

เทคนิคการทำนายสัญลักษณ์นิวคลีโอไทด์ที่คลุมเครือในลำดับดีเอ็นเอ



นางสาว กิติพร พลายมาศ

สถาบันวิทยบริการ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคณนา ภาควิชาคณิตศาสตร์


คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2547

ISBN 974-17-6498-7

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

A TECHNIQUE FOR PREDICTING AN AMBIGUOUS NUCLEOTIDE SYMBOL
IN A DNA SEQUENCE



Miss Kitiporn Plaimas

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computational Science

Department of Mathematics

Faculty of Science

Chulalongkorn University

Academic Year 2004

ISBN 974-17-6498-7

กิติพร พลายมาศ : เทคนิคการทำนายสัญลักษณ์นิวคลีโอไทด์ที่คลุมเครือในลำดับดีเอ็นเอ. A TECHNIQUE FOR PREDICTING AN AMBIGUOUS NUCLEOTIDE SYMBOL IN A DNA SEQUENCE) อ. ที่ปรึกษา : ศาสตราจารย์ ดร. ชิดชนก เหลือสินทรัพย์, 67 หน้า. ISBN 974-17-6498-7.

ลำดับดีเอ็นเอหรือลำดับของนิวคลีโอไทด์ A, T, C และ G ที่สกัดมาจากเซลล์ของสิ่งมีชีวิตโดยเครื่องอ่านลำดับดีเอ็นเอ อาจให้ลำดับดีเอ็นเอไม่สมบูรณ์ ที่มีบางลำดับของดีเอ็นเอเป็นสัญลักษณ์ที่คลุมเครืออย่างสัญลักษณ์ N ที่หมายถึง A, T, C, หรือ G ในงานวิจัยนี้ได้ศึกษาหาวิธีการแก้ปัญหาดังกล่าวไปเป็นปัญหาการรู้จำลำดับก่อนหน้าของสัญลักษณ์ N ด้วยสมมุติฐานที่ว่านิวคลีโอไทด์แต่ละตำแหน่งในลำดับดีเอ็นเอย่อมมีความสัมพันธ์กับนิวคลีโอไทด์ในบริเวณข้างเคียง ดังนั้นตำแหน่งที่สัมพันธ์กันของนิวคลีโอไทด์จึงเป็นรูปแบบหลักที่ใช้ในการสอนและรู้จำของโครงข่ายประสาทเทียม อย่างไรก็ตาม การรู้จำคุณลักษณะทั้งหมดของข้อมูลที่ใช้สอนจะใช้เวลานาน ดังนั้นเราจึงพิจารณาถึงการเพิ่มความเร็วของการรู้จำแบบขนานด้วย และได้ทำการทดสอบกับจีโนมของแบคทีเรียอีโคไลทั้งหมด 4 สายพันธุ์ โดยสุ่มเลือกบริเวณที่มีลำดับการเรียงตัวของนิวคลีโอไทด์ใกล้เคียงกันในดีเอ็นเอด้วยความยาวกว่า 4 หมื่นเบสมาหลายๆ บริเวณด้วยกัน โดยไม่คำนึงถึงว่าเป็นบริเวณที่มีฮีนหรือไม่ แต่ละบริเวณสามารถสอนโครงข่ายประสาทเทียมให้รู้จำความคล้ายคลึงกันและอิทธิพลการเกิดนิวคลีโอไทด์ตัวถัดไปได้เพื่อทำนายสัญลักษณ์ที่แท้จริงของ N ได้ ดังนั้นเมื่อทดลองสุ่มข้อมูลเพื่อทดสอบการทำนายของโครงข่ายประสาทเทียมแล้วให้ความถูกต้องในการรู้จำมากกว่า 80%

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา.....คณิตศาสตร์.....ลายมือชื่อนิสิต.....
สาขาวิชา.....วิทยาการคอมพิวเตอร์.....ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา.....2547.....

4572225323 : MAJOR COMPUTATIONAL SCIENCE

KEY WORD: AMBIGUOUS NUCLEOTIDE / NEURAL NETWORKS / BIOINFORMATICS / SEQUENCING / HIGH PERFORMANCE

KITIPORN PLAIMAS : A TECHNIQUE FOR PREDICTING AN AMBIGUOUS NUCLEOTIDE SYMBOL IN A DNA SEQUENCE. THESIS ADVISOR : PROFESSOR CHIDCHANOK LURSINSAP, Ph.D., [67] pp. ISBN 974-17-6498-7.

DNA sequences obtained from a DNA sequencer usually contain some ambiguous symbol N , which can be interpreted as either A , or T , or C , or G . This ambiguity can effect the informative analysis of the DNA sequence. This research focused on transforming this problem to a problem of recognizing a prefix sequence of symbol N . By our assumption that nucleotides and their positions may be related to their neighboring nucleotides, the relative positions are used as the feature of the sequence during the learning and recognizing processes of a neural network for each nucleotide. However, recognizing these features from a training set may take a lengthy time. The problem of increasing the training speed in forms of parallel recognition was also investigated. Experimenting on four *Eschericia coli* genomes, we selected similar regions of about 40,000 bases from any regions. Each region can train an artificial neural network to recognize all similarity and predict the actual symbol of N . From random query testing sets, the recognition accuracy is more than 80%.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Department Mathematics..... Student's signature.....

Field of study Computational science..... Advisor's signature.....

Academic year 2004.....

ACKNOWLEDGEMENTS

I am greatly indebted to my supervisor, Professor Dr. Chidchanok Lursinsap, for his suggestions, guidance, cheering up and care, help me to overcome the necessary difficulties of the process of study and reseach, and make this thesis possible. Without him my thesis would have never been accomplished. I also thank Supannika Lursinsap for her care and kindness.

I also wish to express my special thanks to the thesis committee, Associate Professor Dr. Wanida Hemakul, Assistant Professor Dr. Paisan Nakmahachalasint, and Dr. Rath Pichyangkura, for their valuable advise, reading and criticizing the manuscript. They helped me focus on my reseach activities. And, I would like to thank Associate Professor Suchada Siripant, who looked after me when I stayed in AVIC research center to do my research.

Furthermore, this work is partially supported by National Center for Genetic Engineering and BioTechnology (BIOTECH) and Chulalongkorn University. I would also like to thank The Development and Promotion of Science and Technology Talents Project (DPST) for financial aid during my study.

I am also grateful to all my colleagues and friends at the Advanced Virtual and Intelligent Computing (AVIC) Center, especially Benjamas Panyangam, Benchaporn Jantarakongkul, Dussadee Praserttitipong, Kodchakorn Na Nakornphanom, Maytee Bamrungrajhirun, Wanyok Atisattapong, and Chantarat Polutith for their care, having encouraged and supported me during my study.

My thanks also go to all my past and present instructors for their valuable lectures and instructions. Finally, I would like to express my sincere gratitude to my parents and family members for their love, hearty encouragement, and unselfish sacrifice, and, especially, to Apichat Suratane, for his love, wormest care, and being patient during my confusing and frustrating stage.

CONTENTS

	Page
ABSTRACT(THAI)	iv
ABSTRACT(ENGLISH)	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER	
1 INTRODUCTION	1
1.1 The Problem Identification	1
1.2 The Objective and Scope	2
1.3 The Outline of the Thesis	2
2 BACKGROUND KNOWLEDGES	3
2.1 DNA Structure and DNA Sequences	3
2.1.1 Components of DNA	3
2.1.2 DNA Sequences	5
2.2 Mutational Changes of DNA Sequences	6
2.3 An Overview of Artificial Neural Networks	8
2.4 Backpropagation Neural Network	10
3 LITERATURE REVIEWS	16
3.1 The Review of Literature Related to DNA Sequencing Problems	16
3.2 The Review of Literature Related to SPM Predictor Derived from Information Theory	21

CHAPTER	Page
4 A PROPOSED TECHNIQUE FOR PREDICTING AMBIGUOUS NUCLEOTIDE SYMBOL	24
4.1 Problem Formulation	24
4.2 Relative Positions	25
4.2.1 Extended Relative Position with Probability of Mutation Rate	26
4.3 Recognition Process	26
4.3.1 Input Representation	26
4.3.2 Partitioning Training Feature Vectors	29
4.3.3 Network Architecture	35
5 EXPERIMENTAL RESULTS	38
5.1 Performance of each Recognition Network	41
5.2 The Results of Resolving Method	42
5.3 The Results of Query Sequences	44
5.4 Performance of Parallel Technique	47
6 CONCLUSION	50
REFERENCES	52
VITAE	54

LIST OF TABLES

TABLE	Page
2.1 Uniform mutation rates with 1% probability of change at each position.	7
2.2 Higher transitions (α) than transversions(β) with 1% probability of change at each position.	8
3.1 Summary of single-letter code recommendations.	18
4.1 Example of nucleotide sequence.	27
4.2 The feature vectors of the training patterns in Table 4.1.	29
4.3 A table of possible output patterns and the resolved symbols.	37
5.1 The results of duplicating process of the original feature vectors vary in the number of relative positions.	41
5.2 The performance of each classification network on four sample regions.	42
5.3 Experimental results using the conclusion in Table 4.3.	43
5.4 Experimental results using the maximum output value.	44
5.5 Experimental results using the patterns in Table 4.3 of query sequence	46
5.6 Experimental results using the maximum output value of query sequences	47
5.7 Experimental results of recognition network comparing with Sampled Pattern Matching(SPM) for 4,000 samples	47
5.8 The performance results of this parallel technique.	48

LIST OF FIGURES

FIGURE	Page
2.1 The Structure of Deoxyribonucleic Acid (DNA)	4
2.2 Structure of purine and pyrimidine bases	4
2.3 Two DNA strands form a double helix.	5
2.4 Double-stranded DNA.	6
2.5 Nucleic acid substitutions diagram.	7
2.6 Analogy between human and artificial neuron.	9
2.7 Illustrating the sample network that has only one input layer, one hidden layer, and one output layer. There are q nodes represented input features in the input layer, many hidden neurons we can adjust for an applicableneural network, and there are any output neurons.	11
3.1 The percentage of correct prediction using SPM for <i>E.coli</i> genome with noncoding and coding region.	22
4.1 Suppose feature vectors have three dimensions and the third feature in Z -axis is the key feature. Therefore, all feature vectors are partitioned into three groups denoted by three blocks. Each group of feature vectors is trained by an individual neural network. . . .	30

LIST OF FIGURES (Continued)

FIGURE	Page
4.2 An example of how feature vectors are partitioned. There are 12 feature vectors whose feature values lying in the range $[0, 5]$. These feature values are divided into five ($n = 5$) intervals. The width of each interval is set to $d = \frac{5-0}{5} = 1$. Step 1 shows the number of values, $N_{value}(i)$, lying in the i -th interval. The important interval is the third interval because most of the values lie in it. Step 2 shows the number of feature vectors, $N_{pattern}(3, k)$, having their feature values lying in the third interval, which is the important interval. The second feature was used as the key feature to divide all feature vectors. Step 3 shows three groups of feature vectors after partitioning. For this example, the number of networks is set to 3. Each groups must contains less than or equal to $avg = \frac{12}{3} = 4$ patterns. Thus, the last group contains the patterns in the remaining intervals	32
4.3 An example of how to recursively apply the partitioning concept. Each block denotes a group of feature vectors. (a) The original given feature vectors to be partitioned. (b) The given feature vectors in (a) are partitioned into four groups. (c) Groups 2 and 3 in (b) are further partitioned into four subgroups.	34
4.4 Network Architecture and the determination by thresholding the outputs from all networks and comparing them with the patterns in Table 4.3.	35
4.5 Network Architecture and the determination by selecting maximum output values.	36

LIST OF FIGURES (Continued)

- 5.1 The relationship between the number of duplicated patterns and the number of relative positions. 40
- 5.2 The relationship between the number of relative positions and the complexity of unduplicated data set. 40
- 5.3 (a) Relationship between the speedup and the number of networks.
(b) Relationship between the efficiency and the number of networks. 49



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER 1

INTRODUCTION

1.1 The Problem Identification

A primary problem arising in molecular biology is how to determine the DNA sequence. Presently, genetic scientists have upgraded the process of DNA sequencing through computer technology [1, 2, 3]. DNA sequences are represented by a set of four basic nucleotide symbols, *A* (*Adenine*), *T* (*Thymine*), *C* (*Cytosine*), and *G* (*Guanine*), which can be obtained from a DNA sequencing machine. However, the sequencing techniques are not perfect, and, as a result, ambiguous symbols, “*N*” letter, are occasionally presented [4, 5, 6, 7]. This sequence may be complete or incomplete due to several uncontrollable factors such as the resolution of the gel degrades and the malfunction of sequencing machine. Several methods to manage errors from DNA sequencing are focused on the algorithm for solving DNA sequencing to get the sequence of symbols from genetic material and oligonucleotides [4, 5]. Although several existing DNA sequencing algorithms produce good results, they have not consider the problem of resolving the ambiguous nucleotide from the sequencing process. This research introduces a new technique to resolve the ambiguity using computational recognition based on the relative positions of all

nucleotides in the prefix sequence of an ambiguous nucleotide N .

1.2 The Objective and Scope

The goal of this research is or was to develop and validate an algorithm for predicting an ambiguous nucleotide in a DNA sequence. The study focuses or focused on a single ambiguous nucleotide in any region on a DNA sequence. The experimental data are or were DNA sequences of four available *Escherichia coli* genomes from EMBL database of European Bioinformatics Institute (EBI): *E.coli strain CFT073*, *strain K12*, *strain O157 : H7 EDL933*, and *strain O157 : H7 substrain RIMD 0509952*.

1.3 The Outline of the Thesis

The organization of this thesis comprised of the followings: Chapter 2 summarizes the related background of DNA sequences and the methodology of neural network, the machine learning. Chapter 3 reviews the DNA sequencing literatures and a related predicting method derived from an information theory. Chapter 4 introduces the proposed technique for predicting an ambiguous nucleotide by the relative position as a special technique for DNA analysis and explains an architecture of the recognition process to resolve the problem. Experimenting on the four *E.coli* genomes, the result correctness are described in Chapter 5. The conclusion of this thesis is in Chapter 6.

CHAPTER 2

BACKGROUND KNOWLEDGES

2.1 DNA Structure and DNA Sequences

In 1953, James Watson and Francis Crick first described the structure of DNA (Deoxyribonucleic acid). They found that DNA is a double-stranded molecule twisted into a helix. The double helix of the DNA is shown in Figure 2.1 along with details of how the bases, sugars and phosphates form the structure of the molecule.

2.1.1 Components of DNA

DNA is a polymer. The monomer units of DNA are nucleotides, and the polymer is known as a “polynucleotide.” Nucleotides are composed of a phosphate group, a sugar deoxyribose, and a nitrogenous base. Four different bases are commonly found in DNA: adenine (A), guanine (G), cytosine (C), and thymine (T). The four bases can be separated into two groups by their chemical structures. A and G have two rings like the compound purine and, therefore, they are called the purines or the purine bases. Other two bases, C and T, have one ring and are similar in structure to the compound pyrimidine and, because of that, they are

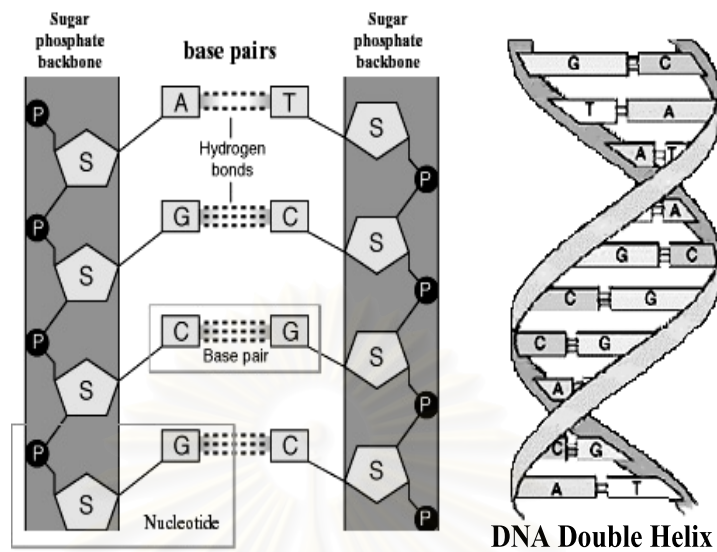


Figure 2.1: The Structure of Deoxyribonucleic Acid (DNA)

called the pyrimidines or the pyrimidine bases (Figure 2.2).

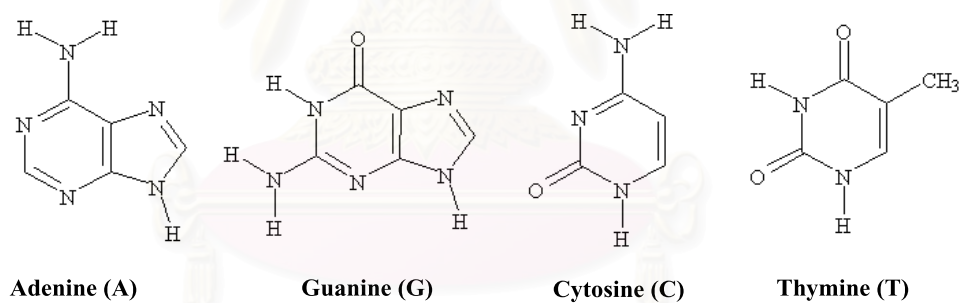


Figure 2.2: Structure of purine and pyrimidine bases

In their common structural configurations, the nitrogenous base is linked to the sugar with the 1' carbon. Phosphates and the sugar are arranged as a backbone with the 3' carbon of one sugar linked through a phosphodiester to the 5' carbon of the next sugar (Figures 2.1 and 2.3). The two strands of DNA are linked together

by hydrogen bonds between purine bases and pyrimidine bases. A and T form two hydrogen bonds while C and G form three hydrogen bonds. Because of the specificity of base pairing, the two strands of DNA are said to be complementary.

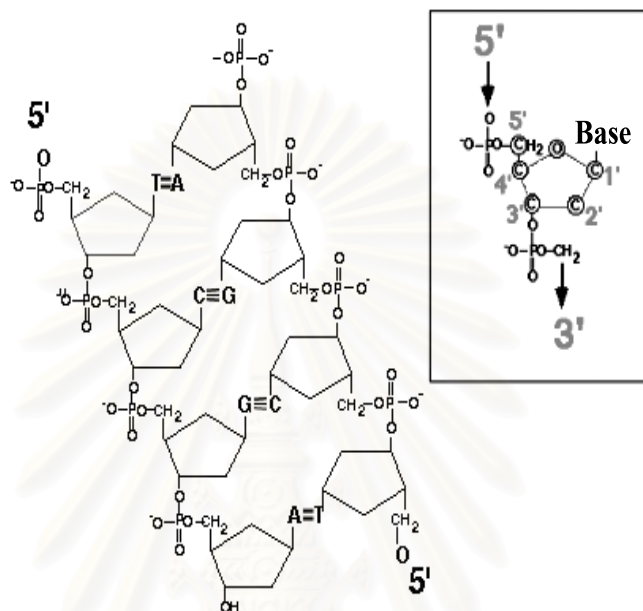


Figure 2.3: Two DNA strands form a double helix.

2.1.2 DNA Sequences

Considering both strands of DNA in Figure 2.4, we see that the two strands run in opposite directions in terms of the 5' and 3' ends. That is to say the strands run antiparallel to each other. Thus this chain in this example described from the 5' to 3' end would read C to G to G and so on. Another way to write this out is in a highly condensed structural formula 5'-CGGAGGACTGTCCT-3'. The sequence of the complementary strand (whose base order is 3'-GCCTCCTGACAGGA-5')

is shown here. Therefore, a DNA sequence can be considered as a sequence from the four-letter alphabet A, C, G, T and obtained from a DNA sequencing machine.

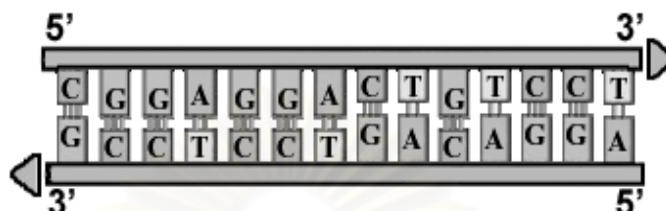


Figure 2.4: Double-stranded DNA.

2.2 Mutational Changes of DNA Sequences

Mutational changes in DNA can be classified by type of change caused by the mutational event into substitutions, deletions, insertions, inversion, and recombination. Further detail can be found on [8, 9]. Since the model of DNA sequence evolution which can account for biases in mutation rates that depend on the identity of neighboring nucleotides [10], the neighboring nucleotides has an influence on other nucleotides. In this research, we focused on substitution mutation to modify relative position of nucleotides.

The nucleotide substitutions can be divided into two classes: transitions (α) and transversions (β). A transition is the substitution of a purine (A or G) for another purine or the substitution of a pyrimidine (C or T) for another pyrimidine (Figure 2.5) [9, 11]. Other types of nucleotide substitutions are called transversions. The mutation matrix or scoring matrix can be divided in uniform mutation

rates ($\alpha = \beta$) or higher transitions than transversions ($\alpha > \beta$). For instance, the scoring matrix with 1% probability of change at each position is shown as in Tables 2.1 and 2.2 [9].

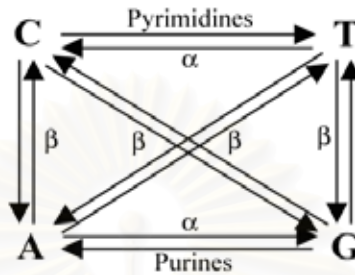


Figure 2.5: Nucleic acid substitutions diagram.

Table 2.1: Uniform mutation rates with 1% probability of change at each position.

	A	C	T	G
A	0.99	0.00333	0.00333	0.00333
C	0.00333	0.99	0.00333	0.00333
G	0.00333	0.00333	0.99	0.00333
T	0.00333	0.00333	0.00333	0.99

The mutation matrix with higher transitions (α) than transversions (β) are commonly used to apply the realistic biological data [9, 12]. We applied the mutation matrix in Table 2.2 to our work discussed in Chapter 4.

Table 2.2: Higher transitions (α) than transversions(β) with 1% probability of change at each position.

	A	C	G	T
A	0.99	0.002	0.006	0.002
C	0.002	0.99	0.002	0.006
G	0.006	0.002	0.99	0.002
T	0.002	0.006	0.002	0.99

2.3 An Overview of Artificial Neural Networks

Artificial neural networks were originally designed to model in some small way the functionality of the biological neural networks which are a part of the human brain. Our brains contain many neurons. Each biological neuron consists of a cell body, a collection of dendrites which bring electrochemical information into the cell and an axon which transmits electrochemical information out of the cell (Figure 2.6). A neuron produces an output along its axon so it fires when the collective effect of its inputs reaches a certain threshold. The axon from one neuron can influence the dendrites of another neuron across junctions called synapses. Some synapses will generate a positive effect in the dendrite, ie one which encourages its neuron to fire, and others will produce a negative effect, ie one which discourages the neuron from firing. A single neuron receives inputs from several synapses. It is still not clear exactly how our brains learn and remember but it appears to be associated with the interconnections between the neurons at the synapses.

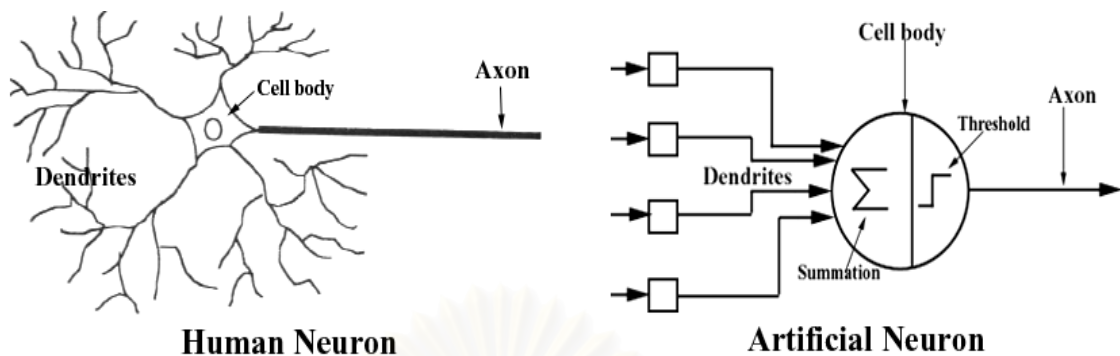


Figure 2.6: Analogy between human and artificial neuron.

Artificial neural networks try to model this low level functionality of the brain. This contrasts with high level symbolic reasoning in artificial intelligence which tries to model the high level reasoning processes of the brain. The neural networks can contain many artificial neurons comparing biological neurons in Figure 2.6. An artificial neuron consists of a processing element which has a number of input connections, each with an associated weight, a transfer function (or a threshold function) which determines the output, given the weighted sum of the inputs, and the output connection itself. An artificial neural network is a network of interconnected neurons. The network may be trained by adjusting the weights associated with the connections in the net to try and obtain the required outputs for given inputs from a training set. This is an analogy between biological (human brain) and artificial neural networks.

2.4 Backpropagation Neural Network

Artificial neural networks with backpropagation [13] are currently and mostly used as a classifier or a recognizer. It successfully performs a variety of input-output mapping tasks for recognition, generalization, and classification [14]. The basis model of a neural network consists of three parts:

- Nodes connected by links organized in layers by weights. There are three layers - input layer, hidden layer, and output layer - from a sample network in Figure 2.7. The total nodes of the constructed neural network are equal to the summation of the number of input nodes, hidden nodes (or hidden neurons), and output nodes (or output neurons).
- The output o_i of neuron i is computed by an activation function known as logistic function of activation values of neuron i . The activation values of neuron i is computed as the dot product between input vector, $\mathbf{x} = [x_1, \dots, x_q]^T$, and weight vector, $\mathbf{w}_i = [w_{i,1}, \dots, w_{i,q}]^T$. Therefore, the output o_i computed as follow.

$$o_i = f_{act}\left(\sum_{j=1}^q w_{i,j}x_j\right) \quad (2.1)$$

In figure 2.7, each layer has a synaptic weight matrix associated with all the connections made from the previous layer to the next layer, that is, $\mathbf{W}^{(\ell)}$, for $\ell = 1, 2$. The first layer has the weight matrix $\mathbf{W}^{(1)} = [w_{ji}^{(1)}] \in \mathfrak{R}^{h \times q}$, the second layer's weight matrix is $\mathbf{W}^{(2)} = [w_{rj}^{(2)}] \in \mathfrak{R}^{m \times h}$, for $i = 1, 2, \dots, q; j = 1, 2, \dots, h; \text{ and } r = 1, 2, \dots, m; .$ The nonlinear input-output mapping $\Omega : \mathfrak{R}^{q \times 1} \rightarrow \mathfrak{R}^{q \times 1}$ can be determined directly from Figure 2.7 as follows.

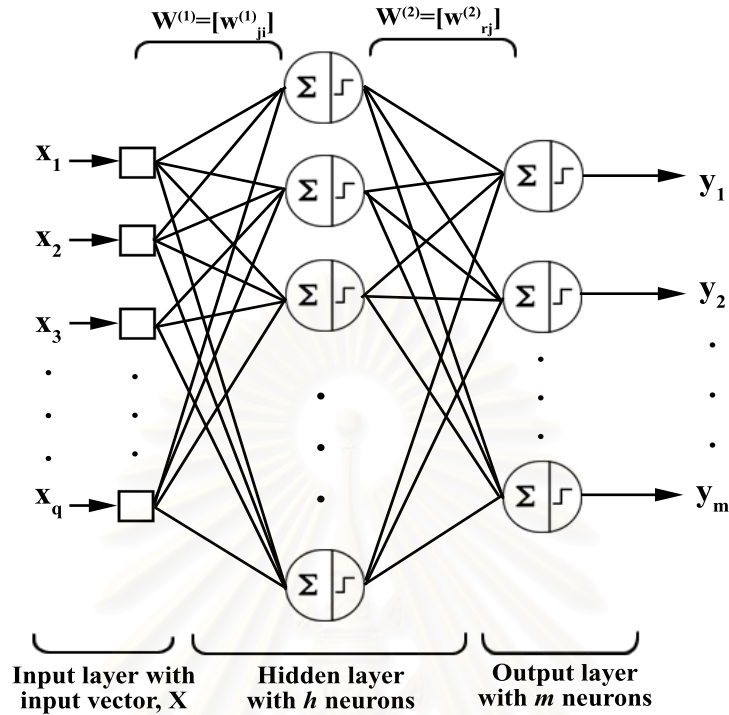


Figure 2.7: Illustrating the sample network that has only one input layer, one hidden layer, and one output layer. There are q nodes represented input features in the input layer, many hidden neurons we can adjust for an applicable neural network, and there are any output neurons.

Therefore, we define that $f_{act}^{(1)}$ is the nonlinear activation function in the hidden layer and that $f_{act}^{(2)}$ is the nonlinear activation function in the output layer. The output of hidden neuron r , $o_r^{(1)}$, and the output of output neuron s , $o_s^{(2)}$, then computed as

$$o_r^{(1)} = f_{act}^{(1)}\left(\sum_{j=1}^q w_{r,j}^{(1)} x_j\right) \quad \text{and} \quad o_s^{(2)} = f_{act}^{(2)}\left(\sum_{j=1}^h w_{s,j}^{(2)} o_j^{(1)}\right) .$$

where $r = 1, 2, \dots, h$, $s = 1, 2, \dots, m$, q is the number of features used to form input vectors, h is the number of hidden nodes, and m is the number of output neurons.

We define the output of output neuron s , $o_s^{(2)}$, is y_s where $s = 1, 2, \dots, m$. In general, there are many input vectors and all input vectors can be captured in forms of a matrix of size $p \times q$ where p is the total number of input vectors (or patterns). Each input vector has its target.

- The output of output neuron s for the μ -th input vector (or pattern), $y_s^{(\mu)}$, should be equal to the target $t_s^{(\mu)}$. The learning performance is measured by the following cost function.

$$E = \frac{1}{p} \frac{1}{2} \sum_{\mu=1}^p \sum_{s=1}^m (t_s^{(\mu)} - y_s^{(\mu)})^2$$

To minimize the cost function, the weight in iteration k can be updated by the learning rule following

$$\Delta w_{i,j}(k) = -\eta \delta_i(k) o_i(k) + \alpha \Delta w_{i,j}(k-1)$$

where η is the learning rate, δ_i is the error for any node and α is the momentum constant. Weight adjusting are discussed in a backpropagation algorithm.

Training a neural network with backpropagation algorithms results in a non-linear mapping or an association task. Thus, given two sets of data, the neural network can have its synaptic weights adjusted by backpropagation algorithm to develop a specific nonlinear mapping. The neural network, with fixed weights

after the training process, can provide an association task for classification, pattern recognition, diagnosis, etc. During the training phase of the neural network, the synaptic weights are adjusted to minimize the disparity between the actual and desired outputs of the neural network, averaged over all input patterns (or learning examples).

Standard backpropagation algorithm

Weight Initialization

Each weight is initialized to a small random value.

Calculation of activation function

1. The activation level of an input unit is determined and fed to the network.
2. The activation level O_j of a hidden and output unit is determined by

$$O_j = F\left(\sum W_{ji}O_i + \theta_j\right) \quad (2.2)$$

where W_{ji} is the weight of an input O_i , θ_j is the node threshold, and F is the activation function.

Weight Training

1. Start at the output units and work backward to the hidden layers recursively to adjust the weights by

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji} \quad (2.3)$$

where $W_{ji}(t)$ is the weight from unit i to unit j at time t (or t the iteration) and ΔW_{ji} is the weight adjustment.

2. The weight change is computed by

$$\Delta W_{ji} = \eta \delta_j O_i \quad (2.4)$$

where η is a trial-independent learning rate ($0 < \eta < 1$, e.g., 0.3) and δ_j is the error gradient at unit j . Convergence is sometimes faster by adding a momentum term:

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_j O_i + \alpha [W_{ji}(t) - W_{ji}(t-1)] \quad (2.5)$$

where $0 < \alpha < 1$

3. The error gradient is given by:

- For the output units:

$$\delta_j = (T_j - O_j) F'(net_j)$$

where T_j is the desired (target) output activation, O_j is the actual output activation at output unit j and $net_j = \sum_i W_{ji} O_i$.

- For the hidden unit

$$\delta_j = F'(net_j) \sum_k \delta_k W_{kj}$$

where unit j is a hidden unit, δ_k is the error gradient at unit k to which a connection points from hidden unit j .

4. Repeat iterations until convergence in terms of the selected error criterion or a maximum number of iterations is reached.

The training set is presented iteratively to operate the network, whereby the weights are updated until their values become stabilized according to the following criterion: (1) a user-defined error tolerance is achieved, or (2) a maximum number of iterations is reached.

All input patterns (or vectors) can be captured in forms of a matrix of size $p \times q$. The computational time for solving the value of each weight depends on the size of this matrix which is constrained by the values of p and q , where p is the number of input patterns (or vectors) and q is the dimensions of an input pattern or the number of input nodes. In addition, the number of hidden neurons are also important. Too fewer hidden neurons are not enough to provide acceptable classification. Whereas too many neurons not only spend more training time to complete the network but also generate the problem of over-fitting the training data. Specifying the number of hidden neurons prior to the training process is not easy. The only feasible approach to speed up the training process is by partitioning the data sets into several smaller data sets and training each small data set with one individual network. The detail of this partition technique is explained in Chapter 4.

CHAPTER 3

LITERATURE REVIEWS

3.1 The Review of Literature Related to DNA Sequencing Problems

Two primary methods of rapid DNA sequencing were developed in 1977, which could be considered the year that the Human Genome Project was born. Two groups of researchers, Maxam and Gilbert and Sanger et al, published seminal papers describing the development of DNA sequencing technologies. Maxam and Gilbert described a method of chemically cleaving DNA that had been terminally labeled [15] while the Sanger paper described a sequencing method using chain terminators [16]. Both methods generate radiolabeled DNA fragments that initiate at a defined point and are random in length but end with a defined type of base either *A*, *T*, *C*, or *G*. The random populations of DNA fragments are then separated on polyacrylamide gels according to size. A high resolution gel is used to separate fragments that differ in as little as one base. Depending on the type and percentage of gel it is possible to read almost 500-600 bases of sequence from a single sample loading. However, the actual number of base pairs in the genome is longer than that; for example, a human genome consists of approximately 3

billion nucleotides. Genetic scientists have upgraded the process of DNA sequencing through computer technology. The DNA sequencing consists of determining a sequence of nucleotides of an examined DNA fragment, cut out from a genome by restriction enzymes or by the shotgun approach. Almost all large scale sequencing projects employ the shotgun sequencing strategy that assembles the target DNA sequence from a set of short DNA fragments determined from a set of DNA pieces randomly sampled from the target sequence [1, 17]. Genome assembly is the job of computer programs known as “assemblers.” These programs work by finding and analyzing overlaps, or identical DNA fragments at either end of two different reads. Eventually, we can get a DNA sequence represented by a set of the four basic nucleotide symbols from a DNA sequencing machine. But this sequence may be complete or incomplete. The errors can occur more and more frequently as the resolution of the gel degrades or genetic material containing allelic mixtures [4, 6, 18]. An incomplete sequence refers to a sequence that some nucleotide symbols are ambiguous. Table 3.1 denotes the nomenclature for incompletely specified symbols in DNA sequences provided by IUPAC and IUBMB [7].

An automatic sequencing machine produces what genome scientists call “raw” sequence. In raw sequence, the reads or short DNA sequences are all jumbled together, like the pieces of a jigsaw puzzle in a just-opened box. Inevitably, raw sequence also contains a few gaps, mistakes, and ambiguities. There is no mechanical substitute for the intuition and intelligence of an experienced finisher, so finishing is currently a bottleneck in the process of DNA sequencing. Automatic sequencing machines can churn out a raw sequence much faster than humans can analyze and polish these sequences.

Table 3.1: Summary of single-letter code recommendations.

Symbol	Meaning	Origin of designation
G	G	Guanine
A	A	Adenine
T	T	Thymine
C	C	Cytosine
R	G or A	puRine
Y	T or C	pYrimidine
M	A or C	aMino
K	G or T	Keto
S	G or C	Strong interaction (3 H bonds)
W	A or T	Weak interaction (2 H bonds)
H	A or C or T	not-G, H follows G in the alphabet
B	G or T or C	not-A, B follows A
V	G or C or A	not-T (not-U), V follows U
D	G or A or T	not-C, D follows C
N	G or A or T or C	aNy Nucleotide

To correct the ambiguous nucleotide from DNA sequencing in the present, there are many following:

- Many of these errors and ambiguities can be resolved by inspection of the traces [4]. In addition, an assembler software compares all the different reads that cover the same stretch of DNA and generates what is known as

a “consensus” sequence. For example, if a certain base comes out as an A nine times and C the tenth, then chances are the base is really an A. An assembler is designed to sift through conflicting information and decide which sequence is likely to be right.

- One method/strategy for eliminating errors is by sequencing the genome more than once. That is to say, we can reconduct the DNA sequencing experiment or sequencing the complementary of DNA sequence is a good way to resolve an ambiguous nucleotide on DNA sequences but may provide an ambiguous one on other positions again.
- Although computer programs can help resolve gaps and uncertainties in a genome sequence, much of the final polishing is still done by people known as finishers. These expert workers identify gaps in the sequence, design experiments to fill in those gaps, and determine how to collect any additional information that are necessary.

In 1996, Tao Jiang and Ming Li tried to model the DNA sequencing problem as learning a string [19] from its randomly drawn substrings and approximated a shortest common superstring of a set of strings. Under certain restrictions, this may be viewed as string learning in Valliant’s distribution-free learning model.

In 2002, Jacek Blazewicz, Piotr Formanowicz, Frederic Guinand and Marta Kasprzak introduced a heuristic technique managing errors for DNA sequencing by hybridization problem [5]. The new method for rebuilding sequences from a set of oligonucleotides is simple and fast. Negative error refers to some missing

oligonucleotides in the spectrum, and positive error refers to erroneous oligonucleotides. If the coordinates of a point on the chip are not correctly read, two errors appear simultaneously: a negative one and a positive one.

In 2003, Izydor Apostol, Philippe Jacquet, and Wojciech Szpankowski introduced a novel prediction algorithm called Sampled Pattern Matching (SPM) predictor [20] that was recently developed in a universal predictor based on pattern matching [21] to analyze molecular sequences. They have an information theory told us biological sequences can be predicted.

In 2004, Pawel Gajer, Michael Schatz and Steven L. Salzberg constructed an automated correction of genome sequence errors by using information from an assembly of a genome. The new program called AutoEditor [4] significantly improves base calling accuracy over that achieved by previous algorithms. This in turn improves the overall accuracy of genome sequences and facilitates the use of these sequences for polymorphism discovery.

From above points, we can assume that nucleotides and their position may be related to their neighboring nucleotides. Therefore, we considered DNA sequences to be the string of letter A, T, C, and G, and applied them to character based prediction problem on an information theory and recognition problem on an artificial neural network. Some interesting information theory applied to biological sequence are explained in the next section.

3.2 The Review of Literature Related to SPM Predictor Derived from Information Theory

The SPM predictor described by Izydor Apostol, Philippe Jacquet, and Wojciech Szpankowski is a special case of a universal predictor [21] based on pattern matching to analyze biological sequences. By the algorithm of the SPM predictor, it is assumed that a biological sequence $x_n = x_1, x_2, \dots, x_n$ is given. Each symbol x_i belongs to a finite alphabet Σ . For a fixed integer $K \geq 1$, the algorithm will predict the next K symbols, that is, $(x_{n+1}, \dots, x_{n+k})$.

Let $0 < \alpha < 1$, the SPM prediction algorithm works as follows:

Step1: Find the largest suffix of x_n whose copy appears somewhere in the string x_n . We denote this suffix by D_n of length l the largest integer such that

$$(x_{n-l+1}, \dots, x_n) = (x_{n-i-l+1}, \dots, x_{n-i}) \text{ for some } 1 \leq i \leq n$$

Step2: Take an α fraction of the maximal suffix of length

$$d_n := [\alpha D_n]$$

That is the suffix x_{n-d_n+1}, \dots, x_n . A fractional suffix defines a marker (i.e., a substring), and the k positions after markers are called the k -tuple marked positions.

Step3: Let now $N(x_1, \dots, x_k)$ be the number of non-overlapping k -tuple (x_1, \dots, x_k) occurrences in the sampled sequence. The SPM predictor assigns

$$(x_{n+1}, \dots, x_{n+k}) = \arg \max N(x_1, \dots, x_k)$$

In words, $(x_{n+1}, \dots, x_{n+k})$ is assigned to the most frequent k -tuple occurring in the sampled sequence.

Consider this example. SPM Predictor for $k = 1$ is presented as a text string with the largest suffix (the bold fragments) and its copy framed (defined in Step 1 of the above algorithm):

SLJZGGDLYGSJ**SLJZ**KGSS**SLJZ**KLJZJGZYGSJ**SLJZ**

In fact, $D_{40} = 8$. Let $\alpha = 0.5$. Then, the fractional suffix SLJZ is used to find all markers. They are shown below:

SLJZGGDLYGSJ**SLJZ**KGSS**SLJZ**KLJZJGZYGSJ**SLJZ**

The sampled sequence is **GKK**, thus the SPM predicts $x_{41} = \mathbf{K}$.

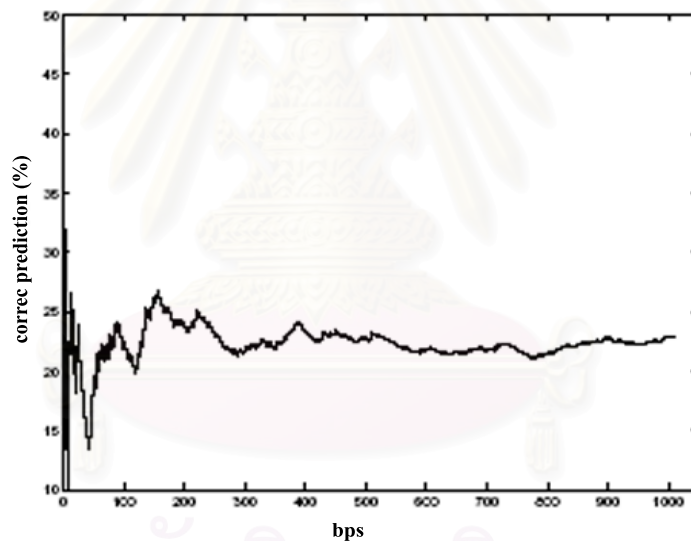
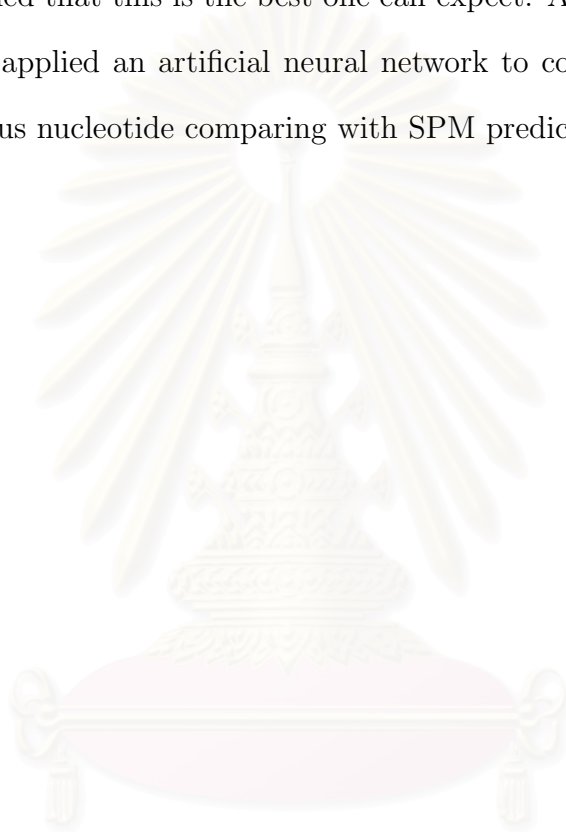


Figure 3.1: The percentage of correct prediction using SPM for *E.coli* genome with noncoding and coding region.

Their SPM predictor on biological sequences implied that the biological sequences can be predicted. The result (Figure 3.1) of their SPM algorithm to a

segment of about 1600 bps length of E.coli genome that runs across noncoding and coding segments showed a change in predictability around 150th base which agree with the transition from noncoding to coding regions. Although this seems to be an unimpressive correction, there exist an information theory (Theorem 1 in [21]) confirmed that this is the best one can expect. According to this implicit prediction, we applied an artificial neural network to construct a predictor of a single ambiguous nucleotide comparing with SPM predictor of those.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER 4

A PROPOSED TECHNIQUE FOR PREDICTING AMBIGUOUS NUCLEOTIDE SYMBOL

4.1 Problem Formulation

To resolve the ambiguous symbol, N , in a given DNA sequence, we formulate the problem as a recognition problem of a given nucleotide string. Let $\Sigma = \{A, T, C, G\}$ be a set of nucleotide symbols and $N \in \Sigma$ an ambiguous symbol. Our problem is defined as follows.

Problem: Given a nucleotide string $S = (A + T + C + G)^n N$, where $n > 0$ is an integer, resolve the actual symbol of N .

$(A + T + C + G)$ means a string, s , consisting of either symbol A , or T , or C , or G . Concatenating string s with itself n times is written as s^n . The value of n depends upon the species. String $(A + T + C + G)^n$ of length n is the prefix string of N .

4.2 Relative Positions

Our assumptions are that the actual symbol of N must depend upon the other nucleotides in string S and the position of each nucleotide has the direct influence to the occurrence of the actual symbol of N . In order to resolve the symbol of N , the feature of string S must be extracted based on our assumptions prior to the recognition process. Let b_i be nucleotide symbol $b \in \Sigma$ at position i in string S . The sequence of positions can be counted from either left to right or from right to left. The relative position, r_{b_i, b_j} , between two nearest identical nucleotide symbols b_i and b_j in string S is defined as

$$r_{b_i, b_j} = j - i; \quad j > i. \quad (4.1)$$

For example, consider this given nucleotide sequence.

symbol:	C	T	C	G	A	T	C	G	A	C	T
position:	70	71	72	73	74	75	76	77	78	79	80

There are only two nucleotides A between positions 70 and 80. Thus, the value of $r_{A_{74}, A_{78}}$ is equal to 4. To analyze realistic biological data, distance or position, between pairs of symbols may be based on the δ -function defined in Σ . We can rewrite the relative position in equation (4.1) in a new formular by adding information of mutation rate [9, 12] of δ -function.

$$\delta(b_i, b_j) = \begin{cases} 1 & \text{if } b_i = b_j \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

In this case, the symbols occur independently on their positions with equal probability. Instead of using r_{b_i, b_j} as in equation (4.1), the value of r_{b_i, b_j} is replaced by

the proposed relative probabilistic position as in equation (4.3).

$$r_{b_i, b_j} = 1 + \sum_{k=i+1}^j |1 - \delta(b_i, b_k)|; \quad j > i \quad (4.3)$$

However, the real-valued matrices or ratios in the mutation case is capable of representing the frequently using symbols in a sequence.

4.2.1 Extended Relative Position with Probability of Mutation Rate

We defined a mutation-rate function σ in $\Sigma \times \Sigma$ as a scoring matrix. This scoring matrix depend on the assigned probability of mutation rate. Obviously, δ -function is a special case of σ -function when $\sigma_{b_i b_j}$ equals $\delta(b_i, b_j)$. So it is the identity matrix. The relative probabilistic position with mutation rates becomes

$$r_{b_i, b_j} = 1 + \sum_{k=i+1}^j |1 - \sigma_{b_i b_k}|; \quad j > i \quad (4.4)$$

The scoring matrix obtained from the assigned probability of mutation rates by changing any nucleotides at each position ($\sigma_{b_i b_j}$ values from Tables 2.1 or 2.2 in Chapter 2).

4.3 Recognition Process

4.3.1 Input Representation

The feature vector of string S , \mathbf{v}_S , consists of four sets of relative positions of each nucleotide in Σ . These four sets are concatenated to form a feature vector.

Since the number of relative positions of different nucleotides in string S may not be equal, according to the amount of each nucleotide, it would be better to equally set the number of relative positions of each nucleotide. Suppose k is the number of relative positions. The feature vector, \mathbf{v}_S , of string S is formed by concatenating the inversely relative positions for each nucleotide as shown in the following format.

$$\mathbf{v}_S = \left(\prod_{\substack{i,j \in P_A \\ j > i}} \frac{1}{r_{A_i, A_j}} \right) \left(\prod_{\substack{i,j \in P_T \\ j > i}} \frac{1}{r_{T_i, T_j}} \right) \left(\prod_{\substack{i,j \in P_C \\ j > i}} \frac{1}{r_{C_i, C_j}} \right) \left(\prod_{\substack{i,j \in P_G \\ j > i}} \frac{1}{r_{G_i, G_j}} \right) \quad (4.5)$$

where P_A, P_T, P_C, P_G are the set of positions having nucleotides A, T, C , and G , respectively, in string S . $|P_A| = |P_T| = |P_C| = |P_G| = k+1$. The symbol Π means concatenating. The inversely relative position is used to handle the significance of the neighboring nucleotides of symbol N .

Table 4.1: Example of nucleotide sequence.

symbols:	A	A	T	C	C	T	G	G	A	G	T	G	C	C	T	C
position:	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55

For example, consider the given sequence S in Table 4.1. The selected position is at position 55, where symbol C appears. Suppose that it is an ambiguous nucleotide and can be represented A or T or C or G . Thus, the target of this given sequence is C . If k is equal to 3 then the feature vector with relative-position equation in (4.1) or (4.3) is

$$\begin{aligned}
\mathbf{v}_S &= \begin{array}{c} \text{A} \\ \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{1} \end{array} \left| \begin{array}{c} \text{T} \\ \frac{1}{1} \quad \frac{1}{4} \quad \frac{1}{5} \end{array} \right| \begin{array}{c} \text{C} \\ \frac{1}{2} \quad \frac{1}{1} \quad \frac{1}{8} \end{array} \left| \begin{array}{c} \text{G} \\ \frac{1}{4} \quad \frac{1}{2} \quad \frac{1}{2} \end{array} \right. \\
&= \begin{array}{ccc|ccc} 0.143 & 0.143 & 1 & 1 & 0.25 & 0.2 \\ \hline & & & 0.5 & 1 & 0.125 \end{array} \left| \begin{array}{ccc} 0.25 & 0.5 & 0.5 \end{array} \right.
\end{aligned}$$

Thus, the feature vector is used as the input representation into the backpropagation network for each sequence. However, if the length of a DNA sequences is selected within fixed window size then this length may create a problem of some missing relative positions. Given a number of relative positions k , it is possible that the number of relative positions of some nucleotide symbols may be less than k . The corresponding feature vector cannot be completely formed. For this situation, any missing relative position is, then, set to the large value because this feature value means the distance between the same type of nucleotides is too far to consider. Obviously, its inversely relative position is closed to 0 that means less significant. The example of input representations for training the neural network are displayed in Table 4.2 with $k = 3$. The appropriate number of relative positions to form a feature vector is set by an acceptable tolerance (τ). The tolerance is a threshold for the error ratio of the number of duplicated vectors being in different classes to the number of unduplicated vectors as data set. For example, suppose that our original data set contains 10,000 feature vectors with a relative number $k = c$, where c is a constant. Approximately 60% of these vectors are duplicated in the same class and only 15 vectors are duplicated in difference classes. So, there are 3,985 remaining vectors as a new data set for being training set. The error ratio is, then, $\frac{15}{3985} \approx 0.00376$. If the tolerance is equal to 0.005 that means the percentage of covering information of data set is up to 99.5 percent, then $k = c$

in this example is an appropriate number of relative positions for these data set. Therefore, the appropriate number of relative positions are the number of those that can extract enough features of prefix sequences covering mostly information of data set and having the least complexity. Because of increasing the number of relative positions, the number of features are also increased. The size of input pattern matrix is very large and complex. Consequently, there are four neural networks for all nucleotides, A , C , G , and T , in the recognition process.

Table 4.2: The feature vectors of the training patterns in Table 4.1.

Target	Position	Input representation											
A	48	$\frac{1}{7}$	$\frac{1}{1}$	0	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{4}$	$\frac{1}{1}$	0	$\frac{1}{1}$	$\frac{1}{1}$	0
G	51	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{1}$	$\frac{1}{1}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{1}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{1}$
T	54	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{1}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{1}$	$\frac{1}{1}$	$\frac{1}{8}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$
C	55	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{1}$	$\frac{1}{1}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{1}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{2}$

4.3.2 Partitioning Training Feature Vectors

In Chapter 3, we introduced the problem of training large set of input vectors (known as feature vectors) by a neural network. For fast training time [22, 23],

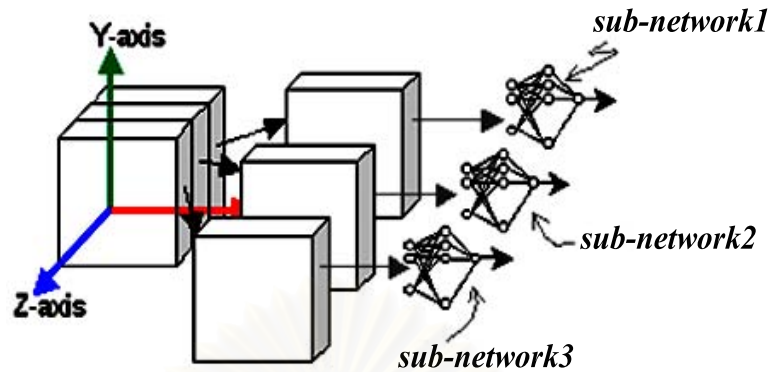


Figure 4.1: Suppose feature vectors have three dimensions and the third feature in Z -axis is the key feature. Therefore, all feature vectors are partitioned into three groups denoted by three blocks. Each group of feature vectors is trained by an individual neural network.

the data sets must be partitioned into several smaller data sets and trained with one individual network in parallel. In this section, we considered the feature vectors as the input vectors for neural networks. Each feature vector discussed in the previous section can be viewed as a vector in q dimensions. Thus, the value of each feature x_i is the location in the i^{th} dimension. Consider the example shown in Figure 4.1. Here, all feature vectors are in a 3-dimensional space comprising of three basic axes, X -axis, Y -axis, and Z -axis. Suppose that Z -axis is the key dimension. All feature vectors are, then, partitioned into three groups based on the values in this dimension. Each partitioned group in Figure 4.1 is denoted by a block sliced along the Z -axis and is separately trained by an individual neural network. The partitioning process consists of the following main steps.

Step1: Partitioning Feature Vectors By Feature Intervals

Consider all the values of all q features of each feature vector. They must lie in any ranges. Their ranges can be divided into sub-intervals of equal width. Suppose all values lie in range $[a, b]$ and they are divided into n intervals. So the width of each interval is $d = (b - a)/n$. The n intervals are $[a, a + d), [a + d, a + 2d), \dots, [a + (i - 1)d, a + id), \dots, [a + (n - 1)d, a + nd = b]$. We define the interval $[a + (i - 1)d, a + id]$ as the i -th interval. An interval having the maximum number of feature values is called an important interval, i_{imp} . For example, consider the patterns shown in Figure 4.2. There are 12 patterns. Each pattern has four features. The important interval is $[2, 3)$ because it contains 14 feature values which is maximum. Let $N_{value}(i)$ be the number of feature values lying in the i -th interval, $[a + (i - 1)d, a + id)$, the important interval is set to

$$i_{imp} = \operatorname{argmax}_i \{N_{value}(i)\},$$

where $i = 1, 2, \dots, n$.

Step2: Selecting Key Feature

The key feature (f_{key}) of each training set is a marker for partitioning the feature vectors into many groups. Considering an important interval, i_{imp} , we can count the number of feature vectors whose feature values of each feature column lay in the i_{imp} -th interval (see Figure 4.2 for an example). Then, the feature column with the minimum number of feature vectors is selected as the key feature, f_{key} . Suppose $N_{pattern}(i, k)$ is the numbers of feature vectors whose feature values of the k -th column lay in the i -th interval. Therefore, the key feature is defined as

$$f_{key} = \operatorname{argmin}_k \{N_{pattern}(i_{imp}, k)\},$$

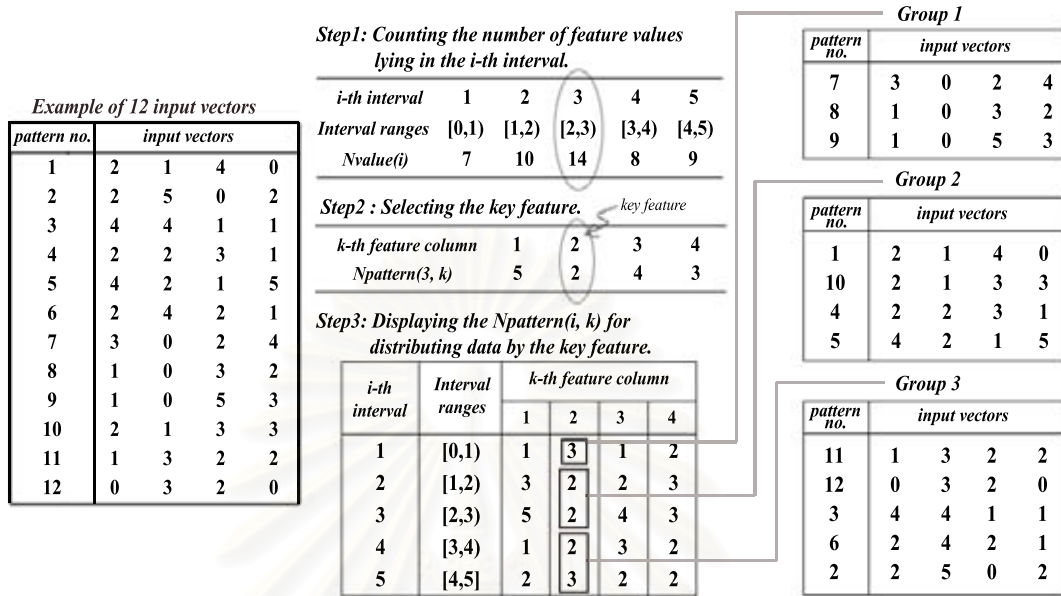


Figure 4.2: An example of how input vectors (or feature vectors) are partitioned. There are 12 feature vectors whose feature values lay in the range $[0, 5]$. These feature values are divided into five ($n = 5$) intervals. The width of each interval is set to $d = \frac{5-0}{5} = 1$. Step 1 shows the number of values, $N_{value(i)}$, lying in the i -th interval. The important interval is the third interval because most of the values lie in it. Step 2 shows the number of feature vectors, $N_{pattern(3, k)}$, having their feature values lying in the third interval, which is the important interval. The second feature was used as the key feature to divide all feature vectors. Step 3 shows three groups of feature vectors after partitioning. For this example, the number of networks is set to 3. Each groups must contains less than or equal to $avg = \frac{12}{3} = 4$ patterns. Thus, the last group contains the patterns in the remaining intervals

where $k = 1, 2, \dots, q$. This f_{key} is the index of the f_{key} -th feature of all feature vectors. The f_{key} for the example in Figure 4.2 is the second feature since there are only two feature values lying in the important interval $[2, 3)$.

Step3: Partitioning Feature Vectors By Key Feature

The average number of feature vectors to be partitioned in each group is computed by this fraction.

$$avg = \frac{\text{Total numbers of feature vectors}}{\text{Number of networks}}$$

The number of networks must be set prior to partitioning process. This number indicates how many neural networks are required to be trained in parallel fashion and also indicates how many blocks of partitioned data are supplied to these networks. It is necessary to set the average number small enough to possibly train the network. The recommended average number is approximated to 2,000-3,000 patterns. The value of avg is just a guideline for partitioning feature vectors into blocks. The size of some groups may be less or greater than the value of avg . After knowing all $N_{value}(i)$, for $1 \leq i \leq n$, and the f_{key} , the number of feature values in the interval corresponding to the value of each $N_{value}(i)$ of the f_{key} -th feature column is counted. The feature vectors are gradually partitioned in groups by using the number of feature values of f_{key} feature column and the value of avg . From the example in Figure 4.2, there are 12 feature vectors and four features. Suppose the number of networks is set to 3. Therefore, the value of avg must be equal to $12/3$ or 4. The important interval is $[2, 3)$ and f_{key} is set to 2, which is the second feature column. In the second feature column, the number of feature values in each interval is: $[0, 1)$ has three feature values; $[1, 2)$ has two feature

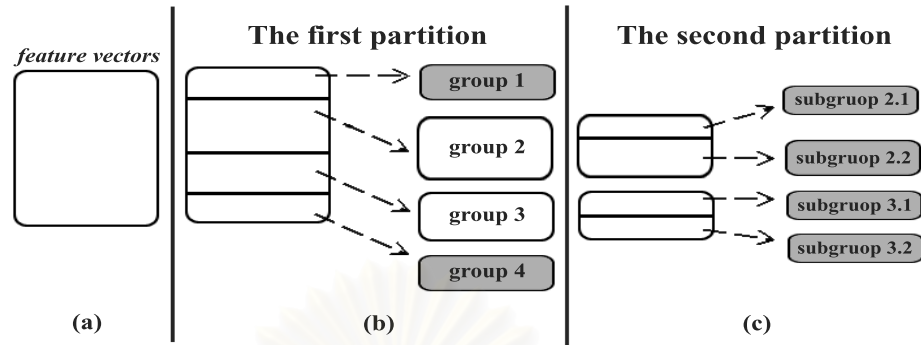


Figure 4.3: An example of how to recursively apply the partitioning concept. Each block denotes a group of feature vectors. (a) The original given feature vectors to be partitioned. (b) The given feature vectors in (a) are partitioned into four groups. (c) Groups 2 and 3 in (b) are further partitioned into four subgroups.

values; $[2, 3)$ has two feature values; $[3, 4)$ has two feature values; $[4, 5]$ has three feature values. With these numbers, all feature vectors are partitioned as follows. The first group contains all feature vectors whose second feature values are in $[0, 1)$. The second group contains all feature vectors whose second values are in $[1, 2)$ and $[2, 3)$. The third group contains all feature vectors in $[3, 4)$, and $[4, 5]$.

However, if any group of feature vectors cannot achieve the desired speed then the partitioning concept can be recursively applied to that group. Figure 4.3 shows an example of how to recursively partition the feature vectors. The given feature vectors are in Figure 4.3(a). The first partition consisting of four groups is shown in Figure 4.3(b). Suppose groups 2 and 3 cannot achieve the desired speed. These two groups are further partitioned into four subgroups.

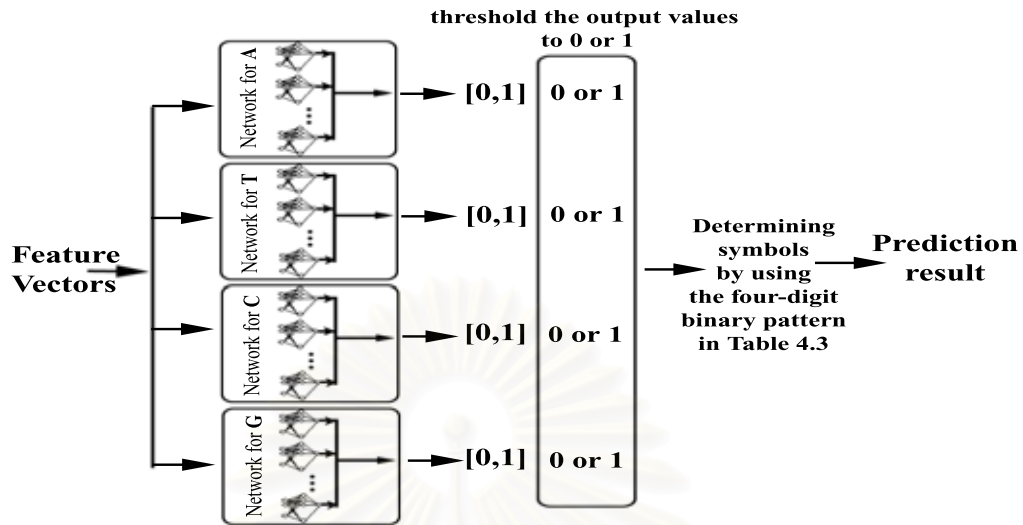


Figure 4.4: Network Architecture and the determination by thresholding the outputs from all networks and comparing them with the patterns in Table 4.3.

4.3.3 Network Architecture

The problem of resolving the actual symbol N is transformed to the problem of recognizing all prefix patterns of symbol N corresponding to each actual symbol in $\Sigma = \{A, T, C, G\}$. Therefore, our network architecture consists of four main recognition networks for each symbol in Σ . Each network composes of several sub-networks for performing parallel training as discussed in the previous section. The pattern recognition process is implemented by using many backpropagation networks in place of sub-networks of each network type. There is one output unit for two targets, as 1 refers to the feature vector can be recognized to be this type of the recognition network and 0 cannot be recognized. Each actual symbol is resolved by one recognition network with one-against-all recognition

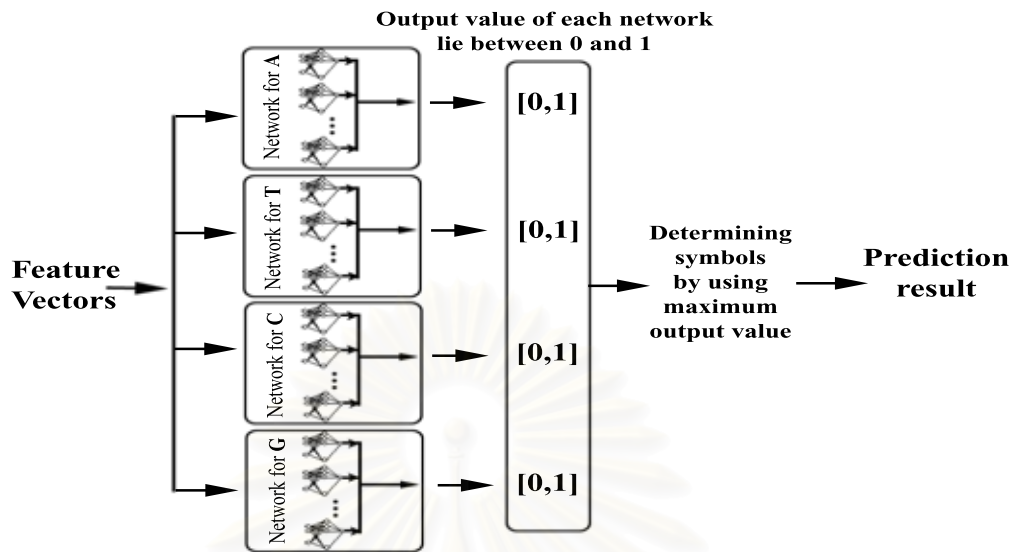


Figure 4.5: Network Architecture and the determination by selecting maximum output values.

scheme. The outputs from all recognition networks are simultaneously considered to determine the final answer. Two possible determinations are implemented. The first approach is by thresholding the output value of each recognition network to 0 or 1 (Figure 4.4). The outputs from four recognition networks are compared with the four-digit binary patterns in Table 4.3 to resolve the symbol. The second approach is to determine the final answer by selecting the maximum output value (Figure 4.5).

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Table 4.3: A table of possible output patterns and the resolved symbols.

Output Pattern from Each Network				Resolved Symbol
Network G	Network C	Network T	Network A	
0	0	0	0	not applicable
0	0	0	1	A
0	0	1	0	T
0	0	1	1	W (A or T)
0	1	0	0	C
0	1	0	1	M (A or C)
0	1	1	0	Y (T or C)
0	1	1	1	H (A or C or T)
1	0	0	0	G
1	0	0	1	R (G or A)
1	0	1	0	K (G or T)
1	0	1	1	D (G or A or T)
1	1	0	0	S (G or C)
1	1	0	1	V (G or C or A)
1	1	1	0	B (G or T or C)
1	1	1	1	N (G or A or T or C)

CHAPTER 5

EXPERIMENTAL RESULTS

Implementing on the four *E.coli* genomes consisting of approximately 4-5 million nucleotides for each strain from EMBL database of European Bioinformatics Institute (EBI): *E.coli strain CFT073*, *E.coli strain K12*, *E.coli strain O157 : H7 EDL933*, and *E.coli strain O157 : H7 substrain RIMD 0509952* (Genome accession: AE014075, U00096, AE005174, and BA000007, respectively), we selected similar regions on the same position to create two data sets: training set and testing set. By our assumption, neighboring regions can extract the relative positions to resolve an ambiguous nucleotide in this region for all strains. For example, we chose similar regions of about 30,000 nucleotides from each strain and obtain 1.2×10^5 feature vectors with the number of relative position $k = 6$. Approximately 60% of these 1.2×10^5 vectors are duplicated in the same class and only 50,000 remaining feature vectors can be used as the data set to train the neural network. Among these data set, 85% and 15% of them are considered as training set and testing set, respectively. So the size of training sets for each region is up to 30,000-40,000 patterns. The number of patterns is too large to be trained by only one neural network. Hence, the training set must be distributed by parallel training as discussed in Chapter 4. Therefore, there are two results show-

ing the performance of predicting of recognition network for resolving ambiguous nucleotide and the performance of the parallel training work.

Figure 5.1 showed the relationship between the number of duplicated patterns and the number of relative positions. When the number of relative positions (k) increased, the duplicated patterns in the different class and the same class were reduced while the unduplicated patterns (or remaining patterns for training) are increased. The appropriate number of relative positions is equal to 6 ($k = 6$) in this experiment because τ is set to 0.005 and the error ratio of the number of duplicated vectors being in different classes to the number of remaining vectors (or unduplicated vectors) is 0.004223. The complexity of data sets can be computed by multiplying the number of remaining vectors (p) and the number of features (q). The number of features, q , is derived from the number of relative positions of four-type symbols and then equal to $k \times 4$. Although the error ratio is less than τ when k is equal or greater than 6, there is the least complexity ($p \times q$) at $k = 6$. Table 5.1 shows the effect of varying the value of k . When the number of relative positions increased, the number of features are also increased. Figure 5.2 shows the relationship between the number of relative positions and the complexity of unduplicated data set.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

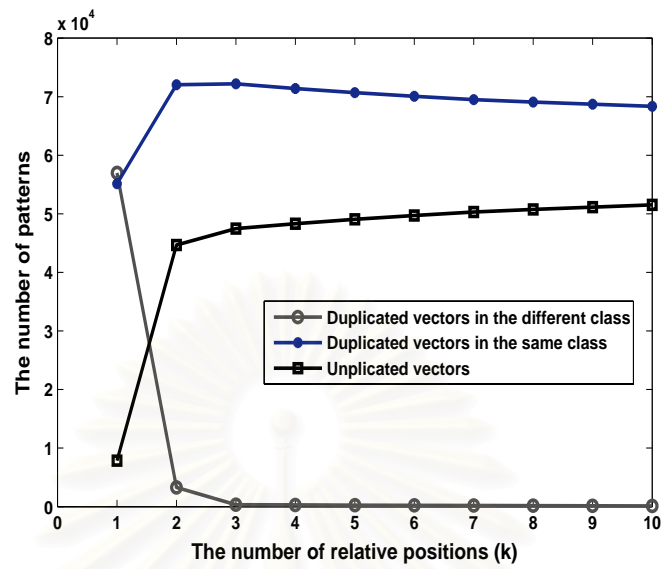


Figure 5.1: The relationship between the number of duplicated patterns and the number of relative positions.

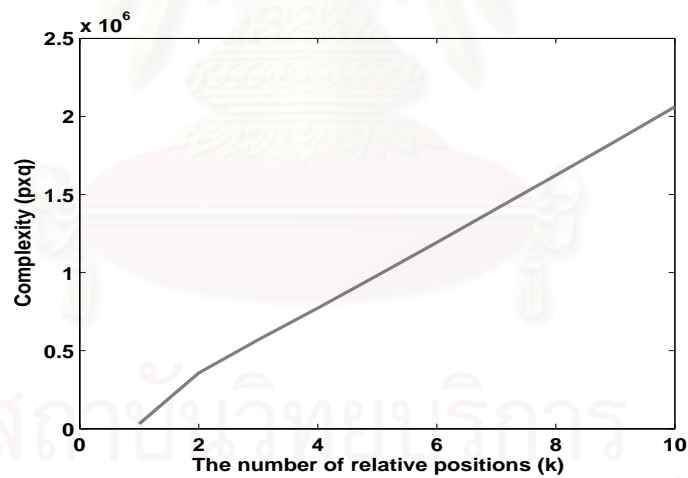


Figure 5.2: The relationship between the number of relative positions and the complexity of unduplicated data set.

Table 5.1: The results of duplicating process of the original feature vectors vary in the number of relative positions.

k	q	Duplicated patterns in different classes	Unduplicated patterns(p)	error ratio	complexity ($p \times q$)
1	4	56993	7863	7.24826	31452
2	8	3297	44667	0.07381	357336
3	12	327	47487	0.00688	569844
4	16	291	48318	0.00602	773088
5	20	258	49062	0.00525	981240
6	24	210	49719	0.00422	1193256
7	28	198	50307	0.00393	1408596
8	32	180	50739	0.00354	1623648
9	36	153	51132	0.00299	1840752
10	40	117	51531	0.0022	2061240

5.1 Performance of each Recognition Network

The performance of each recognition network corresponding to each actual symbol A , T , C , or G can be measured in terms of *sensitivity* (SN) and *specificity* (SP) [?] derived from true positive (TP)¹, true negative (TN)², false positive (FP)³, and false negative (FN)⁴ in equation (5.1). All recognition networks

¹TP equals the total number of correctly identified patterns.

²TN equals the total number of correctly unidentified patterns.

³FP equals the total number of incorrectly identified patterns.

⁴FN equals the total number of incorrectly unidentified patterns.

constructed by any similar regions in our experiment have the performance measurement in average range shown in Table 5.1. The sensitivity is the percentage of total correct patterns while the specificity is 1 - the percentage of total incorrect patterns.

$$SN = \frac{TP}{TP + FN}, \quad SP = \frac{TP}{TP + FP} \quad (5.1)$$

Table 5.2: The performance of each classification network on four sample regions.

	Network A		Network T		Network C		Network G	
	SN	SP	SN	SP	SN	SP	SN	SP
region 1	0.9444	0.8368	0.9433	0.8266	0.9332	0.8405	0.9157	0.7668
region 2	0.9227	0.8122	0.9530	0.8328	0.9631	0.8809	0.9273	0.8762
region 3	0.9500	0.8261	0.9432	0.8384	0.9532	0.8711	0.9503	0.8582
region 4	0.9412	0.8215	0.9368	0.8407	0.8951	0.7884	0.9268	0.8006
average	0.9396	0.8242	0.9441	0.8346	0.9361	0.8452	0.9300	0.8255

5.2 The Results of Resolving Method

There are two possible determinations of the outputs from all recognition networks. The first approach is by thresholding the output value of each network to 0 or 1 and comparing with the four-digit binary patterns in Table 4.3 to resolve the symbol. The second approach is to determine the final answer by the maximum output value. Experimenting on four *Escherichia coli* genomes, the correctness is up to 83.96% by the first approach shown in Table 5.4 and 93.34% by the second

approach shown in Table 5.5. The number of testing pattern is to 1000 from each nucleotide.

Table 5.3: Experimental results using the conclusion in Table 4.3.

Resolved Symbol	Target <i>A</i>	Target <i>T</i>	Target <i>C</i>	Target <i>G</i>
not applicable	121	114	60	23
A	825	7	0	1
T	8	794	2	0
W	19	20	0	0
C	1	10	849	1
M	14	2	35	0
Y	1	28	39	0
H	2	3	3	0
G	2	5	1	907
R	4	1	0	19
K	0	11	0	26
D	2	1	0	0
S	0	4	9	22
V	0	0	0	0
B	1	0	2	1
N	0	0	0	0
correctness(%)	82.45	79.40	84.93	90.68
average correctness (%)	83.96			

Table 5.4: Experimental results using the maximum output value.

Resolved Symbol	Target A	Target T	Target C	Target G
A	905	32	20	6
T	41	984	21	2
C	26	44	956	4
G	27	31	2	987
correctness(%)	90.49	89.35	95.61	98.71
average correctness (%)	93.34			

5.3 The Results of Query Sequences

To test whether the trained network can be used to resolve symbol N in the other regions, a new testing set containing nucleotide string S of length 45 nucleotides with format $(A + T + C + G)^n N$, where $n > 0$ is an integer is created. The number 45 is not a magic number but it is the maximum relative position in the experimental DNA sequence. This length may create a problem of some missing relative positions. Given a number of relative positions k , it is possible that the number of relative positions of some nucleotide symbols may be less than k . The corresponding feature vector cannot be completely formed. For this situation, any missing relative position is, then, set to the large value. So its inversely relative position is closed to 0. For instance, consider this given query nucleotide sequence.

symbols:	C	A	T	T	T	C	T	C	A	C	G	C	T	T	G	T	N
position:	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Suppose k is equal to 3. Nucleotide A has only two relative positions, r_{N_1, A_9} and $r_{A_9, A_{16}}$. Thus, the last relative position for nucleotide A is missing and its inversely relative position is set to 0. Similarly, nucleotide G also has two relative positions, r_{N_1, G_3} and r_{G_3, G_7} . The third inversely relative position of G is set to 0. The feature vector of this given nucleotide sequence is as follows. The testing results are shown in Tables 5.5 and 5.6.

$$\mathbf{v}_S = \begin{array}{c} \\ \\ \\ \end{array} \begin{array}{ccc|ccc|ccc} & \text{A} & & & \text{T} & & & \text{C} & & & \text{G} & \\ \hline & \frac{1}{8} & \frac{1}{7} & 0 & \frac{1}{1} & \frac{1}{2} & \frac{1}{1} & \frac{1}{5} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & 0 \\ \hline = & 0.125 & 0.1429 & 0 & 1.0 & 0.5 & 1.0 & 0.2 & 0.5 & 0.5 & 0.5 & 0.25 & 0 \end{array}$$

Considering the two determinations of output values from four recognition networks, the second approach of using maximum output value is able to determine the final answer better than the first approach of using four-digit binary decision in that both the testing set in the similar area and the query testing set. Therefore, the recognition networks are efficient to produce the output value as probability to predict an ambiguous nucleotide symbol. The modification of the relative position with mutation matrix in Chapter 4 and using the second approach were constructed as a recognition network with mutation changes. For 4,000 query samples, the result of predicting an ambiguous symbol compared with Sampled Pattern Matching (SPM) is shown in Table 5.8.

Table 5.5: Experimental results using the patterns in Table 4.3 of query sequence

Resolved Symbol	Target <i>A</i>	Target <i>T</i>	Target <i>C</i>	Target <i>G</i>
not applicable	135	73	69	49
A	708	21	21	17
T	18	733	17	19
W	29	47	4	4
C	17	32	756	18
M	19	6	58	8
Y	8	40	17	4
H	4	3	3	6
G	10	18	21	734
R	25	1	5	70
K	9	8	1	17
D	1	4	0	6
S	4	3	17	35
V	8	8	4	6
B	5	3	3	7
N	0	1	3	1
correctness(%)	70.79	73.28	75.63	73.40
average correctness (%)	73.26			

Table 5.6: Experimental results using the maximum output value of query sequences

Resolved Symbol	Target <i>A</i>	Target <i>T</i>	Target <i>C</i>	Target <i>G</i>
A	823	52	57	60
T	61	835	50	50
C	65	67	846	49
G	51	45	46	841
correctness(%)	82.34	83.47	84.64	84.12
average correctness (%)	83.63			

Table 5.7: Experimental results of recognition network comparing with Sampled Pattern Matching(SPM) for 4,000 samples

	SPM	Recognition Networks
predictable patterns	1295	3454
correctness	32.37%	86.54%

5.4 Performance of Parallel Technique

The performance of the proposed technique was compared in terms of CPU-time and average CPU-time for all nodes in the neural network defined in Chapter 3. The average time for each node in a neural network is the smallest part of the calculation in the neural network. All simulations of sub-networks for each network type *A*, *T*, *C*, or *G* were carried out on Intel Pentium4 1.50GHz with 256MB RAM PC, using the SNNS 4.2, Stuttgart Neural Network Simulator which is publicly available at <http://www-ra.informatik.uni-tuebingen.de/SNNS/>, on Linux

operating system. By distributing the data to construct the sub-networks and concurrently training them, an ambiguous nucleotide problem can be rapidly resolved. The performance of this technique is shown in Table 5.8 and Figure 5.3 for 20,000 sample feature vectors.

In parallel theory, the good speedup(S) should be equal to the number of processors, Num , and the efficiency(E) of processor, $\frac{speedup}{Num}$, should be equal to 1. But our algorithm can reduce the training time more than the increasing of the number of processors. Therefore, the speedup can be more than the number of processors and the efficiency of each processor can be higher.

Table 5.8: The performance results of this parallel technique.

Num	I/Net	CPU Time	Epoch	N/Net	AT/N	AE/N	S	E
1	20000	157179.99	50000*	175	898.1714	0.018	1	1
2	10000	50528.01	50000*	125	404.2241	0.0081	2.22	1.11
4	5000	3256.14	8736	96	33.9181	0.0039	4.63	1.16
6	3334	535.36	2730	82	6.5288	0.0024	7.51	1.25
8	2500	344.1	2600	76	4.5276	0.0017	10.31	1.29
10	2000	174.31	1864	70	2.4901	0.0013	13.45	1.34
12	1667	113.87	1599	66	1.7253	0.0011	16.64	1.38

*Not Acceptable

N/Net : Nodes per network

S: Speedup

Num : Number of networks

AT/N: Average time per node

E: Efficiency

I/Net : Input patterns per network

AE/N: Average time a epoch per node

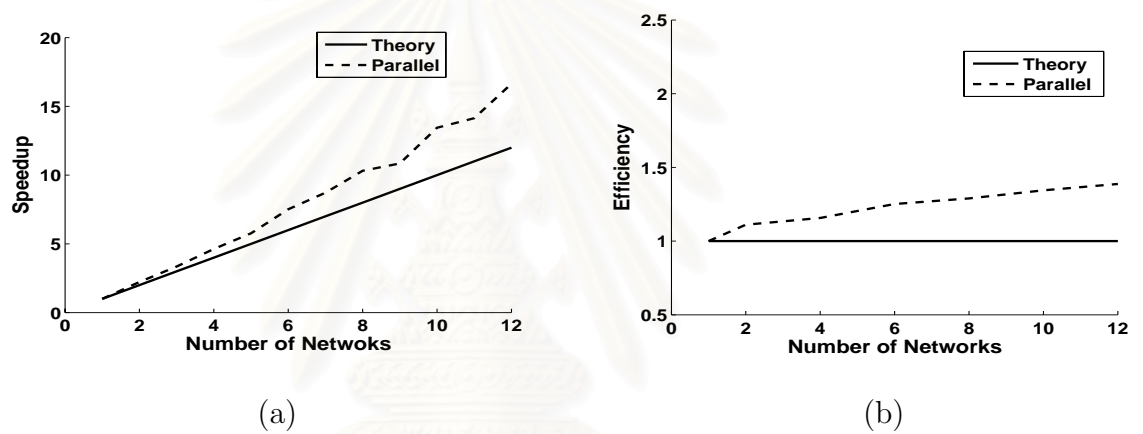


Figure 5.3: (a) Relationship between the speedup and the number of networks.

(b) Relationship between the efficiency and the number of networks.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER 6

CONCLUSION

Two new concepts for resolving the ambiguous nucleotide denoted by symbol N and speeding up the recognition process by parallel training are proposed. The nucleotide resolving algorithm can achieve an impressive average accuracy and the parallel training can increase the speed of training time more than the number of processors. The speedup obtained was super linear since training the whole data set by a single neural networks take a non-applicable time which can be considered as an infinite time. But when the data set are partitioned and trained by each individual network, the convergence can be achieved in a short period. The partitioning technique managing on data sets can be applied to other models of artificial neural networks to reduce computational time.

The approach to resolve the ambiguous symbol N in a DNA sequence based on relative positions, recognition, and classification concepts is implemented by using this parallel technique. The feature vector formed by concatenating the inversely relative positions used as the feature of the sequence during the learning and recognizing processes by a neural network. The output of each recognition networks is similar to emission probability weighted by the relative positions; therefore, the outputs from all recognition networks are simultaneously considered

to determine the final answer.

Experimenting on four *Eachericia coli* genomes, the results of both possible determinations are better than the expected results of SPM. The first approach is by thresholding the output value of each network to 0 or 1 and comparing with the four-digit binary patterns to resolve the symbol. The second approach is to determine the final answer with maximum output value. The results of using maximum output value are able to determine the ambiguous symbol better than the results of using the four-digit binary decision in both the testing set in the similar area and the query testing set.

For further study, this prediction approach will be extended to cover other species to verify the efficiency of the recognition network and extend to predict more than one ambiguous nucleotide.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

REFERENCES

- [1] M. Pop and D. Kosack. Using the TIGR Assembler in shotgun-sequencing projects, Bacterial Artificial Chromosomes, Humana Press 1 (2004): 279-294.
- [2] M. Pop, S. L. Salzberg and M. Shumway. Genome Sequence Assembly: Algorithms and Issues, IEEE Computer (2002): 47-54.
- [3] M.K. Golberg, D.T. Lim and M. Magdon-Ismail. A Learning Algorithm for String Assembly, in Workshop on Data Mining in Bioinformatics with SIGKDD01 Conference: 32-37.
- [4] P. Gajer, M. Schatz and S.L. Salzberg. Automated correction of genome sequence errors, Nucleic Acids Research (2004): 562-569.
- [5] J. Blazewicz, P. Formanowicz, F. Guinand and M. Kasprzak. A heuristic managing errors for DNA sequencing, Bioinformatics Applications note 18 (2002): 652-660.
- [6] Amplicon express. DNA Sequencing Troubleshooting, Available from: http://www.genomex.com/AEX_zone/troubleshooting.html.
- [7] G. P. Moss. International Union of Pure and Applied. Chemistry, Available from: <http://www.chem.qmul.ac.uk/iupac/>.
- [8] Christine Orengo, David Jones, and Janet Thornton. Bioinformatics: genes, proteins and computers. BIOS Scientific Publishers Limited, 2003.
- [9] Masatoshi Nei and Sudhir Kumar. Molecular Evolution and Phylogenetics. (n.p): Oxford University Press, (2000).
- [10] Peter F. Arndt, Christopher B. Burge, and Terene Hwa. DNA Sequence Evolution with Neighbor-Dependent Mutation. Journal of Computational Biology 10, 3-4 (2003): 313-322.
- [11] Matthew J. Gonzales, Jonathan M. Dugan and Robert W. Shafer. Synonymous-non-synonymous mutation rates between sequences containing ambiguous nucleotides(Syn-SCAN), Bioinformatics Applications note 18 (2002): 886-887.
- [12] Hong Wan, Lugang Li, John C. Wootton. Discovering Simple regions in Biological Sequences Associated with Scoring Schemes. Journal of Computational Biology 10, 2 (2003): 171-185.
- [13] Rumelhart. D. E. and McClelland, J. L. (eds). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press 1.

- [14] Cathy H. Wu, Geogr M. Whitson, Chun-Tse Hsiao and Cheng-Fu Huang. Classification Artificial Neural Systems for Genome Research, Conference on High Performance Networking and Computing Proceedings ACM/IEEE conference on Supercomputing (1992): 797-803.
- [15] Maxam AM and Gilbert W. A new method for sequencing DNA. Proceeding Natural Academic Science 74, 2 (February 1977): 560-564.
- [16] Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors, Proceeding Natural Academic Science 74, 12 (December 1977): 5463-5467.
- [17] Sun Kim. A Survey of Computational Techniques for Genome Sequencing, Center for Genomics and Bioinformatics. Indiana University, Bloomington, (2002).
- [18] Rutgers. Sequencing and Amplifying DNA, Available from Molecular Biology and Biochemistry.
- [19] Tao Jiang, Ming Li. DNA Sequencing and String Learning. Mathematical Systems Theory 29, 4 (1996): 387-405.
- [20] I. Apostol, P. Jacquet, and W. Szpankowski. How Predictable Are Biological Sequences?, European Conference on Computational Biology (ECCB2003) (September 2003).
- [21] I. Apostol, P. Jacquet, and W. Szpankowski. Universal Predictor Based on Pattern Matching, IEEE Trans. Information Theory 48 (2002): 1462-1472.
- [22] D. Cornforth, and D. Newth. The Kernel Addition Training Algorithm: Faster Training for CMAC Based Neural Networks, Proceedings Conference Artificial Neural Networks and Expert Systems. Otago, (2001).
- [23] A. Roy, S. Govil and R. Miranda. A Neural-Network Learning Theory and a Polynomial Time RBF Algorithm, IEEE Transactions on Neural Networks 8, 6 (1997): 1301-1313.

VITAE

Kitiporn Plaimas was born in October 26, 1980, in Bangkok. She has been on a DPST scholarship from the Institute for the Promotion of Teaching Science and Technology (IPST) since 1998. She received Bachelor's degree in Mathematics from the Faculty of Science, Mahidol University in 2001.

PUBLICATION

- K. Plaimas, C. Lursinsap, and A. Suratane, "High Performance of Artificial Neural Network of Resolving Ambiguous Nucleotide Problem", *the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS-2005)*, April 4-8, 2005, Denver, Colorado, USA.
- K. Plaimas, C. Lursinsap, and A. Suratane, Partitioning Data Set for Training Neural Network in Parallel to Resolve Ambiguous Nucleotide Problem, *NECSEC2005*, March 31-April 1, 2005, Khon Kaen, Thailand.
- K. Plaimas, C. Lursinsap, and A. Suratane, Resolving Ambiguous Nucleotide Symbols Associated with Scoring Scheme of Mutation Rates, *the 9th Annual National Symposium on Computational Science & Engineering (ANSCSE9)*, March 23-25, 2005, Mahidol University, Bangkok, Thailand.
- K. Plaimas, C. Lursinsap, and A. Suratane, "Resolving Ambiguous Nucleotide Symbols Using Weighted Relative Position Recognition", *ThCSC2004*, December 16-17, 2004, Kasetsart University, Bangkok, Thailand.
- K. Plaimas and C. Lursinsap, "Resolving Ambiguous Nucleotide Symbol from DNA Sequences", *International Conference on Bioinformatics 2004 (InCob2004)*, September 5-8, 2004, Auckland, New Zealand.