification

The results obtained from these data sets are reported in Table 4.2. The prototype rate (PR %) computed from the percentage of prototypes to all training patterns, and the accuracy rate (AR %) computed from the percentage of correctly predicted test patterns to all test patterns obtained using *POC-NN*, *Condensed Nearest Neighbor (CNN)*, *Tomek's algorithm (Tomek)*, *Gowda-Krishna's algorithm (GKA)*, and the accuracy rate (AR %) obtained using all prototypes (*NN*) are given for each data set. N/A (for not applicable/available) in the Table 4.2 signifies that the learning scheme did not finish training. If learning could not be completed within the time period of two weeks then it was terminated and marked N/A, since taking time of two weeks is not feasible for the real world application.

The results obtained by *POC-NN* are mostly better than the results obtained by *CNN*, especially, the accuracy rates using *POC-NN* are better than *CNN* in all cases, and

Table 4.2: The comparison results of *POC-NN*, *CNN*, *Tomek*, *GKA*, and *NN*

Data sets	POC-NN		CNN		Tomek		GKA		NN	
	PR %	AR %	PR %	AR %	PR %	AR %	PR %	AR %	PR %	AR %
1. USPS	28.20	**93.42****	11.84	91.58	N/A	N/A	N/A	N/A	100.00	94.37
2. DNA	60.70	**74.37**	44.30	72.30	N/A	N/A	N/A	N/A	100.00	76.39
3. Sonar	59.62	86.54	57.69	86.54	43.27	**90.38**	52.88	87.50	100.00	92.31
4. Vowel	50.19	**54.11****	19.69	47.19	20.83	51.52	18.56	55.84	100.00	56.28
5. Cancer	8.37	**94.42**	10.30	93.13	10.94	91.85	7.94	91.85	100.00	93.56
6. Wine	43.22	**71.67**	44.07	66.67	42.37	68.33	38.98	68.33	100.00	68.33
7. Liver	58.69	54.78	56.52	53.91	55.65	**58.26**	50.00	57.39	100.00	59.91
8. Thyroid	20.27	88.89	20.97	88.89	16.08	**95.83**	15.38	90.28	100.00	91.67
9. Iris	17.00	**98.00**	20.00	**98.00**	18.00	**98.00**	16.00	**98.00**	100.00	98.00
10. Spiral	56.59	**95.38***	37.98	89.23	37.98	93.85	34.11	89.23	100.00	96.92

Best accuracy rates (AR %) among these algorithms except *NN* are bold-faced. Tomek and GKA are not applicable (N/A) on USPS and DNA, because they require unacceptable time to run on these data. The symbols "**" and "*" indicate the statistically significant difference to *CNN* at the level 95 % and 90 %, respectively.

Tomek and *GKA* in all cases except Sonar, Liver and Thyroid. However, the prototype rates using *POC-NN* are better than the other algorithms in some cases. In the cases of cancer, wine, thyroid and iris data, *POC-NN* shows better results in both accuracy and prototype rates than the results from the *CNN*. Moreover, in case of Wine, *POC-NN* even gives better accuracy rate than *NN* by using all the training set as prototypes, and in case of Iris, *POC-NN* gives equally accuracy rate to *NN* by using all the training set as prototypes. Even though the accuracy rate of the training data is very high, however, it cannot guarantee to obtain a good accuracy rate of test (untrained) data. The consistent set is not necessarily related to the generalization abilities.

From these results, the hypothesis was formulated that using *POC-NN* algorithm is more effective than using *CNN* algorithm. To determine if this hypothesis was statistically significant, the results using a test of significance involving differences of proportions [31] were analyzed. This test showed that the level of significance of the hypothesis is 0.05 and 0.1, indicated by symbols "**" and "*" in the Table 4.2, which provide a 95 % and 90 %, respectively, confidence level according to one-tailed proportions of the normal curve that it is correct. For a mathematical derivation of this statistical result, see Appendix B.

4.2.2 Computational Time

The training time is also compared required for the results shown in Table 4.2. The less computational time is required the more efficiency for an algorithm is obtained, even though it is carried out offline. The time comparisons are summarized in Table 4.3. All experiments were done on the same Pentium III-1GHz computer with 256 MB RAM, and all algorithms were implemented in MatLab version 6.5. These four different training algorithms were implemented in different training programs to obtain the prototype for each data set. The *POC-NN* algorithm has the best training time for all data sets. In case of sonar, *POC-NN*'s training time is approximately 10 times faster than the *CNN*'s training time, and approximately 100 times faster than the *Tomek*'s and the *GKA*'s training time. The *Tomek* and the *GKA* algorithms require very high computational time and are not applicable for some real world problems, such as USPS and DNA. The time and space complexities of the proposed *POC-NN* method are very competitive.

4.3 Evaluate the POC-NN Algorithm

To increase statistical significance of the results on the data set whose training and test sets are not separated, the *K*-fold cross-validation technique was conducted which is one of the simplest and most widely used method for estimating prediction error [32]. The average cross-validation estimates of prototype and accuracy rate are shown in

Table 4.3: The time comparisons of *POC-NN* and the others in seconds

Data sets	Training Time in seconds			
	POC-NN	CNN	Tomek	GKA
1. USPS	297650.00	560370.00	N/A	N/A
2. DNA	10927.00	74474.00	N/A	N/A
3. Sonar	8.94	90.20	925.32	882.68
4. Vowel	55.18	81.29	14045.00	2873.77
5. Cancer	23.42	84.51	826.74	498.82
6. Wine	3.29	20.15	101.51	189.63
7. Liver	4.49	38.07	620.41	1021.83
8. Thyroid	1.38	6.65	56.54	24.89
9. Iris	0.80	2.56	30.42	8.87
10. Spiral	0.66	8.77	41.81	33.12

Tomek and GKA algorithms are not applicable (N/A) on USPS and DNA, since they require unacceptable time (more than two weeks) to run on these data sets.

Table 4.4. Compare with the results in Table 4.2, for the three-fold cross-validation (K=3), *POC-NN* still shows better results in accuracy rate than the results from the *CNN* in all cases, and the best results in accuracy rate among all algorithms except *NN* in all cases except Thyroid and Iris. For the five-fold cross-validation (K=5), *POC-NN* still shows better results in accuracy rate than the others in all cases except Cancer, Wine and Thyroid. The Tomek's algorithm also shows the best results in some cases. However, it requires very high computational time. The symbols "**" and "*" indicate 95 % and 90 %, respectively, confidence interval for estimating the difference between accuracy of *POC-NN* and *CNN* using a one-tailed paired t-test [33]. For a mathematical derivation of this statistical result, see Appendix C.

Table 4.4: The results comparisons of *POC-NN* and the other algorithms by conducting the three-fold (K=3) and five-fold (K=5) cross-validation

K-Fold	Data sets	POC-NN		CNN		Tomek		GKA		NN	
		PR %	AR %	PR %	AR %	PR %	AR %	PR %	AR %	PR %	AR %
K=3	Cancer	10.23	**94.71**	10.09	94.13	11.16	92.42	8.44	92.85	100.00	94.85
	Wine	43.82	**73.05**	41.86	71.37	40.45	**73.05**	38.20	72.50	100.00	74.19
	Liver	57.35	**60.64****	59.81	58.07	57.78	57.19	50.06	57.46	100.00	58.55
	Thyroid	18.15	90.25	19.53	90.24	16.05	**93.49**	14.42	90.23	100.00	93.96
	Iris	16.00	96.00	16.00	94.00	14.67	**98.00**	13.00	92.00	100.00	96.00
	Spiral	58.50	**96.39**	38.40	92.79	38.14	93.82	35.56	91.78	100.00	98.46
K=5	Cancer	9.48	93.71	11.16	**94.13**	10.66	93.13	9.12	92.99	100.00	94.85
	Wine	42.42	71.86	40.45	**73.59**	37.93	73.57	36.52	73.57	100.00	75.25
	Liver	58.91	**60.29****	58.70	59.42	58.26	59.13	53.12	60.00	100.00	62.23
	Thyroid	16.05	89.77	16.86	89.77	16.05	**94.42**	14.19	93.49	100.00	93.49
	Iris	13.67	**99.00***	7.50	96.33	13.83	98.67	12.00	94.00	100.00	96.00
	Spiral	60.17	**97.41**	33.89	96.90	34.79	**97.41**	34.67	94.35	100.00	100.00

Best accuracy rates (AR %) among these algorithms except *NN* are bold-faced.

4.4 Reducing the Complexity

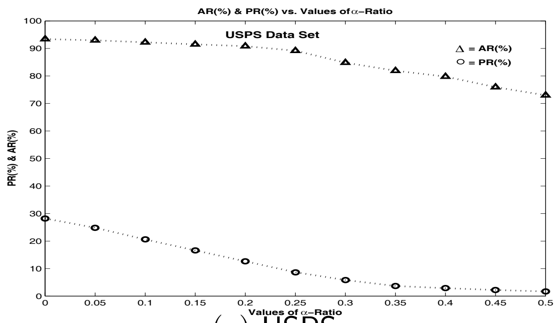
In order to reduce the number of prototypes, the concept of *acceptance interval* (α) is considered. Table 4.5 shows the results performed on the same data sets by using *POC-NN* with different α-ratio and *CNN*. The value of an α-ratio for each data set is chosen in order to maintain the same accuracy rate obtained by using without α-ratio. Most accuracy rates remain the same by reducing the number of prototypes. In many cases, *POC-NN* with α-ratio gives better both accuracy and prototype rates than *CNN*. In the case of DNA, Sonar, Vowel and Liver, there is a significant reduction in the number of prototypes used. So the reduced set of prototypes is very useful. By using the concept of *acceptance interval* (α), the value of α can be regulated in order to control the complexity and to avoid the overfitting of the model.

Table 4.5: The results comparisons of *POC-NN* by using without α-ratio ($\alpha_r = 0$) and with different α-ratio ($\alpha_r <> 0$), and using *CNN*

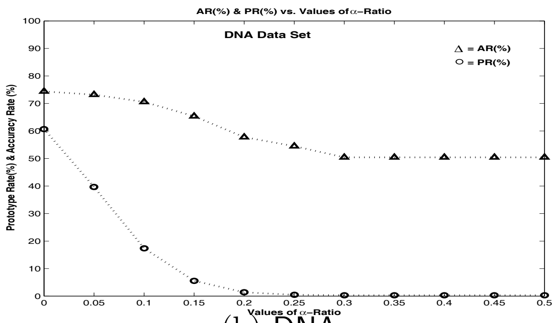
Data sets	POC-NN($\alpha_r = 0$)			POC-NN($\alpha_r <> 0$)				CNN		
	Prototypes		Acc.	α_r	Prototypes		Acc.	Prototypes		Acc.
	#	%	%	ratio	#	%	%	#	%	%
1. USPS	2056	28.20	**93.42****	0.25	631	**8.66**	89.16	863	11.84	91.58
2. DNA	1214	60.70	**74.37***	0.05	793	**39.65**	73.19	886	44.30	72.30
3. Sonar	62	59.62	**86.54**	0.01	48	46.15	**86.54**	60	57.69	**86.54**
4. Vowel	265	50.19	**54.11****	0.50	173	32.76	51.08	104	**19.69**	47.19
5. Cancer	39	8.37	**94.42**	0.01	37	**7.94**	**94.42**	48	10.30	93.13
6. Wine	51	43.22	**71.67**	0.01	45	**38.14**	**71.67**	52	44.07	66.67
7. Liver	135	58.69	**54.78**	0.40	6	**2.61**	**54.78**	130	56.52	53.91
8. Thyroid	29	20.27	**88.89**	0.03	22	**15.38**	**88.89**	30	20.97	**88.89**
9. Iris	17	17.00	**98.00**	0.35	15	**15.00**	**98.00**	20	20.00	**98.00**
10. Spiral	73	56.59	**95.38***	0.08	72	55.81	**95.38***	49	**37.98**	89.23

Best accuracy and prototype rates between *POC-NN* and *CNN* are bold-faced. The symbol "**" and "*" indicate the statistically significant difference to *CNN* at the level 95 % and 90 %, respectively.

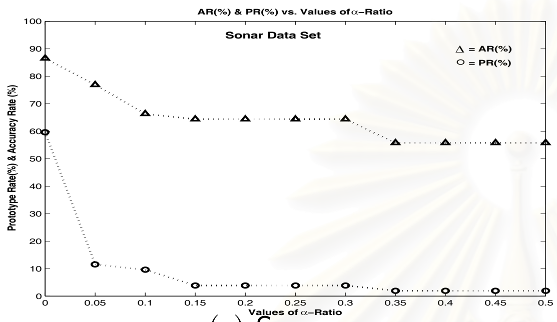
Figure 4.1 shows that the number of *POC-NN* prototypes and the prototype rate depend on choosing a prior value of an α-ratio. The α-ratio can also be employed to define the prototype rate and accuracy rate. In most cases, the accuracy and prototype rates decrease when the α-ratio increases. However, in cases of Liver, better accuracy rate with lesser prototype rate is obtained, even though the α-ratio is increased.



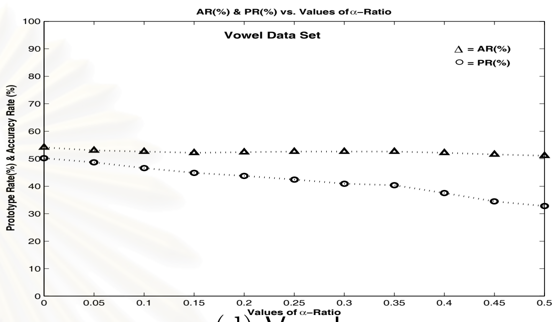
(a) USPS



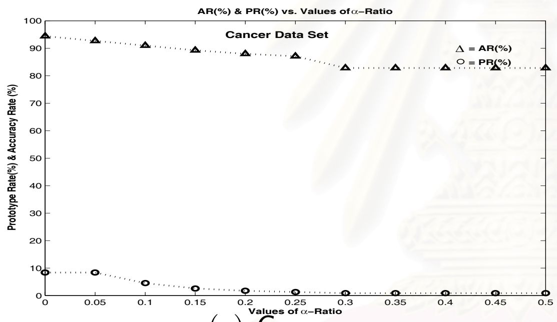
(b) DNA



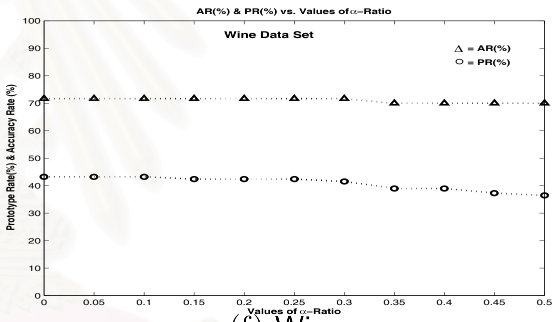
(c) Sonar



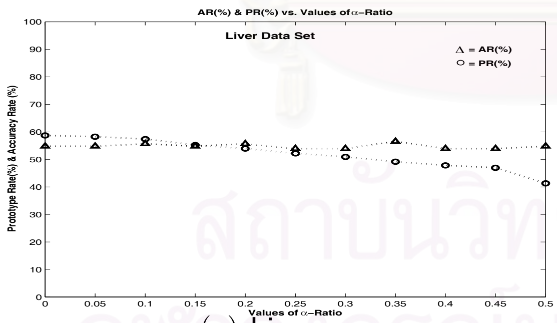
(d) Vowel



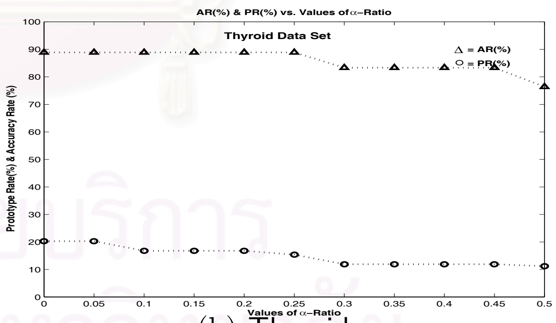
(e) Cancer



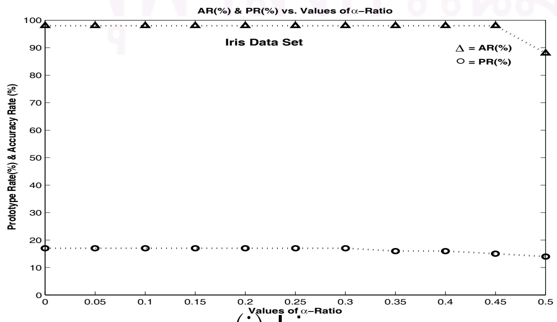
(f) Wine



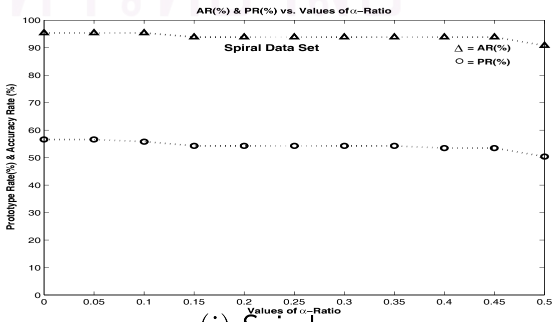
(g) Liver



(h) Thyroid



(i) Iris



(j) Spiral

Figure 4.1: Prototype Rate (PR %) and Accuracy Rate (AR %) as a function of α-ratio

CHAPTER V

THE METHODOLOGY OF POC-NN FOR REGRESSION PROBLEM

In this chapter, *POC-NN* algorithm is generalized so that it can handle regression problems. The idea is to find a subset of the original sampling set that suffices for linear interpolation, and throw away the remaining data. Intuitively, it seems reasonable to keep the original points (patterns) that are used for building the linear interpolation line, while some points lying on or near this line should be discarded.

5.1 Finding POC-NN Patterns Algorithm for Regression

Considering n samples of a given data set S of

$$X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \Re^d \tag{5.1}$$

and their corresponding function value

$$Y = \{y_1 = f(\mathbf{x}_1), \ldots, y_n = f(\mathbf{x}_n)\}, y_i \in \Re, \tag{5.2}$$

where $\Re$ denotes real number.

Before finding prototypes for regression problem for a given data sampling set S with dimension d, all sampling data set S have to be separated into two parts (classes), namely, class 1 and class 2. The criteria through simple odd and even sampling number of data is used to obtain a new data sampling set S' with dimension *d+1*, where

$$S' = \begin{cases} S^{(1)} = \{(\mathbf{x}_i, y_i)\}, & \text{if } i \text{ is odd,} \\ S^{(2)} = \{(\mathbf{x}_j, y_j)\}, & \text{if } j \text{ is even.} \end{cases} \tag{5.3}$$

The algorithm to find the Pairwise Opposite Class-Nearest Neighbor for regression is given as follows:

```
Function FIND-POC-NN-R ($S'$: Dataset)

1. Let $S^{(1)}$ and $S^{(2)}$ be a training set defined by Eq. (5.3),
   and $d+1$ be the dimension of $S'$.
2. Let $\mathbf{x}_1 \in S^{(1)}$ be the first element.
3. Let $\mathbf{x}_{p2} \in S^{(2)}$ be the nearest pattern to $\mathbf{x}_1$.
4. For $1 \leq i \leq d$ do
5.         $Z = \{\mathbf{x}_{p1_i} | \mathbf{x}_{p1_i} \in S^{(1)}$ and let $\mathbf{x}_{p1_i}$ be the $i^{th}$ nearest pattern to $\mathbf{x}_{p2}$.$\}$
6. End
7. Return $(Z, \mathbf{x}_{p2})$ as POC-NN prototypes.
```

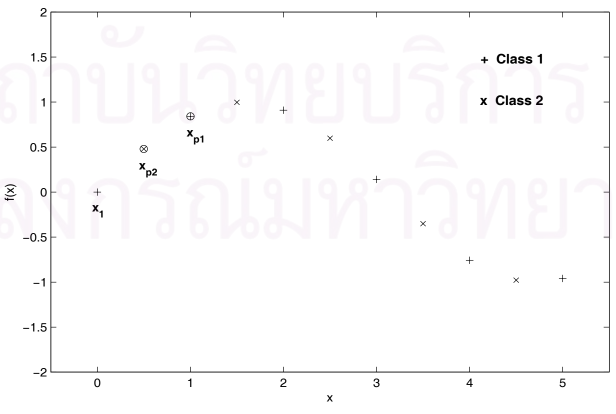


Figure 5.1: This example shows how to find the prototypes for regression.

Figure 5.1 shows an example of how algorithm for finding *POC-NN* prototypes for regression works. A given 11 data sampling set S is a one dimension sine function. Data set S is separated into $S^{(1)}$ and $S^{(2)}$. Each pattern in $S^{(1)}$ and $S^{(2)}$ is denoted by the symbol "+" and "×", respectively. The *POC-NN* prototypes ($\mathbf{x}_{p1_1}$,$\mathbf{x}_{p2}$) enclosed in circle symbols "○".

Once *POC-NN* patterns are found, a separating hyperplane is generated and placed in between these *POC-NN* patterns. This hyperplane has a very nice characteristic for both classification and regression problems. It acts as a perceptron for local classification and as a functional approximator for local sampling data locating near these *POC-NN* prototypes. This characteristic is so simple, however, it is useful for both problems. In case of dimension (d) is greater than one, it is necessary to find *d+1* numbers of *POC-NN* prototypes ($\mathbf{x}_{p1_1}$,$\mathbf{x}_{p1_2}$,...,$\mathbf{x}_{p1_d}$,$\mathbf{x}_{p1_{d+1}}$,$\mathbf{x}_{p2}$) and a separating hyperplane which is generated and placed among all (*d+1*) *POC-NN* prototypes.

5.2 The POC-NN Algorithm for Regression Problem

The algorithm for selecting *POC-NN* patterns as a selected (compressed) data set for regression is as follows. Initially, let S be a sample set with dimension d, and S' be a training set defined by (5.3), and *POC-NN-SET* initially be an empty *POC-NN* prototypes set for regression problem.

```
Function SELECT-POC-NN-R (S′: Dataset)

1. Find POC-NN prototypes in $S'$ by using $(\mathbf{x}_{p1_i},\mathbf{x}_{p2}) =$ FIND-POC-NN-R ($S'$).
2. Create a hyperplane $H$: $\{\mathbf{x}|\mathbf{w}\cdot\mathbf{x}-b=0\}$,
   $\mathbf{x}_i \in S'$ connecting all of d+1 POC-NN prototypes.
3. Save all d+1 POC-NN prototypes and corresponding $H$ into the POC-NN-SET.
4. Divide all patterns $\mathbf{x}_i$ of $S'$ into two regions, namely R1 and R2, where
   $R1 = \{\mathbf{x}_i \in S'|\mathbf{w}\cdot\mathbf{x}_i-b \geq 0\}$ and
   $R2 = \{\mathbf{x}_i \in S'|\mathbf{w}\cdot\mathbf{x}_i-b < 0\}$.
5. Find any misclassification in both regions.
6. If any misclassification exists in region R1
   Then
7.         Consider all data in R1 as a new data set
           Call SELECT-POC-NN-R (R1).
   End
8. If any misclassification exists in region R2
   Then
9.         Consider all data in R2 as a new data set
           Call SELECT-POC-NN-R (R2).
   End
10.If no more misclassification exists
   Then
11.        Return POC-NN-SET as a set of selected prototypes.
12.        Stop.
   End
```

Figure 5.2 shows an example of how the selecting *POC-NN* algorithm for regression works. The initial connecting line (H_1) created by *POC-NN* ($\mathbf{x}_{p1_1}$,$\mathbf{x}_{p2}$) is shown in Figure 5.2(a). However, this line still creates misclassified training patterns on the right side of the line. All training patterns on this side are considered as new data sets. The second (H_2), third (H_3), and last (H_4) as shown in Figure 5.2(b), (c), and (d), respectively, are introduced to resolve any previously misclassified patterns. After performing the algorithm, a set of selecting prototypes corresponding to the *POC-NN-SET* consisting of four pairs of ($\mathbf{x}_{p1_1}$,$\mathbf{x}_{p2}$) is obtained. The number of sampling data is reduced from 11 to 8. Only the eight *POC-NN* prototypes and their corresponding hyperplanes are used as the reconstruction values of the training samples and approximation values of a function. The final result shows in Figure 5.3(a).

5.3 Approximating Algorithm by POC-NN

To reconstruct the value of the original data and also estimate the value of an untrained sample $\mathbf{x}_u$, the *Nearest Neighbor (NN)* rule is used by measuring the nearest distance between $\mathbf{x}_u$ and the set of all *POC-NN* prototypes whose dimension must be reduced from *d+1* to *d*. Let $\mathbf{x}_u \in \Re^d$ be an untrained data pattern, and $\mathbf{x}_p \in \Re^d$ be a *POC-NN* prototype whose dimension is reduced from *d+1* to *d*. The detail of this algorithm is as follows:

Approximating Algorithm

```
1. For each point $\mathbf{x}_u$ Do
2.         Identify the nearest POC-NN prototype ($\mathbf{x}_p$) to $\mathbf{x}_u$ and
           its corresponding hyperplane $H$ as a function approximator $\tilde{f}$.
3.         Calculate the reconstruction/approximation value by $y_u = \tilde{f}(\mathbf{x_p})$.
4. End
```

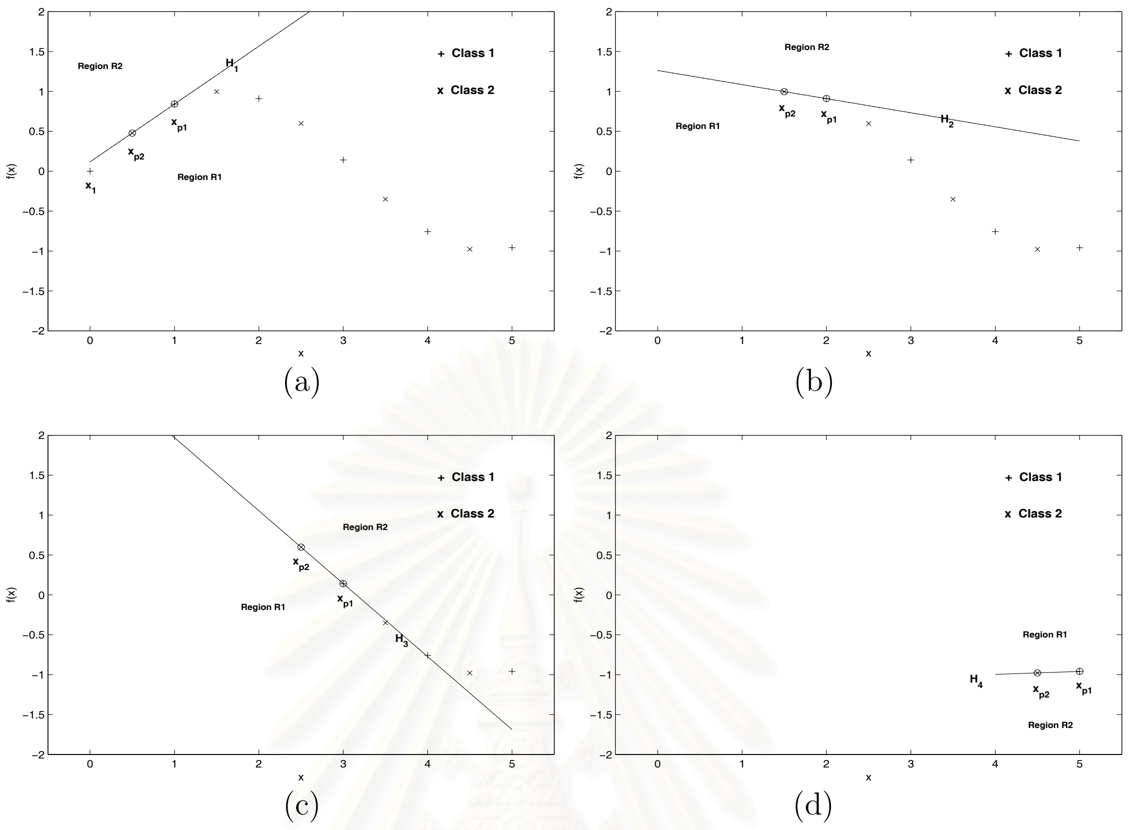


Figure 5.2: This example shows how the proposed algorithms works: (a) initial connecting line (H_1); (b) second connecting line (H_2); (c) third connecting line (H_3); and (d) last connecting line (H_4).

Figure 5.3 shows the approximated function drawn by the solid line using *POC-NN*, classical *1-NN*, linear interpolation, and linear regression algorithm which is a *least square error* curve fitting when the degree of the polynomial is equal to 1. The proposed approximated function looks more similar to the original sine function than the *1-NN*'s approximated function, even though *POC-NN* uses fewer patterns to generate the approximated function. The linear interpolation draws straight lines connecting all data points that creates plots for projecting intermediate values along the lines.

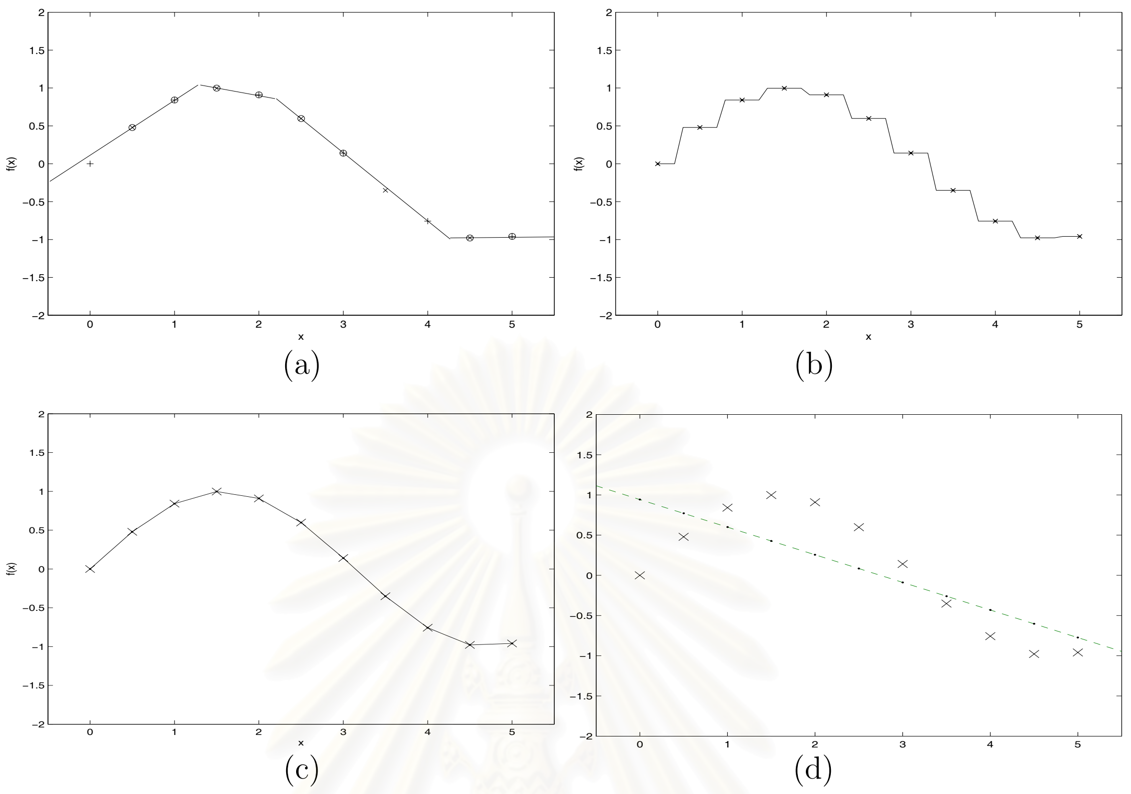


Figure 5.3: This example shows approximation function performed by: (a) POC-NN; (b) 1-NN; (c) Linear Interpolation; and (d) Linear Regression.

5.4 Reducing Complexity

In order to reduce the number of *POC-NN* prototypes, the concept of *acceptance interval* α is applied. The idea is analogous to an *acceptance interval* α for the classification problem. A separating hyperplane can be considered as a *tolerant line* or *soft margin.* All patterns lying within the soft margin are considered as lying on the hyperplane, and treated as correctly classified patterns, which can be ignored or discarded. A desired margin interval (α) is pre-specified by the user. By using the concept of soft margin ($\alpha > 0$), different small values of α can be set in order to control the complexity and the number of *POC-NN* prototypes used for compressing data. Generally, the prototype and accuracy rates decrease when the value of α increases.

CHAPTER VI

EXPERIMENTAL RESULTS FOR REGRESSION

This chapter presents the data sets and experimental results for regression. Besides, the performance in both accuracy rate and training time of *POC-NN* is evaluated and compared with *1-Nearest Neighbor (1-NN)* and *Linear Interpolation* techniques.

6.1 Data sets

All algorithms are tested and evaluated on a number of standard regression data sets of benchmarks, both artificial and real.

The classical standard one dimensional and two dimensional sinc$|x|$ function are defined as follows:

The one dimensional function

$$f(x) = sinc|x| = \frac{sin|x|}{|x|} \tag{6.1}$$

is considered on the basis of a sequence of 128 measurements without noise on the uniform lattice.

The two dimensional function

$$f(x,y) = sinc = \sqrt{x^2 + y^2} \tag{6.2}$$

is considered on the basis of a sequence of 1,681 measurements without noise on the uniform lattice.

The well known chaotic time series, the Mackey-Glass, is described by the delay-differential equation [23]:

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\Delta)}{1+x(t-\Delta)^{10}}, \tag{6.3}$$

with parameters $\Delta = 17$ and 30. These two time series are denoted by MG_{17} and MG_{30}.

The time series of the Lorenz differential equation [24, 34] are also considered. Moreover, the two real world data sets, the Titanium [25] and the Sunspot [26, 35] series from 1700-1799, are used in the experiments.

The properties of the experimental data sets are given in Table 6.1. The original sample sets are also separated into training and test sets (2:1 ratio) by using the same criteria as in the classification.

Table 6.1: Properties of the data sets used for regression problem

Data sets	No. of dimensions	No. of trg. patterns	No. of test patterns
1. Sinc 1d	1	85	43
2. Sinc 2d	2	1120	561
3. MG_{17}	1	133	67
4. MG_{30}	1	133	67
5. Lorenz	1	200	100
6. Titanium	1	32	17
7. Sunspot	1	66	34

The *Percent Root Mean Square Difference (PRD)* is an important performance index parameter of any regression or function approximation algorithm. *PRD* is defined by Eq. (6.4).

$$PRD = (\sqrt{\sum_{i=1}^{n}(\mathbf{x_i} - \tilde{\mathbf{x_i}})^2 / \sum_{i=1}^{n}(\mathbf{x_i})^2}) * 100, \tag{6.4}$$

where $\mathbf{x_i}$ and $\tilde{\mathbf{x_i}}$ are samples of the original and reconstructed/estimated data sequences, respectively. Small value of *PRD* shows the success of the algorithm.

6.2 Experimental Results

6.2.1 Accuracy of Approximation

To compare the results performed by *POC-NN* algorithm and classical *1-NN* algorithm, all original samples were trained on the training set of patterns by using $\alpha = 0$, and ran on the test set.

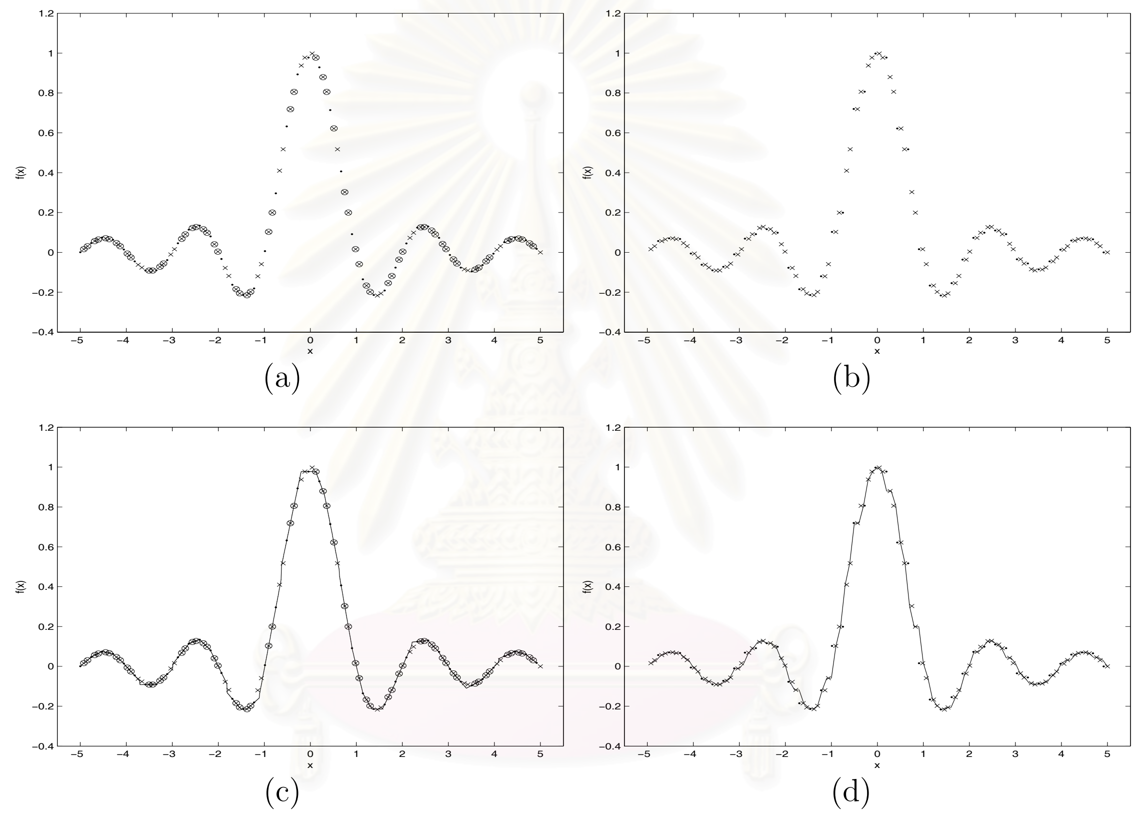
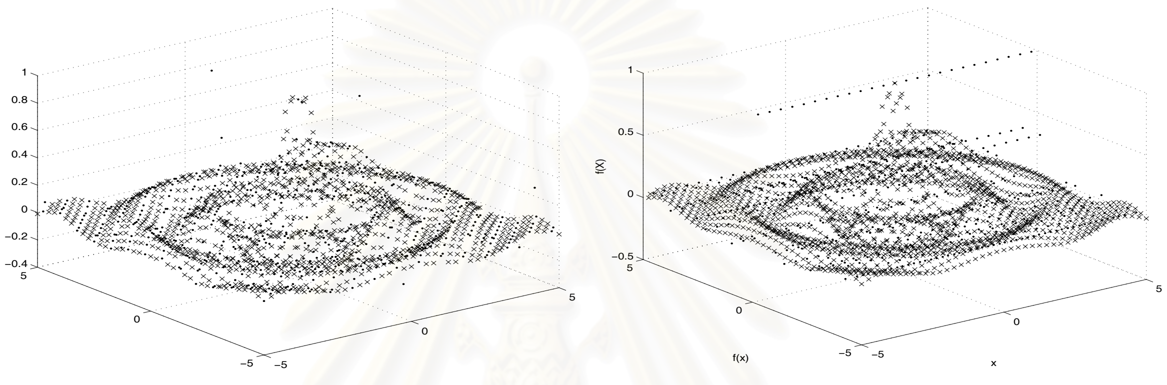


Figure 6.1: The one-dimensional sinc function: (a) PRD=2.30 by using POC-NN with PR=72.94(62/85); and (b) PRD=13.91 by using 1-NN with PR=100. The approximated function by using: (c) POC-NN; and (d) 1-NN.

The one-dimensional sinc function

Figure 6.1(a) shows *POC-NN* patterns enclosed in circle symbol "∘". 62 *POC-NN* patterns were obtained from 85 original training patterns depicted by symbols "×"

(PR=72.94) which brings the approximation ability of PRD to 2.30 after running on 43 test patterns. The estimated test patterns are depicted by the symbol "·". Figure 6.1(b) shows the results performed by the *1-NN* which obtained the PRD to be equal to 13.91. *POC-NN*'s results show better results than *1-NN* method in both, accuracy rate (PRD) and prototype rate (PR). Figure 6.1(c) and (d) show the approximated function in solid line from *POC-NN* and *1-NN* algorithms, respectively.

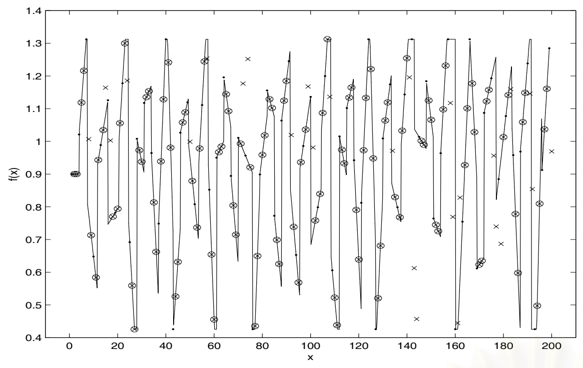


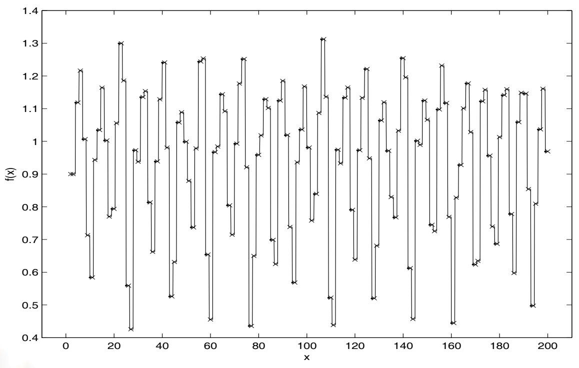
(a) (b)

Figure 6.2: The two-dimensional sinc: (a) PRD=83.68 by using POC-NN with PR=54.46(610/1120); (b) PRD=203.49 by using 1-NN with PR=100.

The two-dimensional sinc function

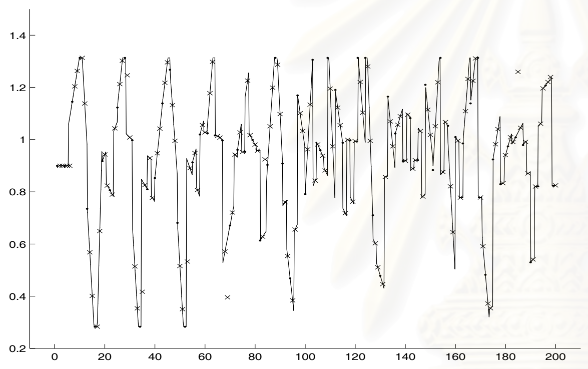
Figure 6.2(a) shows the estimated test patterns depicted by the symbol "·" using *POC-NN*. 610 *POC-NN* patterns were obtained from 1120 original training patterns (PR=54.46) which brings the approximation ability of PRD to 83.68 after running on 610 test patterns. Figure 6.2(b) shows the results performed by the *1-NN* which yielded the PRD to be equal to 203.49. *POC-NN*'s results are better than the *1-NN* method in both accuracy rate (PRD) and prototype rate (PR).

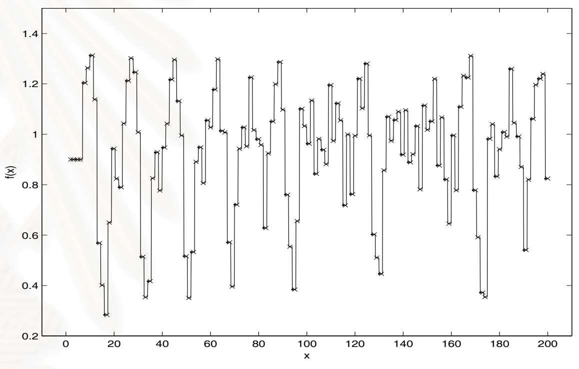




(a) (b)

Figure 6.3: The Mackey-Glass 17 data set: (a) PRD=16.56 by using POC-NN with PR=79.70(106/133); (b) PRD=18.14 by using 1-NN with PR=100.





(a) (b)

Figure 6.4: The Mackey-Glass 30 data set: (a) PRD=14.49 by using POC-NN with PR=93.23(124/133); (b) PRD=16.37 by using 1-NN with PR=100.

The Mackey-Glass data

Figure 6.3(a) shows *POC-NN* patterns enclosed in circle symbol "∘" and the estimated test patterns depicted by the symbol "·" using *POC-NN* algorithm. 106 *POC-NN* patterns were obtained from 133 original Mackey-Glass 17's training patterns depicted by the symbol "×" (PR=79.70) which brings the approximation ability of PRD to 16.56 after running on 67 test patterns. The estimated testing patterns are depicted by the symbol "·". Figure 6.3(b) shows the results performed by the *1-NN* which obtained the

PRD to be equal to 18.14. *POC-NN*'s results show better results than *1-NN* method in both, accuracy rate (PRD) and prototype rate (PR).

Figure 6.4 shows the results performed on the Mackey-Glass 30's data set. *POC-NN*'s results still show better results than *1-NN* method.

The Lorenz data

Figure 6.5(a) shows the *POC-NN* patterns enclosed in circle symbol "○" and shows the estimated testing patterns depicted by the symbol "·" using *POC-NN* algorithm. 146 *POC-NN* patterns were obtained from 200 original training patterns depicted by the symbol "×" (PR=73.70) which brings the approximation ability of PRD to 21.42 after running on 100 test patterns. The estimated testing patterns are depicted by the symbol "·". Figure 6.1(b) shows the results performed by *1-NN*, which obtained the PRD to be equal to 26.51. *POC-NN*'s results are better than *1-NN* method in both, accuracy rate (PRD) and prototype rate (PR).

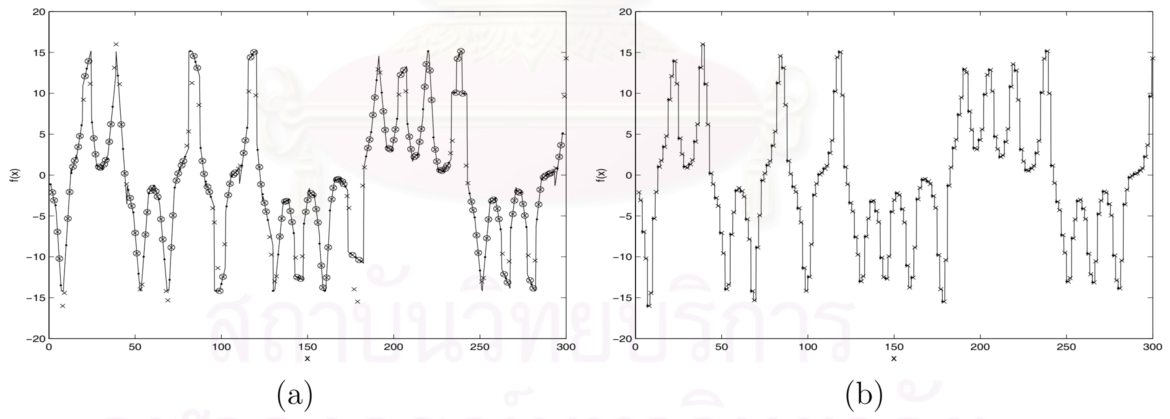
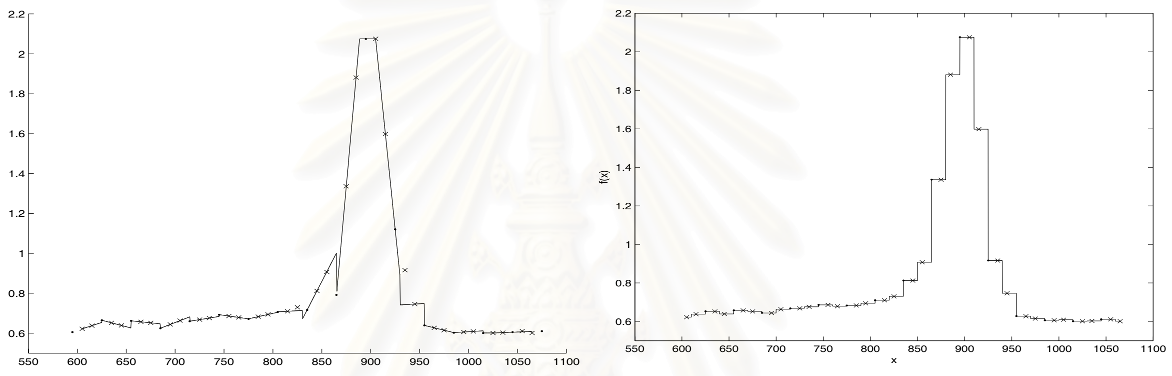


Figure 6.5: The Lorenz data set: (a) PRD=21.42 by using POC-NN with PR=73.70(146/200); (b) PRD=26.51 by using 1-NN with PR=100.

The Titanium data

Figure 6.6(a) shows the estimated testing patterns depicted by the symbol "·" using *POC-NN* algorithm. 28 *POC-NN* patterns were obtained from 32 original training patterns depicted by the symbol "×" (PR=87.50) which brings the approximation ability of PRD to 8.11 after running on 17 testing patterns. The estimated test patterns are depicted by the symbol "·". Figure 6.6(b) shows the results performed by *1-NN*, which obtained the PRD to be equal to 12.20.

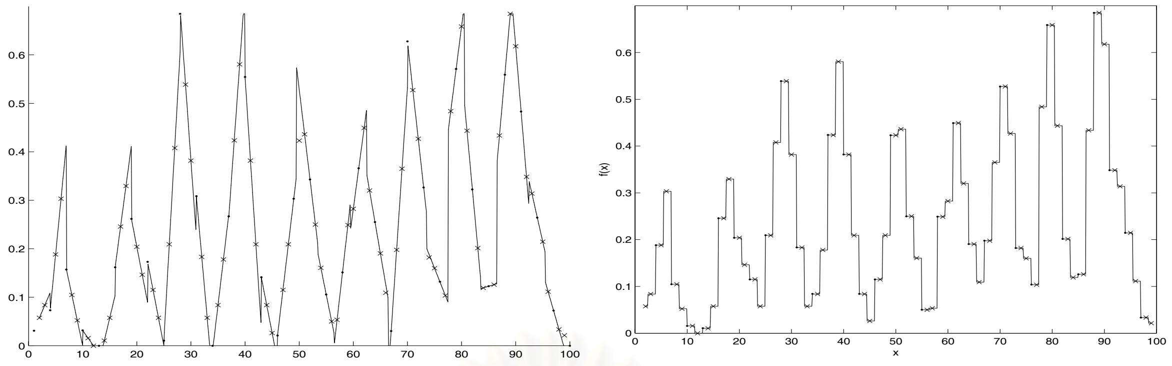


(a) (b)

Figure 6.6: The Titanium data set: (a) PRD=8.11 by using POC-NN with PR=87.50(28/32); (b) PRD=12.20 by using 1-NN with PR=100.

The Sunspot data

Figure 6.7(a) shows the estimated test patterns depicted by the symbol "·" using *POC-NN* algorithm. 62 *POC-NN* patterns were obtained from 66 original training patterns depicted by the symbol "×" (PR=93.93) which brings the approximation ability of PRD to 20.65 after running on 34 test patterns. The estimated test patterns are depicted by the symbol "·". Figure 6.7(b) shows the results performed by *1-NN*, which obtained the PRD to be equal to 26.65.



(a) (b)

Figure 6.7: The Sunspot data set: (a) PRD=20.65 by using POC-NN with PR=93.93(62/66); (b) PRD=26.65 by using 1-NN with PR=100.

All above comparisons of *POC-NN*, *1-NN*, and the Linear Interpolation techniques obtained on these data sets are summarized in Table 6.2. *POC-NN*'s results show better results than the *1-NN* method in both, accuracy rate (PRD) and prototype rate (PR) in all cases, and also better results than the *Linear* interpolation in case of the Sunspot data.

Table 6.2: The comparison results of *POC-NN*, *1-NN*, and *Linear Interpolation*

Data sets	POC-NN		NN		Linear	
	PRD %	PR %	PRD %	PR %	PRD %	PR %
1. Sinc 1d	2.31	72.94	13.91	100.00	1.31	100.00
2. Sinc 2d	83.69	54.46	203.49	100.00	4.1192	100.00
3. MG_{17}	16.56	79.70	18.14	100.00	9.25	100.00
4. MG_{30}	14.49	93.23	16.37	100.00	9.54	100.00
5. Lorenz	21.42	73.00	26.51	100.00	6.00	100.00
6. Titanium	8.11	87.50	12.20	100.00	6.00	100.00
7. Sunspot	20.65	93.94	26.65	100.00	22.20	100.00

Table 6.3: The time comparisons of *POC-NN*, *1-NN*, and *Linear Interpolation* in seconds

Data sets	Time in seconds								
	POC-NN			1-NN			Linear		
	Train	Test	Total	Train	Test	Total	Train	Test	Total
1. Sinc 1d	0.49	0.05	0.54	-	0.05	0.05	-	0.06	0.06
2. Sinc 2d	2.96	0.20	3.16	-	0.22	0.22	-	0.24	0.24
3. MG_{17}	0.65	0.06	0.71	-	0.07	0.07	-	0.09	0.09
4. MG_{30}	0.71	0.08	0.79	-	0.09	0.09	-	0.11	0.11
5. Lorenz	0.54	0.09	0.63	-	0.11	0.11	-	0.12	0.12
6. Titanium	0.22	0.06	0.28	-	0.07	0.07	-	0.11	0.11
7. Sunspot	0.34	0.06	0.40	-	0.06	0.06	-	0.12	0.12

6.2.2 Computational Time

The training and testing time required for the results shown in Table 6.2 are also compared. The time comparisons are summarized in Table 6.3. The *POC-NN* algorithm requires longer training time, but yields the best testing time for all data sets.

6.3 Evaluate the POC-NN Algorithm for Regression

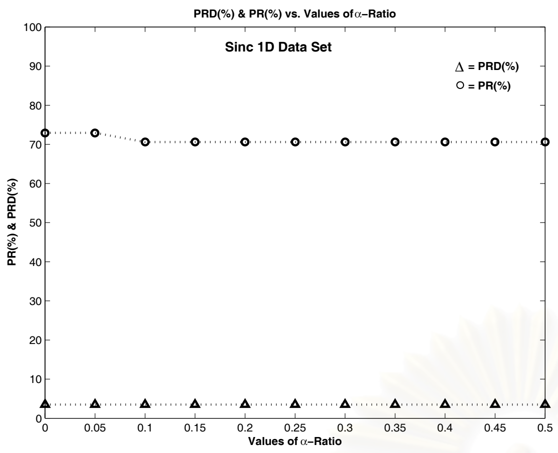
To estimate the difference between accuracy, a three-fold cross-validation and five-fold cross-validation were also conducted to arrive at the average cross-validation estimate of PRD. The performance comparisons among *POC-NN* and other methods are summarized in Table 6.4. The symbols “***”, “**” and “*” indicate 99 %, 95 % and 90 % confidence interval for estimating the difference between accuracy of *POC-NN* and *1-NN* using one-tailed paired t-test [33], respectively.

Table 6.4: The comparison results of *POC-NN*, *1-NN*, and *Linear Interpolation*

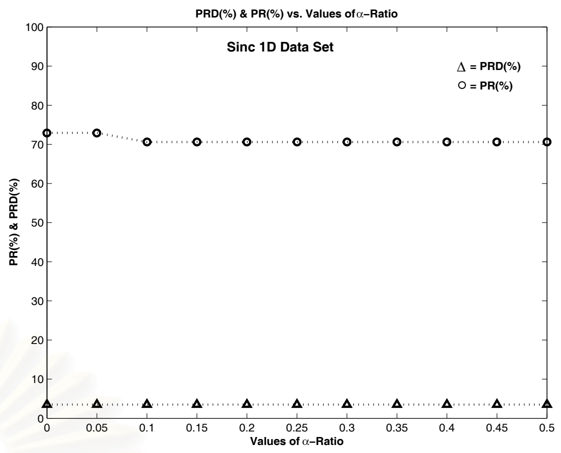
K-Fold	Data sets	POC-NN		1-NN		Linear	
		PRD %	PR %	PRD %	PR %	PRD %	PR %
K=3	1. Sinc 1d	2.46	75.78	13.96	100.00	1.31	100.00
	2. Sinc 2d	151.38	45.33	203.49	100.00	4.12	100.00
	3. MG_{17}	16.85	81.01	19.15	100.00	9.78	100.00
	4. MG_{30}	18.12	96.00	17.51	100.00	9.92	100.00
	5. Lorenz	26.80*	72.67	27.96	100.00	7.59	100.00
	6. Titanium	25.48**	67.42	15.89	100.00	5.65	100.00
	7. Sunspot	37.13***	73.99	26.65	100.00	22.20	100.00
K=5	1. Sinc 1d	4.42	76.96	13.91	100.00	1.30	100.00
	2. Sinc 2d	83.68	54.46	203.49	100.00	4.11	100.00
	3. MG_{17}	21.74	73.25	19.14	100.00	9.73	100.00
	4. MG_{30}	18.61	74.00	17.52	100.00	9.92	100.00
	5. Lorenz	23.19*	72.33	28.02	100.00	7.52	100.00
	6. Titanium	11.96**	68.36	15.59	100.00	5.53	100.00
	7. Sunspot	33.23**	67.50	38.41	100.00	18.78	100.00

6.4 Reducing Complexity for Regression

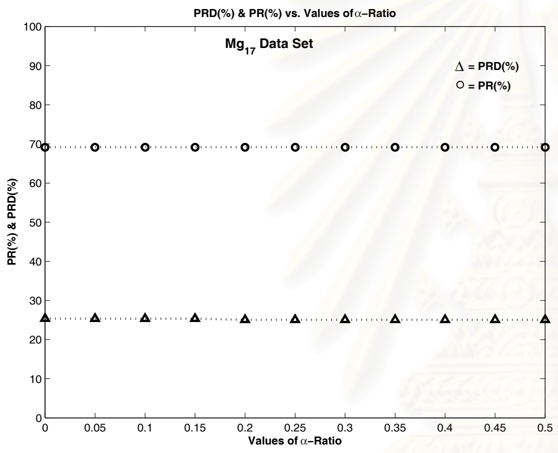
In order to reduce the number of prototypes, the concept of *acceptance interval* (α) is also considered. The value of α can be regulated in order to control the number of *POC-NN* patterns. For regression problem, the data can be normalized before applying *acceptance interval* (α). Figure 6.8 shows the number of *POC-NN* prototypes and prototype rate in relation to α-ratio. The α-ratio can also be employed to define the prototype and accuracy rates, and the approximation accuracy of a function. Obviously, this can reduce the number of *POC-NN* prototypes, which lead to data compression that is the process of eliminating redundancies in a given data set. Redundancy commonly exists whenever neighboring data samples are statistically dependent or there is a representative prototype used by redundant samples without significant error. Since *POC-NN* considers all patterns lying within *acceptance interval* as redundant data samples.



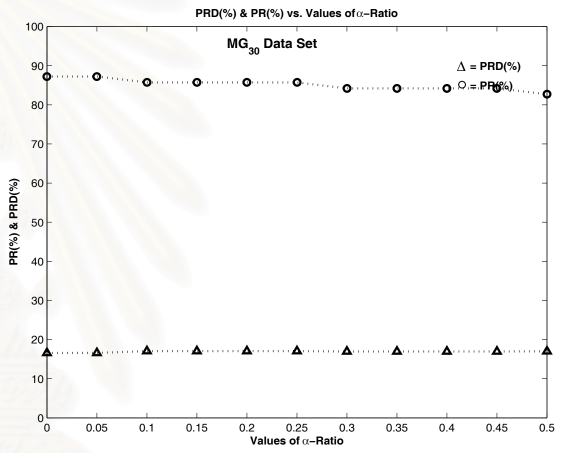
(a) Sinc 1d



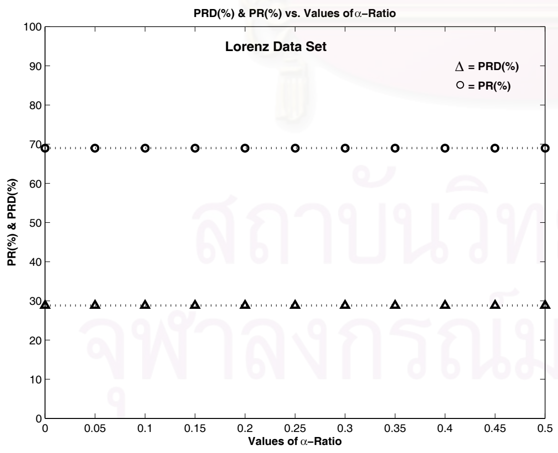
(b) Sinc 2d



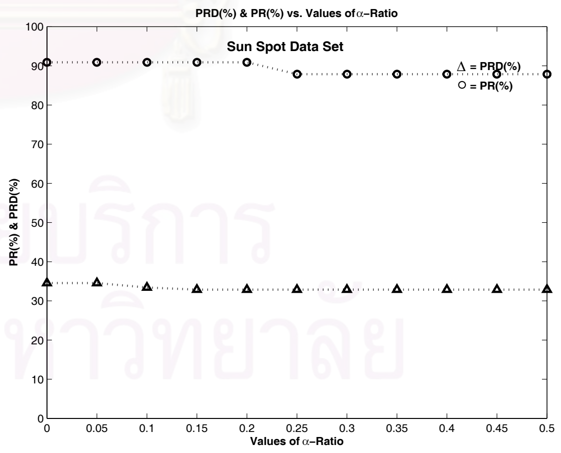
(c) MG_{17}



(d) MG_{30}



(e) Lorenz



(f) Sunspot

Figure 6.8: Prototype Rate (PR %) and Accuracy Rare (AR %) as a function of α-ratio.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

A new *POC-NN* method based on divide-and-conquer approach to prototype selection has been proposed. For a finite number of training data, *POC-NN* algorithm always converges with all patterns correctly separated into the regions of classified classes. The time complexity required to execute *POC-NN* algorithm is in the order of $O(dn^2p)$, where d, n, and p is the number of dimension, training patterns, and *POC-NN* patterns, respectively. *POC-NN* method is fast as well as simple. Moreover, it can overcome some undesirable properties of order dependence and sensitivity of noisy data. The proposed method can be used to solve classification and regression problems. The prototype and accuracy rates can also be regulated by a user-defined parameter, i.e., the acceptance interval ratio (α_r). The prototype and accuracy rates so obtained have been compared with *CNN*, *Tomek*, *GKA*, and *NN* with all the training patterns. For most classification results, the proposed method showed better performances in both prototype and accuracy rates than those from others. For all the above mentioned classification results, the proposed method showed the best training time. For the regression results, the proposed method showed better performances in both prototype and accuracy rates than the results from the classical *1-NN* method, and also better than the *Linear* interpolation in case of the Sunspot data set.

7.2 Future Work

Increasing the accuracy rate and decreasing the prototype rate of *POC-NN* are still open issues, which need a thorough investigation. The Nearest Neighbor Rule relies on a metric or *distance* function between patterns. So far the Euclidean metric in d dimensions as the basis of decision criterion have been assumed, and alternative measures of distance with *POC-NN* algorithm can be investigated in the experiment and result inference.

POC-NN method is based on the binary classifier and the prototype selection method. One can replace the *'one-against-one' (1-v-1)* scheme with the other kinds of multi-class scheme to implement a new multi-class classifier. Moreover, the set of prototypes can be rendered adaptive and used as prototype replacement method.

The relationship between α_r and the average distance among data, as well as its variance, was not studied here. The value of α_r should be locally adaptive, according to the distribution nature of the data. This issue is essential and worth furthers investigation.

When the *NN* rule is carried out in high-dimensional feature space, the nearest neighbors of a pattern can be very far away, causing bias and degrading of performance. This is commonly referred to as the *curse of dimensionality*. The reduction of dimension should be investigated in order to apply the proposed algorithm on real world applications in the areas of classification and regression problems.

REFERENCES

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.

[2] P. Hart, "An asymptotic analysis of the nearest-neighbor decision rule," *Stanford Electron. Lab., Stanford, Calif., Tech. Rep.*, vol. SEL-66-016, pp. 1828–2, 1966.

[3] R. Duda, P. Hart, and S. D.G., *Pattern Classification.* John Wiley and Sons, 2001.

[4] L. Kuncheva and J. Bezdek, "Nearest prototype classification: Clustering, genetic algorithm, or random search?" *IEEE Transactions on Systems Man and Cybernetics, Part C:*, vol. 28(1), pp. 160–164, 1998.

[5] P. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14(3), pp. 515–516, 1968.

[6] G. Gates, "The reduced nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 18, pp. 431–433, 1972.

[7] G. Ritter, H. Wooddruff, S. Lowry, and T. Isenhour, "An algorithm for a selective nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 23, pp. 1179–1184, 1975.

[8] I. Tomek, "Two modifications of cnn," *IEEE Transactions on Systems Man and Cybernetics*, vol. Cybernetics 2, pp. 769–772, 1976.

[9] K. Chidananda Gowda and G. Krishna, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood," *IEEE Transactions on Information Theory*, vol. 25(4), pp. 488–490, 1979.

[10] L. Devroye, L. Gyorfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition.* New York: Springer-Verlag, 1996.

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

[12] D. Wilson, “Asymtotic properties of nearest neighbor rules using edited data,” *IEEE Transactions on Systems Man and Cybernetics*, vol. 2, pp. 408–420, 1972.

[13] J. Sanchez, R. Barandela, A. Marques, A. R., and J. Badenas, “Analysis of new techniques to obtain quality training sets,” *Pattern Recognition Letters*, vol. 24, pp. 1015–1022, 2003.

[14] F. Ferri, J. Albert, and E. Vidal, “Considerations about sample-size sensitivity of a family of edited nearest neighbor rules,” *IEEE Transactions on Systems Man and Cybernetics, Part B*, vol. 29(5), pp. 667–672, 1999.

[15] B. Dasarathy, J. Sanchez, and S. Townsend, “Nearest neighbor editing and condensing tools-synergy exploitation,” *Pattern Analysis and Application*, vol. 3, pp. 19–30, 2000.

[16] C. Chang, “Finding prototypes for nearest neighbor classifiers,” *IEEE Transactions on Computer*, vol. 23(11), pp. 1179–1184, 1974.

[17] G. Toussaint, “Proximity graphs for nearest neighbor decision rules: Recent progress,” in *Proceedings of 34th Symposium on Computing and Statistics*, 2002.

[18] V. Devi and M. Murty, “An incremental prototype set building technique,” *Pattern Recognition*, vol. 35, pp. 505–513, 2002.

[19] E. Alpaydin, “Voting over multiple condensed nearest neighbors,” *Artificial Intelligence Review*, vol. 11, pp. 115–132, 1997.

[20] P. Murphy and D. Aha, “*UCI* repository of machine learning databases [http://www.ics.uci.edu/mlearn/mlrepository.html],” 1994.

[21] USPS, "Public data set of the university of tuebingen [fpt://ftp.kyb.tuebingen.mpg.de/pub/bs/data/]," 1994.

[22] D. Michie, D. Spiegelhalter, and C. Taylor, "Machine learning, neural and statistical classification," in *[ftp://ftp.ncc.up.pt/pub/statlog]*, 1994.

[23] M. Mackey and J. Glass, "Oscillation and chaos in physiological control systems," *Science*, p. 197:287, 1997.

[24] E. Lorenz, "Deterministic non-periodic flow," *Atoms. Sci.*, vol. 26, p. 639, 1963.

[25] P. Dierckx, *Curve and Surface Fitting with Splines.* Monographs on Numerical Analysis. Oxford: Clarendon Press,, 1993.

[26] G. Yule, "On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers," *Phil. Trans. Roy. Soc. London*, vol. A226, pp. 267–298, 1927.

[27] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20(3), pp. 273–297, 1995.

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

[29] D. Roobaert, "Directsvm: A simple support vector machine perceptron," in *IEEE International Workshop on Neural Networks for Signal Processing*, 2000.

[30] G. Toussaint, "A counterexample to tomek's consistency theorem for a condensed nearest neighbor," *Pattern Recognition Letters*, vol. 15, pp. 797–801, 1994.

[31] A. Hayter, *Probability and Statistics for Engineers and Scientists.* Duxbury, 1995.

[32] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* Springer-Verlag, 2001.

[33] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[34] C. Sparrow, *The Lorenz equations.* New York: Springer-Verlag, 1982.

[35] E. Wan, “Combining fossil and sunspot data: Committee predictions,” in *International Conference On Neural Networks (ICNN97)*, 1997.

APPENDICES

Appendix I

Publications

1. International Journal

[I] T. Raicharoen and C. Lursinsap, “A Divide-and-Conquer Approach to The Pairwise Opposite Class-Nearest Neighbor (POC-NN) Algorithm”, in *Journal of Pattern Recognition Letter.*

2. International Conference

[I] T. Raicharoen and C. Lursinsap, “Critical Support Vector Machine Without Kernel Function”, in *The Proceeding of the* 9^{th} *International Conference on Neural Information Processing (ICONIP’ 02)*, November 2002.

[II] T. Raicharoen and C. Lursinsap, “Critical Support Vector Machine For Function Approximation Problems”, in *The Proceeding of the* 3^{rd} *International Joint Conference on Intelligence Technologies and the* 3^{rd} *Vietnam-Japan Symposium on Fuzzy Systems and Applications (InTech/VJFuzzy-2002:)*, December 2002.

[III] T. Raicharoen, P.Sanguanphogai and C. Lursinsap, “Application of Critical Support Vector Machine to Time Series Prediction”, in *the IEEE International Symposium on Circuits and System (ISCAS2003)*, May 2003.

[IV] T. Raicharoen, K.Sookhanaphibarn, F.Lin and C. Lursinsap, “Emergency

Flood Forecasting Methodologies", in *the Workshop on Flood Prevention and Control on Yangtze River (FOCYR2004)*, January 2004.

[V] K.Sookhanaphibarn, T. Raicharoen and C. Lursinsap, "A supervised Neural Network Approach to Invariant Image Recognition", in *the 8. International Conference on Control, Automation, Robotics and Vision (ICARCV2004)*, December 2004.

2. National Conference

[I] T. Raicharoen and C. Lursinsap, "Kernelless Linear Transformation Machine: A supervised Learning Algorithm for Pattern Classification and Functional Approximation Problems", in *The Proceeding of the* 5th *RGJ-Ph.D. Congress V*, April 2004.

Appendix II

A mathematical derivation of the statistical result (one-tailed proportions of the normal test)

Let AR_1 and AR_2 denote respectively the proportions of accuracy rates of the *POC-NN* and *CNN* algorithm, for instance, USPS data set, AR_1=0.9342 and AR_2=0.9158. Furthermore, let n denote the sample size, which is 2007 in this case, and begin by calculating the average proportion of accuracy rate, $AR = \frac{0.9342+0.9158}{2} = 0.9250$.

Now the standard deviation is determined, σ_s, of the difference of proportions, P_1-P_2:

$$\sigma_{AR_1-AR_2} = \sqrt{\frac{2AR(1-AR)}{n}} = \sqrt{\frac{2 \times 0.9250(1-0.9250)}{2007}} \approx 0.0083, \tag{B.1}$$

and normalize to standardized variable Z:

$$Z = \frac{AR_1 - AR_2}{\sigma_{AR_1-AR_2}} \approx 2.21. \tag{B.2}$$

Finally, a table of one-tailed proportions of the normal curve [31] is used, which gives the level of significance given a standardized variable, Z. The level of significance of the hypothesis is found to be approximately 2.21, which yields a 98.64% confidence level.

Appendix III

A mathematical derivation of the statistical result (one-tailed paired t-test)

For estimating the difference between accuracy rates of two learning methods, let $\bar{Y}$ denote the mean difference in accuracy rates from all disjoint subsets. $\bar{Y}$ is the estimate of the difference between the two learning algorithms [33]. The approximate $N\%$ confidence interval for the difference estimation using $\bar{Y}$ is given by

$$(\bar{Y} - t_{N,K-1}S_{\bar{Y}}, \bar{Y} + t_{N,K-1}S_{\bar{Y}}), \tag{C.1}$$

where $\bar{Y}$ is the sample mean defined as

$$\bar{Y} = \frac{1}{K}\sum_{i=1}^{K} Y_i, \tag{C.2}$$

K is the number of fold cross-validation, and Y_i is the difference in accuracy rate between two learning methods from the i^{th} subset, and $S_{\bar{Y}}$ is the estimated standard deviation of the sample mean defined as

$$S_{\bar{Y}} = \sqrt{\frac{1}{K(K-1)}\sum_{i=1}^{K}(Y_i - \bar{Y})^2}. \tag{C.3}$$

BIOGRAPHY

Group Captain Thanapant Raicharoen

Personal Details:

Date of Birth:	March 24, 1963
Place of Birth:	Chachoengsao, Thailand

Academic Education:

Oct'01-May'05	Ph.D., Program in Computer Science, Department of Mathematics, Chulalongkorn University, Thailand.
Apr'05-May'05	Ph.D. Visiting Student, Artificial Intelligent Group, Department of Computer Science, the University of Maryland, DC, USA.
Jan'87-Dec'87	Research Assistant in the SIEMENS AG Company, and in the University of Federal Armed Forces, Munich, Germany.
Jun'86-Dec'87	M.Sc. Program in Computer Science, Faculty of Computer Science, The University of Federal Armed Forces, Munich, Germany.
Oct'83-Aug'86	B.Sc. Program in Computer Science, Faculty of Computer Science, The University of Federal Armed Forces, Munich, Germany.

Professional Education:

Apr'00-Sep'00	Communication and Information Officer Course, U.S. Air Force, Mississippi, USA.
Oct'93-Sep'94	Air Command and Staff College, RTAF, Bangkok, Thailand.
Jun'90-Sep'90	Squadron Officer School, RTAF, Bangkok, Thailand.

Honour Awards:

2004	Best Oral Presentation Award, Ph.D. RGJ-Congress V, Thailand.
2000	Distinguished Graduate, Communication and Information Officer Course, U.S. Air Force, Mississippi, USA.
2000	Excellent Military Researcher Award, Ministry of Defense (MOD), and Excellent Military Researcher Award, Royal Thai Air Force (RTAF).
1998	Distinguished Officer Award, Royal Thai Air Force (RTAF), and Distinguished Officer Award, The RTAF Office of Information Technology.
1989	Distinguished Officer Award, Royal Thai Air Force (RTAF), and Distinguished Officer Award, The RTAF Science and Weapon Systems Development Center.
1981	Distinguished Graduate, The RTAF Academy, the 1st Year, Thailand.

Scholarships:

Oct'01-May'05	Scholarship of the Royal Golden Jubilee (RGJ) Ph.D. Program.
Apr'00-Sep'00	Scholarship of the International Military Education and Training (IMET) USA, the US Air Force Information and Communication Officer Course, Mississippi, USA.
Jan'88-Dec'88	Scholarship of the SIEMENS AG Company and University of Federal Armed Forces in Munich for the Research Assistant, Germany.
Jan'82-Dec'87	Scholarship of the Royal Thai Air Force for B.Sc.and M.Sc. in Germany.