8-12-2016

Modern Computing Techniques for Solving Genomic Problems

Ning Yu

MODERN COMPUTING TECHNIQUES FOR SOLVING GENOMIC PROBLEMS

by

NING YU

Under the Direction of Yi Pan, PhD

ABSTRACT

With the advent of high-throughput genomics, biological big data brings challenges to scientists in handling, analyzing, processing and mining this massive data. In this new interdisciplinary field, diverse theories, methods, tools and knowledge are utilized to solve a wide variety of problems. As an exploration, this dissertation project is designed to combine concepts and principles in multiple areas, including signal processing, information-coding theory, artificial intelligence and cloud computing, in order to solve the following problems in computational biology: (1) comparative gene structure detection, (2) DNA sequence annotation, (3) investigation of CpG islands (CGIs) for epigenetic studies.

Briefly, in problem #1, sequences are transformed into signal series or binary codes. Similar to the speech/voice recognition, similarity is calculated between two signal series and subsequently signals are stitched/matched into a temporal sequence. In the nature of binary operation, all calculations/steps can be performed in an efficient and accurate way. Improving performance in terms of accuracy and specificity is the key for a comparative method. In problem #2, DNA sequences are encoded and transformed into numeric representations for deep learning methods. Encoding schemes greatly influence the performance of deep learning algorithms. Finding the best encoding scheme for a particular application of deep learning is significant. Three applications (detection of protein-coding splicing sites, detection of lincRNA splicing sites and improvement of comparative gene structure identification) are used to show the computing power of deep neural networks. In problem #3, CpG sites are assigned certain energy and a Gaussian filter is applied to detection of CpG islands. By using the CpG box and Markov model, we investigate the properties of CGIs and redefine the CGIs using the emerging epigenetic data.

In summary, these three problems and their solutions are not isolated; they are linked to modern techniques in such diverse areas as signal processing, information-coding theory, artificial intelligence and cloud computing. These novel methods are expected to improve the efficiency and accuracy of computational tools and bridge the gap between biology and scientific computing.

INDEX WORDS: Signal processing, Deep learning, Cloud computing, Biological big data, DNA annotation, Epigenetics, CpG island, CpG box, Markov model

MODERN COMPUTING TECHNIQUES FOR SOLVING GENOMIC PROBLEMS

by

NING YU

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2016

MODERN COMPUTING TECHNIQUES FOR SOLVING GENOMIC PROBLEMS

by

NING YU

Committee Chair:	Yi Pan
Committee:	Raj Sunderraman
	Robert Harrison
	Jing Zhang

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2016

DEDICATION

To my wife, for her understanding and sacrifice;

to my parents, Ruizhi Yu and Fengrong Sui, for their endless love;

to my parents-in-law, Peifeng Liu and Qihuan Li, for their support;

to my sister and brother-in-law for being the best siblings;

and to my sons, Thomas and Charles, and my niece, Sihan, for being the best kids.

I love you all dearly and would most certainly not be who I am without all of you.

ACKNOWLEDGEMENTS

I would like to thank my adviser Dr. Yi Pan. Without his constant encouragement and guidance I would most certainly not have completed much of anything. His guidance is always the best way towards the goal. I could not imagine having selected a different adviser for my Ph.D program. I am truly indebted to him.

I am grateful for the help and support from Dr. Raj Sunderraman who has advised me over the years. His wisdom, knowledge, kindness and generosity positively influence all students in our department including me.

I would also like to thank Dr. Robert Harrison for all his advice in classroom and out of classroom. His advice is always a Midas touch. I am truly thankful for him sharing his wisdom throughout my PhD program.

A special thank to Dr. Jing Zhang. I have learned a lot in her class. Her knowledge in Math and Statistics definitely is a treasure for our further collaboration in the future.

I would like thank faculty members in computer science department of GSU, Dr. Alex Zelikovsky, Dr. Xiaojun Cao, Dr. Yingshu Li, Dr. Michael Weeks, Dr. Zhipeng Cai, Dr. Yanqing Zhang, Dr. K. N. King and Dr. Ying Zhu. This is an amazing faculty team.

I would say thank you to my colleagues, Bing Li, Zeng Yu, Xuan Guo, Chaoyang Li, Long Ma, Guoliang Liu, Wei Pei, Xiangyuan Zhu, Zaobo He, Xu Zheng, Yi Liang, Ziyue Chen, Sunitha Basodi, and Geeta M Karadkhele, for their kind help and concerns.

I thank my former adviser Dr. Qiang Cheng for his constant support. I am grateful for my friends Dr. Di-Wen Chen and Dr. Feng Gu for recommending me to GSU CS PhD program. I thank all my friends and my family members for their unrelenting support throughout my life.

Lastly I would like to thank my wife Yin, my parents and my parents-in-law. I could not have finished without their endless support and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- AE - Auto-encoder
- ANN - Artificial Neural Network
- AUC - Area Under Curve
- CAB - Character Analysis Based
- CAF - Character Analysis Free
- CGI - CpG Island
- CpG - Cytosine-phosphate-Guanine
- DAE - Denoising Auto-encoder
- DAX - DNA As X
- DDAE - Double Denoising Auto-encoder
- DFT - Discrete Fourier Transform
- DL - Deep Learning
- DMR - Differentially Methylated Region
- DNA - Deoxyribonucleic acid
- DNN - Deep Neural Network
- DTW - Dynamic Time Warping
- EIIP - Electron-ion Interaction Pseudopotentials
- FPGA - Field-Programmable Gate Array

- GPU - Graphic Processing Unit
- HDAE - Hidden-layer Denoising Auto-encoder
- HMM - Hidden Markov Model
- NGS - Next Generation Sequencing
- NN - Neural Network
- MCC - Matthews Correlation Coefficient
- MFE - Minimum Free Energy
- MLP - Multiple-layer Perceptron
- ORF - Open Reading Frame
- RDD - Resilient Distributed Dataset
- RNA - Ribonucleic Acid
- ROC - Receiver Operating Characteristic curve
- SVM - Support Vector Machine
- TSS - Transcription Starting Site

CHAPTER 1

INTRODUCTION

1.1 Background and Problems

The exponential growth in biological big data is a big challenge for researchers in life science with the advent of the next generation sequencing technologies [1]. Diverse theories, methods, tools and knowledge are utilized to solve a wide variety of problems. However, the gaps between biology and computing are still large [2]. As an exploration, this dissertation aims to develop state-of-the-art techniques by combining those concepts and principles in the diverse areas of computing techniques, including signal processing, information-coding theory, cloud computing and artificial intelligence in order to solve the following selected problems:

Problem 1 Comparative methods for gene structure prediction. This is a fundamental issue in comparative genome studies. Improving the specificity and the accuracy is critical for this task.

Problem 2 Deep learning methods for DNA sequence annotation. Theoretically, coding and non-coding DNA sequences are distinct from each other in function while the DNA sequences are hard to identify. Complementary to conventional computational models, the emerging methods in deep learning are expected to improve the accuracy in computational annotation of DNA genome sequence.

Problem 3 Investigation of CpG island in human genome sequences. The investigations include the detection of CpG island (CGI), the re-definition of CGI and the exploration of CGI structures since CGI is an important epigenetic marker and the current definition of CGI cannot support the emerging data set such as methylation data. Data-driven

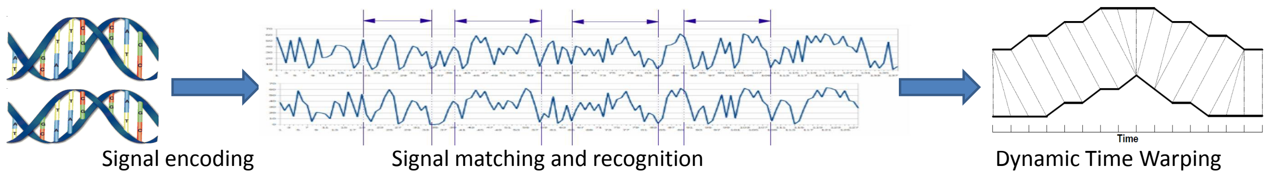


Figure 1.1: An Example of Signal Processing for Comparative Genomic Studies

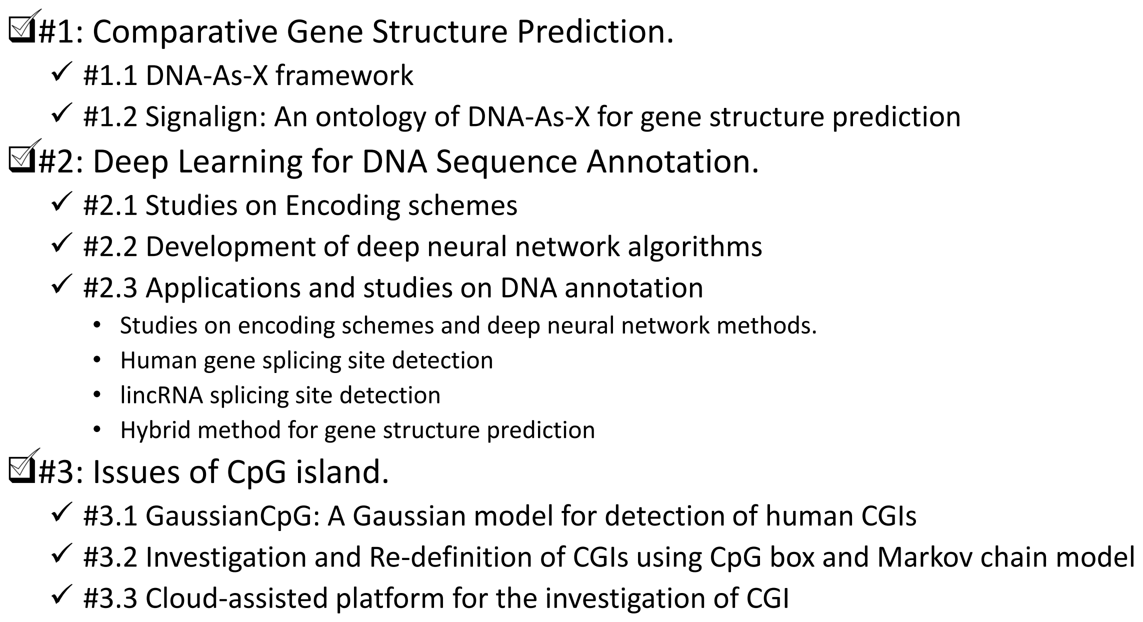
CGI structure analysis can help scientists discover the epigenetic processes in CpG-rich areas.

1.2 Goal and Outline

In order to solve problem #1, many related work have been conducted by scientists [3][4][5][6][7]. In my previous work, a general framework, called DNA-As-X [8] was proposed for character-analysis-free techniques to overcome these shortcomings, where X is the intermediates, such as digit, code, signal, vector, tree, graph network and so on. A simple example in Figure1.1 shows a novel signal processing method for comparative genomic studies [8].

As for problem #2, the computational annotation of DNA sequences is an indispensable task with the advent of next generation sequencing technology. Two primary methods are widely accepted, (1) *ab initio* method that directly detects the DNA sequence without any reference, (2) comparative method that studies the known data base and acquires the knowledge for detection of DNA sequence. Here, I adopt the encoding technique to convert the sequences into signals and use advanced deep learning methods to learn the knowledge of coding DNA sequences, which are expected to have better performance over other conventional methods.

Similarly, as for problem #3, many research have been performed on CGI investigation [9][10]. However, few of them use signal processing to detect CGI; a few of them are seen to redefine the CGI to support the emerging data; and the deep investigations to the CGI structure are barely known. In my novel work, a Gaussian digital model called GaussianCpG [11] is developed for detection of CGI in human genome sequences. The validation results show its superiority on balancing the sensitivity and the specificity over other methods.



☑#1: Comparative Gene Structure Prediction.
- ✓ #1.1 DNA-As-X framework
- ✓ #1.2 Signalign: An ontology of DNA-As-X for gene structure prediction

☑#2: Deep Learning for DNA Sequence Annotation.
- ✓ #2.1 Studies on Encoding schemes
- ✓ #2.2 Development of deep neural network algorithms
- ✓ #2.3 Applications and studies on DNA annotation
  - Studies on encoding schemes and deep neural network methods.
  - Human gene splicing site detection
  - lincRNA splicing site detection
  - Hybrid method for gene structure prediction

☑#3: Issues of CpG island.
- ✓ #3.1 GaussianCpG: A Gaussian model for detection of human CGIs
- ✓ #3.2 Investigation and Re-definition of CGIs using CpG box and Markov chain model
- ✓ #3.3 Cloud-assisted platform for the investigation of CGI

Figure 1.2: The Main Goals in the Dissertation

Moreover, a deep investigation to CGI is performed for solving the problems about CGI definition and property investigation.

novel computing methods integrate signal processing, information coding techniques, clouding computing, statistic analytics, and deep learning and constitute the technological mainline throughout the whole dissertation. Figure1.2 gives the outline of main goals detailed in tasks and sub-tasks.

1.3 Methodology

The primary methods adopted in the dissertation are shown in Figure1.3. These methods can be classified into three categories as follows.

The first class is signal processing and information-coding methods, including (A) Signal encoding, (B) Error tolerance and detection, (C) Dynamic Time Warping for sequential series, (D) Signal-Noise-Ratio (SNR) for measurement , (E) Gaussian filter and (G) erosion digital filter, (J) Signal matching and recognition and (K) Information entropy.

The second class is artificial-intelligence techniques, including (F) Markov chain model,

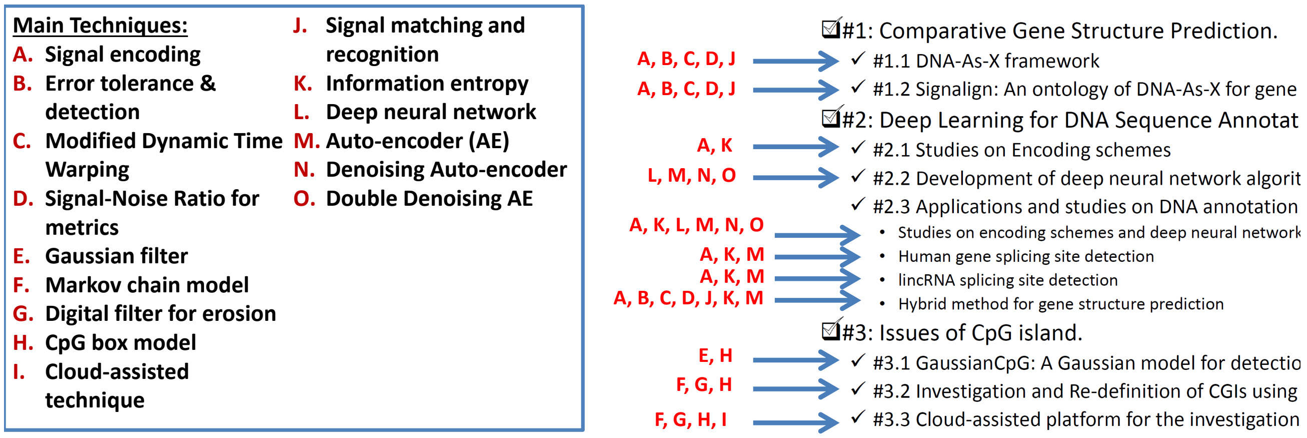

Main Techniques:
A. Signal encoding
B. Error tolerance & detection
C. Modified Dynamic Time Warping
D. Signal-Noise Ratio for metrics
E. Gaussian filter
F. Markov chain model
G. Digital filter for erosion
H. CpG box model
I. Cloud-assisted technique
J. Signal matching and recognition
K. Information entropy
L. Deep neural network
M. Auto-encoder (AE)
N. Denoising Auto-encoder
O. Double Denoising AE
#1: Comparative Gene Structure Prediction.
A, B, C, D, J
#1.1 DNA-As-X framework
A, B, C, D, J
#1.2 Signalign: An ontology of DNA-As-X for gene
#2: Deep Learning for DNA Sequence Annotat
A, K
#2.1 Studies on Encoding schemes
L, M, N, O
#2.2 Development of deep neural network algorit
#2.3 Applications and studies on DNA annotation
A, K, L, M, N, O
Studies on encoding schemes and deep neural network
A, K, M
Human gene splicing site detection
A, K, M
lincRNA splicing site detection
A, B, C, D, J, K, M
Hybrid method for gene structure prediction
#3: Issues of CpG island.
E, H
#3.1 GaussianCpG: A Gaussian model for detectio
F, G, H
#3.2 Investigation and Re-definition of CGIs using
F, G, H, I
#3.3 Cloud-assisted platform for the investigation

Figure 1.3: Methodology for Solving Problems
Red letters denote the methods adopted by each sub-task.

(H) CpG box model, (L) Deep neural network, (M) Auto-encoder, (N) Denoising auto-encoder and (O) Double denoising auto-encoder. (F) and (H) are particularly designed for CpG island for measuring the probabilities between two CpG sites; (L)-(O) are various techniques used in deep learning, especially in deep neural network.

The third class is the cloud-assisted algorithm (I). In problem #3, the specific cloud-assisted algorithms are designed for processing the large scale of biological data.

1.4 Contributions

This project aims to cover some topics in the middle of these interdisciplinary fields and advance computational methods for the discovery of new knowledge in computational biology. All these problems and solutions are not isolated and they are all linked to innovative techniques in the areas of signal processing and information-coding theory, artificial intelligence and cloud computing, which are expected to improve the efficiency and the accuracy in computational tools and help to bridge the gap between biology and scientific computing.

Particularly, in problem 1, a novel method that integrates signal processing and information encoding can improve the accuracy in comparative gene structure prediction. In problem 2, deep learning methods are applied into DNA annotation problems and show the superiority in raising the accuracy in genome analysis. These deep learning based techniques are anticipated to play more important roles in the future research. In problem 3, the CpG box and Markov model are first proposed for investigating and redefining CpG islands in epigenetic studies. In addition, a cloud-assisted platform is established for further analysis in CpG island related issues.

1.5 Organization

This article is organized as follows. Chapter 2 discusses the comparative methods in gene structure detection using signal processing and information-coding techniques. Chapter 3 describes the studies on encoding schemes and deep neural network methods and presents three applications to illustrate the potential power of deep neural network in annotating DNA sequences. Chapter 4 depicts the method of detecting CpG island (CGI) based on Gaussian model and redefines the CGI to meet the emerging methylation data using CpG box and Markov model. Furthermore, many deep and interesting investigations and discussions are described on this chapter. Finally the paper outlines ongoing and future work in Chapter 5.

1.6 Publications

Journal Papers

1. **N. Yu**, X. Guo, A. Zelikovsky, Y, Pan, GaussianCpG: A Gaussian Model for Detection of CpG Island in Human Genome Sequences. *BMC Bioinformatics*, 2016 (Accepted)

2. **N. Yu**, X. Guo, F. Gu, and Y. Pan, Signalign: An Ontology of DNA As Signal for Comparative Gene Structure Prediction Using Information-Coding-and-Processing Techniques. *IEEE Transaction on Nanobioscience*, Vol. 15, No. 2, Mar. 2016

3. **N. Yu**, Z. Yu, B. Li, F. Gu, and Y. Pan, A Comprehensive Review of Emerging Computational Methods for Gene Identification, *Journal of Information Processing Systems*, Vol. 12, No. 1, pp.1-34, Mar. 2016

4. B. Li, J. Zhang, **N. Yu**, Y. Pan. J2M: A Java to Mapreduce Translator for Cloud Computing. *Journal of Supercomputing*, Mar. 2016, pp.1-18, Mar. 2016

5. X. Guo, **N. Yu**, X. Ding, J. Wang, Y. Pan. DIME: A Novel Framework for De Novo Metagenomic Sequence Assembly. *Journal of Computational Biology*, vol.22, Issue.2,pp.159-177, Feb. 2015

6. X. Guo, **N. Yu**, F. Gu, X. Ding, J. Wang, Y. Pan. Genome-Wide Interaction-Based Association of human diseases-A survey. *Tsinghua Science and Technology*, vol.19, no.6, pp.596-616, Dec. 2014

7. X. Guo, Y. Meng, **N. Yu**, and Y. Pan. Cloud computing for detecting high-order genome-wide epistatic interaction via dynamic clustering. *BMC bioinformatics* 15, no. 1 (2014): 102

8. R. A. Tesorero, **N.Yu**, K.H. Cho, Q. Cheng. New Small RNAs in Streptococcus Pyogenes. *PLOS One*, Vol 8, Issue 6, June 2013

Conference / Workshop Papers

1. **N. Yu**, X. Guo, A. Zelikovsky, Y, Pan. GaussianCpG: A Gaussian Model for Human CpG Island Detection Using Gaussian Energy Metrics. *5th IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)* 2015, Miami, FL, Oct, 2015

2. **N. Yu**, X. Guo, F. Gu, Y. Pan. DNA AS X: An Information-Coding-Based Model to Improve the Sensitivity in Comparative Gene Analysis. In *Bioinformatics Research and Applications, ser. Lecture Notes in Computer Science*, R. Harrison, Y. Li, and I. Mndoiu, Eds. Springer International Publishing, 2015, vol. 9096, pp. 366377.

3. **N. Yu**, F. Gu, X. Guo, Z. He. A Fine-Grained Flow Control Model for Cloud-Assisted Data Broadcasting. *2015 Spring Simulation Multi-conference (SpringSim15)*, 324-331, 2015, Alexandria, VA, USA, April, 2015

4. **N. Yu**, K.H. Cho, Q. Cheng, R. A. Tesorero. A Hybrid Computational Approach for the Prediction of Small Non-coding RNAs from Genome Sequences. *International Conference on Computer Science and Engineering*, vol. 2, pp.1071-1076, 2009

Posters

1. **N. Yu**, X. Guo, F. Gu. A proportional algorithm for parallel scheduling based on PID controller and fine-grained energy metric. *In 2014 International IBM Cloud Academy Conference on*, pp. 11-14. IBM, 2014.

CHAPTER 2

COMPARATIVE METHODS FOR GENE STRUCTURE PREDICTION USING SIGNAL PROCESSING AND INFORMATION-CODING TECHNIQUES

In this chapter, the first section gives the motivation that signal processing and information-coding techniques can be applied in comparative genome studies and explains the basic principle behind it. Section 2.2 introduces a generic framework called DNA-As-X for the applications on genome analysis. Section 2.3 depicts the methodology and the implementation of an ontology of DNA-As-Signal. Subsequently, Section 2.4 provides a comprehensive valuation. Finally, Section 2.5 discusses the downside of the method and summarizes this chapter.

2.1 Introduction

2.1.1 Motivation

Since several decades ago characters (A, T, C and G) have been used as symbols to represent the nucleotides in sequences of deoxyribonucleic acid (DNA) and character-analysis-based techniques have underlain the research methodologies in bioinformatics and DNA genome analysis. Many existing computational tools [12][13][14][15] take full advantage of the properties of character representation - readable, understandable and convenient for sequence analysis. However, with the advent of biological big-data era, conventional character-based techniques in DNA genome analysis exhibit three main shortcomings - (1) inefficient to deal with large-volume genome data, (2) inflexible to handle various errors, mutations, insertion-deletions, frame shifts and gaps in DNA genome sequences, and (3) incompatible to well-developed engineering tool kits.

First, in order to overcome the inefficiency, two solutions are come up with. One solution is the applications of various data compression techniques, such as Lempel-Ziv-Welch data compression [16], Burrows-Wheeler transform and local compression [17][18], which greatly decrease the computing resources and promote the system efficiency; In parallel to the first solution, novel numerical representations are proposed and adopted by character-analysis-free techniques (CAF) that are our focus in this chapter.

Second, gapped extension [19][20], seeds and masks [21][22][23], and scoring and substitution matrices [24] are developed in character-analysis-based techniques to make up the flexibility issues. However, it causes another problem - the comparative tools become complicated if one uses more masks for higher sensitivity in some particular applications. Moreover, although seed is a widely adopted method that can speed up the performance in searching character-based context, it does not perform well in error-prone situations [25]. In recent studies [21][23], lots of hidden homology in DNA genome are still not found by current comparative tools despite decades of research.

Third, the gaps between biology and engineering or computer science are still large. Character-based representations make the existing methods/techniques in numeric computing hardly to exert in the interdisciplinary research. No generic methods are proposed to fill the enlarged gaps and to some extent it hinders veterans who work on traditional engineering fields to apply their expertise directly to biological area.

Techniques in signal processing and information-coding theory can help overcome these downsides of conventional character-based methods, because the existing and mature methods in engineering can assist to improve the accuracy and the efficiency and they have the natural advantages in numerical analysis and cloud/parallel computing.

2.1.2 Related Work

The computational methods of comparative analysis in DNA genomes have evolved to the fundamental infrastructure in bioinformatics [26]. Complying with the established character-based rules since the late 1970s, scientists have employed assorted methods to study these characters and hunt various patterns in character strings [19], resulting in the prosperity of character-analysis-based methods. One of the most canonical methods is the sequence local alignment, such as FASTA [27] and BLAST [28].

As the core of BLAST, seed-and-extend algorithm uses k-mer words as seed [28][20] and extends the search around seed, which is widely applied in numerous applications of comparative analysis and homology studies. As k-mer is usually set to the default value of 11, the heuristic method improves the running time significantly. However, the accuracy is negatively affected because the coarse granularity of a seed determines that even subtle mutations or small shifts are not considered. The improvement of this issue is derived from a technique named spaced seed that allows mismatching for gaps and mutations in a k-mer seed [19][20]. Concretely, a mask is applied to a seed in certain positions where mismatch is allowed. For example, a spaced seed of length 11 and weight 8 in the mask of 11110110011 allows mismatching in positions of 0 and exact match in positions of 1. This technique gives relatively more flexibility to character-analysis-based methods. However, for various patterns, more seed masks are needed. In [21][23] lots of new masks are designed to filter diverse cases and complicated syntax for masks are created in the nature of character-analysis-based methods. They do work well in some particular applications but such designs make the system more complicated with the loss in usability.

For other canonical algorithms such as Smith-Waterman and Needleman-Wuncsh [27][29], although they can find the optimal solution for global alignment, two issues are existing. (1) The time complexity is $O(n^2)$. It means that for long sequences it is inefficient in computation. Many present global-alignment tools, such as FASTA, are improved by combining heuristic algorithms, like seed-and-extend algorithm, to decrease the running time. (2) Even though the global alignment algorithm can find the optimal solution, in practice the optimum does not mean that it is the ground truth, especially in structure prediction for a typical example of the conservation search in RNA structure studies. False positive alignments may be generated as two sequences are globally aligned [30].

Additionally, although assembling algorithms such as burrow-wheeler transform have good performances in mapping and sequence assembly, they primarily deal with sequences

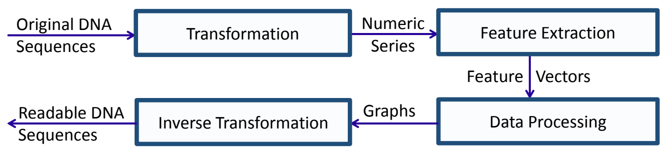


Figure 2.1: Main Procedures in DNA-As-X Framework

high similarity because the compact/compressed indexing structures in suffix tree or suffix array do not scale well for an error-prone query [18]. For query sequences with relatively large distance, the computational loads grow exponentially [31], inefficient to handle high degrees of dissimilarity through string-matching indexing structures.

2.2 Framework of DNA-As-X

Character-analysis-free techniques (CAF) are defined to those technologies that do not use characters or strings as the intermediates for data analysis and processing. In the DNA-As-X framework, two aspects are substantial in CAF techniques: transformation model and processing method. The former is about the non-character transformation, namely the numerical representation or graphical representation, which is the foundation of CAF techniques; the latter is based on the transformed data for further processing.

The purpose of DNA-As-X framework is to deal with above three issues on character representation and character-analysis-based techniques. DNA-As-X is proposed as a generic framework for genome analysis and processing, where X can be various formats of intermediates in other research fields [8]. DNA-As-X is expected to contribute to computational biology and eliminate the hurdle of biological studies for veterans in engineering fields so that more novel means can be introduced and benefit genomic studies.

The generic framework of the DAX model consists of four main parts that represent four processing phases respectively, as shown in the dashed rectangles of Figure 2.1, including transformation, feature extraction, signal processing, and inverse transformation.

(1) Transformation. This phase is responsible for transforming the DNA sequences into signals, where encoding and signalizing may vary dependent on the selected encoding model.

Similar to the quantification from analog signals to digital signals, the transformation from traditional 1-D domain to 2-D spatial/temporal domain needs three essential steps: sampling, quantizing and encoding. In bioinformatics, because of the discreteness of DNA sequences, the sampling can be defined as a process to divide a character-based biological sequence into a sequence of reads each of which contain at least 1 character. In the same way, quantization and encoding in bioinformatics are respectively defined as a process to map a sequence of reads to a set of values in terms of a certain rule, and a process to designate a sequence of reads to a set of codes in terms of a certain rule.

Encoding designates codes to a sequence of reads while quantization maps the sequence of reads to values. Therefore, different from communication engineering, a proper order of three essential steps for transforming 1-D space to 2-D space in bioinformatics is sampling, encoding and quantizing. Additionally, for the generated 2-D domain, you can plot it on the plane of (x, y), where x is the spatial/temporal coordinate and y represents the magnitude of amplitude.

(2) Feature extraction. Features usually are hidden in the profiles that show some same or similar patterns. Exacting these common and subtle features is the task of this phase. Similar to pairwise alignment in bioinformatics, Common features can be extracted by matching two series of signal series and further form the coding vectors that can be processed on the stage of signal processing. These extracted features may be raw and unpruned, which will be further refined in the next phase.

(3) Signal processing [1]. It primarily takes care of outlining the desired patterns from those extracted raw features. Object X in this phase represents quantitative vectors. Because of the nature of vectors, they can form undirected trees/graphs where a vector may be contained in multiple paths. Chaining these vector into a larger path is the goal of global comparative methods [32]. The one with the maximum coverage will be selected as the best path. Similar to the aforementioned phases, the diverse methods of signal processing

[1]Since DAX is primarily based on information coding theory and signal processing, we use the term in signal processing for this phase.

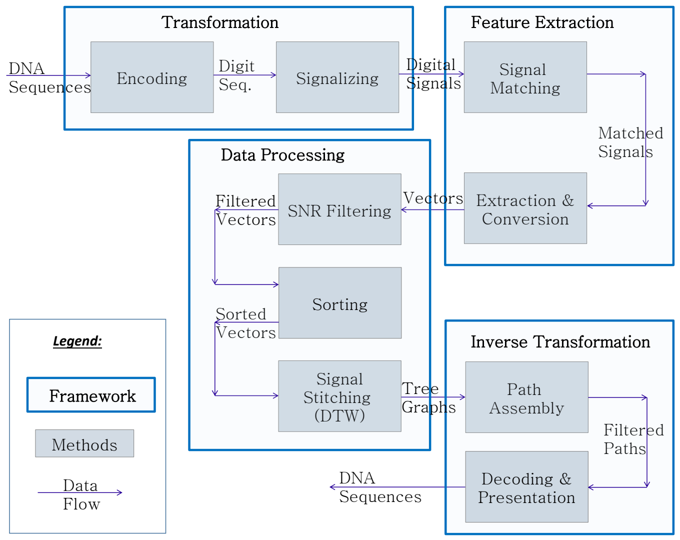


Figure 2.2: Methods Used in the Ontology of DNA-As-Signal

or pattern cognition can be adopted depending on particular cases. For example, in [32], vector-based algorithms are adopted for genome-wide sequence alignment to form paths in a graph network.

(4) Inverse transformation. As the counterpart of transformation, inverse transformation is responsible for converting the intermediate results, graphs, trees, paths, vectors, signals, codes and digits, back into human-readable character sequences, denoted as Equation 2.13. Thus, this phase is expected to implement the function of the reverse transformation of those intermediates and present the final sequences to end users.

As an ontology, Figure2.2 shows the primary procedures used in Section 2.3, reflecting the framework of DNA-As-X.

2.3 Signalign: An ontology of DNA-As-Signal

2.3.1 Bio-chemical Model

Assuming four nucletotides are distributed equiprobably, the entropy brings the maximum information capacity to the sequence. In terms of the definition of information entropy

$$H = -\sum_{i=1}^{4} p_i \log_2 p_i,$$

we have the optimum value of 2 bits. Compatible to modern computing system, we consider the Galois Field GF(2) and the extension of GF(4) for any GF(2) pair [7]. DNA sequences can be encoded to binary codes [33] based on the principles in information coding theory. Obviously, two important rules are needed to consider, nucleotide bio-chemical properties and the features of binary codes. According to the chemical and biological enthalpy values of the nearest nucleotide combinations [34], four nucleotides can be placed in the order of C, T, A, G so that the bio-chemical dynamics manifest the symmetric properties as shown in Figure 2.3. Also, in the ascendant order of molecular physical size and weight, C, T/U, A, G are the best placement corresponding to symmetric codes. Moreover, we observe that all bio-chemical representation models in Section 3.3 have the same order of C, T, A, G except EIIP's order of G, A, T, C (reverse order). By mapping these properties to features of binary numeric coding, eventually we encode C, T, A, and G to 00, 01, 10, and 11 respectively in two bits of binary codes.

On the other hand, the mutations/changes between four nucleotides differentiate the transition and the transversion. Relatively, transition (A-G, C-T) takes place more frequently than transversion (C-G, T-A) as the different colors (light-dark) respectively shown on the right of Figure2.3. Corresponding to the enthalpy values on the left of Figure2.3, we can see the weak bonds between pairs of transitions comparing with the strong bonds between nucleotide pairs of transversions. Mapping to the binary codes, the Hamming distance and the Euclidean distance between two codes precisely reflect the differences between transition

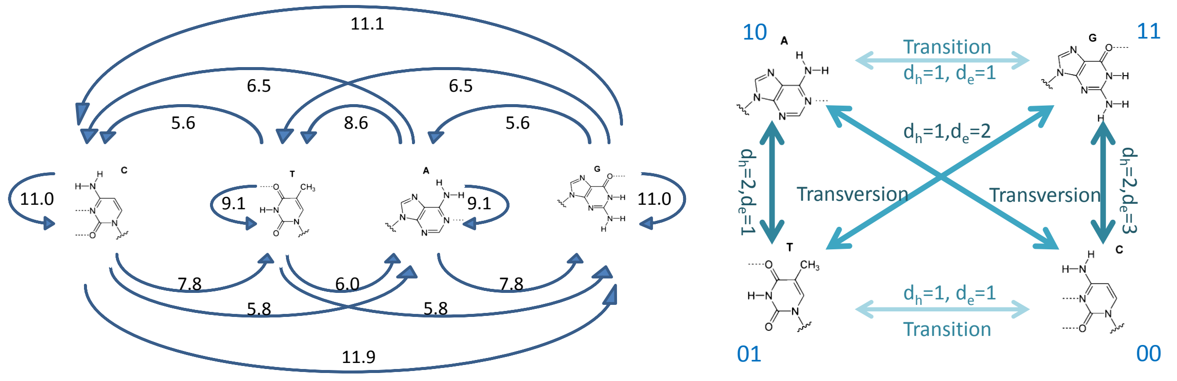


Figure 2.3: Symmetric Thermodynamics Pattern and Encoding Scheme

Left: Symmetric thermodynamics pattern (from left to right) in terms of the enthalpy values of thermodynamic interactions between two molecules [34]. Unit of measurement is k cal/mol. Right: Molecules are encoded for reflecting the bio-chemical relations [35].

and transversion.

2.3.2 Encoding and Signalizing Model

In terms of algebraic coding theory, a single nucleotide can be encoded into c that $c = \psi_1 x + \psi_0$, where ψ_1 and $\psi_0 \in GF(2)$. The Hamming distance (d_h) and the Euclidean distance (d_e) are defined for any two codes c and c' as Equation 2.1 (c, $c' \in GF(4)$).

$$d_h = c \oplus c', d_e = c - c'. \tag{2.1}$$

The matrices of d_h and d_e shown in Equations 2.2 and 2.3 reflect the collection of distances between any two codes c and c' in the order of C, T, A and G.

$$d_h = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}, \tag{2.2}$$

$$d_e = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}. \tag{2.3}$$

d_X is a mixed distance used to highlight the transversion by combining the Euclidean and Hamming distances. It is expressed in Equation 2.4 and its distance matrix is shown in Equation 2.5 in the order of C, T, A and G.

$$d_X = \max(d_h, d_e) = \max(c \oplus c', c - c'), \tag{2.4}$$

$$d_X = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}. \tag{2.5}$$

In practice, a masked distance d_K is necessary to represent the binary relations between different codes. $d_K = 1$ when $c \neq c'$ while $d_K = 0$ when $c = c'$. The distance matrix of d_K is shown in Equation 2.6.

$$d_K = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}. \tag{2.6}$$

Any nucleotide in a sequential context is encoded into that $c = \psi_{2i+1}x^{2i+1} + \psi_{2i}x^{2i}$ according to the extended $GF(4)$ where $i \in \{0, 1, ..., n-1\}$ indicates the location information of any nucleotide in this sequence.

Since ψ_i and $\psi'_i \in GF(2)$, $\psi_{2i+1}x^{2i+1} + \psi_{2i}x^{2i}$ and $\psi'_{2i+1}x^{2i+1} + \psi'_{2i}x^{2i} \in GF(4)$, $i \in \{0, 1, ..., n-1\}$ and n is the length of this DNA sequence, assuming that two DNA reads/tuples with the length of k are denoted as u and u' respectively and that $2k-1$ ($k>0$) is the degree of the expressed polynomial with coefficients from the extension of GF(2) [36], u and u' can be expressed as Equations 2.7 and 2.8.

$$\begin{aligned} u &= \psi_{2k-1}x^{2k-1} + \psi_{2k-2}x^{2k-2} + ... + \psi_1 x^1 + \psi_0 x^0 \\ &= \sum_{i=0}^{2k-1} \psi_i x^i, \end{aligned} \tag{2.7}$$

$$\begin{aligned} u' &= \psi'_{2k-1}x^{2k-1} + \psi'_{2k-2}x^{2k-2} + ... + \psi'_1 x^1 + \psi'_0 x^0 \\ &= \sum_{i=0}^{2k-1} \psi'_i x^i. \end{aligned} \tag{2.8}$$

The hamming distance and the Euclidean distance denoted as D_h and D_e between u and u' are

$$D_h(u,u') = \|u \oplus u'\| = \sum_{i=0}^{k-1} \left\|c_i \oplus c'_i\right\| = \sum_{i=0}^{k-1} d_{h,i}, \tag{2.9}$$

$$D_e(u,u') = \|u - u'\| = \sum_{i=0}^{k-1} \left\|c_i - c'_i\right\| = \sum_{i=0}^{k-1} d_{e,i}. \tag{2.10}$$

The mixed distance D_X between u and u' is

$$D_X(u,u') = \sum_{i=0}^{k-1} d_{X,i}. \tag{2.11}$$

The masked distance D_K between u and u' is

$$D_K(u,u') = \sum_{i=0}^{k-1} d_{K,i}. \tag{2.12}$$

Assuming that two DNA string sequences s and s' have the nucleotide lengths of n and n' respectively, w and w' are the numerical representations of string sequences s and s'. We denote the transformation and the inverse transformation as follows:

$$w = T(s), s = T^{-1}(w). \tag{2.13}$$

Assuming that u and u' are polynomials with the same degree of $2k-1$ ($k > 0$) and coefficients from the extension of $GF(2)$, w and w' can be represented as two series, $w = \{u_0, u_1, ..., u_{n-k}\}$ and $w' = \{u'_0, u'_1, ..., u'_{n'-k}\}$. Therefore, sequential codes u_{i-1}, u_i and u_{i+1} in sequence w can be expressed as the Equations 2.14 and 2.15.

$$\psi_{2(k+i)-1}x^{2k-1} + \psi_{2(k+i)-2}x^{2k-2} + \tfrac{u_{i-1}}{x^2} = u_i, \tag{2.14}$$

$$u_{i+1}x^2 + \psi_{2i+1}x + \psi_{2i} = u_i. \tag{2.15}$$

2.3.3 Error Tolerance and Detection for Feature Extraction

In recent studies, such as [21] [23], the discrimination of transition and transversion and the tolerance of multiple-loci errors were highlighted. Various methods are developed for this purpose. One of them is string mask that was included in Lastz [37]. In contrast, a binary mask in binary contexts is easier to develop. Another typical method is scoring and substitution matrix [24]. The magnitudes of substitution matrix are acquired through statistical experiments although they are limited by sample numbers and available alignments in species. The functions of scoring and substitution matrix may be replaced by distances in binary encoding contexts. For example, the matrices of Euclidean distance, Hamming distance, mixed distance and masked distance are able to measure the difference between sequences.

It is useful for $D_X(u, u')$ and $D_K(u, u')$ to identify the errors[2] between two nucleotide reads/messages with the length of k. Assuming that m_X and m_K indicate the mixed distance/error and the masked distance/error respectively tolerated by k-tuple messages/reads,

[2]Mutations, insertions-deletions, short gaps and small frameshifts are generalized as errors in the system.

by assigning the different condition (k, m_X, m_K), $0 \leq m_X \leq 3k$ and $0 \leq m_K \leq k$, we can have the desired tolerance distance in various patterns between reads. For example, $D_X(u, u') \leq m_X$ and $D_K(u, u') \leq m_K$ mean that the distances between u and u' are less or equal to m_X and m_K respectively.

For sequence $w = \{u_0, u_1, ..., u_{n-k}\}$ and sequence $w' = \{u'_0, u'_1, ..., u'_{n'-k}\}$ with the same polynomial degree for u and u', we denote $v_l = \langle u_{il,jl}, u'_{i'l,j'l}\rangle$ as a feature/signal vector that satisfies the condition (k, m_X, m_K) for sequential pairs of u and u' with the same length of l $(u \in [u_{il}, u_{jl}], u' \in [u'_{i'l}, u'_{j'l}])$. We further denote the set of all feature vectors as $F = \{v_0, v_1, ..., v_q\}$ $(q \geq 0)$. Therefore, the alignment between w and w' can be represented as a set f that contains a sequential series of vectors, $f \subseteq F$.

Feature extraction actually is a procedure of collecting sets of feature vector v before one conducts dynamic time warping to find the set f. One of the most commonly used method is to create a list to store all possible values of u. For any signal u, once it is located, its neighbor signals are tested to see whether it satisfies the condition (k, m_X, m_K). The vector v is the signal region that meets the criteria. Searching a signal u spends $O(\log n)$ time and the search for w to w' can finish in $O(n \log n)$ time, provided that they are of the same length n.

2.3.4 Dynamic Time Warping for Processing Sequential Series

Similar to Needleman-Wuncsh algorithm [29], Dynamic Time Warping (DTW) uses a 2-D table to find the optimal matching paths between series. In [38], a global constraint is adopted to reduce the computational load while in [39] the sparsity of matrix is utilized to simplify the computation. DTW can be computed in dynamic programming for the optimal path with the maximum scores. With some adjustments from the traditional method of DTW in signal processing and similar to [32], we adopt a dynamic programming algorithm to stitch the path of sequential time signals as shown in Algorithm 1.

First of all, the objective of signal stitching is to collect the optimal set of vectors where the signal vectors can form a stitching path. The graph structure of g that can be implemented in a multi-way tree by using pointers. The signal vector contains the spatial/temporal coordinates of two signals, the offset of two coordinates, the length of the signal, the Hamming distance, the Euclidean distance, the mixed distance, the masked distance and the signal-to-noise ratio (SNR) respectively.

Second, SNR is applied as the metric to measure the score of signal vector and path. Maximizing SNR is the measure to stitch the path. SNR is also regarded as a threshold to reduce the number of vectors that determines the computational load.

Third, the search constraint c in the algorithm limits the search distance away from the end of this signal vector. This is a strategy to reduce the computational load and keep the accuracy. c is set to $\min(|x|, |y|)/2$ [38], where $|x|$ and $|y|$ are the lengths of two sequences respectively. Additionally, since the signal vectors are sorted by $abs(x - y)$, x and y where x and y represent signal locations in two sequences respectively, stitching always starts from the most likely region to generate a graph containing most likely paths first.

Assuming that the number of vector v is r, the computing complexity of generating the graph containing all paths takes $O(r)$ (the property of $isParsed$ is set to 1 if it is touched). Calculating all paths from the graph may take the maximum time of $O(r^2)$ in the worse case depending on the structure of the graph. In practice, the number of calculated signal vector r is far less than the length of sequence n. Thus, in the worse case the time complexity for stitching is far less than $O(n^2)$.

2.3.5 Signal-to-Noise Ratio for Metrics

Signal-to-Noise Ratio (SNR) is a metric of scoring system in DNA-As-Signal to determine whether the signal vector/path meets the criteria. Its expression is shown in Equation 2.16 with the unit of measurement db.

$$\begin{aligned} SNR &= 20\log_{10}\tfrac{A_{signal}}{A_{noise}} \\ &\sim \log_{10}\tfrac{A_{signal}}{A_{noise}} = \log_{10}\tfrac{L}{D_X - D_K} \end{aligned} \tag{2.16}$$

Algorithm 1 Modified DTW Algorithm

```
1: $f \leftarrow$ <filter signal vectors>
2: $s \leftarrow$ <sort $f$> {Sorted by $abs(x-y)$,$x$,$y$}
3: $c \leftarrow$ constraint for search
4: $g \leftarrow 0$
5: $ModifiedDTW(prev)$
6: $i \leftarrow$ <seek next proper index of $prev$>
7: while $i < c$ do
8:   if $s[i].isParsed \neq 0$ then
9:     <continue>
10:  else
11:    $s[i].isParsed \leftarrow 1$
12:    $g \leftarrow s[i] +$ <max ModifiedDTW(i)>
13:  end if
14: end while
15: return $g$
```

where A_{signal} and A_{noise} are the amplitudes of signal and noise and here they are measured by the length of signal series L and the offset of the mixed distance D_X and the masked distance D_K. The reason of why we adopt the offset of D_X and D_K is that the offset can tolerate the effect of transition and highlight the difference between transversion and transition.

As a threshold, SNR determines how many signals are filtered from the final results. The proper magnitude of SNR is obtained from empirical training set. We test a training set of 13 genes and 56 cross-species and obtain the system SNR as 1.2 *db*. The result of acquiring the system SNR is shown in Figure2.4. The accumulated percentages of sensitivity and specificity have the maximum magnitude around SNR = 1.0 *db* to 1.3 *db*. Eventually, SNR = 1.2 *db* is chosen as the default system parameter.

2.4 Evaluation and Assessment

2.4.1 Data Set and System Configurations

The standard ROSETTA dataset contains totally 140 orthologous gene pairs and 1,160 cross-species exons from human and mouse [40]. Human and mouse both have the complicated gene structures: introns with various length are located between exons. It results

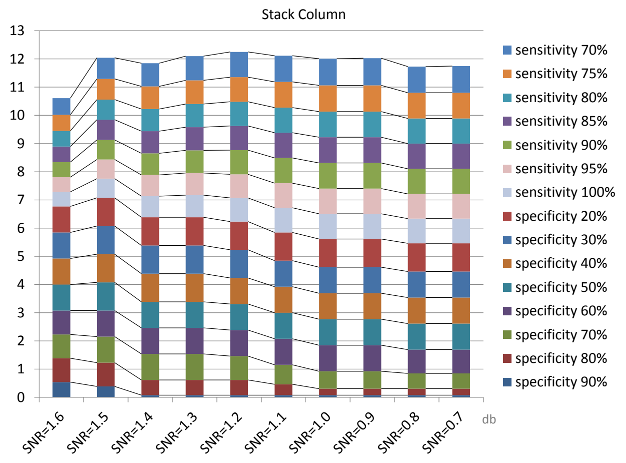


Figure 2.4: Stack Columns of Training Result for Different SNRs

The y-axis represents the magnitude of accumulations in specificity and sensitivity.

in that the degree of homologous regions varies a lot among these genes. Approximately 85% identity at DNA level shows in coding regions and only up to 27% identity exists in non-coding areas. The length of exons and introns also varies in a range from 3 *nt* to a few thousand *nt*. Human introns usually are 50% larger than mouse introns. All these biases make the data set suitable for our experiment.

Five widely used software including Avid [12], Blastn [13], Fasta36 [14], Ngila [15], Lastz [37] are utilized for the comparison with our program named Signalign. Both Blastn and Fasta36 are two of the most famous comparative tools for many years; Lastz origins from Blastz and currently is the core engine for UCSC [41]; Avid and Ngila are two popular global-alignment tools. All software are downloaded in the latest version of June, 2015 except Blastn that is a server version over Internet. All programs except Blastn are compiled in C/C++ and performed in a local system with Intel i7 1.8GHz, 8G RAM, 500G HD and Ubuntu 12.4. The default system parameters and configurations are adopted for all programs and the same criteria are applied to the evaluation of final results.

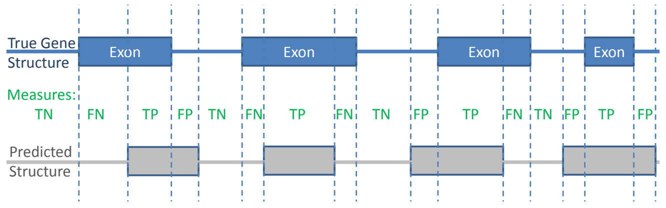


Figure 2.5: An Illustration of Evaluation Measures

2.4.2 Evaluation Measures

Sensitivity (Sn), specificity (Sp), accuracy (Acc) and Matthews correlation coefficient (Mcc) are used as evaluation measures [42]. Their expressions are shown as Equations 2.17, where true positive (TP) is the number of coding regions correctly predicted as coding; false positive (FP) is the number of non-coding regions incorrectly predicted as coding; true negative (TN) is the number of non-coding regions correctly predicted as non-coding; false negative (FN) is the number of coding regions incorrectly predicted as non-coding. A toy example is shown in Figure2.5 as an illustration. All these measurements are measured in nucleotide level.

$$\begin{aligned}
Sn &= \frac{\mathrm{TP}}{\mathrm{TP+FN}}\\
Sp &= \frac{TN}{TN+FP}\\
Acc &= \frac{TP+TN}{TP+FP+FN+TN}\\
Mcc &= \frac{TP\times TN-FN\times FP}{\sqrt{(TP+FN)\times(\mathrm{TN+FP})\times(\mathrm{TP+FP})\times(\mathrm{TN+FN})}}
\end{aligned} \tag{2.17}$$

2.4.3 Experimental Results

Due to the difference of gene structure, the predicted result varies for each pair of human and mouse sequences. Thus, we take into account the percentage of exons predicted in various coverage rates. For example, in the evaluation of sensitivity, we consider how

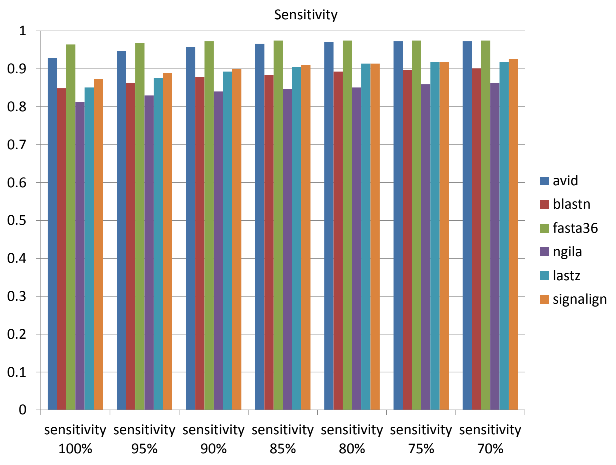


Figure 2.6: Comparison of Sensitivity

many percent of exons are predicted in 100% sensitivity, 95% sensitivity, 90% sensitivity, ..., 70% sensitivity respectively. Similar to sensitivity, other measurements such as specificity, accuracy and Matthews correlation coefficient adopt the percentage categories in order to explicitly demonstrate the distributions of evaluation measures.

From the results in Figure2.6, Avid and Fasta36 show the highest rank in sensitivity. Signalign is ranked in the third place followed by Lastz. Although Signalign is ranked in the middle place in sensitivity, the comparative results in specificity illustrate that Signalign has good performances to narrow the number of candidates for gene prediction in all categories of specificities as shown in Figure2.7.

Following the evaluations in sensitivity and specificity, two other comprehensive evaluation measures, accuracy and Matthews correlation coefficient, shown respectively in Figure2.8 and Figure2.9, illustrate that Signalign has the capability in gene structure prediction.

Blastn and Fast36 provide the local optimized alignments as the final results, which may make the specificity worse because the local alignments are often not optimal for global prediction even if the results are accurate locally. Moreover, even if one can adjust various parameters to filter the results, the final results may still be inferior to others. That is one of reasons why some alignment tools are not suitable to some particular comparative applications.

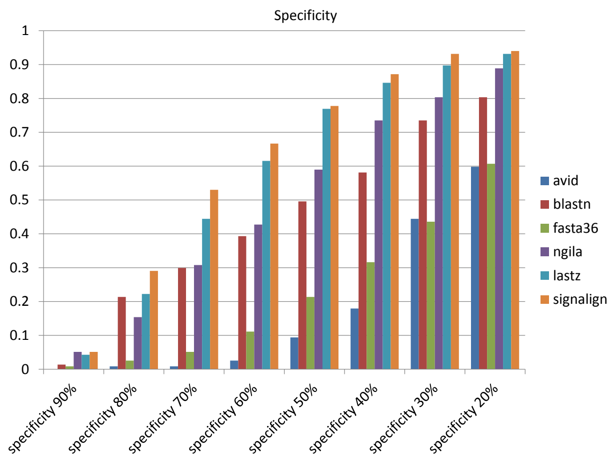


Figure 2.7: Comparison of Specificity

Although Avid and Ngila provide the global alignments as their final results, their performances show a large difference. For Avid, its results in specificity and in sensitivity are mostly complementary to each other. Since it is already a global result, less chance can affect its performance by adjusting its parameter setting. For Ngila, its performance is ranked in the third place according to the comprehensive evaluation. However, its sensitivity is ranked at the last place. Similar to the situation of Avid, adjusting its parameters cannot boost the performance.

As the engine of UCSC, Lastz shows good performance only following Signalign in these performance evaluations. And it does provide a bunch of parameters for various applications. But Lastz is a general alignment software, not a special software for gene structure analysis. Thus, it often needs additional work for tuning. It was criticized [23] for resorting to extra masks and complicated settings to improve its performance in homology prediction.

A comprehensive comparison is given in Figure2.10(a) by plotting TP rate against FP rate for all samples/genes in the data set. In order to illustrate the comparative details in each sample/gene, in Figure2.10(b), we draw the first nine samples and each subplot represents the receiver operating characteristic for a gene. The statistics of area under curve (AUC) are shown by drawing box plots in Figure2.10(c). It manifests that Signalign and Lastz have the similar performance, whereas Signalign has the higher average AUC 0.630175697 and

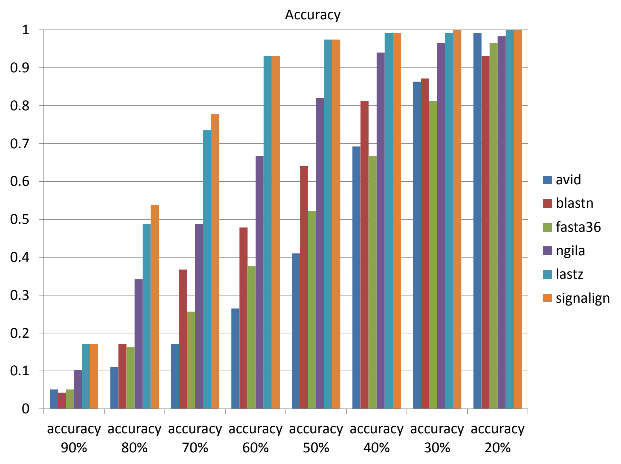


Figure 2.8: Comparison of Accuracy

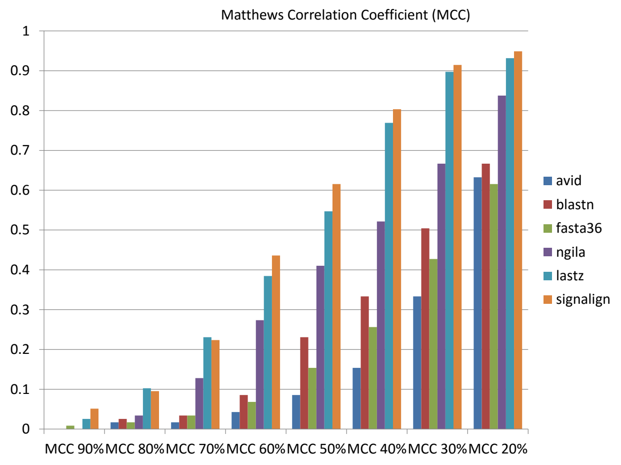


Figure 2.9: Comparison of Matthews Correlation Coefficient

Lastz has the lower average AUC 0.604215947, and the median AUC of Signalign is higher than that of Lastz as shown in the red bar in Figure2.10(c). However, the box length of Lastz is shorter than Signalign. It is caused by some outlier samples that play down the performance of Signalign. On the other side, the superiority of Signalign over Lastz can be better illustrated by the statistics in Figure2.10(d) if one counts the winners sample by sample, which approximately reflects the situation in Figure2.10(b) (Signalign wins 7 bids while Lastz wins 2 bids).

Through these evaluations, DNA-As-Signal demonstrates the potential capability to resolve the problem in biology by using the techniques in engineering, especially in information coding and processing. Certainly, although Signalign shows strong ability to predict gene structure using comparative methods, the extremely small exons still cannot be detected. Figure2.11 shows the original exon similarity, the predicted exon coverage and predicted exon similarity as well as the exon lengths for 39 exons. Signalign can detect almost various lengths of exons with the range from 20+ *nt* to 1,000+ *nt* except extremely short ones (3 *nt* and 6 *nt*).

For the comparison of execution time, Figure2.12 shows the running time spent on various numbers of genes for all software except Blastn. The tested data is around 2.5 Mega Bytes and the average gene pair is about 25 Kilo Bytes. Ngila and Fasta36 use the longest time followed by Avid while Signalign and Lastz are the top two who spend the least time.

2.5 Discussion

Signalign can be used to detect homologous sequences for comparative gene structure prediction. The experimental results manifest its potential capability in comparative gene analysis and processing. However, some questions are still remained for the future work, such as (1) the identification of splicing sites, (2) the genome-wide global search of gene structure, (3) the recognition of conserved non-coding region, (4) the improvement of sensitivity, and so forth. For the first question, detecting splicing sites can help narrow the comparative results and further accurately find the exact boundary of exon and intron. However, recog-

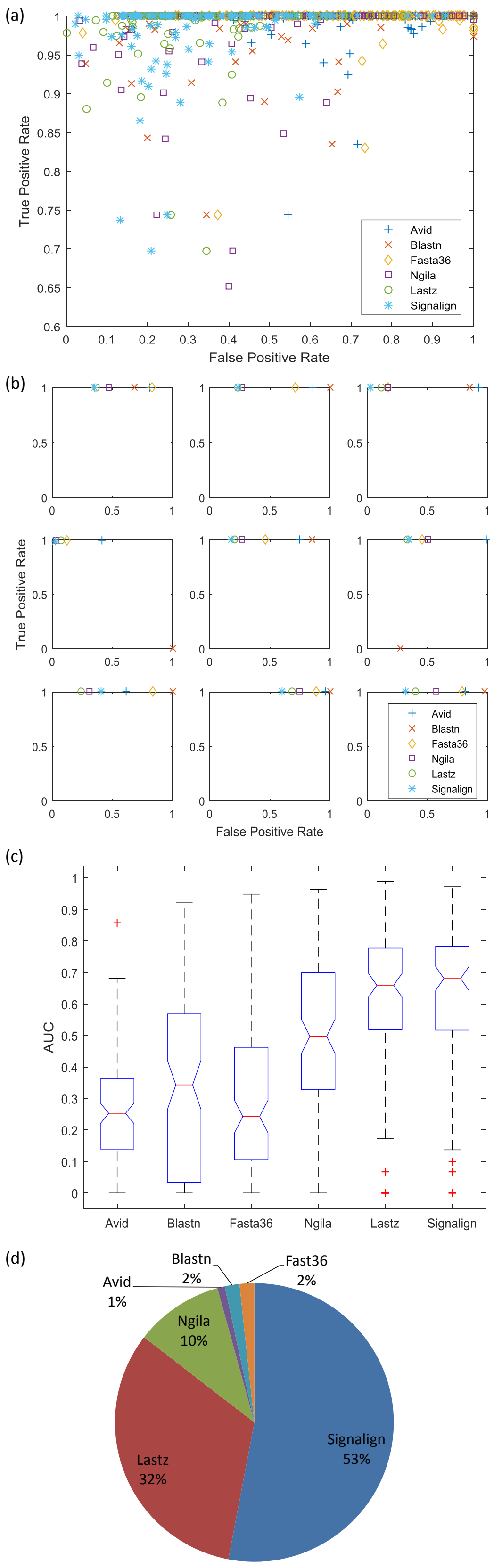


Figure 2.10: ROC Plots and Winning Rate

(a) ROC plots for all testing samples/genes. (b) ROC plots for the first 9 samples/genes from the data set. (c) AUC statistics. (d) Winning rate, when one counts the ROC/AUC winners gene by gene.

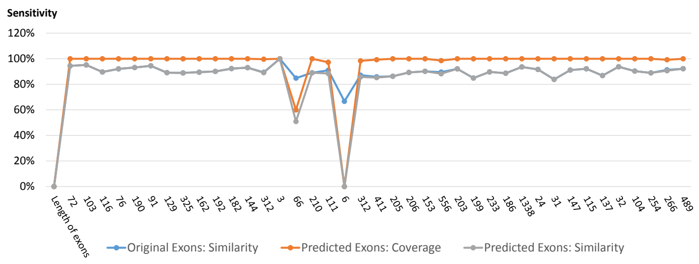


Figure 2.11: The Predictive Result for 39 Exons with Various Lengths

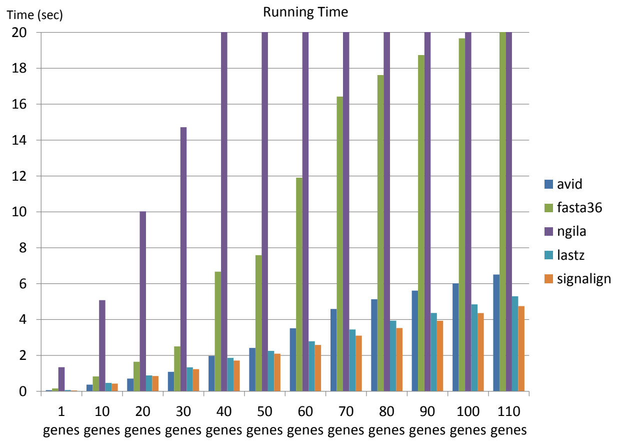


Figure 2.12: Comparison of Running Time

nizing patterns of splicing sites in their contexts needs some statistical knowledge, methods or models [43]. The gene structure prediction method is further improved in Chapter 3 by integrating deep learning method to splicing site detection. For the second question, the detection of gene structure for whole genomes will be our next task in the future. The current version of Signalign does not provide the optimized solution in memory efficiency that is indispensable component for genome-wide search. However, as a good trial for combining the information-coding-and-processing with comparative genomics, Signalign has shown good capability in evaluations and is expected to have the potential of being optimized to fit the memory efficiently in the future since intuitively its binary code has some nature connection with memory efficiency. For the third question, detecting and eliminating the conserved non-coding region is an important challenge to improve the specificity. In order to differentiate conserved coding regions from conserved non-coding regions, a number of statistical experiments and some data mining methods are needed. However, how to apply engineering techniques to these issues remains undiscovered. For the fourth question, sensitivity and specificity are frequently twisted together - one often improves the sensitivity while specificity drops; specificity is enhanced while sensitivity becomes worse. Thus, balancing sensitivity and specificity is a common strategy. However, it does not intend to blur the question - many good software can give attention to both two things. A viable way is to remove constraints in sensitivity while increasing the accuracy of predicted candidates. Searching certain biological patterns such as splicing sites sheds some light on it that is further developed in Chapter 3.

Additionally, although five software show inferior performances in the comparative experiments, it does not mean inferior in other applications and, in fact, each of them shows its strength in various aspects. For example, Avid and Ngila emphasize on the efficiency in global alignment; Blastn and Fasta36 are canonical local alignment software; Lastz manifests its versatility in a wide scope. By modifying and adjusting some functions, they are enabled to have better performance. For example, by introducing some constraints to global alignment and revising their programs, Avid and Ngila may obtain the most significant output as the candidates of gene structure prediction. But these modifications must be implemented by revising their programs or patching extra software.

CHAPTER 3

DEEP LEARNING TECHNIQUES FOR GENOME ANNOTATION

This chapter is composed of the following sections. First section describes problems and introduces the deep learning technology. Section 3.2 discusses previous *ab initio* methods in computationally predicting coding DNA sequences. Section 3.3 gives the primary encoding schemes in the field of genomic analysis. Section 3.4 depicts several deep learning algorithms based on artificial neural network. The subsequent section examines the performance of these deep learning algorithms and discusses canonical encoding schemes on these algorithms. Three applications, illustrated on subsections 3.5.1, 3.5.2 and 3.5.3, provide solutions respectively on detection of protein-coding splicing sites, recognition of lincRNA Transcription and improvement of gene structure prediction respectively. Among them, the last application is a hybrid method combining with Signalign described on Chapter 2.

3.1 Introduction and Contribution

DNA annotation is located at the central position of genomic studies. It refers to a process of identifying the locations of genes, coding regions and other specific locations that are important in DNA sequence. Although the first phase of Encyclopedia of DNA Elements (ENCODE) project has been claimed complete, the annotation of the functional elements is far from completeness [44]. Computational methods in gene identification will continue to play important roles in this area and relevant issues. So far, lots of work have been performed on this area and a plethora of computational methods and avenues have been developed. The methods for protein-coding DNA identification can be divided into three categories: *ab initio*, comparative, and hybrid methods.

ab initio method can detect genes by systematically examining and discriminating signal sensors as well as distinct biological patterns that distinguish gene regions in the single input sequence. The only criteria this method adopted to identify the genes rely on the extracted intrinsic information of DNA sequences.

Comparative methods are homology-based under the assumption that coding sequences are conserved more than non-coding genes are. These conserved areas can be detected by traditional local alignment methods, such as the canonical Smith-Waterman algorithm. In Chapter 2, the novel method named Signalign are based on homologous conservation on DNA sequence for gene structure prediction.

Hybrid methods integrate the advantages of *ab initio* and comparative methods into a particular application. The innovation of hybrid methods primarily relies on a novel combination of techniques in the two mainstream methods in order to achieve performance improvement in a particular application.

An important concern in DNA sequence annotation is about improving the accuracy of annotation. Belonging to the *ab initio* category, deep-leaning method is an emerging cut-edge technology with high prediction accuracy. The methods described in this chapter are derived from deep artificial neural network technology, one of deep learning techniques. Meanwhile, auto-encoder related techniques in deep neural network are primarily studied in this chapter.

Deep learning (DL) method has emerged as the state-of-the-art technique for genomic sequence analysis [45]. Deep Neural Network (DNN) is one of implementations in DL, which generally refers to methods that map data through multiple levels of feed-forward neural network to reveal some intractable and non-linear relation between input data and hidden factors and automatically learn complex functions [46].

Generically, in a deep learning model, the DNA sequences need to be encoded and converted into numeric sequences. After the data preparation, various deep learning methods use these pre-processed data as the input for training and further prediction. Depending on different data representation methods namely encoding schemes, various deep learning techniques can make a good deal of difference from each other on performance. Thus, studying the difference of combinations between deep learning methods and encoding schemes can help practitioners learn the characteristics of different method designs and acquire better performance for their research.

Deep learning models can be applied into a variety of applications. However, their basic procedures are quite similar. A generic procedure applied in this chapter is shown as Figure3.1 and its data flow chart is shown as Figure3.2

In this chapter, four goals are anticipated to achieve: (1) developing the deep learning methods for identifying coding DNA regions. (2) studying the difference between various deep learning methods. (3) discovering how encoding schemes could affect the performance of deep learning and finding the most appropriate encoding schemes for these applications. (4) improving the accuracy on identifying coding DNA sequences.

Deep-learning based methods and relevant studies for DNA annotation are contributed to the bioinformatics community. Meanwhile, the significance of encoding schemes are studied to unveil the influence to deep neural network. In addition, a few applications that use the deep-learning based method are discussed to illustrate its superior performance in improving the accuracy in bioinformatic research.

3.2 Previous Methods

Many *ab initio* methods largely depend on probabilistic models. Among them hidden Markov models (HMM) are the most generative model where the transitions of nucleotide over finite hidden states are ruled by the probabilities of present and previous appearances.

ab initio methods are indispensable for gene prediction because it uses statistical patterns and intrinsic information, especially signal sensors, to detect the boundaries of content and it can greatly increase the specificity of prediction performance. On the other side, one of disadvantages of *ab initio* methods is that it requires a large volume of training sets to collect the near-ground-truth statistical properties of various signal sensors, which inherently limits their applicability to low sample sets. Another disadvantage is that since the boundaries are often variable, it results in overfitting models on small training sets.

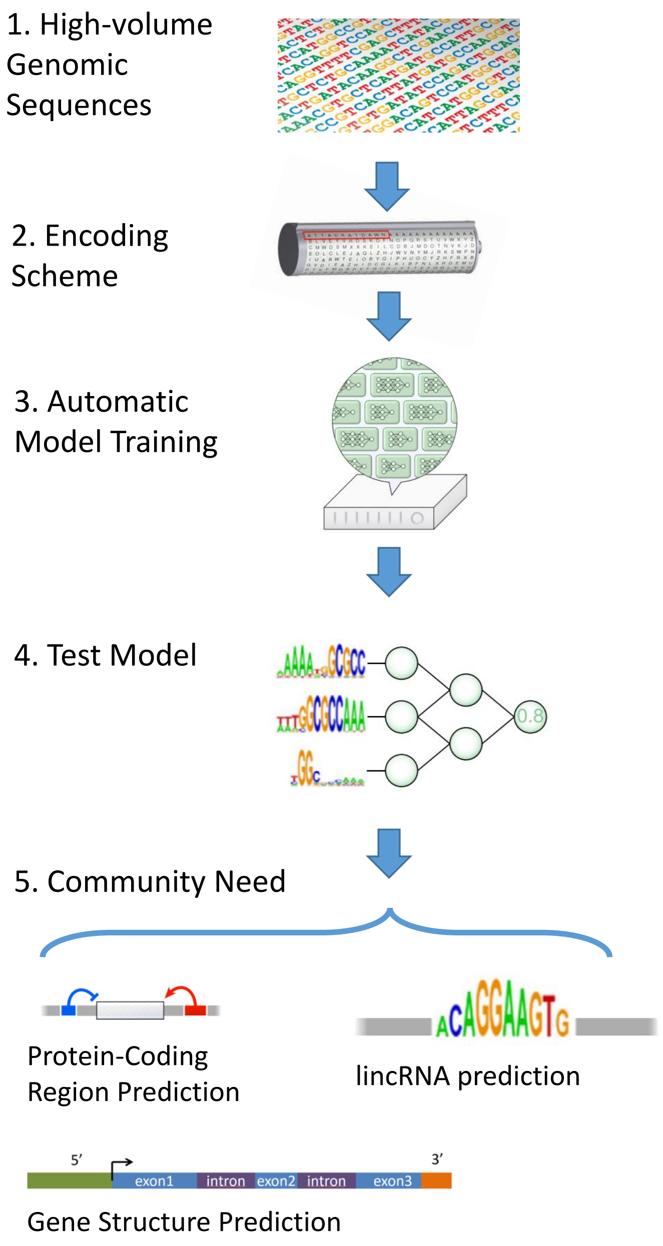

1. High-volume Genomic Sequences
2. Encoding Scheme
3. Automatic Model Training
4. Test Model
0.8
5. Community Need
Protein-Coding Region Prediction
lincRNA prediction
5'
3'
exon1
intron
exon2
intron
exon3
Gene Structure Prediction

Figure 3.1: An Illustration of Basic Steps of Deep Learning

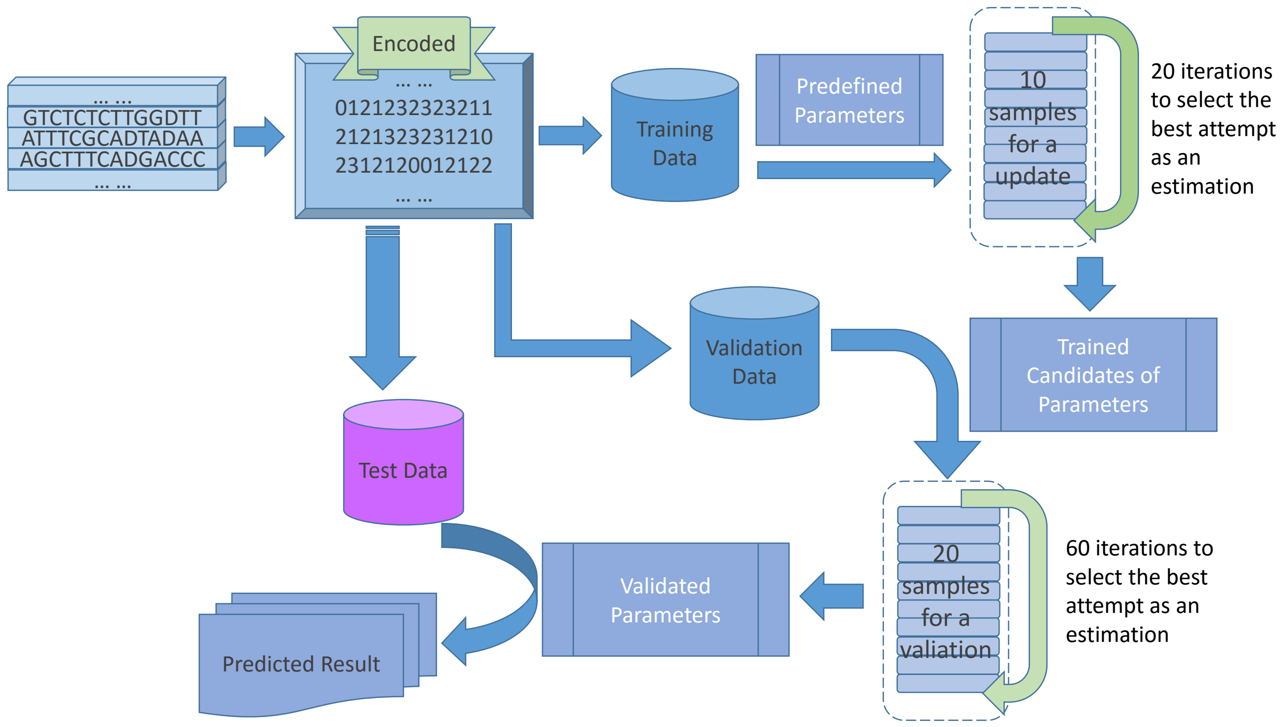


Figure 3.2: Details of Deep Learning Data Flow

3.2.1 Hidden Markov Model (HMM)

The prerequisite of HMM is based on an assumption that the probability of the appearance of a given nucleotide depends on its k previous nucleotides (k is the order of HMM), that is, the model is defined by conditional probabilities $P(X|k)$, where $X \in \{C, T, A, G\}$. A zero-order Markov model assume that each nucleotide occurs independently with a given frequency. The large-order Markov model can better characterize the dependencies between adjacent nucleotides. Most gene prediction methods are 5th-order Markov model that use the compositional words of 6 in gene characteristics. In [47], an observation is noticeable that models with an order higher than five does not make a distinct difference in discriminating coding and non coding regions.

Usually, a training set is necessary to estimate the state transition and nucleotide emission probabilities so that the HMM model can be built. Thus, given a genomic sequence, HMM model outputs the most probable path that generates the observed sequences using

Viterbi algorithm. The definition is as follows: Given a generated DNA sequence S of length L and a parse ϕ also of L, the conditional probability of ϕ can be computed using Bayesian Rule [48] [49].

$$P(\phi|S) = \frac{P(\phi, S)}{\sum\limits_{\varphi \in \phi(L)} P(\varphi, S)},$$

where $\phi(L)$ is the set of all parses of length L. Thus, given a particular DNA sequence S, the parse maximizes the most likelood of generating S.

Extensive models are further developed in gene prediction to improve HMMs. A typical model called generalized HMMs (GHMMs) [50] is most widely used, which extracts different regions into finite states and encapsulates syntactic and statistical properties of each regions into state transitions.

3.2.2 Neutral Network

Neutral Network copes with uncertain, imprecise and approximate problem to achieve robust and tractable outcomes. It is one of artificial intelligence techniques, which represents the learning process of human brain and includes supervised and unsupervised learning algorithms for gene prediction.

A neural network is applied to combine the feature selection output and to predict the location of coding regions. Neutral network takes a training procedure to learn how to deal with the output of feature selection and can make accurate decision about the location of coding regions. To determine the likelihood of a given sequence position, the neutral network extracts the weights of network from training procedure. For example, in Figure3.3, seven features are selected [51]: Frame bias matrix provides the usage of amino acid to calculate the correlation coefficient for reading frame; Fickett algorithm considers several properties of coding sequences; Dinucleotide fractal dimension represents the dinucleotide occurrence difference between that of intron and that of examined window; Coding 6-tuple work preferences examine the frequency of nucleotide words of a given length in a DNA

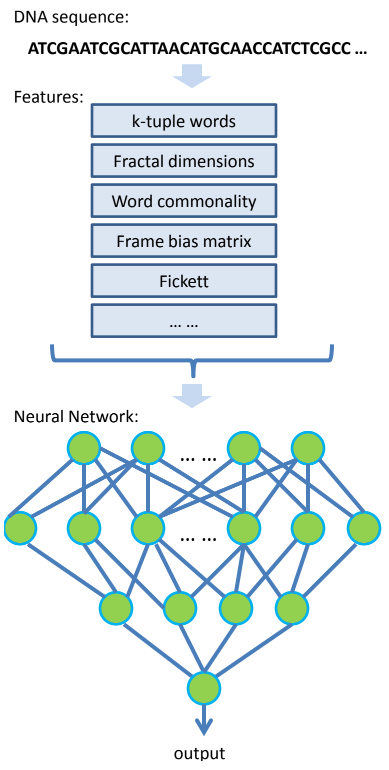


Figure 3.3: Schematic Diagram of Neutral Network Methods

genome; Coding 6-tuple in frame preferences are computed through the observed 6-tuple in coding DNA; Word commonality is calculated by summing all 6-tuple commonalities in the analysis window; Repetitive 6-tuple word preferences reflect the fact that highly repetitive DNA rarely encodes protein.

Comparing with deep neural network, conventional neural network can have the same architecture of multiple feed-forward layers. However, limited to the multiplication problem of the back-propagation, conventional neural network does not achieve the same accuracy as deep neural network because the latter usually has more complicated algorithms, such as auto-encoder and Boltzmann machine etc., that can constrain the error between layers and eliminate the back-propagation problem.

3.2.3 SVM-based

Support vector machines (SVM) and related kernel approaches have demonstrated their capability in accurate prediction of various functional DNA signal sensors/features, such as transcription start sites (TSS) and splice sites [52, 53, 54, 55, 56]. These approaches train support vector machines in the following procedures: (1) detection of interest site candidates, (2) training these candidate SVMs with features capturing patterns of site evolution and (3) scoring candidates of interest site [57]. After splice sites are detected, exon can be predicted by SVM exon model and subsequently transcripts are obtained by chaining exons.

In [58], a SVM-based two-layer approach is constructed,consisting of independent SVM signal and content detectors and hidden semi-Markov(HSM) SVMs. The former layer is SVM feature recognition while the latter one is gene structure reconstruction. The SVMs use task-specific string kernels, including spectrum kernel, the weighted degree (WD) kernel, the WD kernel with shift (WDS) and so forth [55]. The spectrum kernel counts all matching words so that SVM captures the typical sequence composition; the WD kernel considers matching words at the same position of sequences; WDS allows silghtly shifted matching [59]. HSM-SVM is similar to HMMs but it is trained discriminatively. High order content structure and length preferences are exploited and linked to transitions. A scoring function is utilized to comprehend different kinds of features at any position.

3.2.4 Digital Signal Processing

Due to the repetitive 3-periodicity of protein-coding regions, the open problem of gene finding can be dealt with by the methods in digital signal processing (DSP). In general, in order to analyze the DNA sequence, the symbolic-to-numeric transformation is necessary in the first step. Through numerical representations of DNA genome, DSP-based features are extracted, analyzed and classified in the spectral domain or the spacial-temporal domain.

The binary representation is mostly used to represent genome sequences, which converts a DNA sequence of four nucleotides C, T, A, and G into four separate binary sequences, $x_C[n]$, $x_T[n]$, $x_A[n]$, and $x_G[n]$ where 1 or 0 represents the presence or absence respectively in the corresponding positions. Other encoding schemes can also used as the descriptions in Chapter 2.

The most commonly used methods in spectrum analysis is the discrete Fourier transform (DFT) shown as follows.

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi nk/N)},\ 0 \leq k \leq N-1,$$

where $x[n]$ is a finite-length numerical sequence of length N. GeneScan program [48] calculates the signal-to-noise ratio of the peak at $k = N/3$ as $P = S[N/3]/\hat{S}$, where $\mathrm{S[k]} = \sum_{\mathrm{m}} |\mathrm{X_m[k]}|^2$, m=$\{C, T, A, G\}$ and $\hat{S}$ is the average of spectral content of S. In [48], P is assigned to 4 as a critical point where the bulk of coding sequences is distinct from almost 90% of non-coding regions having $P < 4$.

In [60], after a FIR band pass filter of order 8 with central frequency of $2\pi/3$ is applied to numerical sequences, an impulse train of periodicity-3 is multiplied to the filtered numerical sequences in order to emphasize the period-2 property in exonic region.

$$\mathrm{M_A} = \sum_{\mathrm{n=0}}^{\mathrm{N-1}} \mathrm{B_A[n]}\delta[\mathrm{n-3k}]$$

where δ is the pulse function, $\mathrm{B_A}$ is the filtered numerical sequence.

3.3 Encoding Schemes

Conventionally numerical representation and graphical representation are both non-character representation that can be summarized into a few categories: (1) Cartesian coordinate coding, (2) Binary linear code, and (3) Bio-chemical mapping.

First, most graphical representations and many numerical representations can be generalized into the points in Cartesian coordinate system. After the transformation, sequences are converted into a set of 2-dimensionl, 3-dimensional or even higher-dimensional points in Cartesian coordinate. Real representations [61] and complex representations [3] as well as quaternions [4] are in this category. For example, the complex representation [61] $A = 1 + j$, $C = -1 + j$, $G = -1 - j$, and $T = 1 - j$ is 2-dimensional numerical mapping. By choosing the different placement of vertices on a 2-dimensional Cartesian coordinate plane, encoding values for $\{A, C, G, T\}$ are different.

Second, Voss [62] proposed the simplest binary representation for DNA sequences by using four binary sequences for $\{A, C, G, T\}$ respectively and using 1 or 0 to denote the presence or absence for each corresponding nucleotide in the position. Voss representation has been widely accepted as a canonical numerical representation and applied to long-range fractal correlation analysis in DNA sequences and genomic signal processing, especially for discrete-Fourier-transform-related applications. Kent et al [41] use 2-bit format to compress and store the DNA sequences in a compact randomly-accessible format, which gives a 16-byte header to contain the encoding information and pack each DNA nucleotide to two bits per base, T:00, C:01, A:10 and G:11. However, this type of arbitrary assignment is criticized [6] for that it cannot provide real signals to understand biological research.

In binary linear code, a promising representation is about Galois Field encoding that was used in DNA computing. In [7], the encoding scheme was formalized to Galois Field where not only nucleotides are mapped to Galois Field $GF(4)$ but also all operations are restrained to $GF(4)$. It manifests the advantage of information coding in genome analysis where error-correcting coding structure reflects the nature of genome coding and it also shows the efficient effects on detecting genome redundancy and gene mutations.

Third, Bio-Chemical mapping uses the numerical representation to reflect the biological and chemical properties, complying with some commonly accepted rules that are regarded near the ground truth in biology and chemistry. Four typical representations are reviewed: (1) Atomic number [63], (2) Molecule mass [64], (3) Electron-ion interaction pseudopotentials (EIIP) [65] and (4) Thermodynamic values [34].

The single indicator of atomic number for nucleotide is assigned to each nucleotide: C=58, T=66, A=70, G=78. The nucleotide sequence, therefore, is converted into a series of numerical atomic indicators. In [63], this mapping was used to measure the fractal dimension difference between sequences of Human and Chimpanzee. It also gave a set of comparisons to show the diverse results when using different encoding schemes of numerical representations. Similar to atomic number, in terms of the mass of nucleotide molecules [3][64], a mass-based encoding scheme is generated as C=110, T=125, A=134, G=150.

The numerical representation scheme based on electron-ion interaction pseudopotentials (EIIP) for $\{A, C, G, T\}$ was first proposed in [65], where it aimed to replace the four binary indicator sequences proposed in Voss [62]. The energy of delocalized electrons in amino acid and nucleotides has been calculated as the electron-ion interaction pseudopotential that was used in Resonant Recognition Models (RRM) to substitute for the corresponding amino acid in protein sequences. The EIIP value indicators for nucleotides are G=0.0806, A=0.1260, T=0.1335, C=0.1340. If substituting the EIIP values to a DNA sequence, it can be converted into a series of EIIP numerical sequence that denotes the distribution of the free electron energy along the corresponding DNA sequence.

The thermodynamic enthalpy values between two neighboring nucleotides were studies in [6][66]. The encoding scheme is that each nucleotide pair is encoded to the enthalpy value in terms of the energy between the two nucleotides. Thus, DNA sequence is transformed into a numerical sequence that shows all enthalpy values of nucleotide pairs. In [6] the encoding scheme was used for searching certain bio-molecule patterns in DNA sequences.

In addition, three groups of nucleotides in terms of bio-chemical properties are important for encoding schemes [67]. They are: (1) purine $R = \{A, G\}$ and pyrimidine $Y = \{C, T\}$, (2) amino group $M = \{A, C\}$ and keto group $K = \{G, T\}$, (3) weak H-bonds $W = \{A, T\}$ and strong H-bonds $S = \{G, C\}$. They are widely considered by many encoding schemes. For example, in [68], local similarity/dissimilarity was studied in terms of these groups, by combining Chaos Game Representation [69] with the method of DNA walk [70].

From these bio-chemical schemes, we can observe that all these representations are full of sense in biology and chemistry. However, they do not consider the encoding properties in computer system and do not combine the bio-chemical sense with encoding schemes. It limits applicable scopes in computational biology.

Nine encoding schemes, respectively named DAX, arbitrary, EIIP, neural, complementary, enthalpy, entropy, statistic, and Galois, are selected in this project. Basically, DAX, arbitrary, neural and Galois are binary linear code; EIIP, enthalpy, entropy and statistic are bio-chemical mapping; complementary is Cartesian coordinate coding.

3.4 Deep Neural Network

3.4.1 Auto-encoder

Auto-encoder is an artificial neural network that can be used to constitute a multiple-layer percetron architectures for deep learning machince shown in Figure3.5(a). The hidden layer h and the iterative estimation of x^* can be expressed as Equation 3.1 by calculating the weights as illustrated in Figure3.5(b). The iteration becomes stable when it has the minimum distance between x and x^*, as shown in Equation 3.2. The preliminary ideas of shallow/deep neural network had been discussed for long time since 90s, however, mature concepts of deep learning including deep neural network were proposed in mid-2000s [71, 72, 73]. Since then, it has been applied to life sciences and shown tremendous promise [46, 74, 75, 45].

The simplest auto-encoder is based on a feedforward, non-recurrent neural network similar to the multiple-layer perceptron (MLP). The difference is that the output layer of auto-encoder has the same number of nodes as the input layer and an auto-encoder is trained to reconstruct their own inputs instead of being trained to predict the output value. Thus, training the neighboring set of two layers minimizes the errors between layers and eliminates the problem of error propagation that occurs in conventional neural network.

As the core of auto-encoder, the pseudo-code of cost update algorithm is shown in Algorithm 2 following the equations 3.1 and 3.2.

$$\begin{cases} h = f(x) = S_f(Wx + b_h) \\ x^* = g(h) = S_g(W'h + b_x) \end{cases} \tag{3.1}$$

$$\zeta_{DAE}(\theta) = \arg\min \sum\nolimits_{x \in X} E[L(x, x^*)] \tag{3.2}$$

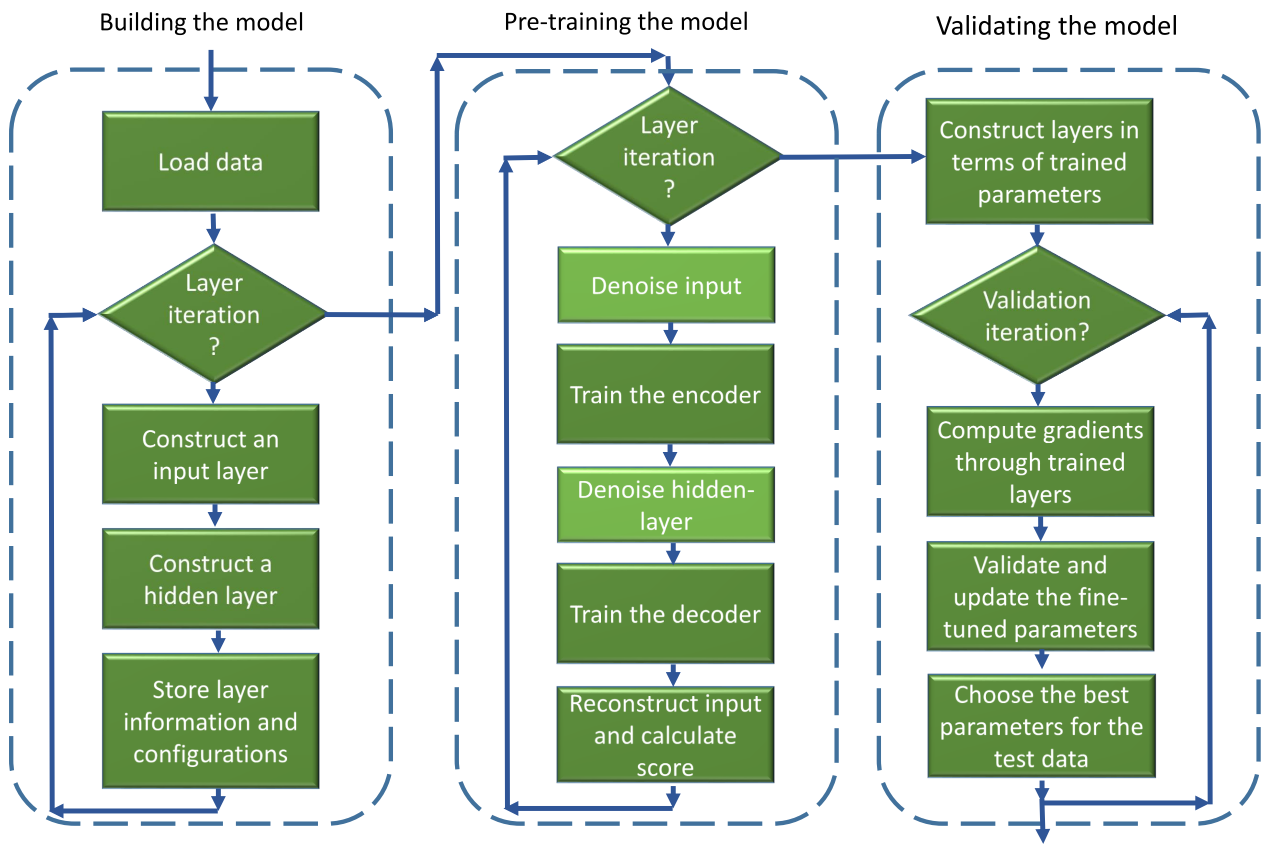

Building the model
Load data
Layer iteration ?
Construct an input layer
Construct a hidden layer
Store layer information and configurations
Pre-training the model
Layer iteration ?
Denoise input
Train the encoder
Denoise hidden-layer
Train the decoder
Reconstruct input and calculate score
Validating the model
Construct layers in terms of trained parameters
Validation iteration?
Compute gradients through trained layers
Validate and update the fine-tuned parameters
Choose the best parameters for the test data

Figure 3.4: Flow Chart for Auto-encoder Method.

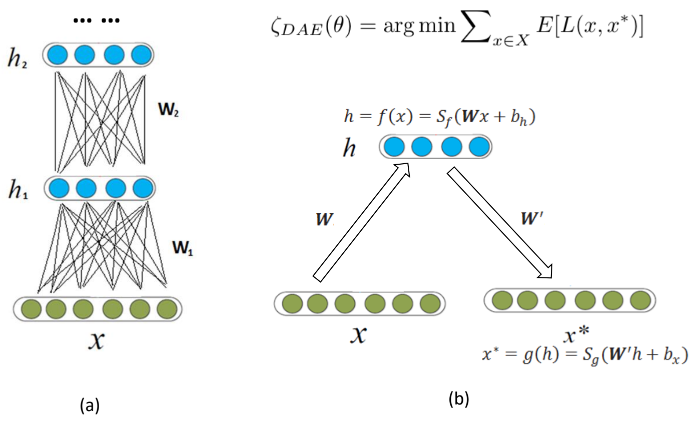

$\zeta_{DAE}(\theta) = \arg\min \sum_{x \in X} E[L(x, x^*)]$
h_2
W_2
h_1
W_1
x
(a)
$h = f(x) = S_f(\mathbf{W}x + b_h)$
h
W
W'
x
x^*
$x^* = g(h) = S_g(\mathbf{W}'h + b_x)$
(b)

Figure 3.5: Architecture of Deep Neural Network

(a) An Illustration of Deep Neural Network Architecture. (b) An Illustration of Auto-encoder.

Algorithm 2 Psudocode of Auto-encoder Cost Update Algorithm

```
1: $x \leftarrow$ <input matrix> //Input data
2: $p \leftarrow$ <parameter matrix> //Parameters
3: $y \leftarrow null$ //Vector for hidden layer
4: $z \leftarrow null$ //Reconstructed $x$
5: $h \leftarrow null$ //Vector for cross entropy
6: $c \leftarrow null$ //Vector for average cross entropy
7: $lr \leftarrow 0.8$ //Learning rate
8: $g \leftarrow null$ //Vector for gradient
9: $u \leftarrow$ <null matrix> //Updates of parameters
10: $l \leftarrow batch\ number$
11: $i \leftarrow 0$
12: while $i < l$ do
13:     $y =$ <gethiddenvalue( $x[i]$ )>
14:     $z =$ <getreconstructed( $y$ )>
15:     $h = -sum(x * \log(z) + (1 - x) * \log(1 - z))$
16:     $c = mean(h)$
17:     $g =$ <gradient( $c, p[i]$ )>
18:     $u[i] = p[i] - lr * g$
19: end while
20: return $u$
```

3.4.2 Denoising Auto-encoder

A denoising auto-encoder partially corrupts input data and uses the corrupted data for training in order to recover the original undistorted input. This technique can robustly obtain a corrupted input that will be useful for recovering the corresponding clean input. This definition includes the following implicit assumptions: The higher level representations are relatively stable and robust to the corruption of the input; It is necessary to extract features that are useful for representation of the input distribution. To train an auto-encoder for denoising data, it is necessary to perform preliminary stochastic mapping in order to corrupt the data and use as input for a normal autoencoder, with the only exception being that the loss should be still computed for the initial input instead of the corrupted one.

An auto-encoder takes an input x and first maps it to a hidden representation $y = f_\theta(x) = s(Wx + b)$, parameterized by $\theta =< W, b >$. The resulting latent representation y is then mapped back to reconstruct a vector $z \in [0, 1]^d$ in input space $z = g_{\theta'}(y) = s(W'y + b')$.

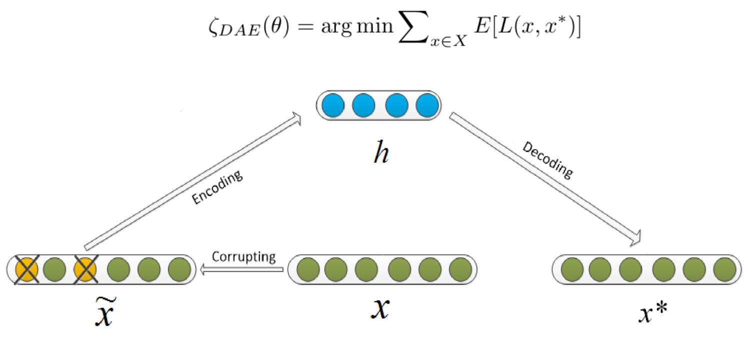

$\zeta_{DAE}(\theta) = \arg\min \sum_{x \in X} E[L(x, x^*)]$
Encoding
Decoding
Corrupting
h
$\tilde{x}$
x
x^*

Figure 3.6: An Illustration of Denoising Auto-encoder

The weight matrix W' can be constrained that $W' = W^T$, in which case the auto-encoder has tied weights. The network is trained such that to minimize the reconstruction error between x and z [76].

When the denosing auto-encoder is training input data, first x is corrupted into $\tilde{x}$, where $\tilde{x}$ is a partially denoised x by means of a stochastic mapping. Subsequently, y is computed as $y = s(W\tilde{x} + b)$ and z is computed as $s(W'y + b')$. The reconstruction error is now measured between z and the uncorrupted input x, which is computed as the cross-entropy : $-\sum_{k=1}^{d}[x_k \log z_k + (1 - x_k)\log(1 - z_k)]$ [77]. The pseudo code of DAE cost update algorithm is shown in Algorithm 3.

3.4.3 Hidden-layer Denoising Auto-encoder

Different from input-layer denoising, hidden-layer denoising auto-encoder model (HDAE) corrupts the units in hidden layer instead of input-layer and reconstructs the hidden layer. The architecture is shown as Figure3.7. Concretely, the hidden-layer is obtained in the same way as that of AE. Afterwards, the denoising in hidden layer occurs in binomial distribution. The corrupted hidden layer units are decoded into $\overline{x}$. Finally, vector $\overline{x}$ is encoded back into h^*. The cross-entropy is calculated as a measure to evaluate the minimum cost of h and h^*. The pseudo code of cost update algorithm is shown in Algorithm 4. Both encoding and decoding are twisted in auto-encoder models such as HDAE and DAE. Meanwhile, HDAE and DAE represent two components of double denoising auto-encoder described in the next

Algorithm 3 Psudocode of Denoising Auto-encoder Cost Update Algorithm

```
1: $x \leftarrow$ [input matrix] //Input data
2: $x_d \leftarrow$ [input matrix] //Denoised input data
3: $p \leftarrow$ [parameter matrix] //Parameters
4: $y \leftarrow null$ //Vector for hidden layer
5: $z \leftarrow null$ //Reconstructed $x$
6: $h \leftarrow null$ //Vector for cross entropy
7: $c \leftarrow null$ //Vector for average cross entropy
8: $lr \leftarrow 0.8$ //Learning rate
9: $il \leftarrow$ <getinputdenoise> //Get denoised level for input
10: $g \leftarrow null$ //Vector for gradient
11: $u \leftarrow$ [null matrix] //Updates of parameters
12: $l \leftarrow batch\ number$
13: $i \leftarrow 0$
14: while $i < l$ do
15:     $x_d[i] =$ <getdenoisedinput( $x[i], il$ )>
16:     $y =$ <gethiddenvalue( $x_d[i]$ )>
17:     $z =$ <getreconstructedinput( $y$ )>
18:     $h = -sum(x * \log(z) + (1 - x) * \log(1 - z))$
19:     $c = mean(h)$
20:     $g =$ <gradient( $c, p[i]$ )>
21:     $u[i] = p[i] - lr * g$
22: end while
23: return $u$
```

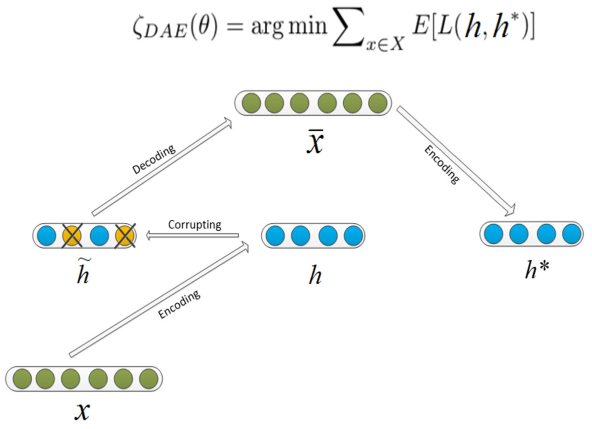


Figure 3.7: An Illustration of Hidden-layer Denoising Auto-encoder

subsection. That is, the input-layer denoising algorithm and the hidden-layer denoising algorithm are combined together for cost calculation in double denoising auto-encoder.

3.4.4 Double Denoising Auto-encoder

A deep neural network usually has a deep architecture that uses multiple layer to learn the feature representation of data and a representation learning procedure is used to discover multiple levels of representation of deep architecture: the higher the level, the more abstract the representation.

Figure3.8 illustrates the architecture of double denoising auto-encoder. An example x is stochastically corrupted to $\tilde{x}$. The auto-encoder then maps it to hidden representation h (via encoding) and attempts to reconstruct x via Decoding, producing reconstruction x^*. Reconstruction error is measured by loss $L(x, x^*)$. Meanwhile, the hidden representation h is also stochastically corrupted to $\tilde{h}$ and then $\tilde{h}$ is mapped to an intermediate reconstructed input $\bar{x}$ (via decoding) and attempts to reconstruct h via encoding, producing reconstruction h^*. Reconstruction error is measured by loss $L(h, h^*)$. Cross-entropy is measured to minimize the distances of $L(x, x^*)$ and $L(h, h^*)$ as shown in Algorithm 5.

Algorithm 4 Psudocode of Hidden-layer Denoising Auto-encoder Cost Update Algorithm

```
1: $x \leftarrow$ [input matrix] //Input data
2: $p \leftarrow$ [parameter matrix] //Parameters
3: $y \leftarrow null$ //Vector for hidden layer
4: $y_d \leftarrow null$ //Vector for denoised hidden layer
5: $v \leftarrow null$ //Reconstructed $\hat{x}$
6: $w \leftarrow null$ //Reconstructed $y$
7: $k \leftarrow null$ //Vector for cross entropy for hidden
8: $c \leftarrow null$ //Vector for average cross entropy
9: $lr \leftarrow 0.8$ //Learning rate
10: $il \leftarrow$ <getinputdenoise> //Get denoised level for input
11: $hl \leftarrow$ <gethiddendenoise> //Get denoised level for hidden layer
12: $g \leftarrow null$ //Vector for gradient
13: $u \leftarrow$ [null matrix] //Updates of parameters
14: $l \leftarrow batch\ number$
15: $i \leftarrow 0$
16: while $i < l$ do
17:     $y =$ <gethiddenvalue( $x[i]$ )>
18:     $y_d =$ <getdenoisedhidden( $y, hl$ )>
19:     $v =$ <getreconstructedinput( $y_d$ )>
20:     $w =$ <getreconstructedhidden( $v$ )>
21:     $k = -sum(y * \log(w) + (1 - y) * \log(1 - w))$
22:     $c = mean(k)$
23:     $g =$ <gradient( $c, p[i]$ )>
24:     $u[i] = p[i] - lr * g$
25: end while
26: return $u$
```

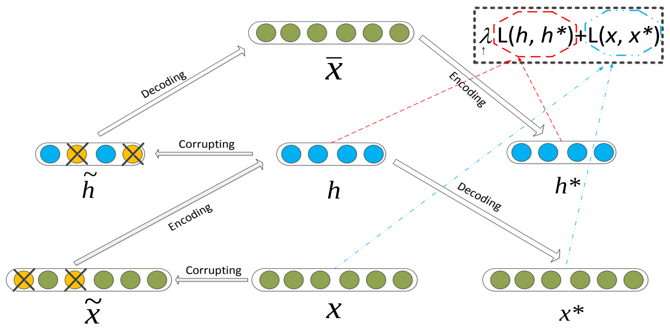
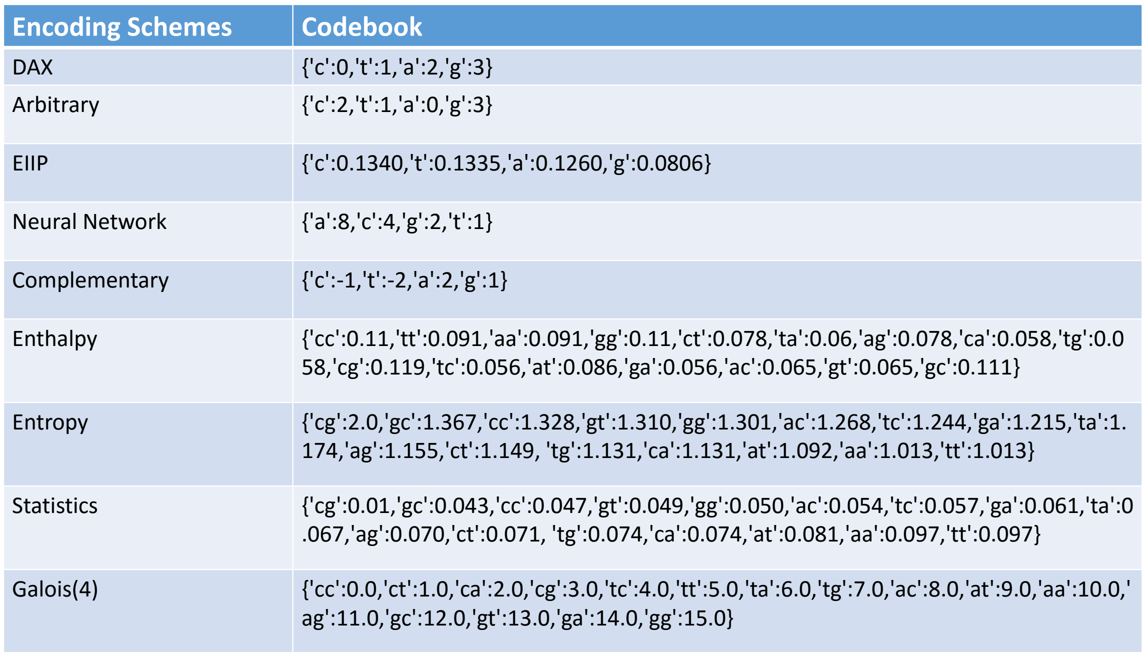


Figure 3.8: Double Denoising Auto-encoder
$\lambda = 0.0005$ is used in this study.

Algorithm 5 Psudocode of Double Denoising Auto-encoder Cost Update Algorithm

```
1: $x \leftarrow$ [input matrix] //Input data
2: $x_d \leftarrow$ [input matrix] //Denoised input data
3: $p \leftarrow$ [parameter matrix] //Parameters
4: $y \leftarrow null$ //Vector for hidden layer
5: $y_d \leftarrow null$ //Vector for denoised hidden layer
6: $z \leftarrow null$ //Reconstructed $x$
7: $v \leftarrow null$ //Reconstructed $\hat{x}$
8: $w \leftarrow null$ //Reconstructed $y$
9: $h \leftarrow null$ //Vector for cross entropy for input
10: $k \leftarrow null$ //Vector for cross entropy for hidden
11: $c \leftarrow null$ //Vector for average cross entropy
12: $lr \leftarrow 0.8$ //Learning rate
13: $cr \leftarrow 0.0005$ //Cost rate
14: $il \leftarrow$ <getinputdenoise> //Get denoised level for input
15: $hl \leftarrow$ <gethiddendenoise> //Get denoised level for hidden layer
16: $g \leftarrow null$ //Vector for gradient
17: $u \leftarrow$ [null matrix] //Updates of parameters
18: $l \leftarrow batch\ number$
19: $i \leftarrow 0$
20: while $i < l$ do
21:     $x_d[i] =$ <getdenoisedinput( $x[i], il$ )>
22:     $y =$ <gethiddenvalue( $x_d[i]$ )>
23:     $y_d =$ <getdenoisedhidden( $y, hl$ )>
24:     $z =$ <getreconstructedinput( $y$ )>
25:     $v =$ <getreconstructedinput( $y_d$ )>
26:     $w =$ <getreconstructedhidden( $v$ )>
27:     $h = -sum(x * \log(z) + (1 - x) * \log(1 - z))$
28:     $k = -sum(y * \log(w) + (1 - y) * \log(1 - w))$
29:     $c = cr * mean(h) + mean(k)$
30:     $g =$ <gradient( $c, p[i]$ )>
31:     $u[i] = p[i] - lr * g$
32: end while
33: return $u$
```



Encoding Schemes	Codebook
DAX	{'c':0,'t':1,'a':2,'g':3}
Arbitrary	{'c':2,'t':1,'a':0,'g':3}
EIIP	{'c':0.1340,'t':0.1335,'a':0.1260,'g':0.0806}
Neural Network	{'a':8,'c':4,'g':2,'t':1}
Complementary	{'c':-1,'t':-2,'a':2,'g':1}
Enthalpy	{'cc':0.11,'tt':0.091,'aa':0.091,'gg':0.11,'ct':0.078,'ta':0.06,'ag':0.078,'ca':0.058,'tg':0.058,'cg':0.119,'tc':0.056,'at':0.086,'ga':0.056,'ac':0.065,'gt':0.065,'gc':0.111}
Entropy	{'cg':2.0,'gc':1.367,'cc':1.328,'gt':1.310,'gg':1.301,'ac':1.268,'tc':1.244,'ga':1.215,'ta':1.174,'ag':1.155,'ct':1.149, 'tg':1.131,'ca':1.131,'at':1.092,'aa':1.013,'tt':1.013}
Statistics	{'cg':0.01,'gc':0.043,'cc':0.047,'gt':0.049,'gg':0.050,'ac':0.054,'tc':0.057,'ga':0.061,'ta':0.067,'ag':0.070,'ct':0.071, 'tg':0.074,'ca':0.074,'at':0.081,'aa':0.097,'tt':0.097}
Galois(4)	{'cc':0.0,'ct':1.0,'ca':2.0,'cg':3.0,'tc':4.0,'tt':5.0,'ta':6.0,'tg':7.0,'ac':8.0,'at':9.0,'aa':10.0,'ag':11.0,'gc':12.0,'gt':13.0,'ga':14.0,'gg':15.0}

Figure 3.9: Summary of Nine Encoding Schemes

3.5 Applications and Experimental Results

Here, three applications including protein-coding splicing sites detection, lincRNA transcriptional splicing sites detection, and improvement of gene structure prediction are described. Nine encoding schemes are applied to this study including DAX, Arbitrary, EIIP, Neural, Complementary, Enthalpy, Entropy, Statistic, Galois, which are briefly summarized in Figure3.9. The auto-encoder based model including original auto-encoder, denoising auto-encoder, hidden-layer denoising auto-encoder, and double denoising auto-encoder, are studied for discovering the relevance with diverse encoding schemes. Meanwhile, the comparison between deep neural network and conventional neural network is also performed.

3.5.1 Application I: Protein-coding Splicing Sites

Although alternative splice sites of exon/intron were discovered in recent literature [78], commonly generalized signals for splice acceptor and donor are AG and GT respectively. These splice sites are punctured along DNA sequences where transcription processes rely on these biological marks, and only 1% dimer AG/GT are identified as the real splice sites in

DNA sequence. Detecting splice sites [79] is an important subject in gene identification and gene structure studies.

The data sets are the standard benchmark from fruitfly.org for predicting gene splicing sites on human genome sequences [80] . The data set I is the Acceptor locations containing 6,877 sequences with 90 features. The data set II is the Donor locations including 6,246 sequences with 15 features. The Acceptor data sets have 70bp in the intron (ending with AG) and 20bp of the following exon. The Donor data sets have 7bp of the exon and 8bp of the following intron (starting with GT). The standard data sets contain real and fake splice sites and a window of +/- 40bp around the actual splice sites D (Donor) A (Acceptor). The data set of cleaned 269 genes is divided into a test and a training data set [80].

Experimental Results Table 3.1 and Table 3.2 show the performance of 2-layer auto-encoder. Complementary scheme shows the superiority over other schemes in Table 3.1 where the data set has more features than that in Table 3.2. DAX scheme shows the best performance in Table 3.2.

Table 3.3 and Table 3.4 show the performance of denoising auto-encoder. Denoising auto-encoder seems not fit to the application of DNA structure prediction because corrupted input data (DNA features) at each location may have a high dependency with others such that denoising makes the prediction messed.

Table 3.5 and Table 3.6 show the performance of hidden-layer denoising auto-encoder. Compared with the performance of input-layer denoising auto-encoder in Tables 3.3 and 3.4, in hidden-layer denoising auto-encoder model, corrupting some nodes on hidden layers makes a less impact than corrupting nodes on input layer. It is probably because in hidden layer some correlations/nodes may be so trivial to be denoised. Complementary scheme manifests its superiority over other schemes on more-feature data set while DAX and arbitrary schemes share the top rank on less-feature data set.

Table 3.7 and Table 3.8 show the performance of double denoising auto-encoder. Complementary encoding scheme continues keeping its superiority over other schemes in large-

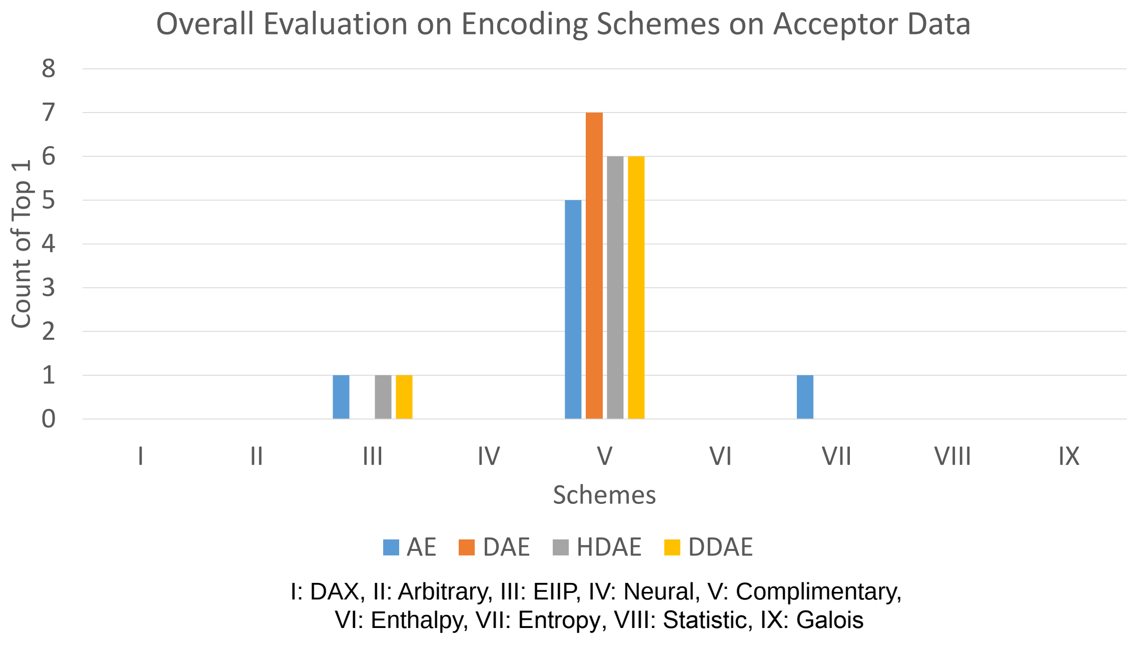


Figure 3.10: Overall Evaluation of Encoding Schemes on Accepor Data

feature data set while DAX and arbitrary scheme share the best performances on measurement in Table 3.8.

In addition, the legends in Figure3.1 and Figure3.2 are shared with those in figures from 3.3 to 3.8.

Discussion In this application, deep learning based methods are developed for detection of coding area splicing sites, including auto-encoding, denoising auto-encoder, hidden-layer auto-encoder and double denoising auto-encoder. Meanwhile, nine encoding schemes are studied for unveiling the best encoding schemes for DNA sequence analysis applications, including DNA-As-X (DAX), arbitrary mapping, EIIP, convention neural network scheme, complementary scheme, enthalpy scheme, entropy method, statistic mapping and Galois encoding. Complementary scheme shows its superiority over other schemes on more-feature data set as shown in Figure3.10 where it wins 24 out of 28 measurements. DAX slightly outperforms others on less-feature data set as shown in Figure3.11 where it totally wins 11 out of 28 measurements ranked at the first place while complementary scheme wins 8

Table 3.1 2-layer Auto-encoder Model on Acceptor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	13.7	11.6	9.8	0.0	14.7	3.3	0.2	3.2	0.0
FP	4.5	5.8	2.0	0.0	4.2	5.6	4.0	3.2	0.0
FN	5.6	7.6	9.5	19.3	4.6	16.0	19.1	16.1	19.3
TN	76.2	75.0	78.7	80.7	76.5	75.1	80.3	77.5	80.7

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	71.0	60.4	50.8	0.0	**76.2**	17.1	1.0	16.6	0.0
Sp	94.4	92.8	97.5	100.0*	94.8	93.1	**99.5**	96.0	100.0*
Acc	89.9	86.6	88.5	80.7	**91.2**	78.4	80.5	80.7	80.7
Mcc	66.9	55.3	59.1	–	**71.5**	14.1	2.8	20.3	–
Ppv	75.3	66.7	**83.1**	–	77.8	37.1	33.3	50.0	–
Pc	57.6	46.4	46.0	0.0	**62.6**	13.3	1.0	14.2	0.0
F1	73.1	63.4	63.0	0.0	**77.0**	23.4	2.0	24.9	0.0

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary,
VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel ^{a}: the measurement of methods.
TP: True positive.
FP: False positive.
FN: False negative.
TN: True negative.

Panel ^{b}: the evaluation of methods.
Sensitivity, $Sn = TP/(TP + FN)$
Specificity, $Sp = TN/(TN + FP)$
Accuracy, $Acc = (TP + TN)/(TP + FP + FN + TN)$
Matthews correlation coefficient,

$$Mcc = \frac{TP \times TN - FN \times FP}{\sqrt{(TP+FN) \times (TN+FP) \times (TP+FP) \times (TN+FN)}}$$

Positive predictive value, $Ppv = TP/(TP + FP)$
Performance coefficient, $Pc = TP/(TP + FN + FP)$
F1 score, the harmonic mean of precision and sensitivity,

$$F1 = 2 \times TP/(2 \times TP + FP + FN)$$

*: Not eligible for comparison due to training failure.
–: Invalid value.

Table 3.2 2-layer Auto-encoder Model on Donor Data

[c]	I	II	III	IV	V	VI	VII	VIII	IX
TP	17.1	17.1	10.4	17.4	17.2	15.9	15.6	15.0	0.0
FP	3.3	3.6	1.8	4.8	3.9	4.3	4.4	3.0	0.0
FN	4.0	4.0	10.7	3.7	3.9	5.2	5.5	6.1	21.1
TN	75.6	75.3	77.1	74.1	75.0	74.6	74.5	75.9	78.9

[d]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	81.0	81.0	49.3	82.5	**81.5**	75.4	73.9	71.1	0.0
Sp	95.8	95.4	**97.7**	93.9	95.1	94.6	94.4	96.2	100.0*
Acc	**92.7**	92.4	87.5	91.5	92.2	90.5	90.1	90.9	78.9
Mcc	**77.8**	77.0	58.6	75.0	76.6	71.0	69.7	71.5	–
Ppv	**83.8**	82.6	8.2	78.4	81.5	78.7	78.0	83.3	–
Pc	**70.1**	69.2	45.4	67.2	68.8	62.6	61.2	62.2	0.0
F1	**82.4**	81.8	62.5	80.4	81.5	77.0	75.9	76.7	0.0

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary, VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a] : the measurement of methods.
TP: True positive.
FP: False positive.
FN: False negative.
TN: True negative.

Panel [b] : the evaluation of methods.
Sensitivity, $Sn = TP/(TP + FN)$
Specificity, $Sp = TN/(TN + FP)$
Accuracy, $Acc = (TP + TN)/(TP + FP + FN + TN)$
Matthews correlation coefficient,

$$Mcc = \frac{TP \times TN - FN \times FP}{\sqrt{(TP+FN)\times(TN+FP)\times(TP+FP)\times(TN+FN)}}$$

Positive predictive value, $Ppv = TP/(TP + FP)$
Performance coefficient, $Pc = TP/(TP + FN + FP)$
F1 score, the harmonic mean of precision and sensitivity,

$$F1 = 2 \times TP/(2 \times TP + FP + FN)$$

*: Not eligible for comparison due to training failure.
–: Invalid value.

Table 3.3 Denoising Auto-encoder Model on Acceptor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	19.2	19.2	0.0	19.2	7.9	0.0	19.2	0.0	19.2
FP	80.8	80.8	0.0	80.8	1.2	0.0	80.8	0.0	80.8
FN	0.0	0.0	19.2	0.0	11.3	19.2	0.0	19.2	0.0
TN	0.0	0.0	80.8	0.0	79.6	80.8	0.0	80.8	0.0

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	100.0*	100.0*	0.0	100.0*	**41.1**	0.0	100.0*	0.0	100.0*
Sp	0.0	0.0	100.0*	0.0	**98.5**	100.0*	0.0	100.0*	0.0
Acc	19.2	19.2	80.8	19.2	**87.5**	80.8	19.2	80.8	19.2
Mcc	–	–	–	–	**54.3**	–	–	–	–
Ppv	19.2	19.2	–	–	**86.8**	–	19.2	–	19.2
Pc	19.2	19.2	0.0	19.2	**38.7**	0.0	19.2	0.0	19.2
F1	32.2	32.2	0.0	32.2	**55.8**	0.0	32.2	0.0	32.2

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary, VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a]: the measurement of methods.
Panel [b]: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

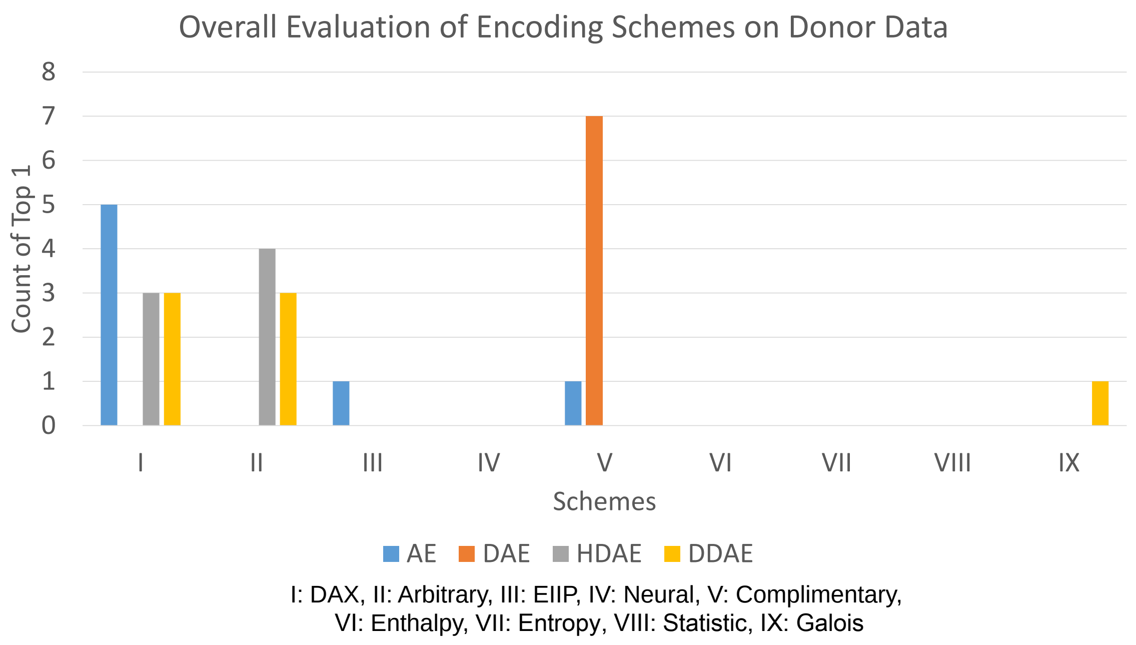


Figure 3.11: Overall Evaluation of Encoding Schemes on Donor Data

Table 3.4 Denoising Auto-encoder Model on Donor Data

a	I	II	III	IV	V	VI	VII	VIII	IX
TP	0.0	0.0	0.0	0.0	1.8	0.0	0.0	0.0	0.0
FP	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0
FN	21.1	21.1	21.1	21.1	19.3	21.1	21.1	21.1	21.1
TN	78.9	78.9	78.9	78.9	78.2	78.9	78.9	78.9	78.9

b	I	II	III	IV	V	VI	VII	VIII	IX
Sn	0.0	0.0	0.0	0.0	**8.5**	0.0	0.0	0.0	0.0
Sp	100.0*	100.0*	100.0*	100.0*	**99.1**	100.0*	100.0*	100.0*	100.0*
Acc	78.9	78.9	78.9	78.9	**80.0**	78.9	78.9	78.9	78.9
Mcc	–	–	–	–	**20.0**	–	–	–	–
Ppv	–	–	–	–	**72.0**	–	–	–	–
Pc	0.0	0.0	0.0	0.0	**8.3**	0.0	0.0	0.0	0.0
F1	0.0	0.0	0.0	0.0	**15.3**	0.0	0.0	0.0	0.0

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary, VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel ^{a}: the measurement of methods.
Panel ^{b}: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

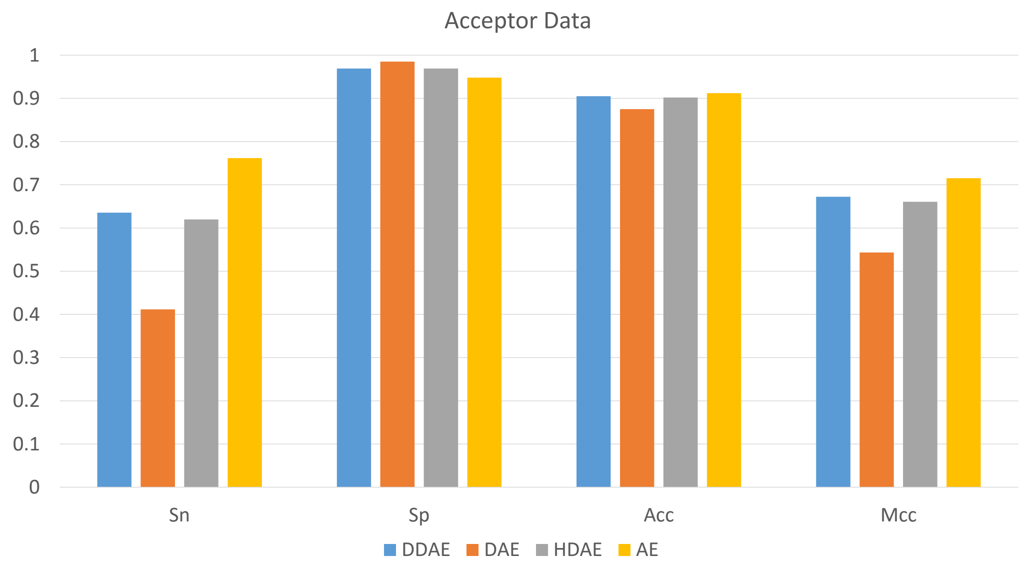


Figure 3.12: Overall Comparisons among Deep Learning Methods on Accepor Data

Table 3.5 Hidden-layer Denoising Auto-encoder Model on Acceptor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	10.4	0.4	0.0	10.3	11.9	0.0	0.0	0.0	10.6
FP	2.9	0.3	0.0	16.4	2.5	0.0	0.0	0.0	6.1
FN	8.8	18.8	19.2	8.9	7.3	19.2	19.2	19.2	8.6
TN	77.9	80.5	80.8	64.4	78.3	80.8	80.8	80.8	74.7

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	54.2	2.1	0.0	53.6	**62.0**	0.0	0.0	0.0	55.2
Sp	96.4	**99.6**	100.0*	79.7	96.9	100.0*	100.0*	100.0*	92.5
Acc	88.3	80.9	80.8	74.7	**90.2**	80.8	80.8	80.8	85.3
Mcc	58.7	8.1	–	29.7	**66.1**	–	–	–	50.3
Ppv	78.2	57.1	–	38.6	**82.6**	–	–	–	63.5
Pc	47.1	2.1	0.0	28.9	**54.8**	0.0	0.0	0.0	41.9
F1	64.0	4.0	0.0	44.9	**70.8**	0.0	0.0	0.0	59.1

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary, VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a]: the measurement of methods.
Panel [b]: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

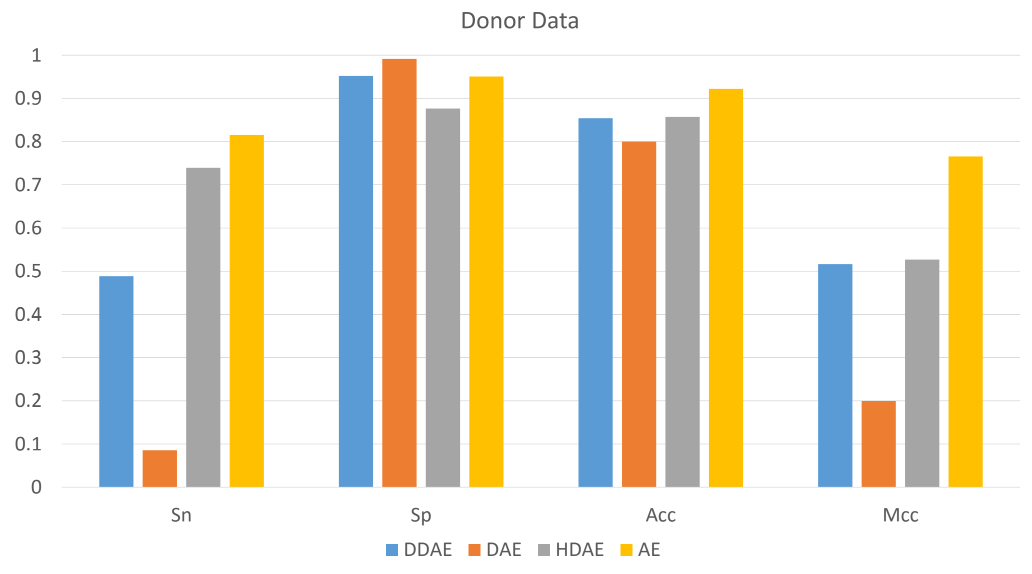


Figure 3.13: Overall Comparisons among Deep Learning Methods on Donor Data

Table 3.6 Hidden-layer Denoising Auto-encoder Model on Donor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	15.5	11.2	0.0	8.0	10.5	0.0	0.0	0.0	0.1
FP	9.1	9.9	0.0	13.1	10.6	0.0	0.0	0.0	0.0
FN	5.6	2.0	21.1	2.2	3.7	21.1	21.1	21.1	21.0
TN	69.8	76.9	78.9	76.7	75.2	78.9	78.9	78.9	78.9

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	73.5	**84.8**	0.0	37.9	73.9	0.0	0.0	0.0	0.5
Sp	88.5	**88.6**	100.0*	85.4	87.6	100.0*	100.0*	100.0*	100.0*
Acc	85.3	**88.1**	78.9	84.7	85.7	78.9	78.9	78.9	79.0
Mcc	58.7	**60.9**	–	47.3	52.7	–	–	–	6.1
Ppv	**63.0**	53.1	–	37.9	49.8	–	–	–	100.0*
Pc	**51.3**	48.5	0.0	34.3	42.3	0.0	0.0	0.0	0.5
F1	**67.8**	65.3	0.0	51.1	59.5	0.0	0.0	0.0	0.9

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary,
VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a]: the measurement of methods.
Panel [b]: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

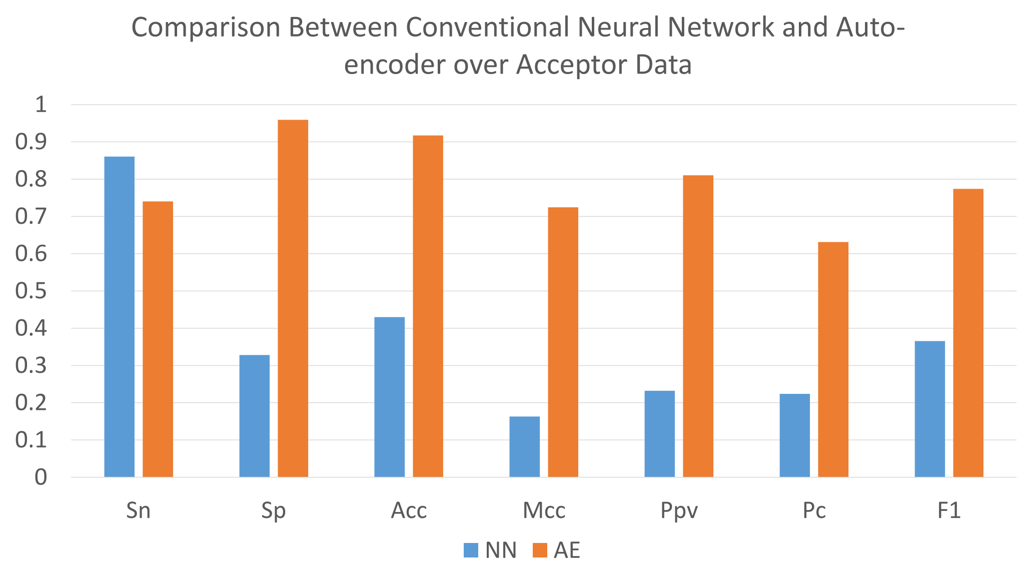


Figure 3.14: Performance Comparisons of Conventional Neural Network and Auto-encoder on Accepor Data

Table 3.7 Double Denoising Auto-encoder Model on Acceptor Data

a	I	II	III	IV	V	VI	VII	VIII	IX
TP	11	0.4	0.0	6.6	12.2	0.0	0.0	0.0	6.5
FP	3.1	0.3	0.0	8.5	2.5	0.0	0.0	0.0	1.6
FN	8.8	18.8	19.2	12.6	7.0	19.2	19.2	19.2	12.7
TN	77.7	80.5	80.8	72.3	78.3	80.8	80.8	80.8	79.2

b	I	II	III	IV	V	VI	VII	VIII	IX
Sn	55.6	2.1	0.0	34.4	**63.5**	0.0	0.0	0.0	33.9
Sp	96.2	**99.6**	100.0*	89.5	96.9	100.0*	100.0*	100.0*	98.0
Acc	88.2	80.9	80.8	78.9	**90.5**	80.8	80.8	80.8	85.7
Mcc	59.2	8.1	–	26.2	**67.2**	–	–	–	46.0
Ppv	78.0	57.1	–	43.7	**83.0**	–	–	–	80.2
Pc	48.0	2.1	0.0	23.8	**56.2**	0.0	0.0	0.0	31.3
F1	64.9	4.0	0.0	38.5	**72.0**	0.0	0.0	0.0	47.6

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary,
VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel ^{a}: the measurement of methods.
Panel ^{b}: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

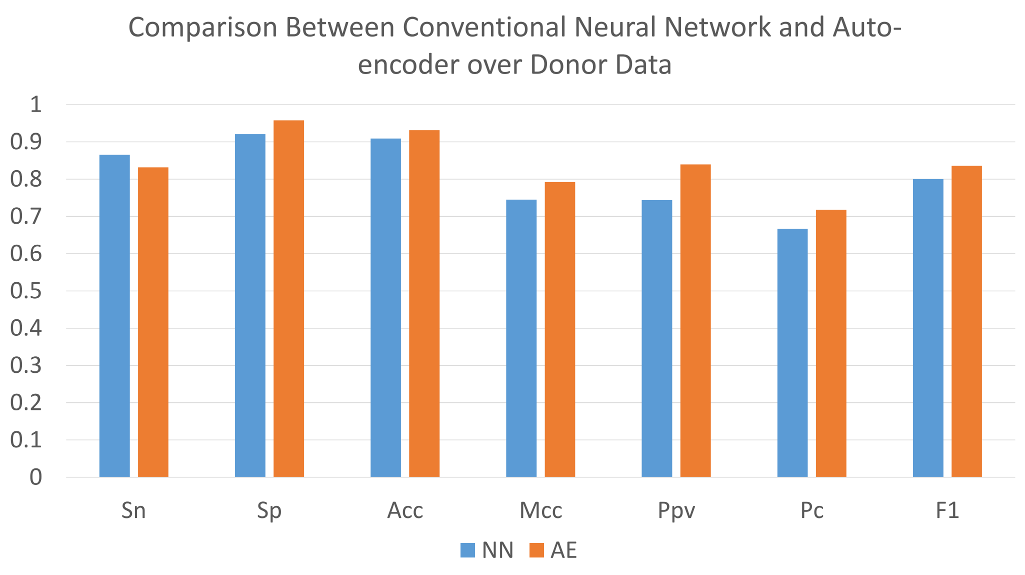


Figure 3.15: Performance Comparisons of Conventional Neural Network and Auto-encoder on Donor Data

Table 3.8 Double Denoising Auto-encoder Model on Donor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	15.5	11.1	0.0	8.2	10.3	0.0	0.0	0.0	3.3
FP	9.0	2.0	0.0	2.8	3.8	0.0	0.0	0.0	0.6
FN	5.6	10.0	21.1	12.9	10.8	21.1	21.1	21.1	17.8
TN	69.9	76.9	78.9	76.1	75.1	78.9	78.9	78.9	78.3

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	**73.5**	52.6	0.0	38.9	48.8	0.0	0.0	0.0	15.6
Sp	88.6	97.5	100.0*	96.5	95.2	100.0*	100.0*	100.0*	**99.2**
Acc	85.4	**88.0**	78.9	84.3	85.4	78.9	78.9	78.9	81.6
Mcc	58.9	**60.6**	–	46.1	51.6	–	–	–	31.4
Ppv	63.3	**84.7**	–	74.5	73.0	–	–	–	84.6
Pc	**51.5**	48.1	0.0	34.3	41.4	0.0	0.0	0.0	15.2
F1	**68.0**	64.9	0.0	51.1	58.5	0.0	0.0	0.0	26.4

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary, VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a]: the measurement of methods.
Panel [b]: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

evaluations.

Figure3.12 and Figure3.13 illustrate the comparisons among these deep neural network methods by selecting complementary scheme as its encoding. The 2-layer auto-encoder method shows better performances over others. It is probably because of close correlation and mutual interactions among nucleotide molecules so that denoising or corrupting any location may cause the deletion of dependency and downplay the performance of neural network.

Deep neural network and conventional neural network are compared on the standard data set while auto-encoder (a deep neural network method) and NNsplice [80] (a conventional neural network method) is chosen for this comparison. Figure3.14 and Figure3.15 show the comparative results. Auto-encoder shows superiority on data set with a large number of features while auto-encoder slightly outperform the conventional neural network on data set with a small size of features.

3.5.2 Application II: Detection of lincRNA Transcription Splicing Sites

LincRNA LincRNA refers to long intergenic non-coding RNAs with the length greater than 200 nucleotides that are transcribed from non-coding DNA sequences between protein-coding regions. Intergenic regions were referred as junk DNA, however, now it is known that intergenic regions can be transcribed and provide functional noncoding RNA genes within intergenic regions [81]. LincRNAs are frequently enriched for various classes of transposable elements and lincRNAs are viewed as elements with some regulatory functions in transcription and translation, for example some lincRNAs attach to messenger RNA to block protein production [82] and families of transposable elements-derived lincRNAs have been implicated in the regulation of pluripotency [83]. In addition, lincRNA is highly tissue-specific, which is frequently related to epigenetic regulation.

Similar to coding region transcription, non-coding regions are split at transcription splicing sites. However, regulatory RNAs rather than message RNAs are generated. That is, the transcribed RNAs participate the biological process as regulatory units instead of generating proteins. Thus, identifying these transcriptional regions is the first step towards lincRNA recognition. Similar to gene structures, lincRNAs have the complicated exon/intron structures, whereas the difference from gene structures is that many of them have two exons or three exons only.

LincRNAs are four times more than coding RNA sequences. However, currently only 21 thousand lincRNAs (about 2M bytes) are computationally discovered [81]. This is also one of the most important findings in lincRNA identification. The new identified lincRNAs are most from the analysis of RNA-seq transcript data. The basic procedure is composed of the following steps [81]: (1) collecting the RNA-seq transcript data from different tissues; (2) compiling the annotated ncRNAs to form known transcripts; (3) configuring filters to remove transcripts overlapping protein coding genes, known non-lincRNA noncoding RNA genes, pseudogenes, small ncRNAs with length less than 200 nt and any transcripts containing or overlapping an open reading frame (ORF) longer than 100 amino acids.

Deep learning related methods are barely seen in lincRNA annotation. Based on those annotated data, deep learning based methods can exert their capability in knowledge learning in order to improve the aforementioned method and discover novel lincRNAs in DNA genomes.

In this project, three goals are set. First, detecting lincRNA transcription splicing sites from the integenic areas. Second, validating the annotated lincRNAs splicing sites and testing the performance of deep learning method. Third, computationally discovering other unidentified splicing sites. For the first goal, auto-encoder method achieves 100% prediction accuracy illustrated in next subsection. For the second and third goal, one unreported splicing site is found by re-scanning the whole human genome through the deep learning method.

Benefiting from the increasing annotation data in lincRNAs, lincRNA's transcriptional splicing site sequences are collected from the annotated human DNA genome data. However, the annotated data sets of lincRNAs are not so many as that of mRNAs. Thus, all of annotated lincRNAs are used for training and testing.

In the same vein to detection of protein-coding splicing sites, auto-encoder neural network method is used for the lincRNA application. In terms of the discussion in application I, a 2-layer auto-encoder model is used for lincRNA detection and encoding schemes are used for evaluating the best performance. This is the first application to adopt the deep learning techniques for identifying lincRNA transcription splicing sites. The experimental results show an excellent predictive performance of deep neural network method on lincRNA data sets.

Results and Discussion Totally 46,983 lincRNAs' splicing site sequences with 90 features and are selected as acceptor data and 89,287 lincRNAs' splicing site sequences with 15 features are classified as donor data. According to the aforementioned experiments, 2-layer auto-encoder neural network method shows the best performance. Thus, it is chosen as the deep learning method for identifying lincRNA transcriptional splicing sites. Nine encoding schemes are still used for comparisons of different encoding performances.

Table 3.9 and Table 3.10 respectively show the comparison results for the two data sets. It shows that 100% predictive rate of deep neural network method with complementary encoding scheme on the acceptor data, meaning that complementary scheme has the strong ability on more-feature data sets. Similar performances among all encoding schemes show the similar ability on less-feature data set.

Figure3.16 shows an unreported splicing site is found by re-scanning the whole human genome through the deep learning method, which is located at 90,763,154 chromosome 12 (hg38) within the annotated lincRNA chr12_90761911_90806776. This result is based on the aforementioned deep learning method that was tested with 100% accuracy in acceptor data set.

This experiment indicates that deep learning method has the extensive ability for lincRNA splicing site prediction. In the future, related methods will be developed for more applications in this area.

Table 3.9 Results on lincRNA Acceptor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	49.4	49.4	49.0	49.4	49.4	49.4	46.0	49.4	49.4
FP	0.0	50.6	0.2	50.6	0.0	1.4	2.0	1.6	50.6
FN	0.0	0.0	0.4	0.0	0.0	0.1	3.4	0.0	0.0
TN	50.5	0.0	50.4	0.0	0.6	49.2	48.6	49.0	0.0

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	**100.0**	100.0*	99.2	100.0*	**100.0**	99.9	93.1	99.9	100.0*
Sp	99.9	0.0	99.6	0.0	**100.0**	97.2	96.0	96.8	0.0
Acc	100.0	49.4	99.4	49.4	**100.0**	98.5	94.6	98.3	49.4
Mcc	99.9	–	98.8	–	**100.0**	97.1	89.2	96.7	–
Ppv	99.9	49.4	99.6	49.4	**100.0**	97.2	95.8	96.8	49.4
Pc	99.9	49.4	98.8	49.4	**100.0**	97.1	89.5	96.7	49.4
F1	**100.0**	66.1	99.4	66.1	**100.0**	98.5	94.5	98.3	66.1

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary,
VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a]: the measurement of methods.
Panel [b]: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

Table 3.10 Results on lincRNA Donor Data

[a]	I	II	III	IV	V	VI	VII	VIII	IX
TP	7.7	10.7	9.0	8.9	8.5	11.2	7.7	10.2	0.0
FP	2.1	3.3	2.7	2.9	2.8	4.5	2.1	4.0	0.0
FN	6.7	3.7	5.4	5.5	5.9	3.2	6.7	4.2	14.4
TN	83.5	82.3	82.9	82.7	82.8	81.1	83.5	81.6	85.6

[b]	I	II	III	IV	V	VI	VII	VIII	IX
Sn	53.2	74.0	62.5	61.5	58.8	**78.1**	53.3	71.0	0.0
Sp	**97.6**	96.1	96.9	96.7	96.7	94.8	97.5	95.3	100.0*
Acc	91.2	**92.9**	91.9	91.6	91.2	92.4	91.2	91.8	85.6
Mcc	60.1	**71.0**	64.9	63.5	61.5	70.2	60.1	66.6	–
Ppv	**78.6**	76.3	77.1	75.6	75.0	71.5	78.5	71.9	–
Pc	46.5	**60.2**	52.7	51.3	49.1	59.5	46.5	55.5	0.0
F1	63.5	**75.1**	69.0	67.8	65.9	74.6	63.5	71.4	0.0

I: DAX, II: Arbitrary, III: EIIP, IV: Neural, V: Complimentary, VI: Enthalpy, VII: Entropy, VIII: Statistic, IX: Galois

Panel [a]: the measurement of methods.
Panel [b]: the evaluation of methods.
*: Not eligible for comparison due to training failure.
–: Invalid value.

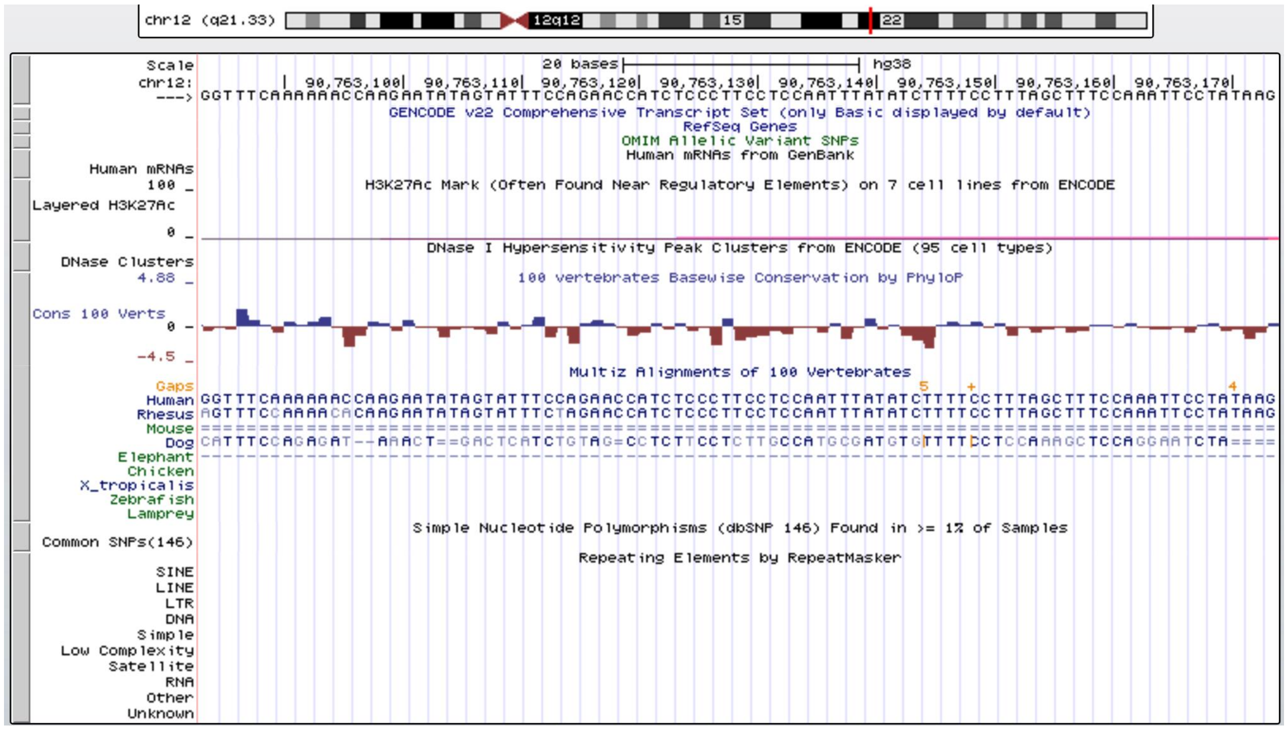

chr12 (q21.33)
12q12
15
22
Scale
20 bases
hg38
chr12:
90,763,100
90,763,110
90,763,120
90,763,130
90,763,140
90,763,150
90,763,160
90,763,170
GGTTTCAAAAAACCAAGAATATAGTATTTCCAGAACCATCTCCCTTCCTCCAATTTATATCTTTTCCTTTAGCTTTCCAAATTCCTATAAG
GENCODE v22 Comprehensive Transcript Set (only Basic displayed by default)
RefSeq Genes
OMIM Allelic Variant SNPs
Human mRNAs from GenBank
Human mRNAs
H3K27Ac Mark (Often Found Near Regulatory Elements) on 7 cell lines from ENCODE
Layered H3K27Ac
100
0
DNase I Hypersensitivity Peak Clusters from ENCODE (95 cell types)
DNase Clusters
4.88
100 vertebrates Basewise Conservation by PhyloP
Cons 100 Verts
0
-4.5
Multiz Alignments of 100 Vertebrates
Gaps
Human
Rhesus
Mouse
Dog
Elephant
Chicken
X_tropicalis
Zebrafish
Lamprey
Simple Nucleotide Polymorphisms (dbSNP 146) Found in >= 1% of Samples
Common SNPs(146)
Repeating Elements by RepeatMasker
SINE
LINE
LTR
DNA
Simple
Low Complexity
Satellite
RNA
Other
Unknown

Figure 3.16: An Unidentified lincRNA Acceptor Splicing Site

3.5.3 Application III: Improvement for Gene Structure Prediction

Hybrid Method The methodology design of DNA annotation mainly relies on genetic characteristics in gene structure, such as promoter, GC content, start and stop codon, coding region, splicing sites, exon and intron length, and compositional properties of coding and non-coding. The information are further integrated into almost all computational approaches as the criteria of determining entire gene structures. However, identifying those real signal sensors is difficult because genomic sequences contain thousands of similar signals/noises that fake themselves in DNA texture. Moreover, some signal sensors are not exactly validated. For example, GC content and TATA box are thought of the important markers in promotor, however, recent research show that TATA box is not present in all Eukaryotic promoters and about 45% promoters contain TATA box [84]. Similarly, GC content manifests various levels in promoter regions [85]. In order to deal with these difficulties, hybrid computing techniques are adopted to increase the accuracy and the specificity.

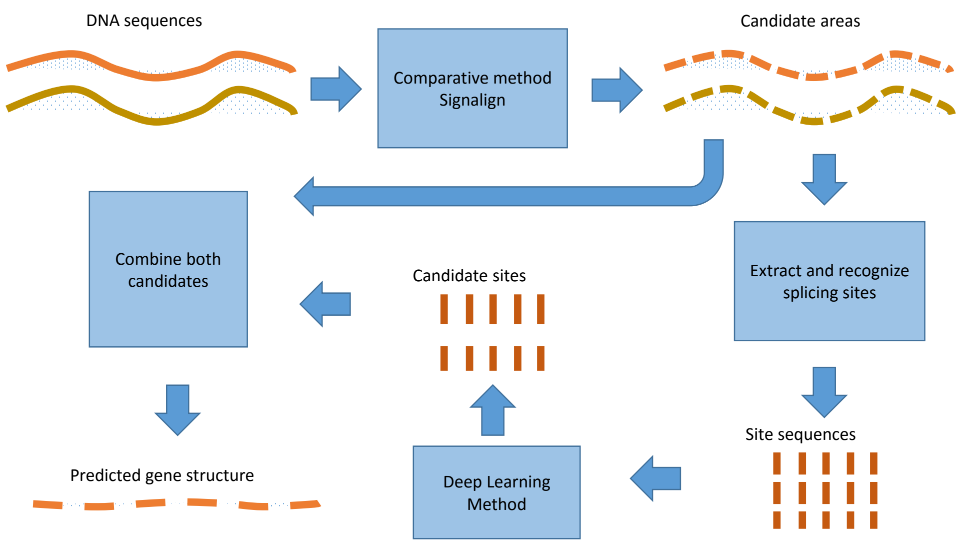


Figure 3.17: An Illustration for the Procedure of Hybrid Method

Hybrid methods integrates the advantages of *ab initio* and comparative methods into this particular application. The innovation of hybrid methods primarily relies on a novel combination of techniques in the two mainstream methods for the performance improvement in a particular application. Hybrid methods simply include two categories: one is the combination in methodology; another is the combination in overlapping result. Even though the latter one looks simpler than the former, the success of a hybrid method heavily depends on the final performance and its particular constraints.

In this application, comparative methods and deep neural network methods are combined for detection of gene structure. The hybrid method is illustrated in Figure3.17. Comparative methods are used for approximately perceiving the candidate location of gene structure whle deep neural network methods are used for discovering whether the candidate is validated exactly as a real exon or intron. In terms of the duplicate criteria, the combination is anticipated to have better performance than a single method.

Results The same data sets and the same measurements adopted in Chapter 2 are used to assess the performance of the hybrid method. Figure3.18 shows the comparisons in sensitivity with other comparative methods. It shows that this method scarifies a little

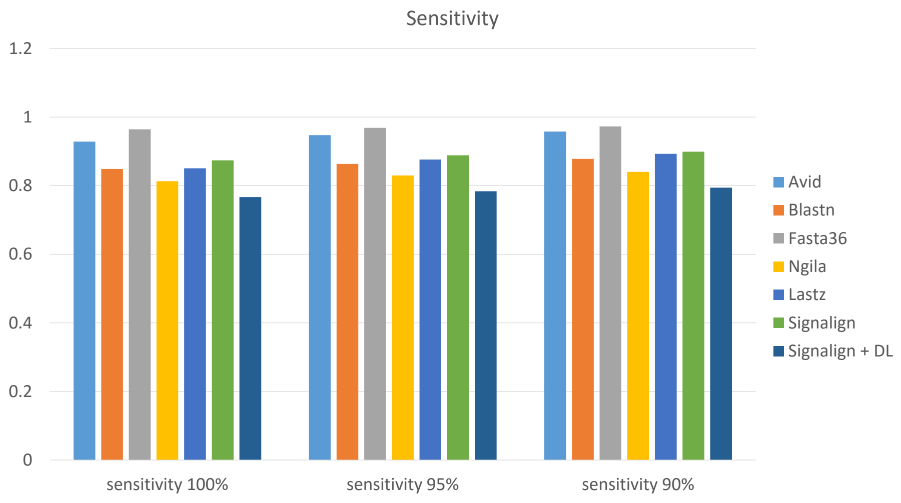


Figure 3.18: Performance of Improved Method in Sensitivity

performance in sensitivity. However, it gains a lot in specificity, accuracy and Matthews correlation coefficient (Mcc), shown respectively in Figure3.19, Figure3.20 and Figure3.21. Especially in the highest level of specificity, accuracy and Matthews correlation coefficient, which are the most important factors for performance evaluation, it shows its remarkable performance over other comparative methods. Noticeably, the hybrid method is two folds better than other methods in specificity.

3.6 Conclusion

In this chapter, deep neural network methods are developed for DNA sequence annotation. Four deep neural network methods and nine encoding schemes are studied to discover the relation of how encoding schemes can affect the performance of deep neural network methods. Three applications based on deep neural network methods are shown. These applications illustrate that deep-learning based methods have a promising future to obtain the better performance on computationally annotating DNA sequences. From the experiments, we can observe that encoding schemes greatly affect the performance of deep learning methods. For DNA genome analysis in deep neural network, direct mapping schemes such as DAX, EIIP and Complementary are better than pre-processed schemes such as Enthalpy,

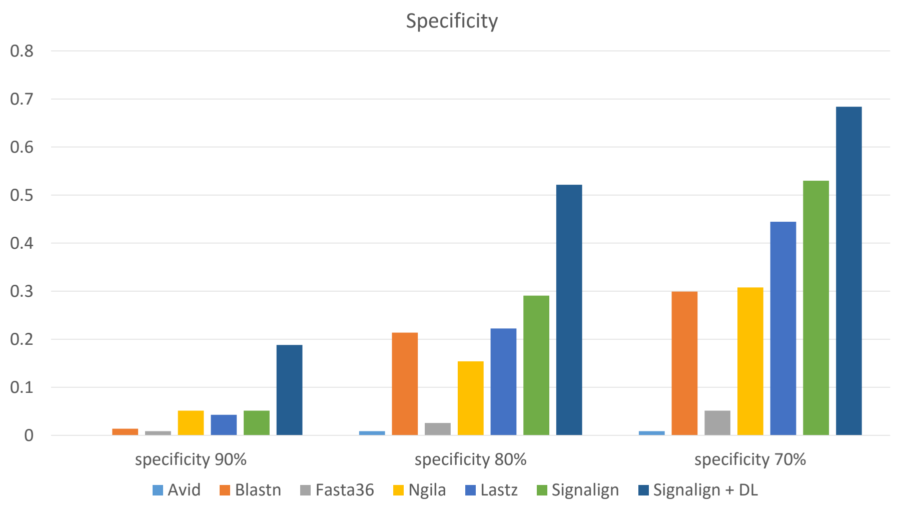


Figure 3.19: Performance of Improved Method in Specificity

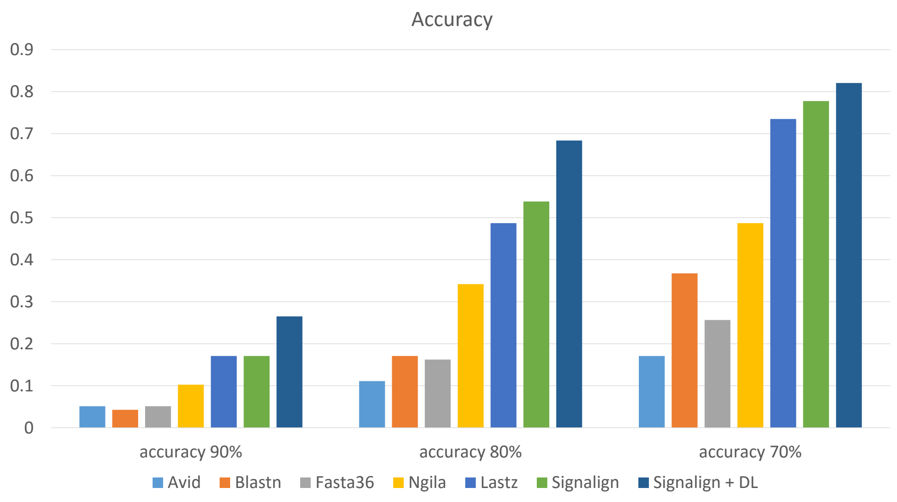


Figure 3.20: Performance of Improved Method in Accuracy

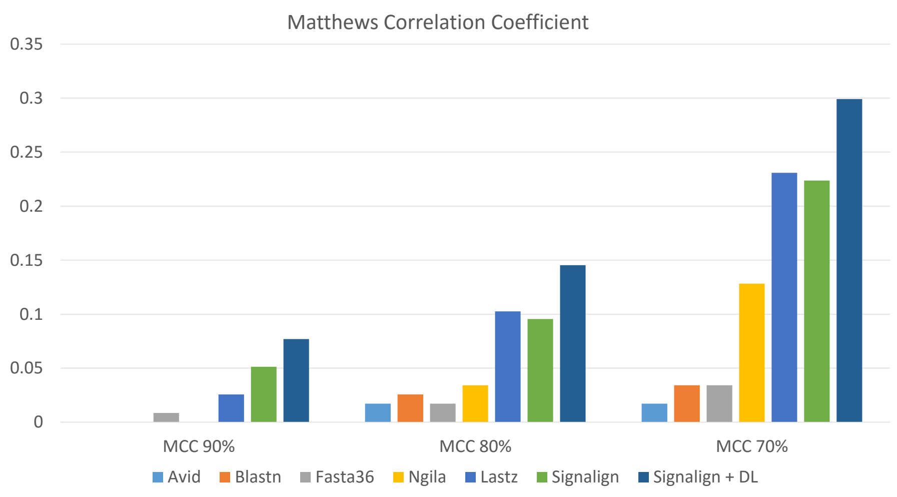


Figure 3.21: Performance of Improved Method in Mcc

Entropy and Galois. It is perhaps because direct mapping does not wrap any information of DNA sequence while pre-processed schemes have hidden some information by encoding them together. Complementary can beat other schemes in more than half of experiments and it is the only one whose sum of the encoding values is zero. DAX manifests a good performance in the areas of similarity analysis and computing, however, it does not mean that it can be superior in other areas such as numeric representation in artificial neural network. Whereas, its performance is closely near Complementary, ranked the 2nd place. From some perspectives, it is still a good encoding scheme.

Detection of LincRNA transcription splicing sites using deep learning method shows a very high accuracy in acceptor data. The auto-encoder method is subsequently used to validate the annotation data and find one unreported splicing site within a lincRNA, which needs to be further validated biologically.

A hybrid method integrating deep learning and proposed comparative method is applied to boost the performance of comparative method, especially the accuracy and specificity in identifying the gene structure. The experimental results show its remarkable increase in various measures and meet our expectation.

CHAPTER 4

INVESTIGATION OF HUMAN CPG ISLAND

In this chapter, CpG islands in human genome sequence are explored by using signal processing, statistic model, cloud-assisted method and other modern computing techniques. Three main questions are included: computational detection of CpG island, redefinition of CpG island and the structural investigation of CpG island. Through the three questions we could gradually approach the hidden truth in CpG island although no one can guarantee whether it is. This chapter is organized as follows. Section 4.1 gives a brief introduction on the problems of CpG island and describes the signal processing method to detect CpG island. Based on the studies of Section 4.1, Section 4.2 tries to redefine and investigate the CpG island to meet the new emerging data criteria using proposed CpG box model and Markov model. In order to speed up the epigenetic analysis, Section 4.3 describes a cloud-assisted platform for CGI investigation.

4.1 GaussianCpG: Detection of CpG Island

4.1.1 Introduction

DNA genomes are punctuated by CpG islands where high profiles of CpG sites are densely contained in certain genome regions. However, CpG contents in the entire DNA genome are generally suppressed to only around 1% comparing with other combinations [86]. Scientists further found that it is in CpG islands where many biological processes occur closely related with high density of CpG contents. In vertebrate, DNA methylation usually occurs in CpG islands and adds an additional methyl to cytosine such that the gene silencing may be caused by the additional methyl. This subtle process can further give rise to gene regulatory problems and epigenetic problems, which can significantly develop to be a hereditary determinant besides genetic factors, such as gene mutation and chromosome rearrangement. However, conventional bisulfite modification-based methods to determine the CpG island are time-consuming [9]. Although new sequencing techniques are developed for whole genome assays, it is reported to be too costly [87]. Thus, computational investigations to CpG islands is efficient and fundamental for many biological studies.

The recently emerging data implies that the definition of CpG island cannot follow these new data. The redefinition and the further investigation of CGI is necessary. These problems are further discussed in Section 4.2.

4.1.2 Related Work

The first article about the computational prediction of CpG islands for vertebrate genome was seen in [88], which proposed CpG island (CGI) problems and gave the definition of CGI, the definition of which has been widely adopted by the later research. A milestone article [89] further constrained the CGIs within only gene promoters and excludes Alu repeat regions. However, recent studies have revealed that CGIs are not only in the area of gene promoters but also contained in the regions of both coding and non-coding [87].

The computational methods for the detection of CpG island can be primarily classified into four categories in terms of their main algorithms. The first type is window-based methods [89] [90] [91], which use a scrolling window to scan through the genome and detect CGIs by these established statistical criteria. A canonical algorithm in [89] shifts a size-adjustable window for 1 *nt* each time to calculate the %G+C content and $\text{CpG}_{obs}/\text{CpG}_{exp}$ within the window until encountering the satisfied CpG island. Subsequently it shifts to next adjacent window and calculates it again until the window does not satisfy the criteria. At that time, it shifts back each *nt* until finding the last satisfied boundary window. This algorithm is widely used because it strictly follows the statistical criteria. Obviously, one of obvious drawbacks of this method primarily is that the window size determines the accuracy of prediction: the larger window increases the predictive granularity and lags the computing speed while the smaller window decreases the computing complexity and increases the probability of omitting a potential CGI. And another drawback is that it probably is too sensitive to predict a whole CGI: a CpG island can be divided into many trivial segments.

The second type is Hidden-Markov-Model-based (HMM) methods [9] [86] [92] [93]. These methods use the statistical transition model to compute transitive probability within CpG island and between CGIs. The transition probability between any two adjacent nucleotides are obtained in the training phase for CGI regions and non-CGI regions respectively. The probability of CG pair in CpG-rich region is much higher than that in non-CGI region. Thus, the log-likelihood ratio of the probabilities of CpG and non-CpG is calculated to reflect the difference between two regions for each possible sequence [93]. However, the variant patterns in CpG islands can easily add some implacable noises to prediction due to insufficient data training, resulting in that the performance of the HMM-based method is negatively affected. Moreover, it is computing-inefficient.

Third, density-based methods [94] intuitively calculate the density of CpG sites, similar to statistical methods in window-based methods. The density of CpG island can be simply computed by taking into account the ratio of the number of CpG sites in the CpG island and the total length of the CpG island. Its basic idea is that it sets initial seeds to iteratively adjust the density variables and expand the CpG-rich regions. That is, initially it sets a low/loose threshold of density to find the approximate border of CpG islands and then use the high/strict thresholds to further detect where the borders are as long as the sequence within the borders meets the density requirement. The main drawback of this method is that the density represents the simply linear relation between the number of CpG sites and the length of CpG island while the ground truth of CpG distribution in CpG islands probably cannot be simply delineated by the linear model.

The fourth is the distance/length-based method [10] [95], which clusters data by the distance between CpG sites and provides a fast way to predict CpG island. Compared with other methods, this method studies the sequence property of primary structure between any two adjacent CpG sites, which provides a new perspective to understand the phenomena of CpG island. However, this method is criticized that it mainly depends on the composition of the sequence, resulting in different outputs for a same CGI in different contexts, and low predictive sensitivity with trivial results [94].

The aforementioned methods cannot pursue both the sensitivity and the specificity simultaneously: either they can have high sensitivity with low specificity, or high specificity can be attained with the loss of the sensitivity. It also implies that the original definition of CGI perhaps deviates from the ground truth.

4.1.3 Methodology

The proposed model aims to fit the niche of previous work by presuming that each CpG site has the information energy that satisfy the Gaussian energy distribution along its primary structure[1]. The Gaussian model is proposed to macroscopically reflect and simplify the principles of microscopical interactions in the complex human genome. The model is computed not by the linear statistical method but by the Gaussian filter. Moreover, the parameters of Gaussian function are not arbitrarily designated but deliberately chosen by optimizing the biological statistics. Thus, it results in that the proposed method shows the better performance than other existing methods in detecting CpG islands.

Assumptions In order to simplify the microscopical interactions in the human genome and macroscopically reflect the general principles of the complex system, we propose the Gaussian model based on the following assumptions: (a) Each CpG site preserves the potential energy and the CpG-rich regions where energy are highly aggregated have more opportunities for methylation. (b) Each CpG island is regarded as an energy field where only the contained CpG sites can affect mutually. (c) The energy of each CpG site is closely related to its primary structure or secondary/tertiary structures. However, due to the uncertainty of unknown secondary or tertiary structures, its primary structure is the first determinant. (d) Since we consider only the primary structure of CpG islands, the energy in a certain location is directly relevant to its neighboring CpG sites [96]. Namely, the energy of each CpG site is distributed across its nearby area along DNA sequence 5' to 3'. (e) The

[1]Here the term of energy can be regarded as the information energy for each CpG [96]. To some extent, it can be replaced by the term of pseudopotential [97].

energy at each nucleotide within the CpG island is the sum of energy distributed by nearby CpG sites. (f) Each CpG site has the same magnitude of information energy.

Notations Assuming that we aim to find all m CpG islands each of which is notated as CGI_i, $i \in \{1, 2, ..., m\}$ in a genome sequence s with the length of n nt. In any CGI_i, its length is l_i, in which k CpG sites lay on. At any CpG site cpg_{ij}, $j \in \{1, 2, ..., k\}$, we assume that it preserves the energy E. The energy is distributed to its nearby nucleotides, which satisfy Gaussian model function $g(x)$ where x is the relative distance to the corresponding CpG site and its directions, $+$ and $-$, represent 5' end and 3' end respectively. The accumulated energy for any nucleotide position x in CGI_i ($x \in \{0, 1, ..., l_i - 1\}$) is denoted as $G_i(x)$, which is the sum of distributed energy $g_{ij}(x)$ at this location.

Gaussian model We assume that each CpG site meets the Gaussian model [96][97] as shown in Equation 4.1.

$$g(x) = \frac{E}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{\sigma^2}}, \tag{4.1}$$

where x is the relative distance from this nucleotide to the CpG site, E is the energy each CpG site preserves and σ determines the smoothness of energy distribution. When $\sigma \to 0$, it converges to an impulse function. From this formula, we can see that when σ becomes large its energy is distributed smoothly. Therefore, σ determines the curve of the distribution and further influences the predictive accuracy of this model.

Further, we can calculate the accumulated energy at any position x' in the CGI_i as Equation 4.2. x' is the absolute location in the CpG island while x is the relative distance to CpG sites. $x' = T(x)$ and $x = T^{-1}(x')$ represent the linear transformation between x and x'.

$$G_i(x') = \sum_{j=1}^{k} g_{ij}(x') = \sum_{j=1}^{k} g_{ij}(T(x)), \tag{4.2}$$

where $j \in 1, 2, ..., k$ and k is the number of CpG sites within this CpG island CGI_i. The mean of pseudopotential energy in CGI_i can be expressed in Equation 4.3.

$$\hat{G}_i = \frac{1}{l_i}\sum_{x=0}^{l_i-1} G_i(T(x)) = \frac{1}{l_i}\sum_{x=0}^{l_i-1}\sum_{j=1}^{k} g_{ij}(T(x)) \tag{4.3}$$

$\hat{G}_i$ is a measure to evaluate the energy in the candidate area: the higher energy it preserves, the more likely the region can be a real CpG island.

Parameters The scarcity of CpG sites in DNA genome determines that CpG sites can bring larger amount of information compared with other regions. From this aspect, the energy proposed in GaussianCpG somehow look similar to information energy. However, in GaussianCpG model, the energy of CpG sites are assumed to distribute to surrounding areas in an energy-rich CpG island. The adjacent CpG sites are presumed to overlap their energy with each other and keep the energy saturated in the region. Obviously, the distances between adjacent CpG sites affect the strength of energy in CpG islands. Additionally, an important assumption is that the influence of CpG sites is only limited to its surrounding area and the far distant CpG sites can barely affect the current location as our model. Thus, before setting the parameters of Gaussian model, we need to cluster the CpG sites so that only nearby CpG sites are considered. That is, identifying the clustering threshold is indispensable prior to setting the GaussianCpG parameters.

In order to investigate the distribution of CpG distances and identify the clustering threshold, we extract all CpGs' distances and observe the distribution of all CpG sites in human genome in Figure4.1, we find that it matches the kernel of exponential distribution. In [10] the curve is locally modeled as an approximate geometric distribution from around 20 nt to 100 nt, which does not reflect the ground truth of its distribution. In Equation 4.4, $f(x)$ is the distribution kernel and x is the distances between CpG sites.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}, \tag{4.4}$$

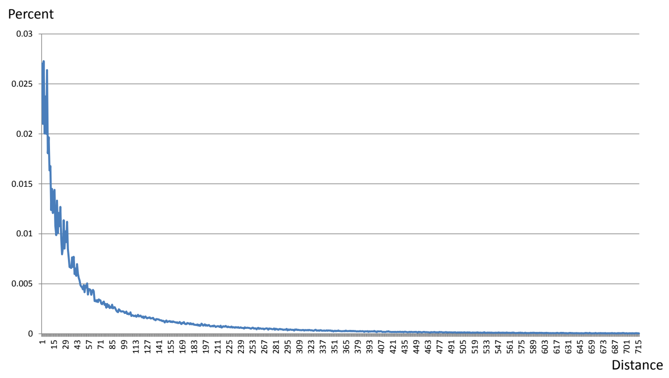


Figure 4.1: Distribution Curve of CpG Distances in Human Genome.

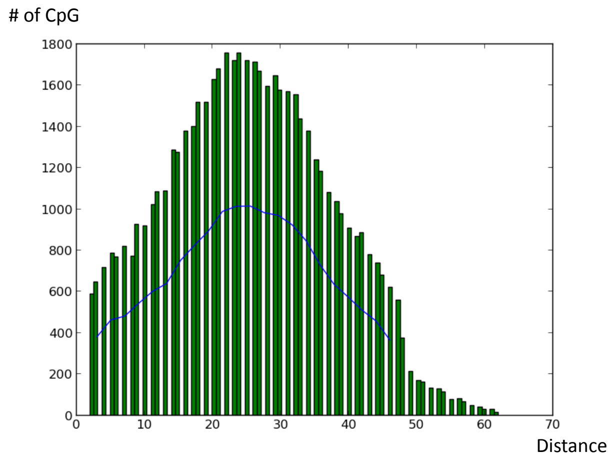


Figure 4.2: Distance Distribution of CGI Candidates in Human Chromosome 21 as an Example and Gaussian Kernel Density Estimation (blue solid line)

where $\lambda = 1/\hat{x}$ and $\hat{x}$ is the mean distance of CpG sites. In terms of the exponential distribution in Equation 4.4, the mean distance is at $\hat{x} = 95$ while at the point of $x = 128$ the third quarter of coverage is $ln4/\lambda$. By removing the under-represented value with large distances, we eventually choose $x = 118$ with 73% coverage as the clustering threshold that eliminates the noises from extra large distances and keeps the most suspicious elements for further processing.

When clustering the CpGs, we can further minimize the range of potential CGI candidates. We extract these CpG distances from potential CGI candidates, draw the distribution chart and find that the kernel density estimation of this distribution fits Gaussian kernel as the blue solid line shown in Figure4.2 where human chromosome 21 is taken as an example.

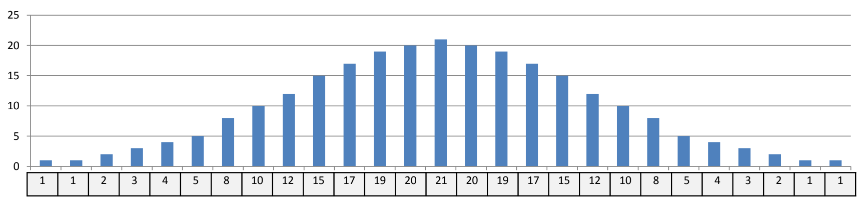


Figure 4.3: Discrete Gaussian Filter

The upper chart shows the value for each location; the lower box is the discrete filter.

At the location of $x = 26$ or $x = 27$, the Gaussian kernel has the peak where the number of distances between two CpG sites approaches the maximum. Thus, 27 is chosen as the digital filter length. In terms of Gaussian model in Equation 4.1, the discrete Gaussian filter is created as shown in Figure4.3.

4.1.4 Algorithm and Implementation

The main procedures of GaussianCpG are shown as Figure4.4: (1) Find all CpGs for each human chromosome; (2) Cluster these CGIs in terms of distance threshold; (3) Apply Gaussian filter to each cluster and calculate the magnitude of Gaussian pseudopotential; (4) Utilize a binary threshold to filter clusters; (5) Collect the filtered clusters; (6) Calculate %G+C for the remaining clusters and pick up those that meet the %G+C content. In the first step, all CpG sites are extracted from genome as well as their properties, such as locations and distances between two adjacent CpG sites[2]. In the second step, using the statistical threshold $x = 118$ we have acquired in statistics, we cluster these CpGs into groups that may contain lots of CpG islands. The basic idea of clustering algorithm is to find all locations where distances are greater than threshold and then cut the sequence from these locations into segments. Subsequently, we apply Gaussian filter to scroll these clusters and calculate their energy value for each location. Segments can have the accumulated energy as well. After that, a binary filter is utilized to the computed loci in order to detect if these loci should be kept as CGI candidates, resulting in that new clusters are generated.

[2] The repeat regions are not included in this project following most previous methods even if some literature [98] did state that repeat area may involve more evolutionary force.

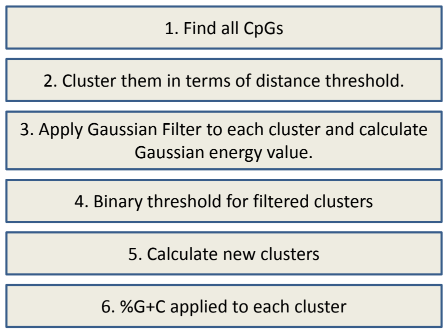

1. Find all CpGs
2. Cluster them in terms of distance threshold.
3. Apply Gaussian Filter to each cluster and calculate Gaussian energy value.
4. Binary threshold for filtered clusters
5. Calculate new clusters
6. %G+C applied to each cluster

Figure 4.4: The Main Procedures of GaussianCpG.

That is, inside the large segment, it might be divided into sub-segments depending on the accumulated energy. The threshold we adopt here is 1.5 times of the average energy across the digital filter because of 2δ containing 95% energy in terms of Gaussian distribution function. Finally, we count the percentage of %G+C content in these sub segments with the threshold of 40% and determine whether they are candidates.

For the computing complexity, the primary computing task is in applying Gaussian filter to clustered CpG sites. To speed up the calculation, we generate a matrix table that stores the computing intermediates to save the computational time. That is, for each location involved Gaussian filter computation, it takes constant times for the calculation. Thus, its time complexity in Gaussian computation is $O(n)$. For the rest computing tasks, extracting CpG sites takes $O(n)$ and sorting the distance takes $O(n \log n)$. Therefore, the time complexity of GaussianCpG is $O(n \log n)$. The program is implemented in Python and its libraries.

4.1.5 Validation and Assessment

Data set and Evaluation metrics In [10], in order to examine the capability of predicting those known CGIs for methods, an artificial dataset was generated from the known dataset, in which real CGIs were embedded into fake genome sequences. By detecting the real CGIs in those fake sequences, the specificity and the sensitivity of the software can be validated. In the same vein, we generate an artificial dataset to test the specificity of

Table 4.1 Hit Rate of Known Human CGIs

Chr#	Known	Predicted	Hit
Chr 1	546	541	99.08%
Chr 2	430	426	99.07%
Chr 3	319	319	100%
Chr 4	272	272	100%
Chr 5	359	356	99.16%
Chr 6	293	292	99.66%
Chr 7	304	298	98.03%
Chr 8	254	253	99.61%
Chr 9	359	356	99.16%
Chr 10	311	311	100%
Chr 11	346	346	100%
Chr 12	363	360	99.17%
Chr 13	200	200	100%
Chr 14	206	205	99.51%
Chr 15	150	150	100%
Chr 17	383	380	99.22%
Chr 18	43	43	100%
Chr 19	315	314	99.68%
Chr 20	259	257	99.23%
Chr 21	133	131	98.50%
Chr 22	215	214	99.53%
Chr X	253	250	98.81%
Chr Y	5	5	100%

Known CGIs: 6786, predicted: 6740, avg. hit rate: 99.32%.

Table 4.2 Comparison in Artificial Data Set

[a]Method:	I	II	III	IV	V
T	6854696	6854696	6854696	6854696	6854696
TP	2101562	3603662	5489738	2531549	5036243
FN	4753134	3251034	1364958	4323147	1818453
F	5919255	5919255	5919255	5919255	5919255
FP	20437	220957	1085303	9319	46906
TN	5898818	5698298	4833952	5909936	5872349

[b]Method:	I	II	III	IV	V
Sn	30.66%	52.57%	**80.09%**	36.93%	73.47%
Sp	99.65%	96.27%	81.66%	**99.84%**	99.21%
Acc	62.63%	72.82%	80.82%	66.08%	**85.40%**
Mcc	99.04%	94.22%	83.49%	**99.63%**	99.08%
Ppv	30.57%	50.93%	69.14%	36.88%	**72.97%**
Pc	40.61%	53.18%	61.61%	45.94%	**74.04%**
F1	46.82%	67.49%	81.75%	53.89%	**84.37%**

I:CpGPlot, II:CpGReport, III:CpGProd, IV:CpGCluster, V:GaussianCpG

For Panel [a]: The unit of measurement is necleotide.
True, T: the length of known CpG islands.
False, F: the length of non-CpG islands.
True positive, TP: the length of predicted known CGIs.
False positive, FP: the length of predicted CGIs not in known CGIs.
False negative, FN: the length of not predicted known CGIs.
True negative, TN: the length of predicted non-CGIs.

For Panel [b]:
Sensitivity, $Sn = TP/(TP + FN)$
Specificity, $Sp = TN/(TN + FP)$
Accuracy, $Acc = (TP + TN)/(TP + FP + FN + TN)$
Matthews correlation coefficient,

$$Mcc = \frac{TP \times TN - FN \times FP}{\sqrt{(TP+FN) \times (TN+FP) \times (TP+FP) \times (TN+FN)}}$$

Positive predictive value, $Ppv = TP/(TP + FP)$
Performance coefficient, $Pc = TP/(TP + FN + FP)$
F1 score, the harmonic mean of precision and sensitivity,

$$F1 = 2 \times TP/(2 \times TP + FP + FN)$$

For Panel [a]&[b]: Default parameters for all software are set.

GaussianCpG. However, different from [10], we create the artificial dataset by using real human DNA sequences that were located at the regions between two CpG-rich areas to pad the gaps between known CpG islands instead of by randomly generating nucleotides in [10]. The artificial data set contains 6,786 known CpG islands from the annotation database [99] with the nucleotide length of 6,854,696 *nt* and 6,786 non-CpG islands with the nucleotide length of 5,919,255 *nt*. And Lengths of CGIs vary from a hundred nucleotides to a few thousand of nucleotides.

In addition to artificial data set, in order to further validate our method, we take the benchmark of real data from UCSC annotation of Human Chromosome 21, which contains 348k annotated CGIs along with 46M DNA genome sequence.

Four mainstream software are examined in the performance evaluation of CpG-island prediction, including CpGPlot [91], CpGReport [91], CpGProd [90] and CpGCluster [10]. In the nucleotide level, the performance of each method is assessed by the observation of True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN), as shown in Panel *a* of Tables 4.2 and 4.3. Furthermore, the comprehensive assessments are defined and calculated, including sensitivity (Sn), specificity (Sp), accuracy (Acc), Matthews correlation coefficient (Mcc), positive predictive value (Ppv), performance coefficient (Pc) and F1 score, as shown in Panel *b* of Tables 4.2 and 4.3.

4.2 CpG Box and Markov Chain Model: A New Method to Measure and Define CpG Islands

4.2.1 Introduction

In order to computationally detect CpG islands, the definition of CpG island is needed to constrain the computing procedure. Generally, CpG islands are defined as CpG-rich regions where the epigenetic processes are highly related with the methylation status. Three criteria are widely accepted as the definition of CpG islands: (a) %G+C content is $\geq 50\%$, (b) the ratio of the observed CpG content and the expected CpG content is ≥ 0.6 [89], and

Table 4.3 Comparison in Real Data Set

[a]Method:	I	II	III	IV	V
T	348930	348930	348930	348930	348930
TP	255732	348546	333015	300315	292732
FN	93198	384	15915	48615	56198
F	46361053	46361053	46361053	46361053	46361053
FP	397423	1680731	1034353	583460	363493
TN	46124740	44417698	45331765	45923959	46075369

[b]Method:	I	II	III	IV	V
Sn	73.29%	**99.88%**	95.43%	86.06%	83.89%
Sp	99.14%	96.35%	97.76%	98.74%	**99.21%**
Acc	98.95%	96.38%	97.75%	98.65%	**99.10%**
Mcc	53.11%	40.65%	47.61%	53.60%	**60.80%**
Ppv	39.15%	17.17%	24.35%	33.98%	**44.60%**
Pc	34.26%	17.17%	24.07%	32.20%	**41.08%**
F1	51.03%	29.31%	38.80%	48.72%	**58.24%**

I:CpGPlot, II:CpGReport, III:CpGProd, IV:CpGCluster, V:GaussianCpG

For Panel [a]&[b]: The setting and metrics are same as those in Table 4.2.

(c) the general length of CGI is greater than 200 nucleotides.

Most existing computational methods for the prediction of CpG island are programmed on these rules. However, many experiments have verified that CpG islands deviate from these threshold-based criteria [9][100]. Experimental data indicate that many cases violate the criteria including %G+C $< 50\%$, $\text{CpG}_{obs}/\text{CpG}_{exp}$ varying, and the length of CGI ranging from eight nucleotides to a few thousand of nucleotides [10]. Moreover, the recent Methyl-seq data show that more than 65% methylation areas are not located at the CGIs of present definition [101]. For example, Methyl-seq assays more than 250,000 methyl-sensitive restriction enzyme cleavage sites that consist of more than 90,000 genomic regions. Among these methylation regions, only 35,528 regions are located at annotated CpG islands, while the remaining 55,084 regions are not located in any conventional CpG islands, including areas like promoters, genes, and intergenic regions [101]. It strongly indicates that the present threshold-based CGI definition needs to be modified and the CGI detection is not just a straightly statistical/computing task. Some unrevealed rules may be hidden in the CpG-enriched regions. The redefinition is expected to comprehensively consider these existing issues.

The Relevance between CGI and RNA structure Recent research has discovered that CpG island structure may play a fundamental role in establishing chromatin structures in the pluripotent genome because the locations of genome-wide CGIs are found relevant to the chromatin structure of nucleosomes H3K4me3 and H3K27me3 [102]. It indicates that there probably are some certain patterns within CpG island. However, these patterns are hidden in numerous genome sequences and not so obvious for direct observations. That is, the re-examination of CpG island and the investigation into CpG island are necessary since the present definition of CpG island has the coarse granularity with the arbitrary-like threshold. Thus, we propose the Markov model and the CpG box for the re-examination and the investigation by taking the advantage of biological big data in genome sequences.

Discovered Structures in CGI Epigenetic silencing involves the aberrant methylation of CpG islands and suppresses the methylation in cancer. CpG islands are susceptible to aberrant methylation and these aberrant methylation can be predicted. It means that certain structural/sequence features may contribute to the protection from or susceptible to aberrant methylation. Some sequence motifs are elicited by classical techniques and classified into two categories: methylation-prone and methylation-resistant. In [103], some motifs are identified and they are found to associate with Alu and other repetitive sequences.

Long-short Repetition Figure4.5 and Figure4.6 indicate the distribution of lengths of all CpG boxes in human genome and manifest the same distribution pattern in all chromosomes except chromosome Y. We can observe the fluctuation of distribution in odd-even lengths and the long-short-bar fluctuations are interrupted at the same positions shown as the red arrows.

We cannot find any previous description from extant documents about the phenomena of coincident fluctuations in CpG box's length distributions for almost all human chromosomes. We speculate that its regularity of fluctuation may be caused by some patterns existing within CpG boxes. However, no prior literature can support our speculation about the fixed pattern of CpG island. It may rely on two reasons: (1) there is no appropriate model to describe the structural patterns of CpG island structure, (2) indeed there is no common patterns for CpG island. Figure4.5 and Figure4.6 imply that some hidden patterns might exist and a new model is needed to detect them. The redefinition of CGI should have some connections with these new findings.

Energy Analysis of CpG Box As the analysis in chapter 2, thermodynamics are conserved in CGI sequences. If encoding the dinucleotides in terms of bio-chemical properties of DNA sequences, the combination of CpG will have the highest quantitative value. Within a CpG box, the starting value and the end value are the peaks which preserve the highest energy and the dinucleotides within the CpG box are the valley between two peaks where less energy is preserved. Therefore, it manifests the special pattern of CGI structure. The

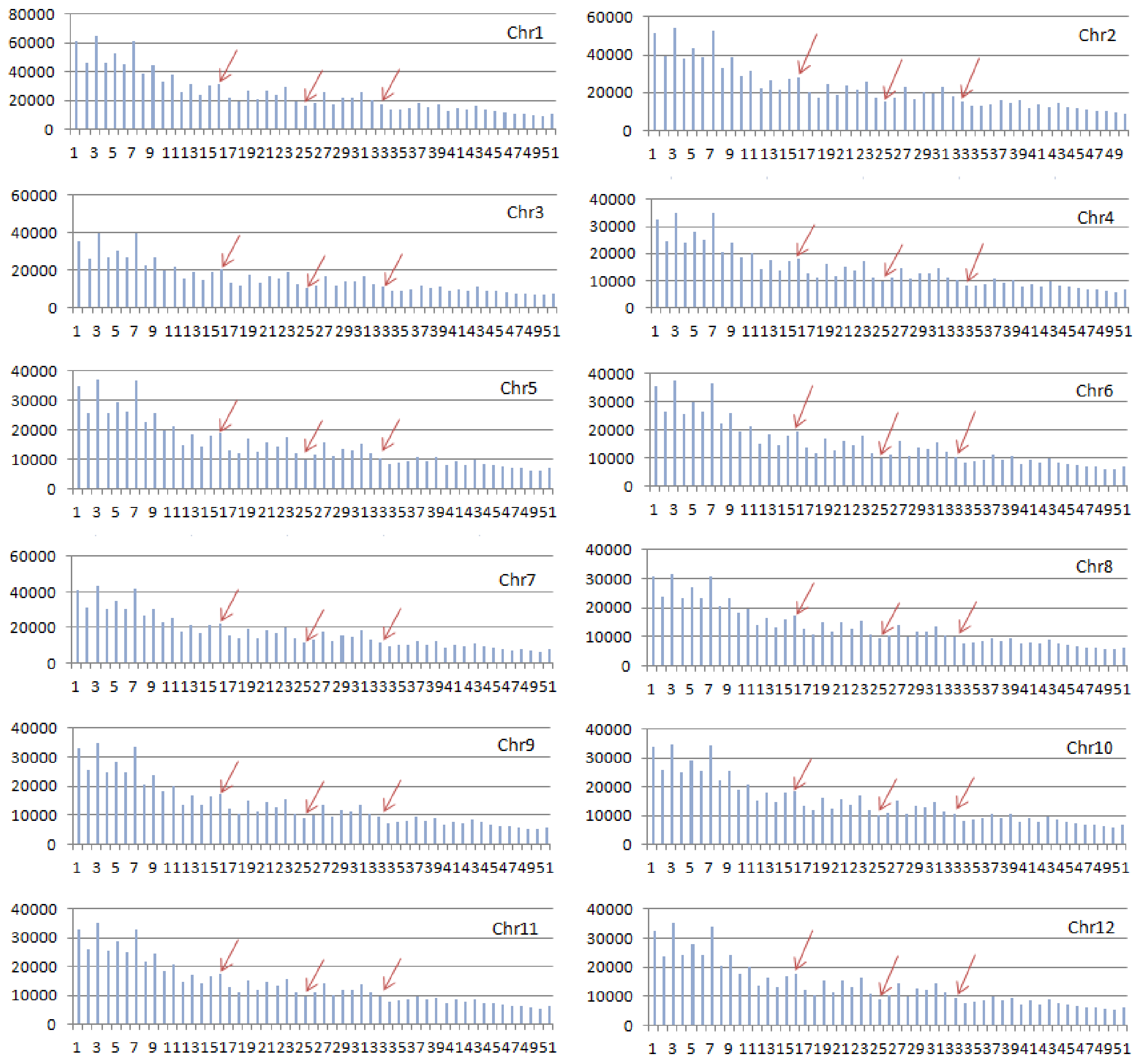


Figure 4.5: Fluctuations of Length Distributions of CpG Box in Human Chromosomes 1-12

Red arrows show the first three locations of long-short-bar interruptions from left to right along x-axis and the remaining interruptions are not indicated. Similar patterns are shown along chromosomes 1-12.

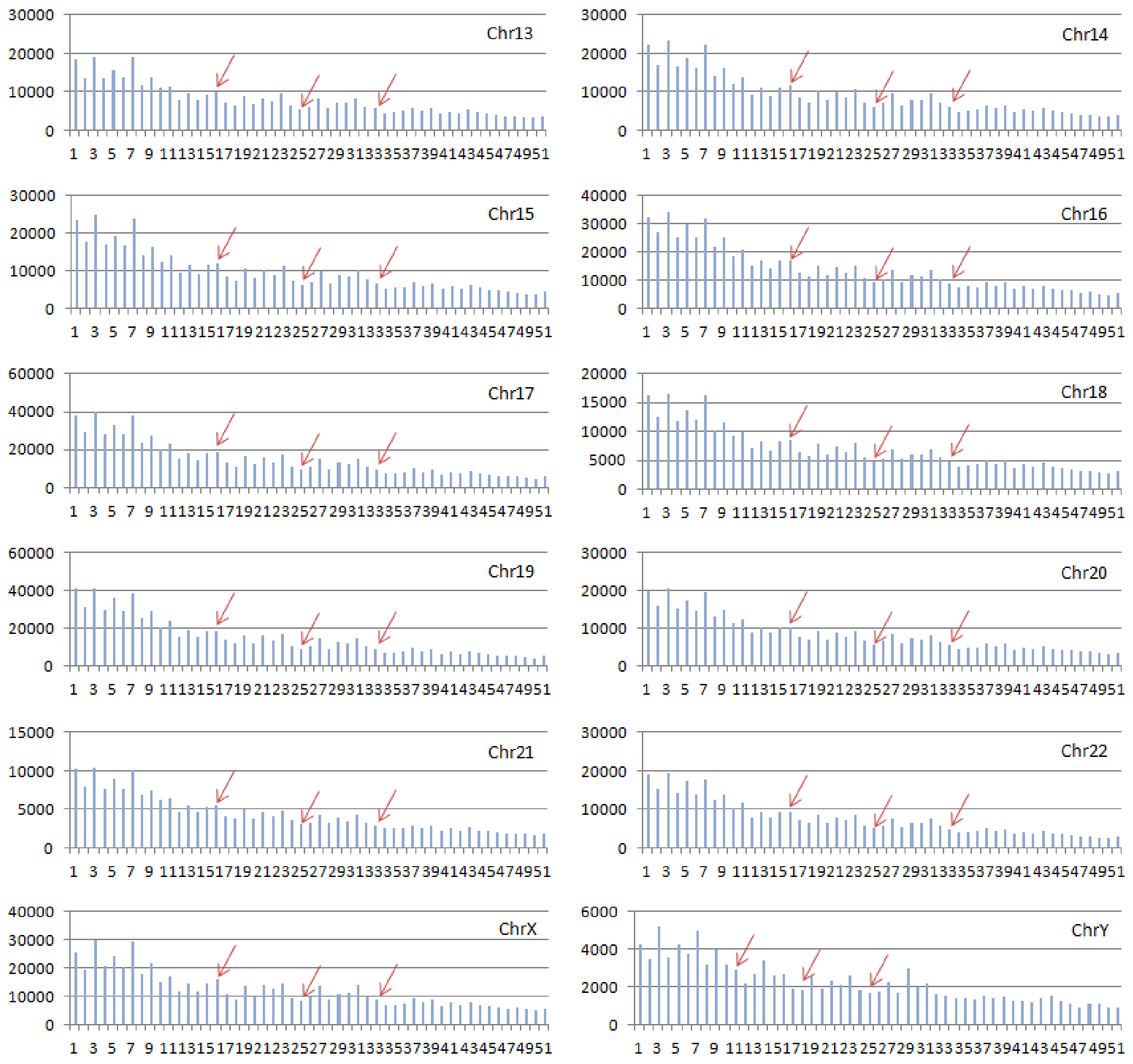


Figure 4.6: Fluctuations of Length Distributions of CpG Box in Human Chromosomes 13-22, X and Y

Red arrows show the first three locations of long-short-bar interruptions from left to right along x-axis and the remaining interruptions are not indicated. All chromosomes, except Chromosome Y, show the similar patterns of regularity and the same locations of interruptions.

thermodynamic analysis provides the representation of visualization for CpG box.

The consensus CGI motif is also calculated throughout all CpG boxes since it is speculated to preserve certain dynamic structures and link to bio-chemical properties.

4.2.2 Previous Work

A CpG box is defined as a tiny sequence where the starting dinucleotide and the end dinucleotide are CpGs. Hence, the CpG island is composed of CpG boxes. The terrain of CpG island structure can be measured by investigating CpG boxes. It is inspired by the distance analysis between CpGs [10] [11] [104]. Actually, the distance analysis is about the property of DNA primary structure and the definition of CpG box can definitely include the properties of primary structure. Thus, Markov chain model can be easily applied to the investigation of CGI and the re-examination of CGI definition.

Due to the high frequency of CpG occurrence in CpG island, present definitions focus on the parameter thresholds from the G+C content, the CpG ratio between observation and expectation, and the length of CpG island. Thus, using the mathematical thresholds to distinguish the CpG-rich regions from other regions is the simplest and the most direct way to find these interest regions. However, it is arbitrary and rough even though the threshold-based definition can quickly identify the potential CGI regions because threshold-based definition may be coarse-grained.

Differentially methylated regions (DMR) [105] frequently overlap with CpG island and some special DNA repeats are more frequently contained in CpG island than randomly selected regions. Besides of Methylation, some studies [98] unveil that the Alu repeats and CpG islands are closely related and widely subject to methylation. Thus, it was speculated whether Alu elements form new CpG islands and give rise to gene expression changes as an evolutionary force. On this aspect, we need more evidences to find the evolutionary relevance between repeats and CGI. Interestingly, scientists find some motifs in CGI [103] that are targeted or protected by methylation processes. The studies combine the analysis of CGIs, methylation and sequence variants in human DNA genome and also find the association

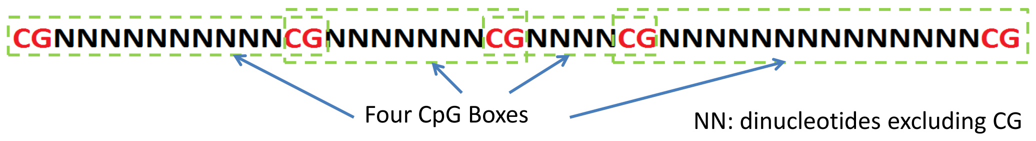

CGNNNNNNNNNNCGNNNNNNNCGNNNNCGNNNNNNNNNNNNNNNCG
Four CpG Boxes
NN: dinucleotides excluding CG

Figure 4.7: A Toy Example to Illustrate the Definition of CpG Box

between CGI motifs and Alu repetitive sequences. In addition to the motif finding in CGI, some special structures in CpG island are identified [102] with respect to two structure-related features of CpG islands: (1) the secondary structure of ncRNA has characteristic CG-rich stem loop structures and (2) CpG islands relevant to transcription starting sites of genes pervasively coincide with small RNAs that show CG-rich hairpin structures. These studies have provided strong supports on the potential structure of CpG islands. However, the existing definition of CpG island can not lend any help to the structural studies on CGI due to its mathematical property. Thus, a new definition is needed for the future studies. Under these circumstances, CpG box model is proposed to resolve the problem.

4.2.3 Model and Method

CpG Box CpG box refers to the regions between two neighboring CpGs where nucleotides within the CpG dinucleotides are encapsulated likely in a black box. An example is shown in Figure4.7 for the illustration of the definition of CpG box. CpG box is different from CpG distance that was proposed in [10]. The latter is about the length of CpG box, a property of CpG box while CpG box is a particular object for studying the properties of CGI. The studies of CpG box are derived from the statistics of CpG distance and the energy analysis in human DNA genome. Figure4.8 shows that CpGs outline the terrain of DNA sequence after encoded into signals through dinucleotide enthalpy analysis, dinucleotide statistics and information entropy encoding.

Assumption and Notation Each CpG box can be analyzed as a stochastic sequence with Markov property. The next state of nucleotide depends only on the state of the current

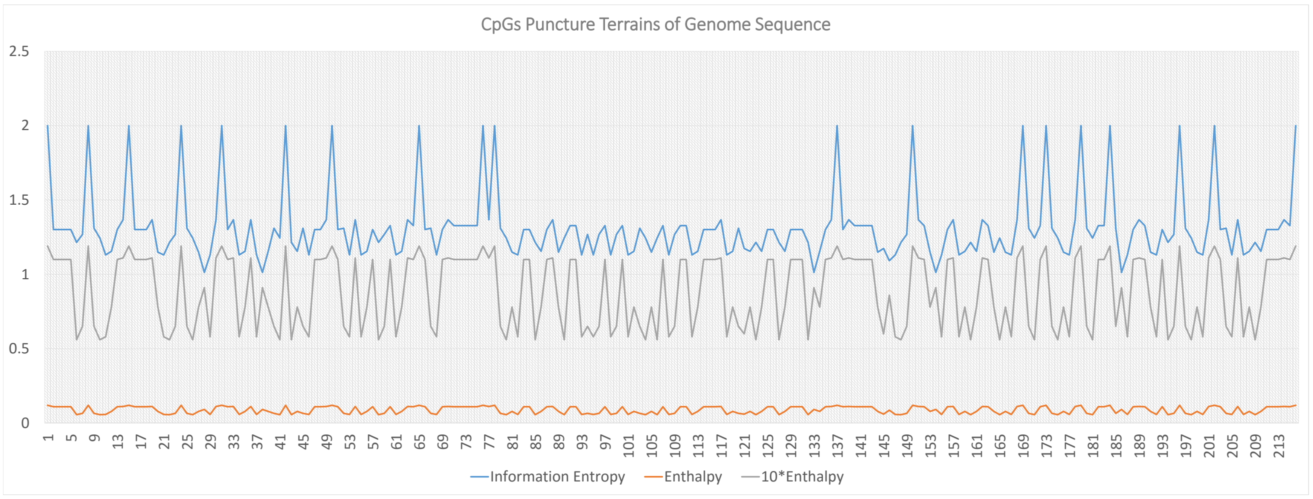


Figure 4.8: CpGs Outline the Terrain of Genome Sequence/CpG Island

Each peak is the location of a CpG.

nucleotide. Generally, it can be formulated as Equation 4.5.

$$\begin{aligned} &P(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) \\ &= P(X_n = x_n | X_{n-1} = x_{n-1}), \end{aligned} \tag{4.5}$$

where X_n is the stochastic process of position n and x_n is a nucleotide at the corresponding position n.

In a CpG box, the stochastic process always starts from a CpG dinucleotide and stop at another CpG dinucleotide. No any other CpG occurs in the box. For a CpG box with the number of nucleotide n, x_1, x_2, x_{n-1} and x_n are fixed where nucleotide is C or G. We have the probability P if the number of nucleotides in a CpG box is n, $P(X_1, X_2, ..., X_n) = P(X_2, ..., X_{n-1})$ since x_3 and x_{n-2} are relevant to x_2 and x_{n-1}. Thus, for computing convenience, we redefine X_1 as the nucleotide position at the nucleotide G of the starting CpG site and X_n as the nucleotide position at the nucleotide C of the end CpG site.

Each dinucleotide is studied in a chain since the neighboring nucleotides constitute the structure of CpG box from the viewpoint of energy analysis [88]. The next dinucleotide is determined by the current dinucleotide because they share a same nucleotide and this nucleotide combines next nucleotide to form the next dinucleotide.

Markov Chain within CpG Box The Markov chain model can be represented as Equation 4.6.

$$\begin{aligned} &P(X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) \\ &= P(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) \\ &\quad \times P(X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) \end{aligned} \tag{4.6}$$

Apply Equation 4.5 and further expand Equation 4.6, we have Equation 4.7.

$$\begin{aligned} &P(X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) \\ &= P(X_n = x_n | X_{n-1} = x_{n-1}) \\ &\quad \times P(X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) \\ &= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1}) \end{aligned} \tag{4.7}$$

Following the Bayesian equation, Equation 4.7 can further be induced to Equation 4.8. Note that $P(X_1 = x_1) = 1$ since the nucleotide is fixed.

$$\begin{aligned} &P(X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) \\ &= \prod_{i=2}^{n} \tfrac{P(X_i = x_i, X_{i-1} = x_{i-1})}{P(X_{i-1} = x_{i-1})} \end{aligned} \tag{4.8}$$

Combing with dinucleotide D_{i-1} that appears in X_i and X_{i-1}, $i \in \{2, ..., n\}$, we have Equation 4.9.

$$\begin{aligned} &P(X_n = x_n, X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) \\ &= \prod_{i=2}^{n} \tfrac{P(D_{i-1})}{P(X_{i-1} = x_{i-1})} \end{aligned} \tag{4.9}$$

The definition of CpG box fits the model well since its properties not only constrain the stochastic walk but also contain the bio-chemical relevance.

CpG box varies in length that is surmised to be affected by bio-chemical structures. And the length of neighboring CpG boxes is the result of reciprocity each other. Thus, the probability of length transition between neighboring CpG boxes is also studied in the later subsections.

Maximum Likelihood In terms of Equation 4.9, from the current database of annotated CGIs, we can sort all CpG boxes in CGIs and acquire the probabilities of each dinucleotide located at particular locations of CpG boxes. In the same way, we also can store all CpG boxes in all non-CGIs and obtain the probabilities of dinucleotide in each type of CpG boxes.

A CpG box has an estimation parameter θ, $\theta \in \Theta$, Θ is the set of $\{\theta_{cgi}, \theta_{noncgi}\}$, simplified as $\{\theta_C, \theta_N\}$. The likelihood of an estimator with a parameter θ and observations $X_n, X_{n-1}, ..., X_1$ is denoted as $L(\theta; X_n, X_{n-1}, ..., X_1)$. In terms of Bayesian theorem, we have Equation 4.10. Furthermore, the maximum likelihood estimator is equivalent to the most probable Bayesian estimator if we assume that parameter θ meets the uniform prior distribution. That is, a prior $P(\theta)$ in Equation 4.10 is uniform distribution and $P(X_n, X_{n-1}, ..., X_1)$ is independent of θ because of the average probability over all parameters θ. Thus, the Equation 4.10 can be further induced to Equation 4.11

$$P(\theta|X_n, X_{n-1}, ..., X_1) = \frac{P(X_n, X_{n-1}, ..., X_1|\theta)P(\theta)}{P(X_n, X_{n-1}, ..., X_1)}. \tag{4.10}$$

$$\begin{aligned} L(\theta; X_n, X_{n-1}, ..., X_1) &\equiv P(\theta|X_n, X_{n-1}, ..., X_1) \\ &\propto P(X_n, X_{n-1}, ..., X_1|\theta)P(\theta) \\ &\propto P(X_n, X_{n-1}, ..., X_1|\theta) \end{aligned} \tag{4.11}$$

Neighboring CpG Boxes It is obvious that not only the composition of CpG boxes determines the primary structure but also the neighboring CpG boxes consist of the structure of CpG islands. Examining the neighboring CpG boxes are indispensable to investigate the forming of CGIs. CpG box B_1 determines CpG box B_2 that subsequently determines CpG box B_3. That is, the current CpG box is dependent on the previous one and determines the next one.

The likelihood of estimator θ_C for $B_i \rightarrow B_{i+1}$ can be expressed as Equation 4.12. In a similar vein of Equation 4.11, the maximum likelihood estimator is equivalent to the most probable Bayesian estimator if we assume that parameter θ meets the uniform prior distribution. $P(B_i \rightarrow B_{i+1})$ is independent of θ because of the average probability over all θ parameters. The probability of estimator θ is assumed as uniform distribution since no estimator probability can be detected for CGI or non-CGI.

$$\begin{aligned} L(\theta; B_i \rightarrow B_{i+1}) &\equiv P(\theta | B_i \rightarrow B_{i+1}) \\ &= \tfrac{P(B_i \rightarrow B_{i+1}|\theta)P(\theta)}{P(B_i \rightarrow B_{i+1})} \\ &\propto P(B_i \rightarrow B_{i+1}|\theta)P(\theta) \\ &\propto P(B_i \rightarrow B_{i+1}|\theta). \end{aligned} \tag{4.12}$$

Data Collection We collect and extract CpG boxes from annotated CGI database(22M) in Human genome, denoted as CGI group, and the other CpG boxes in the remaining Human genome, denoted as pseudo non-CGI (or simply non-CGI) group. After sorting all CpG boxes, the statistic data about CpG boxes of various lengths can be acquired respectively from the two groups of data sets, especially the dinucleotides' statistical information following Equation 4.9. From Equations 4.11, we can calculate $P(\theta|X_n, X_{n-1}, ..., X_1)$ in different estimators and further determine the maximum likelihood of estimators.

In terms of Equation 4.12, the maximum likelihood of neighboring CpG boxes needs the data set for every two neighboring CpG boxes corresponding to each estimator. Thus, we collect two groups of data for neighboring CpG boxes. The CGI data are from the annotated CGIs and the non-CGI data are from the remaining CGIs after removing the annotated CGIs.

4.2.4 Results and Assessment

According to the literature [9], model-based CGI definition is the best method in the world that has been adopted by UCSC genome browser [41] as the standard of annotated CGIs. Therefore, we use the model-based UCSC data set as our target for comparison and assessment.

Structure of CpG Box Evolution-related activities frequently occur in CpG islands. It was asserted that there are no particular structures in CpG islands. Moreover, due to the limited analytic models and methods, the structures of CpG islands are not regarded to be resolved. The use of CpG boxes and Markov chain model brings a new perspective for investigating this problem.

From known annotated CpG islands, all CpG boxes are extracted and analyzed. Figures 4.9 4.10 and 4.11 show the consensus of CpG box with various lengths as the x-axis. The y-axis is entropy-based probability for the frequency of each location. All these logo figures are generated from Weblogo [106]. Due to the effect of a large number of CpG boxes, the frequency of nucleotides in some locations seems even. However, these consensuses manifest that some possible structures may exist in CpG boxes.

From these consensus data, we can observe that there are obvious secondary structures within CpG boxes. It matches the finding in [102] that CGI regions have latent RNA secondary structures, especially hairpin structures and CG-rich stem loop structures. These consensus data also verify that CpG box is so far the best way to measure the structures in CGI.

Transcriptional Starting Sites Transcriptional Starting Sites (TSS) are important markers to measure the relevance between CGI and methylation because the gene silencing may be caused by the methylated CpG sites and the transcription in the transcriptional

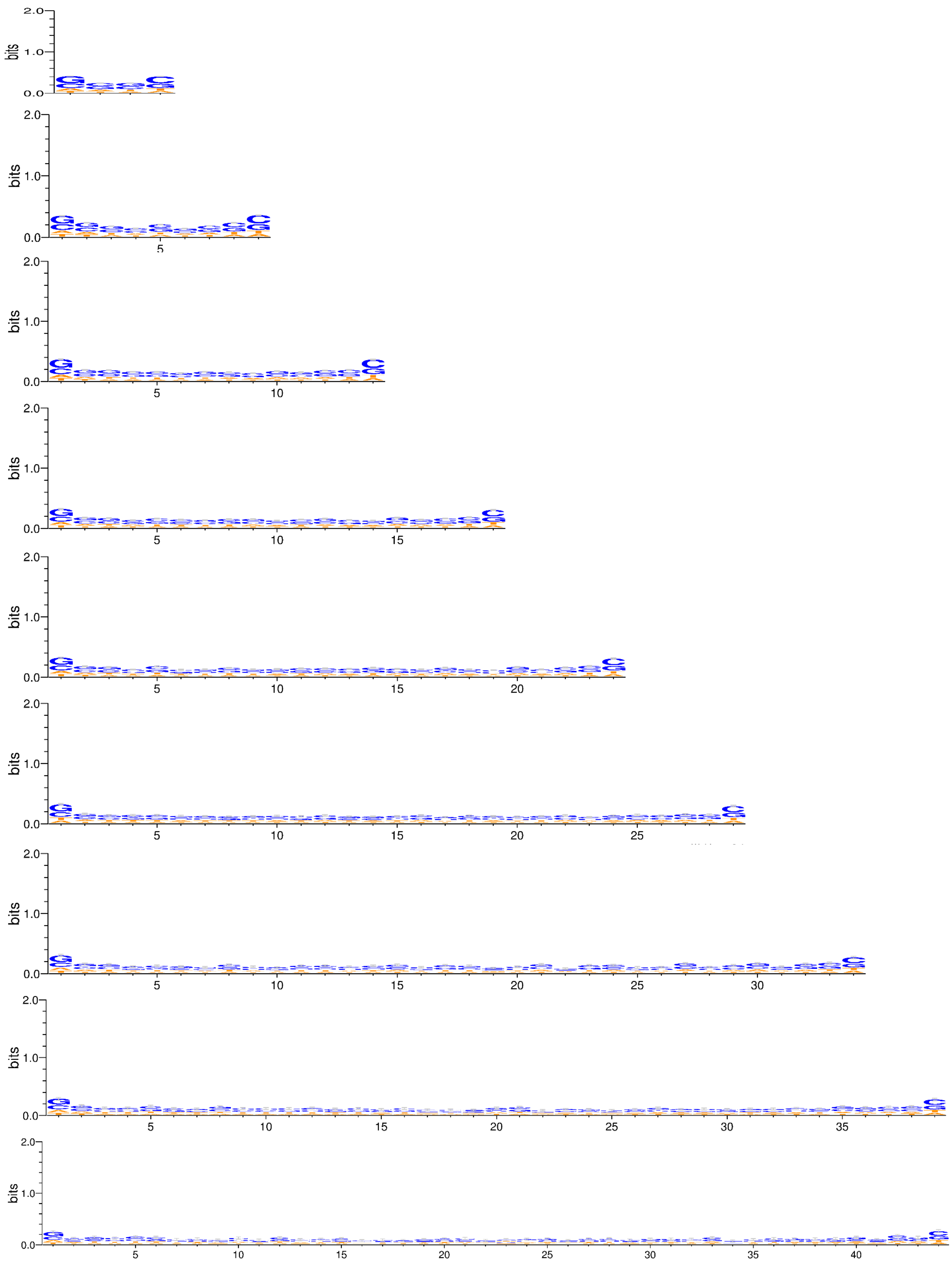


Figure 4.9: Consensus Logos of CpG Box with Length = 5, 10, 15, 20, 25, 30, 35, 40 and 45.

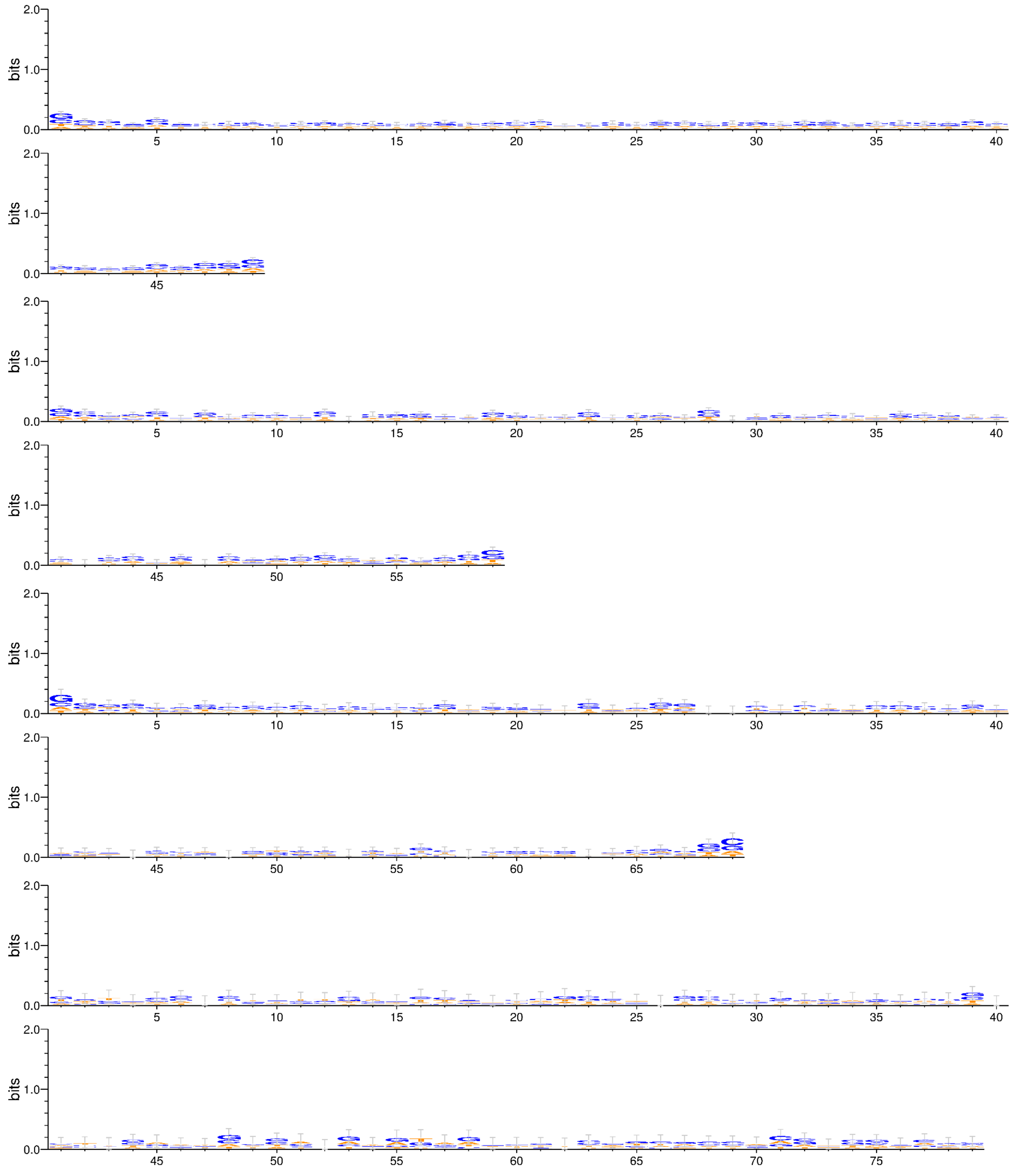


Figure 4.10: Consensus Logos of CpG Box with Length = 50, 60, 70, 80 and 90.

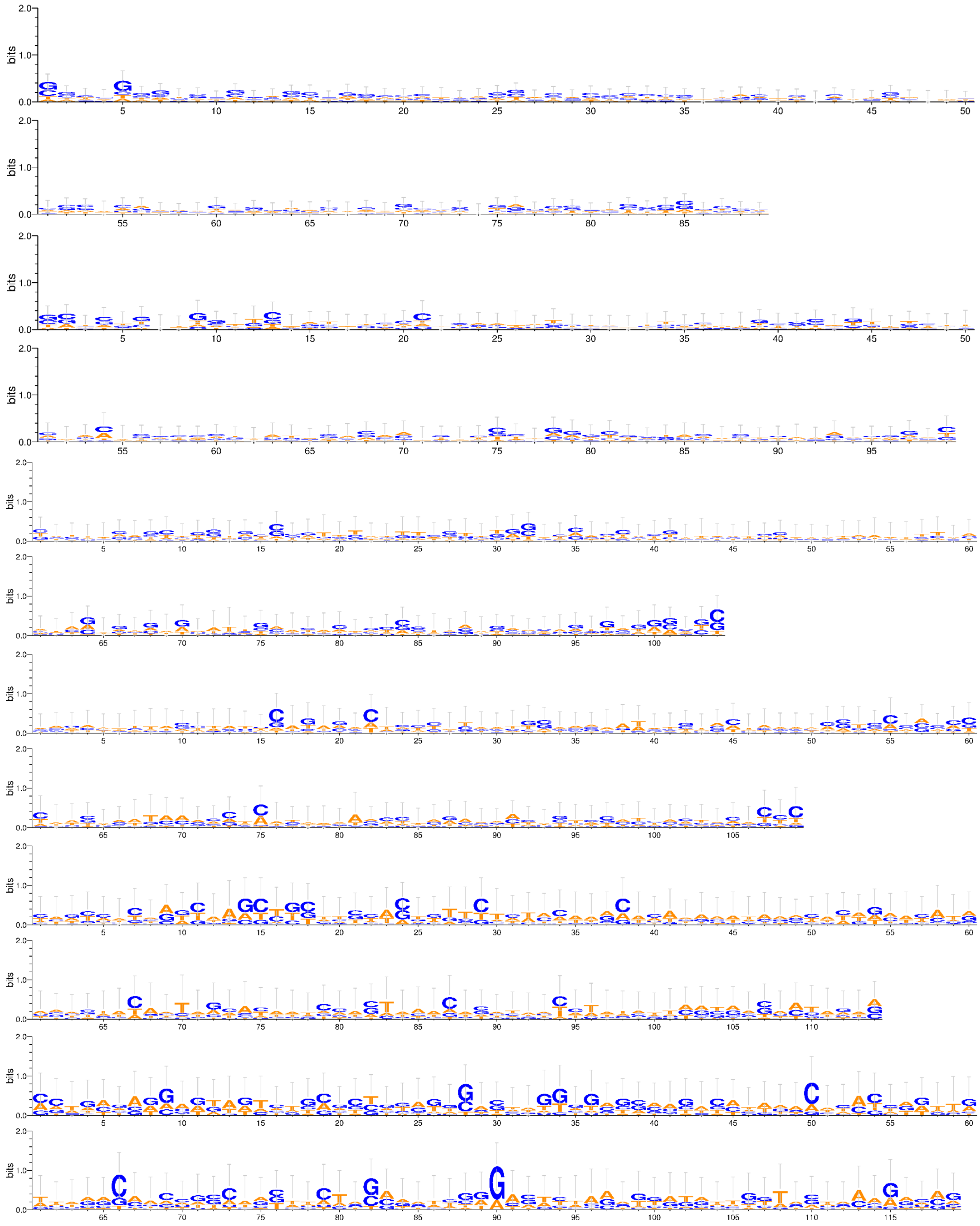


Figure 4.11: Consensus Logos of CpG Box with Length = 100, 105, 110, 115 and 120.

Table 4.4 Details of TSS results

	Gene	Direct Hit	Padding	Coverage	Length
Annotated	1157	343	13	0.307	348484
half eroded	580	146	78	0.386	210860
eroded	1157	296	192	0.421	715429
box+jump	1157	315	248	0.486	975624
box	1157	315	294	0.526	1648812

starting site is hindered by the precise epigenetic mechanism. Thus, many literature uses the TSS as the signal to show the accuracy of their prediction on CGIs since some articles find that up to 50% TSS are related to CpG islands [107].

TSS Data and Measurement The gene annotations of human genome (hg38) are utilized for the assessment. The coverage of transcriptional starting sites for each gene in chromosome 21 are tested as a measure to evaluate the our definition and the annotation of UCSC genome browser. The conventional CGI definition is about the broader genome area of CpG-enriched region than that of the CpG box definition. Thus, CpG box seems thinner than the CGI definition. some TSS might not exactlly locate at the predicted CpG box. Thus, we pad a 106 *nt* area [11] to the predicted CpG box and the same padding areas are also attached to UCSC CGIs.

The results of four parameters for CpG box definition as the top line are shown in Figure4.12. From the figure, it also can be observed that neither the annotated UCSC CGIs nor the predicted data from CpG box are the complete set related to TSS. That is, in specificity, they are possibly at the similar level. However, in sensitivity, our proposed CpG box can beat the annotated UCSC CGIs. Note that the padding area of CpG box is important because CpG sites may not be directly located at the starting sites. In Table 4.4, the details of TSS results are given.

Differentially Methylated Regions Recent findings have revealed that epigenetic markers are not only related with the present defined CGIs but also other non-defined CpG-

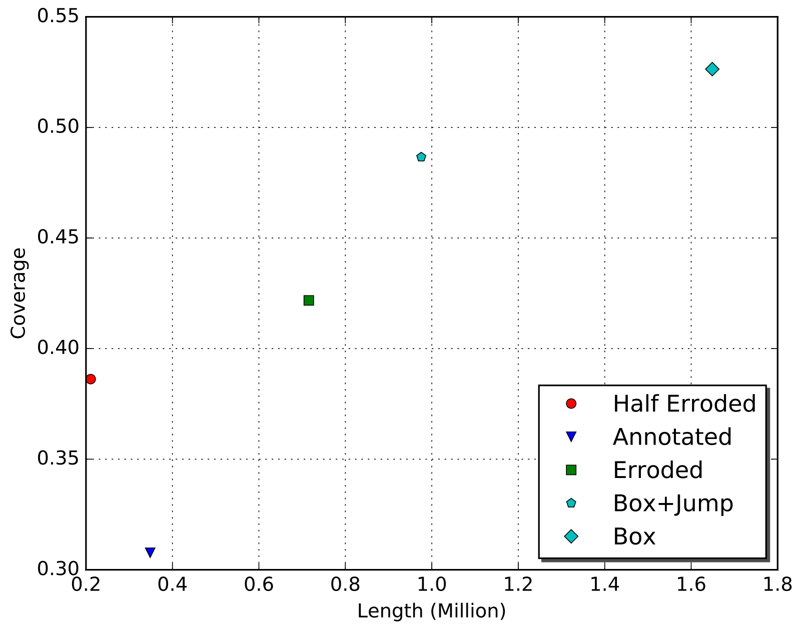


Figure 4.12: TSS Receiver-Operating-Characteristic-like Plot

It shows the TSS coverage used as a measure of the sensitivity and the total lengths for different CGI lists used as a measure of specificity. The plotted line is generated for different parameters. Box means only considering the likelihood of CpG box. Box+jump means considering the combination of the likelihood of CpG box and CpG box jump. Eroded means using erosion algorithm to eliminate the noise. Half eroded means that only the half data are processed for erosion due to the need of the comparison to the annotated CGI list.

rich regions [9]. The data from differentially methylated regions (DMR) further confirm these findings: DMRs appear not only in CGIs based on the current definition but also in CGI shores (a region, within 2,000 *nt* of CGIs), which indicate that (1) the current CGI definition is not complete enough to cover these DMR markers and (2) the higher coverage rate for DMR data means the stronger signal for CGI definition as an epigenetic marker.

DMR Data and Measurement Nine DMR data sets are derived from the methylation region data of different cells for human genome (hg18) using Methyl-seq method that was developed in the Myers laboratory to measure the methylation status at CpG sites throughout the whole genome sequence [101]. It combines DNA digestion by a methyl-sensitive enzyme HpaII and its methyl-insensitive isoschizomer MspI with the Illumina DNA sequencing platform. We take the chromosome 21 as the example again for illustrating the results since it has been a benchmark well studied by many research.

The Methyl-seq data contain more than 250,000 methyl-sensitive restriction enzyme cleavage sites, distributed to more than 90,000 genomic regions [101]. In the data set, over 65% methylation regions are not included in the present definition of CpG island.

In previous literature [108] [107], they measure the DMR coverage within 2000 bp of a CGI. However, since the traditional definition has the limitation on predicting the accurate location of methylated regions, we use the more strict measure to assess the results whether the CpG methylation region is directly located in CGI instead of the CGI shore (2000 bp of a CGI) by taking advantage of the definition of CpG box. We choose the annotated CGI lists of UCSC Genome Browser as the comparison for the assessment.

Four parameters of CpG box definition are assessed. The x-axis represents the size of generated data while y-axis represents the percentage of DMR covered. Only less than 30% DMR regions are related with the annotated CGI lists, namely the present defined CGIs, while the most conservative result of our CpG box definition can cover around 60% DMR regions as shown in Figure4.13. Note that the difference between eroded and box+jump is that the former conducts the signal-noise suppression for the result, that is, the single CpG box may be removed from the final results. It implies that some single CpG boxes may have the differentially methylated marker while they probably are excluded from the final results as the present definition.

From the results, the top line of CpG box definition has the higher sensitivity as well as higher specificity than UCSC genome browser that was regarded as the benchmark to measure the performance. The conventional CGI definition has the coarse granularity for predicting the CGIs because the criteria actually are threshold-based. In contrast, due to the fine-grained definition of CpG box and data-driven Markov model, our definition can better represent CpG island than others.

4.3 Cloud-assisted Platform for Investigating CpG Islands

4.3.1 Introduction

About 100 million to 10 billion human genomes could be sequenced by 2025 as sequencing costs have decreased dramatically in recent years [109]. Those high volumes of genome sequences lead to difficult tasks for scientists in the biological analysis. The conventional computing resources are not geared up to handle the biological big data. On the other side, as an emerging computing force, the cloud computing frameworks such as Apache Spark provide efficient and convenient ways for researchers to handle the procedures of big data and data analytics. Recent research findings in computational biology manifest the advances of cloud computing as an assisting-tool for big data and data analytics [110][111]. In this study, we apply cloud-assisted methods based on Apache Spark framework to the redefinition and the investigation of CpG island, an important epigenetic marker for biological processes on DNA genome sequences.

Spark Core The sequentially computing methods were adopted on a single-node workstation for the human genome sequence investigation (3 Gigabytes), which takes more than six hours to collect the data for only one transition table in Markov chain model. However, we aim to examine more species and more parameters analysis to investigate the

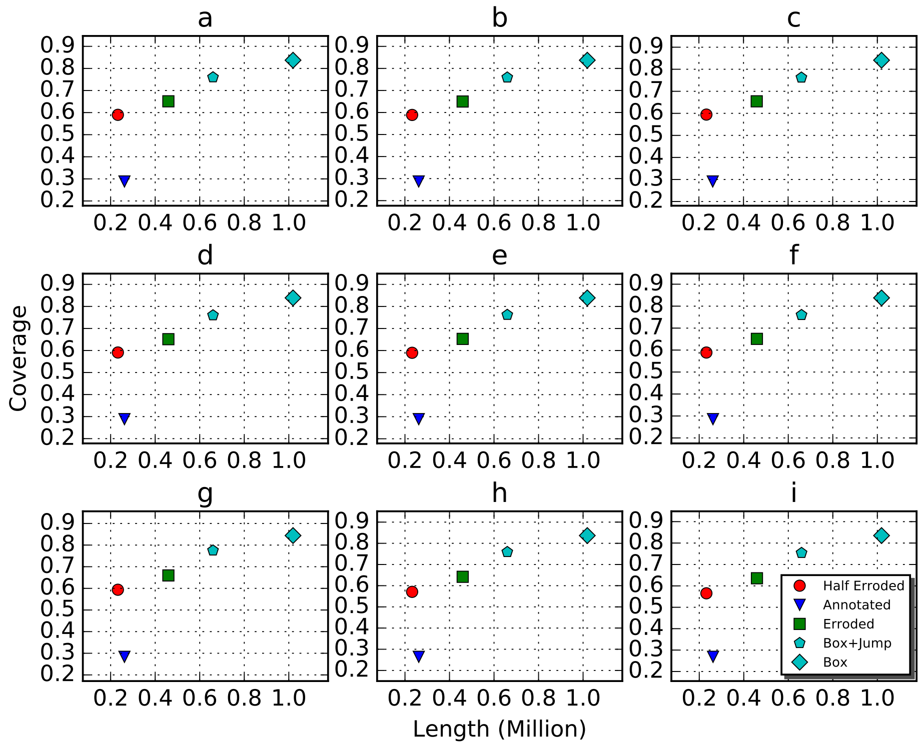


Figure 4.13: DMR Receiver-Operating-Characteristic-like Plot

It shows the DMR coverage used as a measure of the sensitivity and the total lengths for different CGI lists used as a measure of specificity. The plotted line is generated for different parameters. Box means considering only the likelihood of CpG box. Box+jump means considering the combination of the likelihood of CpG box and CpG box jump. Eroded means using erosion algorithm to eliminate the noise. Half eroded means that only the half data are processed for erosion due to the need of the comparison to the annotated CGI list. Nine data sets are a:K562, b:BGO2, c:GM12878, d:MethylSeqCalls.SL578, e:MethylSeqCalls.SL577, f:MethylSeq.605.604, g:MethylseqCalls.SL835.SL831, h:MethylseqCalls.SL832.SL828, and i:MethylseqCalls.SL833.SL829.

relevance of species-specific CpG islands and structural parameters. Obviously, the sequential computing methods cannot satisfy the data-intensive need in our studies.

The main reasons that Spark-based cloud platform is chosen to accommodate the proposed computing models are based on the following facts [112]:

First, Spark is regarded as a lightning fast cluster computing platform, which provides a faster and more generic processing mechanism than Hadoop. By supporting the programming language such as Scala, Java, Python and R with over 80 high-level operator APIs, Spark also makes it possible to write code more efficiently and more promptly. Moreover, Spark Core is the basic engine for large-scale parallel and distributed data processing. It provides multiple built-in functions, including job scheduling and distributing on clusters, memory management, fault recovery, communication and management with storage systems, and so on.

Second, the Resilient Distributed Dataset (RDD) [113] is an excellent concept for genome-based analysis, especially for CpG-box model that can be easily distributed to worker nodes for analytic operations. RDD is uniformed in spark programming paradigm, which is an immutable fault-tolerant and distributed collection of objects that can be executed in parallel. Any type of object can be contained in RDD. Moreover, loading an external data set or distributing a collection from the driver layer can result in the generation of RDD. Two types of operations are included in RDD: transformations and actions. Operations that are performed on a RDD, such as map, filter, group, join, union, and so on, are called transformations, whose results can be contained in a new generated RDD. Transformations in Spark are passive, meaning that they do not compute their results right away. Instead, they pre-configure the operation to be performed and the data set to which the operation is to be performed. On the contrary, so-called actions are operations that return a value after running a computation on a RDD, such as reduce, count, first, and so on. Actions are executed immediately by Spark system without delay.

Third, distinguished from other computing forces, Spark provides cache methods that can preserve an RDD in memory even though each transformed RDD may be recomputed each time by default when you perform an action on it. It results that users can keep the elements around on the cluster for much faster access the next time when query is received. Supported by the cache mechanism on the cluster, Spark can achieve superior computing performances over other computational platforms such as Hadoop [112].

By taking advantage of the aforementioned computational merits of Spark platform, we develop the Scala-based Spark pipeline, which is an ad hoc application on epigenetic analysis and can be executed in an interactive way that opens up the possibility of customizing cloud-based parallel algorithms for epigenetic analysis. The application of cloud-assisted methods on CpG islands is the first application on epigenetic genome analysis over Spark platform.

The rest parts of this paper are organized as follows: Subsection 4.3.2 depicts the methodology, the design and the implementation of the models; Section 4.3.3 shows the results and assessments; Section 4.3.4 makes a conclusion and discusses the future work.

4.3.2 Design and Implementation

Flow Chart The flow chart in Figure4.14 is the pipeline of the main processes. CpG islands are extracted from genome sequences in single node because the extraction does not take long time and frequent communications may downgrade the system performance. All extracted CpG islands are sent to distributed nodes as RDD for grouping operations performed on distributed nodes that sort the CpG islands following the order of their lengths.

The transition matrix of Markov chain model and the possibility matrix of maximum likelihood model are calculated based on the sorted CGIs. These operations are performed by worker nodes and distributed jobs are assigned by Spark built-in job scheduling mechanism. In practice, two transition probability matrices are frequently computed in terms of different parameters (the length of CpG box) following the analysis needs. For example, the length of CpG island greater than 200 *nt* or a particular threshold may be omitted by particular analysis. That is, each threshold indicates different matrices and *de novo* computation. Thus, cloud-assisted method is a pragmatic way to meet different needs, and it turns out efficient for whole genome analysis described in subsection 4.3.3, only 20 seconds with 10

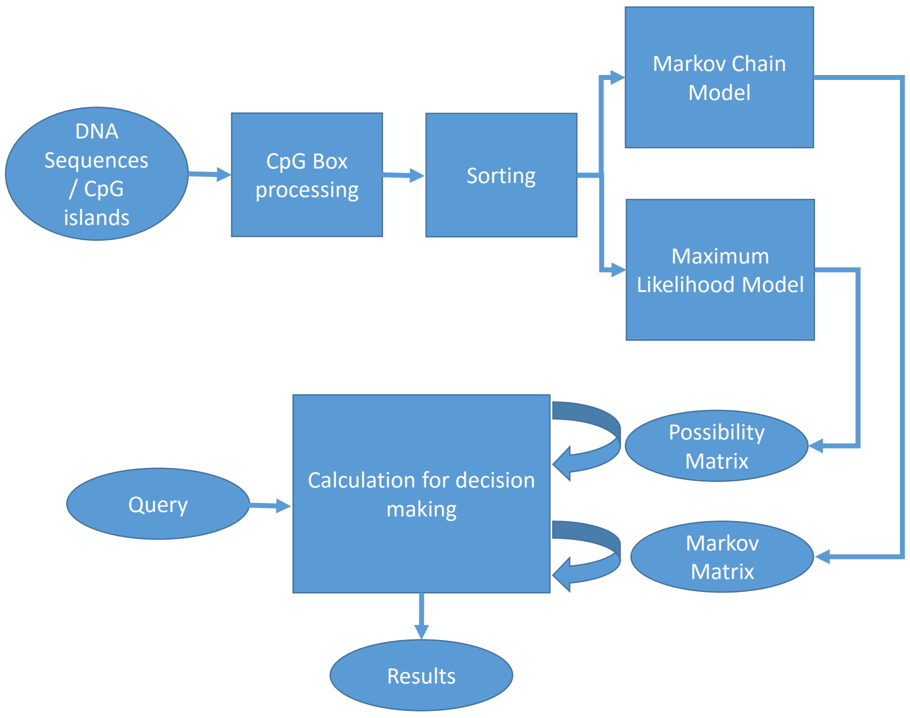


Figure 4.14: A Brief Flow Chart of Program Implementation

cores on 5 nodes.

Finally, the query for an unknown sequence are derived from the user-end. The decision-making procedure can be made in a single node since only a small amount of computing is involved based on the calculated matrices.

Main Procedures and Algorithms The main procedures are about calculating two transition probability matrices. The below code snippet is the calculation of Markov matrix for CpG boxes with the same length. *cpg_list* is a RDD that stores all CpG boxes in distributed nodes. *countLetter* is a map function that counts the probability at each position required by Equation 4.6. That is, all worker nodes concurrently run *countLetter* on data *cpg_list* when *collect*() operation is performed because actions actually drive passive transformation *map*() as we have discussed on Subsection 4.3.1.

```
//Sort the CpG boxes in length.
val cpg_list_groupby = cpg_list.groupBy(x => x.length)
```

```
//Calculate the possibilities.
val result = cpg_list_groupby.map{
    case(n, list:Seq[String])
    => countLetter(n, list) }
    .collect()
```

The key algorithm of calculating the possibility of neighboring CpG boxes is shown as the following code snippet. It makes use of RDD's *zipWithIndex*() and *leftOuterjoin*() operations to take statistics of neighboring CpG boxes. The scala-based Spark programming paradigm makes coding concise and efficient.

```
//Calculate the probability matrix
val box_jump_list =
    cpg_list.zipWithIndex()
    .map { case (v, i) => i -> v
    }.leftOuterJoin(
             cpg_list.zipWithIndex()
               .map {case (v, i)
                  => i - 1 -> v}
    ).flatMap { case (i, (a, b))
       => b.map(a -> _)}
```

4.3.3 Performance Evaluation

Human and mouse chromosome sequences, 24 chromosomes and 21 chromosomes respectively, are used for the validation of our cloud-assisted methods. From three main aspects on computing performance, the experiments are designed including evaluating the unit computing cost, testing the running time of large scale data, and analyzing the speedup rate for multiple cores.

System Configuration The experiments on evaluating the cloud-assisted method are performed on a private cloud. Five computing nodes and one master node are constructed on Apache Spark architecture, where the total executor memory is configured to 50G, and the total number of executor cores on running-time experiments of chromosomes human and mouse is set to 50. Hardware specification is Dual Intel Xeon E5-2650 and 64 GB DDR3 (1866MHz).

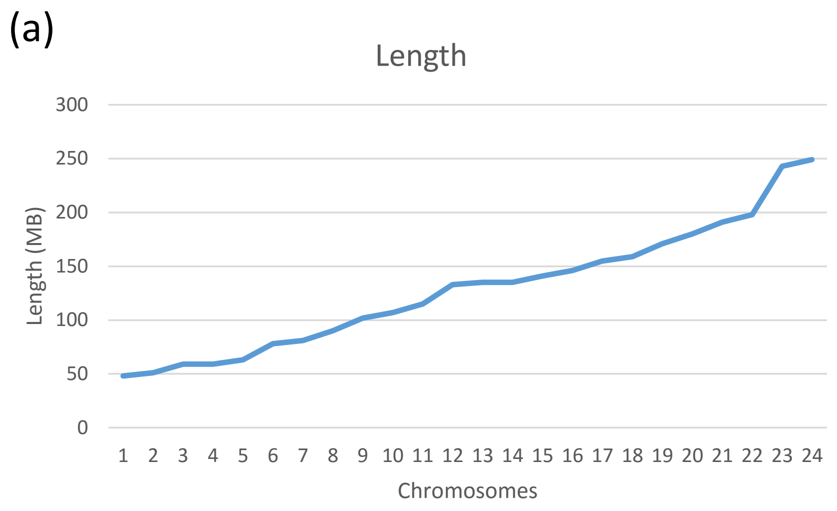
Chromosome Test The first experiment aims to test the executing performance for human and mouse chromosomes respectively. The details are shown as Figure4.15 and Figure4.16. All chromosomes are sorted in length and the running time have an apparent increase with an increasing chromosome length. However, the time/length ratio has an approximately decreasing trend shown as Figure4.15.(c) and Figure4.16.(c), meaning that the cloud-computing method has the ability to better handle large-scale data so that the unit computing time/cost per MBytes decreases. Note that some fluctuations on running time may occur in the Spark execution. It might be caused by some chromosomes contain particular CpG box patterns, such as extra long CpG boxes that are filtered out by pre-configured thresholds.

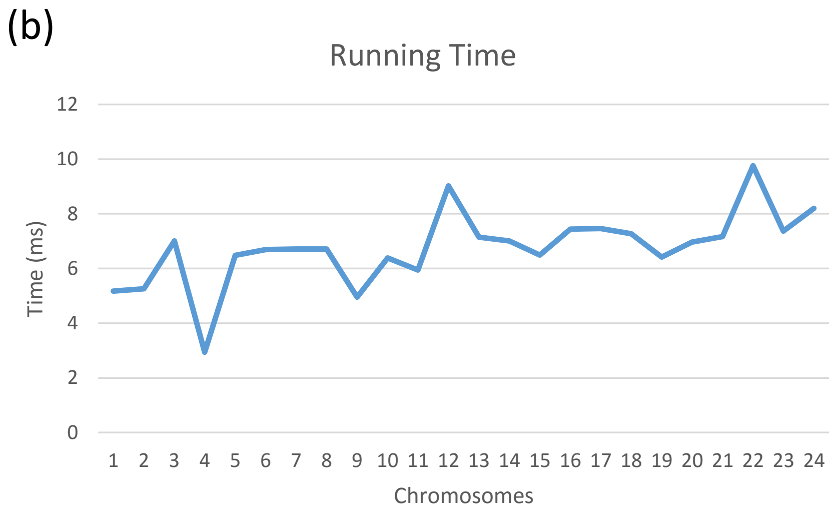
Whole-Genome Test and Speedup Analysis This test aims to evaluate the capability of this cloud-assisted method on handling large-scale data sets. The whole genomes of human (3 GBytes) and mouse (2.7 GBytes) are used to test the consumed time following a particular number of cores on Spark system. The illustration is shown as Figure4.17.

The speedup analysis is shown as Figure4.18 based on the experiment of whole genome test. By analyzing the speedup ratio for different core configuration, we can clearly see the speedup performance of our cloud-assisted method constructed on Spark platform: 6-7 times speedup rate when using 10 cores.

4.3.4 Conclusion

CpG box is a novel measure proposed in this project for the fine-grained measurement and the investigation into the existing CpG islands. Taking advantage of biological big data and existing annotated data, we use CpG box as the elementary unit to establish the Markov chain model and estimate the maximum likelihood for the structure of CpG box





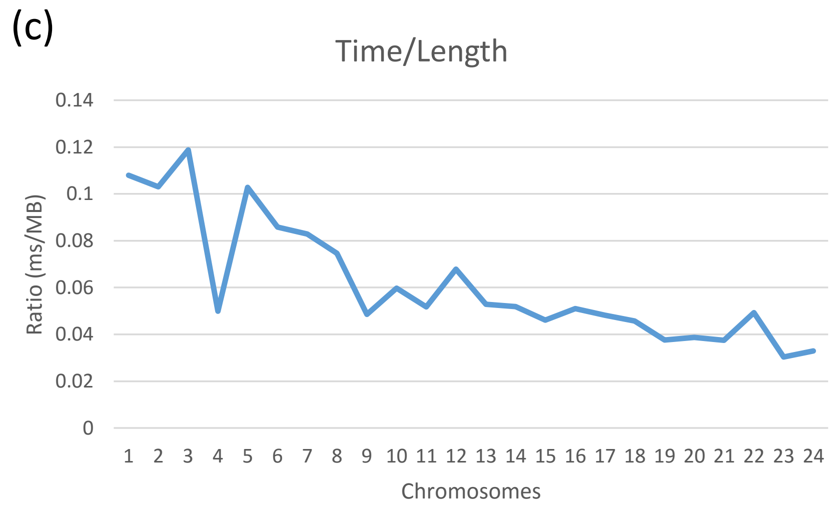


Figure 4.15: Running Time & Length of Human Chromosomes

Note that x-axis represents chromosomes that are sorted in the ascending order of length.

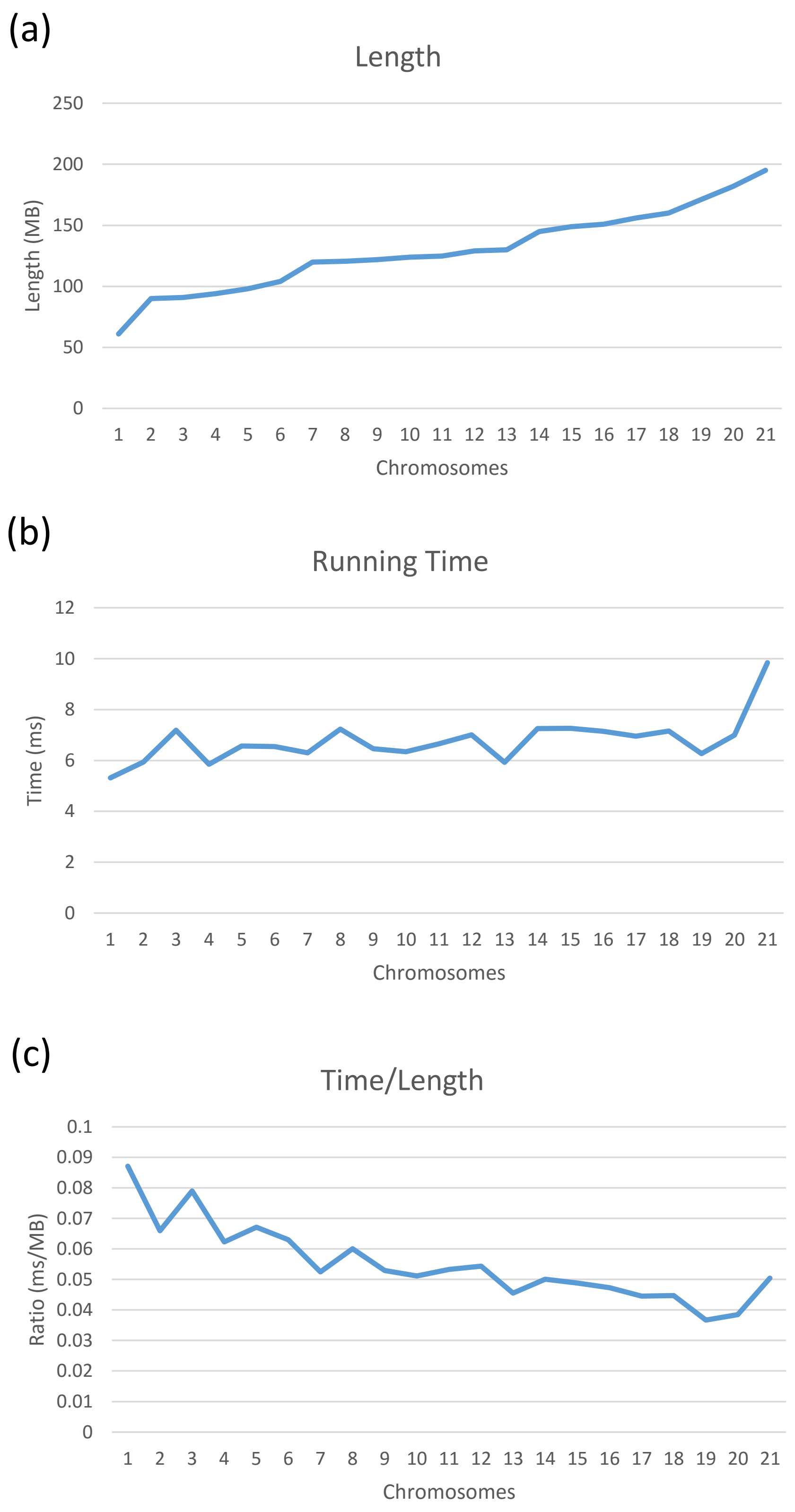


Figure 4.16: Running Time & Length of Mouse Chromosomes

Note that x-axis represents chromosomes that are sorted in the ascending order of length.

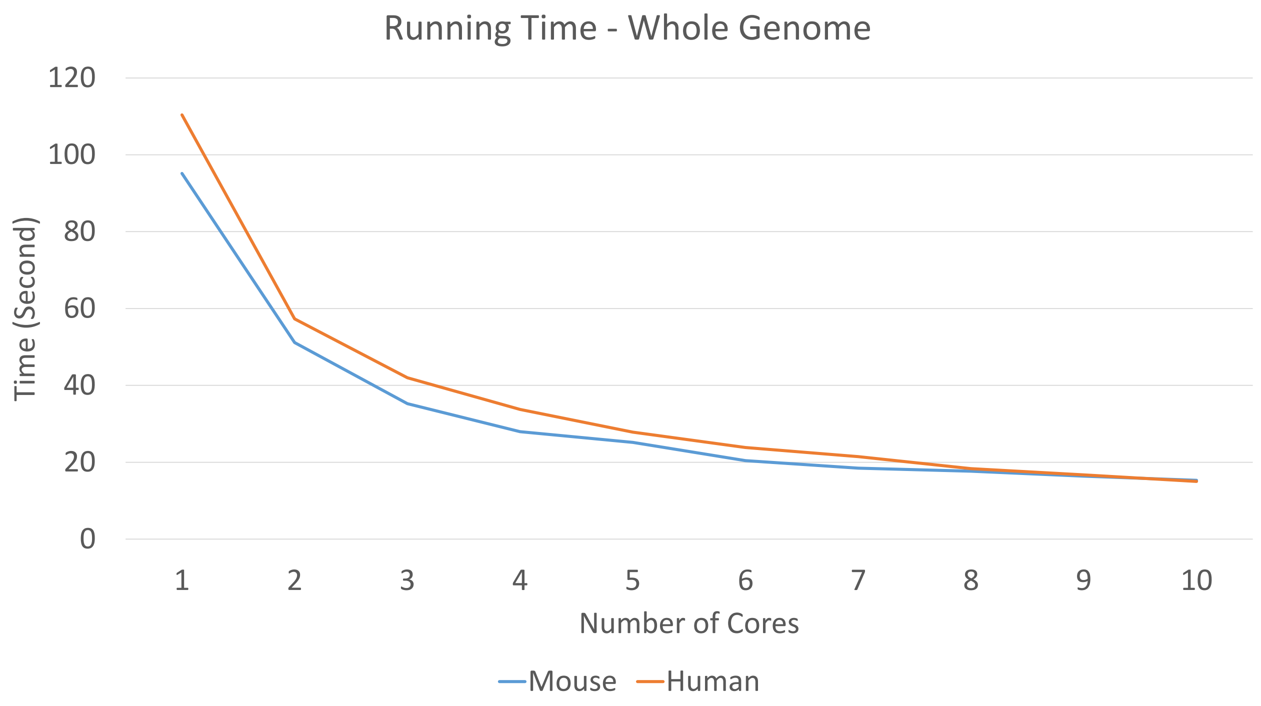


Figure 4.17: The Correlation of Running Time and Cores Using Whole Genomes of Human and Mouse

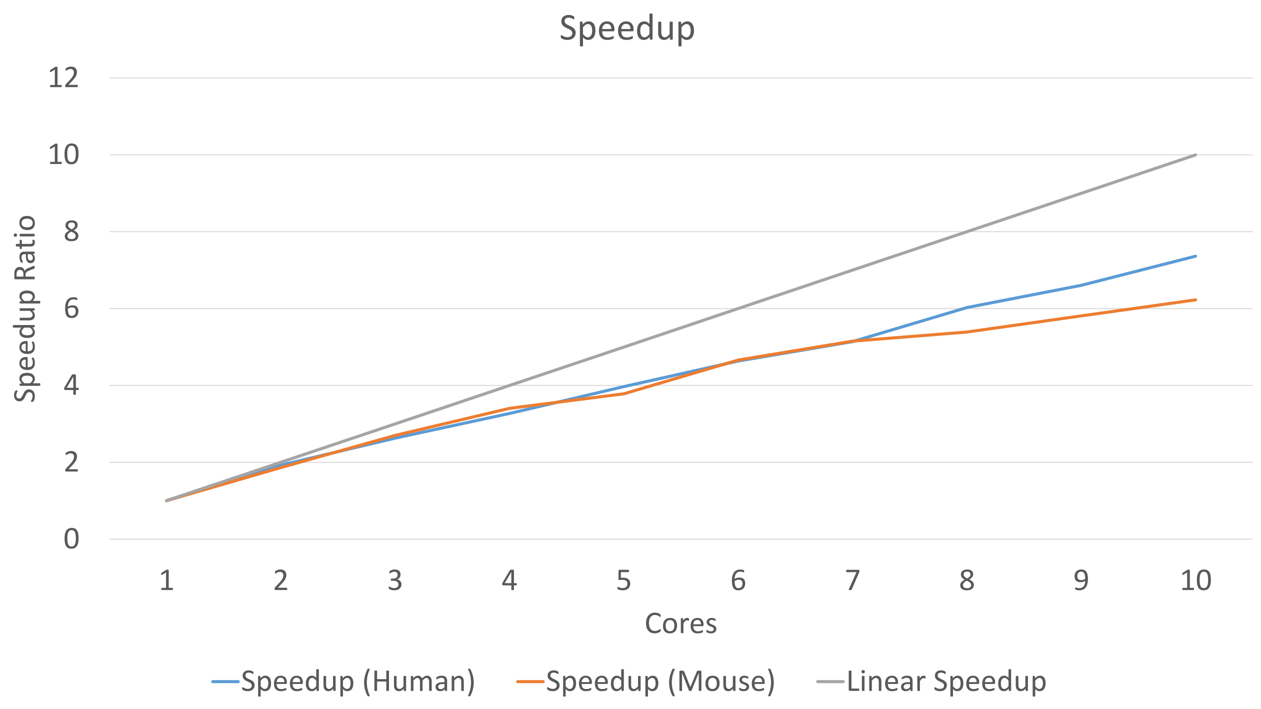


Figure 4.18: Speedup Ratios of Human and Mouse Genomes

and the neighboring CpG box. It not only reflects the biological properties (more related to TSS and DMR signals than the current annotation) but also shows the capability to improve the computing efficiency for whole genome analysis. Furthermore, it perfectly fits the state-of-the-art cloud platform.

Comparing with the current definition of CpG island, the proposed CpG box investigation can better respond to the data sets of differentially methylated regions and transcriptional starting sites in both sensitivity and specificity. Thus, we recommend the novel CpG box definition and the Markov chain model to replace the current criteria of CpG island for the fine-grained definition instead of the coarse-grained, threshold-based and outdated definition.

By using cloud-assisted method based on Spark architecture, we reduce the performing time for handling large-scale genome data. Moreover, this cloud-assisted method is first applied into the epigenetic analysis with the equivalent accuracy and faster processing capability compared with sequential processing. The advantages of Spark system, including Resilient Distributed Dataset, Spark programming paradigms, job scheduling, and so forth, are well manifested on this genome-analysis-based project. The performance evaluation of this cloud-assisted application on epigenetic analysis validates the successful combination of the two areas. Also, it stands for one of future directions about how to accelerate the processing and analysis on big data in biology, not limited to epigenetic analysis. Thus, in the future we will continue to design new methods to assist biologists for resolving biological big data issues. Concretely, a fully-fledged cloud platform is expected to establish for handling multiple big-data analytic tasks, including genomic analysis, sequence operations, genetic analysis and epigenetic analysis, etc.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This dissertation aims to explore the methodology in diverse computing technologies, including digital signal processing technology, information-coding theory, statistics, cloud computing, deep learning, artificial intelligence and so forth, which can be applied in genomic analysis and processing. With the combination of these cutting-edge techniques, it can improve the performance in solving some practical problems in computational biology. In this dissertation three topics/problems are discussed and the performance of modern computing techniques are comprehensively illustrated. The success of these proposed solutions can provide good paradigms for applying these state-of-the-art techniques to biology in the future.

Concretely, in the first problem, a novel signal and information encoding based method is proposed to increase the accuracy in gene structure prediction. This method is an instance of DNA-As-X framework, which converts the DNA sequence into binary code and the distance is calculated in a simple Exclusive-OR operation for similarity analysis. The encoding scheme is proposed based on the bio-chemical properties of DNA sequence and the hamming distance can directly reflect the bio-chemical difference. Thus, the encoding method is more appropriate on homology-based comparative studies than other methods according to performance evaluation.

In the second problem, deep neural network methods are developed for DNA sequence annotation, including applications such as detection of protein-coding splicing sites, recognition of lincRNA transcription splicing sites and the hybrid method for improvement in gene structure prediction. Nine different encoding schemes and four auto-encoder based deep learning techniques are studied for discovering how encoding schemes could affect the performance of deep learning and exploring the most appropriate solutions for these applications.

In the third problem, three main contributions are made. (1) A Gaussian digital filter based method is developed for detection of CpG island. This is a novel method applying digital signal processing in genome analysis. (2) CpG box and Markov model are developed to redefine and investigate the CpG island. According to the experimental results, the novel measures are more related with transcription starting sites and differentially methylated regions. Meanwhile, the measures can be easily used for detecting the structures of CpG island and investigating the epigenetic properties of DNA sequence. (3) A state-of-the-art cloud-assisted platform is developed to accommodate the investigation of CpG island based on the proposed CpG box and Markov model.

The methodology design and its applications are the primary targets in the future. On the aspect of methodology, the hybrid method becomes the main trend in our community, especially in the interdisciplinary fields. As a result, numerous novel methods have been invented by combining the concepts and principles in conventionally different areas. In this dissertation, the philosophy and its practice have been manifested a lot. For example, in problem #1, signal processing, information encoding, and similarity detection are combined so that the performance can be improved. In problem #2, diverse encoding schemes and cutting-edge deep learning algorithms are integrated for various DNA annotation applications. In problem #3, the signal processing, statistical methods and cloud computing are combined for a hybrid method. Thus, in the future, studying these hybrid methods is one of the main ways to solve the problems in interdisciplinary fields, particularly in the big data area.

For future studies, these proposed methods are worthy of further development. For example, three problems in this dissertation are about fundamental issues, which provide potential spaces and a broad range for hybrid techniques to further exert their power. In problem #1, the information-coding method can be extended into many other areas, such as evolutionary studies, mapping and assembly, fast processing in FPGA and cloud computing etc. In problem #2, the cutting-edge deep learning techniques in artificial intelligence is

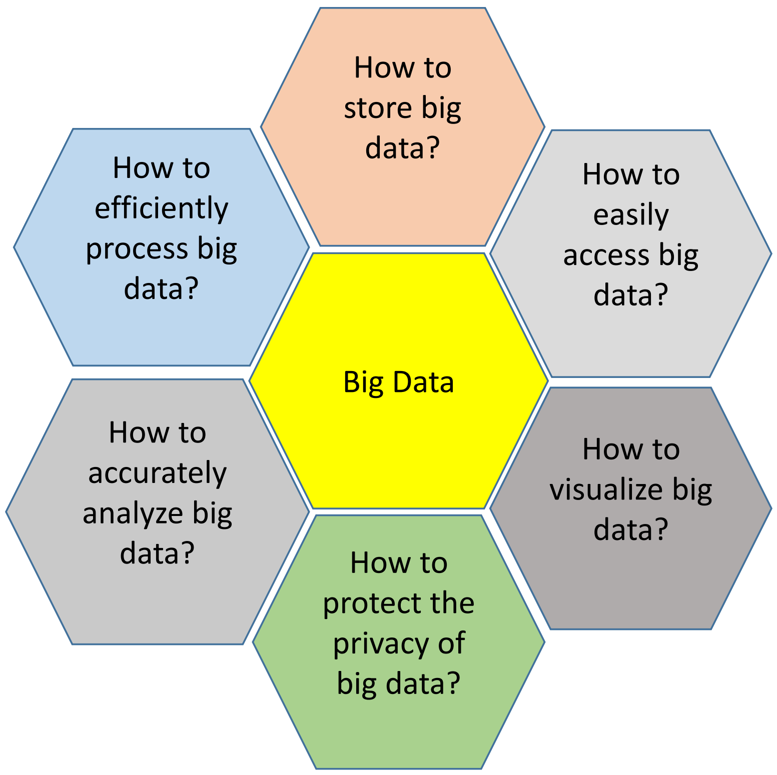

How to store big data?
How to efficiently process big data?
How to easily access big data?
Big Data
How to accurately analyze big data?
How to visualize big data?
How to protect the privacy of big data?

Figure 5.1: Six Main Problems in Big Data.

ubiquitously penetrating into almost all aspects of modern society including bioinformatics, especially pattern-recognition related issues. Further discovery on how to identify lincRNAs and understand their interactions from abundant DNA sequences is the key to unveil many unknown biological mechanisms. LincRNAs are also directly related with epigenetic markers such as CpG islands that are discussed a lot in problem #3. CpG island is the fundamental element in epigenetics to understand methylation and other epigenetic procedures. The CGI investigation in other species such as mouse and dog may be conducted in the future. Also, the correlation of CGI and microRNA needs further discovery using proposed CGI techniques. The latter one is a small non-coding RNA molecule that functions in silencing and gene regulation and probably has certain connections with CGI methylation due to altered expression of microRNAs [114]. In addition, through the studies on CpG island and lincRNA, aging problems and disease related issues can be related to my future application areas in computational biology.

In a big picture, the big data issue, including biological big data, is currently one of the main challenges for scientists, especially for computer scientists. In the big data areas, six main problems are proposed as shown in Figure5.1: (1) how to efficiently process big data; (2) how to accurately analyze big data; (3) how to store big data; (4) how to easily access big data; (5) how to visualize big data; (6) how to protect the privacy of big data. This dissertation is a part of the first two problems. In the future, all my efforts will be around the six problems. Novel hybrid methods are expected to apply into these big data problems and contribute to bridging the gap between computation and biology.

REFERENCES

[1] V. Marx, “Biology: The big challenges of big data,” *Nature*, vol. 498, no. 7453, p. 255260, June 2013.

[2] J. S. Aguilar-Ruiz, J. H. Moore, and M. D. Ritchie, “Filling the gap between biology and computer science,” *BioData Mining*, vol. 1, no. 1, pp. 1–3, July 2008.

[3] H. Kwan, B. Kwan, and J. Kwan, “Novel methodologies for spectral classification of exon and intron sequences,” *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, 2012.

[4] P. D. Cristea, “Conversion of nucleotides sequences into genomic signals,” *Journal of Cellular and Molecular Medicine*, vol. 6, no. 2, pp. 279–303, 2002.

[5] R. F. Voss, “Evolution of long-range fractal correlations and $1/f$ noise in DNA base sequences,” *Phys. Rev. Lett.*, vol. 68, pp. 3805–3808, Jun 1992.

[6] G. Kauer and H. Blcker, “Applying signal theory to the analysis of biomolecules,” *Bioinformatics*, vol. 19, no. 16, pp. 2016–2021, 2003.

[7] G. L. Rosen, “Signal processing for bibiological-inspired gradient source localization and dna sequence analysis,” Ph.D. dissertation, Georgia Institute of Technology, School of Electrical and Computer Engineering, August 2006.

[8] N. Yu, X. Guo, F. Gu, and Y. Pan, “DNA AS X: An information-coding-based model to improve the sensitivity in comparative gene analysis,” in *11th International Symposium on Bioinformatics Research and Applications*, Norfolk, Virginia, June 2015.

[9] H. Wu, B. Caffo, H. A. Jaffee, R. A. Irizarry, and A. P. Feinberg, “Redefining CpG islands using hidden markov models,” *Biostatistics*, vol. 11, no. 3, pp. 499–514, 2010.

[10] M. Hackenberg, C. Previti, P. Luque-Escamilla, P. Carpena, J. Martinez-Aroza, and J. Oliver, "CpGcluster: a distance-based algorithm for CpG-island detection," *BMC Bioinformatics*, vol. 7, no. 1, p. 446, 2006.

[11] N. Yu, X. Guo, A. Zelikovsky, and Y. Pan, "GaussianCpG: A gaussian model for detection of human CpG island," in *IEEE 5th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS),*, Miami, FL, Oct 2015, p. 1.

[12] N. Bray, I. Dubchak, and L. Pachter, "AVID: A global alignment program," *Genome Research*, vol. 13, no. 1, pp. 97–102, 2003.

[13] NCBI, "Standard nucleotide BLAST," Jun. 2015. [Online]. Available: https://blast.ncbi.nlm.nih.gov/

[14] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 85, no. 8, p. 24442448, 1988.

[15] R. A. Cartwright, "Ngila: global pairwise alignments with logarithmic and affine gap costs," *Bioinformatics*, vol. 23, no. 11, pp. 1427–1428, 2007.

[16] T. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, June 1984.

[17] M. Chaisson and G. Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory," *BMC Bioinformatics*, vol. 13, no. 1, p. 238, 2012.

[18] A. M. S. Shrestha, M. C. Frith, and P. Horton, "A bioinformaticians guide to the forefront of suffix array construction algorithms," *Briefings in Bioinformatics*, 2014.

[19] B. Ma, J. Trompand, and M. Li, "Patternhunter: faster and more sensitive homology search," *Bioinformatics*, vol. 18, no. 3, pp. 440–445, 2002.

[20] S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[21] M. C. Frith, “A new repeat-masking method enables specific detection of homologous sequences,” *Nucleic Acids Research*, vol. 39, no. 4, p. e23, 2011.

[22] S. M. Kie *l*basa, R. Wan, K. Sato, P. Horton, and M. C. Frith, “Adaptive seeds tame genomic sequence comparison,” *Genome Research*, vol. 21, no. 3, pp. 487–493, 2011.

[23] M. C. Frith and L. No*é*, “Improved search heuristics find 20 000 new alignments between human and mouse genomes,” *Nucleic Acids Research*, vol. 42, no. 7, p. e59, 2014.

[24] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller, “Humanmouse alignments with BLASTZ,” *Genome Research*, vol. 13, no. 1, pp. 103–107, 2003.

[25] W. Trimble, K. Keegan, M. D’Souza, A. Wilke, J. Wilkening, J. Gilbert, and F. Meyer, “Short-read reading-frame predictors are not created equal: sequence error causes loss of signal,” *BMC Bioinformatics*, vol. 13, no. 1, p. 183, 2012.

[26] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*, 2nd ed. the University of Michigan: Cold Spring Harbor Laboratory Press, 2004.

[27] D. Lipman and W. Pearson, “Rapid and sensitive protein similarity searches,” *Science*, vol. 227, pp. 1435–1441, March 1985.

[28] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, “Basic local alignment search tool,” *Journal of Molecular Biology*, vol. 215, pp. 403–410, Octobor 1990.

[29] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of Molecular Biology*, vol. 48, pp. 443–453, 1970.

[30] I. Korf, P. Flicek, D. Duan, and M. R. Brent, "Integrating genomic homology into gene structure prediction," *Bioinformatics*, vol. 17, no. suppl 1, pp. S140–S148, 2001.

[31] D. W. Enrico Siragusa and K. Reinert, "Fast and accurate read mapping with approximate seeds and multiple backtracking," *Nucleic Acids Research*, vol. 41, no. 7, April 2013.

[32] I. Dubchak, A. Poliakov, A. Kislyuk, and M. Brudno, "Multiple whole-genome alignments without a reference organism," *Genome Res.*, vol. 19, pp. 682–689, April 2009.

[33] R. E. Blahut, *Algebraic Codes for Data Transmission*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003.

[34] K. J. Breslauer, R. Frank, H. Blcker, and L. A. Marky, "Predicting DNA duplex stability from the base sequence," *Proceedings of the National Academy of Sciences*, vol. 83, no. 11, pp. 3746–3750, 1986.

[35] F. Crick, "Codon and anticodon pairing: the wobble hypothesis," *Journal of Molecular Biology*, vol. 19, pp. 548–555, 1966.

[36] S. Lin and D. J. Costello, *Error control coding: fundamentals and applications.* Upper Saddle River: Pearson-Prentice Hall, 2004, vol. 114.

[37] R. S. Harris, "Improved pairwise alignment of genomic DNA," *Ph.D. Thesis, The Pennsylvania State University*, 2007.

[38] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, Feb 1978.

[39] G. Al-Naymat, S. Chawla, and J. Taheri, "SparseDTW: A novel approach to speed up dynamic time warping," *CoRR*, vol. abs/1201.2969, 2012.

[40] S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander, “Human and mouse gene structure: Comparative analysis and application to exon prediction,” *Genome Res.*, vol. 10, p. 950958, July 2000.

[41] W. Kent, C. Sugnet, T. Furey, K. Roskin, T. Pringle, A. Zahler, and D. Haussler, “UCSC genome browser,” *Genome Res.*, vol. 12, no. 6, pp. 996–1006, June 2002.

[42] M. Akhtar, J. Epps, and E. Ambikairajah, “Signal processing in sequence analysis: Advances in Eukaryotic gene prediction,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 3, pp. 310–321, June 2008.

[43] B. J. Yoon, “Hidden markov models and their applications in biological sequence analysis,” *Current Genomic*, vol. 10, pp. 402–415, 2009.

[44] ENCODE, “An integrated encyclopedia of DNA elements in the human genome,” *Nature*, vol. 489, no. 7414, pp. 57–74, September 2012.

[45] M. K. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, “Deep learning of the tissue-regulated splicing code,” *Bioinformatics*, vol. 30, no. 12, pp. i121–i129, 2014.

[46] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[47] W. Zhu, A. Lomsadze, and M. Borodovsky, “Ab initio gene identification in metagenomic sequences,” *Nucleic Acids Research*, vol. 38, no. 12, p. e132, 2010.

[48] C. Burge and S. Karlin, “Prediction of complete gene structures in human genomic DNA,” *Journal of Molecular Biology*, vol. 268, no. 1, pp. 78 – 94, 1997.

[49] M. Borodovsky and J. McIninch, “GENMARK: Parallel gene recognition for both DNA strands,” *Computers & Chemistry*, vol. 17, no. 2, pp. 123 – 133, 1993.

[50] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman, “A generalized hidden Markov model for the recognition of human genes in DNA.” in *Proc Int Conf Intell Syst Mol Biol.*, vol. 1996, no. 4, 1996, pp. 134–142.

[51] E. C. Uberbacher and R. J. Mural, “Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach.” *Proc Natl Acad Sci U S A.*, vol. 88, pp. 11 261–11 265, December 1991.

[52] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Scholköpf, and G. Rätsch, “Support vector machines and kernels for computational biology,” *PLoS Comput Biol*, vol. 4, no. 10, p. e1000173, 10 2008.

[53] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller, “Engineering support vector machine kernels that recognize translation initiation sites,” *Bioinformatics*, vol. 16, no. 9, pp. 799–807, 2000.

[54] S. Sonnenburg, A. Zien, and G. Rätsch, “Arts: accurate recognition of transcription starts in human,” *Bioinformatics*, vol. 22, no. 14, pp. e472–e480, 2006.

[55] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Ratsch, “Accurate splice site prediction using support vector machines,” *BMC Bioinformatics*, vol. 8, no. Suppl 10, p. S7, 2007.

[56] H. Liu, H. Han, J. Li, and L. Wong, “An ¡i¿in-silico¡/i¿ method for prediction of polyadenylation signals in human sequences,” *Genome Informatics*, vol. 14, pp. 84–93, 2003.

[57] D. Rose, M. Hiller, K. Schutt, J. Hackermller, R. Backofen, and P. F. Stadler, “Computational discovery of human coding and non-coding transcripts with conserved splice sites,” *Bioinformatics*, vol. 27, no. 14, pp. 1894–1900, 2011.

[58] G. Schweikert, A. Zien, G. Zeller, J. Behr, C. Dieterich, and C. S. Ong, “mGene:

Accurate SVM-based gene finding with an application to nematode genomes." *Genome Research*, vol. 19, no. 11, p. 21332143, 2009.

[59] G. Rätsch, S. Sonnenburg, and B. Schölkopf, "RASE: recognition of alternatively spliced exons in c.elegans," *Bioinformatics*, vol. 21, no. suppl 1, pp. i369–i377, 2005.

[60] O. Abbasi, A. Rostami, and G. Karimian, "Identification of exonic regions in dna sequences using cross-correlation and noise suppression by discrete wavelet transform," *BMC Bioinformatics*, vol. 12, no. 1, p. 430, 2011.

[61] H. K. Kwan and S. Arniker, "Numerical representation of DNA sequences," in *Electro/Information Technology, 2009. eit '09. IEEE International Conference on*, June 2009, pp. 307–310.

[62] R. F. Voss, "Evolution of long-range fractal correlations and 1/ f noise in DNA base sequences," *Phys. Rev. Lett.*, vol. 68, pp. 3805–3808, Jun 1992.

[63] T. Holden, R. Subramaniam, R. Sullivan, E. Cheung, C. Schneider, G. Tremberger, Jr., A. Flamholz, D. H. Lieberman, and T. D. Cheung, "ATCG nucleotide fluctuation of Deinococcus radiodurans radiation genes," vol. 6694, 2007, pp. 669 417–669 417–10.

[64] H. E. Stanley, S. V. Buldyrev, A. L. Goldberger, Z. D. Goldberger, S. Havlin, S. M. Ossadnik, C.-K. Peng, and M. Simmons, "Statistical mechanics in biology: how ubiquitous are long-range correlations," *Physica A*, vol. 205, no. 1-3, p. 214253, April 1994.

[65] A. Nair and S. Sreenadhan., "A coding measure scheme employing electron-ion interaction pseudopotential (EIIP)," *Bioinformation*, vol. 1, no. 6, pp. 197–202, 2006.

[66] M. Garzon and R. Deaton, "Codeword design and information encoding in DNA ensembles," *Natural Computing*, vol. 3, no. 3, pp. 253–292, 2004.

[67] A. T. M. G. Bari, M. R. Reaz, A. K. M. T. Islam, H.-J. Choi, and B.-S. Jeong, "Effective encoding for DNA sequence visualization based on nucleotideś ring structure." *Evolutionary Bioinformatics*, vol. 2013, no. 9, pp. 251–261, 2013.

[68] W. Deng and Y. Luan, "Analysis of similarity/dissimilarity of DNA sequences based on chaos game representation," *Abstract and Applied Analysis*, vol. 2013, no. 926519, 2013.

[69] S. Vinga, A. Carvalho, A. Francisco, L. Russo, and J. Almeida, "Pattern matching through chaos game representation: bridging numerical and discrete data structures for biological sequence analysis," *Algorithms for Molecular Biology*, vol. 7, no. 1, p. 10, 2012.

[70] C. K. Peng, S. V. Buldyrev, A. L. Goldberger, S. Havlin, F. Sciortino, M. Simons, and H. E. Stanley, "Long-range correlations in nucleotide sequences," *Nature*, vol. 356, no. 6365, pp. 168–170, 1992.

[71] G. Hinton, P. Dayan, B. Frey, and R. Neal, "The "wake-sleep" algorithm for unsupervised neural networks," *Science*, vol. 268, no. 5214, pp. 1158–1161, 1995.

[72] G. E. Hintonemail, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, no. 10, pp. 428 – 434, Oct 2007.

[73] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 8599–8603.

[74] P. Di Lena, K. Nagata, and P. Baldi, "Deep architectures for protein contact map prediction," *Bioinformatics*, vol. 28, no. 19, pp. 2449–2457, 2012.

[75] J. Eickholt and J. Cheng, "Predicting protein residueresidue contacts using deep networks and boosting," *Bioinformatics*, vol. 28, no. 23, pp. 3066–3072, 2012.

[76] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U. D. Montral, and M. Qubec, "Greedy layer-wise training of deep networks," in *In NIPS*. MIT Press, 2007.

[77] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 1096–1103. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390294

[78] M. Q. Zhang, "Computational prediction of eukaryotic protein-coding genes," *Nature Reviews Genetics*, vol. 3, pp. 698–709, September 2002.

[79] C. Trapnell, L. Pachter, and S. L. Salzberg, "TopHat: discovering splice junctions with RNA-Seq," *Bioinformatics*, vol. 25, no. 9, pp. 1105–1111, 2009.

[80] M. Reese, F. Eeckman, D. Kulp, and D. Haussler, "Improved splice site detection in genie'," *J Comp Biol*, vol. 4, no. 3, pp. 311–323, 1997.

[81] M. J. Hangauer, I. W. Vaughn, and M. T. McManus, "Pervasive transcription of the human genome produces thousands of previously unidentified long intergenic noncoding rnas," *PLoS Genet*, vol. 9, no. 6, pp. 1–13, 06 2013.

[82] S. Katayama, Y. Tomaru, T. Kasukawa, K. Waki, M. Nakanishi, M. Nakamura, H. Nishida, C. C. Yap, M. Suzuki, J. Kawai, H. Suzuki, P. Carninci, Y. Hayashizaki, C. Wells, M. Frith, T. Ravasi, K. C. Pang, J. Hallinan, J. Mattick, D. A. Hume, L. Lipovich, S. Batalov, P. G. Engström, Y. Mizuno, M. A. Faghihi, A. Sandelin, A. M. Chalk, S. Mottagui-Tabar, Z. Liang, B. Lenhard, and C. Wahlestedt, "Antisense transcription in the mammalian transcriptome," *Science*, vol. 309, no. 5740, pp. 1564–1566, 2005.

[83] J. Durruthy-Durruthy, V. Sebastiano, M. Wossidlo, D. Cepeda, J. Cui, E. J. Grow, J. Davila, M. Mall, W. H. Wong, J. Wysocka, K. F. Au, and R. A. Reijo Pera, "The primate-specific noncoding rna hpat5 regulates pluripotency during human preimplantation development and nuclear reprogramming," *Nature Genetics*, vol. 48, no. 1, pp. 44–52, January 2016.

[84] C. Yang, E. Bolotin, T. Jiang, F. M. Sladek, and E. Martinez, “Prevalence of the initiator over the TATA box in human and yeast genes and identification of DNA motifs enriched in human TATA-less core promoters,” *Gene*, vol. 389, no. 1, pp. 52–65, 2007.

[85] P. Bucher, “Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences,” *Journal of Molecular Biology*, vol. 212, no. 4, pp. 563 – 578, 1990.

[86] R. Kakumani, O. Ahmad, and V. Devabhaktuni, “Identification of CpG islands in DNA sequences using statistically optimal null filters,” *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2012, no. 1, p. 12, 2012.

[87] A. Meissner, T. S. Mikkelsen, H. Gu, M. Wernig, J. Hanna, A. Sivachenko, X. Zhang, B. E. Bernstein, C. Nusbaum, D. B. Jaffe, A. Gnirke, R. Jaenisch, and E. S. L. and, “Genome-scale DNA methylation maps of pluripotent and differentiated cells,” *Nature*, vol. 454, no. 7205, pp. 766–770, 2008.

[88] M. Gardiner-Garden and M. Frommer, “CpG islands in vertebrate genomes,” *Journal of Molecular Biology*, vol. 196, no. 2, pp. 261 – 282, 1987.

[89] D. Takai and P. A. Jones, “Comprehensive analysis of CpG islands in human chromosomes 21 and 22,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 6, pp. pp. 3740–3745, 2002.

[90] L. Ponger and D. Mouchiroud, “CpGProD: identifying CpG islands associated with transcription start sites in large genomic mammalian sequences,” *Bioinformatics*, vol. 18, no. 4, pp. 631–633, 2002.

[91] P. Rice, I. Longden, and A. Bleasby, “EMBOSS: The european molecular biology open software suite,” *Trends in Genetics*, vol. 16, no. 6, pp. 276–277, 2000.

[92] L.-Y. Chuang, C.-H. Yang, M.-C. Lin, and C.-H. Yang, "CpGPAP: CpG island predictor analysis platform," *BMC Genetics*, vol. 13, no. 1, p. 13, 2012.

[93] B.-J. Yoon and P. Vaidyanathan, "Identification of CpG islands using a bank of IIR lowpass filters DNA sequence detection," in *Digital Signal Processing Workshop, 2004 and the 3rd IEEE Signal Processing Education Workshop. 2004 IEEE 11th*, Aug 2004, pp. 315–319.

[94] S. Ye, A. Asaithambi, and Y. Liu, "CpGIF: an algorithm for the identification of CpG islands." *Bioinformation*, vol. 2, no. 8, pp. 335–338, 2008.

[95] N. Elango and S. V. Yi, "Functional relevance of CpG island length for regulation of gene expression," *Genetics*, vol. 187, no. 4, pp. 1077–1083, 2011.

[96] D. Xu, "Energy, entropy and information potential for neural computation," Ph.D. dissertation, University of Florida, 1999.

[97] P. Schwerdtfeger, "The pseudopotential approximation in electronic structure theory," *ChemPhysChem*, vol. 12, no. 17, pp. 3143–3155, 2011.

[98] P. Deininger, "Alu elements: know the SINEs," *Genome Biology*, vol. 12, no. 12, p. 236, December 2011.

[99] L. E. Heisler, D. Torti, P. C. Boutros, J. Watson, C. Chan, N. Winegarden, M. Takahashi, P. Yau, T. H.-M. Huang, P. J. Farnham, I. Jurisica, J. R. Woodgett, R. Bremner, L. Z. Penn, and S. D. Der, "CpG island microarray probe sequences derived from a physical library are representative of CpG islands annotated on the human genome," *Nucleic Acids Research*, vol. 33, no. 9, pp. 2952–2961, 2005.

[100] C. Bock, J. Walter, M. Paulsen, and T. Lengauer, "CpG island mapping by epigenome prediction," *PLoS Comput Biol*, vol. 3, no. 6, p. e110, 06 2007.

[101] A. L. Brunner, D. S. Johnson, S. W. Kim, A. Valouev, T. E. Reddy, N. F. Neff, E. Anton, C. Medina, L. Nguyen, E. Chiao, C. B. Oyolu, G. P. Schroth, D. M. Absher,

J. C. Baker, and R. M. Myers, “Distinct DNA methylation patterns characterize differentiated human embryonic stem cells and developing human fetal liver,” *Genome Research*, 2009.

[102] D. A. Orlando, M. G. Guenther, G. M. Frampton, and R. A. Young, “CpG island structure and trithorax/polycomb chromatin domains in human cells,” *Genomics*, vol. 100, no. 5, pp. 320 – 326, 2012.

[103] F. A. Feltus, E. K. Lee, J. F. Costello, C. Plass, and P. M. Vertino, “DNA motifs associated with aberrant CpG island methylation,” *Genomics*, vol. 87, no. 5, pp. 572 – 579, 2006.

[104] V. Afreixo, C. A. C. Bastos, A. J. Pinho, S. P. Garcia, and P. J. S. G. Ferreira, “Genome analysis with inter-nucleotide distances,” *Bioinformatics*, vol. 25, no. 23, pp. 3064–3070, 2009.

[105] B. Hutter, V. Helms, and M. Paulsen, “Tandem repeats in the CpG islands of imprinted genes,” *Genomics*, vol. 88, no. 3, pp. 323 – 332, 2006.

[106] G. E. Crooks, G. Hon, J. M. Chandonia, and S. E. Brenner, “Weblogo: A sequence logo generator,” *Genome Research*, vol. 14, pp. 1188–1190, 2004.

[107] R. A. Irizarry, C. Ladd-Acosta, B. Wen, Z. Wu, C. Montano, P. Onyango, H. Cui, K. Gabo, M. Rongione, M. Webster, H. Ji, J. Potash, S. Sabunciyan, and A. P. Feinberg, “Genome-wide methylation analysis of human colon cancer reveals similar hypo- and hypermethylation at conserved tissue-specific cpg island shores,” *Nature Genetics*, vol. 41, no. 2, p. 178186, 2009.

[108] J. L. Glass, R. F. Thompson, B. Khulan, M. E. Figueroa, E. N. Olivier, E. J. Oakley, G. Van Zant, E. E. Bouhassira, A. Melnick, A. Golden, M. J. Fazzari, and J. M. Greally, “CG dinucleotide clustering is a species-specific property of the genome,” *Nucleic Acids Research*, vol. 35, no. 20, pp. 6798–6807, 2007.

[109] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, “Big data: Astronomical or genomical?” *PLoS Biol*, vol. 13, no. 7, pp. 1–11, 07 2015.

[110] M. S. Wiewiórka, A. Messina, A. Pacholewska, S. Maffioletti, P. Gawrysiak, and M. J. Okoniewski, “Sparkseq: fast, scalable and cloud-ready tool for the interactive genomic data analysis with nucleotide precision,” *Bioinformatics*, vol. 30, no. 18, pp. 2652–2653, 2014.

[111] X. Guo, Y. Meng, N. Yu, and Y. Pan, “Cloud computing for detecting high-order genome-wide epistatic interaction via dynamic clustering,” *BMC Bioinformatics*, vol. 15, no. 1, p. 102, 2014.

[112] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets,” in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863113

[113] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI’12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228301

[114] K. Truninger, M. Menigatti, J. Luz, A. Russell, R. Haider, J.-O. Gebbers, F. Bannwart, H. Yurtsever, J. Neuweiler, H.-M. Riehle, M. S. Cattaruzza, K. Heinimann, P. Schär, J. Jiricny, and G. Marra, “Immunohistochemical analysis reveals high frequency of pms2 defects in colorectal cancer,” *Gastroenterology*, vol. 128, no. 5, pp. 1160 – 1171, 2005.