# การทำนายโดยโครงข่ายประสาทเทียมของอันตรกิริยาระหว่างคู่โปรตีนบนพื้นฐานของสัมประสิทธิ์สหสัมพันธ์เชิงเคมีกายภาพและการบูตสแทรปสำหรับการสร้างข้อมูลเทียม

นางสาวพุทธิพร ธนธรรมเมธี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

5073857023

# NEURAL PREDICTION OF PROTEIN-PROTEIN INTERACTIONS BASED ON PHYSICOCHEMICAL CORRELATION COEFFICIENTS AND BOOTSTRAPPING FOR ARTIFICIAL DATA GENERATION

Miss Putthiporn Thanathamathee

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2011

| | |
|---|---|
| Thesis Title | NEURAL PREDICTION OF PROTEIN-PROTEIN INTERACTIONS BASED ON PHYSICOCHEMICAL CORRELATION COEFFICIENTS AND BOOTSTRAPPING FOR ARTIFICIAL DATA GENERATION |
| By | Miss Putthiporn Thanathamathee |
| 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 Doctoral Degree

นางสาวพุทธิพร ธนธรรมเมธี : การทำนายโดยโครงข่ายประสาทเทียมของอันตรกิริยาระหว่างคู่โปรตีนบนพื้นฐานของสัมประสิทธิ์สหสัมพันธ์เชิงเคมีกายภาพและการบูตสแทรปสำหรับการสร้างข้อมูลเทียม. (NEURAL PREDICTION OF PROTEIN-PROTEIN INTERACTIONS BASED ON PHYSICOCHEMICAL CORRELATION COEFFICIENTS AND BOOTSTRAPPING FOR ARTIFICIAL DATA GENERATION) อ. ที่ปรึกษาวิทยานิพนธ์หลัก: ศ. ดร. ชิดชนก เหลือสินทรัพย์, 62 หน้า.

แม้ว่าการใช้เฉพาะสายลำดับโปรตีน อาจจะมีเพียงพอสำหรับการทำนายอันตรกิริยาคู่โปรตีนโดยใช้โครงข่ายประสาทเทียม แต่มีปัญหาที่ต้องพิจารณา คือ การสกัดคุณลักษณะของคู่โปรตีนให้อยู่ในรูปของตัวมูลตัวเลข ปัญหาถัดมา คือ ต้องทำการสงวนรักษาคุณสมบัติของคู่โปรตีนนั้น หลังจากที่ทำให้เป็นเว็กเตอร์ตัวเลขที่มีขนาดเท่ากันในแต่ละคู่โปรตีน วิทยานิพนธ์นี้จึงได้นำเสนอวิธีการทำนายอันตรกิริยาระหว่างคู่โปรตีนจากสายลำดับโปรตีน ที่ใช้เฉพาะขอบข้อมูลเทียมที่ได้สร้างขึ้นจากข้อมูลคู่โปรตีน รวมทั้งหลักการบูตส่งเสริมเพื่อเพิ่มประสิทธิภาพในการทำนายของโครงข่ายประสาทเทียม โดยการสกัดคุณลักษณะของคู่โปรตีนบนพื้นฐานสัมประสิทธิ์สหสัมพันธ์เชิงเคมีกายภาพ ค่าทางสถิติของโครงสร้างทุติยภูมิ และคุณสมบัติที่สำคัญของโปรตีน หลังจากนั้นจะได้คุณลักษณะของคู่โปรตีนที่อยู่ในรูปของข้อมูลเว็กเตอร์ตัวเลขที่มีขนาดเท่ากัน ซึ่งข้อมูลตัวเลขเหล่านี้จะถูกนำมาหาขอบข้อมูล และนำเฉพาะขอบนี้ไปสร้างข้อมูลขอบเทียมโดยใช้หลักการบูตสแทรป และขั้นตอนสุดท้าย ข้อมูลขอบเทียมเท่านั้นจะถูกนำไปใช้ทำนายการเกิดอันตรกิริยาคู่โปรตีนโดยใช้หลักการบูตส่งเสริมโครงข่ายประสาทเทียม ผลการทดลองกับข้อมูลยีสต์ ปรากฏว่าวิธีการที่ได้นำเสนอนั้นสามารถทำนายได้ถูกต้องมากกว่าวิธีการอื่นที่นำมาเปรียบเทียบ มากไปกว่านั้นวิธีที่นำเสนอได้ใช้เฉพาะข้อมูลขอบเทียมมาเป็นข้อมูลการสอน ซึ่งจำนวนข้อมูลสอนจะน้อยกว่าวิธีอื่นด้วย นอกจากนี้ยังประเมินประสิทธิภาพการทำนายด้วยข้อมูลทดสอบโปรตีนข้ามสายพันธุ์ ผลแสดงให้เห็นว่าวิธีการที่นำเสนอนั้นมีประสิทธิภาพการทำนายดีกว่าวิธีการอื่นที่นำมาเปรียบเทียบ

| | |
|---|---|
| ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ | ลายมือชื่อนิสิต |
| สาขาวิชา วิทยาการคอมพิวเตอร์ | ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก |
| ปีการศึกษา 2554 | |

# # 5073857023 : MAJOR COMPUTER SCIENCE

KEYWORDS : PREDICTION OF PROTEIN-PROTEIN INTERACTIONS / CORRELATION COEFFICIENTS / BOOTSTRAPPING TECHNIQUE / ARTIFICIAL DATA GENERATION / NEURAL NETWORK / ADABOOST TECHNIQUE

PUTTHIPORN THANATHAMATHEE : NEURAL PREDICTION OF PROTEIN-PROTEIN INTERACTIONS BASED ON PHYSICOCHEMICAL CORRELATION COEFFICIENTS AND BOOTSTRAPPING FOR ARTIFICIAL DATA GENERATION. ADVISOR : PROF. CHIDCHANOK LURSINSAP, Ph.D. 62 pp.

Although using only protein sequences might be sufficient for predicting, there are major problems in the prediction of protein-protein interactions by classifying technique such as supervised neural network. The first one is extracting the feature of protein pair sequences to form a feature sequence. The second problem is conserving the information when equalizing the lengths of feature sequences before classifying into interacting and non-interacting classes. This dissertation proposed a method to predict protein-protein interactions from amino acid sequences using only artificial boundary data generation and boosting procedures to improve the prediction accuracies. The feature extraction is based on the correlation coefficients of physicochemical properties, the statistical means and standard deviations of secondary structures and protein properties. The important data which lie into the boundary of each subcluster were only used to generate the artificial boundary data by bootstrap resampling technique. Finally, the only artificial boundary data of both positive and negative protein pairs were predicted by boosting method based on neural network classifier. The empirical study has shown that our proposed method yielded better prediction accuracy than the sequence-based methods when performed on *Yeast Saccharomyces Cerevisiae* data set. Moreover, the number of feature and the number of training data were less than others. The prediction models were also evaluated by cross-species test data sets. The result showed that the proposed method also capable to predict with the good performance on cross-species data.

| | |
|---|---|
| Department : Mathematics and Computer Science | Student's Signature |
| Field of Study : Computer Science | Advisor's Signature |
| Academic Year : 2011 | |

# Acknowledgements

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First, I would like to thank my advisor, Professor Dr. Chidchanok Lursinsap for his valuable guidance, assistance and motivation in this dissertation. My work and studied could not be complete without him. It is honor to work with him. He is an exemplary professor and advisor.

I would like to thank Asst. Prof. Dr. Wiphada Wettayaprasit for her valuable guidance, assistance and motivation. She always supports and encourages me in all situation.

I also would like to thank my thesis committee members, Assoc. Prof. Dr. Peraphon Sophatsathit, Dr. Suphakant Phimoltares, Asst. Prof. Dr. Suchart Chanama for their valuable advice, comments, and guidance in this dissertation.

Also, it is my pleasure to acknowledge the financial support from the Office of Higher Education Commission, Ministry of Education, Thailand through Faculty of Informatics (Computer Science), Walailak University.

I would like to thank AVIC members for helping all additional discussions that help improve in this dissertation.

Last but not least, I am deeply grateful to my parents who have given me the opportunity of an education from the best institutions, support and encouragement throughout my life.

Finally, this dissertation is dedicated to my brother for his support and encouragement throughout his life.

# Contents

# List of Tables

# List of Figures

# CHAPTER I

# INTRODUCTION

## 1.1 Problem and Motivation

Nowadays, there are many computational methods have been proposed for predicting protein-protein interactions. Meanwhile, with the increasing of the number of protein sequences, several sequence-based methods have shown that the information of amino acid sequences alone may be sufficient for predicting protein-protein interactions.

Although using only protein sequences may be sufficient for predicting, there are three major problems in the prediction of protein-protein interactions by classifying technique such as supervised neural network. The first one is extracting the feature of protein pair sequences to form a feature sequence. The second problem is conserving the information when equalizing the lengths of feature sequences before classifying into interacting and non-interacting classes. The third is generating artificial data to improve the performance of prediction. To predict protein-protein interactions by boosting neural network, both positive and negative interactions are required for training. Unlike positive interactions, negative protein pairs are not available. The negative protein pairs are generated from the assumption that protein pairs with different sub-cellular localizations do not interact or there is no explicit evidence of an interaction. This assumption generates a much larger number of negative protein pairs than positive protein pairs. Thus, this problem concerns with imbalanced learning data. Standard machine learning algorithms fail to classify imbalanced data which produce high predictive accuracy over negative protein pairs but low predictive accuracy on positive protein pairs because they are learned from imbalanced training data and the output hypothesis are fitted to the majority data. Consequently, test data belonging to the positive protein pairs are misclassified more often than belonging to the negative protein pairs.

From these problems, the new method was proposed to predict protein-protein interactions from amino acid sequences using only artificial boundary data generation and boosting procedures to improve the prediction accuracies of both positive and negative protein pairs. Our feature extraction is based on the correlation coefficients of physicochemical properties, the statistical means and standard deviations of secondary structures and protein properties, i.e.alpha-helix, beta-sheet, beta-turn, coil, parallel beta strand, amino acid composition, hydrophobicity, average area buried, and polarity. To form the new training features by finding the new distribution direction of these features. Then Self-Organizing Map (SOM) was applied to find subclusters in the new training features. The important data which lie into the boundary of each subcluster were only used to generate the artificial boundary data. The artificial boundary data were generated by using bootstrap resampling technique. Finally, the only artificial boundary data of both positive and negative protein pairs were predicted by boosting method based on neural network classifier. The empirical study has shown that our proposed method yielded better prediction accuracy than the sequence-based methods [1], and [2] when performed on *Yeast Saccharomyces Cerevisiae* data set. Moreover, the number of feature and the number of training data are less than others. We also evaluated the prediction models by cross-species data as the test sets. The result showed that our proposed method also capable to predict with the good performance on cross-species data.

## 1.2 Objective

The objectives of this dissertation are the following :

1. To predict protein-protein interactions from amino acid sequences using adaboost neural network.

2. To improve the prediction power of classifier against protein pairs using only artificial boundary data.

## 1.3 Scope and Limitations

1. This proposed method was performed on *Yeast Saccharomyces Cerevisiae* from core subset of database of interacting proteins (DIP).

2. The results of *Yeast Saccharomyces Cerevisiae* data set before and after generating artificial boundary data based on adaboost neural network were compared with the method of predicting protein pairs using support vector machine combined with auto covariance [1] and local descriptor [2].

3. In another evaluation after generating artificial boundary data, we tested the ability of our proposed method for predicting protein-protein interactions in one species using the interactions from different species. Our proposed model was trained on the *Yeast Saccharomyces Cerevisiae*

and we chose the other five species as our cross-species test data sets. The five species data sets are *Drosophila Melanogaster*, *Caenorhabditis elegans*, *Eshcherichia coli*, *Homo sapiens*, and *Mus musculus*.

4. The performance on cross-species data sets were compared with the method of predicting protein pairs using support vector machine combined with auto covariance [1] and local descriptor [2].

## 1.4 Contributions

The main contributions of this dissertation are a new feature extraction from only protein sequences for predicting protein-protein interactions based on the correlation coefficients of physiochemical properties combined with statistical means and standard deviations of protein secondary structures and protein properties, i.e.alpha-helix, beta-sheet, beta-turn, coil, parallel beta strand, amino acid composition, hydrophobicity, average area buried, and polarity. Moreover, the important protein data which lay at the boundary were only used. Then, the artificial boundary data were generated by using bootstrap resampling technique. Finally, the boosting method based on neural network classifier was used to predict artificial boundary protein pairs into interacting class and non-interacting class.

## 1.5 Methodology

1. Review and study related works and documents of predicting protein-protein interactions.
2. Propose a new method for prediction of protein-protein interactions.
3. Experiment with protein pairs data and compare with other techniques.
4. Review and study related works and documents of learning from imbalanced data for improving the performance of prediction.
5. Propose a new method for learning from imbalanced data.
6. Experiment with benchmark data sets for handling imbalanced data and compare with the other techniques.
7. integrate a proposed method of learning from imbalanced data with a proposed method of predicting protein-protein interactions to improve the performance of predicting protein pairs.

8. Experiment with protein pairs data and compute with other techniques.

9. Analyse experimental results and conclude the results.

## 1.6 Dissertation Organization

This thesis is organized as follows. Chapter I provides a brief introduction of the scope and methodology. Chapter II explains the concept of Neural Network, Adaptive Boosting, Self-Organizing Map, Principal Component Analysis and Bootstrap resampling technique. Moreover, briefly reviews related works. Chapter III presents the proposed method of predicting protein-protein interactions. Chapter IV summarizes the experimental results with discussion. The conclusion is in Chapter V.

# CHAPTER II

# RELATED WORKS AND BACKGROUND

## 2.1 Related Works

Proteins carry out the majority of the biological processes in cells and have a large variety of functions which can be categories into many kinds such as antibody, hormone, enzyme, signal protein, and so on [3]. Different kinds of proteins must interact with one another to perform various biological functions. The information of protein-protein interactions help to improve knowledge of the functions, understand biological processes in a cell, and potentially make the discovery of novel drug targets.

Previously, although the protein interaction pairs were detected by co-immunoprecipitation or chromatography, the determination of the protein-protein interactions cannot follow the growth of the newly found proteins and many protein pairs. Hence, various experimental methods have been developed for the large scaled protein-protein interactions analysis such as yeast two-hybrid systems [4] , mass spectrometry [5], protein chip [6] and so on. But these methods are costly and time consuming. Many computational methods have been proposed for predicting protein-protein interactions which are based on *genomic context*, *biological context*, and *structural context* of proteins [7].

*Genomic context approaches*. Complete genome sequencing provided a wealth of genomic information. Therefore, there are many methods used for predicting protein-protein interaction such as protein phylogenetic profiles [8], conservation of gene neighborhood [9], and gene fusion events [10].

*Biological context approaches*. Many of high-throughput methods for investigating the biological context of genes, such as gene expression have been proposed. It has been indicated that many interacting proteins are co-expressed according to microarray analyses [11, 12].

*Structural context approaches*. This approach can determine not only protein pair interaction but also the physical characteristics of the interaction sites at the protein interfaces [7]. Analysis of hydrophobicity of amino acids can be used to predict interaction site [13, 14]. Another study [15] proposed the residue composition can be used to analyze six types of protein-protein interfaces. In [16], they used structure matching technique to predict protein-protein interactions. Moreover, the information of binding sites and binding motifs [17–19] for improving prediction have been considered.

Recently, several sequence-based methods have shown that the information of amino acid sequences alone may be sufficient for predicting protein-protein interactions [20–22]. Moreover, the re-occurring of short polypeptide sequences [23] can also identify novel protein-protein interaction. However, many methods of sequence-based to predict protein-protein interactions have been proposed, these methods achieved the highest accuracy only 80%. In addition, some statistical method for extracting the protein sequence features [24] and using support vector machine (SVM) combined with auto covariance [1] have been proposed to improve the accuracy of prediction. This method [1] predicted protein-protein interaction based on seven physicochemecal and auto covariance with *Yeast Saccharomyces Cerevisiae*. First, amino acid sequences were transformed into numerical values by representing physicochemical properties as vectors with each amino acid represented by normalized valued of seven physicochemical properties. Then Auto Covariance was used to transform numerical vectors into fixed length. After each protein sequence was represented as a vector of auto covariance variable, a protein-protein interaction was characterized by concatenating the feature vectors of two proteins. Finally, predicting protein-protein interactions by support vector machine. Moverover, local descriptor (LD) [2] used an alignment-free approach was applied to underlie amino acid groups. For each local region, three local descriptors, composition (C), transition (T) and distribution (D), were calculated. C stands for the composition of each amino acid group along a local region. T represents the percentage frequency with which amino acid in one group is followed by amino acid in another group. D characterizes the distribution pattern along the entire region by measuring the location of the first, 25, 50, 75 and 100% of residues of a given group. The calculation of descriptors generates 63 attributes in each local region (7 for C, 21 for T and 35 for D). The descriptors for all local regions were combined, and formed the features vector which were predicted by SVM.

In many techniques of predicting protein-protein interactions by machine learning methods [1–3], they failed to classify protein pairs which produced high predictive accuracy over non-interacting protein pairs (negative class) but low predictive accuracy on interaction protein pairs (positive class). Thus, the sampling method was used to improve the performance of classification. The objective is to provide a balanced distribution from *oversampling* and/or *undersampling* techniques to improve overall classification [25]. In regards to artificial sampling, the artificial positive class oversampling technique (SMOTE) [26] has used in various applications. The concept of SMOTE is to produce synthetic data in minority class by selecting some of the nearest minority neighbors of a positive data and generate artificial positive data along with the lines between the positive data and its nearest positive neighbors. In [27, 28], they proposed adaptive sampling methods to generate artificial data. The main idea of Borderline-SMOTE technique [27] is to find out the borderline positive samples. Then, artificial samples were generated along the line between the borderline samples and their nearest neighbors of the same class. The key idea of the ADASYN algorithm [28] was to use a density distribution as a criterion to automatically decide the number of artificial data that need to be generated for each positive data by adaptively changing the weights of different positive data. In [29], they combined boosting and artificial data using SMOTE to improve the prediction of the positive class. Moreover, DataBoost-IM approach [30] was proposed to generate artificial data. The hard examples of both positive and negative classes were identified during each of the iterations of boosting algorithm to generated artificial training data. These artificial examples were added to the original training set and are used for farther training to improve the classification.

In this dissertation, we proposed a feasible method to predict protein-protein interactions from amino acid sequences with boosting neural network. Our feature extraction is based on the correlation coefficients of physicochemical properties and the statistical means, standard deviations of secondary structures and protein properties, i.e.alpha-helix, beta-sheet, beta-turn, coil, parallel beta strand, amino acid composition, hydrophobicity, average area buried, and polarity. Then to find the new distribution direction of these features to form the training data. After that subclusters of each class were discovered by SOM technique. The important training data which lay into the boundary of each subcluster were only used to generated artificial boundary data. These artificial boundary data are generated by using bootstrap resampling technique to improve the performance of classification not only interaction class bus also non-interacting class. Finally, the only artificial boundary data of both classes are predicted by boosting method based on neural network classifier.

## 2.2 Background

### 2.2.1 Neural Network

Artificial neural network is a model of the brain which transforms inputs into outputs to the best of performance. The basic model of artificial neuron is based on the functionality of the biological neuron. The biological neural network is composed of groups of its structure connected or functionally associated neurons. These are cell body or soma, axon, dendrites, and synapse. Figure 2.1 shows the biological neural network. Cell body or soma is the heart of the cell which contains nucleus. There are many dendrites in each neuron that receive signals from other neurons. Generally, a neuron has only one axon which expands from a part of the cell body. The main function of axon is to proceed electronic signals. At the end of axon is split into several branches which the synapse is adhered. Synapse is the area of contact between a neuron to the other neurons and gives the strength of the connection.

Artificial neural network, synapses are weight which represented by a number. Figure 2.2 depicts the artificial neural network. In this figure, a neuron with $n$ dimensional input vector $\{x_1, x_2, x_3, ..., x_n\}$ and $b$ bias value are multiplied by weigh input vector $\{w_{11}, w_{12}, w_{13}, ..., w_{1n}\}$. Then these values are fed into the summing junction. After that the input $v$ are calculated by Equation 2.1. Finally, the output $y$ of the neuron is the outcome from using some activation function $\varphi(\bullet)$ on value $v$ which is calculated as Equation 2.2.

$$v = \Sigma_{i=1}^{n} x_i w_i \tag{2.1}$$

$$y = \varphi(v) \tag{2.2}$$

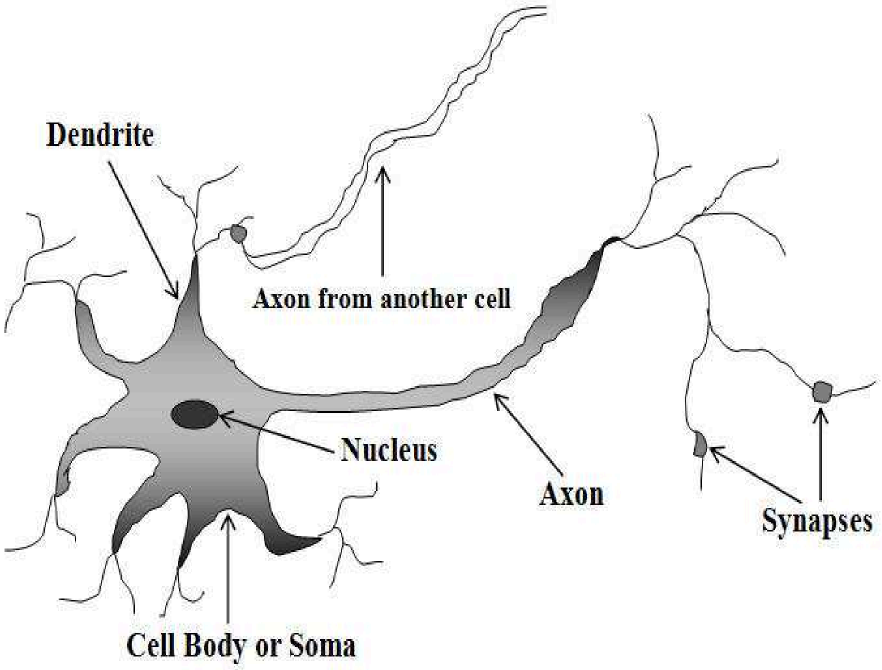


Figure 2.1 The example of biological neural network.

There are many activation functions are used to control the actual output. In general, the three activation functions are commonly used as shown below :

1. Threshold function

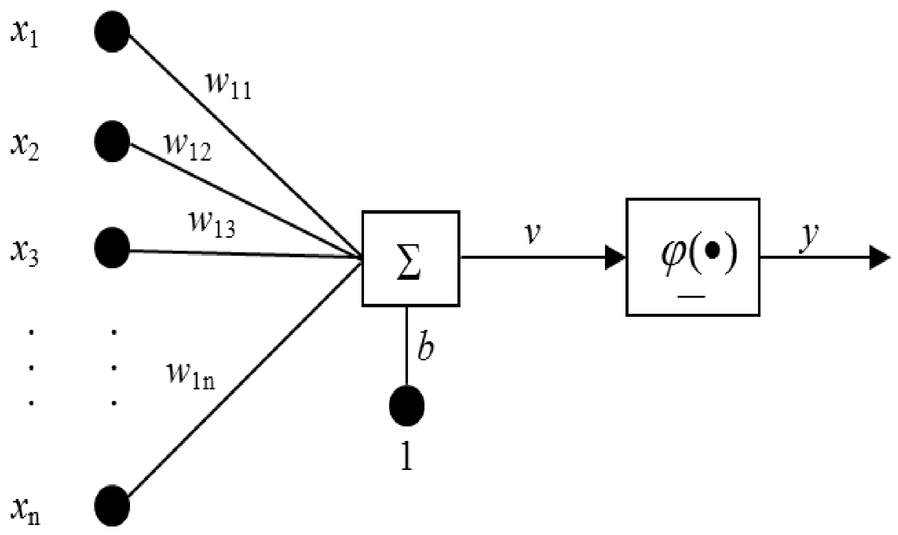


Figure 2.2 The example of artificial neural network.

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

2. Sigmoid function

$$\varphi(v) = \frac{1}{1+exp(-av)} \qquad , a \text{ is slope}$$

3. Signum function

$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases}$$

**A Layer of Neurons**

1. *Single layer neural network.*

   Figure 2.3 depicts the single layer of neurons. This network has *n* elements of input vector and *s* neurons in the layer. The input vector is multiplied by weight vector. Then these inputs are summarized with bias value in each *i*th neuron. Thus, the *i*th neuron has a summer to form its output $y_i$. Finally, the neuron output layer $\{y_1, y_2, ..., y_s\}$ are calculated by the activation function to controls the actual outputs.

2. *Multiple layer neural network.*

   This multiple layer network in Figure 2.4 has *n* elements of input vector and *s* neurons in the layer. The input vector is fed into layer 1 to produce output $\{y^1_1, y^1_2, \ldots, y^1_s\}$. Then these outputs are the inputs of the layer 2 that are fed to produce output $\{y^2_1, y^2_2, \ldots, y^2_s\}$. After that, these outputs are become to inputs of the layer 3. Finally, $\{y^3_1, y^3_2, \ldots, y^3_s\}$. are the final outputs of this network. Each layer in the multiple layer network has the different rules. A layer that produces the final outputs in called the *output layer*. The other layers are called *hidden layers*. Fig 2.4 has the one output layer (layer 3) and two hidden layers (layer 1 and layer 2).

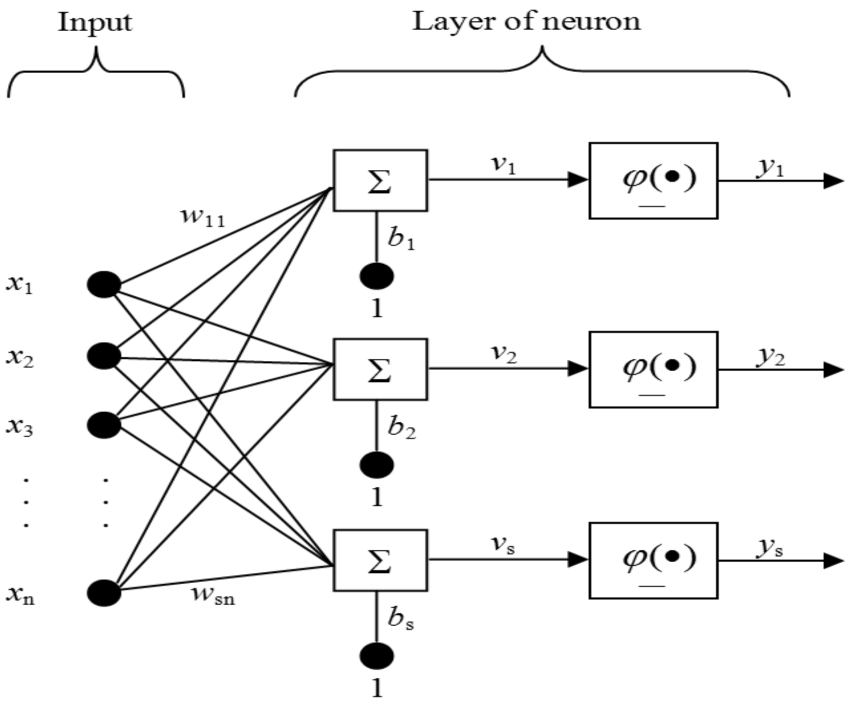


Figure 2.3 The example of single layer of neural network.

**Learning Rules**

The learning rule is applied to train the network by modifying the weights and biases to perform the better learning. Generally, there are two types of the learning rules that are *supervised learning* and *unsupervised learning*.

1. *Supervised learning.*

   The training set is provided to learn in the neural network. This training set is composed of $\mathbf{x_1 t_1}, \mathbf{x_2 t_2}, \ldots, \mathbf{x_Q t_Q}$ where $\mathbf{x}$ is an input to the neural network, $\mathbf{t}$ is a target output, and $\mathbf{Q}$ is a number of training examples. After that, the network outputs are compared with the target

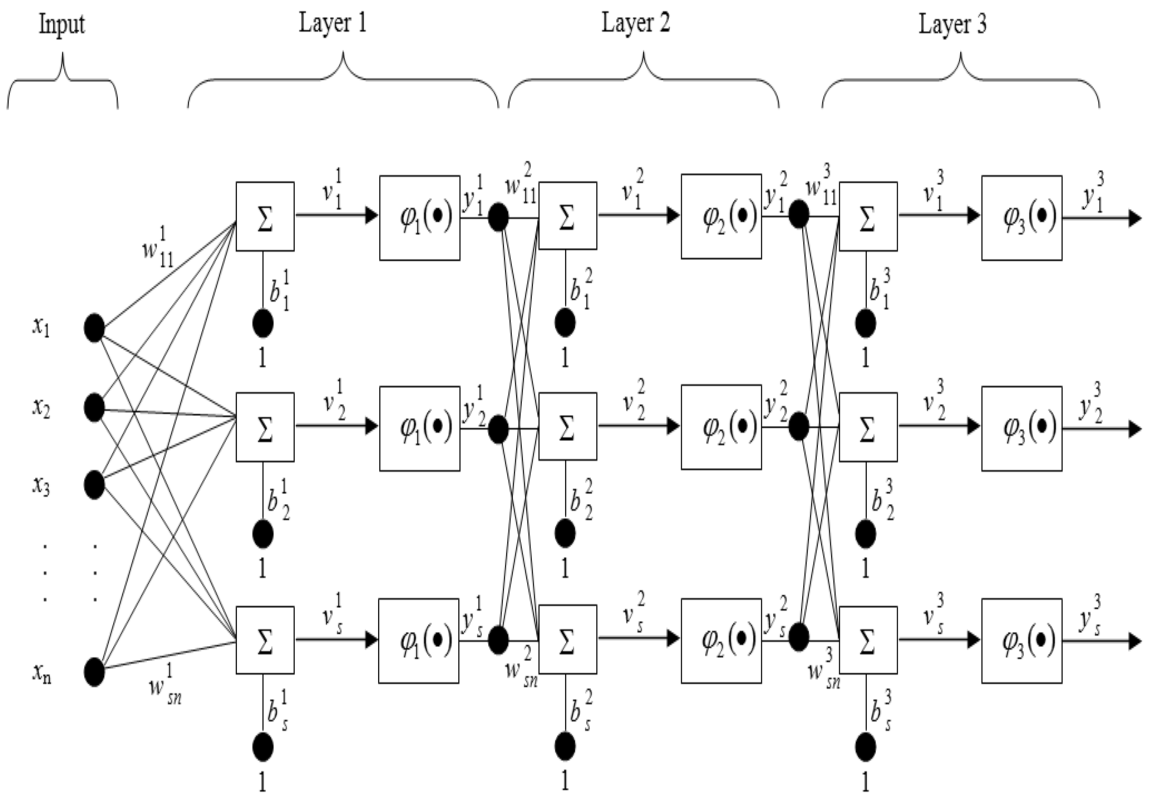


Figure 2.4 The example of multiple layer of neural network.

outputs. If the network outputs do not close to the target outputs, then the weights and biases are adjusted to modify the network outputs closer to the target outputs.

2. *Unsupervised learning.*

   The only inputs are used in this learning. The weight and biases are modified based on the input patterns. In the final step, the learning will categorize input patterns into a finite number of clusters.

**Neural network architecture**

***1. Single layer perceptron neural network.*** The single layer perceptron consists of $s$ perceptron neurons. Figure 2.5 shows only one perceptron neuron which uses threshold function as its activation function to produces the network output $y$. The perceptron neuron produces network output $y = 1$ if the inputs $v$ are fed into the activation function is equal to or greater than 1, otherwise it produces $y = 0$.

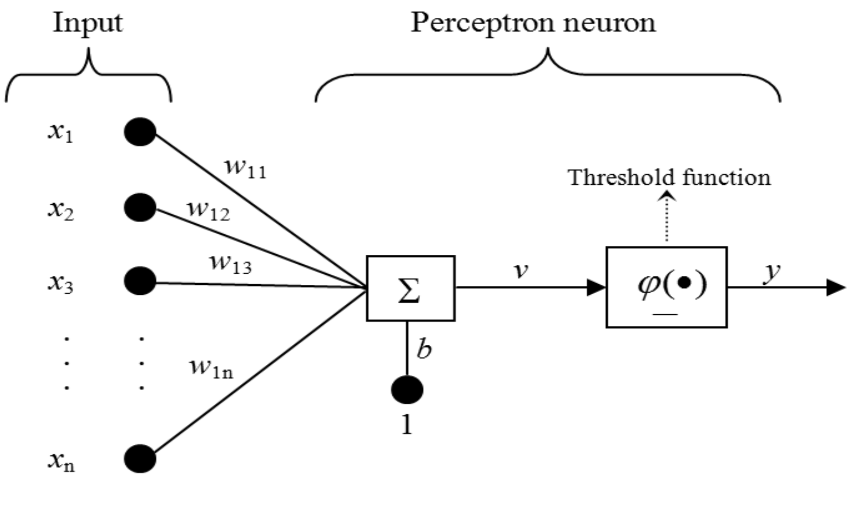


Figure 2.5 The example of single layer perceptron neural network.

*Perceptron learning algorithm*

The perceptrons neural network are trained on training examples which consist of pairs of input and output target.

$$\mathbf{x_1 t_1}, \mathbf{x_2 t_2}, \ldots, \mathbf{x_Q t_Q}$$

where $\mathbf{x}$ is an input to the neural network, $\mathbf{t}$ is a target output, and $\mathbf{Q}$ is a number of training examples. In each time of learning, the perceptron learning rule try to adapt perceptron's weights and biases for reducing the error $\mathbf{e}$ which is calculated by the difference between the target output and an actual output in Equation 2.3.

$$\mathbf{e} = \mathbf{t} - \mathbf{y} \tag{2.3}$$

Moreover, there are three conditions when input vector $\mathbf{x}$ is presented into the network to produce the actual output $\mathbf{y}$ [31].

**Case 1.** If an input vector is presented and actual output from the network is correct, then the weight is not changed. $(\mathbf{a} = \mathbf{t}$ and $\mathbf{e} = \mathbf{0})$.

**Case 2.** If an actual output is 0 but the target output is 1 so the actual output from the network is misclassified, then the input vector $\mathbf{x}$ is added to the weight vector $\mathbf{w}$. This case increases the chance of the input vector is classified as $\mathbf{y} = \mathbf{1}$ in the future. $(\mathbf{y} = \mathbf{0}, \mathbf{t} = \mathbf{1}$ and $\mathbf{e} = \mathbf{1})$.

**Case 3.** If an actual output is 1 but the target output is 0 so the actual output from the network is misclassified, then the input vector $\mathbf{x}$ is subtracted from the weight vector $\mathbf{w}$. This case increases the chance of the input vector is classified as $\mathbf{y} = \mathbf{0}$ in the future. $(\mathbf{y} = \mathbf{1}, \mathbf{t} = \mathbf{0}$ and $\mathbf{e} = -\mathbf{1})$.

Thus, the perceptron learning algorithm can be summarized as Equation 2.4. The training examples are presented iteratively to the network and the weights are updated until a maximum number of iteration is reached.

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{e}\mathbf{x}^{\mathbf{T}} \tag{2.4}$$

***2. Multilayer perceptron neural network.*** Classification problem is almost a non-linearly separable. So neural network with multiple layers and sigmoid activation function are commonly applied to solve this problem. Figure 2.6 depicts a multilayer perceptron neural network which consists of input layer, hidden layer, and output layer. The training examples are fed into a layer-by-layer on the network. Then, backpropagation learning algorithm is used to train the network by changing the weights of network based on the delta rule $(\delta)$. The purpose of adjustment weights is to reduce the error between the target output and the actual output. There are two phases in backpropagation learning algorithm :

*Feed-forward.* The input vector $\mathbf{x}$ is fed into the network and produced a set of actual output from the network.

*Backpropagation.* An error $\mathbf{e}$ is run backward through the network. So, the network's weights are adjusted to minimize the error $\mathbf{e}$ of the network.

The backpropagation learning algorithm is summarized as :

**Case 1.** Adjusting the network's weights between output layer and hidden layer. (*at level k and level j in Figure 2.6.*)

$$\mathbf{w}_{kj}^{new} = \mathbf{w}_{kj}^{old} + \eta\,(\delta_k y_k) \tag{2.5}$$

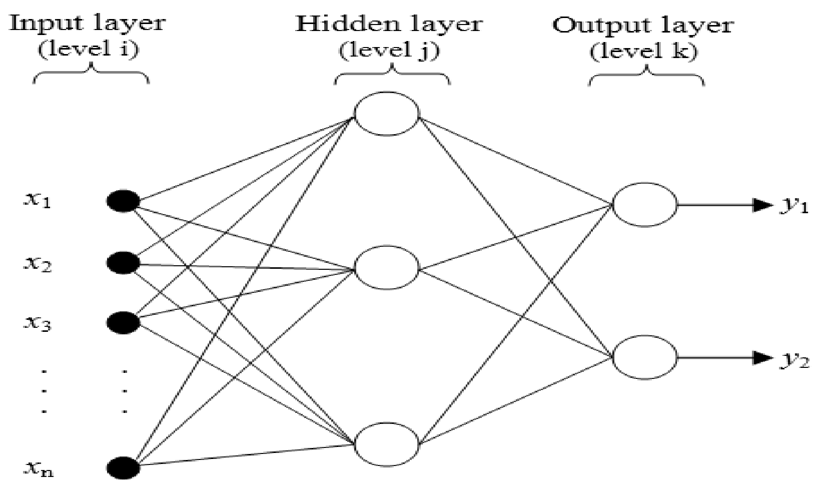


Figure 2.6 The example of multilayer perceptron neural network.

where $\eta$ is learning rate parameter and $\delta_k$ is calculate from Equation 2.6

$$\delta_k = \mathbf{e_k}\varphi'(v_k) \tag{2.6}$$

**Case 2.** Adjusting the network's weights between the hidden level and the previous hidden or input layer. (*at level j and level i in Figure 2.6.*)

$$\mathbf{w}_{ji}^{new} = \mathbf{w}_{ji}^{old} + \eta(\delta_j y_j) \tag{2.7}$$

where $\delta_j$ is calculated from Equation 2.8

$$\delta_j = \varphi'(v_j)\Sigma_k \delta_k \mathbf{w}_{kj}^{old} \tag{2.8}$$

The training examples are presented iteratively to the network and the weights are updated until a maximum number of iteration is reached.

### 2.2.2 Adaptive Boosting

Boosting is an ensemble method which focuses on hard examples that are difficult to classify. The basic idea of boosting is to construct multiple classifiers and the outputs from these classifiers are combined by weighted voting in the final prediction model [30, 32]. Moreover, the variance error is associated with overfitting model. Thus, for improving the performance of the overfitting, the combination of classifiers in boosting is used to reduce the variance error [33]. In each step, the training data are re-weighted. The product of training is a set of *easy examples* with low weights and a set of *hard examples* with high weights. In each iteration, boosting tries to construct the better classifiers that are able to predict hard examples correctly.

Adaptive Boosting or called AdaBoost is used as boosting method. AdaBoost was introduced by Freund and Schapire [34]. AdaBoost is widely used because it is robust, fast, and flexible to combine with any method for finding hard examples. In this dissertation, AdaBoost was used with feed-forward of backpropagation neural network. The algorithm of AdaBoost [32, 35] is given in *Algorithm 1*. The algorithm proceeds as follows. The initial weights probability $D$ of all training examples are set to be $1/m$ where $m$ is the number of training examples. In each iteration, the weight probability $D$ of easy examples that are correctly classified by the current trained classifier are unchanging. On the other hand, the weights probability $D$ of hard examples are increased by multiplying with a factor. Then, the weights are renormalized by dividing the normalized constant. After reaching the last iteration or the error of hypothesis in some iteration of hard examples is higher than 0.5, the final step is to consider only the correctly output examples in each iteration and maximize the sum of the weight probabilities on these examples.

In implementation AdaBoost with neural network classifier, there are three possible conditions to train neural network in each iteration $T$. The first condition is to train the neural network with a new training examples by resampling with replacement once from original training examples according to the weight probability. The larger weight probability has more chances to be selected in the new training examples. The second is to use a different training examples at each epoch by resampling with replacement after each trained epoch, and the third is to combine weight probability with the cost function of neural network. In this paper, the first condition is used to implement.

**Algorithm 1.** AdaBoost Algorithm

```
Input: - m examples $(x_1,y_1)$, ..., $(x_m,y_m)$, where $x_i \in X$,
         $y_i \in Y = \{-1,+1\}$
       - trained classifier (feed-forward of backpropagation neural network).
       - $T$ specifies the number of iterations.
       - set $t = 1$, $err^t$ = -1, where $err^t$ is an error of hypothesis in
         $t$ iteration.
       - initial weight probability of each training example, $D_i^t = 1/m$,
         where $1 \leq i \leq m$.
1. Do while $t \leq T$ and $err_t < 0.5$
2.    Train input examples by feed-forward of backpropagation neural network
```

with weight probability $D_i^t$.

3. Get a hypothesis $h^t : X \rightarrow Y$.

4. Calculate the error of hypothesis $h^t$.

   $err^t = \Sigma_{i:h^t(x_i) \neq y_i} D_i^t$

5. Set $\beta^t = err^t/(1 - err^t)$.

6. Update weight probability of input examples.

   $D_i^{t+1} = \frac{D_i^t}{Z^t} \times \begin{cases} \beta^t, \; if\, h^t(x_i) = y_i \\ 1, \; otherwise \end{cases}$

   where $Z^t$ is a normalization constant.

7. $t = t + 1$.

8. **End Do while**

**Output:** the final hypothesis $h(x) = argmax_{y \in Y} \Sigma_{t=1,h^t(x)=y}^{T} log \frac{1}{\beta_t}$.

### 2.2.3 Self-Organizing Map

Self-Organizing Map or called SOM is an unsupervised neural network. For each learning iteration, the only winning neuron's weight and its neighboring neuron's weight are adjusted. Figure 2.7 depicts an example of SOM architecture. The input layer represents the input vector with $n$ dimensions which is denoted by $\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$. The output layer is usually mapped into two dimensional output space. Each output neuron is connected to the input layer by weights vector $\{\mathbf{w_{j1}}, \mathbf{w_{j2}}, ..., \mathbf{w_{jn}}\}$, where $1 \leq j \leq m$. At each iteration of training, SOM finds the best matching winning neuron by using Euclidean distance. The minimum distance is calculated between each input vector and every weight vector on the output map. Let $\mathbf{c}$ with weight vector $\mathbf{w_c}$ is the best matching winning neuron which is calculated as Equation 2.9.

$$\|\mathbf{x} - \mathbf{w_c}\| = \mathbf{min_j}(\|\mathbf{x} - \mathbf{w_j}\|) \tag{2.9}$$

where $\|\mathbf{x} - \mathbf{w_j}\| = \sqrt{\Sigma_{j=1}^{m}(\mathbf{x} - \mathbf{w_j})}$. After the winning neuron $\mathbf{c}$ is selected, the weight vector of the winning neuron is adjusted by

$$\mathbf{w_j^{new}} = \mathbf{w_j^{old}} + \eta(\mathbf{x} - \mathbf{w_j^{old}}) \tag{2.10}$$

where $\eta$ is the learning rate parameter. Moreover, the weight vector of neighboring neurons of winning neuron $\mathbf{c}$ are also adjusted. In the example of Figure 2.7, the neighboring neurons of winning neuron that represented by the grey circles are also updated.

The SOM algorithm is summarized as :

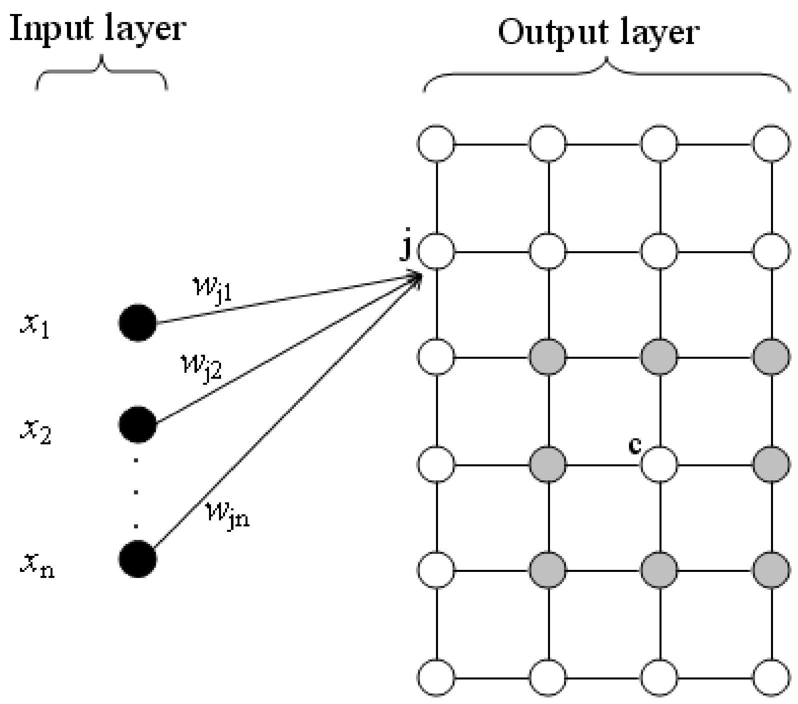


Figure 2.7 The example of SOM architecture.

1. **Initialization :** Choose the random value of the initial weight vectors on the output map.

2. **Sampling :** Draw the input vector **x** and present into the network.

3. **Similarity Matching :** Find the best matching winning neuron **c** by using Euclidean distance as shown in Equation 2.9.

4. **Updating :** Adjust the weight vector of the winning neurons **c** and the weight vector of the neighboring neurons by Equation 2.10.

5. **Continuation :** Repeat steps 2-4 until the weight vectors do not change (convergence).

### 2.2.4 Principal Component Analysis

The main idea of Principal Component Analysis (PCA) is to reduce the dimensionality of the data set without much loss of information. The other advantage of PCA is to discover or identify new meaningful underlying variables. The PCA steps are given below :

1. To calculate mean of data $\mu_X$ and $\mu_Y$, where *n* is the number of data.

$$\mu_X = \frac{1}{n}\Sigma_{i=1}^{n} X_i$$
$$\mu_Y = \frac{1}{n}\Sigma_{i=1}^{n} Y_i$$

2. To calculate covariance matrix.

$$cov(X,Y) = \frac{\Sigma_{i=1}^{n}(X_i - \mu_X) \times (Y_i - \mu_Y)}{n-1}$$

3. To Calculate the eigenvalues $(\lambda)$ and eigenvectors $(v)$ of the covariance matrix $cov(X,Y)$. It is the way of finding the distribution directions in all features of the data in terms of eigenvectors of data covariance matrix and their variances in term of eigenvalues of these eigenvectors. Generally, the eigenvector with the *highest eigenvalue* is the *principal component* of the data set.

4. To select components from ranked eigenvalue by descending order and form a feature vector. This step gives the component in order significance. If data set have *m* dimensions, so calculating *m* eigenvectors and *m* eigenvalues. Then choose only first *p* eigenvectors to form only *p* dimensions of data set.

5. To derive the new data set.

$$NewData = FeatureVector^T \times Data$$

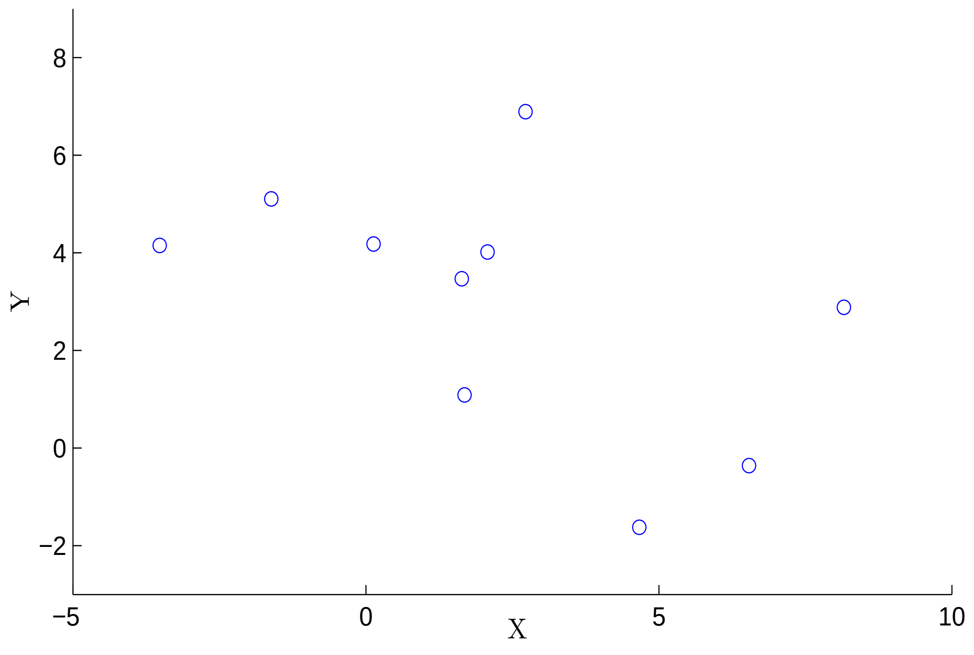
In Figure 2.8 shows a plot of original data before using PCA on the top and a plot of new data after using PCA on the bottom.

### 2.2.5 Bootstrap resampling technique

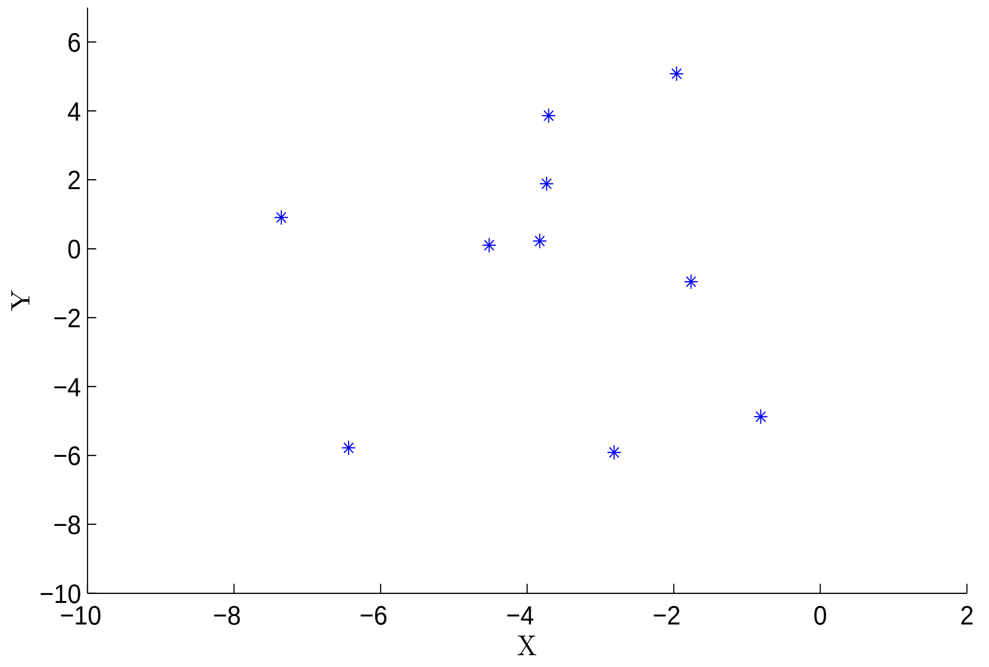
Bootstrap technique was introduced by Efron [36, 37] as a computer-based method to estimate the sampling distribution by repeatedly sampling *with replacement* from the original example. The sampling with replacement means the sample data in the original example have a chance to be drawn more than once or no chance at all. For the sampling distribution in bootstrap technique, it is usually proposed to derive the estimates of standard errors and to find the confidence intervals of a population parameter [37]. The following bootstrap algorithm to find standard error of mean is shown in *Algorithm 2*. Moreover, Figure 2.9 shows an example of bootstrap resampling concept which is used to calculate the standard error of mean. So to find this statistic, we can calculate the mean of each *B* bootstrap samples, where *B* specifies the number of iterations to resample, and the standard deviation of these means. In this example, the top box is an example of size $n = 7$ and $B = 3$. Therefore, the three lower boxes are the resamples from original example. Some values in the original example are repeated in the resamples because of using resampling with replacement method. After that, we calculate the sample mean of these resamples and calculate the standard deviation of all means. The standard error of mean value is shown at the bottom box.

**Algorithm 2.** Bootstrap Algorithm

```
Input: - $m$ examples $(x_1,\dots,x_m)$, where $x_i \in X$,
       - $B$ specifies the number of iteration to do resample.
       - set $t = 1$.
1. Do while $t \leq B$
2.     Draw the resample $(x^*)$ with replacement from $X$.
3.     Calculate the resample mean of each resampling iteration $(\overline{x}_t^*)$.
           $\overline{x}_t^* = \frac{1}{m}\Sigma(x^*)$
4.     $t = t + 1$.
5. End Do while
6. Calculate the standard error of mean $(SE_{boot})$.
           $SE_{boot} = \sqrt{\frac{1}{B-1}\Sigma_{t=1}^{B}(\overline{x}_t^* - mean_{boot})^2}$, where
           $mean_{boot} = \frac{1}{B}\Sigma_{t=1}^{B}\overline{x}_t^*$.
Output: the standard error of mean ($SE_{boot}$).
```



(a)



(b)

Figure 2.8: An example of a plot data. (a) a plot original data before using PCA. (b) a plot new data after using PCA

.

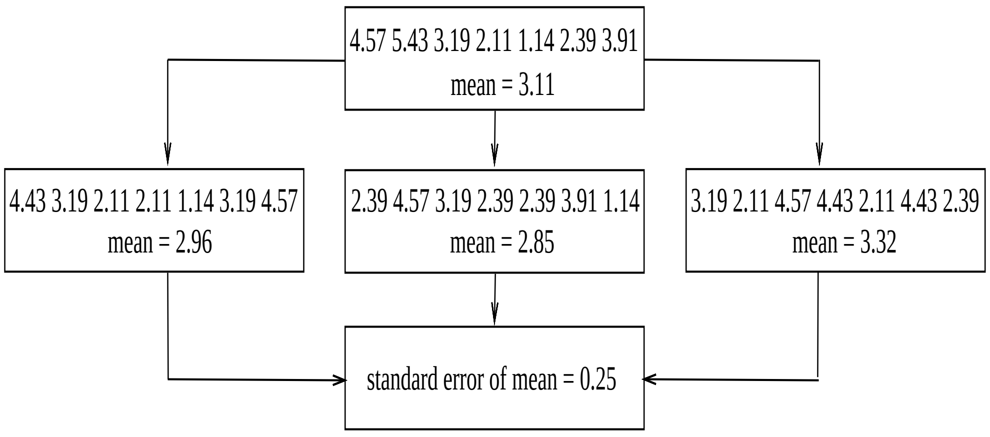


Figure 2.9 The example of bootstrap resampling to find standard error of mean.

# CHAPTER III

# PROPOSED METHODOLOGY

In this dissertation, the prediction of protein-protein interaction depends on three major steps. The first step is to extract the features of sequences in protein pairs to form feature vectors. The second step is to improve the performance of prediction by generating artificial boundary data on both interacting class and non-interacting class. The third step is to classify the set of feature sequences by using AdaBoost with feed-forward neural network into interacting and non-interacting protein classes. The feature extraction steps are as follows:

1. ***Representing amino acids by physiochemical properties***. Each amino acid is represented by its seven physiochemical properties.

2. ***Equalizing lengths of protein sequences***. For each protein sequence, the number of extracted physiochemical features depends on the number of amino acids appearing in the sequence. All current neural learning techniques cannot handle input patterns with various feature lengths. So equalizing these feature sequences is the essential part. The length of each feature sequence extracted from the first step must be equalized by applying the concept of correlation coefficient between physiochemical feature pair in the sequence.

3. ***Feature on secondary structures and protein properties***. In our preliminary experiment, we found that the accuracy of predicting protein-protein interactions based on only physiochemical properties of each amino acid is rather low. The essential factors dictating the interaction must be the structures and the other properties of the protein. So the secondary structures characteristics and protein properties are used as additional features. The other levels of structure, i.e. tertiary, quaternary, are not considered in this study.

   **The artificial boundary data generation step as follows:**

4. ***Finding the distribution direction in terms of eigenvectors and eigenvalues.*** To identify the new meaningful of these features before classifying. The features based on physiochemical properties and structural properties are calculated the new distribution direction and formed the new feature vectors based on eigenvectors and eigenvalues.

5. ***Using Self-Organizing Map to find subclusters.*** The protein features are recursively divided into small subclusters by SOM.

6. ***Finding boundary of each subcluster.*** To Speed up the training process. It is not necessary to use all the data. In this dissertation, the only boundary data will be used.

7. ***Using bootstrap resampling technique to generate artificial boundary data.*** Artificial boundary data in each subcluster are generated based on the bootstrap of maximum standard deviation.

   **The classifying step as follows:**

8. ***Predicting Protein-Protein Interactions by Feed-forward Neural Network with AdaBoosting.*** The only artificial boundary data are predicted by a feed-forward neural network with back-propagation learning rule. AdaBoost algorithm is used to boost the performance of this neural network.

The detail of each part is discussed in the following sections. Let $P_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,m_i})$ be protein sequence $i$ with a set of amino acids $p_{i,k}$ for $1 \le k \le m_i$. $m_i$ is the number of amino acids in $P_i$.

## 3.1 Part 1: Representing Amino Acids by Physiochemical Properties

There are seven physiochemical properties of amino acids [1] reflecting the interaction, which are hydrophobicity, hydrophicility, volumes of size chains of amino acids, polarity, polarizability, solvent-accessible surface area, and net charge index of side chains of amino acids. Let $b_{i,k}$, $c_{i,k}$, $v_{i,k}$, $a_{i,k}$, $r_{i,k}$, $s_{i,k}$, and $n_{i,k}$ be the hydrophobicity, hydrophicility, volumes of size chains of amino acids, polarity, polarizability, solvent-accessible surface area, and net charge index of side chains of amino acids properties of amino acid $p_{i,k}$ in protein $P_i$, respectively. Representing $P_i$ by the physiochemical properties is achieved by *Algorithm 1*.

**Algorithm 1**

```
1. For $p_{i,k}$, $1 \le k \le m_i$ do
2.     represent $p_{i,k}$ by $\{b_{i,k}, c_{i,k}, v_{i,k}, a_{i,k}, r_{i,k}, s_{i,k}, n_{i,k}\}$.
3. End
```

Suppose $P_i = (p_{i,1}, p_{i,2}, p_{i,3})$ and $P_j = (p_{j,1}, p_{j,2}, p_{j,3}, p_{j,4})$. After *Algorithm 1*, $P_i$ and $P_j$ become the sequences as shown in Figure 3.1.

| Protein | Amino Acids | | | |
|---|---|---|---|---|
| $P_i$ | $(p_{i,1}$ | $p_{i,2}$ | $p_{i,3})$ | |
| | $\downarrow$ | $\downarrow$ | $\downarrow$ | |
| | $b_{i,1}$ | $b_{i,2}$ | $b_{i,3}$ | |
| | $c_{i,1}$ | $c_{i,2}$ | $c_{i,3}$ | |
| | $v_{i,1}$ | $v_{i,2}$ | $v_{i,3}$ | |
| | $a_{i,1}$ | $a_{i,2}$ | $a_{i,3}$ | |
| | $r_{i,1}$ | $r_{i,2}$ | $r_{i,3}$ | |
| | $s_{i,1}$ | $s_{i,2}$ | $s_{i,3}$ | |
| | $n_{i,1}$ | $n_{i,2}$ | $n_{i,3}$ | |
| $P_j$ | $(p_{j,1}$ | $p_{j,2}$ | $p_{j,3}$ | $p_{j,4})$ |
| | $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ |
| | $b_{j,1}$ | $b_{j,2}$ | $b_{j,3}$ | $b_{j,4}$ |
| | $c_{j,1}$ | $c_{j,2}$ | $c_{j,3}$ | $c_{j,4}$ |
| | $v_{j,1}$ | $v_{j,2}$ | $v_{j,3}$ | $v_{j,4}$ |
| | $a_{j,1}$ | $a_{j,2}$ | $a_{j,3}$ | $a_{j,4}$ |
| | $r_{j,1}$ | $r_{j,2}$ | $r_{j,3}$ | $r_{j,4}$ |
| | $s_{j,1}$ | $s_{j,2}$ | $s_{j,3}$ | $s_{j,4}$ |
| | $n_{j,1}$ | $n_{j,2}$ | $n_{j,3}$ | $n_{j,4}$ |

Figure 3.1: An example of representing two protein sequences, $P_i$ and $P_j$, by physiochemical properties.

## 3.2 Part 2: Equalizing Lengths of Protein Sequences

Length equalizing process is based on the correlation coefficient computed from the physiochemical feature values of two amino acids $p_{i,j}$ and $p_{i,k}$, $j < k$. This implies that the distance between $p_{i,j}$ and $p_{i,k}$ is $k - j$ units apart. Each $P_i$ will be represented by only seven features of physiochemical properties. The following algorithm is used to equalize the lengths of all protein feature sequences obtained from *Algorithm 1*. Let $S$ be the total number of protein feature sequences and $l_i$ be the number of amino acids in protein $P_i$. The correlation coefficients for hydrophobicity, hydrophicility, volumes of size chains of amino acids, polarity, polarizability, solvent-accessible surface area, and net charge index of side chains of amino acids for protein feature sequence $i$ with distance $\delta$ are denoted by these notations: $C_{b_i,\delta}$, $C_{c_i,\delta}$, $C_{v_i,\delta}$, $C_{a_i,\delta}$, $C_{r_i,\delta}$, $C_{s_i,\delta}$, $C_{n_i,\delta}$, respectively. The use of correlation coefficients is based on the assumption that all amino acids on a protein sequence must correlate to one another in terms of physiochemical properties to form the protein sequence. Since the length of each protein $P_i$ is rather long, it is impractical to consider all possible distances between any pair $p_{i,j}$ and $p_{i,k}$, for $j < k$ and $(k-j) \in \{1,2,\ldots,l_i-1\}$. In this study, the maximum distance of $k-j$ is set to 25.

**Algorithm 2**

```
1.  For each protein feature sequence $i$, $1 \le i \le S$ compute
2.      $\overline{b}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} b_{i,j}$ ;
3.      $\overline{c}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} c_{i,j}$ ;
4.      $\overline{v}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} v_{i,j}$ ;
5.      $\overline{a}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} a_{i,j}$ ;
6.      $\overline{r}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} r_{i,j}$ ;
7.      $\overline{s}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} s_{i,j}$ ;
8.      $\overline{n}_i = \frac{1}{l_i}\Sigma_{j=1}^{l_i} n_{i,j}$ ;
9.      For each distance $\delta \in \{1,2,\ldots,30\}$ compute
10.         $C_{b_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(b_{i,j}-\overline{b}_i)\times(b_{i,j+\delta}-\overline{b}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(b_{i,j}-\overline{b}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(b_{i,j+\delta}-\overline{b}_i)^2}}$ ;
11.         $C_{c_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(c_{i,j}-\overline{c}_i)\times(c_{i,j+\delta}-\overline{c}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(c_{i,j}-\overline{c}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(c_{i,j+\delta}-\overline{c}_i)^2}}$ ;
12.         $C_{v_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(v_{i,j}-\overline{v}_i)\times(v_{i,j+\delta}-\overline{v}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(v_{i,j}-\overline{v}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(v_{i,j+\delta}-\overline{v}_i)^2}}$ ;
13.         $C_{a_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(a_{i,j}-\overline{a}_i)\times(a_{i,j+\delta}-\overline{a}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(a_{i,j}-\overline{a}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(a_{i,j+\delta}-\overline{a}_i)^2}}$ ;
14.         $C_{r_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(r_{i,j}-\overline{r}_i)\times(r_{i,j+\delta}-\overline{r}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(r_{i,j}-\overline{r}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(r_{i,j+\delta}-\overline{r}_i)^2}}$ ;
15.         $C_{s_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(s_{i,j}-\overline{s}_i)\times(s_{i,j+\delta}-\overline{s}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(s_{i,j}-\overline{s}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(s_{i,j+\delta}-\overline{s}_i)^2}}$ ;
16.         $C_{n_i,\delta} = \frac{\Sigma_{j=1}^{l_i-\delta}\left[(n_{i,j}-\overline{n}_i)\times(n_{i,j+\delta}-\overline{n}_i)\right]}{\sqrt{\Sigma_{j=1}^{l_i-\delta}(n_{i,j}-\overline{n}_i)^2\times\Sigma_{j=1}^{l_i-\delta}(n_{i,j+\delta}-\overline{n}_i)^2}}$ ;
17.     End
18. End
```

Since there are 25 distances and 7 correlation coefficients computed from each distance $\delta \in \{1,2,\ldots,25\}$ for each protein sequence, the total number of features after *Algorithm 2* is equal to $25 \times 7 = 175$. To predict the interaction, the features of each protein pair are concatenated to form a single feature sequence of length $175 \times 2 = 350$.

## 3.3 Part 3: Features on Secondary Structures and Protein Properties

The interaction between protein pairs essentially concerns their structures and other protein properties. Hence, the statistical information of the structures and the other properties must be considered. The secondary structures of a protein sequence measured in terms of statistical mean and standard deviation are additionally used. The types of structures considered in our case are alpha-helix, beta-sheet, beta-turn, coil, and parallel beta-strand. The relative frequencies of 29 proteins with 4741 residues in alpha-helix, beta-sheet, beta-turn of a set of globular proteins are calculated from their occurrences based on X-ray crystallographic data [38]. From [39], this technique calculated the conformational parameter for coil based on the 'double prediction method' and consists of a first prediction of the secondary structure from a new algorithm which uses parameters of the type described by Chou and Fasman. Parallel beta-strand were calculated from conformational preference for parallel beta strand by [40]. Moreover, the protein properties i.e.amino acid composition, hydrophobicity, average area buried, and polarity are also used. These statistical means and standard deviations of secondary structures and these protein properties are concatenated with the correlation coefficients computed by *Algorithm 2* to form a complete feature sequence. Since there are 12 basic structures and protein properties, each structure and protein properties have two features, i.e. its mean and standard deviation. Thus, the total number of structural and protein properties features is equal to $2 \times 12 = 24$ and, for any protein pair, the total number of structural and protein properties features becomes $2 \times 24 = 48$. Finally, the total features are 398 in each protein pair.

## 3.4 Part 4: Finding the distribution direction in terms of eigenvectors and eigenvalues.

This part is used to identify the new meaningful underlying these features. It is the way of finding the distribution directions in all features of the data space in terms of eigenvectors of data covariance matrix and their variances in term of eigenvalues of these eigenvectors. The following algorithm is shown in *Algorithm 3*.

**Algorithm 3.** Finding the new distribution direction of protein training data.

```
Input: - $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, where $\mathbf{X}$ is the protein training vector of $n$ features.
       - $\mu = \{\mu_1, \mu_2, ..., \mu_n\}$, where $\mu$ is the mean of each protein training vector.
1. Find covariance matrix $\Sigma$.
```

$\Sigma = E[(\mathbf{X} - \mu)(\mathbf{X} - \mu)^T]$

2. Calculate the eigenvectors **(v)** and eigenvalues $(\lambda)$ of the covariance matrix $\Sigma$.

$\Sigma\mathbf{v} = \lambda\mathbf{v}$

3. Deriving the new protein feature vector **A**

$\mathbf{A} = \mathbf{vX}$

**Output:** the new protein feature vector **A**.

Figure 3.2 shows a plot of the original training example data on the top and a plot of new training data after using *Algorithm 3* at the bottom.

## 3.5 Part 5: Using Self-Organizing Map to find subclusters.

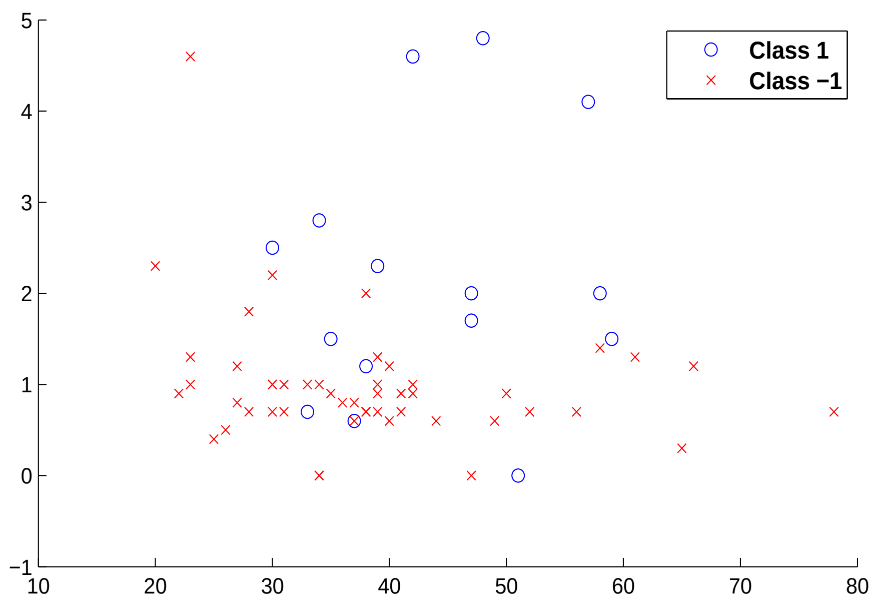
The aim of this process is to recursively divide the new training data into small subclusters for both interacting and non-interacting classes. The dividing process can be used SOM. To consider the example data in Figure 3.3, each subcluster is denoted by a random color. Each "$\bullet$" symbol is represented the data in interacting class and "$\times$" symbol is represented as the data in non-interacting class.

## 3.6 Part 6: Finding boundary of each subcluster.

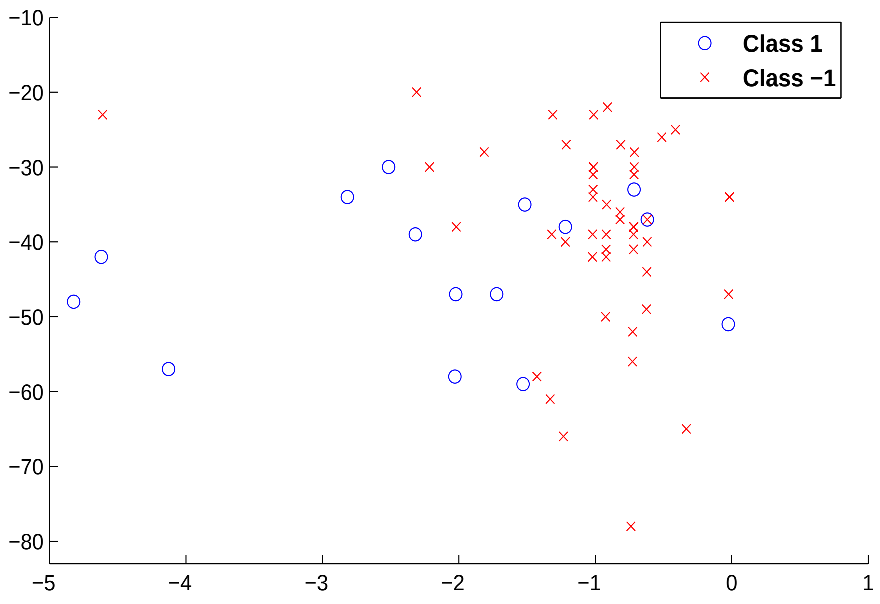
The main idea of this process is the only important data at the boundary between interacting class and non-interacting class will be used. In our dissertation, the *Data Selection* [41, 42] method was applied to find this boundary data. The concept is finding the nearest data or minimum distance between the data in interacting class and non-interacting class by using the Euclidean distance. The detail of finding the boundary of each subcluster between interacting class and non-interacting class is in *Algorithm 4*.

**Algorithm 4.** Finding the boundary of each subcluster between interacting class and non-interacting class.

```
Input: - $q$ interacting subclusters $(smin_1, \ldots, smin_q)$
       - $r$ non-interacting subclusters $(smaj_1, \ldots, smaj_r)$
1. For each interacting subcluster $smin_i$, $1 \le i \le q$
2.    For each non-interacting subcluster $smaj_j$, $1 \le j \le r$
```



(a)



(b)

Figure 3.2: Example of a plot training data (a) a plot original training data, (b) a plot new training data after using *Algorithm 3*.

3. **For** each data $p_k \in smin_i$, $1 \leq k \leq s$, where $s$ is the number of data in $smin_i$

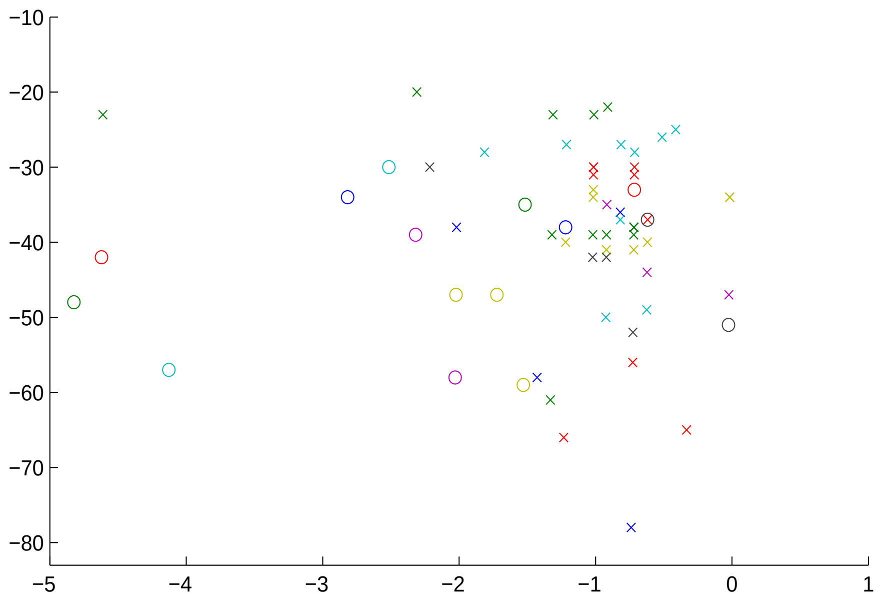


Figure 3.3 The example of subclusters are recursively divided by SOM.

```
4.          For each data $u_l \in smaj_j$, $1 \le l \le t$, where $t$ is the number of
            data in $smaj_j$
5.              Find the minimum distance between $p_k$ and $p_l$.
                $bmaj = \mathbf{MIN}(\|p_k - u_l\|)$, where $bmaj$ is the boundary
                data of non-interacting subcluster.
6.          End
7.        End
8.     End
9. End
10. For each non-interacting subcluster $smaj_j$, $1 \le j \le r$
11.    For each interacting subcluster $smin_i$, $1 \le i \le q$
12.        For each data $u_l \in smaj_j$, $1 \le l \le t$, where $t$ is the number of
           data in $smaj_j$
13.          For each data $p_k \in smin_i$, $1 \le k \le s$, where $s$ is the number of
             data in $smin_i$
14.              Find the minimum distance between $p_l$ and $p_k$.
                 $bmin = \mathbf{MIN}(\|u_l - p_k\|)$, where $bmin$ is the boundary
                data of interacting subcluster.
15.         End
16.       End
17.    End
18. End
```

**Output:** Boundary data of interacting class and non-interacting class in each subcluster.

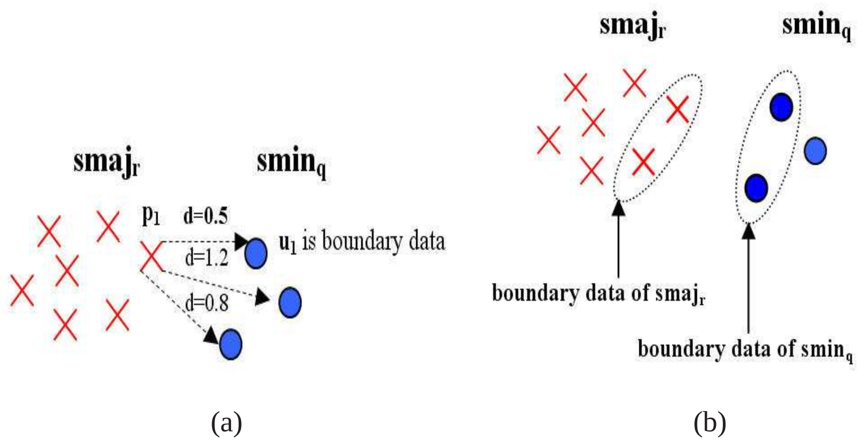


Figure 3.4: An example of finding boundary data of both interacting and non-interacting classes. (a) An example to find boundary data using Euclidean distance. (b) All boundary data of both interacting and non-interacting classes after using *Algorithm 4*.

Figure 3.4 displays the boundary data of both interacting and non-interacting classes. Figure 3.4(a) depicts the example to find boundary data which considering subcluster $smaj_r$ and $smin_q$. The boundary $u_1$ in subcluster $smin_q$ is calculated by finding the distance between data $p_1$ and every data in subcluster $smin_q$. The distance values are 0.5, 1.2, and 0.8, respectively. Thus, $u_1$ is the boundary data because of the minimum distance value 0.5. Figure 3.4(b) shows the all boundary data of both interacting and non-interacting classes after using *Algorithm* 4.

## 3.7 Part 7: Using bootstrap resampling technique to generate artificial boundary data.

In our dissertation, we use a neural network to classify the protein features. The neural network divides the input space by hyper-planes which are formed by connection weights. These hyper-planes are adopted to the class boundaries to classify the data. If the initial connection weights are properly initialized, the hyper-planes will be located near the class boundaries and will have the sufficiency of classification. Thus in this dissertation, the only artificial boundary data are used to train with the neural network.

These artificial boundary data are generated by applying bootstrap resampling technique. From the background section, bootstrap resampling technique is used to estimate the sampling distribution of sample data. Typically, the sampling distribution is based on many random samples from the population. In stead of many samples, bootstrap method builds many resamples by repeatedly random with replacement from only one random sample to represent the sampling distribution. So the bootstrap distribution is nearly close to the sampling distribution of the original data. Fig. 2.9 depicts an example of bootstrap resampling concept. The three lower boxes are the resamples with the sample means which are closed to the mean of the original data in the top box. For this reason, we apply the bootstrap resampling concept combine with the standard deviation of each subcluster in interacting and non-interacting classes to generate artificial boundary data. Bootstrap resampling concept are used to estimate the practical maximum standard deviation from the original sample data in each subcluster. Then the artificial boundary data are generated by this distribution to assure the future unseen data will fall into these class boundaries. Figure 3.5 shows the example of artificial boundary data after using bootstrap resampling method. The detail of using bootstrap resampling technique to generate artificial boundary data is described in *Algorithm 5*. Lines 1-8 compute the standard deviation of distance within each subcluster of interacting class and non-interacting class. Lines 9-15 describe **Case 1: If subcluster of interacting class has the maximum standard deviation of distance**. Then calculating the mean of bootstrap standard deviation. After that using the mean of bootstrap standard deviation to generate the artificial boundary data in both interacting class and non-interacting class (Line 16-27). Line 28-34 describes **Case 2: If subcluster of non-interacting class has the maximum standard deviation of distance.** Then calculating the mean of bootstrap standard deviation of this subcluster. After that using the mean of bootstrap standard deviation to generate the artificial boundary data in both interacting class and non-interacting class (Line 35-47). Figure 3.6 shows an example of new artificial boundary data. Figure 3.6(a) is the original boundary data and Figure 3.6(b) depicts the new artificial boundary data after using *Algorithm 5*.

**Algorithm 5.** Using bootstrap resampling technique to generate artificial boundary data for both interacting and non-interacting class.

**Input:** - $q$ interacting subclusters $(smin_1, ..., smin_q)$

- $r$ non-interacting subclusters $(smaj_1, ..., smaj_r)$

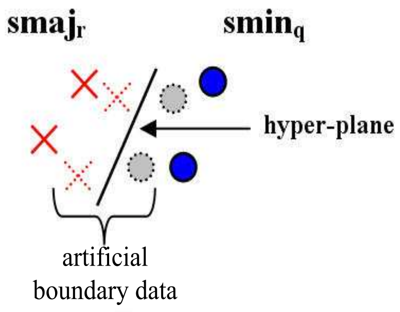


Figure 3.5 The example of new artificial boundary data.

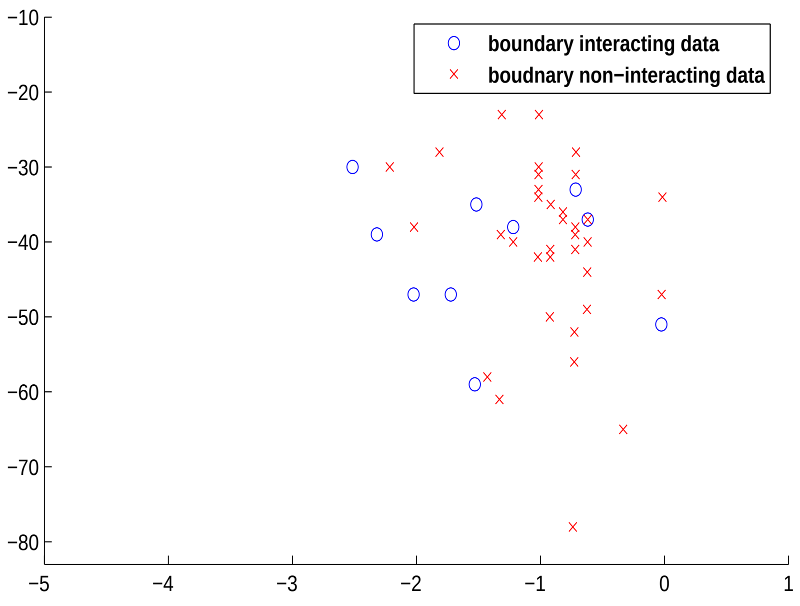
- standard deviation of data in each interacting subcluster $(SDmin_1, \ldots, SDmin_q)$
- standard deviation of data in each non-interacting subcluster $(SDmaj_1, \ldots, SDmaj_q)$
- $T = 1$, $B = 25$

Find the standard deviation of distance in each interacting subcluster and non-interacting subcluster.

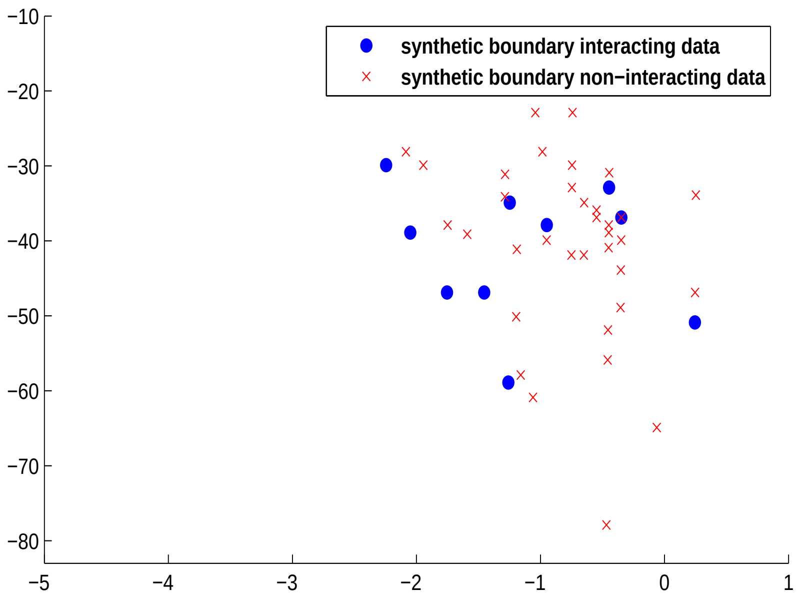
```
1. For each interacting subcluster $smin_i$, $1 \le i \le q$
2.     Calculate the average distance within interacting subcluster $smin_i$.
       $avmin_i = \frac{1}{s}\Sigma_{k=1}^{s}(\mathbf{MIN}_{\forall \mathbf{p}_m \in smin_i; k \neq m}\|\mathbf{p}_k - \mathbf{p}_m\|)$
3.     Calculate the standard deviation of distance
       $sdmin_i = \sqrt{\frac{1}{s-1}\Sigma_{k=1}^{s}(\mathbf{MIN}_{\forall \mathbf{p}_m \in smin_i; k \neq m}\|\mathbf{p}_k - \mathbf{p}_m\| - avmin_i)^2}$
4. End
5. For each non-interacting subcluster $smaj_j$, $1 \le j \le r$
6.     Calculate the average distance within non-interacting subcluster $smaj_j$.
       $avmaj_j = \frac{1}{t}\Sigma_{l=1}^{t}(\mathbf{MIN}_{\forall \mathbf{p}_v \in smaj_j; l \neq v}\|\mathbf{p}_l - \mathbf{p}_v\|)$
7.     Calculate the standard deviation of distance
       $sdmaj_j = \sqrt{\frac{1}{t-1}\Sigma_{l=1}^{t}(\mathbf{MIN}_{\forall \mathbf{p}_v \in smaj_j; l \neq v}\|\mathbf{p}_l - \mathbf{p}_v\| - avmaj_j)^2}$
8. End
```

Use bootstrap resampling method to calculate the mean of bootstrap standard deviation. Suppose $sdmin_q$ is the maximum value.

```
9.  If $sdmin_q$ is the maximum value Then
10.     Do while $T \le B$
11.         Draw the resample data with replacement from $smin_q$.
```



(a)



(b)

Figure 3.6: Example of a plot of boundary data (a) a plot original boundary data, (b) a plot new artificial boundary data after using *Algorithm 5*.

```
12.       Calculate the standard deviation of each resample $std^*_T$.
            $std^*_T = \sqrt{\frac{1}{s-1}\Sigma^s_{k=1}(p_k - \frac{1}{s}\Sigma^s_{k=1} p_k)^2}$
13.       $T = T+1$
14.     End Do while
15.     Calculate the mean of bootstrap standard deviation $mean_{std^*}$
            $mean_{std^*} = \frac{1}{B}\Sigma^B_{T=1} std^*_T$
   Generate artificial boundary data based on $mean_{std^*}$ value for both
interacting and non-interacting classes.
16.     For each interacting subcluster $smin_i$
17.       Calculate new standard deviation of data based on $mean_{std^*}$.
            $newSDmin_i = SDmin_i + (mean_{std^*} - SDmin_q)$
18.             For each data $p_k \in smin_i$
19.               Generate artificial boundary data.
                  $syn_k = p_k + (newSDmin_i - SDmin_i)$
20.             End
21.     End
22.     For each non-interacting subcluster $smaj_j$
23.       Calculate new standard deviation of data based on $mean_{std^*}$.
            $newSDmaj_j = SDmaj_j + (mean_{std^*} - SDmin_q)$
24.             For each data $p_l \in smaj_j$
25.               Generate artificial boundary data.
                  $syn_l = p_l + (newSDmaj_j - SDmaj_j)$
26.             End
27.     End
Suppose $sdmaj_r$ is the maximum value.
28.  Else
29.     Do while $T \leq B$
30.       Draw the resample data with replacement from $smaj_r$.
31.       Calculate the standard deviation of each resample $std^*_T$.
            $std^*_T = \sqrt{\frac{1}{t-1}\Sigma^t_{l=1}(p_l - \frac{1}{t}\Sigma^t_{l=1} p_l)^2}$
32.       $T = T+1$
33.     End Do while
```

```
34.  Calculate the mean of bootstrap standard deviation $mean_{std^*}$
         $mean_{std^*} = \frac{1}{B}\Sigma_{T=1}^{B} std_T^*$
   Generate artificial boundary data based on $mean_{std^*}$ value for both
interacting and non-interacting classes.
35.  For each interacting subcluster $smin_i$
36.      Calculate new standard deviation of data based on $mean_{std^*}$.
             $newSDmin_i = SDmin_i + (mean_{std^*} - SDmaj_r)$
37.          For each data $p_k \in smin_i$
38.              Generate artificial boundary data.
                     $syn_k = p_k + (newSDmin_i - SDmin_i)$
39.          End
40.  End
41.  For each non-interacting subcluster $smaj_j$
42.      Calculate new standard deviation of data based on $mean_{std^*}$.
             $newSDmaj_j = SDmaj_j + (mean_{std^*} - SDmaj_r)$
43.          For each data $p_l \in smaj_j$
44.              Generate artificial boundary data.
                     $syn_l = p_l + (newSDmaj_j - SDmaj_j)$
45.          End
46.  End
47. EndIf
Output: - The artificial boundary data of interacting class.
        - The artificial boundary data of non-interacting class.
```

## 3.8 Part 8: Predicting Protein-Protein Interactions by Feed-forward Neural Network with AdaBoosting.

The only artificial boundary features of two interacting and non-interacting proteins are trained by a feed-forward neural network with backpropagation learning rule which AdaBoost technique has been applied to this neural network classifier. The interacting protein pairs are assigned to class 1 and those non-interacting pairs are assigned to class $-1$.

# CHAPTER IV

# RESULTS AND DISCUSSION

## 4.1 Data Sets

### 4.1.1 *Training data set*

*Yeast Saccharomyces Cerevisiae*

In our experiments, we used a data set of physical protein interactions obtained from [1], protein pairs containing a protein less than 50 amino acids or have $\geq$ 40% sequence identity were removed. These data are composed by 5594 positive data and 5594 negative data. The negative data were generated by the assumption that proteins with different sub-cellular localizations do not interact. The final data set consists of 11188 protein pairs.

### 4.1.2 *Cross-species data set*

For protein pairs of *Drosophila Melanogaster* from [3], the protein pairs containing a protein with less than 50 amino acids were removed. So the protein pairs has 4220 entries.

The other data set contains four species (i.e. *Caenorhabditis elegans*, *Escherichia coli*, *Homo sapiens*, and *Mus musculus*) in DIP database as our cross-species test data set which obtained from [43]. *Caenorhabditis elegans* consists of 4013 protein pairs. *Escherichia coli* consists of 6954 protein pairs. *Homo sapiens* consists of 1412 protein pairs, and *Mus musculus* consists of 313 protein pairs.

## 4.2 Performance Evaluation

In this dissertation, the performance of a classifier is evaluated by various measures [30, 44], *Overall Accuracy*, *Sensitivity*, *Precision*, *G-Mean* and *F-Measures*. The confusion matrix as shown in Table 4.1 represents the contingency table [28] for evaluating the performance of machine learning algorithm on the classification learning problems. Let $\{p,n\}$ be the positive and negative testing examples and $\{Y,N\}$ be the classification results given by a learning algorithm for positive and negative predictions [28, 44]. In this dissertation uses the interacting class as the positive class and non-interacting class as the negative class.

Table 4.1 Confusion Matrix.

| | Predicted Positve (Y) | Predicted Negative (N) |
|---|---|---|
| Actual Positive (p) | TP (True Positives) | FN (False Negatives) |
| Actual Negative (n) | FP (False Positives) | TN (True Negatives) |

Based on Table 4.1, **TP** is true positive defined as the right recognition of true interacting protein pairs. **TN** is true negative defined as the right recognition of true non-interacting protein pairs. **FN** is false negative defined as the wrong recognition of true interacting protein pairs. **FP** is false positive defined as the wrong recognition of true non-interacting protein pairs. The evaluation metrics used to assess the prediction protein pairs data sets are defined as:

Overall Accuracy (OA) :

$$OA = \frac{TP+TN}{TP+FP+FN+TN} \tag{4.1}$$

TP Rate (Sensitiviry):

$$TPRate = \frac{TP}{TP+FN} \tag{4.2}$$

FP Rate :

$$FPRate = \frac{FP}{FP+TN} \tag{4.3}$$

Precision :

$$Precision = \frac{TP}{TP+FP} \tag{4.4}$$

Recall :

$$Recall = \frac{TP}{TP+FN} \tag{4.5}$$

F-Measure :

$$F-Measure = \frac{(1+\beta^2)\times Recall \times Precision}{\beta^2 \times Recall + Precision} \tag{4.6}$$

where $\beta$ is a coefficient to adjust the relative importance of precision versus the recall (usually $\beta = 1$). *F-Measure* is high when both the *recall* and *presicion* are high. It indicates that the *F-Measure* is the measure of **goodness** of a learning algorithm on the interest class [30].

G-mean :

$$G - Mean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \tag{4.7}$$

*G-Mean* is used to evaluate a performance of classifier on the skew data. $TP/(TP+FN)$ is called *Positive Accuracy* and $TN/(TN+FP)$ is called *Negative Accuracy*. The idea of *G-Mean* is to maximize the accuracy on each of two classes while these accuracies still balanced [30].

## 4.3 The proposed method results

### 4.3.1 *Assessment of prediction capability*

The result for *Yeast Saccharomyces Cerevisiae* data set as show in Table 4.2. The final data set consists of 11188 protein pairs, where half are from the positive data set and half from the negative data set. Three-fifths of the protein pairs from the positive and negative data set were respectively randomly chosen as the training set, and the remaining two-fifths were used as the test set. To minimize data dependence on the prediction model, five training sets and five test sets were respectively prepared. Each training data set were averaged over *five* times experiments. Moreover, each training set an ensemble of *ten* component classifiers was created. In the experiments, a feed-forward neural network with backpropagation learning rule was used as based classifier.

We present the results achieved when using Auto covariance [1], Local descriptor [2], and our proposed method. Our proposed method shows the results of predicting protein-protein interactions before generating artificial boundary data and after generating artificial boundary data. For each method in Table 4.2 illustrates the results in terms of the *Overall Accuracy*, *Sensitivity*, *Precision*, *F-measures*, *G-mean*, *TP rates*, and the average prediction performance across five runs.

Our proposed method *(using original training data)* which using 398 feature vectors yields a protein-protein interactions model with the average overall accuracy, sensitivity, precision, F-measures of both classes, G-Mean, and TP rates of both classes at 90.02%, 88.01%, 91.69%, 89.81%, 90.22%, 90.00%, 88.01% and 92.03%, respectively. The average prediction overall accuracy, sensitivity, precision, F-measures of both classes, G-Mean, and TP rates of both classes of the Auto covariance method and the Local descriptor method were also calculated. It can be found that Local descriptor method which using 1260 feature vectors achieves the better performance than Auto covariance

Table 4.2: Results of *Yeast Saccharomyces Cerevisiae* data set : *Overall accuracy*, *Sensitivity*, *Precision*, *F-measure* of interacting (positive) class, *F-measure* of non-interacting (negative) class, *G-mean*, *True Positive rate* of interacting (positive) class, and *True Positive rate* of non-interacting (negative) class using (1) Auto covariance, (2) Local descriptor, and (3) our proposed method.

| **Methods** | **Number of Feature** | **Number of Training Data** | **Test set** | **Overall Accuracy** | **Sensitivity** | **Precision** | **F-measure of positive class** | **F-measure of negative class** | **G-Mean** | **TP rate of positive class** | **TP rate of negative class** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Auto covariance | 420 | 6713 | 1 | 88.66 | 85.85 | 90.96 | 88.33 | 88.97 | 88.62 | 85.85 | 91.47 |
| | | | 2 | 88.13 | 85.42 | 90.31 | 87.80 | 88.44 | 88.08 | 85.42 | 90.83 |
| | | | 3 | 88.64 | 86.67 | 90.22 | 88.41 | 88.86 | 88.62 | 86.67 | 90.61 |
| | | | 4 | 88.66 | 87.16 | 89.86 | 88.49 | 88.83 | 88.65 | 87.16 | 90.16 |
| | | | 5 | 88.44 | 86.40 | 90.07 | 88.20 | 88.67 | 88.42 | 86.40 | 90.47 |
| | | | Average | 88.51 | 86.30 | 90.28 | 88.25 | 88.75 | 88.48 | 86.30 | 90.71 |
| Local descriptor | 1260 | 6713 | 1 | 89.24 | 87.19 | 90.92 | 89.02 | 89.46 | 89.22 | 87.19 | 91.29 |
| | | | 2 | 89.29 | 86.76 | 91.38 | 89.01 | 89.55 | 89.25 | 86.76 | 91.82 |
| | | | 3 | 88.86 | 87.05 | 90.32 | 88.66 | 89.06 | 88.84 | 87.05 | 90.67 |
| | | | 4 | 89.53 | 87.48 | 91.23 | 89.32 | 89.75 | 89.51 | 87.48 | 91.59 |
| | | | 5 | 88.55 | 86.36 | 90.32 | 88.29 | 88.80 | 88.52 | 86.36 | 90.74 |
| | | | Average | 89.12 | 86.97 | 90.83 | 88.88 | 89.35 | 89.09 | 86.97 | 91.26 |
| our proposed method (original training data) | 398 | 6713 | 1 | 89.71 | 87.68 | 91.39 | 89.50 | 89.91 | 89.67 | 87.68 | 91.74 |
| | | | 2 | 90.59 | 89.13 | 91.80 | 90.45 | 90.72 | 90.57 | 89.13 | 92.04 |
| | | | 3 | 89.91 | 87.32 | 92.09 | 89.64 | 90.17 | 89.87 | 87.32 | 92.50 |
| | | | 4 | 89.58 | 87.30 | 91.47 | 89.34 | 89.81 | 89.55 | 87.30 | 91.86 |
| | | | 5 | 90.32 | 88.64 | 91.72 | 90.15 | 90.48 | 90.30 | 88.64 | 91.99 |
| | | | Average | 90.02 | 88.01 | 91.69 | 89.81 | 90.22 | 90.00 | 88.01 | 92.03 |
| our proposed method (artificial boundary data) | 398 | 6654 | 1 | 90.04 | 88.62 | 91.22 | 89.90 | 90.18 | 90.03 | 88.62 | 91.47 |
| | | | 2 | 90.65 | 88.55 | 91.80 | 90.45 | 90.85 | 90.63 | 88.55 | 92.75 |
| | | | 3 | 90.38 | 88.13 | 92.29 | 90.16 | 90.59 | 90.35 | 88.13 | 92.63 |
| | | | 4 | 90.63 | 88.42 | 92.51 | 90.42 | 90.83 | 90.60 | 88.42 | 92.84 |
| | | | 5 | 90.65 | 88.55 | 92.44 | 90.45 | 90.85 | 90.63 | 88.55 | 92.75 |
| | | | Average | **90.47** | **88.45** | **92.18** | **90.28** | **90.66** | **90.45** | **88.45** | **92.49** |

method which using 420 feature vectors. Otherwise, the performance of our proposed method *(using original training data)* is better than the other two methods. The overall accuracy, sensitivity, precision, F-measures of both classes, G-Mean, and TP rates of both classes values obtained from our proposed method are 1.51%, 1.71%, 1.41%, 1.56%, 1.47%, 1.52%, 1.71% and 1.32% which higher than the Auto covariance method. Moreover, the overall accuracy, sensitivity, precision, F-measures of both classes, G-Mean, and TP rates of both classes values obtained from our proposed method are 0.9%, 1.04%, 0.86%, 0.93%, 0.87%, 0.91%, 1.04% and 0.77% which also higher than the Local descriptor method. These results show the best performance of our proposed method compared to the other two methods.

The average prediction overall accuracy, sensitivity, precision, F-measures of both classes, G-Mean, and TP rates of both classes of our proposed method ***(using only artificial boundary data)*** are 90.47%, 88.45%, 92.18%, 90.28%, 90.66%, 90.45%, 88.45% and 92.49%, respectively. Our proposed method using only 6654 artificial boundary training data and 398 feature vectors which less than Auto covariance and Local descriptor method. Compare to Auto covariance, Local descriptor, and our proposed method *(using original training data)*, our proposed method which using ***only artificial boundary data*** achieves the best performance. The overall accuracy, sensitivity, precision, F-measures of both classes, G-Mean, and TP rates of both classes values obtained from our proposed method are 1.96%, 2.15%, 1.9%, 2.03%, 1.91%, 1.97%, 2.15% and 1.78% which higher than the Auto covariance method. The performance values obtained from our proposed method are 1.35%, 1.48%, 1.35%, 1.4%, 1.31%, 1.36%, 1.48% and 1.23% which higher than the Local descriptor method. This our result indicated that our proposed method which using ***only artificial boundary data*** is large improvement against with Auto covariance and Local descriptor method. Furthermore, to consider in F-measure of both classes, our proposed method consistently produced higher performance than other methods. Hence, these results indicate that our proposed method does not only achieve learning at one class, but also achieve learning at both classes of interacting and non-interacting class against other methods.

### 4.3.2 *Performance on cross-species data set*

In another evaluation, we tested the ability of our proposed method for predicting protein-protein interactions with cross-species data set. The results from section ***Assessment of prediction capability*** provide the final five training models. In order to evaluate our training models, we tested these model against the data sets which are independent of the training data set. Thus, we chose the other five species as our cross-species test data set. The performance of our proposed method is summarized in Table 4.3, Table 4.4, Table 4.5, Table 4.6, and Table 4.7. The average performance of our proposed method which using only artificial boundary data to predict on *Caenorhabditis elegans*, *Drosophila Melanogaster*, *Escherichia coli*, *Homo sapiens*, and *Mus musculus* achieved at 81.27%, 79.18%, 76.70%, 84.65%, and 86.77%, respectively. These average performance are shown that our proposed method can predict correctly in the interacting protein pairs with the accuracy over 80% on *Caenorhabditis elegans*, *Homo sapiens*, and *Mus musculus*. However the average performance on *Drosophila Melanogaster* and *Escherichia coli* data set have lower 80%, the average performance of our proposed method still outperforms other methods. It claims that our proposed method which using only artificial boundary data is also able to predict the cross-species data with the better performance than other two methods.

Table 4.3: The prediction results of *Caenorhabditis elegans* data set based on final five training models of *Yeast Saccharomyces Cerevisiae* data set using (1) Auto covariance, (2) Local descriptor, and (3) our proposed method.

| **Methods** | **Test set** | **Overall Accuracy** |
|---|---|---|
| | 1 | 73.34 |
| | 2 | 72.79 |
| Auto covariance | 3 | 74.41 |
| | 4 | 74.91 |
| | 5 | 72.86 |
| | Average | 73.66 |
| | 1 | 77.37 |
| | 2 | 75.93 |
| Local descriptor | 3 | 75.50 |
| | 4 | 77.52 |
| | 5 | 77.17 |
| | Average | 76.70 |
| | 1 | 83.10 |
| | 2 | 81.34 |
| Our proposed method | 3 | 80.19 |
| (artificial boundary data) | 4 | 81.16 |
| | 5 | 80.56 |
| | Average | **81.27** |

Table 4.4: The prediction results of *Drosophila Melanogaster* data set based on final five training models of *Yeast Saccharomyces Cerevisiae* data set using (1) Auto covariance, (2) Local descriptor, and (3) our proposed method.

| **Methods** | **Test set** | **Overall Accuracy** |
|---|---|---|
| | 1 | 74.88 |
| | 2 | 74.05 |
| Auto covariance | 3 | 75.12 |
| | 4 | 75.88 |
| | 5 | 75.02 |
| | Average | 74.99 |
| | 1 | 78.08 |
| | 2 | 78.25 |
| Local descriptor | 3 | 76.64 |
| | 4 | 79.43 |
| | 5 | 76.54 |
| | Average | 77.79 |
| | 1 | 82.91 |
| | 2 | 76.94 |
| Our proposed method | 3 | 80.50 |
| (artificial boundary data) | 4 | 76.78 |
| | 5 | 78.79 |
| | Average | **79.18** |

Table 4.5: The prediction results of *Escherichia coli* data set based on final five training models of *Yeast Saccharomyces Cerevisiae* data set using (1) Auto covariance, (2) Local descriptor, and (3) our proposed method.

| **Methods** | **Test set** | **Overall Accuracy** |
|---|---|---|
| Auto covariance | 1 | 76.03 |
| | 2 | 75.11 |
| | 3 | 76.19 |
| | 4 | 77.14 |
| | 5 | 74.42 |
| | Average | 75.78 |
| Local descriptor | 1 | 63.50 |
| | 2 | 65.14 |
| | 3 | 63.29 |
| | 4 | 65.11 |
| | 5 | 63.34 |
| | Average | 64.08 |
| Our proposed method (artificial boundary data) | 1 | 77.96 |
| | 2 | 75.47 |
| | 3 | 76.68 |
| | 4 | 77.73 |
| | 5 | 75.68 |
| | Average | **76.70** |

Table 4.6: The prediction results of *Homo sapiens* data set based on final five training models of *Yeast Saccharomyces Cerevisiae* data set using (1) Auto covariance, (2) Local descriptor, and (3) our proposed method.

| **Methods** | **Test set** | **Overall Accuracy** |
|---|---|---|
| Auto covariance | 1 | 78.97 |
| | 2 | 79.46 |
| | 3 | 79.39 |
| | 4 | 79.46 |
| | 5 | 79.39 |
| | Average | 79.33 |
| Local descriptor | 1 | 75.50 |
| | 2 | 73.51 |
| | 3 | 75.35 |
| | 4 | 74.15 |
| | 5 | 74.79 |
| | Average | 74.66 |
| Our proposed method (artificial boundary data) | 1 | 87.89 |
| | 2 | 83.50 |
| | 3 | 84.77 |
| | 4 | 83.92 |
| | 5 | 83.14 |
| | Average | **84.65** |

Table 4.7: The prediction results of *Mus musculus* data set based on final five training models of *Yeast Saccharomyces Cerevisiae* data set using (1) Auto covariance, (2) Local descriptor, and (3) our proposed method.

| **Methods** | **Test set** | **Overall Accuracy** |
|---|---|---|
| Auto covariance | 1 | 79.87 |
| | 2 | 81.15 |
| | 3 | 83.07 |
| | 4 | 83.39 |
| | 5 | 83.07 |
| | Average | 82.11 |
| Local descriptor | 1 | 75.72 |
| | 2 | 74.44 |
| | 3 | 75.08 |
| | 4 | 72.84 |
| | 5 | 72.84 |
| | Average | 74.19 |
| Our proposed method (artificial boundary data) | 1 | 89.46 |
| | 2 | 86.58 |
| | 3 | 87.22 |
| | 4 | 84.98 |
| | 5 | 85.62 |
| | Average | **86.77** |

Our proposed method at **the artificial data generation step** not only using as the method to predict protein-protein interactions but also is applied to handle the imbalanced data problem.

Standard machine learning algorithms have an error to classify imbalanced data sets which the number of data in one (majority) class is larger than the other (minority) classes. The performance of traditional classifiers produce high predictive accuracy over majority class but low predictive accuracy on minority class. We proposed a new method for handling the imbalanced data sets. The main idea of our proposed is based on the fact that only artificial boundary data are generated on both minority and majority classes and using boosting technique to improve the performance of classification. Firstly, the new direction distribution of input data sets are calculated and used to form the new input data. Then using Self-Organizing Map (SOM) to partition these new data into many subclusters of minority and majority classes. After that only boundary data of each subcluster are used to generate synthetic boundary data based on bootstrap resampling technique. To increase efficiency of prediction, only artificial boundary data are classified base on Adaptive Boosting (AdaBoost) algorithm. Our proposed method applied to ten highly imbalanced data sets using feed-forward backpropagation neural network as base classifier. In addition, the *F-measures, G-mean* and *overall accuracy* are used to evaluated the merit of classification results. From the experimental results, our proposed method is very efficient to handle imbalanced data and our performance is higher than other methods.

## 4.4 Experimental Results on Imbalanced Data Sets

### 4.4.1 Benchmark Data Sets

Ten data sets are used to test with our proposed method as summarized in Table 4.8. Table 4.8 shows the number of example in the data set, the number of minority class, the number of majority class, the class distribution, and the number of input features. These data sets are available from the UCI Machine Learning Repository [45]. Moreover, these data sets is used to test the learning of imbalanced data problem from two-class, we made modifications on several of the originals data sets according to the literary results from similar experiment [28, 30]. A brief description of modification is discussed as follows.

1. *Monk Data set.* This data set includes 169 examples and 2 classes. Each example is represented by 6 attributes [30]. There are 105 majority class examples and 64 minority class examples.

2. *Ionosphere Data set.* This data set includes 351 examples with 2 classes (good radar returns and bad radar returns). Each example is represented by 34 numeric features. We choose 225

Table 4.8 Summary of the data sets used in this paper.

| **Data set** | **Case** | **Min. Class** | **Maj. Class** | **Class Dist.** | **Feature** |
|---|---|---|---|---|---|
| Monk2 | 169 | 64 | 105 | 0.37:0.63 | 6 |
| Ionosphere | 351 | 126 | 225 | 0.35:0.65 | 34 |
| Breast-W | 699 | 241 | 458 | 0.34:0.66 | 9 |
| Vehicle | 846 | 199 | 647 | 0.23:0.77 | 18 |
| Hepatitis | 155 | 32 | 123 | 0.20:0.80 | 19 |
| Glass | 214 | 29 | 185 | 0.13:0.87 | 9 |
| Vowel | 990 | 90 | 900 | 0.09:0.91 | 13 |
| Abalone | 731 | 42 | 689 | 0.06:0.94 | 8 |
| Yeast | 483 | 20 | 463 | 0.04:0.96 | 8 |
| Car | 1728 | 65 | 1663 | 0.04:0.96 | 6 |

examples of good radar returns as majority class and 126 examples of bad radar returns as minority class.

3. *Breast Cancer Wisconsin Data set.* This data set has a total of 699 examples and 2 classes (benign, malignant). Each example is represented by 9 attributes. We use benign class as the majority class and malignant as minority class, which give us 458 majority class examples and 241 minority class examples.

4. *Vehicle Data set.* This data set has a total of 846 data examples and 4 classes (opel, saab, bus and van). Each example is represented by 18 attributes. Class *Van* is selected as the minority class and the remaining classes is used as the majority class. This gives us an imbalanced two-class dataset, with 199 minority class examples and 647 majority class examples.

5. *Hepatitis Data set.* This data set includes 199 examples with 2 classes (die and live). Each example is represented by 19 features. We choose 123 examples of class live as majority class and 32 examples of class die as minority class.

6. *Glass identification Data set.* This data set has a total of 214 examples and 6 classes. Each example is represented by 9 attributes. We choose class *headlamps* as the minority class and the remaining classes are combined as the majority class. This gives an imbalanced two-classes

dataset, with 29 minority class examples and 185 majority class examples.

7. *Vowel recognition Data set.* The original dataset includes 990 examples and 11 classes. Each example is represented by 13 attributes. Since each vowel in the original data has 10 examples, we choose the first vowel as the minority class and the remaining examples are in the majority class [28]. Therefore, there are 90 examples in minority class and 990 examples in majority class.

8. *Abalone Data set.* The original dataset includes 4177 examples and 29 classes. Each example in represented by 8 attributes. We choose class *18* as the minority class and class *9* as the majority class [28, 30]. This gives us 42 minority class examples and 689 majority class examples.

9. *Yeast Data set.* The original dataset includes 1484 examples and 10 classes. Each example in represented by 8 attributes. We choose class *POX* as the minority class and class *CYT* as the majority class [30]. This gives us 20 minority class examples and 463 majority class examples.

10. *Car evaluation Data set.* The original dataset includes 1728 examples and 4 classes. Each example in represented by 6 attributes. We choose class *v-good* as the minority class and the remaining class as the majority class. This gives us 65 minority class examples and 1663 majority class examples.

### 4.4.2 The proposed method result on benchmark data sets

Results for the ten data sets, as shown in Table 4.9 and 4.10. These data sets were averaged over five standard 10-fold cross validation experiments. For each 10-fold cross validation, the data set was first partitioned into 10 equal sized sets and each set was in turn used as the test set while the classifier trains on the other nine sets [30]. For each fold an ensemble of *ten* component classifiers was created. In the experiments, a feed-forward neural network with backpropagation learning rule was used as based classifier.

For each data set, we present the results achieved when using the feed-forward neural network (NN), AdaBoostM1, SMOTEBoost [29], DataBoost-IM [30], and our proposed method. For each method, Table 4.9 and 4.10 present the results in terms of the *overall accuracy*, *F-measures*, *G-mean* and *TP rates*.

The results as shown in Table 4.9 indicate that our proposed method performs very well in term of *overall accuracy*, *F-measures* of both minority and majority classes, and *TP rate* of majority class. In some cases, the value of *G-Mean* and *TP rate* of minority class are slightly higher or the same

Table 4.9: Results of imbalanced data sets : *Overall accuracy*, *F-measure* of minority class, *F-measure* of majority class, *G-mean*, *true positive rate* of minority class, *true positive rate* and of majority class using (1) the NN classifier, (2) AdaBoostM1, (3) SMOTEBoost, (4) DataBoost-IM, and (5) our proposed method.

| **Data set Name** | **Methods** | **Number of Training Data** | **Overall Accuracy** | **F-measure of min. class** | **F-measure of maj. class** | **G-Mean** | **TP rate of min. class** | **TP rate of maj. class** |
|---|---|---|---|---|---|---|---|---|
| Hepatitis | NN | 140 | 82.13 | 59.22 | 88.30 | 71.64 | 60.83 | 88.01 |
| | AdaBoostM1 | 140 | 79.46 | 51.83 | 86.75 | 67.12 | 54.17 | 86.28 |
| | SMOTEBoost | 313 | 80.75 | 54.83 | 87.62 | 70.51 | 60.83 | 86.22 |
| | DataBoost-IM | 140 | 80.79 | 54.49 | 87.59 | 69.07 | 57.50 | 87.12 |
| | **Our proposed method** | **120** | **84.00** | **59.55** | **89.94** | 71.00 | 57.50 | **91.15** |
| Ionosphere | NN | 316 | 86.64 | 78.60 | 90.23 | 81.98 | 71.47 | 95.14 |
| | AdaBoostM1 | 316 | 87.21 | 79.47 | 90.65 | 82.39 | 71.47 | 96.05 |
| | SMOTEBoost | 722 | 85.79 | 77.39 | 89.53 | 81.03 | 70.64 | 94.21 |
| | DataBoost-IM | 785 | 88.34 | 80.56 | 91.62 | 82.85 | 70.58 | 98.22 |
| | **Our proposed method** | **258** | **89.21** | **81.66** | **92.33** | **83.66** | 71.47 | **99.11** |
| Vehicle | NN | 761 | 96.34 | 92.30 | 97.60 | 95.32 | 93.50 | 97.22 |
| | AdaBoostM1 | 761 | 96.93 | 93.43 | 97.99 | 95.84 | 93.97 | 97.83 |
| | SMOTEBoost | 1473 | 96.34 | 92.34 | 97.59 | 95.63 | 94.47 | 96.91 |
| | DataBoost-IM | 1542 | 97.17 | 94.07 | 98.14 | 96.56 | 95.50 | 97.68 |
| | **Our proposed method** | **459** | **97.52** | **94.70** | **98.38** | **96.58** | 94.95 | **98.30** |
| Monk2 | NN | 152 | 68.70 | 58.03 | 74.84 | 65.67 | 58.33 | 75.45 |
| | AdaBoostM1 | 152 | 74.00 | 64.31 | 79.23 | 70.84 | 64.29 | 80.27 |
| | SMOTEBoost | 290 | 70.95 | 56.17 | 76.39 | 62.66 | 55.71 | 80.09 |
| | DataBoost-IM | 326 | 69.25 | 58.74 | 75.24 | 66.24 | 59.29 | 75.55 |
| | **Our proposed method** | **151** | **76.43** | **66.20** | **81.81** | **72.79** | **64.76** | **84.09** |
| Abalone | NN | 658 | 94.39 | 39.80 | 97.06 | 52.01 | 35.50 | 97.97 |
| | AdaBoostM1 | 658 | 95.49 | 50.71 | 97.63 | 60.68 | 42.50 | 98.70 |
| | SMOTEBoost | 1142 | 92.20 | 51.56 | 95.73 | 79.47 | 69.00 | 93.62 |
| | DataBoost-IM | 1359 | 95.08 | 49.57 | 97.41 | 62.25 | 45.50 | 98.11 |
| | **Our proposed method** | **376** | **96.17** | **54.01** | **98.00** | 60.30 | 47.50 | **99.13** |

Table 4.10: Results of imbalanced data sets : *Overall accuracy*, *F-measure* of minority class, *F-measure* of majority class, *G-mean*, *true positive rate* of minority class, *true positive rate* and of majority class using (1) the NN classifier, (2) AdaBoostM1, (3) SMOTEBoost, (4) DataBoost-IM, and (5) our proposed method.

| **Data set Name** | **Methods** | **Number of Training Data** | **Overall Accuracy** | **F-measure of min. class** | **F-measure of maj. class** | **G-Mean** | **TP rate of min. class** | **TP rate of maj. class** |
|---|---|---|---|---|---|---|---|---|
| Vowel | NN | 891 | 99.19 | 95.66 | 99.55 | 98.01 | 96.67 | 99.44 |
| | AdaBoostM1 | 891 | 99.09 | 95.14 | 99.50 | 97.96 | 96.67 | 99.33 |
| | SMOTEBoost | 1719 | 98.89 | 94.08 | 99.39 | 97.84 | 96.67 | 99.11 |
| | DataBoost-IM | 1915 | 98.59 | 92.39 | 99.22 | 96.14 | 93.33 | 99.11 |
| | **Our proposed method** | **510** | **99.39** | **96.72** | **99.67** | **98.64** | **97.78** | **99.56** |
| Glass | NN | 193 | 96.26 | 86.00 | 97.84 | 91.72 | 86.67 | 97.84 |
| | AdaBoostM1 | 193 | 95.35 | 81.57 | 97.32 | 89.06 | 83.33 | 97.31 |
| | SMOTEBoost | 561 | 95.28 | 84.10 | 97.22 | 92.70 | 90.00 | 96.17 |
| | DataBoost-IM | 640 | 95.35 | 82.48 | 97.31 | 90.69 | 86.67 | 96.78 |
| | **Our proposed method** | **123** | **96.28** | **86.38** | 97.84 | **93.20** | 90.00 | 97.34 |
| Yeast | NN | 435 | 97.30 | 49.67 | 98.61 | 55.17 | 45.00 | 99.57 |
| | AdaBoostM1 | 435 | 97.30 | 49.67 | 98.61 | 55.17 | 45.00 | 99.57 |
| | SMOTEBoost | 708 | 95.02 | 40.67 | 97.38 | 58.43 | 45.00 | 97.18 |
| | DataBoost-IM | 901 | 97.30 | 49.67 | 98.61 | 55.17 | 45.00 | 99.57 |
| | **Our proposed method** | **311** | **97.72** | **61.00** | **98.82** | **67.99** | **60.00** | 99.35 |
| Breast-W | NN | 629 | 94.84 | 92.45 | 96.08 | 94.10 | 92.10 | 96.28 |
| | AdaBoostM1 | 629 | 95.13 | 92.98 | 96.27 | 94.67 | 93.35 | 96.07 |
| | SMOTEBoost | 1142 | 95.27 | 93.07 | 96.40 | 94.91 | 94.17 | 95.85 |
| | DataBoost-IM | 1295 | 95.00 | 92.72 | 96.17 | 94.36 | 92.53 | 96.28 |
| | **Our proposed method** | **294** | **95.85** | **93.87** | **96.86** | **95.19** | 93.35 | **97.15** |
| Car | NN | 1555 | 99.83 | 97.79 | 99.91 | 99.20 | 98.57 | 99.88 |
| | AdaBoostM1 | 1555 | 99.88 | 98.46 | 99.94 | 99.23 | 98.57 | 99.94 |
| | SMOTEBoost | 2324 | 99.76 | 97.03 | 99.88 | 99.25 | 98.57 | 99.76 |
| | DataBoost-IM | 2913 | 99.94 | 99.09 | 99.97 | 99.13 | 98.33 | 100.00 |
| | **Our proposed method** | **626** | 99.94 | **99.23** | 99.97 | 99.26 | 98.57 | 100.00 |

value as other techniques. In Table 4.10, the results of our proposed method perform well in term of *F-measure* of minority class and *G-Mean*. Moreover, the *overall accuracies* of our proposed method are higher than other techniques. In some cases, the value of *F-measure* of majority class and *TP rate* of both classes are slightly higher or the same value as other techniques.

1. *Hepatitis Data set.* Our proposed method using only 120 training data which less than the other four methods. The *overall accuracy*, *F-measure* of both classes and *TP rate* of majority class indicate that our proposed method performs very well when comparing with the base classifier NN, AdaBoostM1, SMOTEBoost, and DataBoost-IM methods. The proposed method achieved an *overall accuracy* 84%, *F-measures* of both classes at values 59.55%, and 89.94% respectively, *TP rate* of majority class at values 91.15%.

2. *Ionosphere Data set.* There are 258 training data which are used in our proposed method, 316 training data are used in NN and AdaBoostM1, 561 training data are used in SMOTEBoost, and 640 training data are used in DataBoost-IM. The result shows that our proposed method performs well in terms of *overall accuracy*, both minority and majority classes of *F-measures*, *G-Mean*, and *TP rate* of majority class which are large improvement against with base classifier NN, AdaBoostM1, SMOTEBoost, and DataBoost-IM methods. The proposed method achieved an *overall accuracy* 89.21%, both *F-measures* of minority and majority classes at values 81.66% and 92.33%, *G-Mean* of 83.66%, and *TP rate* of majority class at values 99.11%. In addition, the *TP rate* of minority class, our proposed method is the same as NN, AdaBoostM1 but higher than SMOTEBoost, DataBoost-IM.

3. *Vehicle Data set.* Our proposed method using only 459 training data is less than other four methods. The *overall accuracy*, *F-measures* of both classes, *G-Mean* and *TP rates* of both classes in our proposed method is higher than the base classifier NN, AdaBoostM1, and SMOTEBoost. Otherwise, our the *overall accuracy*, *F-measures* of both classes, *G-Mean* and *TP rates* of both classes perform slightly higher than DataBoost-IM. The *TP rate* of minority class, our proposed method is lower than DataBoost-IM by only 0.55%.

4. *Monk Data set.* In our proposed method, there are only 151 training data are used. The *overall accuracy*, both *F-measures*, *G-Mean*, and both *TP rates* of minority and majority classes of our proposed method surpasses of all the other methods. The proposed method achieved an *overall accuracy* 76.43%, both *F-measures* of minority and majority classes at values 66.20% and 81.81%, *G-Mean* of 72.79%, both *TP rates* of minority and majority classes at values 64.76%

and 84.09%, respectively.

5. *Abalone Data set.* The only 376 training data are used in our proposed. Our proposed method achieved 96.17% *overall accuracy*, *F-measures* and *TP rates* of both minority and majority at 54.01%, 98%, 47.50%, and 99.13%, respectively which indicate that there are large improvements of our proposed method than NN, AdaBoostM1, and DataBoost-IM.

6. *Vowel recognition Data set.* There are only 510 training data which are used in our proposed method. Our proposed method performs 99.39% *overall accuracy*, 96.72% *F-measure* of minority class, 99.67% *F-measure* of majority class, 98.64% of *G-Mean*, 97.78% *TP rate* of minority class, and 99.56% *TP rate* of majority class. The results in *F-measure* of minority class, *G-Mean* and *TP rate* of minority class surpass that of all the other methods and the majority class of *F-measure* and *TP rate* are slightly higher than other methods.

7. *Glass identification Data set.* There are 123 are used for our proposed method. Our performance is evaluated by *overall accuracy*, *G-Mean* and *F-measures* are comparable with others. The proposed method performs 96.28% *overall accuracy* which higher than AdaBoostM1, SMOTEBoost, and DataBoost-IM methods, and slightly higher than NN method. The values of both *F-measures* minority and majority class of our proposed method are 86.38% and 97.84% which higher than AdaBoostM1, SMOTEBoost, and DataBoost-IM or slightly higher than NN method. Moreover, *G-mean* value and *TP rate* of minority class of our proposed method are highest of all other techniques at values 93.20% and 90%, respectively. TP rate of majority class at 97.34% higher than AdaBoostM1, SMOTEBoost and DataBoost-IM but lower than NN method by 0.5%.

8. *Yeast Data set.* There are only 311 training set being used in our proposed method. Our *overall accuracy* and *F-measure* of majority class are slightly better than NN, AdaBoostM1, and DataBoost-IM. Otherwise, our proposed method performs highest in terms of *F-measure* of minority class, *G-Mean* and *TP rated* of minority class which achieved a minority class *F-measure* of 61.00%, *G-Mean* of 67.99% and a minority class *TP rate* of 60.00%. Furthermore, the performance of all measures in our proposed method are absolutely higher than SMOTEBoost method.

9. *Breast Cancer Wisconsin Data set.* There are only 294 training data are used in our proposed method. The minority class *F-measure* of 93.87%, *G-Mean* of 95.19% and the majority class *TP rate* of 97.15% show that our proposed method is the highest performance. However in

terms of *overall accuracy* at 95.85%, *F-measure* of majority class at 96.86% and the minority TP rate of 93.35% indicate that our proposed method performs slightly higher than NN, AdaBoostM1, and DataBoost-IM.

10. *Car evaluation Data set.* In our proposed method using only 626 training data while there are at least 1555 training data are used in the other techniques. Our proposed method performs 99.94% *overall accuracy*, 99.23% *F-measure* of minority class, 99.97% *F-measure* of majority class, 99.26% of *G-Mean*, 98.57% *TP rate* of minority class and *TP rate* of majority class at value 100%. Our proposed method produced slightly higher or similar to other four methods in all terms of evaluation. In this data set, our proposed method is obtained that the only 626 training data still achieve with high performance.

In this work, our proposed method using only artificial boundary training data which the number of training data are less than other four methods. These results show the superiority of our proposed method compared to the others. Moreover, our proposed method achieved promising results when considering *overall accuracy* and *F-measures* of both classes. Such as Hepatitis, Ionosphere, Monks2 and Abalone the *overall accuracy* and *F-measure of minority class* surpass of all the others.

# CHAPTER V

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

This dissertation proposes a new feature extraction from only primary protein sequences for predicting protein-protein interactions. Each protein sequence is characterized by its correlation coefficients with physiochemical properties, statistical information of its secondary structures and protein properties as additional features. The prediction is achieved by using a feed-forward neural network with boosting technique. The experimental results indicated that the proposed method performs better than the others' method. The number of features of *Yeast Saccharomyces Cerevisiae* data in our case is obviously less than those of the other's.

Moreover, bootstrap resampling technique is used to generate artificial boundary data. So the only artificial boundary data are used to construct the final prediction model. The proposed method can improve the performance of predicting protein-protein interactions in terms of the *Overall Accuracy*, *Sensitivity*, *Precision*, *F-measures*, *G-mean*, *TP rates*. The performance prediction shows the superiority of proposed method which using only artificial boundary data compared to other methods. In addition, to consider in *F-measure* of both classes, these results indicated that our proposed method does not only improve predicting at one class, but also achieve predicting at both classes of interacting and non-interacting class.

The prediction performance of the final model which using only artificial boundary was evaluated on cross-species data set to obtain a more reliable assessment. It can be found that the proposed method is capable of prediction over cross-species which outperforms other methods. Hence these features from the proposed method can be represented as the features of *Caenorhabditis elegans*, *Drosophila Melanogaster*, *Escherichia coli*, *Homo sapiens*, and *Mus musculus*.

In addition, Our proposed method at **the artificial data generation step** not only using as the method to predict protein-protein interactions but also is applied to handle the imbalanced data problem. Our results are promising and show that the proposed method compares well in comparison with a neural network base classifier, a standard boosting algorithm and two advanced boosting-based algorithms for handling imbalanced data set. The results indicate that the proposed method does not achieve one class prediction, but also produce high predictions against both minority and majority classes.

## 5.2 Future Work

There are some solutions which can make this research more effective as follows:

- The other physicochemical properties and other protein properties will be considered as feature representation. Since the number of feature vector of protein pairs may be reduced which still have the good performance of predictions.
- To speed up the finding boundary data process, the other algorithm to find the boundary data should be instead.
- Overall, the proposed method is expected to be a powerful tool for expediting the study of protein interaction networks.

# REFERENCES

[1] Guo, Y., Yu, L., Wen, Z., and Li, M. Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. Nucleic Acids Res 36 (2008): 3025-3030.

[2] Yang, L., Xia, J., and Gui, J. Prediction of Protein-Protein Interactions from Protein Sequence Using Local Descriptors. Protein & Peptide Letters 17 (2010): 1085-1090.

[3] Liu, L., Cai, Y., Lu, W., Feng, K., Peng, C., and Niu, B. Prediction of protein-protein interactions based on PseAA composition and hybrid feature selection. Biochemical and Biophysical Research Communications 380 (2009): 318-322.

[4] Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., and Sakaki, Y. A comprehensive two-hybrid analysis to explore the yeast protein interactome. Proc Natl Acad Sci USA 98 (2001): 4569-4574.

[5] Ho, Y., Gruhler, A., Heilbut, A., Bader, G., Moore, L., Adams, S., Millar, A., Taylor, P., Bennett, K., Boutilier, K., Yang, L., Wolting, C., Donaldson, I., Schandorff, S., Shewnarane, J., Vo, M., Taggart, J., Goudreault, M., Muskat, B., Alfarano, C., Dewar, D., Lin, Z., Michalickova, K., Willems, A., Sassi, H., Nielsen, P., Rasmussen, K., Andersen, J., Johansen, L., Hansen, L., Jespersen, H., Podtelejnikov, A., Nielsen, E., Crawford, J., Poulsen, V., Sorensen, B., Matthiesen, J., Hendrickson, R., Gleeson, F., Pawson, T., Moran, M., Durocher, D., Mann, M., Hogue, C., Figeys, D., and Tyers, M. Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. Nature 415 (2002): 180-183.

[6] Zhu, H., Bilgin, M., Bangham, R., Hall, D., Casamayor, A., Bertone, P., Lan, N., Jansen, R., Bidlingmaier, S., Houfek, T., Mitchell, T., Miller, P., Dean, R., Gerstein, M., and Snyderm, M. Global analysis of protein activities using proteome chips. Science 293 (2001): 2101-2105.

[7] Skrabanek, L., Saini, H., Bader, G., and Enright, A. Computational Prediction of Protein-Protein Interactions. Mol Biotechnol 38 (2008): 1-17.

[8] Pellegrini, M., Marcotte, E., Thompson, M., Eisenberg, D., and Yeates, T. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. Proc Natl Acad Sci USA 96 (1999): 4285-4288.

[9] Tamames, J., Casari, G., Ouzounis, C., and Valencia, A. Conserved clusters of functionally related genes in two bacterial genomes. J Mol Evol 44 (1997): 66-73.

[10] Enright, A., Iliopoulos, I., Kyrpides, N., and Ouzounis, C. Protein interaction maps for complete genomes based on gene fusion events. Nature 402 (1999): 86-90.

[11] Grigoriev, A. A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage T7 and the yeast Saccharomyces cerevisiae. Nucleic Acids Res 29 (2001): 3513-3519.

[12] Jansen, R., Greenbaum, D., and Gerstein, M. Relating whole-genome expression data with protein-protein interactions. Genome Res 12 (2002): 37-46.

[13] Gallet, X., Charloteaux, B., Thomas, A., and Brasseur, R. A fast method to predict protein interaction sites from sequences. J Mol Biol 302 (2000): 917-926.

[14] Keskin, O., Gursoy, A., Ma, B., and Nussinov, R. Principles of protein-protein interactions: what are the preferred ways for proteins to interact?. Chem Rev 108 (2008): 1225-1244.

[15] Ofran, Y. and Rost, B. Analysing six types of protein-protein interfaces. J Mol Biol 325 (2003): 377-387.

[16] Ogmen, U., Keskin, O., Aytuna, A., Nussinov, R., and Gursoy, A. PRISM: protein interactions by structural matching. Nucleic Acids Res 33 (2005): W331-336.

[17] Bradford, J. and Westhead, D. Improved prediction of protein-protein binding sites using a support vector machines approach. Bioinformatics 21 (2005): 1487-1494.

[18] Li, X., Keskin, O., Ma, B., Nussinov, R., and Liang, J. Protein-protein interactions: hot spots and structurally conserved residues often locate in complemented pockets that pre-organized in the unbound states: implications for docking. J Mol Biol 344 (2004): 781-795.

[19] Li, H. and Li, J. Discovery of stable and significant binding motif pairs from PDB complexes and protein interaction datasets. Bioinformatics 21 (2005): 314-324.

[20] Bock, J. and Gough, D. Predicting protein–protein interactions from primary structure. Bioinformatics 14 (2001): 455-460.

[21] Chou, K. and Cai, Y. Predicting protein-protein interactions from sequences in a hybridization space. J Proteome Res 5 (2006): 316-322.

[22] Shen, J., Zhang, J., Luo, X., Zhu, W., Yu, K., Chen, K., Li, Y., and Jiang, H. Predicting protein-protein interactions based only on sequences information. Proc Natl Acad Sci USA 104 (2007): 4337-4341.

[23] Pitre, S., Dehne, F., Chan, A., Cheetham, J., Duong, A., Emili, A., Gebbia, M., Greenblatt, J., Jessulat, M., Krogan, N., Luo, X., and A, G. PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs. BMC Bioinformatics 7 (2006): 365.

[24] Ma, Z., Zhou, C., Lu, L., Ma, Y., Sun, P., and Cui, Y., Predicting protein-protein interactions based on BP neural network. Proceedings 2007 IEEE international conference on bioinformatics and biomedicine: 2-4 November 2007; Silicon Valley, USA, (2007): 3-7.

[25] Estabrooks, A., Jo, T., and Japkowicz, N. A Multiple Resampling Method for Learning from Imbalanced Data Sets. Computational Intelligence 20 (2004): 18-36.

[26] Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer, W. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16 (2002): 321-357.

[27] Hui, H., Wenyuan, W., and Binghuan, M., Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. ICIC, (2005): 878-887.

[28] Haibo, H., Yang, B., Garcia, E., and Shutao, L., ADASYN: Adaptive synthetic sampling approach for imbalanced learning. Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, (2008): 1322-1328.

[29] Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K., SMOTEBoost: improving prediction of the minority class in boosting. In Proceedings of the Principles of Knowledge Discovery in Databases, (2003): 107–119.

[30] Guo, H. and Viktor, H. Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach. SIGKDD Explorations 6 (2004): 30-39.

[31] Neural Network Toolbox For Use with MATLAB.

[32] Murphey, Y., Chen, Z., and Guo, H., Neural learning using AdaBoost. Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on, (2001): 1037-1042.

[33] Sun, Y., Kamel, M., Wong, A., and Wang, Y. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition 40 (2007): 3358-3378.

[34] Freund, Y. and Schapire, R. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1997): 119-139.

[35] Guo, H. and Viktor, H., Boosting with data generation: improving the classification of hard to learn examples. Proceedings of the 17th international conference on Innovations in applied artificial intelligence, (2004): 1082-1091.

[36] Efron, B. Bootstrap Methods: Another Look at the Jackknife. Ann. Statistics 7 (1979): 1-26.

[37] Efron, B. and Tibshirani, R. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. Statistical Science 1 (1986): 54-75.

[38] Chou, P. and Fasman, G. Prediction of the secondary structure of proteins from their amino acid sequence. Adv Enzymol Relat Areas Mol Biol 47 (1978): 45-148.

[39] Deleage, G. and Roux, B. An algorithm for protein secondary structure prediction based on class prediction. Protein Eng 1 (1987): 289-294.

[40] Lifson, S. and Sander, C. Antiparallel and parallel beta-strands differ in amino acid residue preferences. Nature 282 (1979): 109-111.

[41] Hara, K. and Nakayama, K., A training method with small computation for classification. Proc. of the IEEE-INNS-ENNS International Joint Conference, (2000): 543-548.

[42] Wattanachon, U. and Lursinsap, C. SPSM: a New Hybrid Data Clustering Algorithm for Nonlinear Data Analysis. IJPRAI 23 (2009): 1701-1737.

[43] Xia, J., Zhao, X., and Huang, D. Predicting proteinprotein interactions from protein sequences using meta predictor. Amino Acids 39 (2010): 1595-1599.

[44] Haibo, H. and Garcia, E. Learning from Imbalanced Data. Knowledge and Data Engineering, IEEE Transactions on 21 (2009): 1263-1284.

[45] Frank, A. and Asuncion, A., UCI Machine Learning Repository., 2010.

# Biography

**Education:**

- Ph.D. Program in Computer Science, Chulalongkorn University, Thailand (June 2007 - May 2012).
- M.Sc. Program in Computer Science, Prin of Songkla University, Thailand (June 2004 - May 2007).
- B.Sc. Program in Management Information System, Walailak University, Thailand (June 2000 - May 2004).

**Publication:**

- Thanathamathee, P.; Lursinsap, C., "Predicting Protein-Protein Interactions Using Correlation Coefficient and Principal Component Analysis", in *3rd International Conference on Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009.*, Beijing, China, 11-13 June 2009, pp. 1-4, 2009.

**Scholarship:**

- The Office of Higher Education Commission, Ministry of Education, Thailand through Faculty of Informatics (Computer Science), Walailak University.