Signal Processing Methods for Genomic Sequence Analysis

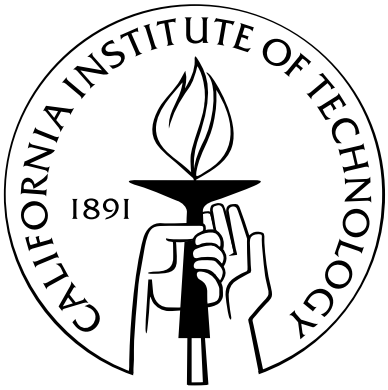
Thesis by

Byung-Jun Yoon

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2007

(Defended November 22, 2006)

Acknowledgments

First of all, I would like to express my deepest gratitude to my advisor, Professor P. P. Vaidyanathan, for his excellent guidance and support throughout my graduate studies. In fact, he was the best advisor, mentor, teacher, and researcher, I could ever think of. From my very first day at Caltech, he was always there for me whenever I needed his guidance. He taught me so many important things that are fundamental to a good researcher, for example, how to think creatively, how to develop ideas, how to write good papers, how to be a good presenter, and most importantly, how to stand alone as an independent researcher. It has been my greatest pleasure to learn from P. P., who was the perfect role model.

I also would like to thank the members of my defense and candidacy committees: Professor Yaser S. Abu-Mostafa, Professor Babak Hassibi, Professor Christina D. Smolke, Professor Tracey Ho, Professor John Doyle, Professor Changhuei Yang, and Professor Morteza Gharib. I also would like to thank the Microsoft Research, the Korea Foundation for Advanced Studies (KFAS), the National Science Foundation (NSF), and the Office of Naval Research (ONR) for their generous financial support during my graduate studies at Caltech.

One of the greatest things at Caltech was the opportunity to work with some of the brightest people in the world! I truly enjoyed working with my labmates in the Digital Signal Processing (DSP) lab, and my warmest appreciation goes to them: Dr. Bojan Vrcelj, Dr. Andre Tkacenko, Sriram Murali, Borching Su, Michael Larsen, Chun-Yang Chen, and Ching-Chih Weng. Thanks to these wonderful guys, working in the DSP lab was a sheer pleasure. I will deeply miss our discussions and conversations as well as the many conference trips that we made together. I also would like to thank Andrea Boyle, our wonderful secretary, for her kind assistance and professional support.

During my Ph.D., I had the wonderful opportunity to work at the Microsoft Research (MSR) in Redmond as an intern. I spent three summers at MSR, where I worked with Dr. Henrique (Rico)

Malvar during the first two internships and with Dr. Ivan Tashev during the third one. Rico was an extremely good mentor, who showed me what it means to maintain a perfect balance between theoretical research and practical applications. Although he was really busy with his work (well, you may not believe how many meetings he has as a director of MSR), he always found time for discussions and informal chats. I met Ivan during my third internship, where we worked on a project to develop a new adaptive microphone array processing algorithm. Ivan was a passionate mentor and he was very supportive and helpful throughout my internship. In fact, he taught me from A to Z about microphone arrays, although he himself was occupied with so many other things. During the project, I had a chance to build a linear microphone array by myself (it involved designing the array, cutting the plastic board using a laser cutter, soldering the wires, etc.), which was great fun! I am truly grateful to my MSR mentors, Rico and Ivan, who gave me these invaluable experiences that I will never forget.

I also would like to express my appreciation to my spiritual mentors, Pastor Chul-Min Kim and Pastor Byungjoo Song. I personally got to know Pastor Kim when I was a college student. At that time, I was still young and immature, and I was not so much interested in spiritual things. However, great passion is contagious, and my life started to change due to his unceasing passion for God and his constant love, care, and prayer for me. If Pastor Kim was the one who planted the seed of faith in me, it was Pastor Song who watered it so that it can grow further. Pastor Song was my mentor for the one-on-one discipleship training at the All Nations Church (a.k.a. Onnuri Church, LA). Being a passionate preacher, good theologian and apologist, he taught me many of the important Christian doctrines in a clear and logical manner. Both mentors have had a crucial impact in shaping myself as a Christian, and I hope I can live as I was taught and as I now believe.

One of life's greatest joys is to meet someone who can (and wants to) truly understand you. In this respect, I was greatly blessed to have many friends with whom I could share every aspect of my life. I am especially thankful to my Caltech friends Wonjin Jang and Hyunjoo Lee, for their friendship and support throughout my graduate studies. I can hardly imagine myself studying at Caltech without these awesome friends. My warmest thanks also goes to Chul-Gi Min and Jun-Sang Lee, my best friends in Korea, whom I have known since my (junior) high-school days.

Also, I would like to thank my parents, Choong-Yeol Yoon and Hyo-Kyung Yoo, for their unconditional love and unceasing support throughout my entire life. Their continuous encouragement and prayer have been the very source of power that sustained me during this tumultuous time of

my life, and I am really indebted to them for their sacrificial love. I am also grateful to my sister, Young-Ran Yoon, for being the best sister I could ever dream of! I also would like to thank my relatives for their encouragements and kind support. Especially, I would like to thank my uncle and aunt, Kang-Yeol Yoon and Nam-Hee Shin, and also aunt Hyo-Im Yoo for their amazing love!

Last, but by no means least, I would like to thank God, who has been always so faithful to me. As the apostle Paul has confessed, I am what I am by the amazing grace of God. Even though I did not deserve it, He never stopped loving me, and He has been (and He always will be) leading my life according to His perfect plan that is far beyond my imagination! I pray that I may glorify Him through everything I do, and enjoy Him – and *only* Him – throughout my entire life.

Soli Deo Gloria! To Him be the glory forever and ever. Amen.

Abstract

Signal processing is the art of representing, transforming, analyzing, and manipulating signals. It deals with a wide range of signals, from speech and audio signals to images and video signals, and many others. Signal processing techniques have been found very useful in diverse applications. Traditional applications include signal enhancement, denoising, speech recognition, audio and image compression, radar signal processing, and digital communications, just to name a few. In recent years, signal processing techniques have been also applied to the analysis of biological data with considerable success. For example, they have been used for predicting protein-coding genes, analyzing ECG signals and MRI data, enhancing and normalizing DNA microarray images, modeling gene regulatory networks, and so forth.

In this thesis, we consider the application of signal processing methods to the analysis of biological sequences, especially, DNA and RNA molecules. We demonstrate how conventional signal processing techniques–such as digital filters and filter banks–can contribute to this end, and also show how we can extend the traditional models–such as the hidden Markov models (HMMs)–to better serve this purpose.

The first part of the thesis focuses on signal processing methods that can be utilized for analyzing RNA sequences. The primary purposes of this part are to develop a statistical model that is suitable for representing RNA sequence profiles and to propose an effective framework that can be used for finding new homologues (i.e., similar RNAs that are biologically related) of known RNAs. Many functional RNAs have secondary structures that are well conserved among different species. The RNA secondary structure gives rise to long-range correlations between distant bases, which cannot be represented using traditional HMMs. In order to overcome this problem, we propose a new statistical model called the context-sensitive HMM (csHMM). The csHMM is an extension of the traditional HMM, where certain states have variable emission and transition probabilities that depend on the context. The context-sensitive property increases the descriptive

power of the model significantly, making csHMMs capable of representing long-range correlations between distant symbols. Based on the proposed model, we present efficient algorithms that can be used for finding the optimal state sequence and computing the probability of an observed symbol string. We also present a training algorithm that can be used for optimizing the parameters of a csHMM. We give several examples that illustrate how csHMMs can be used for modeling various RNA secondary structures and recognizing them.

Based on the concept of csHMM, we introduce profile-csHMMs, which are specifically constructed csHMMs that have linear repetitive structures (i.e., state-transition diagrams). Profile-csHMMs are especially useful for building probabilistic representations of RNA sequence families, including pseudoknots. We also propose a dynamic programming algorithm called the sequential component adjoining (SCA) algorithm that can systematically find the optimal state sequence of an observed symbol string based on a profile-csHMM. In order to demonstrate the effectiveness of profile-csHMMs, we build a structural alignment tool for RNA sequences and show that the profile-csHMM approach can yield highly accurate predictions at a relatively low computational cost. At the end, we describe how the profile-csHMM can be used for finding homologous RNAs, and we propose a practical scheme for making the search significantly faster without affecting the prediction accuracy.

In the second part of the thesis, we focus on the application of digital filters and filter banks in DNA sequence analysis. Firstly, we demonstrate how we can use digital filters for predicting protein-coding genes. Many coding regions in DNA molecules are known to display a period-3 behavior, which can be effectively detected using digital filters. Efficient schemes are proposed that can be used for designing such filters. Experimental results will show that the digital filtering approach can clearly identify the coding regions at a very low computational cost. Secondly, we propose a method based on a bank of IIR lowpass filters that can be used for predicting CpG islands, which are specific regions in DNA molecules that are abundant in the dinucleotide CpG. This filter bank is used to process the sequence of log-likelihood ratios obtained from two Markov chains, where the respective Markov chains model the base transition probabilities inside and outside the CpG islands. The locations of the CpG islands are predicted by analyzing the output signals of the filter bank. It will be shown that the filter bank approach can yield reliable prediction results without sacrificing the resolution of the predicted start/end positions of the CpG islands.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Signal processing is the art of representing, transforming, analyzing, and manipulating signals. It deals with a large variety of signals, including speech and audio signals, images and video signals, and even biological signals such as DNA sequences, ECG signals, MRI signals, and so forth. Signal processing techniques have been found useful in truly diverse applications, such as signal enhancement [33], denoising [14, 101], speech recognition [82], audio [73] and image compression [100, 118], radar signal processing [84], and digital communications [42, 43], just to name a few.

More recently, signal processing techniques have been also applied to the analysis of biological data with considerable success. For example, they have been utilized for the prediction of protein-coding genes [3, 106], characterization of ECG signals [63, 94], representation of gene regulatory networks [99], analysis of DNA microarray images [13, 115], and many others. The reader who is interested in the recent advances of signal processing methods in biology is referred to the following tutorial reviews [3, 22, 23, 112, 113, 130].

The primary focus of the thesis lies in the application of signal processing concepts to the analysis of genomic sequence data. Among other things, this thesis presents various signal processing methods–such as digital filters, filter banks, and variants of HMMs (hidden Markov models)–that have been found especially useful in analyzing DNA and RNA sequence data and identifying regions of specific interest–e.g., protein-coding genes, CpG islands, and noncoding RNA (ncRNA) genes–inside these sequences. A more detailed outline of the thesis is presented in Section 1.6 of this Chapter.

The main purpose of this introductory Chapter is to equip the readers with some fundamentals in signal processing and genomics that are needed to understand the technical details of the discussions that will follow. We make every attempt to make this Chapter as self-contained as possible, in

order to serve this purpose. In Section 1.1 we briefly describe the discrete Fourier transform (DFT) that can be used for isolating the signal component with certain periodicity. The basic concepts of Markov chains and hidden Markov models (HMMs) are reviewed in Section 1.2 and Section 1.3, respectively. In Section 1.4, we review some fundamentals in genomics, and in Section 1.5, we briefly describe the concept of RNA secondary structures.

The material presented in this Chapter is only meant to be a quick introduction to signal processing and genomics, and it is by no means a complete and comprehensive coverage of these topics. For a more extensive treatment of these topics, the reader is referred to the references given at the end of each section.

1.1 Discrete Fourier transform (DFT)

Let us consider a finite length signal $x(n)$ whose length is N. We assume that $x(n) = 0$ outside the range $0 \leq n \leq N-1$. The *discrete Fourier transform (DFT)* of $x(n)$ is defined as

$$X[k] = \sum_{n=0}^{N-1} x(n) W^{kn}, \tag{1.1}$$

where $W = e^{-j2\pi/N}$. Sometimes, we may assume that the signal $x(n)$ has length N even when its actual length is smaller. For example, it is possible that $x(n) = 0$ outside the range $0 \leq n \leq L = M-1$, where $M \leq N$. For this reason, the transform in (1.1) is sometimes called the N-point DFT, in order to prevent any ambiguity. As the Fourier transform of $x(n)$ is defined as

$$X(e^{j\omega}) = \sum_{n} x(n) e^{-jwn}, \tag{1.2}$$

we can view the DFT $X[k]$ of a signal $x(n)$ as a uniformly sampled version of $X(e^{j\omega})$ at frequencies $\omega_k = \frac{2\pi k}{N}$ for $k = 0, 1, \ldots, N-1$. Given the DFT coefficients $X[k]$, we can reconstruct the original signal $x(n)$ as follows

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W^{-kn}. \tag{1.3}$$

This transform is called the *inverse-DFT* of $X[k]$. There exist efficient algorithms for computing the DFT and its inverse (IDFT), which are collectively called the fast Fourier transform (FFT) algorithms [24]. While the direct computation of the (N-point) DFT coefficients requires $O(N^2)$ opera-

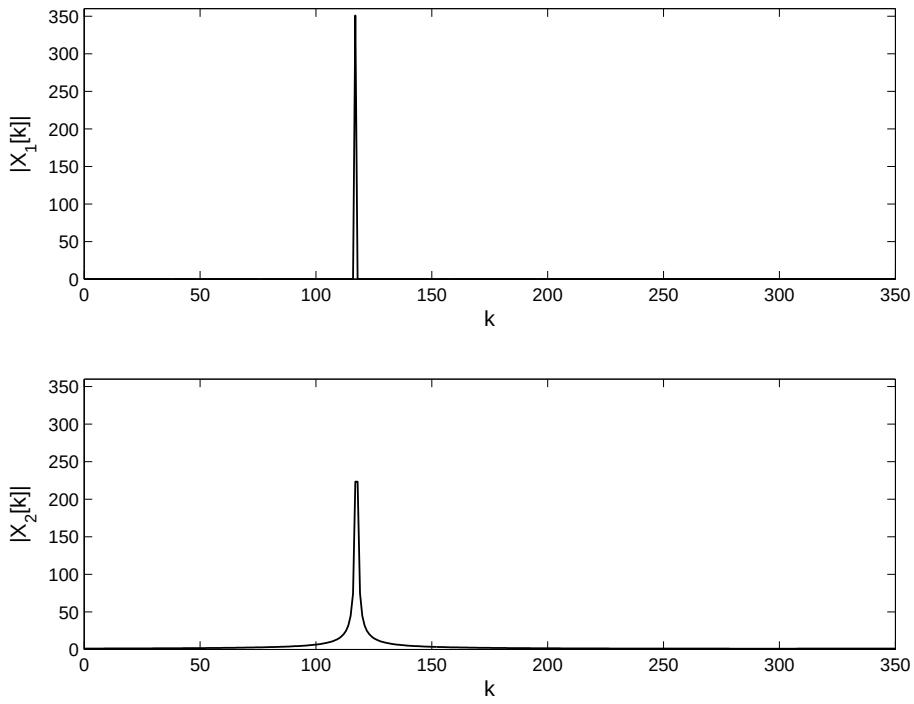


Figure 1.1: DFTs of periodic signals. (Top) Magnitude plot of the DFT of a signal with a period $T = 3$. (Bottom) Magnitude plot of the DFT of a signal with a period $T = 351/117.5$.

tions, the FFT algorithm can compute the DFT in only $O(N \log N)$ operations.[1]

The DFT $X[k]$ of a signal $x(n)$ can effectively analyze the frequency components contained in $x(n)$. For example, the DFT coefficient $X[k]$ shows the strength of the signal component whose frequency is located at (or near) $\omega = 2\pi k/N$. This corresponds to the period $T = N/k$ in the time domain. Let us consider a finite-length periodic signal $x_1(n)$ defined as follows

$$x_1(n) = e^{j2\pi n/3},\ 0 \le n \le N-1, \tag{1.4}$$

where the length of the signal is $N = 351$. As the period of $x_1(n)$ if 3, its DFT $X_1[k] = \text{DFT}[x_1(n)]$ should have a peak at $\omega = 2\pi/3$. This is demonstrated in Figure 1.1 (Top), where the magnitude of the DFT $X_1[k]$ has only a single peak at $k = N/3 = 117$. When the frequency of the input signal is not located at one of the frequencies $\omega_k = \frac{2\pi k}{N}$ ($k = 0, 1, \ldots, N-1$), its energy will be distributed

[1]The notation $O(\cdot)$ means "on the order of".

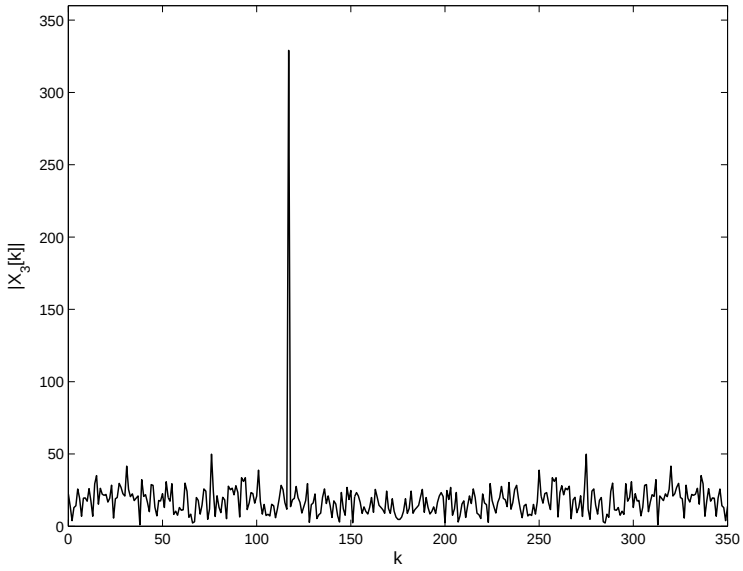


Figure 1.2: The DFT of a periodic signal with a period $T = 3$ buried in Gaussian noise. We can observe a clear peak at $k = N/3$.

over several frequency bins. For example, let us consider another periodic signal $x_2(n)$ defined as

$$x_2(n) = e^{j2\pi \frac{(N/3)+0.5}{N} n},\ 0 \leq n \leq N-1,$$

where $N = 351$ is the same as before. The magnitude plot of the DFT $X_2[k] = \text{DFT}[x_2(n)]$ is shown in Figure 1.1 (Bottom). We can see that the there are many non-zero DFT coefficients in this case, although the peak is located near $k = \frac{N/3+0.5}{N}$.

Finally, let us consider a periodic signal that is buried in noise. We define

$$x_3(n) = x_1(n) + z(n),\ 0 \leq n \leq N-1$$

where $x_1(n)$ is a periodic signal with period $T = 3$ as defined in (1.4) and $z(n)$ is white Gaussian noise with unit variance. The magnitude plot of $X_3[k] = \text{DFT}[x_3(n)]$ is shown in Figure 1.2. In Figure 1.2, we can clearly observe the peak at $k = N/3$ that corresponds to the period $T = 3$ of the signal $x_1(n)$.

In Chapter 5, we will show how this property can be used for identifying protein-coding regions in DNA sequences. Further details of the DFT and its basic properties can be found in [74].

1.2 Markov chain

Let us consider a system that can be described by one of a finite number of states $\mathcal{S} = \{S_1, \ldots, S_M\}$. At each discrete time index n, the state y_n of the system takes one of the values $y_n \in \mathcal{S}$. The system makes a state-transition at each unit time, which gives rise to a sequence of states

$$y_0 \rightarrow y_1 \rightarrow y_2 \rightarrow \ldots \rightarrow y_n. \tag{1.5}$$

We say that this discrete-time stochastic process satisfies the *Markov property*, if the probability distribution of the future state y_{n+1} depends only on the present state y_n and not on the past states y_{n-k} $(k \geq 1)$. This can be written as

$$P(y_{n+1} = S_j | y_n = S_i, y_{n-1} = S_k, y_{n-2} = S_\ell, \ldots) = P(y_{n+1} = S_j | y_n = S_i). \tag{1.6}$$

In this case, the state sequence in (1.5) is called a first-order Markov chain (Markov model). If the transition probability shown in (1.6) is time independent such that

$$P(y_{n+1} = S_j | y_n = S_i) = t(S_i, S_j),$$

for all n, it is called a *stationary* Markov chain. The transition probabilities $t(S_i, S_j)$ satisfy

$$\begin{aligned} t(S_i, S_j) &\geq 0 \qquad (1 \leq i, j \leq M), \\ \sum_{j=1}^{M} t(S_i, S_j) &= 1 \qquad (1 \leq i \leq M). \end{aligned}$$

As we can see from above, a stationary Markov chain is completely governed by its state transition probabilities $t(S_i, S_j)$, and it can be conveniently represented by a state transition diagram. An example of such a diagram is shown in Figure 1.3. Unless mentioned otherwise, we assume that the Markov chain that is being used is stationary.

In many applications, Markov chains are used to model the correlations between observable physical events. Every state represents a distinct event, and the state transition probabilities describe the correlations between these events. Once we have constructed a model that properly describes the system that gives rise to the observable events, this model can be used to evaluate the probability of observing a series of events. For example, let us consider the following problem.

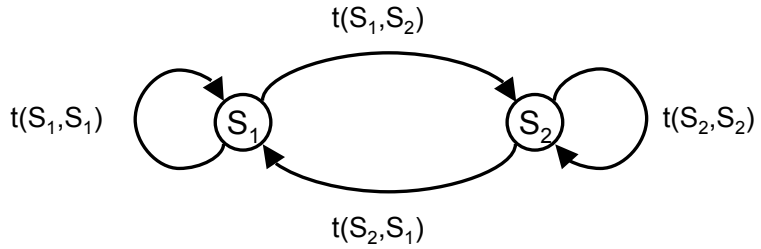


Figure 1.3: Example of a state transition diagram that represents a Markov chain with two distinct states S_1 and S_2.

Given that the current state is $y_0 \in \mathcal{S}$, how can we compute the probability that the next L states will be exactly $y_1 y_2 \dots y_L$? Using the Markov property, this probability can be simply computed as

$$\begin{aligned} P(y_1 y_2 \dots y_L | y_0) &= \prod_{n=1}^{L} P(y_n | y_{n-1}) \\ &= \prod_{n=1}^{L} t(y_{n-1}, y_n). \end{aligned} \tag{1.7}$$

Consider the case when we have multiple Markov chains, where each model describes the behavior of a system under different conditions. In this case, we can use the observation probability in (1.7) to determine which model describes the observed events best.

The Markov chains are utilized in Chapter 6, where they are used to represent the base sequences inside CpG islands and those outside CpG islands. It is demonstrated that they can be effectively used for discriminating CpG islands from the non-CpG island regions. For further details on Markov chains, the reader is referred to [89].

1.3 Hidden Markov model (HMM)

For many real world problems, the assumption that each state in the Markov chain corresponds to an "observable" event may be too restrictive. In such cases, we can use the hidden Markov model (HMM) that is an extension of the simpler Markov model. The HMM is a doubly embedded stochastic process that consists of an invisible process of hidden states and a visible process of observable symbols (or events). The hidden state sequence satisfies the Markov property, which is governed by the state transition probabilities associated with the model. The probability distribution of the observed symbols depend on the underlying states. More formally, this can be written

as follows. Let $x_n \in \mathcal{A}$ be the observed symbol at time n, where $\mathcal{A} = \{a_1, \ldots, a_N\}$ is the set of all observable symbols. We denote the underlying state at time n as $y_n \in \mathcal{S}$, where $\mathcal{S} = \{S_1, \ldots, S_M\}$ is the set of distinct states in the hidden Markov model. As the state sequence satisfies the Markov property, we have

$$\begin{aligned} P(y_{n+1} = S_j | y_n = S_i, y_{n-1} = S_k, \ldots) &= P(y_{n+1} = S_j | y_n = S_i) \\ &= t(S_i, S_j). \end{aligned}$$

At time n, the emission probability of the observed symbol x_n depends on the hidden state y_n

$$P(x_n = a_k | y_n = S_\ell) = e(a_k | S_\ell).$$

Note that $t(S_i, S_j)$ is the stationary state transition probability from state S_i to state S_j, and $e(a_k|S_\ell)$ is the stationary symbol emission probability of symbol a_k at state S_ℓ. The stochastic process of hidden states and the process of observable symbols are illustrated in Figure 1.4. In general, the hidden state sequence (typically called a "path") cannot be directly inferred from the observed symbol sequence, although some information about the state sequence can be obtained from the observation. An HMM is completely defined by the set of parameters $\Theta = \{T, E, \pi\}$, where the matrices $T = \{t_{ij}\}$ and $E = \{e_{\ell k}\}$ and the vector $\pi = \{\pi_i\}$ are defined as follows.

$$\begin{aligned} t_{ij} &= t(S_i, S_j), \\ e_{\ell k} &= e(a_k | S_\ell), \\ \pi_i &= P(y_1 = S_i), \end{aligned}$$

for $1 \leq i, j, \ell \leq M$ and $1 \leq k \leq N$. T is the transition probability matrix, E is the emission probability matrix, and π is the initial state distribution of the model.

Like Markov chains, HMMs can also be conveniently represented by state transition diagrams. Figure 1.5 shows an example of such a diagram. The HMM shown in Figure 1.5 has the following

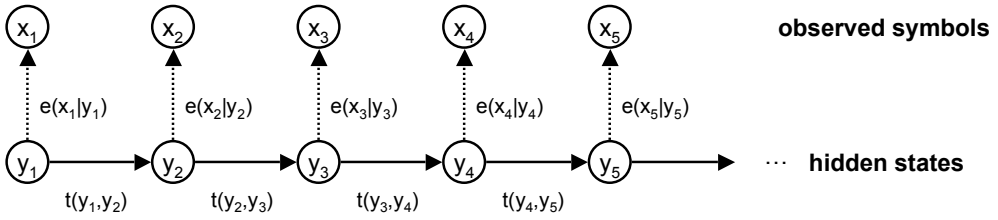


Figure 1.4: Illustration of the doubly embedded stochastic process in HMMs. The stochastic process consists of an observable symbol sequence and a hidden state sequence.

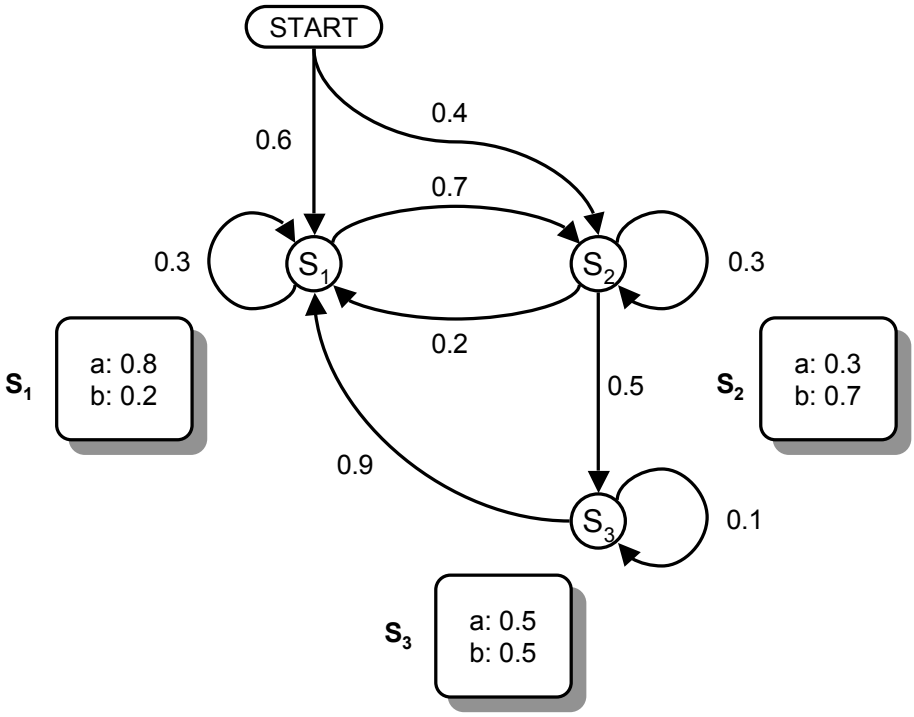


Figure 1.5: Example of a state transition diagram that represents a hidden Markov model with three states $\mathcal{S} = \{S_1, S_2, S_3\}$ and two distinct observation symbols $\mathcal{A} = \{a, b\}$. The initial state distribution and the state transition probabilities are shown along the edges and the emission probabilities are shown in the boxes.

set of parameters $\Theta = \{T, E, \pi\}$

$$
\begin{aligned}
T &= \begin{bmatrix} 0.3 & 0.7 & 0.0 \\ 0.2 & 0.3 & 0.5 \\ 0.9 & 0.0 & 0.1 \end{bmatrix}, \\
E &= \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \\ 0.5 & 0.5 \end{bmatrix}, \\
\pi &= \begin{bmatrix} 0.6 & 0.4 & 0.0 \end{bmatrix}.
\end{aligned}
$$

As an example, let us consider the following observation sequence $\mathbf{x}$ with the underlying state sequence $\mathbf{y}$ as shown below

$$
\begin{aligned}
\mathbf{x} &= a \quad a \quad b \quad b \quad a, \\
\mathbf{y} &= S_1 \quad S_1 \quad S_2 \quad S_3 \quad S_1.
\end{aligned}
$$

The probability $P(\mathbf{x}, \mathbf{y})$ can be computed as

$$
\begin{aligned}
P(\mathbf{x}, \mathbf{y}) &= P(y_1 = S_1) \times e(a|S_1) \times t(S_1, S_1) \times e(a|S_1) \times t(S_1, S_2) \\
&\quad \times e(b|S_2) \times t(S_2, S_3) \times e(b|S_3) \times t(S_3, S_1) \times e(a|S_1) \\
&= 0.6 \times 0.8 \times 0.3 \times 0.8 \times 0.7 \times 0.7 \times 0.5 \times 0.5 \times 0.9 \times 0.8 \\
&= 1.016064 \times 10^{-2}.
\end{aligned}
$$

There are three important problems that have to be solved in order to apply the HMMs to real world applications. These problems are typically called the *alignment problem*, the *scoring problem*, and the *training problem*. These problems are described in the following.

Alignment problem

Given an observed symbol sequence $\mathbf{x} = x_1 x_2 \ldots x_L$ and an HMM defined by the set of parameters Θ, how can we find the optimal state sequence $\mathbf{y} = y_1 y_2 \ldots y_L$ that maximizes the observation probability $P(\mathbf{y}|\mathbf{x}, \Theta)$? This is called the "alignment problem" because it tries to find the best alignment between the symbol sequence $\mathbf{x}$ and the given HMM. As the number of paths increases exponentially with the length L of the symbol string $\mathbf{x}$, comparing all paths is practically infeasible. However, there exists an efficient dynamic programming algorithm called the *Viterbi algorithm*, which can find the optimal state sequence in a systematic

way [116]. The complexity of the Viterbi algorithm is only $O(LM^2)$, which increases linearly with respect to the sequence length L, where M is the number of states.

Scoring problem

Given an observed symbol sequence $\mathbf{x} = x_1 x_2 \ldots x_L$, how can we compute its observation probability based on a given model Θ? As this probability can be used to score different models to choose the one that best describes the observation sequence, it is usually called the "scoring problem." This problem can be efficiently solved using the *forward algorithm* [56, 81], which is closely related to the Viterbi algorithm. The complexity of the forward algorithm is also $O(LM^2)$.

Training problem

Finally, we have to address the problem of how to choose the model parameters in an optimal manner, based on a number of training sequences. One popular solution to this problem is an EM (expectation-maximization) algorithm called the Baum-Welch algorithm [6]. This algorithm can iteratively find the parameters that achieve a local maximum of the observation probability of the training sequences.

The algorithms that can be used for solving these problems for HMMs are described in considerable detail in [56, 81].

HMMs are well known for their effectiveness in modeling short-term dependencies between adjacent symbols. For this reason, they have been extensively used in various fields, including speech recognition [56, 81] and bioinformatics [28, 59]. In Chapter 2, we extend the traditional HMM so that we can also describe long-range correlations between distant symbols. The extended model, called the *context-sensitive HMM (csHMM)* has important applications in RNA sequence analysis, as will be demonstrated in Chapter 3 and Chapter 4.

1.4 Review of some fundamentals in genomics

In this section, we briefly review some basics in genomics that are needed to understand the details of the thesis.

1.4.1 DNA and RNA

DNA (deoxyribonucleic acid) is a nucleic acid that contains the genetic information for cellular life forms and viruses. They are responsible for propagating the hereditary information in all living organisms. A single strand of DNA consists of many nucleotides that are linked to each other forming a long chain. A nucleotide consists of a base, a sugar, and a phosphate (or a phosphate group). The structure of a nucleotide is illustrated in Figure 1.6. The nucleotides that form a DNA strand can have four different kinds of bases, namely, *adenine*, *cytosine*, *guanine*, and *thymine*. For convenience, these nucleotides (or the bases) are typically represented by the four letters A, C, G, and T.

In general, a single strand of DNA forms a *double helix* with another single strand of DNA via hydrogen bonding between the bases. An illustration of a DNA double helix is shown in Figure 1.7. The nucleotide A in one strand is linked to T in the other strand (and vice versa), and the nucleotide C in one strand is connected to G in the other strand (and vice versa). As a result, one strand in a DNA double helix completely determines the nucleotide sequence in other strand, hence they are called *complementary strands*.

Figure 1.8 shows an example of a short DNA double helix that has been straightened out for simplicity. As indicated in the figure, the sugar-phosphate forms the backbone of each DNA strand, and the two strands that run towards opposite directions are linked to each other by the chemical bonds formed between the complementary bases. As the nucleotide sequence in one strand determines the nucleotide sequence in the other strand, a double-stranded DNA can be unambiguously represented by either strand. For this reason, a DNA molecule is represented by the nucleotide sequence of the forward strand, which is read from the so-called 5′-end to the 3′-end. For example, let us consider the DNA shown in Figure 1.8:

$$5' - A - A - T - C - G - G - C - T - A - C - 3' \quad \text{(forward strand)}$$
$$3' - T - T - A - G - C - C - G - A - T - G - 5' \quad \text{(backward strand)} .$$

This can be simply represented by the forward strand

$$5' - A - A - T - C - G - G - C - T - A - C - 3'.$$

Therefore, from a signal processing perspective, we can view a DNA molecule as a symbol se-

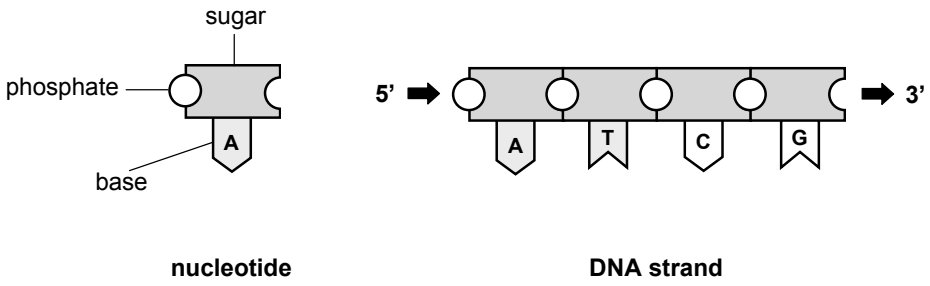


Figure 1.6: Illustration of a nucleotide and a DNA strand.

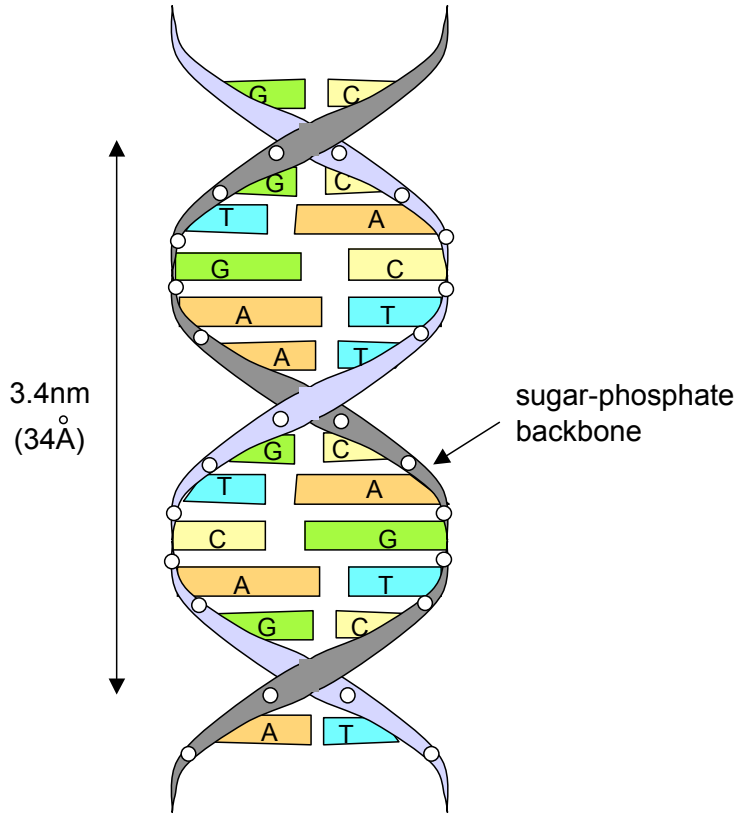


Figure 1.7: Illustration of a DNA double helix.

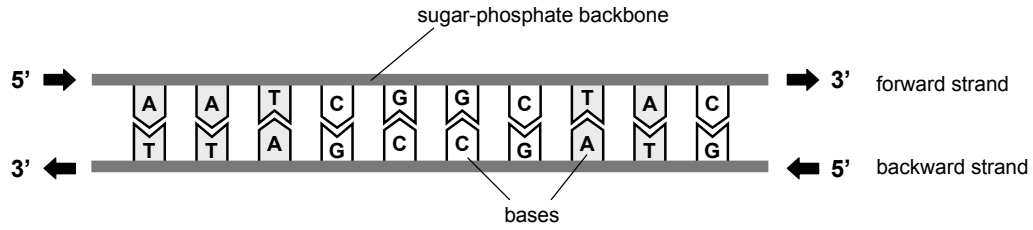


Figure 1.8: Example of a DNA double helix that has been straightened out for simplicity.

quence, where the symbols are taken from a finite alphabet.

The RNA (ribonucleic acid) is another type of nucleic acid that is closely related to the DNA. It also consists of four kinds of bases like the DNA, except that *uracil* (U) is used instead of thymine (T). Unlike DNA molecules, the RNA is typically a single-stranded molecule.

1.4.2 Protein synthesis

A protein is a complex biomolecule that consists of a long chain of amino acids. The amino acids are linked to each other by strong covalent bonding called peptide bonds, and the amino acid chain is also known as a polypeptide. There are 20 different kinds of amino acids in proteins, where each amino acid has a different side-chain. Therefore, a protein can be conveniently represented as a sequence of amino acids, where each of the 20 distinct amino acids is denoted by a 3-letter code or an 1-letter code. For example, the amino acid *alanine* is denoted by 'Ala' or 'A,' and *cysteine* is denoted by 'Cys' or 'C.'

Proteins are involved in every single biological process in all cells, hence playing a crucial role in all living organisms. The information that is needed for encoding proteins is stored in the DNA. Portions in the DNA that contain the information for producing proteins are called *protein-coding genes*, or often simply *genes*.[2] Each gene in the DNA is first copied into an RNA molecule (*transcription*), which is then used to produce proteins (*translation*). Therefore, it can be said that the genetic information flows from DNA to RNA to protein. This basic principle is typically called the *central dogma* of molecular biology [1], and it explains how the genetic instructions contained in the DNA are used to synthesize RNAs and proteins. Figure 1.9 illustrates this principle in a simple diagram. The main steps in a typical protein synthesis process are shown in Figure 1.10. Each step in the process is discussed in the following subsections.

[2]Note that there exist also *ncRNA (noncoding RNA) genes*, which are portions of DNA that give rise to functional RNAs that are not translated into proteins.

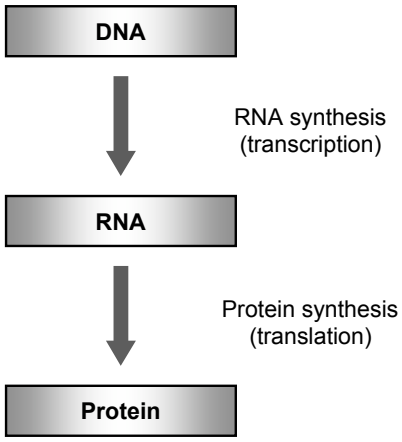


Figure 1.9: The central dogma of molecular biology states that the genetic information flows from DNA to RNA to protein.

1.4.2.1 Transcription

The process of copying the content of a gene into an RNA is called *transcription*. The transcription process is carried out by an enzyme called *RNA polymerase*, where an *enzyme* is a protein that catalyzes a specific chemical reaction. Initially, the RNA polymerase binds to a special region in the DNA called the *promoter*, which is located upstream of a gene and is used to designate the starting point of the transcription process. During transcription, the RNA polymerase uses one strand of the DNA (called the *template strand*) to copy the content into an RNA molecule. While copying the content from DNA to RNA, a thymine (T) in the original DNA sequence is replaced by a uracil (U) in the RNA that is being synthesized. The resulting transcript of a protein-coding gene is called a *pre-mRNA* (pre-messenger RNA).

Living organisms can be categorized into two types, namely, *prokaryotes* and *eukaryotes*. Prokaryotes are simple organisms (mostly unicellular) that do not have a cell nucleus. Bacteria are common examples of prokaryotes. On the other hand, eukaryotes are organisms that have complex cells with membrane-bound nuclei. Most of them are multicellular, and higher organisms such as worms, plants, insects and mammals belong to eukaryotes. Most protein-coding genes in eukaryotes consist of two types of regions called exons and introns (see Figure 1.10).[3] The introns are removed from the pre-mRNA and the remaining exons are concatenated to form a *mRNA* (messenger RNA). This process is called *splicing*. Sometimes, one pre-mRNA gives rise to multiple mRNAs

[3]The protein-coding genes of prokaryotes do not have introns.

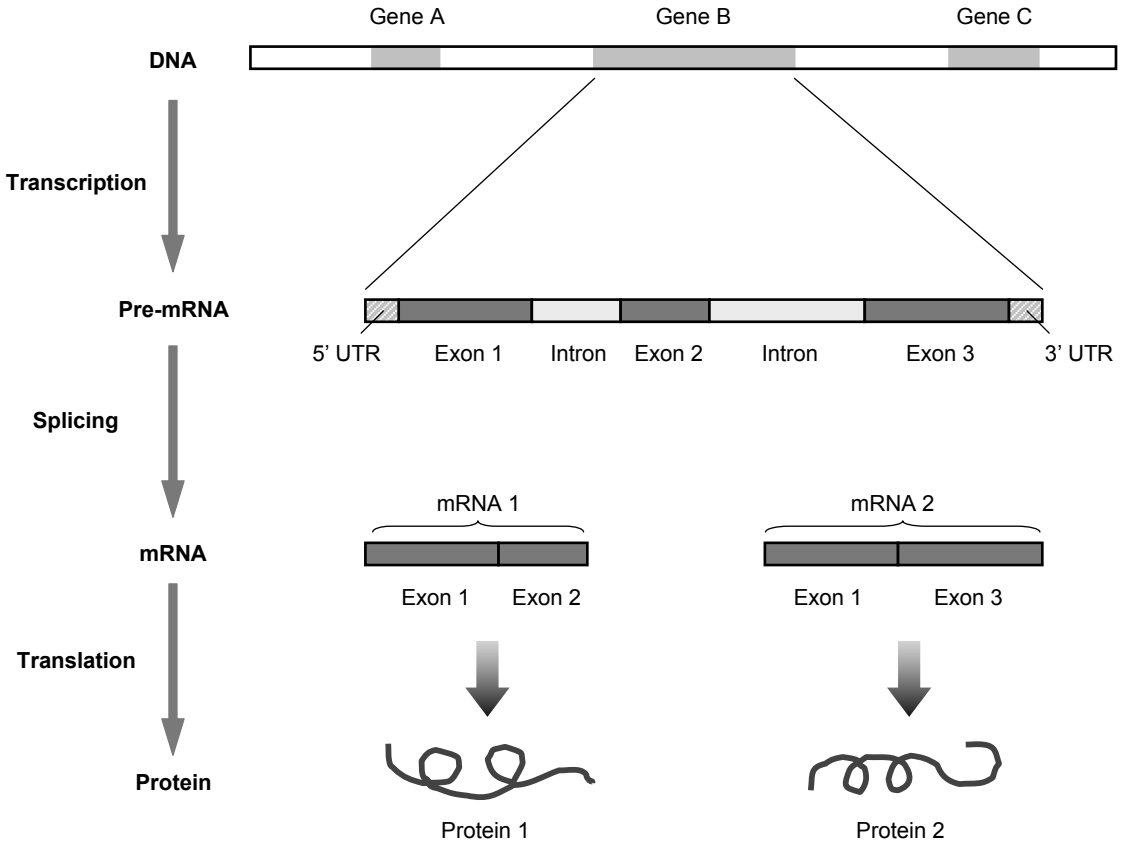


Figure 1.10: Illustration of a typical protein synthesis process.

UUU : Phenylalanine UUC : Phenylalanine UUA : Leucine UUG : Leucine	UCU : Serine UCC : Serine UCA : Serine UCG : Serine	UAU : Tyrosine UAC : Tyrosine UAA : Stop UAG : Stop	UGU : Cysteine UGC : Cysteine UGA : Stop UGG : Tryptophan
CUU : Leucine CUC : Leucine CUA : Leucine CUG : Leucine	CCU : Proline CCC : Proline CCA : Proline CCG : Proline	CAU : Histidine CAC : Histidine CAA : Glutamine CAG : Glutamine	CGU : Arginine CGC : Arginine CGA : Arginine CGG : Arginine
AUU : Isoleucine AUC : Isoleucine AUA : Isoleucine AUG : Methionine, Start	ACU : Threonine ACC : Threonine ACA : Threonine ACG : Threonine	AAU : Asparagine AAC : Asparagine AAA : Lysine AAG : Lysine	AGU : Serine AGC : Serine AGA : Arginine AGG : Arginine
GUU : Valine GUC : Valine GUA : Valine GUG : Valine	GCU : Alanine GCC : Alanine GCA : Alanine GCG : Alanine	GAU : Aspartic acid GAC : Aspartic acid GAA : Glutamic acid GAG : Glutamic acid	GGU : Glycine GGC : Glycine GGA : Glycine GGG : Glycine

Figure 1.11: The genetic code.

by combining different exons. This phenomenon is called *alternative splicing*, and it is widely observed in eukaryotes.

1.4.2.2 Translation

During the *translation* process, the mRNA that was transcribed from DNA is decoded by the ribosome and *tRNAs* (transfer RNA) to generate a polypeptide (or a protein). A polypeptide is a long sequence of amino acids that are interconnected via peptide bonds. The translation of mRNAs into proteins is governed by the *genetic code* that maps each of the 64 *codons* (triplets of nucleotides) into one of the 20 different amino acids. Figure 1.11 shows the genetic code that holds true for most genes in the vast majority of organisms. However, deviations from the standard code shown in Figure 1.11 are also widespread. For example, in several human mitochondrial mRNAs, the triplet 'UGA' was observed to code a tryptophan instead of serving as a stop codon [11].

For a comprehensive introduction to genomics and cell biology, see [1, 11].

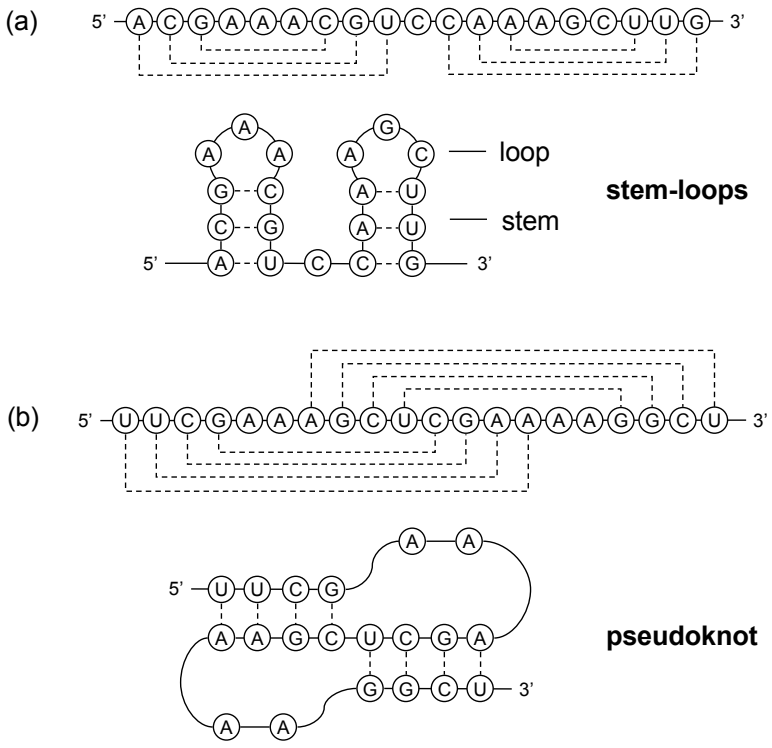


Figure 1.12: Two examples of RNAs with secondary structures. The primary sequence of each RNA is shown along with its structure after folding. The dashed lines indicate interactions between bases. (a) RNA with two stem-loops. (b) RNA with a pseudoknot.

1.5 RNA secondary structure

As mentioned earlier, the RNA is a nucleic acid that consists of a chain of nucleotides. There are four distinct types of nucleotides, A, C, G, and U, where U is chemically similar to T. Just like in the DNA, A and U can form a hydrogen-bonded base pair, and similarly, C and G can also form a pair.[4] In general, the RNA is a single-stranded molecule. If there exist complementary parts in a given RNA, these parts can form contiguous base pairs, making the RNA fold onto itself intramolecularly. This complementary base pairing determines the three-dimensional structure of the RNA to a considerable extent, and the two-dimensional structure resulting from the base pairing is referred as the *RNA secondary structure*. In contrast, the one-dimensional string of nucleotides is sometimes called the *primary sequence* of the RNA.

Figure 1.12 shows two examples of RNA secondary structures. We can see that both RNAs

[4]Sometimes, the bases G and U can also form pairs.

display characteristic secondary structures after folding. As indicated in Figure 1.12 (a), the contiguous base pairs that are stacked onto each other after folding is called a *stem*, and the sequence of unpaired bases bounded by base pairs is called a *loop*. The secondary structure of the RNA in Figure 1.12 (a) consists of two *stem-loops* (or *hairpins*). In many cases, the base pairings occur in a nested manner, where no interactions between bases cross each other. To be more precise, consider a base pair between locations i and j $(i < j)$, and another base pair between locations k and ℓ $(k < \ell)$. We say that these two base pairs are nested if they satisfy $i < k < \ell < j$ or $k < i < j < \ell$. The RNA shown in Figure 1.12 (a) has only nested interactions. Secondary structures with crossing interactions, where there exist base pairs at (i, j) and (k, ℓ) that satisfy $i < k < j < \ell$ or $k < i < \ell < j$, are called *pseudoknots*. One such example is shown in Figure 1.12 (b). Although RNA pseudoknots are observed less frequently than secondary structures with only nested base pairs, there are still many RNAs that are known to contain functionally important pseudoknots [104].

Many interesting RNAs are known to conserve their secondary structures among different species [25]. The conserved secondary structure gives rise to complicated long-range correlations between distant bases in the primary sequence of an RNA. More detailed discussion on this topic will be presented in Chapter 3.

For additional details on RNA secondary structures and RNA sequence analysis, the reader is referred to [25, 30].

1.6 Outline of the thesis

This thesis is organized as follows.

1.6.1 Context-sensitive hidden Markov models (Chapter 2)

In many applications, biological sequences are often treated as unstructured one-dimensional symbol sequences. However, they usually have higher dimensional structures that play important roles in carrying out their biological functions within cells. For example, an RNA molecule often folds onto itself to form a specific RNA secondary structure as shown in Section 1.5. This structure gives rise to complicated long-range correlations between distant bases in the RNA, which cannot be handled by simple models such as the Markov chains and the hidden Markov models (HMMs).

In Chapter 2, we introduce the concept of context-sensitive HMMs (csHMMs), which can be

used for representing symbol sequences with long-range correlations. The csHMM is an extension of the traditional HMM, where the emission probabilities and the transition probabilities at certain states depend on the previous emission, called the "context." This context-sensitive property increases the descriptive power of the model tremendously, making the csHMMs capable of handling complicated correlations between nonadjacent symbols. Due to the increased descriptive power of the model, we cannot use the algorithms that have been utilized for analyzing the traditional HMMs (e.g., the Viterbi algorithm, the forward algorithm, and the Baum-Welch algorithm). In Chapter 2, we propose dynamic programming algorithms that can be used with csHMMs for finding the optimal state sequence (the "alignment problem") and computing the observation probability (the "scoring problem") of a symbol sequence. In addition to this, we also propose a parameter reestimation algorithm that can be used for finding the optimal parameters of a csHMM based on a set of training sequences.

1.6.2 RNA sequence analysis using context-sensitive HMMs (Chapter 3)

The context-sensitive HMMs proposed in Chapter 2 have important applications in RNA sequence analysis. In Chapter 3, we focus on the role of csHMMs in the computational identification and analysis of the so-called noncoding RNAs (ncRNAs). For a long time, it has been believed that proteins are responsible for most of the important biological functions within cells. In the meanwhile, the RNA was mainly viewed as a passive intermediary that interconnects DNA and proteins. However, recent results indicate that ncRNAs, which are RNA molecules that function without being translated into proteins, play pivotal roles in various biological processes, especially in controlling the regulatory mechanisms in the cells [30, 44].

In Chapter 3, we show how the csHMMs can be utilized for building probabilistic representations of ncRNA families and finding new ncRNA genes. We give examples of csHMMs that represent the correlations in symbol sequences that arise from various RNA secondary structures. Then, we propose a dynamic programming algorithm that can be used for searching a large database to find similar sequences that closely match the original RNA that is represented by the csHMM at hand.

Unlike most RNA sequences that adopt a single "biologically correct" structure, there exist many regulatory RNAs that can choose from alternative secondary structures [53, 108]. These RNAs can differentially fold depending on the environmental cues, thereby controlling the ex-

pression level of certain genes. Alternative folding in RNAs introduces some complications, as the resulting correlation structure is significantly more complex than those of typical RNAs that take only one structure. At the end of Chapter 3, we propose a method based on csHMMs, which can be used for modeling and identifying RNAs with alternative secondary structures. The proposed method provides a good prediction performance at a reasonably low complexity, making it practically usable in real applications.

The main emphasis of Chapter 3 lies on building tools that can be used to find new members (or homologues) of known ncRNA families.

1.6.3 Profile context-sensitive hidden Markov models (Chapter 4)

In Chapter 4, we present a subclass of context-sensitive HMMs, called profile-csHMMs, which are especially useful in representing RNA profiles. The Profile-csHMM is a csHMM with a linear structure that repetitively uses three kinds of states, namely, match states, delete states, and insert states. Unlike traditional profile-HMMs, some of the match states are made context sensitive such that we can represent pairwise correlations between the bases that form a complementary base pair in the RNA secondary structure. Profile-csHMMs can be easily constructed from an RNA multiple sequence alignment, in a simple and intuitive manner.

One of the most important advantages of profile-csHMMs is that they are capable of modeling any kind of RNA pseudoknots. For example, models such as CMs (covariance models) [26] that have been extensively used for modeling RNAs, can only describe nested correlations, hence not capable of handling pseudoknots. More recent models such as PSTAGs (pair stochastic tree adjoining grammars) [66] can also deal with many pseudoknots, but not all of them. Based on profile-csHMMs, we propose a dynamic programming algorithm, which is called the sequential component adjoining (SCA) algorithm, that can be used for finding the optimal state sequence of profile-csHMMs.

To demonstrate the effectiveness of the propose model, we build a structural alignment tool that can be used for aligning RNA pseudoknots and predicting their secondary structures. Experimental results indicate that the profile-csHMM based approach can achieve a high prediction accuracy that is comparable to the state-of-the-art method, while it can also deal with a much larger class of pseudoknots. Furthermore, the proposed structural alignment method runs much faster than the previous method (based on PSTAGs) without degrading the accuracy.

Finally, we consider the problem of performing an RNA homology search using profile-csHMMs. Although the profile-csHMM alignment algorithm runs reasonably fast, it is still slow if we want to use it for scanning a large database. At the end of Chapter 4, we propose an efficient pre-filtering scheme for making a profile-csHMM search significantly faster, without affecting its prediction accuracy.

1.6.4 Predicting protein-coding genes using digital filters (Chapter 5)

In Chapter 5, we present digital filtering methods for identifying protein-coding genes. It is well known that protein-coding regions in DNA sequences frequently display period-3 behaviors that are not observed in noncoding regions. Therefore, we can exploit this property for identifying protein-coding regions in a given DNA sequence. Traditionally, these regions have been identified with the help of windowed DFT (discrete Fourier transform) [3, 106]. From a digital filtering perspective, we can view the DFT approach as digital filtering using a bandpass filter whose passband is centered at $2\pi/3$. In this way, we can extract the period-3 component to measure the strength of the periodic behavior in a specific region. However, the DFT-based filter does not have a high stopband attenuation, which leaves a considerable amount of undesirable noise after filtering.

We can overcome this problem by designing a better digital filter with a higher stopband attenuation. In Chapter 5, we propose two different methods for designing digital filters that can be used for identifying protein-coding genes. The first method is based on allpass-based antinotch filters and the second method is based on multistage digital filtering. Experimental results indicate that both methods can isolate the period-3 components from the noisy background considerably better than the traditional DFT approach. Furthermore, the digital filters that are used in the proposed methods can be very efficiently designed, providing a significant advantage in terms of computational cost.

The gene identification problem is quite complex in nature, and we need more powerful statistical models to achieve a high prediction accuracy. However, such models are computationally more expensive than the proposed digital filtering methods, which makes the speed of the gene finder quite slow. Therefore, we can use the prediction methods proposed in Chapter 5 for a fast prescreening of the genome and use a more descriptive model (such as an HMM) in the second stage in order to expedite the overall gene identification process.

1.6.5 Identification of CpG islands using filter banks (Chapter 6)

The CpG islands are specific regions in DNA molecules that are abundant in the dinucleotide CpG. They are usually located upstream of the transcription start regions of many genes, hence can be used as good gene markers. Furthermore, the methylation of CpG islands is known to play an important role in gene silencing, genomic imprinting, carcinogenesis, and so forth. For this reason, the computational identification of CpG islands has been of interest to many researchers.

In Chapter 6, we propose a method for finding CpG islands based on a filter bank that consists of IIR (infinite impulse response) lowpass filters. The proposed method models the CpG island region and the non-CpG island region respectively, using two different Markov chains. Based on the two Markov chains, it computes the log-likelihood ratio for every transition between the adjacent bases in a given DNA sequence. This log-likelihood ratio is filtered using a bank of lowpass filters, whose output signals are then analyzed to find the transition points between the CpG island regions and the non-CpG island regions. The filter bank based prediction approach provides a convenient way for obtaining reliable prediction results without degrading the resolution of the start/end positions of the predicted CpG islands.

Chapter 2

Context-Sensitive Hidden Markov Models

Although biological sequences are often treated as unstructured one-dimensional symbol sequences for simplicity, they usually have three-dimensional structures that play important roles in carrying out their biological functions in cells. For example, a polypeptide (a long chain of amino acids) is biologically inactive until it folds into a correct three-dimensional protein structure. This is typically called *protein folding* [11]. RNAs, which usually exist as single-stranded molecules, often fold onto themselves intramolecularly to form consecutive base-pairs. The three-dimensional structure of an RNA is determined by this complementary base-pairing to a considerable extent, and the two dimensional structure that results from this base-pairing is referred as the *RNA secondary structure* [25].[1] Due to these structures, many biological sequences–such as proteins and noncoding RNAs (ncRNAs)– exhibit complicated correlations between nonadjacent symbols [25]. Such correlations cannot be effectively handled by simple models such as *Markov chains* and *hidden Markov models (HMMs)*. In fact, these models belong to *stochastic regular grammars (SRGs)* according to the so-called *Chomksy hierarchy of transformational grammars*, which cannot model symmetric sequences (or *palindromes*). As RNAs with conserved secondary structures can be viewed as "biological palindromes," these models are incapable of handling RNA sequences. This will be described in more detail in Section 2.2.

In order to overcome this limitation, we propose a new statistical model in this chapter, which is called the *context-sensitive hidden Markov model (csHMM)*. The csHMM is an extension of the conventional HMM, where the probabilities at some states are made context sensitive. This increases the descriptive capability of the model tremendously, making csHMMs capable of describ-

[1] Examples of RNA secondary structures can be found in Chapter 1 and Chapter 3.

ing long-range correlations between nonadjacent symbols.[2] The proposed model has several advantages over other existing models, including the stochastic context-free grammars (SCFG), as will be demonstrated later. The csHMMs are especially useful for modeling ncRNAs with conserved secondary structures and for building RNA sequence analysis tools.

The content of this chapter is mainly drawn from [131], and portions of it have been presented in [123, 127, 128].

2.1 Outline

The organization of this chapter is as follows. In Section 2.2, we briefly review the Chomsky hierarchy of transformational grammars, and explain where the conventional HMMs are located in this hierarchy. We show that the descriptive capability of HMMs is limited to sequences with sequential dependencies and give examples that cannot be effectively modeled by the HMMs.

In Section 2.3, we introduce the concept of context-sensitive HMM (csHMM), which is an extension of the HMM that can be used for representing long-range correlations between distant symbols. We first elaborate on the basic elements of a csHMM in Section 2.3.1, and explain in Section 2.3.2 how these elements can be used to build an actual csHMM.

In Section 2.4, we consider the "alignment problem" of csHMMs. It is shown that the Viterbi algorithm cannot be used for finding the optimal state sequence of a csHMM due to the context-sensitive property of the model. In Section 2.4.1, we briefly describe how we can implement a dynamic programming algorithm that can be used for finding the optimal state sequence that maximizes the probability of an observation sequence, based on a given csHMM. The algorithmic details are described in Section 2.4.2 and Section 2.4.3, and the overall complexity of the algorithm is analyzed in Section 2.4.4.

The "scoring problem" of csHMMs is considered in Section 2.5. In this section, we show how we can compute the observation probability of a symbol sequence in a systematic way. The details of the scoring algorithm is described in Section 2.5.1. In addition to this, we describe the outside algorithm for csHMMs in Section 2.5.2, which can be used along with the scoring algorithm for estimating the model parameters of csHMMs.

In Section 2.6, we describe a parameter re-estimation algorithm for csHMMs. We can iteratively

[2] It has to be noted that the *context-sensitive HMMs* proposed in this chapter are not related to the so-called *context-dependent HMMs* that have been widely used in speech recognition [62, 64, 97]. They are regular HMMs, whose basic building blocks are built by considering the phonetic context, hence called context-dependent HMMs.

apply the proposed algorithm to train a csHMM based on a set of training sequences. Experimental results are given in Section 2.7, which demonstrate the effectiveness of the algorithms proposed in Section 2.4, Section 2.5, and Section 2.6.

In Section 2.8, we discuss several interesting issues related to csHMMs. For example, we consider extending the proposed model for representing non-pairwise correlations in Section 2.8.1 and explain how csHMMs can be used for modeling crossing correlations in Section 2.8.2. In Section 2.8.3, we compare the proposed model with other variants of HMMs. The csHMM is also compared to other transformational grammars (e.g., context-free grammars, context-sensitive grammars) in Section 2.8.4, and we show that csHMMs have several advantages over these grammars. Concluding remarks are given in Section 2.9.

Finally, in Appendix A, we give an example of a context-free grammar (CFG) that cannot be represented by a csHMM. As there also csHMMs that cannot be represented by CFGs (shown in Section 2.8.2), this demonstrates that neither the csHMMs nor the CFGs fully contain the other (see Figure 2.19). In Appendix B, we describe simplified versions of the alignment algorithm and the scoring algorithm that are proposed in Section 2.4 and Section 2.5, respectively. The simplified algorithms can be used for analyzing sequences with *single nested correlations*, and they are computationally more efficient compared to the original algorithms.

2.2 HMMs and transformational grammars

Hidden Markov models (HMMs) have been widely used in many fields. They are well known for their efficiency in modeling short-term dependencies between adjacent symbols, which made them popular in diverse areas. Traditionally, HMMs have been successfully applied to speech recognition, and many speech recognition systems are built upon HMMs and their variants [56, 81]. They have been also widely used in digital communications, and more recently, HMMs have become very popular in computational biology as well. They have been proved to be useful in various problems such as gene identification [25, 58, 96], multiple sequence alignment [25, 27], and so forth. Due to its effectiveness in modeling symbol sequences, the HMM gave rise to a number of useful variants that extend and generalize the basic model [35, 49, 50, 70, 79, 80, 136].

Although HMMs have a number of advantages, the basic HMM and its variants have also inherent limitations. For example, they are capable of modeling sequences with strong correlations between adjacent symbols, but they cannot describe long-range interactions between symbols that

Type	Allowed production rules
Regular Grammar	$A \longrightarrow aB \mid a \mid \epsilon$
Context-Free Grammar	$A \longrightarrow \alpha$
Context-Sensitive Grammar	$\alpha A \gamma \longrightarrow \alpha \beta \gamma$ or $S \longrightarrow \epsilon$
Unrestricted Grammar	$\alpha A \gamma \longrightarrow \delta$

Table 2.1: The Chomsky hierarchy of transformational grammars.

are distant from each other. Therefore, the resulting model always displays local dependencies,[3] and more complex sequences with non-sequential dependencies cannot be effectively represented using the conventional HMMs.

2.2.1 Transformational grammars

In computational linguistics, a *transformational grammar* is defined as a set of rules that can be used to describe (or generate) a set of symbol sequences over a given alphabet. It was first formally proposed by the computational linguist Noam Chomsky [16]. A transformational grammar can be characterized by the following components: *terminal symbols*, *nonterminal symbols*, and *production rules*. Terminal symbols are the observable symbols that actually appear in the final symbol sequence, and nonterminal symbols are abstract symbols that are used to define the production rules. A production rule is defined as $\alpha \rightarrow \beta$, where α and β are strings of terminal and/or nonterminal symbols. It describes how a given string can be transformed into another string. We can generate various symbol sequences by applying these production rules repetitively, where the generation process starts from the start nonterminal S and terminates when there are no more nonterminals.

As an example, let us consider the following grammar which has a single nonterminal $\{S\}$ and two terminals $\{a, b\}$:

$$S \longrightarrow aS,\ S \longrightarrow b.$$

This simple grammar can generate any sequence of the form $a \ldots ab$. For example, we can generate the sequence $aaab$ by applying the above rules as follows

$$S \longrightarrow aS \longrightarrow aaS \longrightarrow aaaS \longrightarrow aaab.$$

In his work on transformational grammars, Chomsky categorized transformational grammars

[3]By *local dependencies*, we imply that the probability that a symbol appears at a certain location depends only on its immediate preceding neighbors.

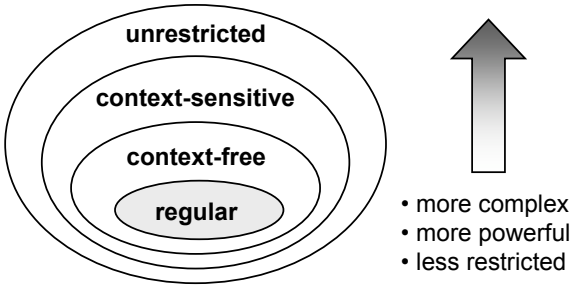


Figure 2.1: The Chomsky hierarchy of transformational grammars nested according to the restrictions on the allowed production rules.

into four classes. These are the *regular grammars*, *context-free grammars*, *context-sensitive grammars* and *unrestricted grammars*, in the order of decreasing restrictions on the production rules. The production rules allowed in each class are summarized in the Table 2.1. A and B are single nonterminals, a is a single terminal, and ϵ is the empty sequence. α, γ, δ are any string of terminals and/or non-terminals, and β is any nonempty string of terminals and/or non-terminals. (The notation '|' means 'or.') These four classes comprise the so-called *Chomsky hierarchy of transformational grammars*, which is illustrated in Figure 2.1. As can be seen from the diagram, regular grammars are the simplest among the four, and they have the most restricted production rules.

HMMs can be viewed as stochastic regular grammars (SRG), according to this hierarchy. Due to the restrictions on their production rules, regular grammars have efficient algorithms such as the *Viterbi algorithm* [116] for finding the optimal state sequence (popularly used in digital communication receivers), the *forward algorithm* [56, 81] for computing the probability of an observed symbol string, and the *Baum-Welch algorithm* [6] for re-estimation of the model parameters. Other transformational grammars that belong to a higher-order class in the hierarchy have less restrictions on the allowed production rules, and therefore they have greater descriptive power to represent more complicated dependencies between symbols. However, the computational complexity for analyzing an observation sequence (e.g., computing the observation probability, finding the optimal state sequence) increases very quickly, which makes the use of higher-order grammars sometimes impractical.

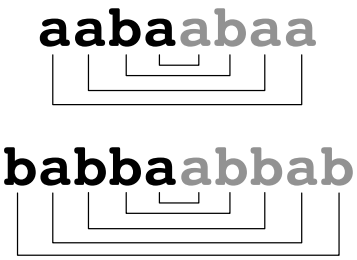


Figure 2.2: Examples of sequences that are included in the palindrome language. The lines indicate the pairwise correlations between distant symbols.

2.2.2 Palindrome language

One interesting language that cannot be represented using regular grammars (or equivalently, using HMMs) is the *palindrome language* [16]. The palindrome language is a language that contains all strings that read the same forwards and backwards. For example, if we consider a palindrome language that uses an alphabet of two letters $\{a, b\}$ for *terminal symbols*, it contains all symbol sequences of the form aa, bb, $abba$, $aabbaa$, $abaaba$, and so on. Figure 2.2 shows examples of symbol strings that are included in this language. The lines in Figure 2.2 that connect two symbols indicate the pairwise correlations between symbols that are distant from each other. Similarly, RNAs with conserved secondary structures display long-range correlations between nonadjacent bases, due to the existence of symmetric (or *reverse complementary*, to be more precise) portions in their primary sequences. This kind of long-range interactions between symbols cannot be described using regular grammars.

It is of course possible that a regular grammar generates such palindromes as part of its language. However, we cannot force the model to generate *only such palindromes*. Therefore regular grammars are not able to effectively discriminate palindromic sequences from non-palindromic ones. In fact, in order to describe a palindrome language, we have to use higher-order grammars such as the context-free grammars. Context-free grammars are capable of modeling nested dependencies between symbols that are shown in Figure 2.2.

In the following section, we describe how the conventional HMMs can be extended such that they can represent long-range symbol correlations, including those observed in palindromes.

2.3 Context-sensitive hidden Markov models

The context-sensitive HMM can be viewed as an extension of the traditional HMM, where some of the states are equipped with auxiliary memory [123, 131]. Symbols that are emitted at certain states are stored in the memory, and the stored data serves as the *context* that affects the emission probabilities and the transition probabilities at certain future states. This context-sensitive property increases the descriptive power of the model significantly, compared to the traditional HMM. Let us first formally define the basic elements of a context-sensitive HMM.

2.3.1 Basic elements of a csHMM

Similar to the traditional HMMs, the csHMM is also a *doubly-stochastic process*, which consists of a non-observable process of hidden states and a process of observable symbols. The process of the hidden states is governed by state-transition probabilities that are associated with the model, and the observation process is linked to the hidden process via emission probabilities of the observed symbol that is conditioned on the hidden state. A csHMM can be characterized by the following elements.

2.3.1.1 Hidden states

We assume that the csHMM has M distinct states. The set of hidden states $\mathcal{V}$ is defined as

$$\mathcal{V} = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C} \cup \{\text{start},\ \text{end}\}, \tag{2.1}$$

where {start, end} is the set of special states that are used to denote the *start* state and the *end* state of the model. As can be seen in (2.1), there are three different classes of states, namely, *single-emission states* S_n, *pairwise-emission states* P_n, and *context-sensitive states* C_n. $\mathcal{S}$ is the set of single-emission states

$$\mathcal{S} = \{S_1, S_2, \ldots, S_{M_2}\}, \tag{2.2}$$

where M_2 is the number of single-emission states in the model. Similarly, $\mathcal{P}$ and $\mathcal{C}$ denote the set of pairwise-emission states and the set of context-sensitive states

$$\mathcal{P} = \{P_1, P_2, \ldots, P_{M_1}\}, \quad \mathcal{C} = \{C_1, C_2, \ldots, C_{M_1}\}. \tag{2.3}$$

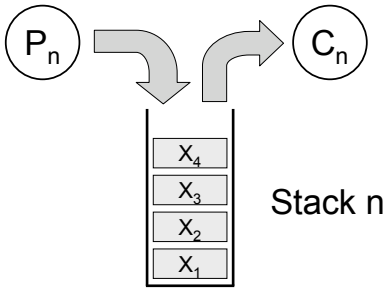


Figure 2.3: The states P_n and C_n associated with a stack Z_n.

As shown in (2.3), the number of pairwise-emission states is the same as the number of context-sensitive states. Therefore, we have $M = 2M_1 + M_2 + 2$ hidden states in total. The states P_n and C_n always exist in pairs. For example, if there are two pairwise-emission states P_1 and P_2 in the model, then the HMM is required to have also two context-sensitive states C_1 and C_2. The two states P_n and C_n are associated with a separate memory element Z_n, such as a *stack* or a *queue*. We may also use other memory types depending on the type of correlations that we want to model. Figure 2.3 shows an example where P_n and C_n are associated with a stack Z_n.

Differences between the three classes of states. The differences between the three classes of states are as follows.

(1) Single-emission state $\mathbf{S_n}$. The single-emission state S_n is similar to a regular hidden state in traditional HMMs. As we enter the state, S_n emits an observable symbol according to the associated emission probabilities. After the emission, S_n makes a transition to the next state according to the specified transition probabilities.

(2) Pairwise-emission state $\mathbf{P_n}$. The pairwise-emission state P_n is almost identical to the single-emission state S_n, except that the symbols emitted at P_n are stored in the auxiliary memory Z_n dedicated to P_n and C_n. The data stored in the memory affects the emission probabilities and the transition probabilities of C_n in the future. After storing the emitted symbol in the memory, a transition is made to the next state according to the transition probabilities of P_n.

(3) Context-sensitive state $\mathbf{C_n}$. The context-sensitive state C_n is considerably different from the other states, in the sense that its emission probabilities and the transition probabilities are not fixed. In fact, these probabilities depend on the *context*, or the data stored in the associated

memory Z_n, which is the reason why C_n is called a context-sensitive state. When entering C_n, it first accesses the memory Z_n and retrieves a symbol x. Once the symbol is retrieved, the emission probabilities of C_n are adjusted according to the value of x. For example, we may adjust the emission probabilities of C_n such that it emits the same symbol x with high probability (possibly, with probability one). Transition probabilities at C_n also depend on the context, as will be explained later.

We denote the hidden state process as $\mathbf{s} = s_1 s_2 \dots s_L$, where s_i is the state at time i and L is the length of the entire sequence. Each state takes a value from $s_i \in \mathcal{V} - \{\text{start}, \text{end}\}$. The virtual start state s_0 and the end state s_{L+1} are assumed to be $s_0 = \textit{start}$ and $s_{L+1} = \textit{end}$.

2.3.1.2 Observation symbols

We denote the observation process as $\mathbf{x} = x_1 x_2 \dots x_L$, where x_i is the observed symbol at time i. Each symbol x_i takes a value from an alphabet $x_i \in \mathcal{A}$. Note that the virtual start state s_0 and the end state s_{L+1} do not make any emission.

2.3.1.3 Transition probabilities

Let us define the probability that the model will make a transition from a state $s_i = v$ to the next state $s_{i+1} = w$. For $v \in \mathcal{S} \cup \mathcal{P}$, we define the probability as

$$P(s_{i+1} = w | s_i = v) = t(v, w).$$

Note that the transition probabilities are stationary and do not depend on the time index i. As mentioned earlier, the transition probabilities at a context-sensitive state C_n depend on the context Z_n. C_n uses two different sets of transition probabilities, depending on whether the associated memory Z_n is empty or not. For each context-sensitive state C_n, we define the following sets

$$\begin{aligned} \mathcal{E}_n &= \{ \text{ Subset of } \mathcal{V} \text{ that contains the states to which } C_n \text{ can make transitions} \\ &\qquad \text{when the associated memory element } Z_n \text{ is empty } \}, \end{aligned} \tag{2.4}$$

$$\begin{aligned} \mathcal{F}_n &= \{ \text{ Subset of } \mathcal{V} \text{ that contains the states to which } C_n \text{ can make transitions} \\ &\qquad \text{when the associated memory element } Z_n \text{ is not empty } \}, \end{aligned} \tag{2.5}$$

where $\mathcal{E}_n \cap \mathcal{F}_n = \varnothing$. At a context-sensitive state C_n, the memory is examined after making the emission. If the memory is empty, $v = C_n$ can make a transition only to $w \in \mathcal{E}_n$. Similarly, if the memory is not empty, $v = C_n$ makes a transition to $w \in \mathcal{F}_n$. Based on this setting, we define the two sets of transition probabilities when $v \in \mathcal{C}$ as follows

$$P(s_{i+1} = w | s_i = v, Z_n) = \begin{cases} t_e(v, w) & \text{if } Z_n \text{ is empty,} \\ t_f(v, w) & \text{if } Z_n \text{ is not empty.} \end{cases}$$

Since $\mathcal{E}_n \cap \mathcal{F}_n = \varnothing$, the probabilities $t_e(v, w)$ and $t_f(v, w)$ cannot have non-zero values at the same time. Therefore, we can let $t(v, w) = t_e(v, w) + t_f(v, w)$ without any ambiguity. Now, the transition probability from $s_i = v \in \mathcal{C}$ to $s_{i+1} = w$ can be simplified as

$$P(s_{i+1} = w | s_i = v, Z_n) = t(v, w).$$

Note that we have $\sum_{w \in \mathcal{E}_n} t(v, w) = 1$ and $\sum_{w \in \mathcal{F}_n} t(v, w) = 1$ in this case. The probability $t(\textit{start}, v)$ is used to define the initial state distribution $P(s_1 = v)$, and $t(w, \textit{end})$ denotes the probability that the HMM will terminate after the state w.

Preventing degeneracies. The restrictions on the types of states to which a context-sensitive state $v \in \mathcal{C}$ is allowed to make transitions depending on the context, can be conveniently used to maintain the number of P_n and that of C_n identical in a state sequence. In this way, we can prevent degenerate situations due to a mismatch between the two states. Let $\mathbf{s} = s_1 s_2 \dots s_L$ be a *feasible* state sequence of an observed symbol string $\mathbf{x} = x_1 x_2 \dots x_L$. The csHMM should be constructed such that the number of occurrences of P_n in the sequence $\mathbf{s}$ is kept the same as the number of occurrences of C_n in $\mathbf{s}$. This restriction is reasonable for the following reasons. In the first place, if there are more C_n states than there are P_n states, the emission probabilities of the context-sensitive state C_n cannot be properly determined. On the other hand, if there are more P_n states than C_n states, the symbols that were emitted at the "surplus" P_n states do not affect the probabilities in the model at all, hence they may be simply replaced by single-emission states.

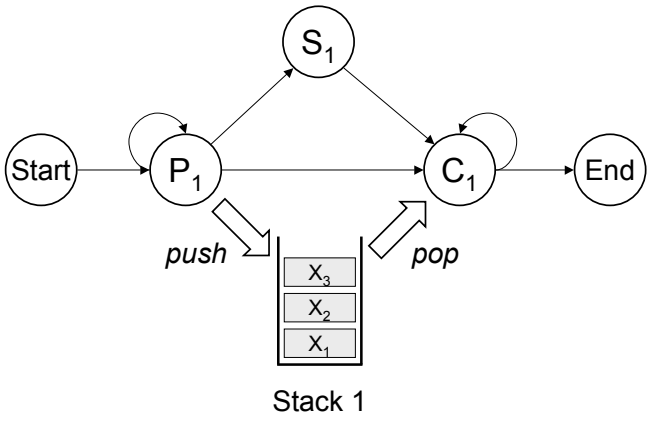


Figure 2.4: An example of a context-sensitive HMM that generates only palindromes.

2.3.1.4 Emission probabilities

The probability of observing a symbol $x_i = x$ depends on the underlying hidden state $s_i = v$. For $v \in \mathcal{S} \cup \mathcal{P}$, this emission probability can be defined as

$$P(x_i = x | s_i = v) = e(x|v).$$

For $v \in \mathcal{C}$, the emission probability depends on both $s_i = v$ and the context Z_n, hence it is defined as

$$P(x_i = x | s_i = v, Z_n) = e(x|v, Z_n). \tag{2.6}$$

In case the emission probability depends only on a single symbol x_p in the memory Z_n (e.g., if Z_n uses a stack, x_p may be the symbol on the top of the stack), the emission probability in (2.6) can be simply written as $e(x|v, x_p)$.

2.3.2 Constructing a csHMM

By using the proposed context-sensitive HMM, we can easily construct a simple model that generates *only* palindromes. For example, we may use the structure shown in Figure 2.4 for this purpose. As can bee seen in Figure 2.4, there are three hidden states S_1, P_1, and C_1 in the model, where the state-pair (P_1, C_1) is associated with a stack. Initially, the model begins at the pairwise-emission state P_1. It makes several self-transitions to generate a number of symbols, which are pushed onto the stack. At some point, it makes a transition to the context-sensitive state C_1. Once we enter the

context-sensitive state C_1, the emission probabilities and the transition probabilities of C_1 are adjusted, such that the state always emits the symbol on the top of the stack and makes self-transitions until the stack becomes empty. In this way, C_1 emits the same symbols as were emitted by P_1, but in the reverse order, since the stack is a last-in-first-out (LIFO) system. If we denote the number of symbols that were emitted by P_1 as N, the generated string will always be a palindrome of the form $x_1 \dots x_N x_N \dots x_1$ (even length sequence) or $x_1 \dots x_N x_{N+1} x_N \dots x_1$ (odd length sequence).

In the following discussions, we mainly focus on those context-sensitive HMMs that generate sequences with *nested interactions*. These models include the ones that generate palindromic sequences as illustrated in Figure 2.4. As in Figure 2.4, we assume that every state-pair (P_n, C_n) is associated with a stack. Based on these settings, we describe efficient dynamic programming algorithms that can be used for analyzing symbol sequences, and we also introduce a training algorithm that can be used for estimating the model parameters of a given csHMM.

2.4 Finding the most probable path

Let us consider an observation sequence $\mathbf{x} = x_1 x_2 \dots x_L$. As described in Section 2.3, we denote the underlying state of x_i as s_i. Assuming that there are M distinct states in the model, we have M^L different paths. Given the observation sequence $\mathbf{x}$, how can we find the path that is most probable among the M^L distinct paths? This problem is tradictionally called the *optimal alignment problem*, since we are trying to find the best alignment between the observed symbol string and the given HMM.

One way to find the most probable path would be to compute the probabilities of all paths, and pick the one with the highest probability. However, this approach is impractical, since the number of paths increases exponentially with the length L of the sequence. When using traditional HMMs, this problem can be solved very efficiently by the Viterbi algorithm [116], which is widely used in digital communication receivers. The Viterbi algorithm exploits the fact that if $s_1 \dots s_{i-1} s_i$ is the optimal path for $x_1 \dots x_{i-1} x_i$ among all paths that end with the state s_i, then $s_1 \dots s_{i-1}$ must be the optimal path for $x_1 \dots x_{i-1}$ among all paths that end with the state s_{i-1}. Therefore, in order to find the optimal path for $x_1 \dots x_i$ with $s_i = v$, we only have to consider the M optimal paths for $x_1 \dots x_{i-1}$ that end with $s_{i-1} = 1, \dots, M$, the transition probability from each of these states to the state $s_i = v$, and the probability of emitting the symbol x_i at the state s_i. This makes the computational complexity of the Viterbi algorithm only $O(LM^2)$, which is considerably better than

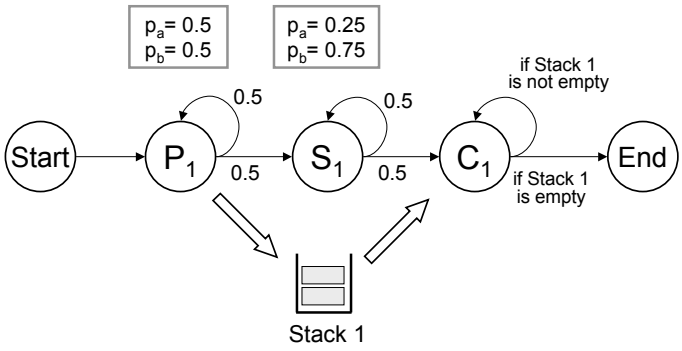


Figure 2.5: An example of a simple context-sensitive HMM.

$O(LM^L)$ of the exhaustive search.

Unfortunately, the same intuition does not hold for context-sensitive HMMs. Since the emission probabilities and the transition probabilities of context-sensitive states C_n depend on the previously emitted symbols at the pairwise-emission states P_n, we have to keep track of the previous states in order to compute the probability of a certain path. Therefore, the optimal path for $x_1 \dots x_i$ cannot be found simply by considering the optimal paths for $x_1 \dots x_{i-1}$ and extending it.

In order to see this, let us consider the example in Figure 2.5. This context-sensitive HMM has three hidden states P_1, C_1, and S_1, where each of these states emits a symbol in the alphabet $\mathcal{A} = \{a, b\}$. The emission probabilities and the transition probabilities of P_1 and S_1 are shown in the figure. The symbols emitted at P_1 are pushed onto the stack, and this data affects the probabilities at the state C_1. Once we enter the context-sensitive state C_1, a symbol is popped out from the stack and is emitted. After the emission, the stack is examined to check whether it is empty. If it is empty, the model terminates. Otherwise, the model makes a transition back to C_1 and continues emitting the symbols that are stored in the stack. Now, let us consider the symbol sequence $abbba$. Assuming that this string comes from the model in Figure 2.5, what is the most probable path $\mathbf{s}^*$? It is not difficult to see that there are only two feasible paths: $\mathbf{s}_1 = P_1S_1S_1S_1C_1$ and $\mathbf{s}_2 = P_1P_1S_1C_1C_1$. Since both paths pass the state S_1 in the middle, let us first consider the optimal path for the first three symbols abb. We denote the subpaths of $\mathbf{s}_1$ and $\mathbf{s}_2$ up to the third symbol as $\hat{\mathbf{s}}_1 = P_1S_1S_1$ and $\hat{\mathbf{s}}_2 = P_1P_1S_1$, respectively. If we compute the probabilities of $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$, we get

$$P(\hat{\mathbf{s}}_1) = \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} = \frac{9}{128},$$

and

$$P(\hat{\mathbf{s}}_2) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} = \frac{6}{128},$$

hence the optimal path for the first three symbols abb is $\hat{\mathbf{s}}_1$. However, if we compute the probabilities of the two paths $\mathbf{s}_1$ and $\mathbf{s}_2$, we obtain

$$P(\mathbf{s}_1) = \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times 1 \times 1 = \frac{27}{2048},$$

and

$$P(\mathbf{s}_2) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times 1 \times 1 \times 1 \times 1 \ = \frac{48}{2048},$$

which shows that the optimal path for $abbba$ is $\mathbf{s}_2$. Apparently, the globally optimal path $\mathbf{s}^* = \mathbf{s}_2$ is not an extension of $\hat{\mathbf{s}}_1$, and this example clearly demonstrates that the Viterbi algorithm cannot be used for finding the most probable path in context-sensitive HMMs.

2.4.1 Alignment of csHMM

Although the Viterbi algorithm cannot be used for finding the optimal path in a context-sensitive HMM, we can develop a polynomial-time algorithm that solves the alignment problem in a recursive manner, similar to the Viterbi algorithm. The proposed algorithm is conceptually similar to the *Cocke-Younger-Kasami (CYK) algorithm* [51, 60] that can be used for parsing SCFGs. The main reason why the Viterbi algorithm cannot be used in context-sensitive HMMs is because the interactions between symbols are not sequential. Since the Viterbi algorithm basically considers only sequential depdendencies, it cannot take care of nested interactions between distant symbols. However, if we implement an algorithm that starts from the inside of the given sequence and proceeds to the outward direction by taking the nested interactions into account, it is possible to find the optimal state sequence in a recursive manner.

When searching for the most probable state sequence, we assume that all pairwise interactions between P_n and C_n are nested and they do not cross each other, as mentioned earlier. Figure 2.6 illustrates several examples of interactions that are allowed as well as those that are prohibited. The nodes in the figure denote the observed symbols in the sequence, and the dotted lines that connect two symbols indicate the pairwise interactions between them. Figures 2.6 (a)–(c) show sequences with nested dependencies. On the other hand, the example in Figure 2.6 (d) shows a sequence with a crossing interaction, which is not considered in this chapter.

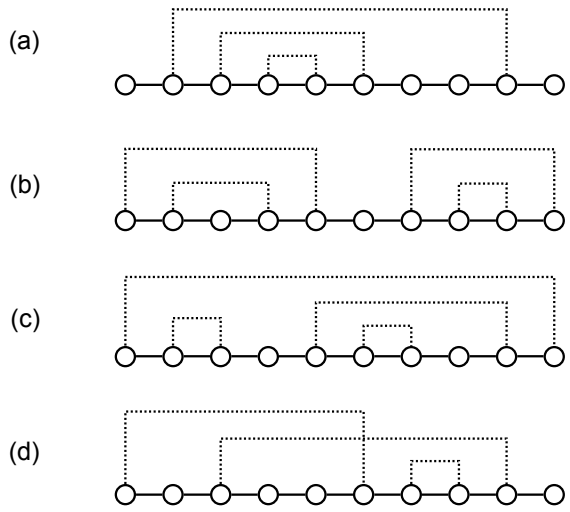


Figure 2.6: Examples of interactions in a symbol string. The dotted lines indicate the pairwise dependencies between symbols. (a), (b), (c) Nested interactions. (d) Crossing interactions.

Before describing the algorithm, let us first define the variables that are needed in the proposed algorithm. $\mathbf{x} = x_1 \dots x_L$ is the observation sequence and $\mathbf{s} = s_1 \dots s_L$ is the underlying state sequence. We assume that the csHMM has M distinct states, which we simply denote by $\mathcal{V} = \{1, 2, \dots, M\}$. The state $v = 1$ denotes the *start* state of the HMM and $v = M$ denotes the *end* state. For $v \in \mathcal{P} \cup \mathcal{C}$, we define $\bar{v}$ as the complementary state of v as follows,

$$v = P_n \rightarrow \bar{v} = C_n, \qquad v = C_n \rightarrow \bar{v} = P_n.$$

The emission probability of a symbol x at a state v is defined as $e(x|v)$ for $v \in \mathcal{S} \cup \mathcal{P}$, and $e(x|v, x_p)$ for $v \in \mathcal{C}$, where x_p is the symbol that was previously emitted at the corresponding pairwise-emission state $\bar{v}$. The transition probability from v to w is defined as $t(v, w)$. Finally, let us define $\gamma(i, j, v, w)$ to be the log-probability of the optimal path among all subpaths $s_i \dots s_j$ with $s_i = v$ and $s_j = w$. In computing $\gamma(i, j, v, w)$, we consider only those paths where all the pairwise-emission states P_n in the $s_i \dots s_j$ are paired with the corresponding context-sensitive states C_n. Examples of subpaths that are considered in computing $\gamma(i, j, v, w)$ are shown in Figure 2.7 (a). The paths shown in Figure 2.7 (b) are not considered due to unpaired P_n or C_n states, or due to crossing interactions. The variable $\gamma(i, j, v, w)$ will ultimately lead to the probability $\log P(\mathbf{x}, \mathbf{s}^*|\Theta)$, where

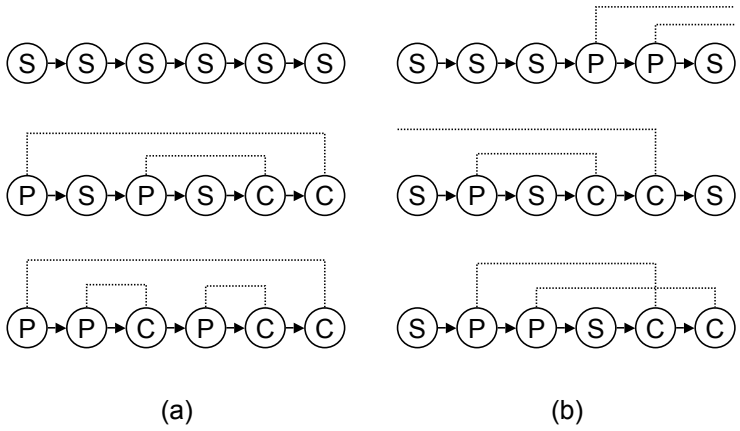


Figure 2.7: Examples of state sequences (a) that are considered in computing $\gamma(i, j, v, w)$ and (b) those that are not considered.

$\mathbf{s}^*$ is the optimal path that satisfies

$$\mathbf{s}^* = \arg\max_{\hat{\mathbf{s}}} P(\mathbf{x}, \mathbf{s} = \hat{\mathbf{s}}|\Theta),$$

where Θ is the set of model parameters. Additionally, we define the variables $\lambda_\ell(i, j, v, w)$ and $\lambda_r(i, j, v, w)$ that will be used for tracing back the optimal state sequence $\mathbf{s}^*$.

2.4.2 Computing the log-probability of the optimal path

Now, the alignment algorithm can be described as follows.

(1) Initialization

For $i = 1, \ldots, L, v = 2, \ldots, M-1$.

$$\begin{aligned}
\gamma(i, i, v, v) &= \begin{cases} \log e(x_i|v) & v \in \mathcal{S} \\ -\infty & \text{otherwise} \end{cases} \\
\lambda_\ell(i, i, v, v) &= (0, 0, 0, 0) \\
\lambda_r(i, i, v, v) &= (0, 0, 0, 0)
\end{aligned}$$

(2) Iteration

For $i = 1, \ldots, L-1, j = i+1, \ldots, L$ and $v = 2, \ldots, M-1, w = 2, \ldots, M-1$.

(i) $v \in \mathcal{C}$ or $w \in \mathcal{P}$

$$\begin{aligned} \gamma(i,j,v,w) &= -\infty \\ \lambda_\ell(i,j,v,w) &= (0,0,0,0) \\ \lambda_r(i,j,v,w) &= (0,0,0,0) \end{aligned}$$

(ii) $v \in \mathcal{P}, w \in \mathcal{S}$

$$\begin{aligned} \gamma(i,j,v,w) &= \max_u \Big[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w)\Big] \\ u^* &= \arg\max_u \Big[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w)\Big] \\ \lambda_\ell(i,j,v,w) &= (i,j-1,v,u^*) \\ \lambda_r(i,j,v,w) &= (j,j,w,w) \end{aligned}$$

(iii) $v \in \mathcal{S}, w \in \mathcal{C}$

$$\begin{aligned} \gamma(i,j,v,w) &= \max_u \Big[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w)\Big] \\ u^* &= \arg\max_u \Big[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w)\Big] \\ \lambda_\ell(i,j,v,w) &= (i,i,v,v) \\ \lambda_r(i,j,v,w) &= (i+1,j,u^*,w) \end{aligned}$$

(iv) $v = P_n, w = C_m$ $(n \neq m), j < i+3$

$$\begin{aligned} \gamma(i,j,v,w) &= -\infty \\ \lambda_\ell(i,j,v,w) &= (0,0,0,0) \\ \lambda_r(i,j,v,w) &= (0,0,0,0) \end{aligned}$$

(v) $v = P_n, w = C_m$ $(n \neq m), j \geq i+3$

$$\begin{aligned}
\gamma(i,j,v,w) &= \max_u \left(\max_{k=i+1,\ldots,j-2} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \right) \\
(k^*,u^*) &= \underset{(u,k),k=i+1,\ldots,j-1}{\arg\max} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \\
\lambda_\ell(i,j,v,w) &= (i,k^*,v,\bar{v}) \\
\lambda_r(i,j,v,w) &= (k^*+1,j,u^*,w)
\end{aligned}$$

(vi) $v = P_n, w = C_n, j = i+1$

$$\begin{aligned}
\gamma(i,j,v,w) &= \log e(x_i|v) + \log t(v,w) + \log e(x_j|w,x_i) \\
\lambda_\ell(i,j,v,w) &= (0,0,0,0) \\
\lambda_r(i,j,v,w) &= (0,0,0,0)
\end{aligned}$$

(vii) $v = P_n, w = C_n, j > i+1$

$$\begin{aligned}
\gamma_1 &= \max_u \left(\max_{k=i+1,\ldots,j-2} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \right) \\
(k^*,u^*) &= \underset{(u,k),k=i+1,\ldots,j-1}{\arg\max} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \\
\gamma_2 &= \max_{u_1,u_2} \Big[\log e(x_i|v) + \log t(v,u_1) \\
&\qquad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i) \Big] \\
(u_1^*,u_2^*) &= \underset{(u_1,u_2)}{\arg\max} \Big[\log e(x_i|v) + \log t(v,u_1) \\
&\qquad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i) \Big] \\
\gamma(i,j,v,w) &= \max(\gamma_1,\gamma_2)
\end{aligned}$$

If $\gamma_1 \geq \gamma_2$,

$$\begin{aligned}
\lambda_\ell(i,j,v,w) &= (i,k^*,v,w) \\
\lambda_r(i,j,v,w) &= (k^*+1,j,u^*,w).
\end{aligned}$$

Otherwise,

$$\begin{aligned}\lambda_\ell(i,j,v,w) &= (i+1, j-1, u_1^*, u_2^*)\\ \lambda_r(i,j,v,w) &= (0,0,0,0).\end{aligned}$$

(viii) $v \in \mathcal{S}, w \in \mathcal{S}$

In this case, the variable $\gamma(i,j,v,w)$ can be updated using any of the update formulae in (ii) or (iii).

(3) Termination

$$\begin{aligned}\log P(\mathbf{x}, \mathbf{s}^*|\Theta) &= \max_{v,w}\Big[\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M)\Big]\\ (v^*, w^*) &= \arg\max_{(v,w)}\Big[\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M)\Big]\\ \lambda^* &= (1, L, v^*, w^*)\end{aligned}$$

■

As shown in the **initialization** step of the algorithm, we start by initializing the values of $\gamma(i,i,v,v)$ for $i = 1, 2, \ldots, L$ and $v = 2, 3, \ldots, M-1$. Since we consider only state sequences where all the pairwise-emission states and the context-sensitive states are paired, the value of $\gamma(i,i,v,v)$ is set to $-\infty$ for $v \in \mathcal{P}$ or $v \in \mathcal{C}$. For single-emission states $v \in \mathcal{S}$, $\gamma(i,i,v,v)$ is simply the logarithm of the emission probability of the symbol x_i at state v. Therefore, we set $\gamma(i,i,v,v) = \log e(x_i|v)$ for $v \in \mathcal{S}$.

Now, let us consider the **iteration** step. As we can see in (i) and (iv), the variable $\gamma(i,j,v,w)$ is set to $-\infty$, whenever the states P_n and C_n do not form pairs. For example, in case (i), if the leftmost state s_i of the subpath $s_i \ldots s_j$ is a context-sensitive state, it cannot be paired with the corresponding pairwise-emission state, since there are no more states to the left of s_i. This is also true when the rightmost state s_j is a pairwise-emission state. In case (iv), the state sequence is either $s_i s_{i+1}$ or $s_i s_{i+1} s_{i+2}$. As $s_i = P_n$ and $s_j = C_m$ where $n \neq m$, the states s_i and s_j cannot form a pair. Moreover, since there are not enough states between s_i and s_j such that both s_i and s_j can form pairs respectively, the probability of such a state sequence is zero. Case (ii) in the **iteration** step deals with the case when $s_i = v$ is a pairwise-emission state while $s_j = w$ is a single-emission state. Since there can be no interaction between s_j and any other state s_k $(i \leq k \leq j-1)$, all the pairwise-emission states and the corresponding context-sensitive states should form pairs inside the subpath $s_i \ldots s_{j-1}$. As $\gamma(i, j-1, v, u)$ is the log-probability of the optimal path among

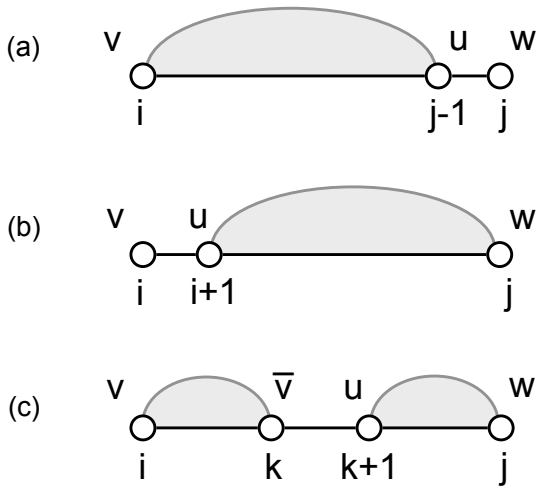


Figure 2.8: Illustration of the iteration step of the algorithm.

all feasible paths $s_i \dots s_{j-1}$, we can compute $\gamma(i, j, v, w)$ by extending $\gamma(i, j-1, v, u)$ to the right by one symbol. We first take the summation of $\gamma(i, j-1, v, u)$ and $\log t(u, w)$ and $\log e(x_i|w)$, and then compute the maximum value of this sum over all u, as described in (ii) of the **iteration** step. Figure 2.8 (a) illustrates this case, where the shaded area indicates that all P_n and C_n states are paired inside the subpath $s_i \dots s_{j-1}$. Similar reasoning holds also for the case when $s_i = v$ is a single-emission state and $s_j = w$ is a context-sensitive state. In this case, $\gamma(i, j, v, w)$ can be obtained by extending $\gamma(i+1, j, u, w)$ as in (iii) of the **iteration** step. This is illustrated in Figure 2.8 (b).

Figure 2.8 (c) depicts the case when $s_i = P_n$ and $s_j = C_m$, where $n \neq m$. In this case, the pairwise-emission state s_i and the context-sensitive state s_j cannot form a pair. Therefore $s_i = P_n$ should pair with $s_k = \bar{v} = C_n$ for some k ($i+1 \leq k \leq j-2$). Similarly, $s_j = C_m$ should form a pair with $s_\ell = \bar{w} = P_m$ for some ℓ ($k+1 \leq \ell \leq j-1$). Consequently, all pairwise-emission states and context-sensitive states inside $s_i \dots s_k$ and $s_{k+1} \dots s_j$ have to exist in pairs. Therefore, we can obtain $\gamma(i, j, v, w)$ by adding $\gamma(i, k, v, \bar{v})$, the transition probability $\log t(\bar{v}, u)$, and $\gamma(k+1, j, u, w)$, and maximizing this sum over all u and k, as shown in (v).

Finally, let us focus on the case when $s_i = P_n$ and $s_j = C_n$. If $j = i+1$, we can simply compute $\gamma(i, j, v, w)$ as in (vi) of the **iteration** step. As s_i pairs with s_j, we consider the emission of the symbols x_i and x_j at the same time. In this way, we know the emitted symbol x_i, and therefore the emission probabilities at the context-sensitive state $s_j = C_n$ can be decided correspondingly. When $j \neq i+1$, the situation is a little bit more complicated. In this case, we have the following two

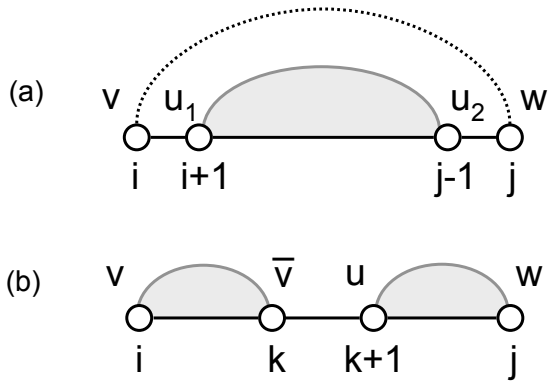


Figure 2.9: Illustration of the iteration step of the algorithm for the case when $s_i = P_n$ and $s_j = C_n$.

possibilities. One possibility is that s_i forms a pair with s_j as shown in Figure 2.9 (a). The dotted line that connects s_i and s_j indicates the pairwise interaction between the two symbols. Since s_i and s_j form a pair, the pairwise-emission states and the context-sensitive states in $s_{i+1} \dots s_{j-1}$ should necessarily exist in pairs. Therefore, the log-probability of the most probable path, where $s_i = P_n$ and $s_j = C_n$ form a pair can be computed as follows

$$\max_{u_1,u_2} \Big[\log e(x_i|v) + \log t(v,u_1) + \gamma(i+1, j-1, u_1, u_2) \\ + \log t(u_2, w) + \log e(x_j|w, x_i) \Big]. \tag{2.7}$$

Another possibility is that $s_i = P_n$ pairs with $s_k = C_n$ for some k between $i+1$ and $j-2$. In this case, $s_j = C_n$ has to pair with $s_\ell = P_n$ for some ℓ between $k+1$ and $j-1$. Therefore, all P_n and C_n states inside $s_i \dots s_k$ and $s_{k+1} \dots s_j$ have to exists in pairs as illustrated in Figure 2.9 (b). The log-probability of all feasible paths, where $s_i = P_n$ does not pair with $s_j = C_n$ can be computed by

$$\max_u \left(\max_{k=i+1,\dots,j-2} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v}, u) + \gamma(k+1, j, u, w) \Big] \right). \tag{2.8}$$

By comparing (2.7) and (2.8) as in (vii) of the **iteration** step, we can compute the log-probability of the most probable path among all subpaths $s_i \dots s_j$ with $s_i = P_n$ and $s_j = C_n$.

Once we have completed the **iteration** step, the log-probability $\log P(\mathbf{x}, \mathbf{s}^*|\Theta)$ of the most probable path $\mathbf{s}^*$ can be computed by comparing $\gamma(1, L, v, w)$ for all $v, w = 2, 3, \dots, M-1$. This is shown in the **termination** step.

2.4.3 Trace-back

Now that we have obtained the log-probability of the optimal path, we can trace back the path $\mathbf{s}^*$ that gave rise to this probability. The variables $\lambda_\ell(i,j,v,w)$ and $\lambda_r(i,j,v,w)$ are used in the trace-back procedure, and we also need a stack T. For notational convenience, let us define $\lambda_t = (i,j,v,w)$. The procedure can be described as the following.

(1) Initialization

$s_i = 0$ $(i = 1, 2, \ldots, L)$.

Push λ^* onto T.

(2) Iteration

Pop $\lambda_t = (i,j,v,w)$ from stack T.

If $\lambda_t \neq (0,0,0,0)$

If $s_i = 0$ then $s_i = v$.

If $s_j = 0$ then $s_j = w$.

$\lambda_\ell(\lambda_t)$ onto T.

$\lambda_r(\lambda_t)$ onto T.

If T is empty then go to **termination** step.

Otherwise, repeat the **iteration** step.

(3) Termination

The optimal path is $\mathbf{s}^* = s_1 s_2 \ldots s_L$. ■

2.4.4 Computational complexity

Let us examine the computational complexity of the alignment algorithm. The algorithm iterates for all $i = 1, \ldots, L-1$, $j = i+1, \ldots, L$ and $v = 2, \ldots, M-1$, $w = 2, \ldots, M-1$. The complexity of each **iteration** step depends on the type of the states v and w. Table 2.2 summarizes the computational complexity of each case of the **iteration** step of the alignment algorithm in Sec. 2.4.2. From this

Case	Complexity for one iteration	Number of iterations	Overall complexity
i	$O(1)$	$O(L^2M_1M)$	$O(L^2M_1M)$
ii	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
iii	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
iv	$O(1)$	$O(LM_1^2)$	$O(LM_1^2)$
v	$O(ML)$	$O(L^2M_1^2)$	$O(L^3M_1^2M)$
vi	$O(1)$	$O(LM_1)$	$O(LM_1)$
vii	$O(ML)+O(M^2)$	$O(L^2M_1)$	$O(L^3M_1M)+O(L^2M_1M^2)$
viii	$O(M)$	$O(L^2M_2^2)$	$O(L^2M_2^2M)$

Table 2.2: Computational complexity of the csHMM alignment algorithm.

table, we can compute the total complexity of the alignment algorithm as follows

$$O(L^3M_1^2M)+O(L^2M_1M^2)+O(L^2M_2^2M). \tag{2.9}$$

Although the complexity in (2.9) is higher than $O(LM^2)$ of the Viterbi algorithm, it is still a polynomial in L and M, which is much more efficient than $O(LM^L)$ of the exhaustive search approach. The computational complexity of the alignment algorithm for general SCFGs in Chomsky normal form is $O(L^3M^3)$ [25, 60]. As we can see, the computational cost of both algorithms increases with $O(L^3M^3)$, in general.

2.5 Computing the probability of an observed sequence

Another important problem that arises in using HMMs for real-world applications is the following. Given an observation sequence $\mathbf{x} = x_1 \dots x_L$, how can we efficiently compute the probability $P(\mathbf{x}|\Theta)$ that this sequence was generated by the HMM with the set of parameters Θ? This is typically called the *scoring problem* for the following reason. Assume that we have K different models, each with different set of parameters $\Theta_k (k = 1, 2, \dots, K)$. Among these K HMMs, which one should we choose such that the probability of observing $\mathbf{x}$ is maximized? In order to choose the best model, we have to score each model based on the observation sequence $\mathbf{x}$, where the probability $P(\mathbf{x}|\Theta)$ is the natural choice for the score. Since $P(\mathbf{x}|\Theta)$ can be used for scoring different HMMs, the problem of computing this probability is called the scoring problem.[4]

For regular HMMs, we can use the *forward algorithm* for solving this problem, whose complexity

[4]Given a number of symbol sequences, we can find the best match to the given model by computing the observation probability of each sequence and comparing these probabilities. In this case, the probability is used for scoring the observed sequences. This is another reason why the given problem is called a *scoring problem*.

is the same as that of the Viterbi algorithm. However, due to the context-sensitive property of csHMMs, this algorithm cannot be directly used for scoring csHMMs. Even though the forward algorithm cannot be used for computing the probability $P(\mathbf{x}|\Theta)$ in context-sensitive HMMs, we can adopt a similar approach that was previously used in the optimal alignment algorithm. In Section 2.5.1, we propose a dynamic programming algorithm for scoring csHMMs. In addition to this, we also propose the outside algorithm for csHMM in Section 2.5.2. This algorithm can be used together with the scoring algorithm for training context-sensitive HMMs, as will be elaborated in Section 2.6.

2.5.1 Scoring of csHMM

The csHMM scoring algorithm can be viewed as a variant of the alignment algorithm, where the *max* operators are replaced by *sums*. Conceptually, this algorithm is somewhat similar to the *inside algorithm* [60] that is used for scoring SCFGs. As in the alignment algorithm, we start from the inside of the observed symbol sequence and iteratively proceed to the outward direction. During this process, the pairwise-emission state P_n and the context-sensitive state C_n that interact with each other are considered at the same time.

In order to describe the algorithm, we use the same notations as in Sec. 2.4.2. In addition to this, we define the *inside variable* $\alpha(i, j, v, w)$ as the probability of all subpaths $s_i \dots s_j$ with $s_i = v$ and $s_j = w$. It is assumed that all pairwise-emission states P_n inside the path are paired with the corresponding context-sensitive states C_n. Now, the scoring algorithm can be described as follows.

(1) Initialization

For $i = 1, \dots, L, v = 2, \dots, M-1$.

$$\alpha(i, i, v, v) = \begin{cases} e(x_i|v) & v \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

(2) Iteration

For $i = 1, \dots, L-1, j = i+1, \dots, L$ and $v = 2, \dots, M-1, w = 2, \dots, M-1$.

(i) $v \in \mathcal{C}$ or $w \in \mathcal{P}$

$$\alpha(i, j, v, w) = 0$$

(ii) $v \in \mathcal{P}, w \in \mathcal{S}$

$$\alpha(i,j,v,w) = \sum_u \Big[\alpha(i,j-1,v,u)t(u,w)e(x_j|w)\Big]$$

(iii) $v \in \mathcal{S}, w \in \mathcal{C}$

$$\alpha(i,j,v,w) = \sum_u \Big[e(x_i|v)t(v,u)\alpha(i+1,j,u,w)\Big]$$

(iv) $v = P_n, w = C_m\ (n \neq m), j < i+3$

$$\alpha(i,j,v,w) = 0$$

(v) $v = P_n, w = C_m\ (n \neq m), j \geq i+3$

$$\alpha(i,j,v,w) = \sum_u \sum_{k=i+1}^{j-2} \alpha(i,k,v,\bar{v})t(\bar{v},u)\alpha(k+1,j,u,w)$$

(vi) $v = P_n, w = C_n, j = i+1$

$$\alpha(i,j,v,w) = e(x_i|v)t(v,w)e(x_j|w,x_i)$$

(vii) $v = P_n, w = C_n, j > i+1$

$$\begin{aligned}\alpha(i,j,v,w) &= \sum_u \sum_{k=i+1}^{j-2} \alpha(i,k,v,w)t(w,u)\alpha(k+1,j,u,w) \\ &\quad + \sum_{u_1}\sum_{u_2} \Big[e(x_i|v)t(v,u_1)\alpha(i+1,j-1,u_1,u_2)t(u_2,w)e(x_j|w,x_i)\Big]\end{aligned}$$

(viii) $v \in \mathcal{S}, w \in \mathcal{S}$

In this case, the variable $\alpha(i,j,v,w)$ can be updated using any of the update formulae in (ii) or (iii).

(3) Termination

$$P(\mathbf{x}|\Theta) = \sum_v \sum_w t(1,v)\alpha(1,L,v,w)t(w,M)$$ ■

At the end of the algorithm, we can obtain the probability $P(\mathbf{x}|\Theta)$ that the given csHMM will generate the observation sequence $\mathbf{x}$. The computational complexity of this algorithm is the same as the complexity of the alignment algorithm, which is shown in (2.9).

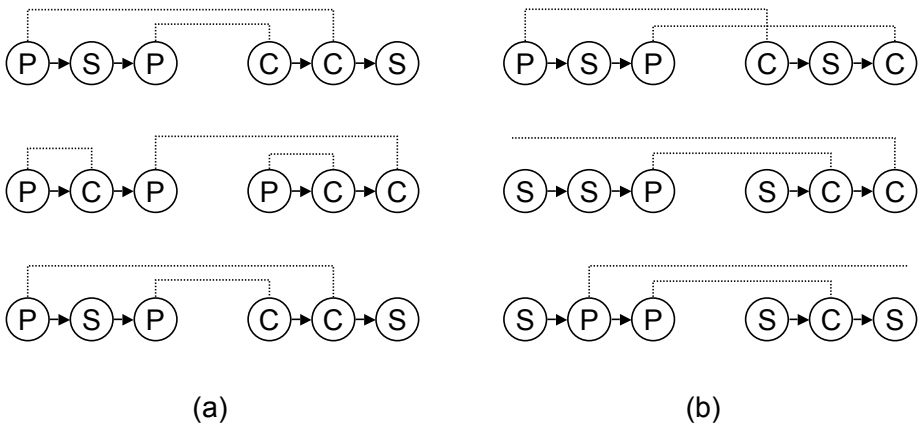


Figure 2.10: Examples of state sequences (a) that are considered in computing $\beta(i, j, v, w)$ and (b) those that are not considered.

2.5.2 The outside algorithm

In a similar fashion, we can define the *outside variable* $\beta(i, j, v, w)$ to be the probability of all subpaths $s_1 \dots s_i s_j \dots s_L$, where $s_i = v$ and $s_j = w$. In other words, $\beta(i, j, v, w)$ contains the probability of the entire sequence excluding $x_{i+1} \dots x_{j-1}$. This variable is needed for parameter re-estimation of csHMM, which will be elaborated in Section 2.6. As in Section 2.5.1, we assume that all pairwise-emission states P_n in $s_1 \dots s_i s_j \dots s_L$ are paired with the corresponding context-sensitive states C_n in a nested manner. Figure 2.10 illustrates the state sequences that are considered in computing the variable $\beta(i, j, v, w)$, and the ones that are not taken into account.

In the outside algorithm, we start computing $\beta(i, j, v, w)$ from the outside of the sequence and proceed to the inward direction. As in the scoring algorithm, whenever there is an interaction between two symbols, the emission of these symbols are considered together. The inside variable $\alpha(i, j, v, w)$, which has been computed previously, is needed for computing the outside variable $\beta(i, j, v, w)$. Now, we can solve for $\beta(i, j, v, w)$ as follows.

(1) Initialization

For $i = 1, \ldots, L, v = 1, \ldots, M$.

$$\beta(0, L+1, v, w) = \begin{cases} 1 & v = 1, w = M \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(i, L+1, v, w) = \begin{cases} \sum_u t(1, u)\alpha(1, i, u, v) & w = M \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(0, i, v, w) = \begin{cases} \sum_u \alpha(i, L, w, u)t(u, M) & v = 1 \\ 0 & \text{otherwise} \end{cases}$$

(2) Iteration

For $i = 1, \ldots, L-1, j = i+1, \ldots, L$ and $v = 1, \ldots, M, w = 1, \ldots, M$.

(i) $v = 1$ or $w = M$

$$\beta(i, j, v, w) = 0$$

(ii) $v \in \mathcal{P}$, $w \in \mathcal{P}$

$$\beta(i, j, v, w) = \sum_u \sum_{k=j+2}^{L+1} \beta(i, k, v, u)\alpha(j, k-1, w, \bar{w})t(\bar{w}, u)$$

(iii) $v \in \mathcal{C}$, $w \in \mathcal{C}$

$$\beta(i, j, v, w) = \sum_u \sum_{k=0}^{i-2} \beta(k, j, u, w)\alpha(k+1, i, \bar{v}, v)t(u, \bar{v})$$

(iv) $v \in \mathcal{C}$, $w \in \mathcal{P}$

$$\beta(i, j, v, w) = \sum_{u_1, u_2} \sum_{k_1=0}^{i-2} \sum_{k_2=j+2}^{L+1} \beta(k_1, k_2, u_1, u_2)\alpha(k_1+1, i, \bar{v}, v) \times \alpha(j, k_2-1, w, \bar{w})t(u_1, \bar{v})t(\bar{w}, u_2)$$

(v) $v \notin \mathcal{S}$, $w \in \mathcal{S}$

$$\beta(i, j, v, w) = \sum_u \beta(i, j+1, v, u)t(w, u)e(x_j|w)$$

(vi) $v \in \mathcal{S}, w \notin \mathcal{S}$

$$\beta(i,j,v,w) = \sum_u \beta(i-1,j,u,w)t(u,v)e(x_i|v)$$

(vii) $v = P_n, w = C_m (n \neq m)$

$$\beta(i,j,v,w) = 0$$

(viii) $v = P_n, w = C_n$

$$\beta(i,j,v,w) = \sum_{u_1,u_2} \beta(i-1,j+1,u_1,u_2)t(u_1,v)e(x_i|v)e(x_j|w,x_i)t(w,u_2)$$

(ix) $v \in \mathcal{S}, w \in \mathcal{S}$

In this case, the variable $\alpha(i,j,v,w)$ can be updated using either (v) or (vi).

(3) Termination

$$P(\mathbf{x}|\Theta) \quad = \quad \sum_{v,w} \beta(i,i+1,v,w)t(v,w) \qquad \text{for any } i \qquad \blacksquare$$

Let us first look at the **initialization** step. For the case of an empty string, i.e., when $i = 0$ and $j = L+1$, we set $\beta(0, L+1, 1, M)$ to unity. When $i \geq 1$ and $j = L+1$, all the pairwise interactions have to occur inside the subpath $s_1 \ldots s_i$. Since $\alpha(1,i,u,v)$ is the probability of all subpaths for $x_1 x_2 \ldots x_i$ with $s_1 = u$ and $s_i = v$, we can compute $\beta(i, L+1, v, M)$ by taking the product of the transition probability from state 1 to state u and the inside variable $\alpha(1,i,u,v)$, and then adding this product over all u. The case when $i = 0$ and $j \leq L$ can be treated similarly. These are shown in the **initialization** step.

After the initialization of the outside variable $\beta(i,j,v,w)$, we proceed into the **iteration** step. Firstly, consider the case when $v \in \mathcal{P}$ and $w \in \mathcal{P}$. Since all pairwise-emission states have to be paired with the corresponding context-sensitive states in a nested manner, $s_j = w$ has to pair with $\bar{w}$ between $j+1$ and $k-1$ as shown in Figure 2.11 (a). As in Figure 2.8 and Figure 2.9, the shaded regions indicate that all P_n and C_n states are paired inside each region. Similarly, $s_i = v$ has to form a pair with $\bar{v}$ between k and L, and all the interactions in the subpath $x_1 \ldots x_i x_k \ldots x_L$ should be paired in a nested manner. Since the probability of each subpath $s_j \ldots s_{k-1}$ and $s_1 \ldots s_i s_k \ldots s_L$ is contained in $\alpha(j,k-1,w,\bar{w})$ and $\beta(i,k,v,u)$ respectively, we can compute $\beta(i,j,v,w)$ as described in (ii) of the **iteration** step. Figure 2.11 (b) illustrates the case when $v \in \mathcal{C}$ and $w \in \mathcal{C}$. In this case,

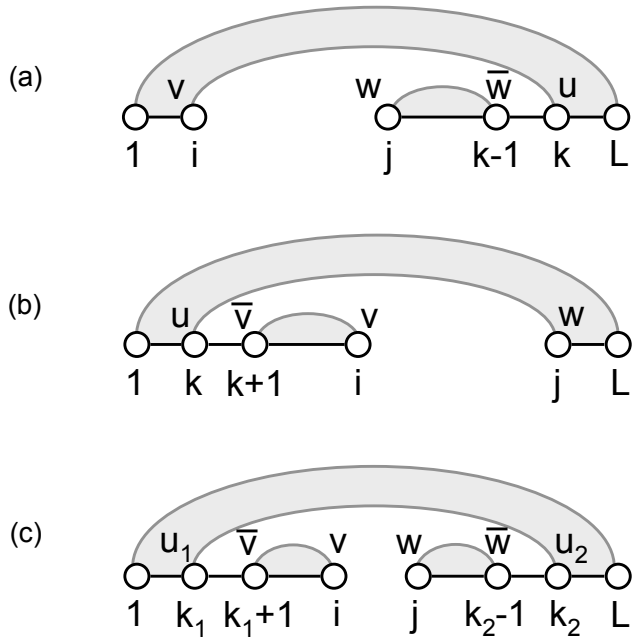


Figure 2.11: Illustration of the iteration step of the outside algorithm. (a) Case (ii). (b) Case (iii). (c) Case (iv).

$\beta(i, j, v, w)$ can be updated in a similar manner as shown in (iii). Figure 2.11 (c) shows the case when $v \in \mathcal{C}$ and $w \in \mathcal{P}$. As shown in the figure, $s_i = v$ has to pair with $\bar{v}$ between $k_1 + 1$ and $i - 1$ and $s_j = w$ also has to pair with $\bar{w}$ between $j + 1$ and $k_2 - 1$. All the other interactions have to be confined within the state sequence $s_1 \dots s_{k_1} s_{k_2} s_L$. Therefore, $\beta(i, j, v, w)$ can be computed as in (iv) of the **iteration** step.

When w is a single-emission state, $\beta(i, j, v, w)$ can be obtained simply by extending $\beta(i, j + 1, v, u)$ by one sample, as depicted in Figure 2.12 (a). As shown in (v) of the **iteration** step, we first compute the product of $\beta(i, j + 1, v, u)$ and the transition probability $t(w, u)$ and the emission probability of the symbol x_j at the state $s_j = w$, and add the product over u. $\beta(i, j, v, w)$ can be computed likewise when $v \in \mathcal{S}$, as described in (vi). If both v and w are single-emission states, we may use either (v) or (vi) for updating the outside variable $\beta(i, j, v, w)$. Finally, let us consider the case when $v = P_n$ and $w = C_m$. Since there can be no crossing interactions, $s_i = P_n$ and $s_j = C_m$ have to interact with each other, as illustrated in Figure 2.12 (c). The dotted line indicates the pairwise interaction between x_i and x_j. For this reason, n has to be the same as m, and $\beta(i, j, v, w)$ is set to zero if $n \neq m$. For $n = m$, we can compute $\beta(i, j, v, w)$ by extending $\beta(i - 1, j + 1, u_1, u_2)$ as

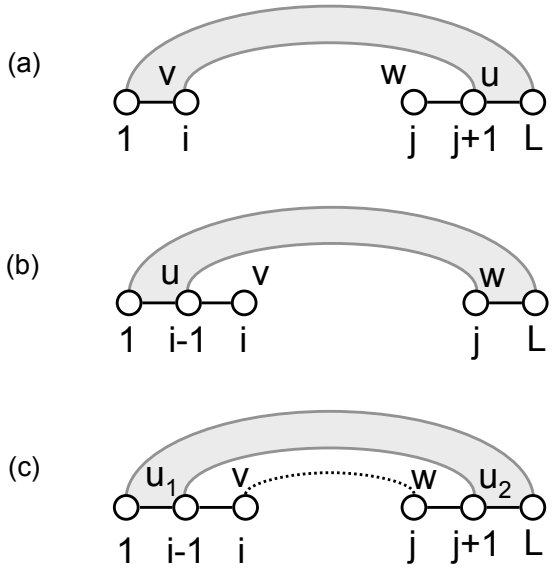


Figure 2.12: Illustration of the iteration step of the outside algorithm. (a) Case (v). (b) Case (vi). (c) Case (viii).

shown in (viii) of the **iteration** step.

Once the **iteration** step is complete, the **termination** step of the outside algorithm also yields the probability $P(\mathbf{x}|\Theta)$ like the scoring algorithm in Section 2.5.1. The computational complexity of the outside algorithm is usually not an issue, since it is mainly used for training the model offline.

2.6 Estimating the model parameters

In order to apply context-sensitive HMMs to real-world problems, it is crucial to adjust the model parameters in an optimal way. Therefore, it is important to find a method for optimizing the set of model parameters Θ, such that the probability $P(\mathbf{x}|\Theta)$ of the given observation sequence $\mathbf{x}$ is maximized. The process of finding these optimal parameters is typically called "training." Although it is infeasible to find an analytical solution for the optimal parameters, we can use the *EM (expectation-maximization)* approach for finding parameters that achieve a local maximum of $P(\mathbf{x}|\Theta)$. In traditional HMMs, *Baum-Welch algorithm* [6] has been widely used for iterative update of the parameters. Similarly, there exists an EM algorithm, called the *inside-outside algorithm* [60], which can be used for optimizing the model parameters of a SCFG. Both algorithms compute an estimate $\widehat{\Theta}$ of the model parameters based on the given observation sequence and the current set of

parameters Θ. The current set of model parameters Θ is then updated by this estimate $\widehat{\Theta}$, and this re-estimation procedure is repeated until a certain stopping criterion is satisfied.

A similar approach can also be used for iterative re-estimation of the model parameters in a context-sensitive HMM. In order to describe the re-estimation algorithm, let us first define the following variables.

$$\begin{aligned}
\tau_i(v,w) &= \text{The probability that } s_i = v \text{ and } s_{i+1} = w \text{ given the model } \Theta \\
&\quad \text{and the observed symbol string } \mathbf{x} \\
\sigma_i(v) &= \text{The probability that } s_i = v \text{ given } \Theta \text{ and } \mathbf{x} \\
\delta_v(i,j) &= \text{The probability that } s_i = v \text{ and } s_j = \bar{v} \text{ have an interaction with} \\
&\quad \text{each other}
\end{aligned}$$

Firstly, $\tau_i(v,w)$ can be computed as follows

$$\tau_i(v,w) = \frac{\beta(i,i+1,v,w)t(v,w)}{P(\mathbf{x}|\Theta)}. \tag{2.10}$$

The probability $\sigma_i(v)$ can be obtained simply by adding $\tau_i(v,w)$ over all w

$$\sigma_i(v) = \sum_w \tau_i(v,w). \tag{2.11}$$

Finally, the probability $\delta_v(i,j)$ can be written as

$$\delta_v(i,j) = \frac{\sum_{u_1,u_2} \alpha(i,j,v,\bar{v})\beta(i-1,j+1,u_1,u_2)t(u_1,v)t(\bar{v},u_2)}{P(\mathbf{x}|\Theta)}. \tag{2.12}$$

Based on these probabilities, we can compute the expected number of occurrences of a state v in the path as well as the number of transitions from a state v to another state w. For example, if we add $\tau_i(v,w)$ over all locations i, we get

$$\sum_{i=0}^{L} \tau_i(v,w) = \text{Expected number of transitions from } v \text{ to } w. \tag{2.13}$$

Similarly, if we add $\sigma_i(v)$ over all i, we obtain the following

$$\sum_{i=0}^{L} \sigma_i(v) = \text{Expected number of transitions from } v. \tag{2.14}$$

Now, we can re-estimate the model parameters of the csHMM using the following method. To begin with, let us first compute an estimate of the transition probability from v to w, where $v \in \mathcal{P}$ or $v \in \mathcal{S}$. In this case, the estimate is given by

$$\begin{aligned} \hat{t}(v,w) &= \frac{\text{Expected number of transitions from } v \text{ to } w}{\text{Expected number of transitions from } v} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v,w)}{\sum_{i=0}^{L} \sigma_i(v)}. \end{aligned} \tag{2.15}$$

For $v = C_n$, the set of states to which v can make a transition differs depending on whether the corresponding stack is empty or not. If $w \in \mathcal{E}_n$, i.e., if w is a state to which $v = C_n$ can make a transition when the stack is empty,

$$\begin{aligned} \hat{t}(v,w) &= \frac{\text{Expected number of transitions from } v = C_n \text{ to } w \in \mathcal{E}_n}{\text{Expected number of transitions from } v = C_n \text{ to any state in } \mathcal{E}_n} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v,w)}{\sum_{i=0}^{L} \sum_{u \in \mathcal{E}_n} \tau_i(v,u)}. \end{aligned} \tag{2.16}$$

If $w \in \mathcal{F}_n$, then we can obtain the estimate by

$$\begin{aligned} \hat{t}(v,w) &= \frac{\text{Expected number of transitions from } v = C_n \text{ to } w \in \mathcal{F}_n}{\text{Expected number of transitions from } v = C_n \text{ to any state in } \mathcal{F}_n} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v,w)}{\sum_{i=0}^{L} \sum_{u \in \mathcal{F}_n} \tau_i(v,u)}. \end{aligned} \tag{2.17}$$

Now, let us estimate the emission probability $e(x|v)$ and $e(x|v,x_p)$. For $v \in \mathcal{P}$ or $v \in \mathcal{S}$, the emission probability does not depend on the context. Therefore, we can compute the estimate $\hat{e}(x|v)$ of the emission probability as follows

$$\begin{aligned} \hat{e}(x|v) &= \frac{\text{Expected number of times that the symbol } x \text{ was emitted at state } v}{\text{Expected number of occurrences of state } v} \\ &= \frac{\sum_{i=1|x_i=x}^{L} \sigma_i(v)}{\sum_{i=1}^{L} \sigma_i(v)}. \end{aligned} \tag{2.18}$$

In contrast, if v is a context-sensitive state, the emission probability is dependent on the symbol x_p that was emitted at the corresponding pairwise-emission state $\bar{v}$. Bearing this in mind, we can

estimate $\hat{e}(x|v, x_p)$ as follows

$$\begin{aligned}\hat{e}(x|v, x_p) &= \frac{\text{Expected number of emissions of } x \text{ at state } v \in \mathcal{C} \text{ given } x_p}{\text{Expected number of emissions at state } v \in \mathcal{C} \text{ given } x_p} \\ &= \frac{\sum_{j=2|x_j=x}^{L} \sum_{i=1|x_i=x_p}^{j-1} \delta_v(i,j)}{\sum_{j=2}^{L} \sum_{i=1|x_i=x_p}^{j-1} \delta_v(i,j)}. \end{aligned} \tag{2.19}$$

Although we derived these update formulae based on a single observation sequence $\mathbf{x}$, they can be easily extended for multiple training sequences. When we have more than one observation sequence for training, we simply add all the expected counts over all sequences, and use these numbers for estimating the model parameters.

Now that we have the estimates $\hat{t}(v, w)$, $\hat{e}(x|v)$ and $\hat{e}(x|v, x_c)$, we can update the model parameters by these estimates

$$\begin{aligned} t(v, w) &\longleftarrow \hat{t}(v, w) \\ e(x|v) &\longleftarrow \hat{e}(x|v) \\ e(x|v, x_p) &\longleftarrow \hat{e}(x|v, x_p). \end{aligned}$$

We repeat this re-estimation procedure until a certain stopping criterion is satisfied. As mentioned earlier, the training of the model is performed offline, and therefore the computational cost of the re-estimation algorithm is usually not a critical issue.

2.7 Experimental results

In order to test the proposed algorithms, let us consider the example in Figure 2.13. This csHMM

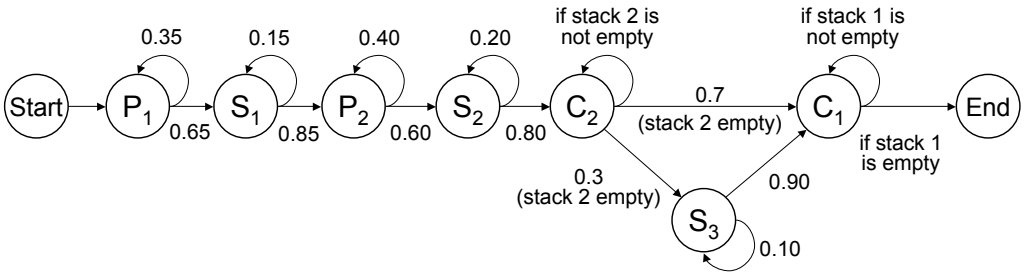


Figure 2.13: An example of a context-sensitive HMM.

	A	C	G	U
P_1	0.55	0.05	0.05	0.35
S_1	0.15	0.35	0.35	0.15
P_2	0.40	0.05	0.05	0.50
S_2	0.05	0.60	0.30	0.05
S_3	0.05	0.10	0.75	0.10

Table 2.3: Emission probabilities $e(x|v)$.

generates sequences with long-range correlations between distant symbols. Such pairwise dependencies are commonly found in the so-called *iron response elements (IREs)* in RNA sequences [54]. The model in Figure 2.13 has three single-emission states S_1, S_2 and S_3, and two pairs of pairwise-emission states and context-sensitive states. Each pair (P_1, C_2) and (P_2, C_2) is associated with a separate stack. The transition probabilities are shown in Figure 2.13 along the edges. Each state emits one of the four symbols $\mathcal{A} = \{A, C, G, U\}$, where the emission probabilities are as shown in Table 2.3. Every row in Table 2.3 contains the emission probabilities that each output symbol will be emitted at the given state. For example, the first row in Table 2.3 shows the probabilities that the symbols A, C, G, and U will be emitted at P_1. Therefore, each row adds up to unity. The emission probabilities at C_n are dependent on the symbol x that was emitted at the corresponding state P_n. In this example, we set the emission probabilities of C_1 and C_2 such that they always emit the "complementary" symbol of x ($A \leftrightarrow U$ and $C \leftrightarrow G$ are complementary to each other).

Now, let us assume that the observed symbol string is $\mathbf{x} = AUCUACUAAU$. What is the optimal state sequence $\mathbf{s}^* = s_1 s_2 \dots s_{10}$ that maximizes the probability of observing $\mathbf{x}$ based on the specified model? Using the alignment algorithm elaborated in Section 2.4, we obtained

$$\mathbf{s}^* = P_1 P_1 S_1 P_2 P_2 S_2 C_2 C_2 C_1 C_1, \tag{2.20}$$

where the log-probability of $\mathbf{s}^*$ was $\log_2 P(\mathbf{x}, \mathbf{s}^*|\Theta) = -12.2165$. In order to check the validity of this result, we performed an exhaustive search over all possible paths. Since the length of the sequence is $L = 10$, and as there are $M - 2 = 7$ emitting states, we have $(M-2)^L = 7^{10} = 282,475,249$ possibilities. By comparing the log-probabilities of all paths, we obtained the same optimal path as (2.20) with the same log-probability, which shows that the optimal alignment algorithm works as expected.

Similarly, we computed the probability of the sequence $\mathbf{x}$, given the model in Figure 2.13. Using

	A	C	G	U
P_1	0.5503	0.0617	0.0348	0.3533
S_1	0.1258	0.4094	0.3481	0.1167
P_2	0.4067	0.0628	0.0400	0.4905
S_2	0.0477	0.5543	0.3528	0.0452
S_3	0.0870	0.1364	0.7073	0.0693

Table 2.4: Estimated emission probabilities $e(x|v)$ after 10 iterations.

the scoring algorithm in Section 2.5, we obtained

$$P(\mathbf{x}|\Theta) = 2.1146 \times 10^{-4}. \tag{2.21}$$

Again, we computed the probability using the brute-force approach by considering all possible paths and adding the probability of each path. As a result, we obtained

$$P(\mathbf{x}|\Theta) = \sum_{\mathbf{s}} P(\mathbf{x}, \mathbf{s}|\Theta) = 2.1146 \times 10^{-4},$$

which is the same as (2.21). As we can see from these results, the proposed scoring and alignment algorithms are capable of finding the same solutions as the brute-force methods in a much more efficient manner.

Now, let us consider the training of the csHMM. In order to test the parameter re-estimation algorithm, we first generated 200 symbol sequences based on the model in Figure 2.13. Then, we randomly initialized the transition and emission probabilities of the model, and ran the algorithm in Section 2.6 to optimize the model parameters. Figure 2.14 shows the arithmetic mean and the geometric mean of the sequence probabilities after each iteration. As we can see, the mean values are nearly zero in the beginning, since the parameters have been randomly initialized. The model parameters quickly converged to the final values after only a few iterations, and the converged values were very close to the original values. Table 2.4 shows the estimated emission probabilities after 10 iterations. By comparing it with Table 2.3, we can see that the estimated values are close to the original ones.

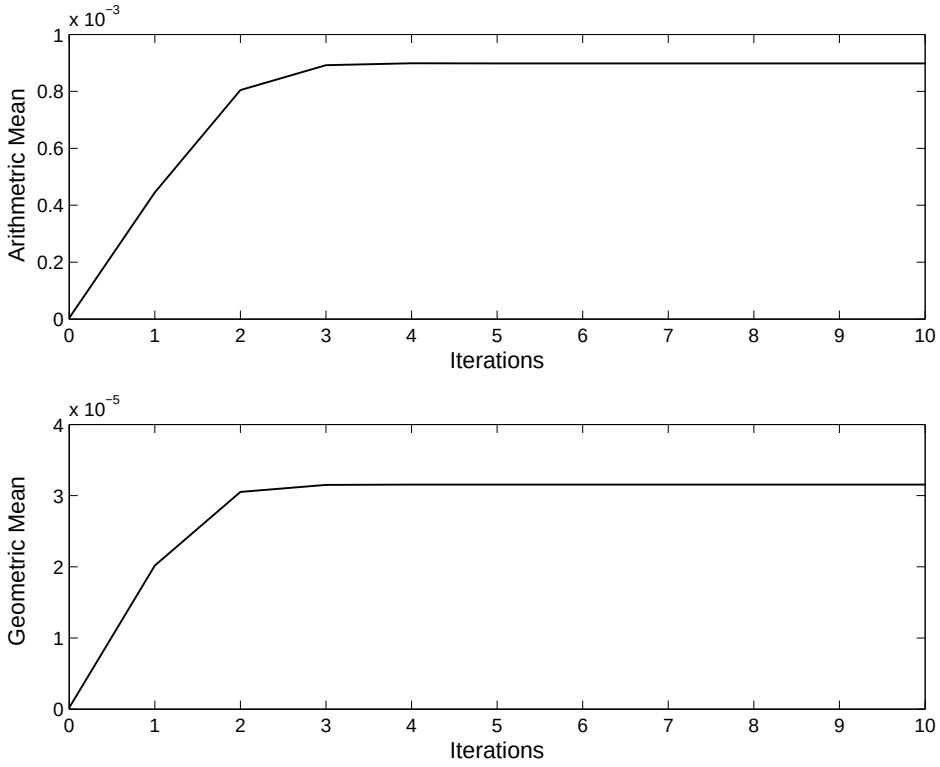


Figure 2.14: The arithmetic mean (top) and the geometric mean (bottom) after each iteration.

2.8 Discussions

As we have seen, context-sensitive HMMs can be effectively used for modeling pairwise interactions between distant symbols in a symbol string. In this section, we consider possible extensions of the basic model and discuss several interesting issues regarding the csHMM.

2.8.1 Emission of multiple symbols

In this chapter, we assumed that every state in the csHMM emits only one symbol at a time. Based on this assumption, we considered only sequences with *pairwise* dependencies between distant symbols that are arranged in a nested manner. However, we can easily extend the basic model such that it can also describe *non-pairwise* dependencies, by allowing the states to emit two or more symbols at a time. For example, we may modify the model in Figure 2.4 such that the context-sensitive state C_1 emits two symbols at a time. When we enter C_1, the symbol x that is on the top of the stack is popped out, and the emission probabilities of C_1 are adjusted so that it emits xx. In

Figure 2.15: An example sequence that can be generated by the modified model of Figure 2.4.

this way, the modified model will generate sequences of the form $x_1x_2 \ldots x_N x_N x_N \ldots x_2x_2x_1x_1$. An example of such a symbol sequence is shown in Figure 2.15. As shown in this figure, the correlations still occur in a nested manner, but they are not limited to pairwise correlations any more. Such modifications can be easily incorporated into the algorithms described in the previous sections. For example, we may change the second term in the update formula (vii) in Section 2.5.1 to

$$\sum_{u_1}\sum_{u_2}\Big[e(x_i \ldots x_{i+\delta_n^p-1}|v)t(v,u_1)\alpha(i+\delta_n^p, j-\delta_n^c, u_1, u_2) \times t(u_2,w)e(x_{j-\delta_n^c+1}\ldots x_j|w, x_i \ldots x_{i+\delta_n^p-1})\Big],$$

when the csHMM is modified such that the pairwise-emission state P_n emits δ_n^p symbols at a time and the corresponding context-sensitive state C_n emits δ_n^c symbols at a time.

2.8.2 Modeling crossing correlations

Although we have mainly focused on context-sensitive HMMs that generate sequences with nested correlations, the descriptive power of the proposed model is not restricted to such a correlation structure. In fact, csHMM can be used to represent sequences with various correlations between symbols, including crossing dependencies. Figure 2.16 shows an example of such a csHMM. Note that the csHMM in Figure 2.16 still uses stacks, but the P_n and C_n states are arranged such that the model gives rise to crossing interactions between symbols. Furthermore, we may also replace the *stack* by a *queue* to represent other types of interactions. For example, we can describe the *copy language* by using a csHMM with a queue. The copy language includes all sequences that consist of the concatenation of two identical sequences. The model illustrated in Figure 2.17 can effectively represent such a language.

When the given csHMM generates sequences with crossing interactions, the algorithms in Section 2.4, Section 2.5, and Section 2.6 cannot be directly used. However, it is possible to extend the

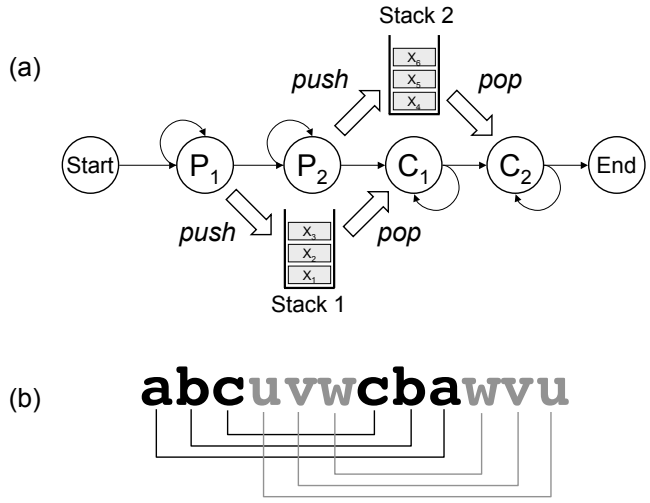


Figure 2.16: (a) A csHMM that results in crossing interactions. (b) An example of a generated symbol sequence. The lines indicate the correlations between symbols.

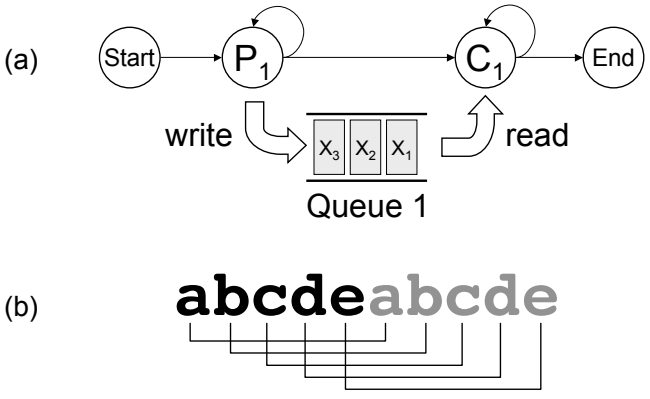


Figure 2.17: (a) A csHMM that represents a copy language. (b) An example of a generated symbol sequence.

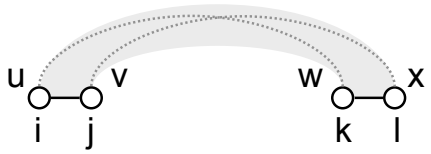


Figure 2.18: An illustration of the basic concept of the algorithm that can be used when there exist crossing interactions. The dotted lines show examples of correlations that can be taken into consideration based on this setting.

proposed algorithms such that they can be used for csHMMs with crossing interactions as those shown in Figure 2.16 and Figure 2.17. For example, for scoring such csHMMs, we may define the variable $\alpha(i, j, k, \ell, u, v, w, x)$ as the probability of the subsequence $x_i \dots x_j x_k \dots x_\ell$ ($i \leq j < k \leq \ell$), where $s_i = u, s_j = v, s_k = w, s_\ell = x$ and all P_n states are paired with the corresponding C_n states inside the subpath $s_i \dots s_j s_k \dots s_\ell$. We can compute $\alpha(\dots)$ in a recursive manner by considering crossing correlations between s_i and s_k, s_j and s_ℓ, and so forth.[5] This is illustrated in Figure 2.18. In this case, the computational complexity of the algorithm will be considerably higher than $O(L^3 M^3)$.

2.8.3 Comparison with other variants of HMM

As mentioned earlier, there exist many interesting variants of the traditional HMM, which extend the basic model in various ways [35, 49, 50, 70, 79, 80, 136]. For example, the *hidden semi-Markov model (HSMM)* allows us to associate an explicit state occupancy distribution with each state [35, 49, 50, 70, 136], instead of using the implicit geometric state occupancy distribution in the basic HMM. However, the hidden states in the HSMM are not context sensitive, and the emission and transition probabilities of the future states do not explicitly depend on the symbols that have been emitted previously. Therefore, these models cannot explicitly model pairwise correlations between distant symbols as the csHMM does.

There exists another interesting generalization of the HMM called the *pairwise Markov chain (PMC)* [80]. The PMC assumes that the pair of the random variables (x_i, s_i) is a Markov chain. This model is mathematically more general than the HMM, which is a special case of the PMC, where the hidden state s_i satisfies the Markov property. Since the pair (x_i, s_i) is a Markov chain, the probabilities associated with x_i, s_i, and (x_i, s_i) do not depend on the previous emissions, and the

[5] For example, we can implement a dynamic programming algorithm that can be used for aligning csHMMs with crossing interactions, in a similar manner as the algorithm proposed in [85].

PMC cannot be used for describing complex correlations such as the ones observed in palindromes. This is also the case with the *triplet Markov chain (TMC)* [79], which is a further generalization of the PMC, and there exists a fundamental difference between the csHMM and the PMC/TMC.

2.8.4 Comparison with other stochastic grammars

As HMMs are equivalent to stochastic regular grammars (SRG), the csHMM can be viewed as an extension of the SRG with specific context-sensitive production rules. Therefore, the SRG is a proper subset of the proposed csHMM. The context-sensitive property of the csHMM enables the model to describe explicit dependencies between distant symbols, which are beyond the descriptive power of SRGs. As a result, the csHMM is capable of modeling sequences with nested correlations, which are characteristic of languages that are described by SCFGs. This implies that the csHMM can be used as a good alternative to SCFGs, in many practical situations. Moreover, the csHMM is also capable of modeling crossing correlations as illustrated in the examples shown in Figure 2.16 and Figure 2.17. This cannot be done using a SCFG, and we have to resort to higher-order grammars such as the stochastic context-sensitive grammars (SCSG). However, there exist also languages that can be described by a context-free grammar but not by a csHMM. One such example can be found in Appendix A. This shows that even though there is a considerable overlap between csHMMs and SCFGs, neither of them fully includes the other. Finally, the csHMM can be viewed as a stochastic formal grammar that uses only *non-contracting* production rules.[6] It is known that for any non-contracting grammar there exists an equivalent context-sensitive grammar [51]. This implies that the csHMM is a subset of the stochastic context-sensitive grammars (SCSG). The full relationship between the csHMM and other stochastic grammars is illustrated in the Venn diagram shown in Figure 2.19.

The capability of modeling various correlations (including nested and/or crossing interactions) based on a single framework is a significant advantage of csHMMs over SRGs and SCFGs. Another advantage of the proposed model is that it can *explicitly* describe the dependencies between distant symbols. This allows us to model the symbol sequences of our interest in a simple and a direct way, which can be an advantage (although arguable) compared to the SCFGs, unless a tree-structured design is preferred for some reason. When modeling sequences with crossing interactions, this capability stands out more prominently. Although the SCSGs can represent sequences with crossing

[6] This means that none of the production rules decrease the length of the symbol string [51].

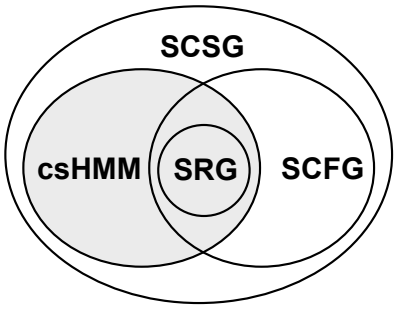


Figure 2.19: The csHMM in the Chomsky hierarchy.

interactions, they cannot directly generate the crossing interactions in the symbol sequence. For example, when modeling the *copy language*, the crossing dependencies between symbol pairs cannot be directly generated [25]. Instead, the grammar generates the two related non-terminals in a non-crossing manner, and applies the context-sensitive re-ordering rules later on, in order to obtain the final sequence that has crossing correlations. For this reason, context-sensitive grammars can be quite complex even for simple languages.

2.9 Conclusion

In this chapter, we have introduced the idea of context-sensitive HMMs. They can be viewed as an extension of the traditional HMM, where some of the states are equipped with auxiliary memory. Symbols that are emitted at certain states are stored in this memory, and the stored data serves as the context of the system, which affects the emission probabilities and the transition probabilities of the model. In this way, we can represent long-range interactions between distant symbols, which cannot be done using traditional HMMs. The csHMM is a very efficient tool for modeling sequences with complex dependencies, and it can be used as a good alternative to stochastic grammars such as the SCFG and the SCSG. We also proposed efficient polynomial-time algorithms for finding the optimal state sequence and for computing the probability of an observed symbol string. These algorithms can be used for solving the alignment problem and the scoring problem of context-sensitive HMMs with nested interactions. Furthermore, a parameter re-estimation algorithm has been introduced, which can be used for training a csHMM based on a number of training sequences. The proposed model has an interesting application in Bioinformatics, especially in RNA sequence analysis [123, 125, 126, 129, 130, 132]. This will be discussed in Chapter 3.

Chapter 3

RNA Sequence Analysis Using Context-Sensitive HMMs

The *central dogma* of molecular biology states that the genetic information flows from DNA to RNA to protein. This dogma has exerted a substantial influence on our understanding of the genetic activities in the cells. Under this influence, the prevailing assumption until the recent past was that genes are basically repositories for protein coding information, and proteins are responsible for most of the important biological functions in all cells. In the meanwhile, the importance of RNAs has remained rather obscure, and the RNA was mainly viewed as a passive intermediary that bridges the gap between DNA and protein. Except for classic examples such as *tRNAs* (transfer RNAs) and *rRNAs* (ribosomal RNAs), functional noncoding RNAs were considered to be rare.

However, this view has experienced a dramatic change during the last decade, as systematic screening of various genomes identified myriads of *noncoding RNAs (ncRNAs)*, which are RNA molecules that function without being translated into proteins [30, 44]. It has been realized that many ncRNAs play important roles in various biological processes. As RNAs can interact with other RNAs and DNAs in a sequence-specific manner, they are especially useful in tasks that require highly specific nucleotide recognition [30]. Good examples are the *miRNAs* (microRNAs) that regulate gene expression by targeting *mRNAs* (messenger RNAs) [5, 52] , and the *siRNAs* (small interfering RNAs) that take part in the *RNAi* (RNA interference) pathways for gene silencing [39, 68, 69]. Recent developments show that ncRNAs are extensively involved in many gene regulatory mechanisms [34, 46].

The roles of ncRNAs known to this day are truly diverse. These include transcription and translation control, chromosome replication, RNA processing and modification, and protein degradation and translocation [44], just to name a few. These days, it is even claimed that ncRNAs dom-

inate the genomic output of the higher organisms such as mammals, and it is being suggested that the greater portion of their genome (which does not encode proteins) is dedicated to the control and regulation of cell development [67].

As more and more evidence piles up, greater attention is paid to ncRNAs, which have been neglected for a long time.[1] Researchers began to realize that the vast majority of the genome that was regarded as "junk," mainly because it was not well understood, may indeed hold the key for the best kept secrets in life, such as the mechanism of alternative splicing, the control of epigenetic variations [67]. The complete range and extent of the role of ncRNAs are not so obvious at this point, but it is certain that a comprehensive understanding of cellular processes is not possible without understanding the functions of ncRNAs [119].

Although several systematic searches for ncRNAs in recent years have unveiled a large number of novel ncRNAs, it is believed that there are still numerous ncRNAs that are waiting to be discovered [30, 44, 67]. Typical estimates of the number of ncRNAs in the human genome are in the order of tens of thousands [67, 120], but the present genome annotation on ncRNAs is too incomplete to derive a more accurate estimate. As a result of several genome sequencing projects, including the human genome project that has been completed very recently [105], a huge amount of genomic data is publicly available these days. Given the vast amount of genomic data, it is practically impossible to identify all ncRNAs solely by experimental means. In order to expedite the annotation process, we desperately need the help of computational methods that can be used for identifying novel ncRNAs.

In this chapter, we describe how the context-sensitive HMMs, which were proposed in Chapter 2, can be used for the computational identification and analysis of ncRNAs. We focus on how to build probabilistic representations of RNA families based on csHMMs, and show how they can be used to identify new ncRNA genes, which are portions of DNA that give rise to ncRNA transcripts. The main emphasis of the discussion lies on the method that can be used for finding new members (or homologues) of known ncRNA families.

The content of this chapter is mainly drawn from [130, 126, 132] and portions of it have been presented in [123, 125].

[1] In 2006, the Nobel prize in physiology or medicine was awarded to A. Z. Fire and C. C. Mello for their discovery of "RNA interference (RNAi)–the gene silencing mechanism by double-stranded RNA (dsRNA)" [39].

3.1 Outline

This chapter is organized as follows. In Section 3.2, we show that RNA secondary structures play important roles in carrying-out the functions of ncRNAs, hence many ncRNAs have well conserved secondary structures.

In Section 3.3, we consider the problem of searching for homologous RNAs. We give an overview of sequence-based homology search methods in Section 3.3.1. In Section 3.3.2, we show that in order to build an RNA homology search tool with a good prediction accuracy, we need more powerful statistical models that can reasonably combine the contributions from the structural similarity as well as the sequence similarity.

In Section 3.4, we show how the context-sensitive HMMs (csHMMs) introduced in Chapter 2 can be utilized in an RNA homology search. In Section 3.4.1, we first give several examples of csHMMs that represent various RNA secondary structures. A database search algorithm that can be used with csHMMs is introduced in Section 3.4.2. Experimental results are given in Section 3.4.3, which demonstrate the effectiveness of the csHMM-based search method.

RNAs with alternative secondary structures are considered in Section 3.5. We show in Section 3.5.1, how we can use csHMMs to represent the base correlations in RNAs with alternative folding. In Section 3.5.2, we show experimental results which indicate that the proposed approach can effectively discriminate between the RNAs that can alternatively fold and the RNAs that cannot, at a low computational cost.

In Section 3.6, we briefly mention the problem of identifying novel ncRNAs, and we conclude the chapter in Section 3.7.

3.2 RNA secondary structure

As we have shown in Section 1.5, functional RNAs typically fold intramolecularly to form characteristic RNA secondary structures. RNA secondary structures are known to play crucial roles in carrying out the functions of many ncRNAs. An intriguing example can be observed in *riboswitches*, which are regulatory RNA elements that have been recently found [65, 108]. Riboswitches are highly structured RNA domains that are found in the noncoding regions of various mRNAs. They make structural changes upon binding specific metabolites, thereby regulating the expression of the corresponding genes. Two common mechanisms of riboswitches in bacteria are illustrated in Fig-

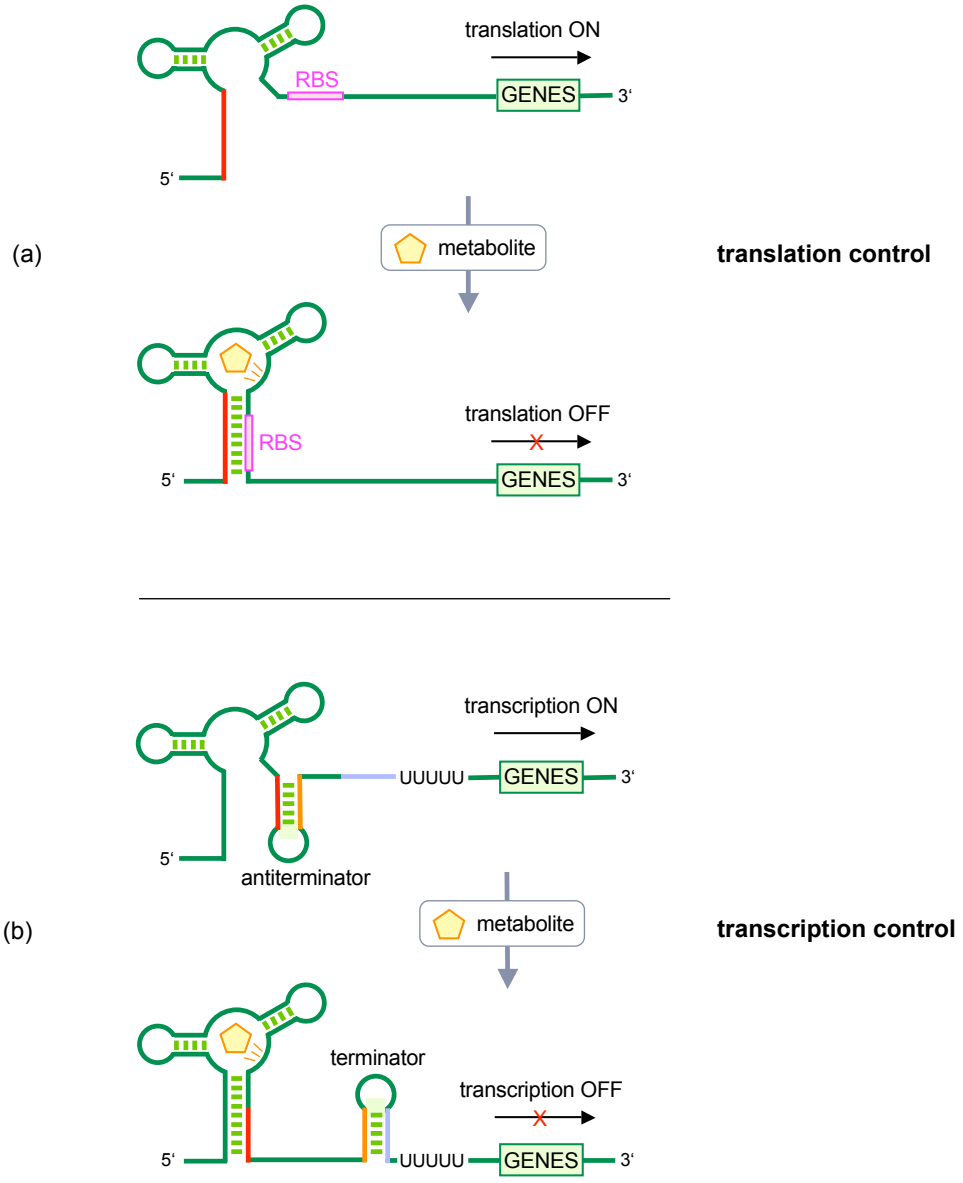


Figure 3.1: Two common mechanisms of riboswitches in bacteria. (a) *Translation control*. In the presence of the effector metabolite, the riboswitch changes its structure and sequesters the ribosome-binding site (RBS). This inhibits the translation initiation, thereby down-regulating the gene. (b) *Transcription control*. Upon binding the metabolite, the riboswitch forms a terminator stem, which prevents the generation of the full-size mRNA.

ure 3.1. The first mechanism works by *translation control* as shown in Figure 3.1 (a). In the presence of the effector metabolite, the riboswitch changes its conformation by binding it. This structural change sequesters the *ribosome-binding site* (RBS), which prevents the ribosome from binding to the mRNA. The second mechanism is based on *transcription control*. In this case, the riboswitch forms a terminator stem upon binding the metabolite. This causes a premature termination of transcription, preventing the synthesis of the full-size mRNA. Riboswitches play pivotal roles in regulating several metabolic pathways, and they are prevalent in bacteria [65, 108]. Recent results show that similar metabolite-binding RNA domains are also present in eukaryotes (organisms with a cell nucleus) such as plants and fungi, although their gene-control mechanisms may be different from those in bacteria [102].

As we can see in this example, the structure of an RNA molecule is closely related to its biological function. As a result, many ncRNAs conserve their secondary structures as well as their primary sequences. This property can be utilized when searching for ncRNA genes to improve the prediction performance. In the following section, we consider the problem of performing an RNA homology search in more detail.

3.3 Searching for homologous RNAs

Similar to protein-coding genes, ncRNA sequences can also be grouped into families of related sequences [47]. Sequences that belong to the same family perform similar functions (or functions that are related in certain ways) in the cellular mechanism. Typically, individual sequences in the family share one or more common statistical features with other sequences that belong to the same family. Such sequences are said to be homologous to each other, hence called *homologues*. Given a new sequence, we can take advantage of these family-specific characteristics to determine whether it belongs to a specific sequence family. Its membership in a certain family can often be used to infer the function of the sequence.

In fact, many computational methods for biological sequence analysis make use of the above idea in one way or another [25], especially those used for gene identification. Suppose we have a set of related sequences that belong to the same family (e.g., tRNAs). Based on these sequences, we can extract the common features of the sequence family, and use them to search the database in order to find new sequences (e.g., novel tRNAs) that share these features. Such computational screening may identify new members of a known sequence family, in a fast and efficient manner.

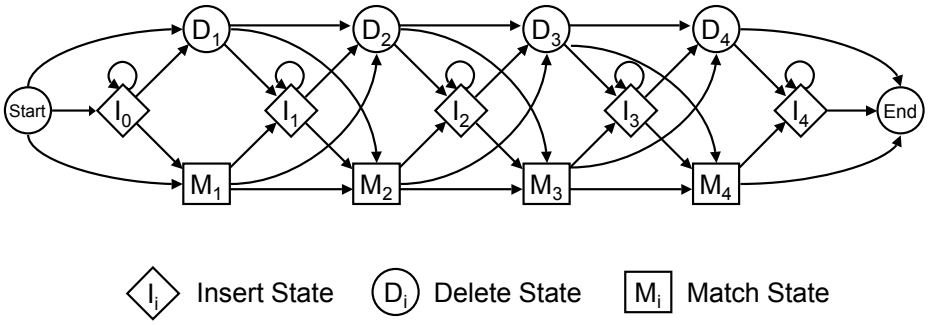


Figure 3.2: An example of a profile-HMM. It repetitively uses a set of match, insert, and delete states to model each position in a multiple sequence alignment.

This approach is typically called *homology search* (or *similarity search*).

3.3.1 Sequence-based homology search

Most of the search methods that have been used for finding homologous protein-coding genes have been based on *sequence similarity*. Popular search algorithms such as *BLAST (Basic Local Alignment Search Tool)* [2] and *FASTA* [75] use known members in a sequence family to look for high-scoring local alignments in the target database. Another approach picks up common "patterns" or "motifs" in a set of related sequences and searches the database for regions that match these patterns. One example of such an approach is the *PROSITE database* [4], which has compiled biologically significant patterns of protein families. A more general approach would be to build a *probabilistic representation* of an entire sequence family and employ it in the search.

One of the most popular models for constructing such a representation is the *profile-HMM (profile hidden Markov model)* [25, 29], which is an HMM with a linear structure that repetitively uses a set of three states (*match*, *insert*, *delete*). They can effectively describe distinct symbol probabilities at different locations and easily deal with additional insertions and deletions at any location. An example of a profile-HMM is shown in Figure 3.2. As profile-HMMs can effectively describe distinct symbol probabilities at different locations and easily deal with additional insertions and deletions at any location, they have been widely used in several applications such as protein-coding gene identification [58] and sequence alignment [25].

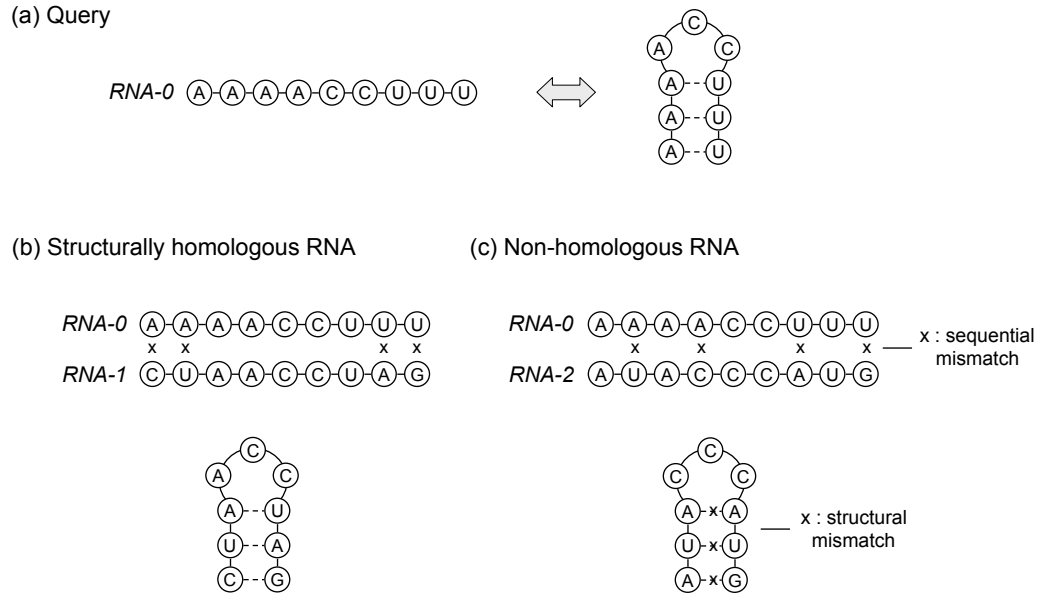


Figure 3.3: Ungapped alignment between two RNA sequences. (a) An RNA with a stem-loop structure is used as the query sequence. (b) A structurally homologous RNA that has also a stem-loop structure. (c) A structurally nonhomologous RNA that does not fold to a stem-loop structure.

3.3.2 Statistical model for RNA sequences

The sequence-based methods described in the previous section (BLAST, FASTA, PROSITE, profile-HMM) are very useful for identifying homologous DNAs and proteins, but they often behave poorly when applied to RNA homology search. The main reason is the following. As mentioned earlier, many functional ncRNAs preserve their secondary structures as well as their primary sequences [25]. Sometimes, these base paired structures are still preserved among related RNAs, even when their similarity in the primary sequence level can be hardly recognized. Therefore, when evaluating the similarity between two RNA molecules, it is important to take both their primary sequences and their secondary structures into consideration.

As observed by Eddy in [31], this combined scoring scheme is much more effective in comparing (and also aligning) RNA sequences, and it can greatly enhance the discriminative power of an RNA homology search. This can be clearly seen from the example illustrated in Figure 3.3. In this example, we have a query sequence that has a stem-loop structure. Let us perform ungapped pairwise alignments between the query sequence and each of the RNAs shown in Figure 3.3 (b) and Figure 3.3 (c). Both RNA-1 and RNA-2 differ from the query sequence RNA-0 at four locations.

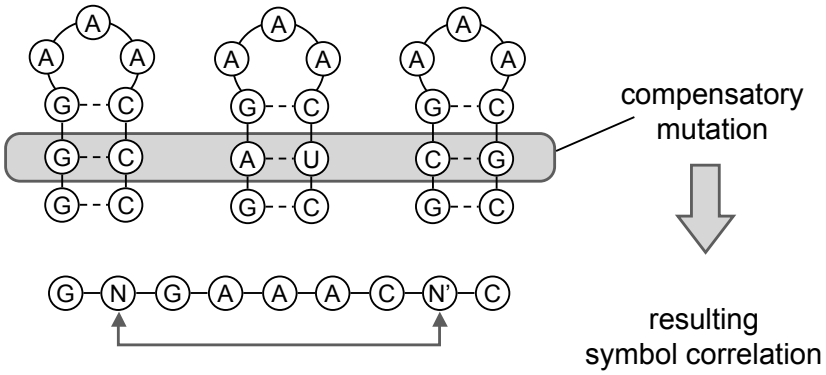


Figure 3.4: Compensatory mutations give rise to strong pairwise correlations in the primary sequence of an RNA.

As the four mismatches (or "base substitutions") in both alignments are identical, the primary sequence alignment score for RNA-1 and RNA-0 will be exactly the same as the alignment score for RNA-2 and RNA-0. However, we can see in Figure 3.3 (b) and Figure 3.3 (c) that RNA-1 preserves the secondary structure of the original query sequence, while RNA-2 does not. Apparently, RNA-1 is a better match to the query RNA-0, and therefore we should give it a higher score than RNA-2.

As this example shows, when computing a similarity measure between RNAs, it is important to consider their resemblance in the (secondary) structural level as well as in the (primary) sequence level. Now, the question is how to combine the contributions from the sequence similarity and the structural similarity in a reasonable way. To answer this question, let us examine the effect of a conserved RNA secondary structure on its primary sequence. RNA sequences often undergo *compensatory mutations* in order to preserve their secondary structures. For a given base pair in an RNA molecule, if the base in one side is changed to another base, the base in the other side is also changed such that the base pair is still maintained. As a result, we can observe strong correlation between the two base positions in homologous RNAs as illustrated in Figure 3.4. From this point of view, we can understand base pairing in an RNA secondary structure in terms of pairwise correlations between distant bases in the primary sequence of the RNA. This shows that in order to model RNAs with conserved secondary structures, we need a statistical model which can describe such pairwise correlations. However, most statistical models that have been used for analyzing DNAs and proteins (including profile-HMMs) do not have the descriptive power to deal with such complex base correlations.

RNA sequences with secondary structures can be viewed as a kind of biological *palindromes*. Palindromes are symmetric sequences that read the same forwards and backwards, such as "I prefer pi," "step on no pets," and so on. Similarly, the base pairing in an RNA secondary structure gives rise to symmetric (or *reverse complementary*, to be more precise) regions in its primary sequence that are analogous to palindromes. As we have seen in Chapter 2, HMMs can be viewed as *stochastic regular grammars* according to the *Chomsky hierarchy of transformational grammars* [16]. Regular grammars are the simplest among the four classes in the hierarchy, and it is known that they are inherently incapable of describing a palindromic language.[2] As a result, regular grammars are not suitable for constructing RNA profiles.

In order to represent complex correlations that are frequently observed in ncRNA sequences, we need more complex models with larger descriptive power than the regular grammars. One model that has been especially popular for representing RNA families is the *covariance model (CM)* [25, 26]. CMs can be viewed as *profile-SCFGs (profile stochastic context-free grammars)*, which are capable of handling nested correlations.

Another possibility is to use the context-sensitive HMMs. In fact, csHMMs can effectively describe the long-range correlations between distant bases, hence they provide a simple and intuitive way for modeling RNAs with conserved secondary structures.

In the following section, we show how the csHMMs can be used for representing RNA secondary structures and finding homologous RNAs.

3.4 Database search using csHMMs

In order to perform an RNA homology search, we first have to construct a context-sensitive HMM that closely reflects the characteristics of the RNA of interest. The constructed model can then be used for finding similar regions in a sequence database.

3.4.1 Modeling RNA secondary structures

Given a consensus secondary structure, designing a csHMM that generates sequences with the specified structure is relatively easy. In order to demonstrate this, let us consider several examples

[2] As we have mentioned in Chapter 2, it is of course possible that a regular grammar generates a palindrome as part of its language, but the point is that it is not capable of generating *only* such palindromes. Therefore, regular grammars cannot effectively discriminate palindromic sequences from nonpalindromic ones.

of RNA secondary structures. Note that the csHMM that represents a given structure is not unique. The final implementation of the model depends on the specific application.

Figure 3.5 (a) illustrates a typical *stem-loop (hairpin)* structure, which is the simplest of all RNA secondary structures. RNA sequences with a conserved stem-loop structure can be represented by a simple context-senstive HMM that is shown in Figure 3.5 (b). In this model, the pairwise-emission state P_1 and the context-sensitive state C_1 are associated with a stack, and they together generate the stem part of the structure. When we enter C_1, it retrieves a symbol x from the stack, which was previously emitted by P_1. Note that x represents a base in the RNA sequence, which takes one of the four values A, C, G, and U. After retrieving x, the context-sensitive state C_1 emits the complimentary base of x. In this way, the state-pair (P_1, C_1) can generate the stem. The single-emission state S_1 is used for generating the loop, since the bases in the loop are not correlated to other bases. If we need to model a *bulge*, which is a nonpaired base inside a stem, it can be done by adding more states to the HMM as shown in Figure 3.5 (c).

As demonstrated in this example, whenever there exist a pairwise interaction between two bases, we can represent it using a pair of a pairwise-emission state and a context-sensitive state. For unpaired bases that form a loop, a bulge, and so forth, we can use single-emission states for representing them.

Based on this principle, it is not difficult to construct context-sensitive HMMs that can represent RNA sequences with more complex secondary structures. For example, Figure 3.6 (a) depicts the consensus secondary structure of the so-called *iron response element (IRE)*. The iron response elements are found in the 5′ or 3′ UTRs (untranslated regions) of various messenger RNAs. It is known that the *iron regulatory proteins (IRPs)* bind to the IREs in order to control the iron metabolism inside the cell [54]. The IRE has a well conserved stem-loop with an *interior loop* as shown in Figure 3.6 (a). These RNAs can be modeled using the csHMM in Figure 3.6 (b).

Another example of an RNA secondary structure is illustrated in Figure 3.7 (a), which shows the typical structure of a *tRNA (transfer RNA)*. The tRNA is a short RNA molecule that usually consists of 74–93 nucleotides. It attaches a specific amino acid to the protein chain that is being synthesized, during the *translation* procedure of mRNA into protein [11]. The tRNAs have a highly conserved secondary structure with three stem-loops, which is called the *cloverleaf structure* due to its shape. As shown in Figure 3.7 (b), the cloverleaf structure can be modeled using four pairs of (P_n, C_n), where a separate stack is dedicated to each state-pair. Note the similarity between the

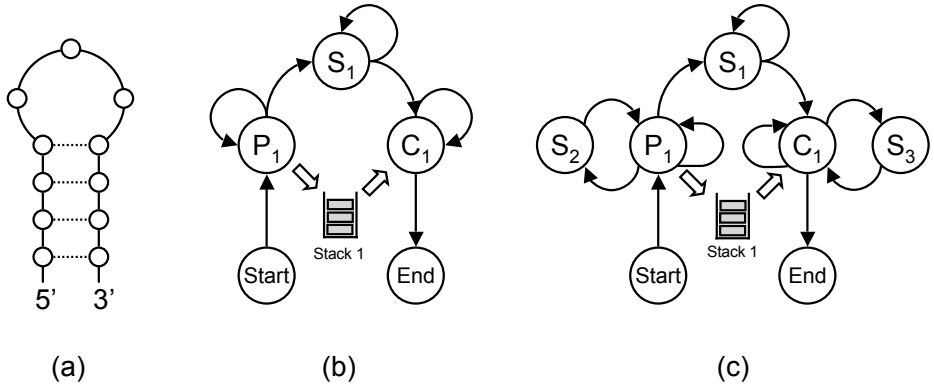


Figure 3.5: (a) A typical stem-loop. The nodes represent the bases in the RNA, and the dotted lines indicate the interactions between bases that form complementary base pairs. (b) An example of a csHMM that generates a sequence with a stem-loop structure. (c) A csHMM that models a stem-loop with bulges.

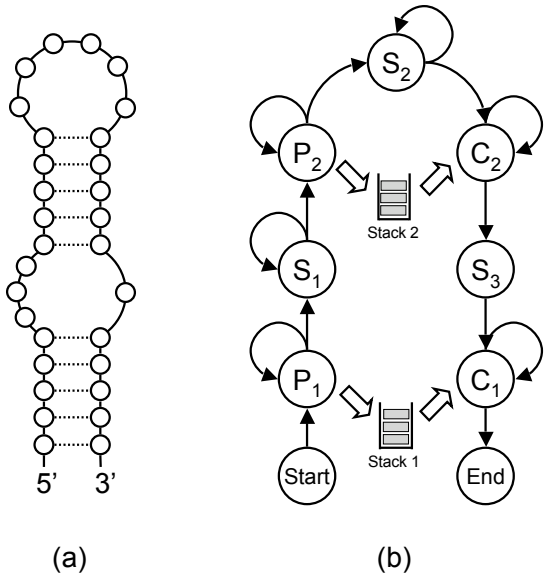


Figure 3.6: (a) A typical structure of an iron response element. (b) An example of a csHMM that generates sequences with the given secondary structure.

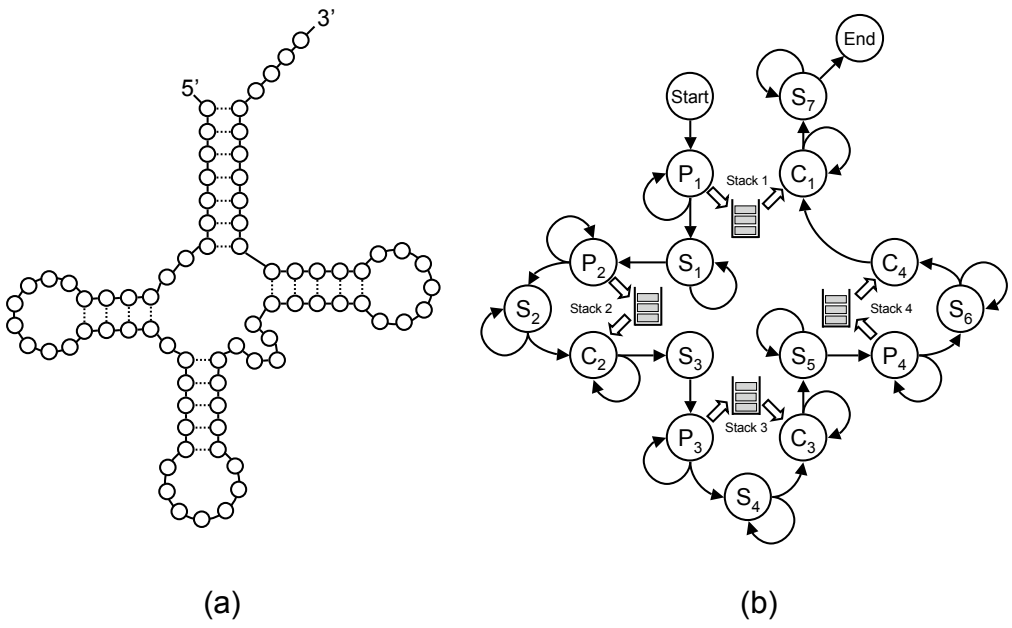


Figure 3.7: (a) A typical tRNA cloverleaf structure. (b) An example of a csHMM that can generate sequences with the cloverleaf structure.

original consensus RNA structure and the constructed context-sensitive HMM. As every state in the HMM corresponds to one or more base locations in the RNA sequence, the design procedure of context-sensitive HMMs is very simple and intuitive.

3.4.2 Database search algorithm

Once we have constructed a good model that closely represents the consensus sequence of a ncRNA family, we can use it to search the database to find similar regions that match the given csHMM reasonably well. In this section, we propose an efficient database search algorithm that can be used with csHMMs. For simplicity, we consider the case when we are looking for RNAs with a single stem-loop. The algorithm presented in this section is a variant of the simplified optimal alignment algorithm described in Appendix B. For finding RNAs with multiple stem-loops (e.g., tRNAs), we can implement a search algorithm based on the alignment algorithm described in Section 2.4, in a similar manner.

Let us first define the variables that are needed to describe the algorithm. We use similar notations as in Chapter 2. Let $\mathbf{x} = x_1 x_2 ... x_L$ be the observed symbol sequence, where L is the length

of the observation. The underlying state sequence (path) is denoted as $\mathbf{y} = y_1 y_2 \dots y_L$. We assume that there are M states in the context-sensitive HMM. M_1 is the number of state-pairs (P_n, C_n) and M_2 is defined as the number of single-emission states, hence we have $M = 2M_1 + M_2 + 2$ including the "start" and "end" states. It is assumed that all pairwise interactions between P_n and C_n occur in a nested manner (with a single nested structure) and do not cross each other. For notational convenience, we also define the following sets $\mathcal{P} = \{P_1, \dots, P_{M_1}\}$, $\mathcal{C} = \{C_1, \dots, C_{M_1}\}$, and $\mathcal{S} = \{S_1, \dots, S_{M_2}\}$. We denote the transition probability from state v to w as $t(v, w)$. The emission probability of a symbol x at a single-emission state $v \in \mathcal{S}$ or a pairwise-emission state $v \in \mathcal{P}$ is defined as $e(x|v)$. Since the emission probabilities at a context-sensitive state $v \in \mathcal{C}$ depends on the symbol x_p that was previously emitted at the corresponding pairwise-emission state, we denote the emission probability at $v \in \mathcal{C}$ as $e(x|v, x_p)$.

In the simplified alignment algorithm [127], the variable $\gamma(i, j, v, w)$ is defined as the log-probability of the optimal path among all sub-paths $y_i \cdots y_j$ with $y_i = v$ and $y_j = w$, where it is assumed that all pairwise-emission states P_n are paired with the corresponding context-sensitive states C_n inside the sub-path. In many cases, we can limit the maximum length of the ncRNA gene, which reduces the overall computational complexity and makes the search algorithm practically executable. Let us define $d = j - i + 1$ to be the length of the sub-sequence, where we restrict it to be $d \leq D$ for some D. Based on this setting, we may define either $\gamma(i, d, v, w)$ or $\gamma(j, d, v, w)$, in a similar manner. Using one of these variables instead of $\gamma(i, j, v, w)$ can minimize the memory requirement as well. Depending on which variable we use, there exist two different schemes for computing these variables iteratively. This is illustrated in Figure 3.8.

In the following algorithm, we use the variable $\gamma(j, d, v, w)$, hence adopting the update scheme in Figure 3.8 (b), which is similar to the scheme elaborated in [25]. Now, the database search algorithm can be defined as follows.

Database search algorithm

For $j = 1, \dots, L, d = 1, \dots, \min(D, j)$ and $v = 1, \dots, M, w = 1, \dots, M$.

(i) $d = 1$ $(v = w)$

$$\gamma(j, d, v, v) = \begin{cases} \log e(x_i|v) & v \notin \mathcal{P}, \mathcal{C} \\ -\infty & \text{otherwise} \end{cases}$$

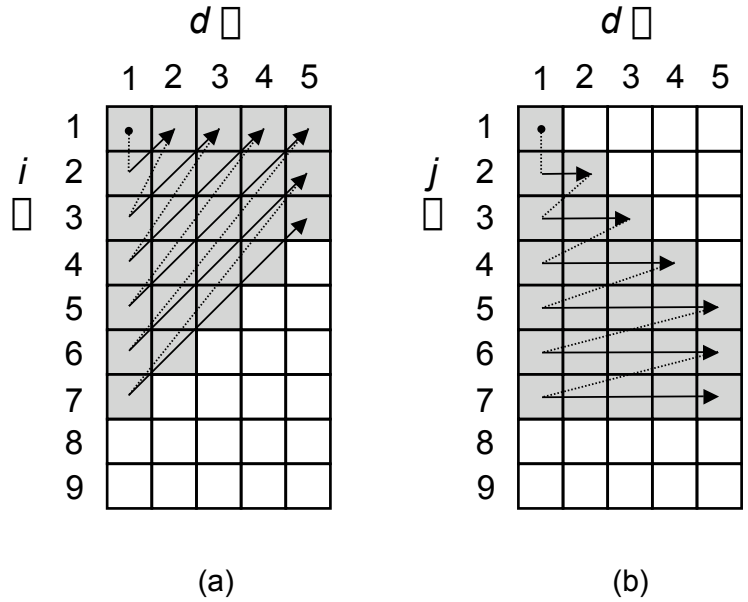


Figure 3.8: Two different update schemes. (a) When using the variable $\gamma(i, d, v, w)$. (b) When using the variable $\gamma(j, d, v, w)$.

(ii) $d = 1$ $(v \neq w)$

$$\gamma(j, d, v, w) = -\infty$$

(iii) $v = P_n, w = C_m (n \neq m)$, or $v \in \mathcal{C}$, or $w \in \mathcal{P}$

$$\gamma(j, d, v, w) = -\infty$$

(iv) $v = P_n, w = C_n, d = 2$

$$\gamma(j, d, v, w) = \log e(x_{j-1}|v) + \log t(v, w) + \log e(x_j|w, x_{j-1})$$

(v) $v = P_n, w = C_n, d > 2$

$$\begin{aligned}\gamma(j, d, v, w) \quad = \quad & \max_{u_1, u_2} \Big[\log e(x_{j-d+1}|v) + \log t(v, u_1) + \gamma(j-1, d-2, u_1, u_2) \\ & + \log t(u_2, w) + \log e(x_j|w, x_{j-d+1}) \Big]\end{aligned}$$

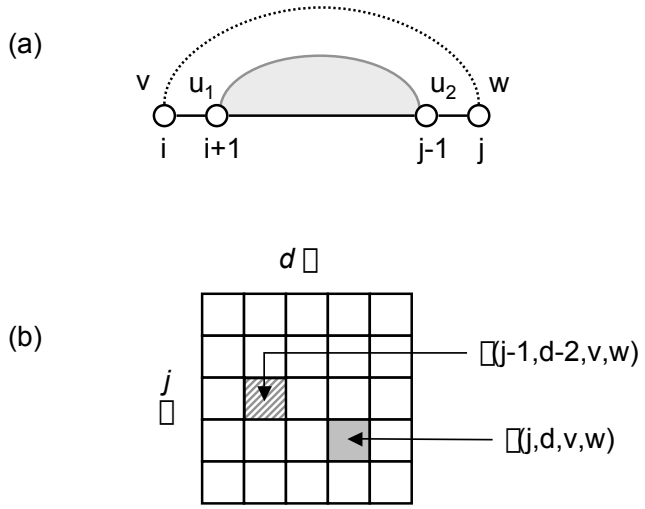


Figure 3.9: Illustration of step (v).

(vi) $v \in \mathcal{P}, w \notin \mathcal{C}$

$$\gamma(j,d,v,w) = \max_u \Big[\gamma(j-1,d-1,v,u) + \log t(u,w) + \log e(x_j|w)\Big]$$

(vii) $v \notin \mathcal{P}, w \in \mathcal{C}$

$$\gamma(j,d,v,w) = \max_u \Big[\log e(x_{j-d+1}|v) + \log t(v,u) + \gamma(j,d-1,u,w)\Big]$$

(viii) $v \notin \mathcal{P}, w \notin \mathcal{C}$

In this case, $\gamma(j,d,v,w)$ can be updated using either the update formula (vi) or (vii). ■

As we can see from above, the log-probability $\gamma(j,d,v,w)$ is computed in an iterative manner, starting from a shorter sequence and extending it progressively. Whenever there exist pairwise correlations between symbols, the emission of these symbols are considered at the same time. As mentioned earlier, when computing $\gamma(j,d,v,w)$, we consider only those paths, where all the P_n states are paired with the corresponding C_n states inside the path. Therefore, the log-probability $\gamma(j,d,v,w)$ is set to $-\infty$, whenever P_n and C_n do not form pairs. For example, in case (iii), when the leftmost state is a context-sensitive state $y_i\,(= y_{j-d+1}) \in \mathcal{C}$, it cannot be paired with the corresponding pairwise-emission state, since there are no more states to the left of y_i. This is also true

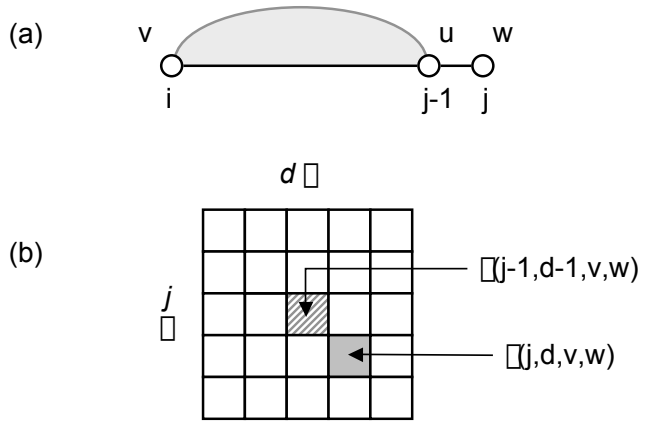


Figure 3.10: Illustration of step (vi).

when the rightmost state is a pairwise-emission state $y_j \in \mathcal{P}$.

Now, let us consider the case when $v = P_n$ and $w = C_n$. When $d = 2$, we can simply compute $\gamma(j, d, v, w)$ as in (iv), by considering the emission of $x_i (= x_{j-d+1})$ and x_j together. When $d > 2$, we can compute $\gamma(j, d, v, w)$ as follows. Since y_i has to form a pair with y_j as shown in Figure 3.9 (a) by the dotted line, the pairwise-emission states and the corresponding context-sensitive states inside $y_{i+1} \dots y_{j-1}$ have to exist in pairs. This is indicated by the shaded region in Figure 3.9 (a). As the log-probability of the optimal path for $y_{i+1} \dots y_{j-1}$ is already stored in $\gamma(j-1, d-2, u_1, u_2)$, we can compute $\gamma(j, d, v, w)$ by extending $\gamma(j-1, d-2, u_1, u_2)$ as shown in (v).

Figure 3.10 illustrates the case when $v \in \mathcal{P}$ and $w \notin \mathcal{C}$. Since there can be no interaction between y_j and any other state y_k $(i \le k \le j-1)$, all the pairwise-emission states and the context-sensitive states inside $y_i \dots y_{j-1}$ should exist in pairs. Therefore $\gamma(j, d, v, w)$ can be computed by extending $\gamma(j-1, d-1, v, u)$ to the right by one symbol, as described in step (vi) of the algorithm. Similarly, when $v \notin \mathcal{P}$ and $w \in \mathcal{C}$ as in Figure 3.11, we can compute $\gamma(j, d, v, w)$ based on $\gamma(j, d-1, u, w)$ as described in (vii).

Careful examination of the search algorithm shows that its computational complexity is

$$O(LDM_1M^2) + O(LDM_2^2M), \tag{3.1}$$

which grows linearly with the length L of the entire sequence, or the size of the database. If we do not limit the maximum length to be D, the complexity will be $O(L^2M_1M^2) + O(L^2M_2^2M)$, which is not a linear function of L any more.

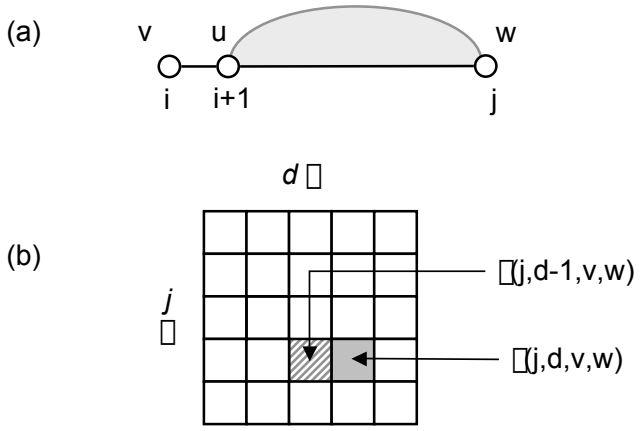


Figure 3.11: Illustration of step (vii).

3.4.3 Predicting iron response elements

In order to demonstrate the effectiveness of the proposed search algorithm, we constructed a csHMM that can be used for finding the regions in a given DNA sequence, which are transcribed into *iron response elements (IREs)*. IREs are found in the 5′ or 3′ UTRs of various messenger RNAs. It is known that the *iron regulatory proteins (IRPs)* bind to the IREs in order to control the iron metabolism within cells [54]. The IRE has a well conserved hairpin structure that has either an interior loop or a bulge. The consensus secondary structures of the IREs are shown in Figure 3.12. As shown in this figure, bases at certain positions are especially well conserved in the IREs. For example, the IREs have a loop that consists of six bases, which has the pattern "CAGWGH." In this pattern, W can be either A or U (T), and H can be either A, C or U (T). There can be noncanonical base pairs in the stem such as the GU/UG pairs, and the lower stem can be of variable length.

We used the context-sensitive HMM in Figure 3.13 to represent the IREs with conserved secondary structures. The lower stem is modeled using the pairwise-emission state P_1 and the context-sensitive state C_1. This state-pair is associated with a stack as shown in Figure 3.13. Similarly, the upper stem is modeled by the state-pair (P_2, C_2), which uses a separate stack. Note that (P_1, C_1) and (P_2, C_2) are capable of representing stems of variable lengths. The loop and the bulge are modeled using single-emission states, since the bases in these parts are not correlated to other bases. Each single-emission state S_n uses a different set of emission probabilities, in order to specify which base is conserved at each location.

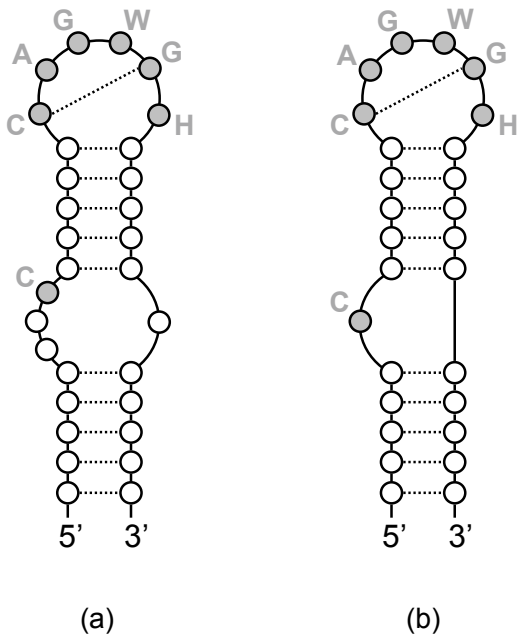


Figure 3.12: The consensus secondary structures of the iron response elements (IREs).

Based on this context-sensitive HMM, we used the database search algorithm elaborated in Section 3.4.2 to find IREs in several DNA sequences. We chose four DNA sequences in the human genome that are known to contain functional IREs. These sequences have been previously used for testing the performance of *PatSearch*, a pattern matching program that can find functional elements with various patterns in DNA or protein sequences [77]. We ran the search algorithm for finding high-scoring regions in the database. When there were overlaps between several high-scoring regions, only the one with the highest score was stored as a match. The search results are summarized in Table 3.1. The first column in in Table 3.1 shows the EMBL accession number and the second column shows the UTRdb ID of each DNA sequence [78]. The search results of the csHMM-based IRE finder are shown in the third column. The fourth column contains the prediction results of the PatSearch program. As summarized in Table 3.1, the csHMM-based IRE finder was able to find all the IREs in the given DNA sequences, and there were no false predictions. Interestingly enough, the csHMM-based approach was able to identify the start position and the end position of the IREs more precisely than the PatSearch. For example, in the second DNA sequence (accession number: Y09188), the proposed method predicted that there exists an IRE between 6 and 31, which matches the data in [78] and the Rfam database [47] exactly. Similarly, in the third sequence (accession num-

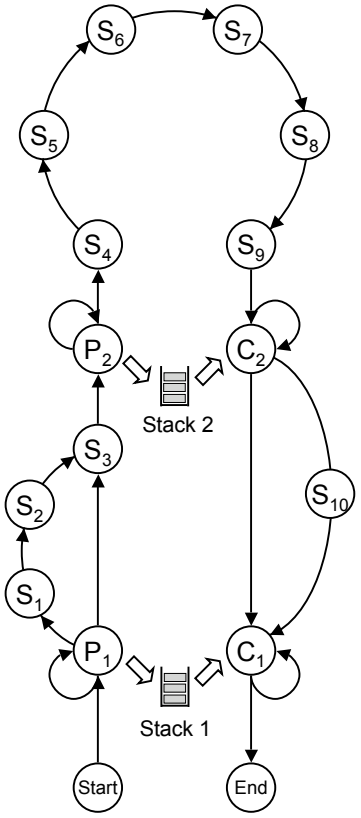


Figure 3.13: A csHMM that represents the IREs.

EMBL AC	UTRdb ID	csHMM	PatSearch
X60364	5HSA001988	13–35	13–35
Y09188	5HSA003829	6–31	8–30
D28463	5HSA003858	32–59	35–57
J04755	5HSA013930	951–978	34–56

Table 3.1: The database search result for finding IREs.

ber: D28463), the csHMM-based method predicted the location of the IRE to be between 32 and 59, which matches the data in the Rfam database. In the fourth sequence (accession number: J04755), the predicted location of the csHMM-based method was identical to the location stored in the Rfam database. In [77], it is reported that the PatSearch software has predicted the location of the IRE to be between 34 and 56, which is completely different from the true location. Since the sequence between 34 and 56 does not match the typical pattern of the IREs, the wrong position reported in [77] is probably a simple typo.

3.5 Identification of RNAs with alternative folding

Recent research on gene regulation has revealed that many ncRNAs are actively involved in controlling various genetic networks [46]. These regulatory RNAs include *microRNAs (miRNAs)* [5], *riboregulators* [34], and *riboswitches* [108].

As we have seen in the previous sections, many functional ncRNAs have well conserved secondary structures, as these structures are crucial in carrying out their biological functions. Typically, an RNA sequence adopts a single "biologically correct" secondary structure. However, there exist also examples of RNAs that can choose from alternative structures, thereby changing their characteristics. In fact, many regulatory RNAs can make conformational changes depending on one or more environmental cues to regulate the expression level of certain genes [53, 108]. *Riboswitches*, which were considered in Section 3.2 (see Figure 3.1), are good examples of such RNAs [108]. They are highly structured RNAs that are usually found in the 5′ untranslated regions (UTRs) of certain mRNAs. Riboswitches change their secondary structures upon binding to specific metabolites, thereby controlling the expression of the corresponding metabolic genes.

In addition to natural RNAs with differential folding, there are also engineered RNAs that can be used for controlling gene expression based on a similar mechanism. For example, the *antiswitch* designed by Bayer and Smolke is an RNA-based regulator that can directly control the expression of a target transcript in a ligand-dependent manner [7]. Figure 3.14 illustrates the general mechanism of an antiswitch regulator. When the effector ligand is absent, the antisense domain in the antiswitch (which is complementary to the target mRNA transcript) is sequestered. As the antiswitch cannot bind to the target, the gene expression of the target is turned on. In the presence of a specific ligand, the antiswitch binds to the ligand, resulting in a change in its secondary structure as shown in Figure 3.14 (b). This conformational change releases the antisense domain, thereby allowing the antiswitch to bind to the target mRNA, which is illustrated in Figure 3.14 (c). As a result, the antiswitch regulator will suppress the expression of the target gene.

3.5.1 Modeling alternative structures

The existence of alternative secondary structures introduce complex correlations in the primary sequence of the RNA. As an example, let us consider the primary sequence of the antiswitch shown in Fig 3.15. In the absence of the target ligand, region 1 (the antisense domain) forms base pairs with region 2. Therefore, there exist correlations between bases in region 1 and those in region 2.

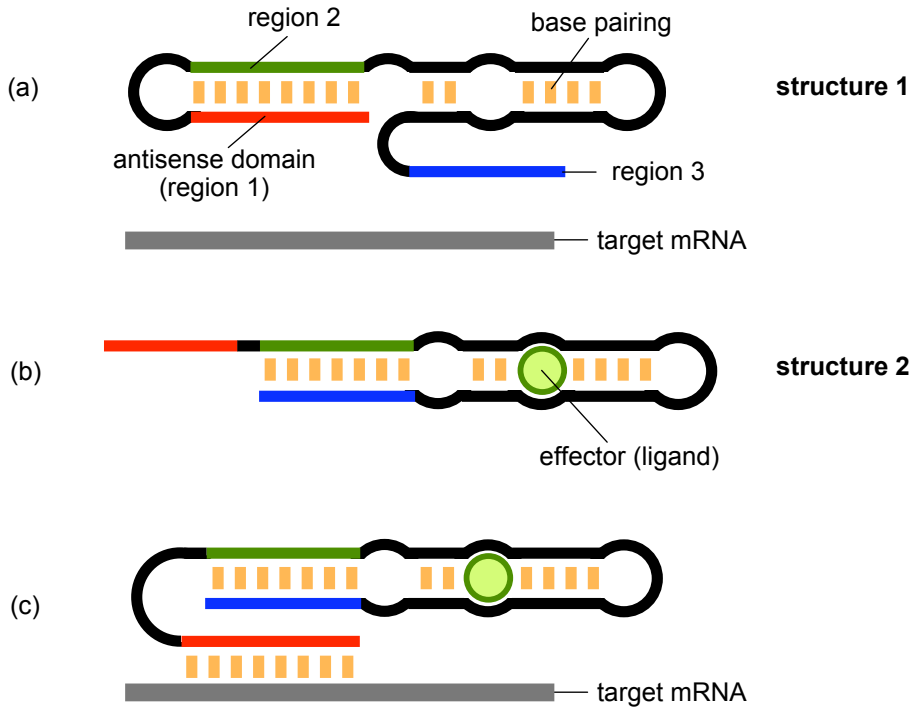


Figure 3.14: An antiswitch regulator. (a) Secondary structure of the antiswitch in the absence of ligand. (b) The structure changes upon binding the ligand. (c) In the presence of ligand, the antiswitch can bind to the target mRNA, suppressing its expression.

When the target ligand is present, region 3 can fold onto region 2, hence there exist also correlations between these two regions. As a result, the bases in region 2 are correlated to the bases in region 1 and also to the bases in region 3. The overall base correlations are depicted in Figure 3.15 (a), where the arcs indicate the correlations between bases. Such correlations cannot be modeled using a SCFG [25] or a csHMM [131], and we have to resort to more general grammars such as *context-sensitive grammars (CSGs)*. However, CSGs that can represent sequences with correlations shown in Figure 3.15 (a) tend to get very complex. Moreover, parsing CSGs is an NP-complete problem, and there is no polynomial-time algorithm that can be used in general [25, 41].

However, we can circumvent these difficulties by adopting the following strategy. Instead of modeling the overall correlations in the RNA sequence by a single model, we can use multiple csHMMs to represent the respective correlations that arise from each of the alternative RNA secondary structures. For example, we can use a csHMM to describe the correlations that arise from "structure 1" (shown in Figure 3.15 (b)) and use another csHMM to describe the correlations that arise

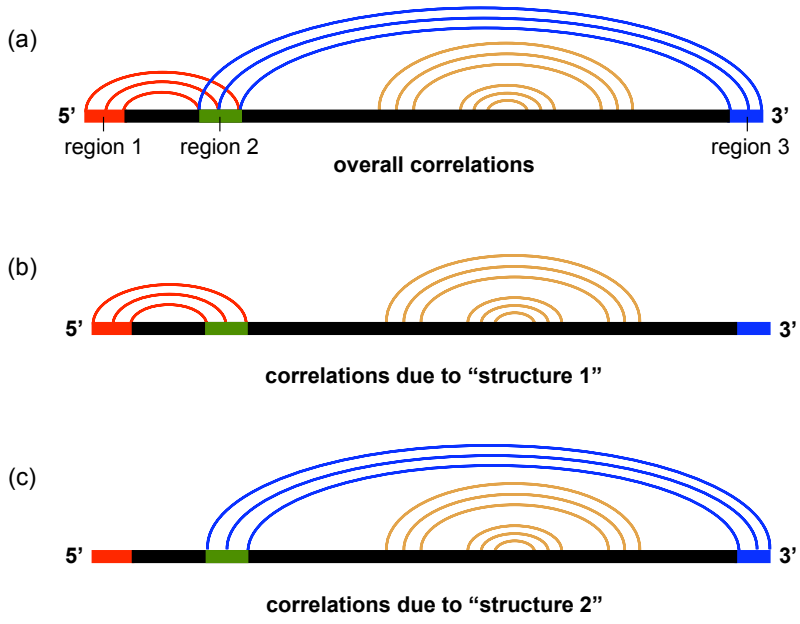


Figure 3.15: Base correlations in the primary sequence of an antiswitch. (a) Overall correlations. (b) Correlations due to structure 1 (in the absence of ligand). (c) Correlations due to structure 2 (in the presence of ligand).

from "structure 2" (shown in Figure 3.15 (c)). Given a novel RNA sequence, we can score it based on the two csHMMs using an efficient polynomial-time algorithm (introduced in Appendix B [128]. We can combine the two scores to determine how close the given RNA sequence is to the original RNA sequence modeled by the csHMMs.

3.5.2 Experimental results

To demonstrate the efficacy of the proposed method, we constructed two csHMMs as shown in Figure 3.16. The csHMM in Figure 3.16 (a) models the correlations that arise from "structure 1," which are shown in Figure 3.15 (b). Note that S_n is a single-emission state, P_n is a pairwise-emission state, and C_n is the context-sensitive state that corresponds to P_n. The states P_1 and C_1 work together to generate the antisense stem that sequesters the antisense domain in the absence of ligand. Similarly, the state-pairs (P_2, C_2) and (P_3, C_3) generate the other stems in "structure 1." Likewise, the csHMM illustrated in Figure 3.16 (b) represents the correlations shown in Figure 3.15 (c), which

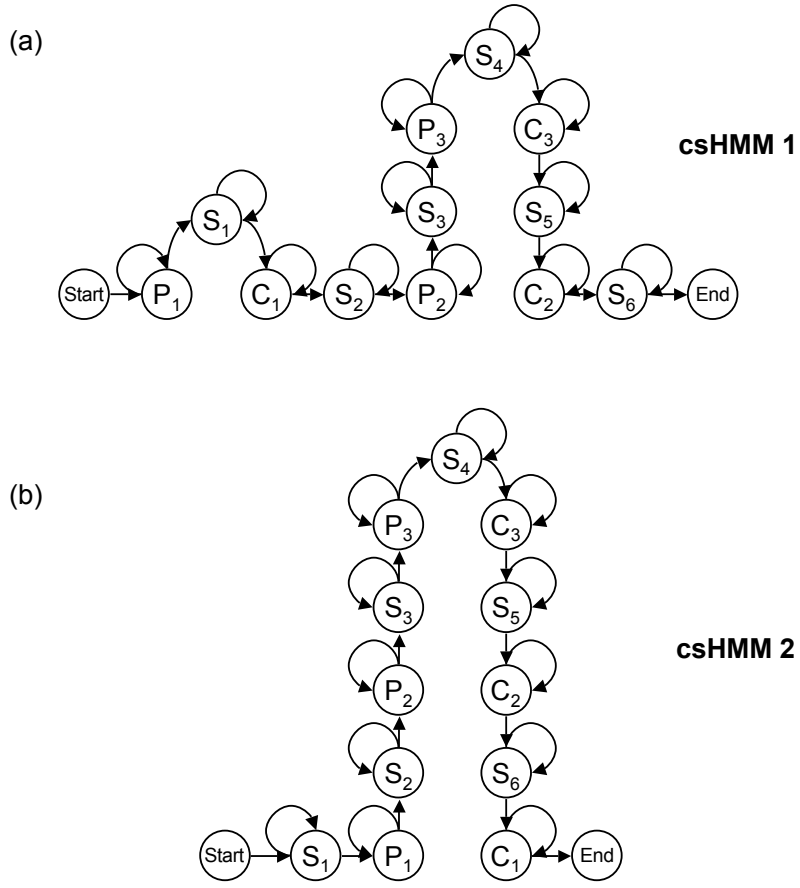


Figure 3.16: (a) The csHMM that represents structure 1. (b) The csHMM that represents structure 2.

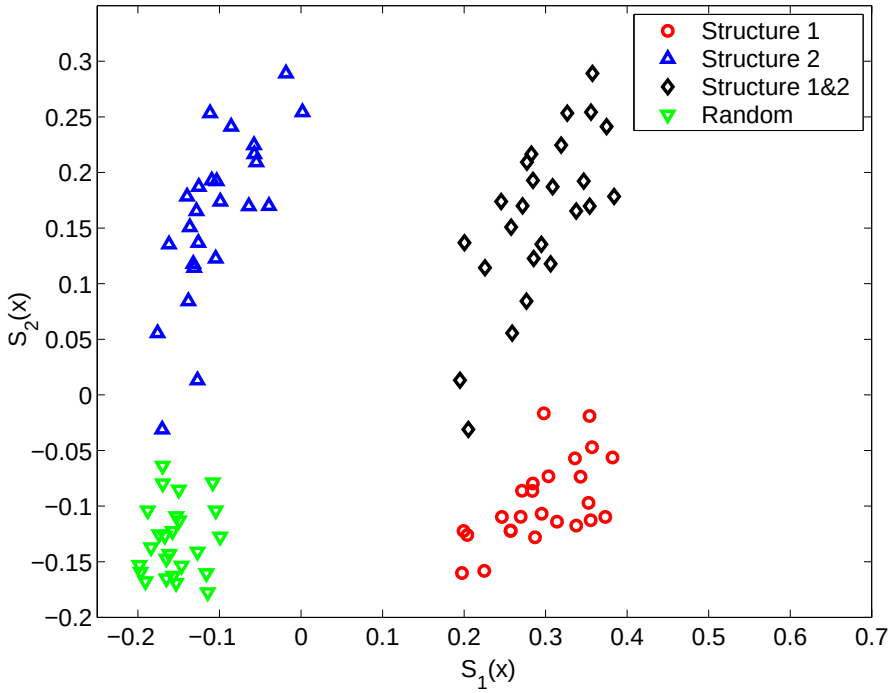


Figure 3.17: Plot of $(S_1(\mathbf{x}), S_2(\mathbf{x}))$.

arise from "structure 2." Based on these csHMMs, we can compute

$$S_1(\mathbf{x}) = \frac{1}{L} \log_2 \frac{P(\mathbf{x}|\text{csHMM1})}{P(\mathbf{x}|H_0)},$$

and

$$S_2(\mathbf{x}) = \frac{1}{L} \log_2 \frac{P(\mathbf{x}|\text{csHMM2})}{P(\mathbf{x}|H_0)},$$

for a given RNA sequence $\mathbf{x}$, where L is the length of $\mathbf{x}$ and H_0 is the random model with i.i.d. assumption. Figure 3.17 shows a plot of $(S_1(\mathbf{x}), S_2(\mathbf{x}))$ for 100 test sequences $\mathbf{x}$. As we can see in Figure 3.17, sequences with alternative structures (depicted by black diamonds) are well separated from sequences with either "structure 1" or "structure 2," or unstructured sequences. Therefore, given an RNA sequence $\mathbf{x}$, we can first compute the two similarity scores $S_1(\mathbf{x})$ and $S_2(\mathbf{x})$ and then use a classification algorithm (e.g., a *support vector machine* [12]) to decide whether $\mathbf{x}$ is a true homologue with alternative folding or not.

3.6 Beyond homology search: Identifying novel ncRNAs

The main focus of this chapter was to propose methods that can be used for identifying new members (homologues) of known ncRNA families. Finding novel ncRNAs that have not been identified yet is a much more challenging task, and we briefly discuss this problem in this section.

Until now, various signal processing techniques have been applied to the prediction of protein-coding genes, which include DFT [3, 106], digital filters [112, 113], hidden Markov models (HMMs) [28, 29, 59], and many others. Among them, HMM-based methods have been especially successful. State-of-the-art gene finders (primarily based on HMMs) boast high prediction ratios that are far above 90%, achieving nearly perfect prediction results in simple organisms such as bacteria and yeast.

However, these methods are not suitable for predicting ncRNA genes due to the following reasons. First of all, many ncRNAs lack the various statistical cues that have been used for identifying protein-coding genes. Unlike coding genes, their primary sequences do not display strong composition bias with strength comparable to the codon bias in protein-coding genes [31]. They do not have *open reading frames (ORFs)*[3] that were effectively used in coding-gene finders [71]. Moreover, many ncRNAs are considerably shorter than coding genes, where a typical ncRNA has less than a few hundred nucleotides [47]. (An extreme example is the miRNA which has only about 21-25 nucleotides, in general [93].) This makes it difficult to judge whether the statistical property inside the ncRNA genes is different from that of the rest in a statistically meaningful manner.

Although traditional protein-coding gene finders cannot be directly used for identifying novel ncRNA genes, we can utilize the native characteristics of RNAs for building ncRNA gene finders. For example, as many ncRNAs have well conserved secondary structures, we can exploit this property for finding ncRNA genes. However, an RNA sequence can have a large number of thermodynamically plausible secondary structures that have no biological significance [32]. In fact, it has been realized that the existence of a plausible secondary structure is not a sufficient evidence for detecting ncRNAs [87]. What is more important is whether the given secondary structure is preserved across different species, which can serve as a compelling evidence of its biological significance. For this reason, most ncRNA gene-prediction algorithms take advantage of multiple sequence data for finding novel ncRNAs [18, 21, 88, 119].

[3] An ORF is any sequence of DNA that can potentially encode a protein. It starts with a start codon and ends with a stop codon [11]. Usually, the existence of a long ORF is a reasonable indication of the presence of a protein-coding gene.

A common strategy of many general purpose ncRNA gene finders–such as *QRNA* [88], *ddbRNA* [21], *MSARI* [18], and *RNAz* [119]–can be summarized as follows [71]. They first look for regions in genome sequences that are conserved across different species, and form a multiple sequence alignment between these regions. Based on the alignment, they investigate whether there exists a common secondary structure that is preserved in all sequences. This information is used to decide whether these regions correspond to a functional ncRNA or not. Some of these algorithms have been used for screening the genomes of several organisms, and the detection results indicate that the aforementioned strategy is indeed quite effective. For example, RNAz - which is the current state-of-the-art algorithm for predicting novel ncRNAs - achieves an average sensitivity of 84.17% at 96.42% specificity, and 75.27% sensitivity at 98.93% specificity [119]. Recently, RNAz has been used to perform a comparative screening of several vertebrate genomes, and it predicted more than 30,000 putative ncRNA genes in the human genome [120]. Among them, almost a thousand ncRNA genes were conserved in all four vertebrate genomes included in the screening, which strongly suggests that these ncRNAs are biologically functional.

Despite the initial success of these ncRNA gene finders, there is yet much room for improvement. In fact, the average prediction ratios of the existing algorithms are not as high as one might hope, and they still do not work well for certain classes of RNAs.[4] However, the performance of ncRNA gene finders has been improving at a fast pace, and it is clear that computational gene finders will play important roles in unveiling more and more novel ncRNAs in the future.

3.7 Conclusion

Unlike protein-coding genes, ncRNA genes have remained unnoticed until relatively recently. Compared to the annotation of protein-coding genes, which is nearly complete in many genomes that have been sequenced so far, the annotation of ncRNA genes has just begun. At present, it is even difficult to give a reliable estimate of the total number of ncRNAs in a genome. Given the enormous amount of genomic data, which is still increasing, we cannot stress strongly enough the importance of computational methods in finding ncRNA genes and analyzing them. As we have shown in this chapter, context-sensitive HMMs can provide a convenient framework for building ncRNA gene finders and RNA sequence analysis tools.

[4] For example, the sensitivity of RNAz for U70 snoRNAs (small nucleolar RNAs) is below 62%, and for tmRNAs (transfer-messenger RNA) it is below 25% [120].

Chapter 4

Profile Context-Sensitive Hidden Markov Models

Systematic screening of many genomes has identified a large number of novel ncRNAs, whose roles in the cell machinery are truly diverse [30, 44]. However, unlike protein-coding genes, the annotation of ncRNA genes (which are portions in the genome that give rise to biochemically functional ncRNAs) is still far from complete. In order to expedite the annotation process, we need efficient methods that can be used for automatic identification of ncRNA genes.

A practical way of finding new ncRNA genes is to perform a *similarity search* (or a *homology search*) of the database to look for homologues of known ncRNA sequences. As many ncRNAs have secondary structures that are well conserved among different species, it is important to incorporate this structural information in the search. In fact, scoring schemes that effectively combine contributions from the sequence similarity and the structural similarity are known to be much more discriminative than schemes that are based on sequence similarity alone [31].

Until now, a number of methods have been proposed that can be used for representing RNA secondary structures and performing structural alignment of RNA sequences [26, 66, 95]. However, many models including the *covariance models (CMs)* [26] that have been widely used in RNA sequence analysis and the *pair hidden Markov models on tree structures (PHMMTSs)* [95], which are a more recent development, cannot handle RNAs with pseudoknot structures. As there exist many RNAs with functionally important pseudoknots [47, 114], this can be potentially a serious limitation. Recently, Matsui et al. proposed an interesting method based on *pair stochastic tree adjoining grammars (PSTAGs)* that can be used for structural alignment of pseudoknots [66]. The PSTAG is capable of aligning and predicting simple pseudoknots with 2-crossing property, which includes many known pseudoknots but not all of them.

In this chapter, we propose a new approach for representing and aligning RNA pseudoknots. The proposed method is based on *profile context-sensitive hidden Markov models (profile-csHMMs)* [129], which can in principle handle *any kind of pseudoknots*. To demonstrate the effectiveness of the new approach, we build a structural alignment tool for RNAs, which uses a single RNA sequence with structural annotation to align unfolded RNAs and predict their structures. Experimental results will show that the profile-csHMM based approach can achieve high prediction ratios, providing an effective framework for analyzing RNA secondary structures and building tools for identifying new ncRNAs. In addition to this, we also propose an efficient scheme that can make a profile-csHMM based homology search significantly faster. The proposed scheme utilizes a prescreening filter that is built using a profile-HMM, and it will be shown that it can make the search speed around eighty times faster, without affecting the overall prediction accuracy.

The content of this chapter is mainly drawn from [129], [133], and [134].

4.1 Outline

In Section 4.2, we introduce the basic concept of profile-csHMMs. In Section 4.2.1, we explain how a profile-csHMM can be constructed from a multiple sequence alignment, and elaborate how the constructed model can represent long-range correlations between distant bases. Then the descriptive power of profile-csHMMs is considered in Section 4.2.2.

The optimal alignment problem of profile-csHMMs is considered in Section 4.3. We propose a dynamic programming algorithm, called the sequential component adjoining (SCA) algorithm, that can be used for solving this problem. The initialization step of the SCA algorithm is described in Section 4.3.1, and the adjoining rules that can be iteratively applied to find the optimal alignment are described in Section 4.3.2. In Section 4.3.3, we explain about the adjoining order that defines how the adjoining rules should be applied to obtain the final alignment. The termination step and the trace-back procedure of the SCA algorithm are elaborated in Section 4.3.4 and Section 4.3.5, respectively. The computational complexity of the algorithm is considered in Section 4.3.6.

In Section 4.4, we propose a new method for finding the structural alignment of RNAs based on profile-csHMMs. Section 4.4.1 describes the basic idea of the proposed method, and Section 4.4.2 shows how we can reduce its complexity by restricting the search region for finding the matching bases. Several examples of structural alignments are given in Section 4.4.3, which demonstrates the generality of the proposed method. In Section 4.4.4, experimental results for aligning pseudoknots

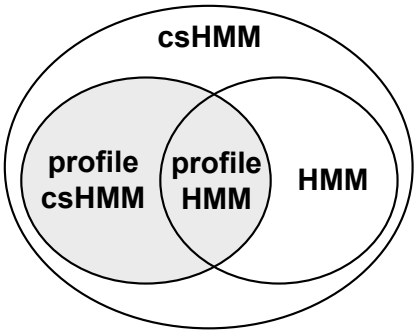


Figure 4.1: Relation between profile-csHMMs and other statistical models.

and predicting their secondary structures are given, which will clearly indicate the advantages of the profile-csHMM based approach.

In Section 4.5, we propose a practical method that can make a profile-csHMM based database search significantly faster. The basic idea of the proposed method is described in Section 4.5.1, and its technical details are given in Section 4.5.2. In Section 4.5.3, we prove that the proposed method does not affect the prediction accuracy. Finally, we show experimental results in Section 4.5.4, and we conclude the chapter in Section 4.6.

4.2 Profile context-sensitve HMM

Profile hidden Markov models (profile-HMMs) are specifically constructed HMMs that are well suited for modeling the key motives and common features in a set of related sequences [25]. They have been briefly described in Section 3.3.1. Due to their effectiveness in representing probabilistic profiles of sequence families, profile-HMMs have been widely used in biological sequence analysis, especially for building protein-coding gene finders. However, profile-HMMs are inherently incapable of modeling correlations between distant bases, hence they cannot be used for representing RNA secondary structures.

We can overcome this limitation by using the profile-csHMMs [129] instead of the profile-HMMs for representing RNA sequence profiles. Profile-csHMMs are a subclass of context-sensitive HMMs (csHMMs) that have been introduced in Chapter 2. The relation between profile-csHMMs and other models are illustrated in Figure 4.1.

4.2.1 Model construction

The structure of a profile-csHMM is similar to that of traditional profile-HMMs.[1] A profile-csHMM has a linear structure that repetitively uses three kinds of states, namely, *match states* M_k, *delete states* D_k, and *insert states* I_k, to describe base substitutions, deletions and additional insertions at different locations.

4.2.1.1 Constructing an ungapped model

The match state M_k represents the case when a base in the observed RNA sequence matches the kth base in the consensus RNA sequence. For example, if the observed RNA sequence $\mathbf{x} = x_1 x_2 \dots x_L$ exactly matches the consensus sequence without any gap, the corresponding state sequence of the profile-csHMM will be simply $\mathbf{y} = y_1 y_2 \dots y_L = M_1 M_2 \dots M_L$, where y_ℓ is the underlying hidden state of x_ℓ. Consequently, the number of match states is identical to the number of bases in the consensus RNA sequence, and we can obtain an *ungapped model* for the consensus sequence by interconnecting the match states.

The main difference beween profile-csHMMs and traditional profile-HMMs is as follows. Unlike profile-HMMs, profile-csHMMs have three different types of match states: *single-emission* match states, *pairwise-emission* match states, and *context-sensitive* match states. Single-emission match states are identical to regular states in ordinary HMMs, and they are used to represent base positions in the consensus sequence that are not involved in base pairing. On the other hand, for two bases that form a base pair, we use a pairwise-emission match state and the corresponding context-sensitive match state to model the correlation between them. Each pair of pairwise-emission match state and context-sensitive match state has a distinct memory dedicated to it. The auxiliary memory can be either a stack or a queue, whichever is more convenient for modeling the correlations that are under consideration. A pairwise-emission match state stores the emitted symbol in this memory before making a transition to the next state. When we enter a context-sensitive match state, it first accesses the memory and reads the symbol that was previously emitted at the corresponding pairwise-emission match state. The emission probabilities are adjusted according to this observation. For example, when a C was emitted at the pairwise-emission match state, the emission probabilities at the context-sensitive match state may be adjusted such that it emits a G with high probability (or simply, with probability one).

[1] For an introduction to conventional profile-HMMs, see [25] or [29].

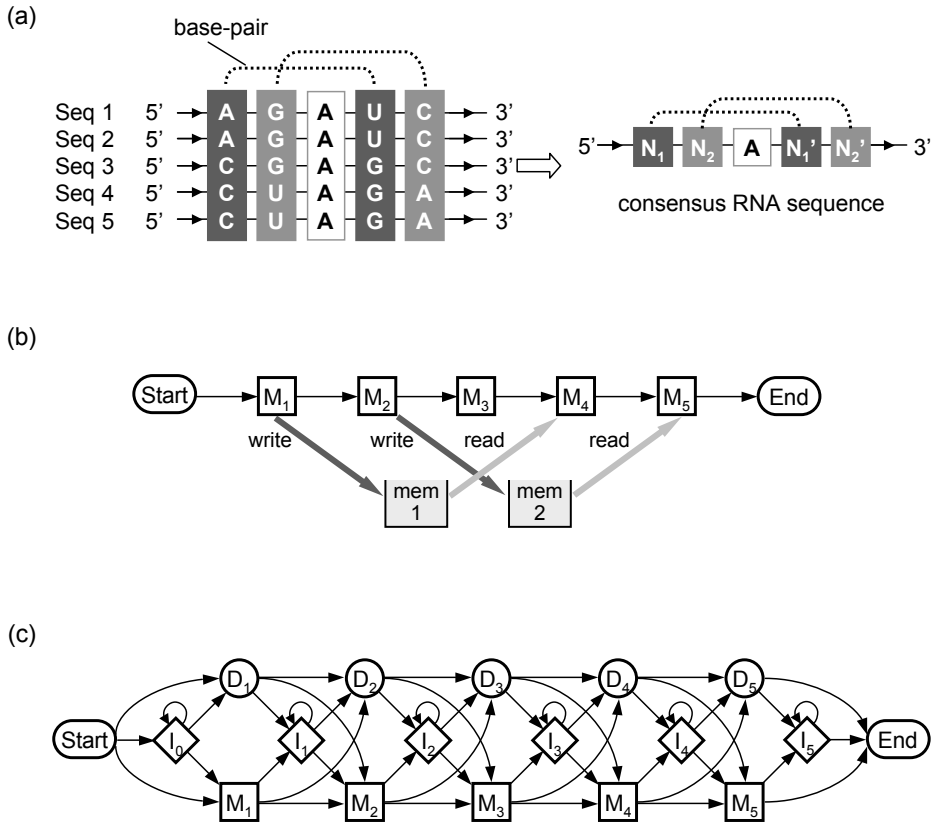


Figure 4.2: Constructing a profile-csHMM from an RNA multiple sequence alignment. (a) An alignment of five RNA sequences. The consensus RNA secondary structure has two base pairs. (b) Ungapped profile-csHMM that represents the consensus RNA sequence. (c) The final profile-csHMM that allows additional insertions and deletions at any location.

4.2.1.2 Representing insertions and deletions

Once we have constructed the ungapped model that forms the backbone of the profile-csHMM, we add delete states D_k and insert states I_k to obtain the final model. Sometimes, an observed RNA sequence can be shorter than the consensus RNA sequence at hand. In such cases, if we align the two sequences, there will be one or more bases in the consensus sequence that are not present in the observed sequence. Such gaps are represented by the delete states, where D_k models the deletion of the kth symbol in the consensus sequence. Note that the delete states are non-emitting states, and they are simply used as placeholders that interconnect other states. On the contrary, when the observed RNA sequence has additional bases that are not present in the consensus sequence, these bases are modeled by insert states. The state I_k is used for representing insertions between the positions k and $k+1$ in the original consensus sequence.

4.2.1.3 Constructing a profile-csHMM from an RNA sequence alignment

Following the previous idea, it is quite straightforward to construct a profile-csHMM based on an RNA multiple sequence alignment with structural annotation. As an example, let us consider constructing a profile-csHMM from the alignment shown in Figure 4.2 (a). In this alignment, five RNA sequences are aligned to each other. Since the length of the consensus RNA sequence is five, the corresponding profile-csHMM should have five match states $M_1, M_2, \ldots, M_5$. Note that the first and the fourth bases in the consensus sequence form a base pair. In order to model the correlation between these bases, we use a pairwise-emission state for M_1 and a context-sensitive state for M_4. These two states share a common auxiliary memory, where M_1 stores its emission in the memory and M_4 reads the stored symbol from the memory to adjust its emission probabilities. Similarly, we use a pairwise-emission state for M_2 and a context-sensitive state for M_5 to model the base pair that is formed between the second and the fifth bases. Lastly, we use a single-emission state for M_3, since the third base in the consensus sequence does not form a base pair. Now that the type of every match state is determined, we can interconnect the match states to obtain an ungapped model for the consensus RNA as illustrated in Figure 4.2 (b). Finally, we add delete states D_k and insert states I_k to the ungapped model to obtain the final profile-csHMM, which is shown in Figure 4.2 (c).

4.2.2 Descriptive power

As we can see in the previous example, profile-csHMMs provide a simple and intuitive way of representing RNA sequence profiles. One important advantage of profile-csHMMs is their large descriptive power. In fact, profile-csHMMs can represent *any* kind of base pair correlations by arranging the pairwise-emission match states and the context-sensitive match states in an appropriate manner, hence capable of representing any kind of pseudoknots unlike other existing models. As mentioned earlier, CMs [26] and PHMMTSs [95] can only represent RNA secondary structures with nested correlations, hence incapable of dealing with pseudoknots. PSTAGs [66] are capable of representing pseudoknots with exactly 2-crossing property. A secondary structure is said to have a *m-crossing property*, if there exist m (≥ 2) base pairs in the given secondary structure such that any two pairs in these m base pairs cross each other. Figure 4.3 (a) shows an example of an RNA secondary structure with 2-crossing property. PSTAGs can handle many known pseudoknots, as a large portion of known pseudoknots has 2-crossing property. But there also exist more complex pseudoknots that are beyond the descriptive power of PSTAGs. One such example is the Flavivirus 3′ UTR pseu-

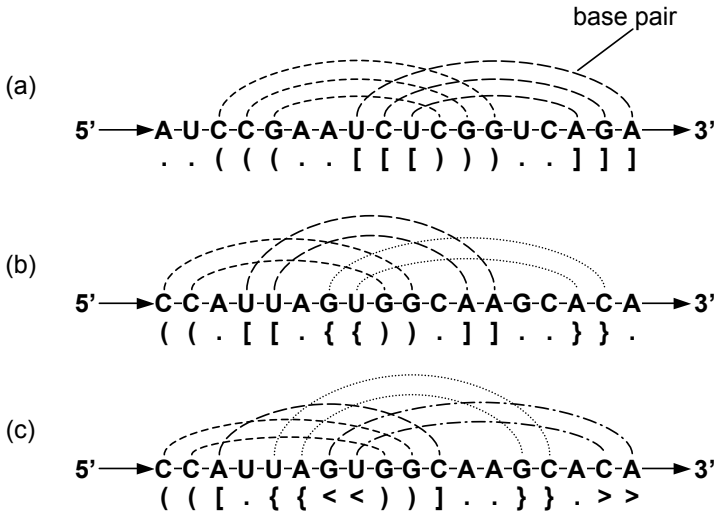


Figure 4.3: Various types of RNA secondary structures. (a) RNA with 2-crossing property. (b) RNA with 3-crossing property. (c) RNA with 4-crossing property.

doknot family [98], which will be considered in our experiments presented in Section 4.4.4. It will be shown that profile-csHMMs can be used for modeling and predicting the secondary structure of these pseudoknots. Profile-csHMM can also represent RNAs with even more complex secondary structures as those shown in Figure 4.3 (b) (RNA with 3-crossing property) and Figure 4.3 (c) (RNA with 4-crossing property), in a similar manner as described in Section 4.2.1.

4.3 Optimal alignment of profile-csHMM

Let us assume that we have constructed a profile-csHMM that represents the consensus sequence of an RNA family. Given an unfolded RNA sequence with no structural annotation, how can we find the best structural alignment between the new sequence and the consensus RNA sequence at hand? In fact, this alignment can be obtained by finding the optimal state sequence, or the "optimal path," of the profile-csHMM that maximizes the observation probability of the unfolded RNA sequence. For example, let us consider aligning the RNA sequence in Figure 4.4 (a) to the consensus RNA sequence shown in Figure 4.2 (a). In order to do this, we first find the optimal state sequence of the profile-csHMM in Figure 4.2 (c) that maximizes the probability of the given RNA. Assume that the optimal path is $M_1M_2D_3M_4M_5I_5$ as depicted in Figure 4.4 (b). The presence of the delete state "D_3" implies that the third base in the consensus RNA sequence is omitted in the

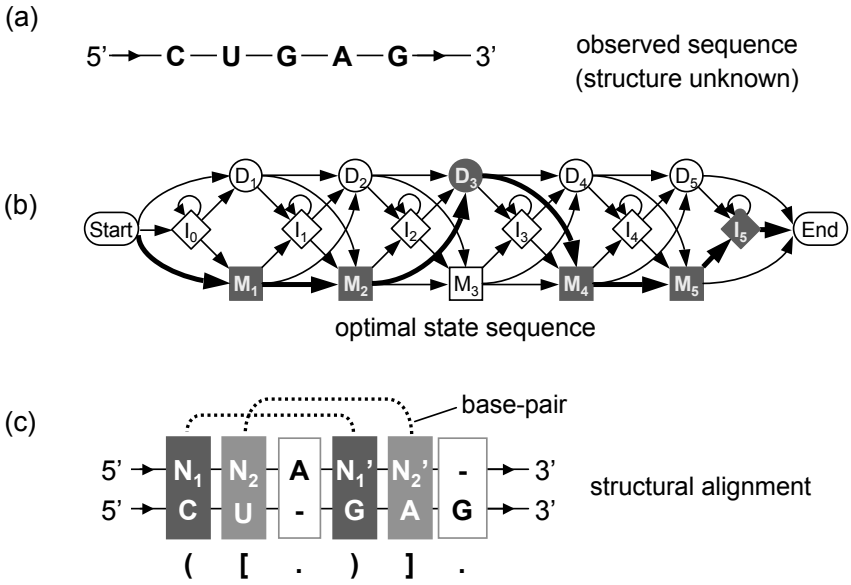


Figure 4.4: (a) An RNA sequence with no structural annotation. (b) Optimal state sequence of the observed RNA sequence in the given profile-csHMM. (c) Structural alignment obtained from the optimal state sequence.

observed RNA sequence. Furthermore, the insert state "I_5" indicates that the observed sequence has an additional base right after the fifth position of the original consensus sequence. Based on these observations, we can obtain the best structural alignment between the new sequence and the given profile as illustrated in Figure 4.4 (c), and the secondary structure of the new RNA sequence can be immediately inferred from this alignment.

The optimal state sequence of a profile-csHMM can be systematically found by using the *sequential component adjoining (SCA) algorithm* [129]. The SCA algorithm can be viewed as a generalization of the *Viterbi algorithm* (used for conventional HMMs) [116] and the *Cocke-Younger-Kasami (CYK) algorithm* (used for context-free grammars) [60]. Like these two algorithms, the SCA algorithm finds the optimal path in an iterative manner, first finding the optimal state sequence of short subsequences and then using this information to find the optimal state sequence of longer sequences.

The major difference between the SCA algorithm and the other algorithms is as follows. Firstly, the SCA algorithm can define and use subsequences that consist of multiple non-overlapping regions in the original sequence. Instead of using a fixed number of indices to designate a subsequence, the SCA algorithm uses a set of variable number of closed intervals to describe a subse-

quence. For example, given a sequence $\mathbf{x} = x_1 x_2 \dots x_L$, we can define its subsequences as follows.

$$\begin{aligned} \mathcal{N}_1 = \{[1,3]\} \quad &\rightarrow \quad \mathbf{x}(\mathcal{N}_1) = x_1 x_2 x_3 \\ \mathcal{N}_2 = \{[1,1],[5,7]\} \quad &\rightarrow \quad \mathbf{x}(\mathcal{N}_2) = x_1 \;\; x_5 x_6 x_7 \\ \mathcal{N}_3 = \{[1,2],[5,7],[9,9]\} \quad &\rightarrow \quad \mathbf{x}(\mathcal{N}_3) = x_1 x_2 \;\; x_5 x_6 x_7 \;\; x_9 \end{aligned}$$

In general, if we have a set of I non-overlapping intervals $\mathcal{N} = \{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_I\}$, where the intervals $\mathbf{n}_i = [n_i^\ell, n_i^r]$ satisfy

$$n_i^r < n_j^\ell \;\text{ for }\; i < j, \tag{4.1}$$

the corresponding subsequence $\mathbf{x}(\mathcal{N})$ is defined as

$$\mathbf{x}(\mathcal{N}) = \underbrace{x_{n_1^\ell} \dots x_{n_1^r}}_{\mathbf{n}_1} \;\; \underbrace{x_{n_2^\ell} \dots x_{n_2^r}}_{\mathbf{n}_2} \;\; \dots \;\; \underbrace{x_{n_I^\ell} \dots x_{n_I^r}}_{\mathbf{n}_I}.$$

This generalization significantly increases the number of ways in which the intermediate subsequences can be defined and extended during the iterative process of finding the optimal path. Secondly, the SCA algorithm explicitly specifies how the optimal paths of shorter subsequences should be extended and adjoined together to obtain the optimal path of a longer subsequence. As there are innumerable ways of defining the intermediate subsequences, the SCA algorithm cannot simply find the optimal path either *left-to-right* (as the Viterbi algorithm) or *inside-to-outside* (as the CYK algorithm). In practice, we have to define a model-dependent "adjoining order" in such a way that takes all the correlations in the given profile-csHMM into consideration. Following the specified order, we iteratively apply a set of sequence adjoining and extension rules until we obtain the optimal probability of the entire sequence $\mathbf{x}$.

Before describing the algorithm, let us first define the notations. Let K be the length of the profile-csHMM, i.e., the number of match states, and L be the length of the observation sequence $\mathbf{x}$. We denote the emission probability of a symbol x at a single-emission state or a pairwise-emission state v as $e(x|v)$. The emission probability of a symbol x_c at a context-sensitive state w is denoted as $e(x_c|w, x_p)$, where x_p is the symbol that was previously emitted at the corresponding pairwise-emission state. We denote the transition probability from state v to state w as $t(v, w)$. As before, we let $\mathcal{N} = \{\mathbf{n}_1, \dots, \mathbf{n}_I\}$ be an ordered set of I intervals, and we define $\mathcal{S} = \{\mathbf{s}_1, \dots, \mathbf{s}_I\}$ to be an ordered set of state-pairs $\mathbf{s}_i = (s_i^\ell, s_i^r)$, where s_i^ℓ and s_i^r denote the hidden states $y_{n_i^\ell}$ and $y_{n_i^r}$ at each

end of the ith interval $\mathbf{n}_i = [n_i^\ell, n_i^r]$. Finally, we define $\alpha(\mathcal{N}, \mathcal{S})$ to be the maximum log-probability of the subsequence $\mathbf{x}(\mathcal{N})$

$$\alpha(\mathcal{N}, \mathcal{S}) = \max_{\mathbf{y}(\mathcal{N})} \Big[\log P\left(\mathbf{x}(\mathcal{N}), \mathbf{y}(\mathcal{N})\right) \Big],$$

where its underlying state-sequence

$$\mathbf{y}(\mathcal{N}) = \underbrace{y_{n_1^\ell} \dots y_{n_1^r}}_{\mathbf{n}_1} \; \underbrace{y_{n_2^\ell} \dots y_{n_2^r}}_{\mathbf{n}_2} \; \cdots \; \underbrace{y_{n_I^\ell} \dots y_{n_I^r}}_{\mathbf{n}_I}.$$

satisfies $y_{n_i^\ell} = s_i^\ell$ and $y_{n_i^r} = s_i^r$ for all $i = 1, \dots, I$. We also define the variables $\lambda_a(\mathcal{N}, \mathcal{S})$ and $\lambda_b(\mathcal{N}, \mathcal{S})$ that will be used later for tracing back the optimal state sequence $\mathbf{y}^*$.

4.3.1 Initialization

Initially, we compute $\alpha(\mathcal{N}, \mathcal{S})$ for single bases and single base pairs.

(i) For a position k $(1 \leq k \leq K)$ in the profile-csHMM where M_k is a single-emission match state, we let

$$\begin{aligned}
\alpha\Big(\{[n,n]\}, \{(M_k, M_k)\}\Big) &= \log e(x_n | M_k), \\
\lambda_a\Big(\{[n,n]\}, \{(M_k, M_k)\}\Big) &= (\varnothing, \varnothing), \\
\lambda_b\Big(\{[n,n]\}, \{(M_k, M_k)\}\Big) &= (\varnothing, \varnothing),
\end{aligned}$$

for all positions $1 \leq n \leq L$. Similarly, we let

$$\begin{aligned}
\alpha\Big(\{[n,n-1]\}, \{(D_k, D_k)\}\Big) &= 0, \\
\lambda_a\Big(\{[n,n-1]\}, \{(D_k, D_k)\}\Big) &= (\varnothing, \varnothing), \\
\lambda_b\Big(\{[n,n-1]\}, \{(D_k, D_k)\}\Big) &= (\varnothing, \varnothing),
\end{aligned}$$

for all $1 \leq n \leq L+1$.

(ii) For positions j and k $(1 \leq j < k \leq K)$ where M_j is a pairwise-emission state and M_k is the corresponding context-sensitive state, we let $\mathcal{N}_1 = \{[n,n],[m,m]\}$, $\mathcal{S}_1 = \{(M_j, M_j), (M_k, M_k)\}$ and

compute

$$\begin{aligned}
\alpha(\mathcal{N}_1, \mathcal{S}_1) &= \log e(x_n|M_j) + \log e(x_m|M_k, x_n), \\
\lambda_a(\mathcal{N}_1, \mathcal{S}_1) &= (\varnothing, \varnothing), \\
\lambda_b(\mathcal{N}_1, \mathcal{S}_1) &= (\varnothing, \varnothing),
\end{aligned}$$

for all positions $1 \leq n < m \leq L$. Furthermore, we initialize the log-probability $\alpha(\mathcal{N}_2, \mathcal{S}_2)$ for $\mathcal{N}_2 = \{[n, n-1], [m, m-1]\}$ and $\mathcal{S}_2 = \{(D_j, D_j), (D_k, D_k)\}$ as follows

$$\begin{aligned}
\alpha(\mathcal{N}_2, \mathcal{S}_2) &= 0, \\
\lambda_a(\mathcal{N}_2, \mathcal{S}_2) &= (\varnothing, \varnothing), \\
\lambda_b(\mathcal{N}_2, \mathcal{S}_2) &= (\varnothing, \varnothing),
\end{aligned}$$

for all n and m ($1 \leq n \leq m \leq L+1$).

(iii) For single bases emitted at insert states, we initialize the log-probabilities as follows

$$\begin{aligned}
\alpha\Big(\{[n, n]\}, \{(I_k, I_k)\}\Big) &= \log e(x_n|I_k), \\
\lambda_a\Big(\{[n, n]\}, \{(I_k, I_k)\}\Big) &= (\varnothing, \varnothing), \\
\lambda_b\Big(\{[n, n]\}, \{(I_k, I_k)\}\Big) &= (\varnothing, \varnothing),
\end{aligned}$$

for $0 \leq k \leq K$ and $1 \leq n \leq L$.

4.3.2 Adjoining subsequences

During the initialization process, we computed the log-probability for all subsequences that consist of a single base or a single base pair. Now, these subsequences can be recursively adjoined to obtain the probability of longer subsequences by applying the following adjoining rules.

Rule 1 Consider the log-probabilities $\alpha(\mathcal{N}^a, \mathcal{S}^a)$ and $\alpha(\mathcal{N}^b, \mathcal{S}^b)$ of the two subsequences $\mathbf{x}(\mathcal{N}^a)$ and $\mathbf{x}(\mathcal{N}^b)$, where

$$\mathcal{N}^a = \{\mathbf{n}_1^a, \ldots, \mathbf{n}_{I_a}^a\},\ \mathcal{S}^a = \{\mathbf{s}_1^a, \ldots, \mathbf{s}_{I_a}^a\},$$
$$\mathcal{N}^b = \{\mathbf{n}_1^b, \ldots, \mathbf{n}_{I_b}^b\},\ \mathcal{S}^b = \{\mathbf{s}_1^b, \ldots, \mathbf{s}_{I_b}^b\}.$$

We assume that there is no overlap between the symbol sequences $\mathbf{x}(\mathcal{N}^a)$ and $\mathbf{x}(\mathcal{N}^b)$ nor between the underlying state sequences $\mathbf{y}(\mathcal{N}^a)$ and $\mathbf{y}(\mathcal{N}^b)$. In this case, we can apply the following adjoining rule to compute the optimal log-probability of a longer sequence $\mathbf{x}(\mathcal{N})$

$$\begin{aligned} \alpha(\mathcal{N}, \mathcal{S}) &= \alpha(\mathcal{N}^a, \mathcal{S}^a) + \alpha(\mathcal{N}^b, \mathcal{S}^b), \\ \lambda_a(\mathcal{N}, \mathcal{S}) &= (\mathcal{N}^a, \mathcal{S}^a), \\ \lambda_b(\mathcal{N}, \mathcal{S}) &= (\mathcal{N}^b, \mathcal{S}^b), \end{aligned}$$

where $\mathcal{N}$ and $\mathcal{S}$ are unions of the smaller sets

$$\mathcal{N} = \mathcal{N}^a \cup \mathcal{N}^b = \{\mathbf{n}_1, \ldots, \mathbf{n}_I\},\ \mathcal{S} = \mathcal{S}^a \cup \mathcal{S}^b = \{\mathbf{s}_1, \ldots, \mathbf{s}_I\},$$

where $I = I_a + I_b$ and the intervals $\mathbf{n}_i$ are relabeled such that they satisfy (4.1) and $\mathbf{s}_i \in \mathcal{S}$ corresponds $\mathbf{n}_i \in \mathcal{N}$. ■

Rule 2 Assume that there exist two intervals $\mathbf{n}_i, \mathbf{n}_{i+1} \in \mathcal{N}$ that satisfy $n_i^r + 1 = n_{i+1}^\ell$, which implies that the two intervals $[n_i^\ell,\ n_i^r]$ and $[n_{i+1}^\ell,\ n_{i+1}^r]$ are adjacent to each other. For simplicity, let us assume that $i = I - 1$. In this case, we can combine the two intervals $\mathbf{n}_{I-1}$ and $\mathbf{n}_I$ to obtain a larger interval

$$\mathbf{n}'_{I-1} = [n_{I-1}^\ell,\ n_I^r] = \{n |\ n_{I-1}^\ell \le n \le\ n_I^r\},$$

where the corresponding state-pair is $\mathbf{s}'_{I-1} = (s_{I-1}^\ell,\ s_I^r)$. Now, the log-probability $\alpha(\mathcal{N}', \mathcal{S}')$ for

$$\mathcal{N}' = \{\mathbf{n}_1, \ldots, \mathbf{n}_{I-2}, \mathbf{n}'_{I-1}\},\ \mathcal{S}' = \{\mathbf{s}_1, \ldots, \mathbf{s}_{I-2}, \mathbf{s}'_{I-1}\}$$

can be computed as follows

$$\begin{aligned}
\alpha(\mathcal{N}', \mathcal{S}') &= \max_{n^r_{I-1}} \left(\max_{s^r_{I-1}, s^\ell_I} \Big[\alpha(\mathcal{N}, \mathcal{S}) + \log t(s^r_{I-1}, s^\ell_I) \Big] \right), \\
(n^*, s^*_r, s^*_\ell) &= \underset{(n^r_{I-1}, s^r_{I-1}, s^\ell_I)}{\arg\max} \Big[\alpha(\mathcal{N}, \mathcal{S}) + \log t(s^r_{I-1}, s^\ell_I) \Big], \\
\mathcal{N}^* &= \left\{ \mathbf{n}_1, \ldots, \mathbf{n}_{I-2}, [n^\ell_{I-1}, n^*], [n^* + 1, n^r_I] \right\}, \\
\mathcal{S}^* &= \left\{ \mathbf{s}_1, \ldots, \mathbf{s}_{I-2}, (s^\ell_{I-1}, s^*_r), (s^*_\ell, s^r_I) \right\}, \\
\lambda_a(\mathcal{N}', \mathcal{S}') &= (\mathcal{N}^*, \mathcal{S}^*), \\
\lambda_b(\mathcal{N}', \mathcal{S}') &= (\varnothing, \varnothing).
\end{aligned}$$

For $i < I-1$, we can similarly combine the two adjacent intervals $\mathbf{n}_i$ and $\mathbf{n}_{i+1}$ to obtain the optimal log-probability $\alpha(\mathcal{N}', \mathcal{S}')$ of the updated sets $\mathcal{N}'$ and $\mathcal{S}'$. ■

For simplicity, we have described the adjoining process in two distinct steps; (i) adjoining non-overlapping subsequences and (ii) combining adjacent intervals in a single subsequence. However, in many cases, it is possible (and usually more convenient) to apply the two rules at the same time. For example, if we know the optimal probability of two adjacent subsequences, where each subsequence consists of a single interval, we can adjoin the two sequences and combine the two intervals to compute the optimal probability of a longer subsequence that has also a single interval.

4.3.3 Adjoining order

As mentioned earlier, when using the SCA algorithm, we have to specify the order according to which the adjoining rules should be applied. This adjoining order can be obtained from the consensus RNA sequence that was used to construct the profile-csHMM. Based on the consensus sequence, we first find out how the bases and the base pairs in the given sequence can be adjoined one by one to obtain the entire sequence. During this procedure, we try to minimize the number of intervals that is needed to describe the intermediate subsequences, as a larger number of intervals leads to a higher computational cost for adjoining the subsequences. An example is shown in Figure 4.5, which illustrates how we can obtain the consensus sequence in Figure 4.2 (a) by sequentially adjoining its base and base pairs. Note that the numbers inside the squares in Figure 4.5 indicate the original base-positions. We follow this order to compute the optimal log-probability

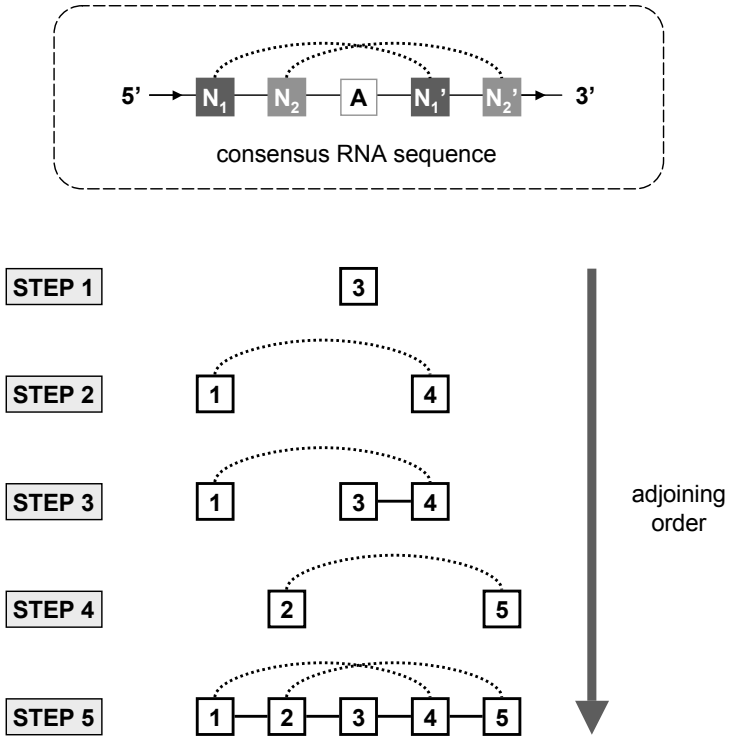


Figure 4.5: The adjoining order of the profile-csHMM shown in Figure 4.2. This illustrates how we can adjoin the base and the base pairs in the consensus RNA sequence to obtain the entire sequence.

of the subsequences (of the observed RNA sequence) that correspond to the subsequence (of the consensus sequence) at each step.

At each step, we first compute the log-probability $\alpha(\mathcal{N}, \mathcal{S})$ of those subsequences, whose terminal states (i.e., s_i^ℓ, s_i^r) do not contain insert states. For example, at STEP 1, we compute $\alpha(\mathcal{N}_1, \mathcal{S}_1)$ for $\mathcal{N}_1 = \{[n, n]\}$, $\mathcal{S}_1 = \{(M_3, M_3)\}$ and $\mathcal{N}_1 = \{[n, n-1]\}$, $\mathcal{S}_1 = \{(D_3, D_3)\}$ using the initialization rules described in Section 4.3.1. Note that the states in $\mathcal{S}_1$ correspond to the third base position in the original consensus sequence. Similarly, we compute the log-probability $\alpha(\mathcal{N}_2, \mathcal{S}_2)$ at STEP 2 (and also $\alpha(\mathcal{N}_4, \mathcal{S}_4)$ at STEP 4) based on the base pair initialization rules in Section 4.3.1. At certain steps, the optimal log-probability is obtained by combining the log-probabilities computed in the previous steps. For example, at STEP 3, we can compute $\alpha(\mathcal{N}_3, \mathcal{S}_3)$ for $\mathcal{N}_3 = \{[n_1, n_1], [n_2^\ell, n_2^r]\}$ and $\mathcal{S}_3 = \{(M_1, M_1), (M_3, M_4)\}$, by combining $\alpha(\mathcal{N}_1, \mathcal{S}_1)$ and $\alpha(\mathcal{N}_2, \mathcal{S}_2)$ as follows

$$\alpha(\mathcal{N}_3, \mathcal{S}_3) = \max_v \Big[\alpha(\mathcal{N}_1, \mathcal{S}_1) + \log t(v, M_4) + \alpha(\mathcal{N}_2, \mathcal{S}_2)\Big],$$

where $\mathcal{N}_1 = \{[n_2^\ell, n_2^r - 1]\}$, $\mathcal{S}_1 = \{(M_3, v)\}$ and $\mathcal{N}_2 = \{[n_1, n_1], [n_2^r, n_2^r]\}$, $\mathcal{S}_2 = \{(M_1, M_1), (M_4, M_4)\}$. In the example in Figure 4.5, the log-probability $\alpha(\mathcal{N}_1, \mathcal{S}_1)$ is computed at STEP 1 and $\alpha(\mathcal{N}_2, \mathcal{S}_2)$ is computed at STEP 2.

After computing these log-probabilities, we move on to compute the log-probabilities of those subsequences that have one or more insertions at the beginning and/or end of some intervals. For example, at STEP 1, we can compute $\alpha(\mathcal{N}_1, \mathcal{S}_1)$ for $\mathcal{N}_1 = \{[n, n+1]\}$ and $\mathcal{S}_1 = \{(M_3, I_3)\}$ from

$$\alpha(\mathcal{N}_1, \mathcal{S}_1) = \alpha(\mathcal{N}_1^a, \mathcal{S}_1^a) + \log t(M_3, I_3) + \alpha(\mathcal{N}_1^b, \mathcal{S}_1^b),$$

where $\mathcal{N}_1^a = \{[n, n]\}$, $\mathcal{S}_1^a = \{(M_3, M_3)\}$, $\mathcal{N}_1^b = \{[n+1, n+1]\}$, $\mathcal{S}_1^b = \{(I_3, I_3)\}$. In a similar manner, we can also deal with a left insertion as well as multiple insertions.

4.3.4 Termination

By iteratively applying the adjoining rules as described in Section 4.3.2, we can ultimately obtain the log-probability $\alpha(\mathcal{N}, \mathcal{S})$ for $\mathcal{N} = \{[1, L]\}$ and $\mathcal{S} = \{(s^\ell, s^r)\}$, for all $s^\ell \in \{I_0, M_1, D_1\}$ and $s^r \in \{I_K, M_K, D_K\}$. Based on this result, the log-probability of the optimal state sequence $\mathbf{y}^*$ can be computed from

$$\begin{aligned}
\log P(\mathbf{x}, \mathbf{y}^*) &= \max_{\mathbf{y}} \Big[\log P(\mathbf{x}, \mathbf{y}) \Big] \\
&= \max_{s^\ell, s^r} \Big[\log t(\text{START}, s^\ell) + \alpha(\mathcal{N}, \mathcal{S}) + \log t(s^r, \text{END}) \Big], \\
(s_\ell^*, s_r^*) &= \arg\max_{(s_1^\ell, s_1^r)} \Big[\log t(\text{START}, s^\ell) + \alpha(\mathcal{N}, \mathcal{S}) + \log t(s^r, \text{END}) \Big], \\
\lambda^* &= \Big([1,\ L], (s_\ell^*, s_r^*) \Big).
\end{aligned}$$

Note that $t(\text{START}, s^\ell)$ is the probability that the profile-csHMM begins at the state s^ℓ, and $t(s^r, \text{END})$ is the probability that the model terminates after s^r.

4.3.5 Trace-back

Now that we have obtained the maximum probability of the sequence $\mathbf{x}$, we can easily trace back the algorithm to find the optimal path that gave rise to this probability. For notational convenience, let us define $\lambda_t = (\mathcal{N}, \mathcal{S})$. We also need a stack T during the process. The trace-back procedure can be described as follows.

STEP 1 Let $y_i = 0$ $(i = 1, 2, \ldots, L)$.

STEP 2 Push λ^* onto the stack T.

STEP 3 Pop $\lambda_t = (\mathcal{N}, \mathcal{S})$ from T. If $\lambda_t = (\varnothing, \varnothing)$, go to STEP 6. Otherwise, proceed to STEP 4.

STEP 4 If $\lambda_a(\lambda_t) \neq (\varnothing, \varnothing)$ push $\lambda_a(\lambda_t)$ onto T. Otherwise, $y_{n_i^\ell} = s_i^\ell$, for all $\mathbf{n}_i = [n_i^\ell,\ n_i^r] \in \mathcal{N}$ and the corresponding $\mathbf{s}_i = [s_i^\ell,\ s_i^r] \in \mathcal{S}$. (Note that when $\lambda_a(\lambda_t) = (\varnothing, \varnothing)$, we have $n_i^\ell = n_i^r$ and $s_i^\ell = s_i^r$.)

STEP 5 If $\lambda_b(\lambda_t) \neq (\varnothing, \varnothing)$ push $\lambda_b(\lambda_t)$ onto T.

STEP 6 If T is empty, proceed to STEP 7. Otherwise, go to STEP 3.

STEP 7 Let $\mathbf{y}^* = y_1 y_2 \ldots y_L$ and terminate.

At the end of the trace-back procedure, we can obtain the optimal state sequence $\mathbf{y}^*$ that maximizes the probability of observing $\mathbf{x}$ based on the profile-csHMM at hand.

4.3.6 Computational complexity

The computational complexity of the SCA algorithm is not fixed, and it depends on the structure of the RNA that is represented by the given profile-csHMM. For example, for a simple RNA with only one stem-loop, the computational complexity will be $O(L^2K)$, where L is the length of the unfolded RNA sequence (the "target" sequence) and K is the length of the consensus RNA sequence (the "reference" sequence) with a known structure. Note that K is proportional to the number of states in the profile-csHMM. The complexity for aligning RNAs with multiple stem-loops will be $O(L^3K)$, and the complexity for aligning most known pseudoknots will be $O(L^4K)$.[2] In general, the complexity of the SCA algorithm for analyzing a given RNA secondary structure is identical to or less than that of an existing algorithm (such as the Viterbi, CYK, and PSTAG algorithms) that can be used to analyze the same structure.

4.4 Structural alignment of RNA pseudoknots

To demonstrate the effectiveness of the proposed method, we have built a program that can be used for structural alignment of RNA sequences including pseudoknots. Similar to the PSTAG-

[2]The complexity for aligning pseudoknots in the Rivas and Eddy class [86] will be at most $O(L^6K)$. For RNAs outside the R&E class, the computational complexity can be higher.

based alignment tool developed by Matsui et al. [66], it uses a single structured RNA sequence as a reference and aligns unfolded RNA sequences to it. However, unlike PSTAGs that can only handle pseudoknots with 2-crossing property, the given program can deal with a much larger class of RNAs (i.e., the so-called Rivas and Eddy class [17, 86]).

4.4.1 Building an alignment tool using the profile-csHMM

The program proceeds as follows. It first constructs a profile-csHMM based on the reference RNA and its structural annotation. Instead of using a fully stochastic model with position-dependent emission probabilities, in this implementation, we have used the non-stochastic scoring matrix suggested by Gorodkin et al. [45], as it has been used by several RNA analysis tools with good performance.

Secondly, the program automatically finds the adjoining order that can be used for predicting the optimal state sequence of the profile-csHMM. The adjoining order is obtained in the following way. Let us consider a subsequence $\hat{\mathbf{x}}$ of the reference RNA that consists of $I\ (\leq I_{\max})$ intervals. We define $\Gamma(\hat{\mathbf{x}})$ as the number of base pairs that are fully contained in the subsequence $\hat{\mathbf{x}}$. For a given $\hat{\mathbf{x}}$, we try to split it into two subsequences $\hat{\mathbf{x}}_\ell$ and $\hat{\mathbf{x}}_r$, where each of them may have up to $I_{\max}$ intervals, such that $\Gamma(\hat{\mathbf{x}}) = \Gamma(\hat{\mathbf{x}}_\ell) + \Gamma(\hat{\mathbf{x}}_r)$ is maximized. We begin this process by finding the best "division" for the entire sequence, and proceed in a recursive manner until we reach the point where every subsequence consists of a single base or a single base pair. The adjoining order of the profile-csHMM can be simply obtained by reversing this division process.

Currently, the program can deal with RNA secondary structures which can be handled by the SCA algorithm using subsequences with up to two intervals ($I_{\max} = 2$). This corresponds to the entire class of RNA secondary structures that can be represented by the grammar proposed by Rivas and Eddy [86]. The so-called Rivas and Eddy class is regarded as the most general RNA class that is known today, and it covers almost all RNA secondary structures that have been identified until now [17]. Note that although the current implementation of the structural alignment program covers only the R&E class, it has to be noted that the capability of the profile-csHMMs and the SCA algorithm goes beyond the R&E class. For RNAs that are outside the R&E class, we can easily extend the current program to handle them.

One advantage of the given program is that it does not reject RNAs even if they are outside the descriptive capability of the current implementation. For example, if the reference RNA has a

complex structure that is outside the R&E class, the program chooses the subset with maximum number of base pairs such that the resulting secondary structure is contained in the R&E class.

Now, the constructed profile-csHMM can be used for carrying out a structural alignment between the reference RNA and a target RNA with unknown structure. We can follow the adjoining order that has been obtained in the previous step to find the optimal state sequence of the target RNA, which in turn yields the prediction of its secondary structure.

4.4.2 Restricting the search region

In implementing the SCA algorithm, we have introduced a parameter D, which is the length of the search region for finding the matching bases in the sequence alignment. This is motivated by the following observation. When we align two RNA sequences that are biologically relevant, the matching bases in the reference RNA and the target RNA are usually located very close to each other. Figure 4.6 (a) shows an example of a typical sequence alignment, where the maximum distance between a base in the reference sequence and the matching base in the target sequence is two. Alignments with a large distance between the matching bases, as the one shown in Figure 4.6 (b), are generally less probable. Based on this observation, we limit the search region as illustrated in Figure 4.7. When looking for the base x_ℓ in the target RNA $\mathbf{x} = x_1 \dots x_L$ that matches the kth base in the reference RNA, we only consider the bases between $\max(k - d_1, 1)$ and $\min(k + d_2, L)$, hence the maximum length of the search region is $D = d_1 + d_2 + 1$.

Restricting the search region has several advantages. First of all, it significantly reduces the overall complexity of the alignment algorithm, making the program practically usable in real applications. The computational complexity of aligning pseudoknots will be reduced from $O(L^4K)$ to $O(D^4K)$ (or from $O(L^6K)$ to $O(D^6K)$ in the worst case). For example, assume that we want to align two pseudoknots in the TOMBUS_3_IV RNA family (seed alignment) in the Rfam database [47]. The average length of these RNAs is around $L = 91$, and $D = 7$ is enough for finding the optimal alignment between any two members in the given family. In this case, limiting the search region reduces the overall complexity to around $(D/L)^4 \approx 0.35\%$ of the original. Secondly, when the structural alignment score obtained from the SCA algorithm is used for finding homologues of an RNA family, restricting the search region can yield better discrimination between homologues and non-homologues, as long as D is large enough to obtain the optimal alignment. As the optimal alignment of homologues is contained within the search space, the imposed restriction does not

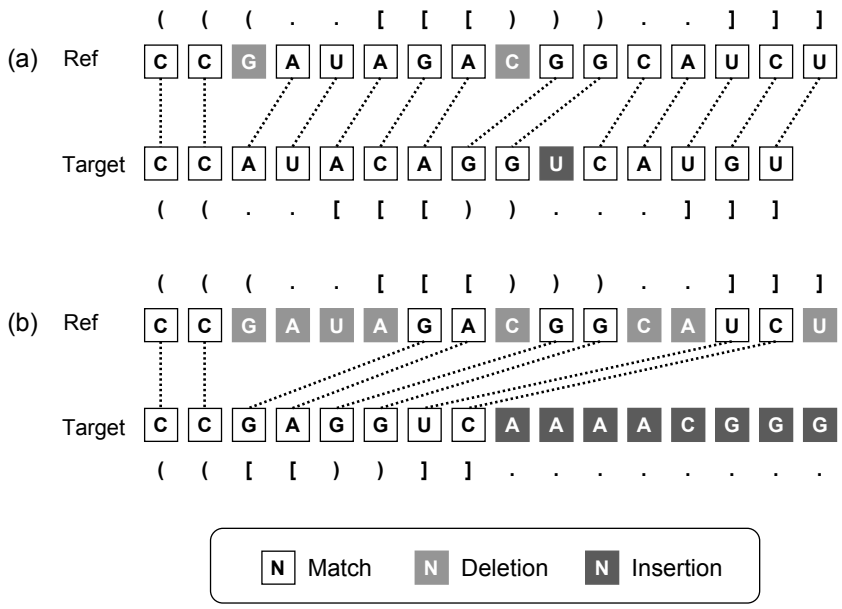


Figure 4.6: Matching bases in an RNA sequence alignment. (a) The maximum distance between the matching bases is two. (b) The maximum distance between the matching bases is seven.

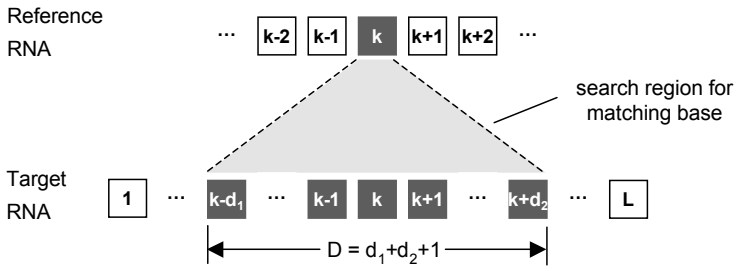


Figure 4.7: Limiting the search region for finding the matching bases can significantly reduce the overall complexity of the structural alignment.

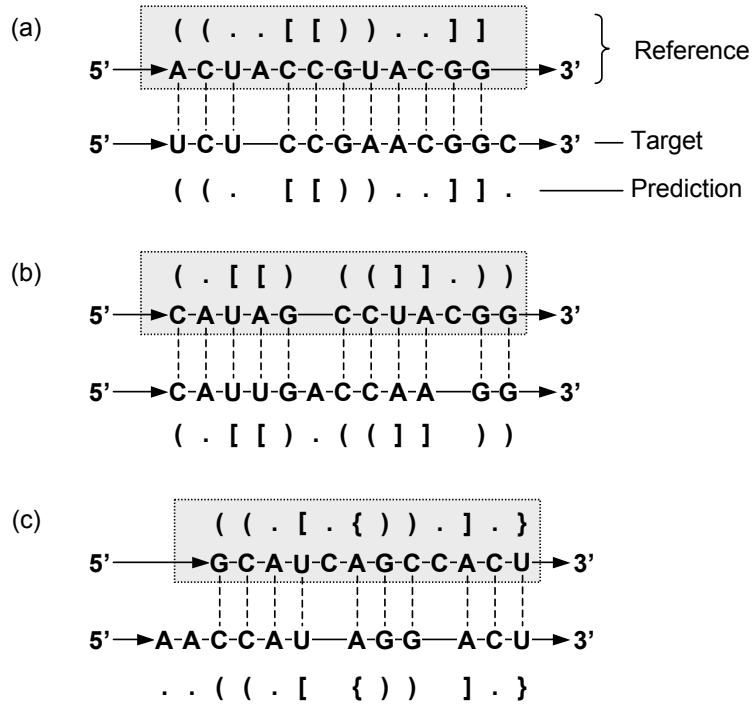


Figure 4.8: Structural alignments of RNAs with various secondary structures.

affect the alignment score of homologous sequences. However, limiting the search region will lead to an overall decrease in the alignment score of non-homologues, hence providing better discrimination between homologues and non-homologues.

A good way of choosing D is to compute the range between the matching bases in the original sequence alignment, and make it slightly larger than this range. Another method for estimating D is to construct a simple profile-HMM from the sequence alignment and find a (sequence-based) alignment between the target sequence and the profile-HMM. This alignment can be found very quickly, since the computational complexity for aligning a profile-HMM is only $O(LK)$. Then we compute the maximum distance between the matching bases in the given alignment, and use it to estimate D. This is indeed a very efficient strategy that results in a tremendous reduction in the CPU time needed for finding the structural alignments, while providing a good prediction performance, as will be demonstrated in Section 4.4.4.

4.4.3 Examples of structural alignments

Figure 4.8 shows a few examples of structural alignments obtained from the program that has been just described. RNAs illustrated in Figure 4.8 (a) and Figure 4.8 (b) have 2-crossing property and Figure 4.8 (c) has a 3-crossing property. In each example, a target RNA with unknown structure is aligned to the reference RNA whose structure is known. As we can see in this example, the proposed approach can find good structural alignments for RNAs with various secondary structures.

4.4.4 Experimental results

We tested the performance of our program using several pseudoknots included in the *Rfam database* [47]. The Rfam database provides a large collection of various RNA families, where the member sequences in each family are aligned to each other. In our experiments, we have used the sequences in the "seed alignment" of each RNA family, as they are hand curated and have reasonably reliable structural annotation. For each sequence family, we chose one of its members as the reference RNA, and used it along with its structural annotation to predict the secondary structure of all the other sequences in the same family. The predicted secondary structure has been compared to the annotated structure in the database, and we counted the number of correctly predicted base pairs (*true-positives*; TP), the number of incorrectly predicted base pairs (*false positives; FP*), and the number of base pairs in the annotated structure that were not predicted by the program (*false negatives; FN*). These numbers have been used to compute the *sensitivity (SN)* and the *specificity (SP)* of the program that are defined as follows

$$\mathrm{SN} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \quad \mathrm{SP} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{4.2}$$

To obtain reliable estimates of these quantities, we performed a cross-validation experiment by repeating the previous process for every member in the given RNA family and computed the overall prediction ratios.

In order to compare the prediction performance of the proposed method with the performance of PSTAGs, we first tested the program for three RNA families, CORONA_PK3, HDV_RIBOZYME, and TOMBUS_3_IV, which have all pseudoknot structures. Table 4.1 shows the prediction result of the proposed method along with the prediction result of PSTAGs. In each case, the higher prediction ratio is shown in bold. As we can see in Table 4.1, profile-csHMMs yielded accurate

	Profile-csHMM		PSTAG	
	SN (%)	SP (%)	SN (%)	SP (%)
CORONA_PK3	**95.7**	**96.5**	94.6	95.5
HDV_RIBOZYME	**94.5**	95.3	94.1	**95.6**
TOMBUS_3_IV	96.9	96.9	**97.4**	**97.4**
FLAVI_PK3	**94.5**	**96.4**	-	-

Table 4.1: Prediction results of the proposed method for several RNA families with pseudoknot structures. The prediction results of PSTAGs are also shown for comparison. The prediction results of PSTAGs are obtained from [66].

	Average CPU Time	
	Profile-csHMM	PSTAG
CORONA_PK3	1.24 sec	37.18 sec
HDV_RIBOZYME	1.72 sec	207.46 sec
TOMBUS_3_IV	0.95 sec	270.93 sec
FLAVI_PK3	6.77 sec	-

Table 4.2: CPU time for aligning two sequences in each RNA family. The experiments have been performed on a PowerMac G5 Quad 2.5 GHz with 4 GB memory.

prediction results that are comparable to PSTAGs for all three RNAs that have been tested. For CORONA_PK3, the proposed approach yielded higher sensitivity and specificity than the PSTAG algorithm, whereas the PSTAG algorithm worked slightly better for TOMBUS_3_IV. For the RNA family HDV_RIBOZYME, the profile-csHMM had higher sensitivity while the PSTAG had higher specificity. But profile-csHMMs are more general than the PSTAGs as we demonstrate in the next example.

Second, we tested the performance of the proposed method for the FLAVI_PK3 family that has a more complex secondary structure than the previous RNA families, which cannot be handled by the PSTAGs. The secondary structure of FLAVI_PK3 is similar to the example shown in Figure 4.8 (b), and it has two stems and additional base pairs that cross the base pairs in both stems. As we have already seen in Figure 4.8 (b), profile-csHMMs are capable of dealing with such structures. Figure 4.9 shows a structural alignment of two RNAs in the FLAVI_PK3 family obtained using the proposed approach. Note that most base pairs have been correctly predicted. There were two false positives and a false negative in the predicted structure when compared to the annotated structure in Rfam.

Despite the generality of the proposed method, its computational cost was much smaller than that of the PSTAGs. Table 4.2 shows the average CPU time that was needed for finding the struc-

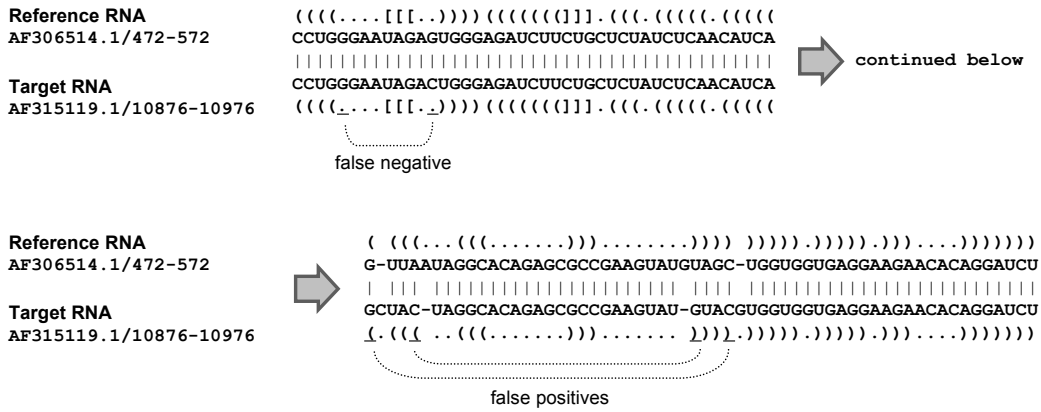


Figure 4.9: Structural alignment of two RNAs in the FLAVI_PK3 family. The secondary structure of the target RNA has been predicted from the given alignment. Incorrect predictions (one false negative and two false positives) have been underlined.

tural alignment between two sequences in each RNA family. In our experiments, the parameter D was automatically estimated by performing a simple sequence-based alignment. We first constructed a conventional profile-HMM based on the reference RNA, and aligned the target RNA to the given profile-HMM. Then we computed the maximum distance between the matching bases and used it to estimate D. It has to be noted that the initial alignment obtained by the profile-HMM is only used for estimating D, hence it does not affect the final structural alignment of the profile-csHMM.

The prediction performance of the proposed method does not strongly rely on the sequence similarity. In order to show this, we randomly mutated the RNA sequences that were used in the previous experiments, such that the sequence similarity between the homologous RNAs got completely removed. During this process, bases that form complimentary base pairs were covaried to preserve the original secondary structure. The experimental results are summarized in Table 4.3. As we can see in Table 4.3, the prediction ratios have decreased only slightly, indicating that the proposed approach does not depend too much on sequence similarity.

	Profile-csHMM	
	SN (%)	SP (%)
CORONA_PK3	90.4	90.7
HDV_RIBOZYME	91.3	91.5
TOMBUS_3_IV	93.3	93.4
FLAVI_PK3	87.8	88.5

Table 4.3: Prediction results of the proposed method using randomly mutated RNA sequences.

4.5 Fast search using prescreening filters

Although the structural alignment algorithm proposed in Section 4.4 runs relatively fast, it is still too slow for scanning a large database. Recently, Weinberg and Ruzzo suggested the use of heuristic profile-HMM filters to expedite CM-based searches [122]. They showed that using such filters can make the scanning speed significantly faster with virtually no loss of performance. They also proposed rigorous filters that are slower than the heuristic filters but guarantee no loss in the prediction accuracy [121]. In a similar manner, it is possible to incorporate profile-HMM based prescreening filters to speed up the database search based on profile-csHMMs. In this section, we elaborate how we can construct such a prescreening filter based on a given profile-csHMM. The content of this section is mainly drawn from [134].

4.5.1 Searching for similar sequences

Assume that we have constructed a profile-csHMM that reflects the common characteristics of an RNA sequence family. Now, this model can be used to look for 'similar' sequences in a database. An essential problem in performing a similarity search is how we can quantitatively measure the similarity between a new observed sequence and the statistical model at hand. A widely used approach is to compute the optimal probability of the observation based on the given model, and use it as a similarity measure.

Let $\mathbf{x} = x_1 \dots x_L$ be an observed symbol sequence and let us denote its underlying state sequence as $\mathbf{y} = y_1 \dots y_{L_s}$. Note that the length of the state sequence L_s can be larger than the length L of the observed sequence, when there exist deleted symbols. We also define Θ, which is the set of model parameters of the profile-csHMM at hand. The *similarity score* $S(\mathbf{x}, \Theta)$ between the observation $\mathbf{x}$ and the profile-csHMM can be computed as follows

$$S(\mathbf{x}, \Theta) = \max_{\mathbf{y}} S(\mathbf{x}, \mathbf{y}|\Theta) = S(\mathbf{x}, \mathbf{y}^*|\Theta), \tag{4.3}$$

where $S(\mathbf{x}, \mathbf{y}|\Theta)$ is the score for $\mathbf{x}$ whose underlying state sequence is $\mathbf{y}$. Note that $\mathbf{y}^*$ is the optimal state sequence that maximizes the similarity score $S(\mathbf{x}, \mathbf{y}|\Theta)$. If this similarity score is larger than a predefined threshold λ such that $S(\mathbf{x}, \Theta) \geq \lambda$, we can view the observed sequence as a good candidate that is likely to be a new member of the same sequence family. On the contrary, if $S(\mathbf{x}, \Theta) < \lambda$, we can conclude that $\mathbf{x}$ is unlikely to be a member of the given family. Therefore, when we search

a database to find new members, only those sequences that satisfy $S(\mathbf{x}, \Theta) \geq \lambda$ will be reported as a "match."

When using profile-csHMMs to represent sequence families, we can utilize the *sequential component adjoining (SCA) algorithm* [129], which was proposed in Section 4.3 for finding $\mathbf{y}^*$ and computing $S(\mathbf{x}, \Theta)$. As mentioned earlier, the computational complexity of the SCA algorithm is variable, and it depends on the correlation structure of the profile-csHMM [129]. For example, the complexity for computing $S(\mathbf{x}, \Theta)$ for typical RNA pseudoknots ranges between $O(L^4)$ and $O(L^6)$, which can be very large for long RNA sequences.

One advantage of using profile-csHMMs in a similarity search is the increased specificity. When computing the similarity score, profile-csHMMs combine contributions from sequence similarity as well as structural similarity (in terms of symbol correlations). This makes it possible to reject false candidates that look similar to the reference sequences in the sequence level, but do not preserve the original correlation structure.

However, when performing a similarity search, there will be typically many sequences that look very different from the reference sequences in the sequence level, such that their similarity scores cannot exceed the threshold λ even after combining the contributions from their structural similarity. As the measure of sequence level similarity can be quickly computed using a simpler model, such as the profile-HMM, we do not have to use a profile-csHMM in such cases.

Based on this observation, we propose a practical strategy that can make the database search much faster, compared to the search based on profile-csHMM alone. The proposed approach is as follows. In the first place, we construct a prescreening filter using a profile-HMM. The profile-HMM will have the same size as the original profile-csHMM, and its model parameters Θ_p will be derived from the parameters Θ of the profile-csHMM. When given a new observation sequence, we first compute the sequence level similarity score $S_p(\mathbf{x}, \Theta_p)$ using the prescreening filter. This score is compared with a new threshold λ_p to decide whether the overall similarity score $S(\mathbf{x}, \Theta)$ can exceed the original threshold λ after combining the contributions from the structural similarity. If this is possible, the sequence is handed over to the full-blown profile-csHMM to compute $S(\mathbf{x}, \Theta)$. Otherwise, the observation will be rejected. The overall algorithm is illustrated in the flow chart shown in Figure 4.10.

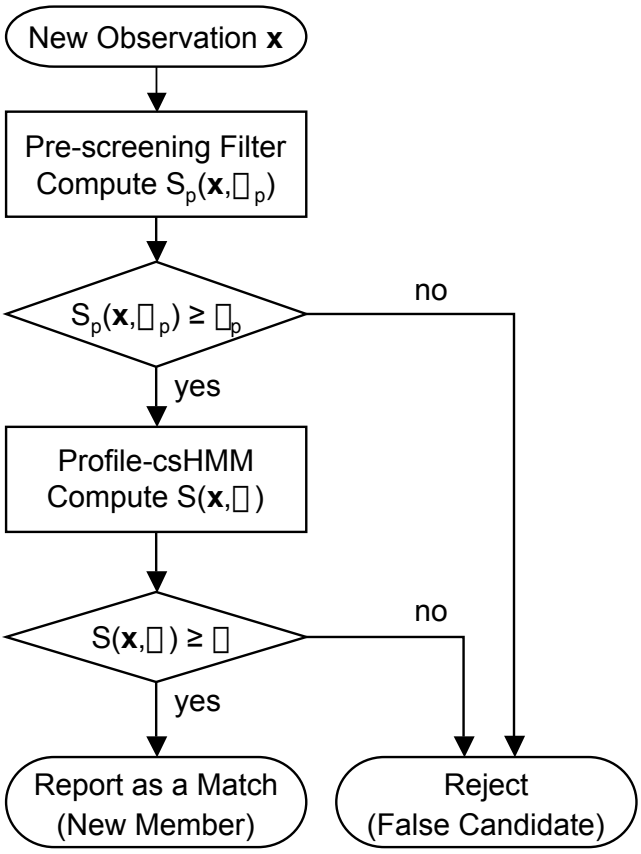


Figure 4.10: Illustration of the proposed algorithm.

4.5.2 Constructing the prescreening filter

Now the question is how to choose the model parameters of the profile-HMM and how we should choose the threshold λ_p so that there will be no degradation in the prediction accuracy. In order to answer this question, let us first define several notations. Firstly, we define the set $\mathcal{K} = \{k|\ M_k \text{ is a context-sensitive state}\}$. At single-emission states and pairwise-emission states, we denote the emission score of a symbol x at state v as $s_e(x|v)$. At context-sensitive states, the emission score of a symbol x_c at state v is denoted as $s_e(x_c|v, x_p)$, where x_p is the symbol that was previously emitted at the corresponding pairwise-emission state. A typical choice of the emission scores would be the logarithm of the emission probabilities, but we can also use other scoring schemes. The transition score from state v to state w is defined as $s_t(v, w|m)$ for $w = D_k, M_k, I_{k-1}$, where $k \in \mathcal{K}$. The variable $m \in \{0, 1\}$ indicates whether the memory associated with the context-sensitive

state M_k is empty ($m = 0$) or not ($m = 1$). For all other w, the transition score is simply defined as $s_t(v, w)$. A typical choice for the transition scores would be the logarithm of the transition probabilities, but we can also use other scores.

Based on the parameters of the original profile-csHMM defined above, we choose the parameters of the prescreening filter as follows. In the first place, the emission scores are chosen as

$$s_e^p(x|v) = \begin{cases} \min_{x_p}[s_e(x|v, x_p)] & \text{for } v = M_k \ (k \in \mathcal{K}), \\ s_e^p(x|v) = s_e(x|v) & \text{for other emitting states } v. \end{cases}$$

We also define $\Delta_e(k)$ for $k \in \mathcal{K}$ as follows

$$\Delta_e(k) = \max_x \left(\max_{x_p}[s_e(x|v, x_p)] - \min_{x_p}[s_e(x|v, x_p)] \right). \tag{4.4}$$

In the second place, the transition score of the profile-HMM for a transition from state v to state w is chosen as follows

$$s_t^p(v, w) = \begin{cases} s_t(v, D_k|m = 0) & w = D_k \ (k \in \mathcal{K}), \\ s_t(v, M_k|m = 1) & w = M_k \ (k \in \mathcal{K}), \\ \min_m s_t(v, I_{k-1}|m) & w = I_{k-1} \ (k \in \mathcal{K}), \\ s_t(v, w) & \text{otherwise.} \end{cases}$$

In addition to this, we define $\Delta_t(k)$ for $k \in \mathcal{K}$

$$\Delta_t(k) = \left(\max_m[s_t(v, I_{k-1}|m)] - \min_m[s_t(v, I_{k-1}|m)] \right). \tag{4.5}$$

Finally, we choose the threshold of the prescreening filter to be $\lambda_p = \lambda - \Delta$, where $\Delta = \sum_{k \in \mathcal{K}} [\Delta_e(k) + \Delta_t(k)]$.

4.5.3 No degradation in the prediction accuracy

Based on the prescreening filter constructed as described in Section 4.5.2, we can compute the sequence level similarity score as follows

$$S_p(\mathbf{x}, \Theta_p) = \max_{\mathbf{y}} S_p(\mathbf{x}, \mathbf{y}|\Theta_p) = S_p(\mathbf{x}, \mathbf{y}^*|\Theta_p), \tag{4.6}$$

where $\mathbf{y}^*$ is the optimal state sequence. Using the score in (4.6) with the threshold λ_p guarantees that there will be no loss in the prediction accuracy. This can be shown as follows.

Theorem For an observed sequence $\mathbf{x}$, if the score $S_p(\mathbf{x}, \Theta_p)$ computed from the prescreening filter is smaller than λ_p, its score $S(\mathbf{x}, \Theta)$ from the original profile-csHMM cannot exceed λ.

Proof If $S_p(\mathbf{x}, \Theta_p) < \lambda_p$, we have

$$\max_{\mathbf{y} \in \mathcal{Y}} S_p(\mathbf{x}, \mathbf{y}|\Theta_p) \leq \max_{\mathbf{y}} S_p(\mathbf{x}, \mathbf{y}|\Theta_p) < \lambda_p,$$

where $\mathcal{Y}$ is the set of all feasible state sequences in the original profile-csHMM. Then we have

$$\begin{aligned} S(\mathbf{x}, \Theta) &= \max_{\mathbf{y}} S(\mathbf{x}, \mathbf{y}|\Theta) \\ &= \max_{\mathbf{y} \in \mathcal{Y}} S(\mathbf{x}, \mathbf{y}|\Theta) \\ &\leq \underbrace{\max_{\mathbf{y} \in \mathcal{Y}} S_p(\mathbf{x}, \mathbf{y}|\Theta_p)}_{<\lambda_p} + \underbrace{\max_{\mathbf{y} \in \mathcal{Y}} \Big[S(\mathbf{x}, \mathbf{y}|\Theta) - S_p(\mathbf{x}, \mathbf{y}|\Theta_p) \Big]}_{\leq \Delta} \\ &< (\lambda - \Delta) + \Delta = \lambda. \end{aligned}$$

■

This shows that the prescreening filter will reject only those sequences that are guaranteed to be rejected by the original profile-csHMM, hence there will be no degradation in the prediction accuracy.

4.5.4 Experimental results

To demonstrate the proposed idea, we carried out an experiment using real RNA sequences. We first constructed a profile-csHMM for the CORONA-PK3 RNA family in the Rfam database [47]. We used the "seed alignment" for constructing the model, as it provides a reasonably reliable structural annotation of the RNA family. Note that the secondary structure of CORONA-PK3 contains pseudoknots, hence they cannot be modeled using CMs (or SCFGs). Based on the constructed profile-csHMM, we have built a profile-HMM prescreening filter by following the procedure elaborated in Section 4.5.2.

After constructing the models, we evaluated the performance of the profile-csHMM search and

that of the proposed prescreening approach. For evaluation, we used a database that consists of real CORONA-PK3 RNA sequences and 10,000 random RNA sequences. As expected, the prescreening filter did not miss any RNA that was reported as a "match" by the original profile-csHMM. Consequently, the prediction accuracies of both methods were identical. The average CPU time used by the prescreening filter to compute the similarity score $S_p(\mathbf{x}, \Theta_p)$ was 0.0093 sec, which is much smaller than 28.1 sec of the profile-csHMM.[3] The rejection rate of the prescreening filter was around 98.8%, hence only 1.2% of the inspected RNAs was passed to the profile-csHMM in the second stage. As a result, the average CPU time used by the proposed method was around 0.34 sec, which is around eighty times faster than the search method based on a profile-csHMM alone.

It is important to note that the rejection rate of the prescreening filter has a crucial impact on the overall reduction in the search time. Ideally, the prescreening filter should reject most sequences that will be rejected by the profile-csHMM, and pass only a small fraction to the second stage for further inspection. However, there can be also occasions when the rejection rate is quite small, in which case the reduction in the search time will not be significant. For many applications, the criterion used in Section 4.5.2 for deriving the parameters of the prescreening filter and the threshold λ_p will be too stringent, and it may be beneficial to relax it a little bit to make the search faster, at a slight loss of the prediction accuracy.

4.6 Conclusion

In this chapter, we proposed the concept of profile-csHMMs that can provide an effective framework for representing RNAs with various secondary structures including pseudoknots. Based on profile-csHMMs, we proposed a structural alignment algorithm that can be used for aligning pseudoknots and predicting their secondary structures. Experimental results indicate that the prediction accuracy of the profile-csHMM approach is comparable to the state-of-the-art method. In addition to this, profile-csHMMs can handle a considerably larger class of secondary structures than the existing models at a low computational cost.

The good prediction performance of the proposed scheme, as well as its generality and the relatively low computational cost makes profile-csHMMs an attractive choice for building homology search tools for noncoding RNAs. For example, we can build a family-specific prediction program similar to the tRNA CM [26], which finds new candidates that may belong to the given RNA fam-

[3] We used a fixed search region size of $D = 7$.

ily. Although we have used a non-stochastic scoring matrix in our structural alignment algorithm described in Section 4.4, we can also use a fully stochastic model with position-dependent probabilities to improve the specificity of the prediction. These probabilities can be easily obtained from the multiple sequence alignment of the RNA family that is under consideration.

Another interesting application would be to build a BLAST-like tool that uses a single RNA with a known structure for finding structural homologues. Klein and Eddy developed a database search program called *RSEARCH* [57] that finds homologues of single structured RNAs, and they showed that it outperforms primary sequence-based programs (including *BLAST*) in many cases. As RSEARCH is based on covariance models, they cannot be used for finding pseudoknots. We can develop a more general search program based on profile-csHMMs that can practically deal with any kind of RNA secondary structures.

When implementing a homology search algorithm based on profile-csHMMs, we can incorporate prescreening filters as proposed in Section 4.5, in order to make the search faster. As shown in Section 4.5.2, this prescreening filter can be constructed using a simple profile-HMM whose parameters are chosen based on the parameters of the original profile-csHMM. This prescreening filter rejects only those sequences that are guaranteed to be rejected by the original profile-csHMM. This leads to a considerable reduction in the overall search time without any degradation in the prediction accuracy of the search. We may improve the search speed even further by optimizing the parameters of the prescreening filter or by applying heuristic methods for choosing the filter parameters that will make the search even faster with a small trade-off in the prediction accuracy.

Chapter 5

Predicting Protein-Coding Genes Using Digital Filters

Analysis of various genomes has revealed that there exist strong period-3 patterns in the protein-coding regions of DNA sequences [15, 37, 106, 107]. Such periodic patterns have been observed in the coding region of various organisms, including both simple organisms such as *S. cerevisiae* (budding yeast) and *C. elegans* (nematode), and also complex organisms such as humans. Research shows that this periodicity is frequently observed in coding regions, while non-coding regions rarely contain such periodic components [106]. One explanation for the period-3 behavior in protein-coding sequences is the *codon*, which is a triplet of nucleotides. The protein synthesis process is governed by the so-called "genetic code," which is a set of rules that is used to translate the information encoded in DNA sequences into proteins, where a protein is a sequence of amino acids. During the synthesis process, a codon is mapped into an amino acid. Codon composition bias in coding-regions is viewed as an important source of the 3-periodicity, while the bias in amino acid composition among naturally occurring proteins may also have a significant effect on this phenomenon [106, 137].

Although there are exceptions [106], the period-3 property is generally viewed as a reasonable indicator of protein-coding genes. As the discrete Fourier transform (DFT) can effectively separate a signal into components with different frequencies (or periods), there have been several techniques based on DFT that exploited this property to identify coding regions [3, 20, 106].

In a similar manner, we can use digital filters for predicting protein-coding genes [48, 92, 110, 111, 112]. Digital signal filtering techniques provide more efficient ways to identify coding regions than the conventional DFT-based approaches. In this chapter, we propose simple and efficient gene prediction schemes based on digital filters. The content of this chapter has been mainly drawn

from [110, 111, 112].

5.1 Outline

In Section 5.2 we elaborate on the period-3 property that is commonly observed in many protein-coding regions, and show that this property can be used for identifying these regions in DNA molecules. The putative source of this behavior is also briefly discussed.

In Section 5.3, we review the conventional DFT-based prediction methods that have been proposed by several researchers [3, 106]. In order to use the DFT approach, we need to convert the DNA sequence into a numerical sequence. The indicator sequence for each of the four bases (A, C, G, T), which is defined in Section 5.3.1, can be used for this purpose. In Section 5.3.2, we review the concept of DNA spectrogram that can be used for isolating the period-3 components. Relation between the DFT approach and digital filtering is given in Section 5.3.3.

Based on the observation in Section 5.3.3, we propose a new gene prediction method in Section 5.4, which uses antinotch filters. As will be shown in Section 5.4, the antinotch filter approach is computationally very efficient and also provides a good prediction performance. In Section 5.4.1, we first consider how such filters can be obtained. Then the efficient implementation of such antinotch filters using the lattice structure is considered in Section 5.4.2. Experimental results based on a real DNA sequence is shown in Section 5.4.3, which demonstrates the effectiveness of the proposed method.

The prediction results can be further improved by using digital filters that are designed using the multistage filtering approach, which is introduced in Section 5.5. We first show in Section 5.5.1 how we can design a good bandpass filter using this approach. An example of a low complexity implementation of a multistage filter that can be used for identifying the coding regions is given in Section 5.5.2. Finally, we compare the performance of the DFT approach, the antinotch filter approach, and the multistage filtering approach in Section 5.5.3.

Concluding remarks are given in Section 5.6, where we briefly mention other gene identification schemes and also suggest how the prediction methods proposed in Section 5.4 and Section 5.5 can be incorporated into them.

5.2 Period-3 patterns in protein-coding regions

As mentioned earlier, the protein-coding regions of many DNA sequences exhibit a period-3 pattern [15, 37, 106, 107]. Since this periodicity is rarely observed in non-coding regions, such as introns and intergenic regions, the period-3 property can be used for discriminating between coding and non-coding regions. Although it is obvious that this periodicity comes from the fact that the codon consists of three nucleotides, it is not so clear why this periodicity is mainly observed in coding regions.

There have been several explanations for the source of this periodicity. For example, some researchers have attributed this behavior to the nonuniform usage of codons. As 64 codons are mapped into 20 different amino acids, there exist amino acids that can be coded by two or more distinct codons. This is commonly referred as the *degeneracy of the genetic code*. Although multiple codons can encode the same amino acid, not all of them are used with equal frequency. In fact, the codon usage is generally not uniform, resulting in a codon composition bias in coding-regions. However, the work by Tiwari et al. [106] indicate that the codon bias may not be the primary source of the period-3 behavior. In fact, experimental results in [106] suggest that a more important source of this periodicity may be the biased usage of amino acids in naturally occurring proteins [137].

Although the period-3 patterns are frequently observed in the coding regions of DNA sequences, there are also exceptions. For example, it has been reported in [106] that the genes of the mating-type locus of *S. cerevisiae* and the genes of the amoebapores of *E. histolytica* lack this property.

5.3 Finding genes from the DNA spectrum

As the period-3 property provides a reasonable preliminary indicator of protein-coding regions, there have been several methods based on DFT (discrete Fourier transform) that exploit this property to identify coding regions in a DNA sequence [3, 106]. In this section, we briefly review the basic concept of the DFT-based gene prediction algorithms.

5.3.1 Indicator Sequence

In order to use the DFT approach, we first have to convert the DNA sequence into a numerical sequence so that we can compute the DFT coefficients of its segments. For this purpose, we define the *indicator sequences* for the four bases A, C, G, and T as follows. As an example, let us consider

the following DNA sequence

$$d(n) = \text{A C A G G T T A C G C C T A G G } \ldots$$

For a base "B," we define its indicator sequence $x_B(n)$ of the symbol sequence $d(n)$ as follows

$$x_B(n) = \begin{cases} 1 & \text{if } d(n) = \text{B}, \\ 0 & \text{otherwise.} \end{cases}$$

Following this definition, the indicator sequences $x_B(n)$ for the fours bases B∈{A, C, G, T} can be defined as follows

$$\begin{aligned} x_A(n) &= 1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0 \\ x_C(n) &= 0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0 \\ x_G(n) &= 0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1 \\ x_T(n) &= 0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,. \end{aligned}$$

Note that

$$x_A(n) + x_C(n) + x_G(n) + x_T(n) = 1, \quad \forall\, n$$

from the definition of $x_A(n)$, $x_C(n)$, $x_G(n)$, and $x_T(n)$.

5.3.2 Using DFT for detecting the period-3 patterns

Let us consider a length-N block of an indicator sequence $x_B(n)$. For simplicity, we assume that this block is located in $0 \le n \le N-1$. The DFT of $x_B(n)$ is defined as

$$X_B[k] = \sum_{n=0}^{N-1} x_B(n) e^{-j2\pi kn/N},$$

for $0 \le k \le N-1$. The DFT coefficient $X_B[k]$ gives us a measure of the strength of the signal components in the frequency band around $\omega = 2\pi k/N$. After computing the DFTs $X_A[k], X_C[k], X_G[k]$, and $X_T[k]$ of the four indicator sequences, we combine them to compute the DNA spectrogram $S[k]$

$$S[k] \triangleq |X_A[k]|^2 + |X_C[k]|^2 + |X_G[k]|^2 + |X_T[k]|^2.$$

In order to compute the strength of the period-3 component (whose frequency is $f = 1/3$), we can simply look at the value of the spectrogram at the point $k = N/3$. In order to have an integer value for $k = N/3$, the length N is chosen to be a multiple of 3. If the original DNA sequence has a strong

period-3 component, $S[N/3]$ will be large, and if this periodic component is absent, $S[N/3]$ will have a relatively small value. Therefore, if the DNA belongs to a coding region, $S[k]$ is likely to have peak at $k = N/3$, while such a peak is not observed in non-coding regions. Such behaviors have been demonstrated in many papers [3, 106], although the strength of the peak highly depends on the respective gene. The peak at $k = N/3$ can be sometimes very pronounced, and sometimes quite weak.

If we slide the length-N window by one or more bases and plot the value of $S[N/3]$ for each location, we can obtain an evolving picture of how $S[N/3]$ changes over different locations in the DNA sequence. This picture allows us to infer the locations of the protein-coding regions. It is important to make the window size N sufficiently large so that the peak due to the periodicity dominates the noisy background. For example, the methods proposed in [3, 106] both use $N = 351$. However, we cannot make the window size too large, since using a longer window requires more computations and also degrades the resolution for identifying the start and end positions of the exons.

5.3.3 Relation to digital filtering

The DFT-based sliding window method in the previous section can be viewed as digital filtering followed by a decimator, where the decimation ratio depends on the separation between the adjacent locations of the window [110, 112]. The corresponding digital filter has a very simple impulse response $w(n)$ as follows

$$w(n) = \begin{cases} e^{j\omega_0 n} & 0 \le n \le N-1, \\ 0 & \text{otherwise}, \end{cases}$$

where $\omega_0 = 2\pi/3$. This is a bandpass filter with a passband centered at $\omega_0 = 2\pi/3$. The magnitude response of $w(n)$ is shown in Figure 5.1. As we can see in Figure 5.1, the minimum stopband attenuation of this filter is around 13 dB, which is not very high. If we design a better bandpass filter $h(n)$ that has a higher stopband attenuation than the simple DFT window $w(n)$, we will be able to isolate the period-3 component from the rest of the signal more effectively. Furthermore, we can also use efficient methods for designing and implementing such filters that can reduce the overall computational complexity.

Let us assume that we have designed a bandpass filter $H(z)$, whose passband is centered at

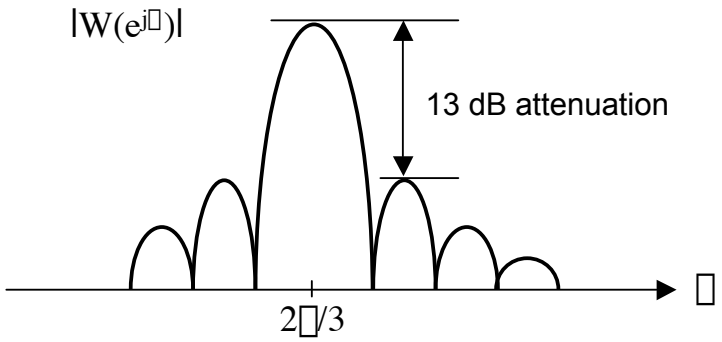


Figure 5.1: The magnitude response of the sliding window $w(n)$.

$\omega_0 = 2\pi/3$. We can pass the indicator sequence $x_B(n)$ ($B \in \{A, C, G, T\}$) through $H(z)$ to obtain

$$y_B(n) = h(n) * x_B(n).$$

In the coding regions, we expect $x_B(n)$ to contain a period-3 pattern, hence most of its energy is concentrated in the passband of $H(z)$. Consequently, $y_B(n)$ is expected to have a large magnitude in the coding regions. On the contrary, in non-coding regions, where $x_B(n)$ does not have a period-3 pattern, most of the energy of the indicator signal $x_B(n)$ will be suppressed by $H(z)$. So, the magnitude of $y_B(n)$ will be small in these regions. This idea is illustrated in Figure 5.2. Based on the output signals for the four indicator sequences, we define

$$Y(n) = |y_A(n)|^2 + |y_C(n)|^2 + |y_G(n)|^2 + |y_T(n)|^2.$$

As the DFT coefficient $S[N/3]$ could be used for predicting the coding regions, we can use $Y(n)$ as a preliminary indicator of these regions.[1]

Now, the question is how we can design a bandpass filter $H(z)$ that has a narrow passband centered at $\omega_0 = 2\pi/3$ and also has a low complexity. In the following sections, we introduce some efficient methods for designing such filters.

[1]Interestingly enough, in a number of coding regions, using only $|y_G(n)|^2$ yields better prediction results than using all four output signals.

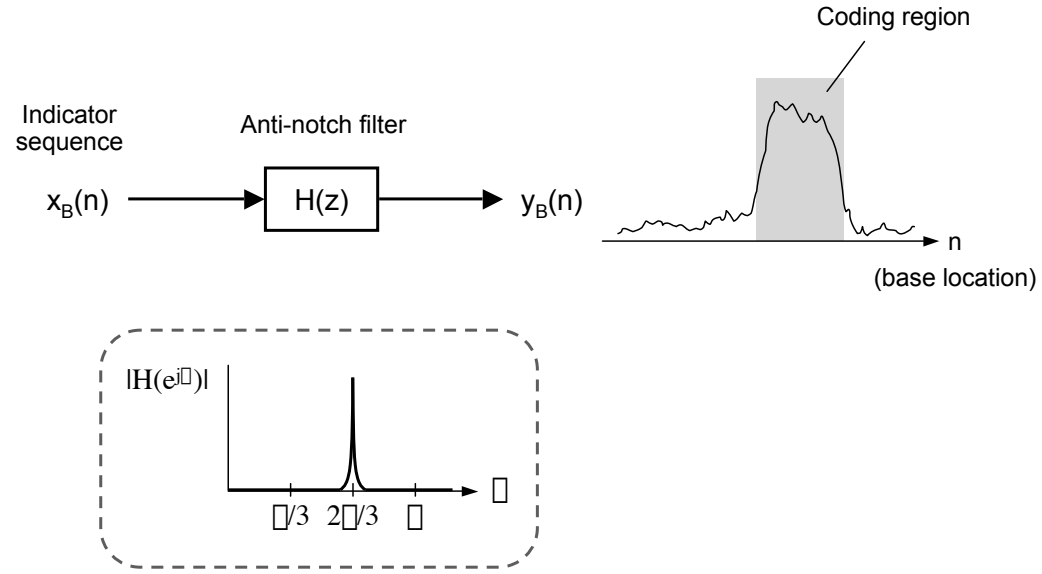


Figure 5.2: The indicator sequence $x_B(n)$ is passed through the bandpass filter $H(z)$. The output $y_B(n)$ will be large in the coding regions due to the period-3 component.

5.4 IIR antinotch filters

In [110], we proposed a gene prediction method based on IIR (infinite impulse response) *antinotch filters* (i.e., complements of *notch filters*). In this section, we describe how such antinotch filters can be designed, and present some experimental results that demonstrate the effectiveness of this approach.

5.4.1 Designing antinotch filters

Anti-notch filters can be effectively designed from a second-order allpass filter

$$A(z) = \frac{R^2 - 2R\cos\theta z^{-1} + z^{-2}}{1 - 2R\cos\theta z^{-1} + R^2 z^{-2}}.$$

This allpass filter $A(z)$ has poles at $Re^{\pm j\theta}$ and zeros at $1/Re^{\pm j\theta}$. Let us consider a filter bank with two filters $G(z)$ and $H(z)$ that are obtained from the allpass filter

$$\begin{bmatrix} G(z) \\ H(z) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ A(z) \end{bmatrix}. \tag{5.1}$$

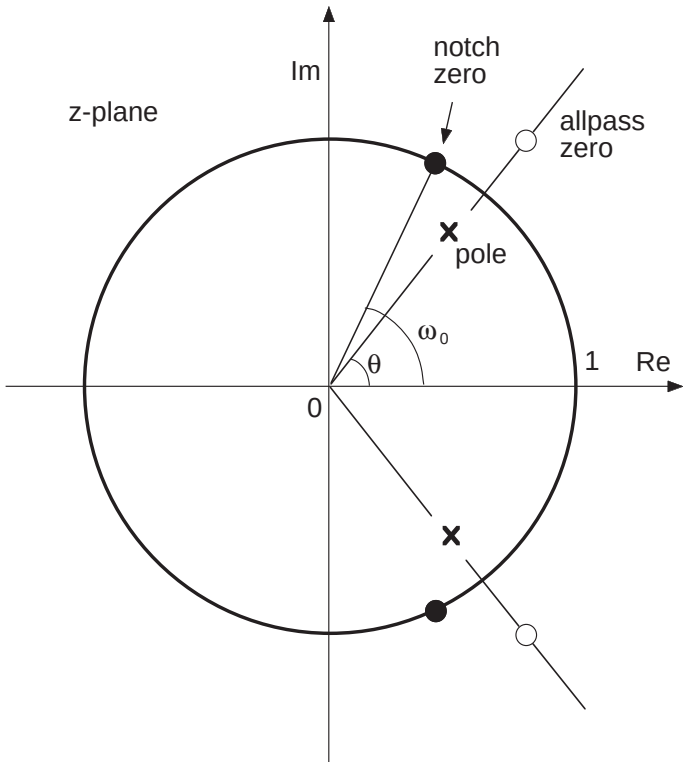


Figure 5.3: Poles and zeros of the notch filter $G(z)$ and the allpass filter $A(z)$.

From (5.1), the filter $G(z)$ can be written as

$$G(z) = K\left(\frac{1 - 2\cos\omega_0 z^{-1} + z^{-2}}{1 - 2R\cos\theta z^{-1} + R^2 z^{-2}}\right),$$

where

$$\cos\omega_0 = \frac{2R\cos\theta}{1 + R^2}, \tag{5.2}$$

and

$$K = \frac{R^2 + 1}{2}.$$

This shows that the filter $G(z)$ is a notch filter [83], whose zeros are located at the frequency ω_0. We can see from (5.2), that the frequency ω_0 gets close to θ when R is close to unity. In this case, the pole and the zero of the filter $G(z)$ are very close to each other as illustrated in Figure 5.3. Therefore, at frequencies that are sufficiently away from ω_0, the magnitude response of $G(z)$ will be close to unity. This is demonstrated in Figure 5.4, which shows the magnitude response of the filter $G(z)$ for two different values of R.

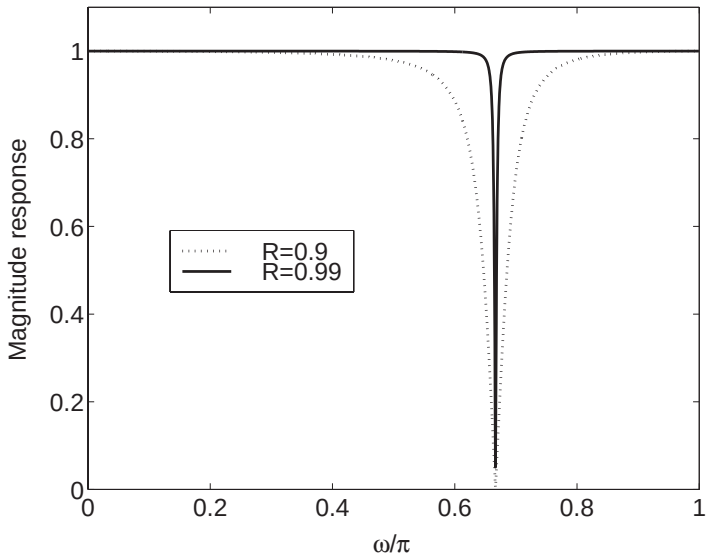


Figure 5.4: Magnitude response of the notch filter $G(z)$ for two values of R.

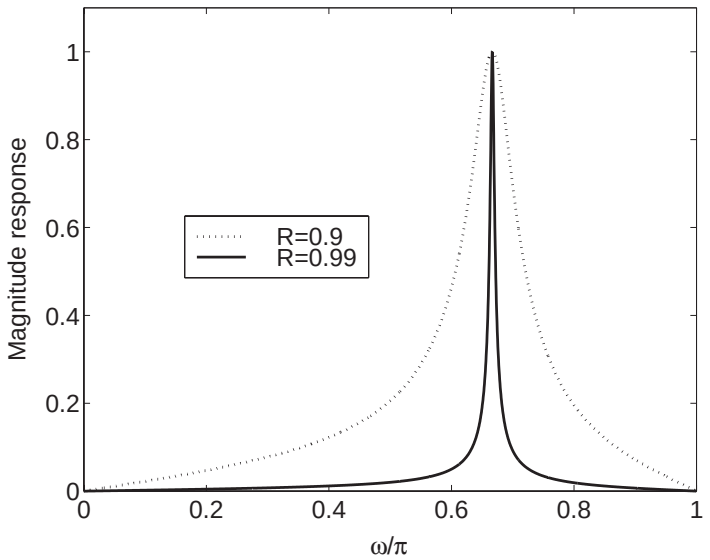


Figure 5.5: Magnitude response of the antinotch filter $H(z)$ for two values of R.

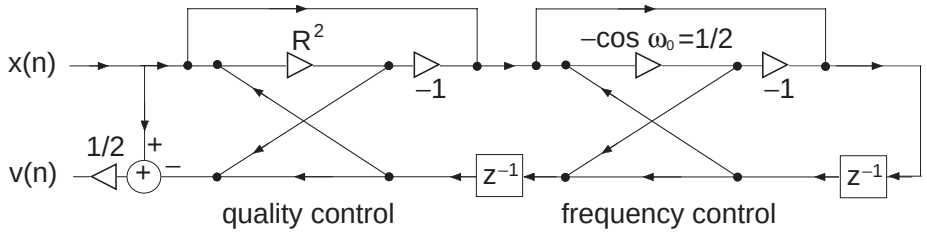


Figure 5.6: Implementation of the antinotch filter $H(z) = V(z)/X(z)$ using a lattice structure.

From (5.1), we can see that the frequency response of $G(z)$ and $H(z)$ can be written as

$$\begin{bmatrix} G(e^{j\omega}) \\ H(e^{j\omega}) \end{bmatrix} = \frac{\mathbf{U}}{\sqrt{2}} \begin{bmatrix} 1 \\ A(e^{j\omega}) \end{bmatrix}, \tag{5.3}$$

where $\mathbf{U}$ is a unitary matrix that satisfies

$$\mathbf{U}^H \mathbf{U} = I.$$

From (5.3), we can see that

$$\left|G(e^{j\omega})\right|^2 + \left|H(e^{j\omega})\right|^2 = \frac{1 + \left|A(e^{j\omega})\right|^2}{2} = 1,$$

since $|A(e^{j\omega})| = 1$ as $A(z)$ is an allpass filter. This shows that the two filters $G(z)$ and $H(z)$ are *power complementary* [109]. As $G(z)$ is a good notch filter as shown in Figure 5.4, the power complementary filter $H(z)$ becomes a good antinotch filter, as we can see in Figure 5.5. By choosing $\omega_0 = 2\pi/3$, the filter $H(z)$ can be used to identify the regions in a DNA sequence that display strong period-3 property.

5.4.2 Implementation of the antinotch filter using a lattice structure

The allpass filter $A(z)$ that is used to obtain the antinotch filter $H(z)$ can be implemented either in a direct form structure [74] or in a cascaded lattice structure [83, 109]. The lattice structure is especially attractive as it is computationally efficient. Figure 5.6 shows the implementation of the filter $H(z)$ using a lattice structure. The structure shown in Figure 5.6 has two multipliers which are used as the lattice cofficients

$$k_1 = R^2, k_2 = -\cos\omega_0.$$

Since $\omega_0 = 2\pi/3$ in our application, we have

$$k_2 = -\cos\omega_0 = 1/2,$$

which can be implemented using a binary shift. Therefore, the only significant multiplier is R^2, which controls the quality of the antinotch filter without affecting the frequency $\omega_0 = 2\pi/3$. We can adjust the value of R^2 to control the base-domain resolution as desired.

5.4.3 Experimental results

In order to demonstrate the proposed idea, we performed the following experiment. We first took a segment of DNA sequence in *C. elegans* chromosome III (GenBank accession number AF099922; base locations 7,021–15,080) and computed the four indicator sequences $x_A(n)$, $x_C(n)$, $x_G(n)$, and $x_T(n)$. This DNA segment contains the protein-coding gene F56F11.4 that consists of five exons. Firstly, we used the DFT-based method to compute $S[N/3]$. Figure 5.7 (Top) shows the plot of $S[N/3]$ as function of relative base location ($n = 0$ corresponds to the base location 7,021 in the original DNA sequence). As we can see in Figure 5.7 (Top), the last four exons have clearly visible peaks. However, the peak that arises from the first exon is somewhat buried in the noisy background and it is not easily distinguishable from the spurious peaks. The plot in Figure 5.7 (Bottom) shows the output $Y(n)$ obtained by the allpass-based antinotch filter with a pole radius $R = 0.992$. In this plot, the first peak is also larger than the spurious peaks in the background, showing the location of the first exon more clearly. This result shows that the antinotch filter approach works very well, while providing additional advantages in implementation.

5.5 Multistage filters

Although it has been demonstrated that the IIR antinotch approach works quite well, we can improve the performance further by designing a filter with a better stopband attenuation at a slight increase in the number of multipliers.. Such filters are essential for suppressing the background noise (for example, the so-called $1/f$ noise that arises from the long-range correlations in DNA sequences [76, 117, 135]). In this section, we introduce a method based on the idea of multistage filtering [109]. This method was proposed in [111, 112].

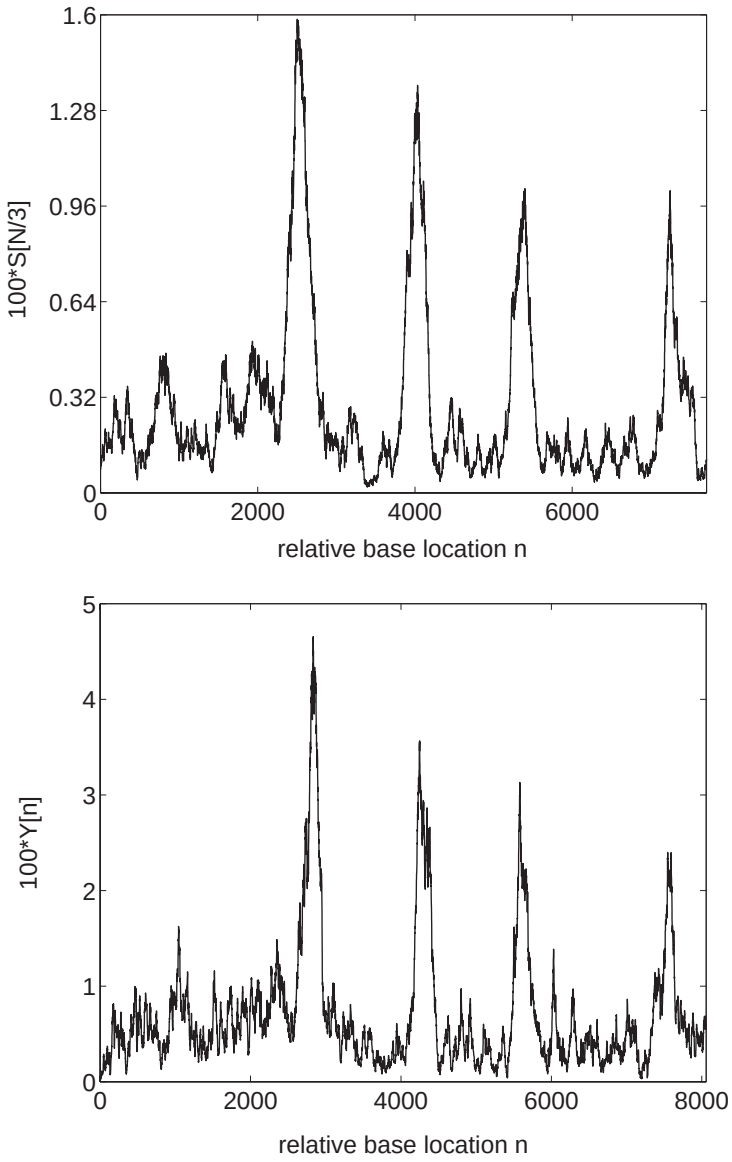


Figure 5.7: Exon prediction results for the gene F56F11.4 in the *C. elegans* chromosome III. (Top) Plot of $S[N/3]$ computed using the DFT. (Bottom) Plot of $Y(n)$ that is computed using the antinotch filter.

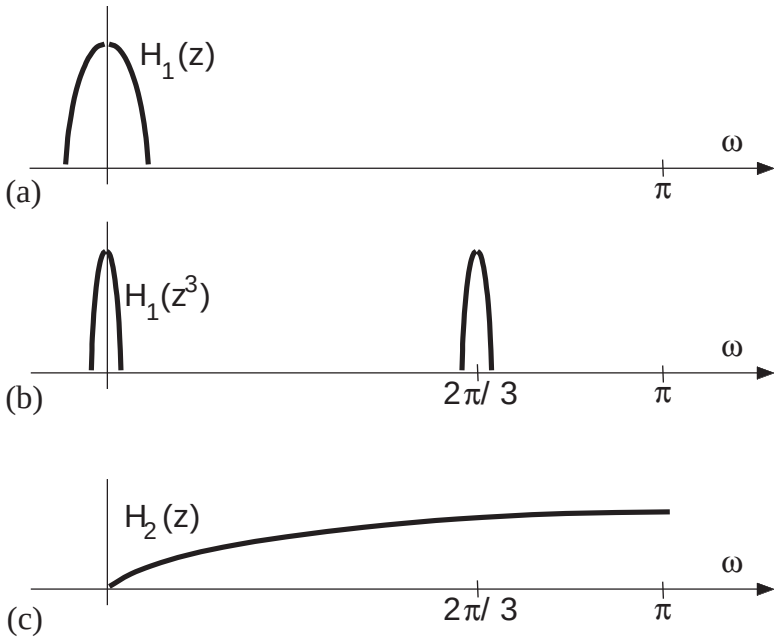


Figure 5.8: Designing a bandpass filter based on multistage filtering. (a) Magnitude response of the prototype narrowband lowpass filter $H_1(z)$. (b) Magnitude response of $H_1(z^3)$. (c) Magnitude response of $H_2(z)$ that can be used to eliminate the undesirable passband of $H_1(z^3)$ at $\omega = 0$.

5.5.1 Designing a multistage filter

The multistage approach provides an efficient way to design a bandpass filter. In order to see how it works, let us consider a narrowband lowpass filter $H_1(z)$ as shown in Figure 5.8 (a). If we replace every delay z^{-1} in the prototype filter $H_1(z)$ with z^{-3}, we get a new filter $H_1(z^3)$ whose magnitude response is as shown in Figure 5.8 (b). We can see that $H_1(z^3)$ has two passbands which are centered at $\omega = 0$ and $\omega = 2\pi/3$. Note that the width of each passband is narrower than that of the original filter $H_1(z)$. In order to eliminate the undesirable passband at $\omega = 0$, we cascade $H_1(z^3)$ with a highpass filter $H_2(z)$ that is shown in Figure 5.8 (c). This idea is illustrated in Figure 5.9. The overall response of the cascaded filter is

$$H(z) = H_1(z^3)H_2(z).$$

This multistage filter $H(z)$ is a narrowband filter with a passband centered at $2\pi/3$, which can be used for isolating the period-3 component in a given signal. The basic concept of the multistage filter design is similar to the so-called IFIR (interpolated FIR) method proposed by Neuvo et al. [72].

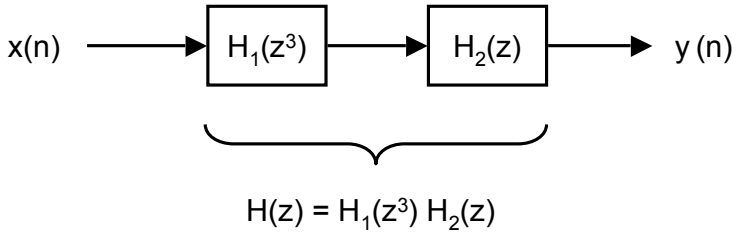


Figure 5.9: Multistage filtering.

5.5.2 Low complexity implementation

We can obtain a good bandpass filter $H(z)$ with a high stopband attenuation from $H_1(z)$ and $H_2(z)$ with very low complexity. Figure 5.10 illustrates such an example. In this example, the prototype filter $H_1(z)$ is a third-order elliptic filter. The magnitude response of $H_1(z)$ and $H_1(z^3)$ is shown in Figure 5.10 (a) and Figure 5.10 (b), respectively. The highpass filter $H_2(z)$ is chosen to be a simple FIR filter with two zeros at $\omega = 0$

$$H_2(z) = (1 - z^{-1})^2,$$

whose response is shown in Figure 5.10 (c). The overall response of the cascaded filter $H(z) = H_1(z^3)H_2(z)$ is shown in Figure 5.10 (d). We can see that $H(z)$ has a narrow passband centered at $\omega = 2\pi/3$ and excellent stopband attenuation at most frequencies.

When implemented in direct form [74], $H_1(z)$ requires 5 multipliers, and $H_2(z)$ is multiplier-less. However, we can also implement the elliptic filter $H_1(z)$ using the allpass decomposition method [109], which requires even fewer multipliers. Based on the allpass decomposition method, the third-order elliptic filter $H_1(z)$ can be written as

$$H_1(z) = \frac{A_0(z) + A_1(z)}{2},$$

where $A_0(z)$ is a first order allpass filter and $A_1(z)$ is a second-order allpass filter, both with real coefficients. Using the lattice structure as in Section 5.4.2, we can implement the filter $A_0(z)$ with a single multiplier, and the filter $A_1(z)$ with two multipliers, so that $H_1(z)$ requires only three multipliers in total. To summarize, the filter designed by the multistage approach has a significantly better characteristic than the allpass-based antinotch filter, at a slightly higher complexity.

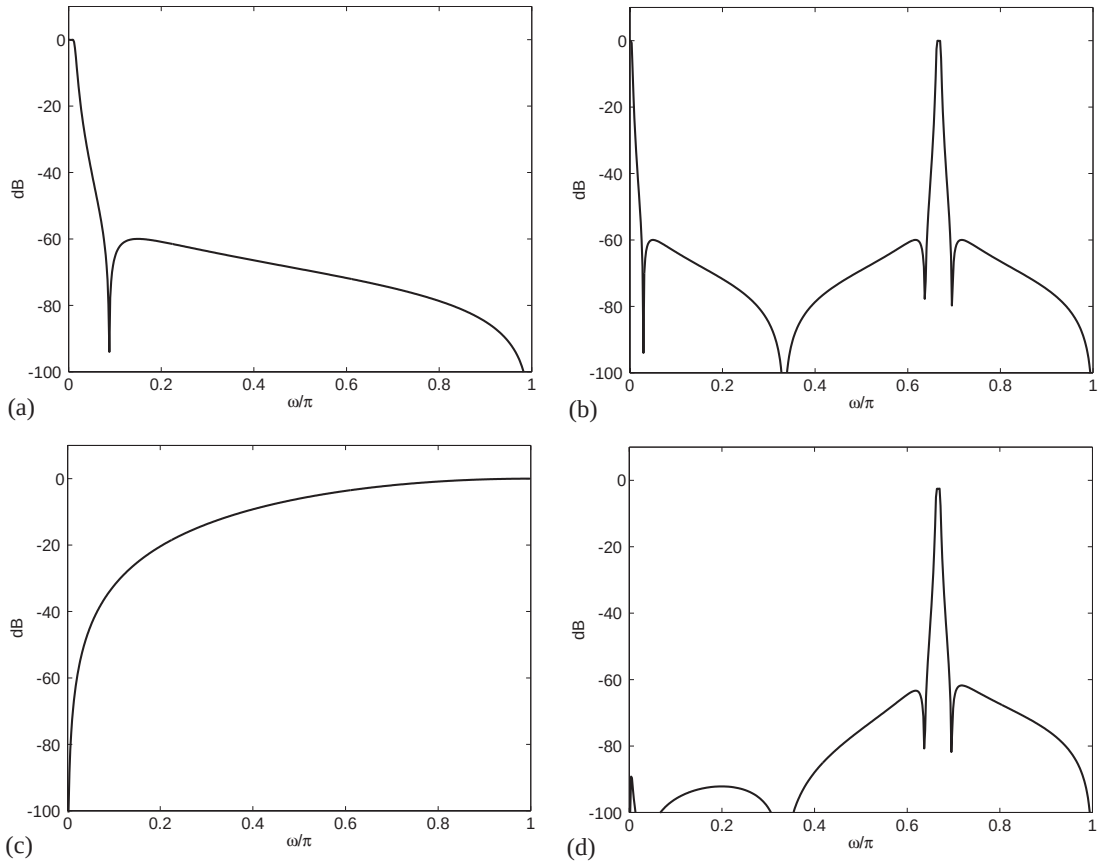


Figure 5.10: Magnitude response of the filters used in the multistage design method. (a) Lowpass elliptic filter $H_1(z)$. (b) The response of the filter $H_1(z^3)$. (c) FIR lowpass filter $H_2(z)$. (d) The magnitude response of the cascaded filter $H(z) = H_1(z^3)H_2(z)$.

5.5.3 Experimental results

In order to evaluate the performance of the multistage filtering approach, we used the same DNA sequence (*C. elegans* chromosome III; GenBank accession number AF099922; base locations 7,021–15,080) that was used to demonstrate the performance of the antinotch filter method. Figure 5.11 (c) shows the output of the multistage filter, along with the output of the DFT approach (shown in Figure 5.11 (a)) and the output of the antinotch filter (shown in Figure 5.11 (b)). As we can see in this example, the proposed multistage filtering approach shows all peaks that correspond to the five exons in the gene F56F11.4 very clearly. Unlike the output signals of the DFT method and the antinotch filter method, the background noise is almost completely removed in Figure 5.11 (c) owing to the high stopband attenuation of the multistage filter.

5.6 Conclusion

As explained in detail in [38], gene identification is a complicated problem, and the identification of the period-3 patterns is only a first step towards gene and exon prediction. Due to the complex nature of the gene identification problem, we usually need a more powerful model that can effectively represent the characteristics of protein-coding genes. In fact, many state-of-the-art protein-coding gene finders are built on hidden Markov models (HMMs) and their variants [25, 28, 58, 59, 96], as HMMs are very effective in representing the conserved patterns in DNA sequences as well as the short-term correlations between adjacent bases. However, HMMs are computationally much more expensive than the prediction methods based on digital filters. Therefore, we can use the prediction methods proposed in Section 5.4 and Section 5.5 for a fast prescreening of the genome and use a more powerful model (such as an HMM) in the second stage in order to expedite the overall gene identification process.

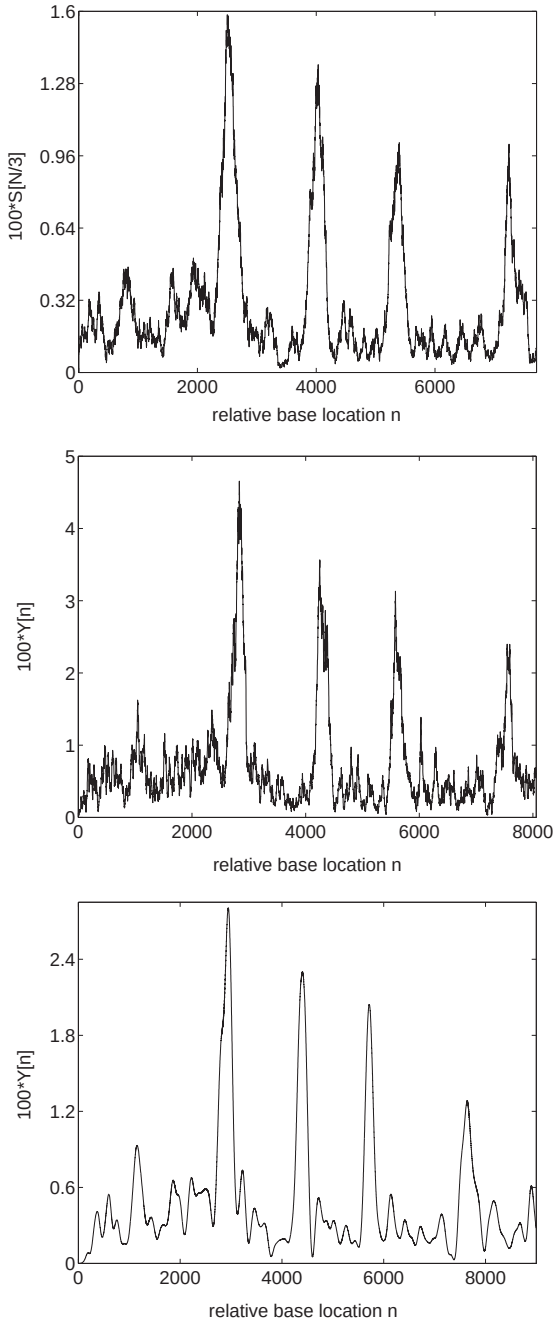


Figure 5.11: Exon prediction results for the gene F56F11.4 in the *C. elegans* chromosome III. (Top) Plot of the output $S[N/3]$ of the DFT-based approach. (Middle) Plot of the output of the antinotch filter method. (Bottom) Plot of the output of the multistage filter approach.

Chapter 6

Identification of CpG Islands Using Filter Banks

CpG islands are specific regions in DNA molecules that are abundant in the dinucleotide CpG. This dinucleotide is usually denoted as CpG in order to distinguish it from the C-G base pair across the two strands in a DNA double helix. The CpG islands are of our interest for many biological reasons. For example, it has been shown that CpG islands can be used as gene markers, since they are located upstream of the transcription start regions of many genes [61]. Analysis of the human genome shows that all housekeeping genes (which are genes that are expressed in all cells throughout the body, and produce proteins that are necessary for basic maintenance and cellular functions) and 40% of the tissue-specific genes are associated with CpG islands [61]. This makes them useful landmarks for identifying protein-coding regions in the human genome. Moreover, experiments have shown that the methylation[1] of CpG islands plays an important role in *gene silencing* [10], *genomic imprinting* [36], *carcinogenesis* [55], and so forth.

As the CpG islands have unique characteristics, such as the frequent occurrence of CpG dinucleotides and the high G+C content, we can use computational techniques for identifying CpG islands in DNA sequences. Until now, several CpG island prediction methods have been proposed, each with its own strength and weakness [25, 40, 90, 103].

In this chapter, we propose a novel scheme for detecting the CpG islands. The proposed method is based on a digital signal processing technique that uses a bank of IIR lowpass filters. Despite the simplicity of the proposed method, it is capable of identifying CpG islands efficiently at a very low computational cost. The content of this chapter is mainly drawn from [124].

[1]Methylation refers to the replacement of a hydrogen atom with the methyl group. For example, in DNA methylation, the cytosine is replaced by a 5-methylcytosine [8].

6.1 Outline

This chapter is organized as follows.

In Section 6.2, we review some of the conventional CpG island prediction methods that have been proposed in the literature [25, 40, 103]. We especially focus on the method based on Markov chains, and describe its technical details in Section 6.2.1. Experimental results obtained from the conventional Markov chain approach is presented in Section 6.2.1.

In Section 6.3, we propose a new method for identifying CpG islands based on a bank of IIR low-pass filters. Firstly, we give a digital filtering interpretation of the conventional approach described in Section 6.2.1, and propose the filter-bank approach in Section 6.3.1. We show the experimental results of the proposed method in Section 6.3.2, and then explain in Section 6.3.3 how this result can be used for predicting the exact location of the CpG island.

We conclude the chapter in Section 6.4.

6.2 Identification of CpG islands

The first large-scale computational analysis of CpG islands traces back to the work of Gardiner-Garden and Frommer in 1987 [40]. They defined CpG islands as regions of at least 200 bp (base pairs) length, with a G+C content higher than 50% and the observed CpG to expected CpG ratio equal to or above 0.6. The exact definition of CpG islands is somewhat arbitrary, since the choice of the cut-off parameters can have a critical impact on which regions are included in the definition of the CpG islands. For example, Takai and Jones redefined the CpG island as a region of DNA whose length is at least 500 bp with a G+C content equal to or above 55% and observed CpG to expected CpG ratio above 0.65 [103]. By using the new definition, they could find regions that are more likely to be associated with the 5′ regions of genes while excluding most of the so-called *Alu-repeats* [103].

In addition to these simple schemes, there are other interesting approaches that make use of more sophiscated–hence, more powerful–techniques [19, 25, 90]. One such example is the technique based on Markov chains, which we present next.

6.2.1 Markov chains

Markov chains can effectively model the short-term dependencies between the adjacent symbols in a symbol sequence. For example, in [25], Markov chains are used to describe the different sequence

characteristics inside a CpG island and outside CpG islands. In this scheme, one Markov chain is used to model CpG islands and another chain is used to model the rest of the genome, where the two Markov chains have different statistics (i.e., different transition probabilities). Based on the two Markov chains, the prediction scheme works as follows. Given a short DNA sequence, we first compute the log-score of this sequence based on each Markov chain. Then the two scores are compared to each other to choose the more likely one. This allows us to decide whether the given DNA segment belongs to a CpG island or not.

Let us consider a sequence of nucleotides $x(n) \in \{A, C, T, G\}$. We assume that $x(n)$ forms a first-order Markov chain, where the probability of each symbol $x(n+1)$ depends only on the current symbol $x(n)$. Now, let us denote the transition probability from a base β to a base γ in a CpG island and that in a non-CpG island region as $p^+_{\beta\gamma}$ and $p^-_{\beta\gamma}$, respectively. For example, p^+_{AC} is the probability that the next symbol will be a C, given that the current symbol is an A inside a CpG island. Using these notations, the probability of observing the sequence $x(n)x(n+1)\cdots x(n+L-1)$, assuming that it belongs to a CpG island and that the previous symbol was $x(n-1)$ can be written as

$$\begin{aligned} P(n|\text{CpG}) &= P(x(n)\cdots x(n+L-1)|x(n-1), \text{CpG model}), \\ &= \prod_{i=0}^{L-1} p^+_{x(n-1+i)x(n+i)} \end{aligned}$$

Similarly, the probability of observing this sequence, assuming that it belongs to a non-CpG island region is

$$\begin{aligned} P(n|\text{non-CpG}) &= P(x(n)\cdots x(n+L-1)|x(n-1), \text{non-CpG model}), \\ &= \prod_{i=0}^{L-1} p^-_{x(n-1+i)x(n+i)} \end{aligned}$$

If $P(n|\text{CpG})$ is greater than $P(n|\text{non-CpG})$, we can conclude that the DNA sequence $x(n)x(n+1)\cdots x(n+L-1)$ belongs to a CpG island. Otherwise, it is more likely that the sequence does not belong to a CpG island. Therefore, if we define

$$S(n) = \log \frac{P(n|\text{CpG})}{P(n|\text{non-CpG})}, \tag{6.1}$$

which is the log-likelihood ratio. $S(n) > 0$ implies that the given DNA sequence is more likely to

$p^+_{\beta\gamma}$	A	C	G	T
A	0.1598	0.2914	0.4247	0.1241
C	0.1299	0.3862	0.3093	0.1746
G	0.1425	0.3675	0.3675	0.1225
T	0.0758	0.3742	0.3687	0.1813

Table 6.1: Transition probabilities inside the CpG island region.

$p^-_{\beta\gamma}$	A	C	G	T
A	0.2499	0.2209	0.3526	0.1766
C	0.2810	0.3352	0.0941	0.2897
G	0.2159	0.2586	0.3397	0.1858
T	0.1283	0.2624	0.3594	0.2499

Table 6.2: Transition probabilities in the non-CpG island region.

belong to a CpG island, whereas $S(n) < 0$ implies that the sequence is likely to belong to a non-CpG island region.

6.2.2 Experimental results

Despite the simplicity of this idea, it has been shown that this method works quite well [25]. In order to see this, let us consider the following experiment. First, we took a DNA sequence of length 219,447 from the human chromosome X (GenBank accession number L44140) that has been already annotated, and computed the transition probabilities for the two regions. These are shown in Table 6.1 and Table 6.2. Each row in the table contains the transition probabilities from a specific base to each of the four bases. For example, the first row of Table 6.1 contains the probabilities that each of the four bases will follow the base A inside CpG islands. Therefore, every row in the tables adds up to unity. By comparing Table 6.1 and Table 6.2, we can find an interesting fact about the transition probabilities. In Table 6.1, we can see that the probability that a C will be followed by a G is very high inside the CpG islands, resulting in many CpG dinucleotides. However, this is not the case outside the CpG islands. Table 6.2 shows that it is rather unlikely that a C will be followed by a G. This is known to be a result of the *methylation* process which mutates a C into a T with a relatively high probability, whenever it finds a CpG dinucleotide [9]. As a consequence, CpG dinucleotides appear much less frequently than they are expected. However, this methylation process is suppressed in the CpG islands, hence we can observe more CpG dinucleotides than usual [9].

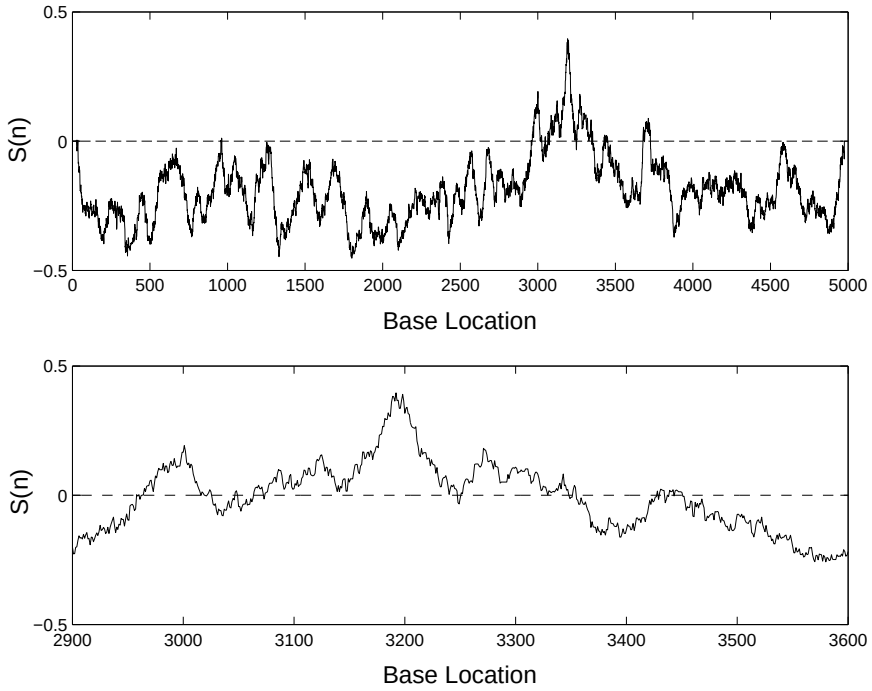


Figure 6.1: (Top) CpG island prediction result obtained by the conventional Markov chain method. (Bottom) Magnified plot. We can see that there are many undesirable zero crossings due to the fluctuations of $S(n)$.

Figure 6.1 shows the prediction result of CpG islands based on this approach. Between the base locations 1 and 5,000, there is only one CpG island of length 332 between 3,095 and 3,426. At this location, $S(n) > 0$ which implies that it is very likely that this region overlaps with a CpG island. Outside this region, $S(n)$ is mostly negative although there are some fluctuations. This plot shows that the CpG/non-CpG regions can be reasonably discriminated by looking at the sign of the log-likelihood ratio $S(n)$.

However, if we take a closer look at the plot, we can see that there are a lot of fluctuations around zero, resulting in many unwanted zero crossings. The bottom plot in Figure 6.1 shows the magnified plot around the CpG island. We can see that the region around the base location 3,000 has positive values although it does not belong to a CpG island. Moreover, there are several zero crossings inside the CpG island. Apparently, this is not what we expect. In the next section, we propose a new method that can eliminate these problems and improve the prediction results.

6.3 Identifying CpG islands using a bank of IIR lowpass filters

When using the method elaborated in Section 6.2, it is not very obvious how to choose the window size L for computing $S(n)$. This is an important issue, since the choice of the window size can have a significant effect on the detection results. Larger windows usually enhance the reliability of the result but degrade the resolution of the output. On the contrary, smaller windows are able to catch up with the changes of the statistical properties very quickly, but $S(n)$ may fluctuate around zero more often, thereby making the identification results less reliable.

In [124], we proposed a CpG island prediction method that can effectively solve this problem. The details of the proposed method is elaborated in the following.

6.3.1 Filtering the log-likelihood ratios using a filter bank

Let us consider again the log-likelihood ratio $S(n)$ in (6.1). If we define $y(n)$ as the log-likelihood ratio of a single transition

$$y(n) = \log\left(\frac{p^{+}_{x(n-1)x(n)}}{p^{-}_{x(n-1)x(n)}}\right), \tag{6.2}$$

then $S(n)$ can be rewritten as

$$\begin{aligned} S(n) &= \frac{1}{L}\sum_{i=0}^{L-1} y(n+i) \\ &= y(n) * h_{ave}(n), \end{aligned} \tag{6.3}$$

where the symbol "$*$" denotes convolution. Here, $h_{ave}(n)$ is a simple averaging filter that is defined as

$$h_{ave}(n) = \begin{cases} \frac{1}{L} & -L+1 \le n \le 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $h_{ave}(n)$ can be viewed as a simple lowpass filter. Instead of using a single filter that is rectangular in shape, we may use a bank of M filters, where each filter is a lowpass filter with a different bandwidth. By looking at the outputs altogether, we can predict the location of the CpG islands more precisely. This idea is illustrated in Figure 6.2.

One way to construct such a filter bank is to use the following one-pole filter

$$H_k(z) = \frac{1-\alpha_k}{1-\alpha_k z^{-1}} \tag{6.4}$$

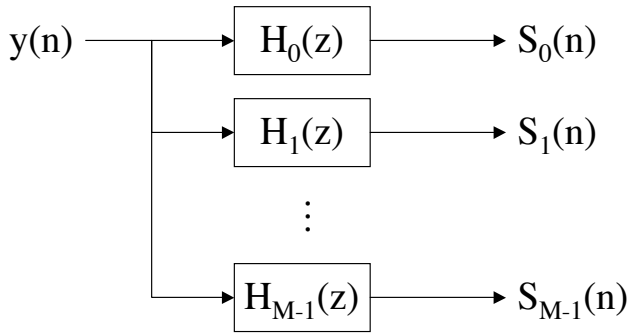


Figure 6.2: A bank of M lowpass filters with different bandwidths.

in the kth channel, where the poles are chosen such that

$$0 < \alpha_0 < \alpha_1 < \cdots < \alpha_{M-1} < 1.$$

This corresponds to $h_k(n) = (1 - \alpha_k)\alpha_k^n u(n)$ in the time domain. Choosing the filter $h_k(n)$ in this way results in weighted averaging of $y(n)$, where the newer inputs are given larger weights than the older ones. Also note that every $h_k(n)$ satisfies

$$\sum_n h_k(n) = 1,$$

serving as a proper weighting function.

6.3.2 Experimental results

In order to demonstrate the idea, we performed the following experiment. We chose α_k ($k = 0, 1, \ldots, 40$) from 0.95 to 0.99 by increasing its value by 0.001, and $H_k(z)$ was chosen as (6.4). We computed $y(n)$ defined as (6.2) from the same input sequence (the human chromosome X) that was used to test the averaging method in Section 6.2.2. Then we filtered $y(n)$ using the filters $h_k(n)$ to obtain

$$S_k(n) = y(n) * h_k(n),$$

for all k. We combined these outputs altogether to obtain the contour plot shown in Fig. 6.3. The contour plot in Figure 6.3 clearly shows the band which corresponds to the CpG island located between 3,095 and 3,426 (colored in orange and red). This is more prominent in Figure 6.4, which

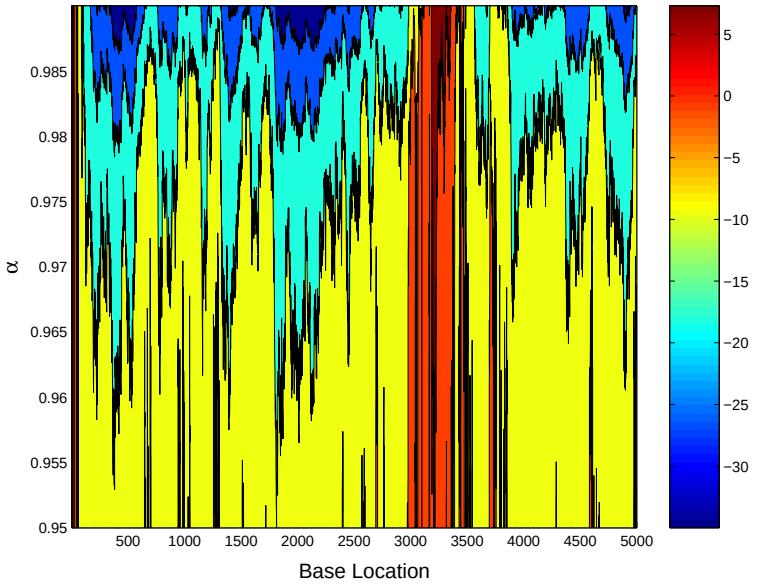


Figure 6.3: Contour plot of the outputs $S_k(n)$. The red band in the middle clearly indicates the existence of a CpG island.

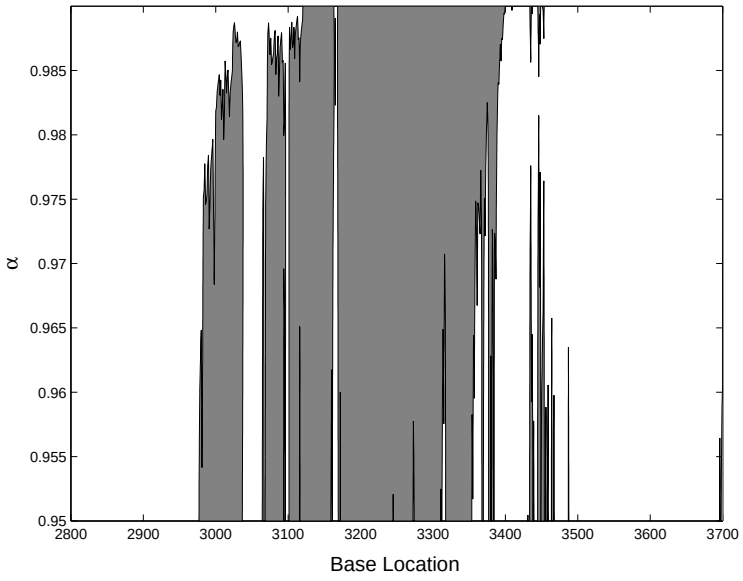


Figure 6.4: Two level contour plot of the outputs $S_k(n)$. The level curve is located at zero, separating the positive and the negative regions.

is a two-level contour plot of the same output $S_k(n)$. The level curves are located at $S_k(n) = 0$, and the shaded area indicates where $S_k(n)$ is positive. From Figure 6.4, we can see that as α_k increases, there are less fluctuations around zero. For example, for $\alpha_k = 0.99$ there is only one small region inside the shaded band where $S_k(n)$ goes below zero. On the contrary, for $\alpha_k = 0.95$ there are more than 10 regions inside the band where $S_k(n)$ is negative. Another interesting point that we can notice in Figure 6.4 is the fact that the shaded region slightly bends to the right as α_k increases. This shows that the response time of the filter $h_k(n)$ is longer for larger α_k. If α_k is large, the past samples are given more weights than when we use a smaller α_k. Conceptually, this implies that more samples of $y(n)$ are taken into account in computing $S_k(n)$. This allows us to obtain a smoother output with less fluctuation, but at the same time, the filter is slower in catching up with the changes in the input statistics. So, there is a trade-off between the responsiveness of the filter and the stability of the output. This is indeed a very similar problem to that of choosing the window size L when using the averaging method in section 6.2, and this is the reason why we have to look at all the output signals at the same time, instead of depending on a single output.

6.3.3 Predicting the transition points between different regions

Now that we are given a number of outputs corresponding to different α_k, how can we predict the start and end points of the CpG islands more accurately? In order to answer this question, let us first consider the following problem.

6.3.3.1 Rectangular window

Assume that we have two distinct regions–a CpG island and a non-CpG island region–each of which can be modeled using a first-order Markov chain with different statistics. Now, using a rectangular window of length L, let us compute the weighted sum of the log-likelihood ratio $y(n)$ inside the window. Since we are using a rectangular window in this case, all $y(n)$ are weighted equally as in (6.3). Initially, let us assume that this window is inside the first region and does not overlap with the second region at all. Then the expectation of $S(n)$ can be simply written as

$$\begin{aligned} E[S(n)] &= \frac{1}{L}\sum_{i=0}^{L-1} E[y(n+i)] \\ &= \frac{1}{L}(L \cdot E[y(n)]) \\ &= E[y(n)]. \end{aligned}$$

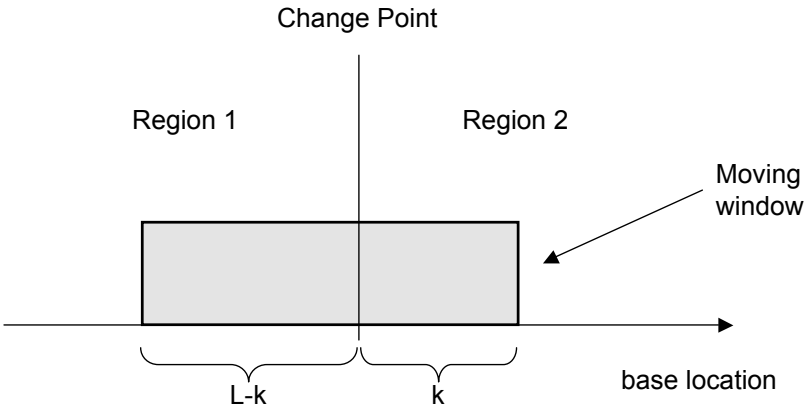


Figure 6.5: A rectangular window of length L located around the transition point.

Therefore, the expectation of $S(n)$ inside the CpG islands will be $E[S(n)|\text{CpG}] = E[y(n)|\text{CpG}] > 0$, and $E[S(n)|\text{non-CpG}] = E[y(n)|\text{non-CpG}] < 0$ outside the CpG islands. Now, consider gradually shifting the window to the right one by one. At some point, it will cross the transition points between those two regions, and the window will have an overlap with both regions as shown in Figure 6.5. If we let k be the length of the part of the window that overlaps with the second region, the expectation of $S(n)$ can be written as

$$\begin{aligned} E[S(n)] = & \frac{1}{L}\sum_{i=0}^{L-1} E[y(n+i)] \\ = & \frac{1}{L}\Big((L-k)\cdot E[y(n)|\text{region 1}] \\ & \qquad +k\cdot E[y(n)|\text{region 2}]\Big) \\ = & \frac{L-k}{L}E[y(n)|\text{region 1}] + \frac{k}{L}E[y(n)|\text{region 2}], \end{aligned} \tag{6.5}$$

where the sign of $E[S(n)]$ depends on L, k, $E[y(n)|\text{region 1}]$ and $E[y(n)|\text{region 2}]$. As we shift the window further, the overlap with the first region will decrease, and finally the whole window will be located inside the second region. Since the sign of $E[S(n)]$ in each region is different, at some point we can observe the change of sign of $E[S(n)]$. Let us denote the k $(0 < k < L)$ that satisfies $E[S(n)] = 0$ as k^*. If we solve for k^*, we get

$$k^* = \frac{E_2 L}{E_2 - E_1}, \tag{6.6}$$

where $E_i = E[y(n)|\text{region } i]$. In fact, this is the point where we can first recognize the change between regions in the given sequence. Therefore, k^* can be viewed as the *delay* of the detection algorithm, and it can be computed if we know E_1 and E_2. Since we know the statistics of the respective Markov chains used for modeling the CpG/non-CpG island regions, we can compute $E^+ = E[S(n)|\text{CpG}]$ and $E^- = E[S(n)|\text{non-CpG}]$ that are needed to compute the delay. We have

$$E^+ = \sum_{\beta,\gamma\in\{A,C,G,T\}} p^+_{\beta\gamma} \log\left(\frac{p^+_{\beta\gamma}}{p^-_{\beta\gamma}}\right), \tag{6.7}$$

and

$$E^- = \sum_{\beta,\gamma\in\{A,C,G,T\}} p^-_{\beta\gamma} \log\left(\frac{p^+_{\beta\gamma}}{p^-_{\beta\gamma}}\right). \tag{6.8}$$

Using the transition probabilities in Table 6.1 and Table 6.2, we get $E^+ = 0.0427$ and $E^- = -0.0412$. Therefore, when entering a CpG island from a non-CpG island region, we expect a delay of

$$k^* = \frac{E^+ L}{E^+ - E^-} = 25.04,$$

for $L = 51$. Similarly, when we leave a CpG island and enter a non-CpG island region, the expected delay is

$$k^* = \frac{E^- L}{E^- - E^+} = 25.96.$$

As we are using a rectangular window, we expect the delay to be around $L/2$, which is indeed the case.

6.3.3.2 Exponentially decaying window

Now let us consider using an exponentially decaying window defined as (6.4). This window is shown in Figure 6.6. Again, we want to find the k that satisfies $E[S(n)] = 0$. In this case, $E[S(n)]$ can be written as

$$\begin{aligned} E[S(n)] &= \sum_{i=-\infty}^{L-1} (1-\alpha)\alpha^{L-1-i} E[y(n+i)] \\ &= \alpha^k E_1 + (1-\alpha^k) E_2. \end{aligned} \tag{6.9}$$

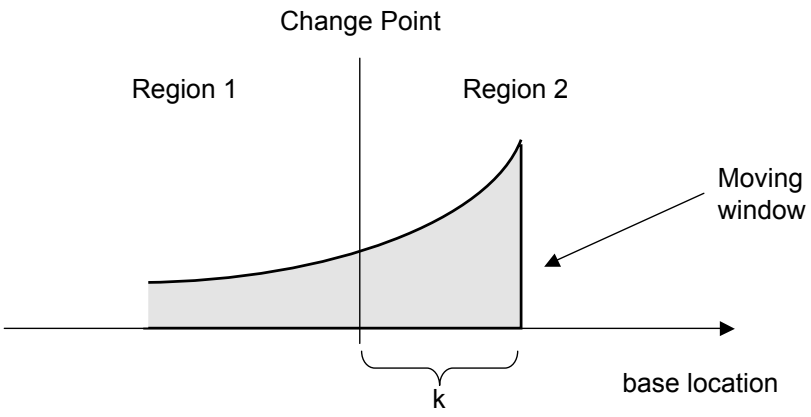


Figure 6.6: An exponentially decaying window located around the transition point.

If we solve for the k^* that makes $E[S(n)] = 0$, we get

$$k^* = \log_\alpha \left(\frac{E_2}{E_2 - E_1} \right). \tag{6.10}$$

Based on the transition probabilities in Table 6.1 and Table 6.2, we get the plot of the delay k^* as a function of α as shown in Figure 6.7.

Now that we have computed the expected delays corresponding to different values of α, let us compare these with the actual zero crossing points. We generated a random sequence of A, C, G, and T based on the transition probabilities in Table 6.1 and Table 6.2. We computed $S_k(n)$ for $0.95 < \alpha_k < 0.99$ and computed the zero crossing points. Figure 6.8 shows the level curves for $S(n) = 0$ with the theoretical curve obtained from (6.10). It can be seen that the theoretical curves are very close to the actual curves. Therefore, in order to predict the changing point of the two regions more accurately, we may first find the level curves for $S(n) = 0$ and find the theoretical curve of the zero crossing points that matches the actual curve best. From this, we can make up for the delay and predict the actual transition point more precisely.

6.4 Conclusion

In this chapter, we proposed a novel scheme for predicting CpG islands, which is based on a bank of IIR filters. Preliminary experimental results indicate that the proposed method can effectively locate the CpG islands in a given DNA sequence at a relatively low computational cost. When

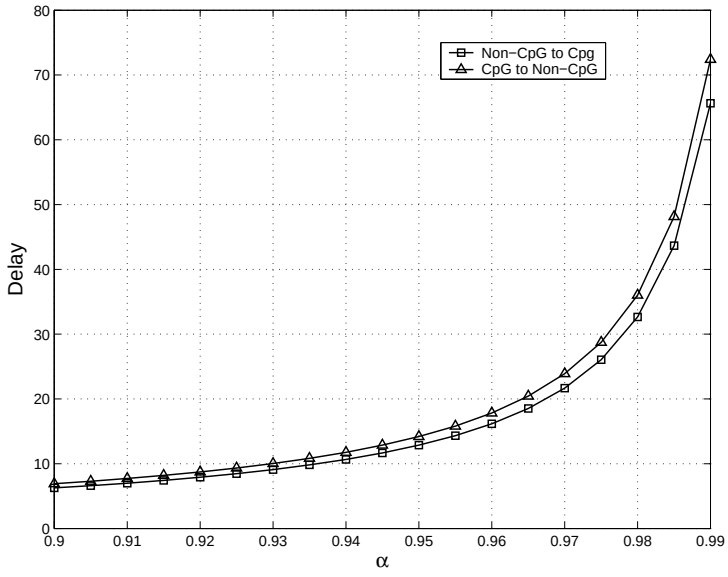


Figure 6.7: The delay k^* corresponding to the value α_k.

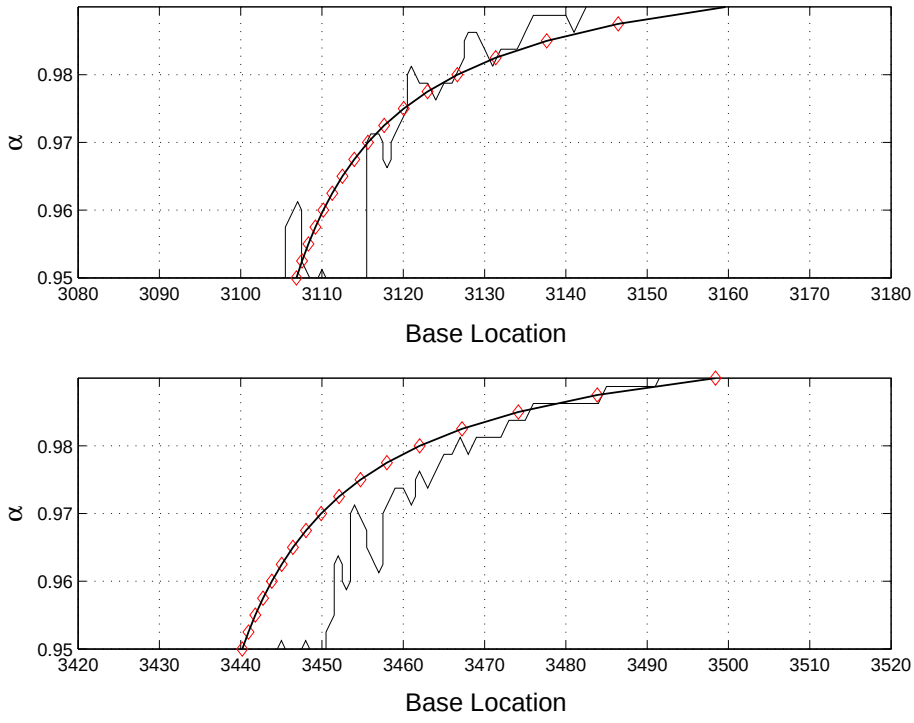


Figure 6.8: Actual level curves for $S(n) = 0$ against the theoretical curve (thick line with diamonds) computed from the transition probabilities. (Top) Region changes from a non-CpG island region to a CpG island. (Bottom) Region changes from a CpG island to a non-CpG island region.

building an actual CpG island prediction tool based on the proposed approach, we may also incorporate additional properties of typical CpG islands (such as their length distribution) to improve the overall performance. We may also use lowpass filters with better passband/stopband details to improve the prediction results.

Chapter 7

Conclusion

In this thesis, we have presented various signal processing techniques that can be applied to the computational analysis of biological sequences. The main focus of the thesis was on introducing signal processing methods that can be utilized for identifying regions of specific biological significance (e.g., noncoding RNA genes, protein-coding genes, CpG islands) in a large sequence database. Depending on the type of the region of interest, we proposed different techniques that can be utilized for recognizing the unique characteristics within those regions.

The first part of the thesis focused on signal processing methods that can be utilized in RNA sequence analysis. Many functional RNAs have well-conserved secondary structures that give rise to complicated long-range correlations between distant bases. We first proposed the concept of context-sensitive HMM (csHMM) that extends the conventional HMM so that it can represent these long-range correlations. Based on the proposed model, we introduced efficient dynamic programming algorithms for finding the optimal state sequence and computing the probability of an observed symbol string. Furthermore, we proposed a parameter re-estimation algorithm that can be used for finding the optimal parameters of a csHMM in an iterative manner. We demonstrated that the csHMMs can be effectively used for modeling and recognizing RNAs with various secondary structures.

Based on csHMMs, we proposed the concept of profile-csHMMs, which are specifically constructed csHMMs that are especially useful in building probabilistic representations of RNA sequence families. Profile-csHMMs are the first model that can represent any kind of RNA pseudoknots, which is a significant advantage over the existing models that have been utilized for RNA sequence analysis. In addition to this, we proposed the sequential component adjoining (SCA) algorithm that can solve the optimal alignment problem of profile-csHMMs. To demonstrate the ef-

fectiveness of the proposed model, we used them to build a tool for finding the structural alignment of RNAs including pseudoknots. It was shown that the profile-csHMM approach could achieve a high prediction accuracy at a relatively low computational cost. Finally, we proposed a practical scheme based on pre-screening filters, which can expedite a profile-csHMM based search. Experimental results indicate that the proposed scheme can make the search speed significantly faster, without affecting the prediction accuracy.

In the second part of the thesis, we focused on the application of digital filters and filter banks in DNA sequence analysis. Firstly, we demonstrated that digital filters can be effectively used for identifying the periodic components that are frequently observed in protein-coding regions. We showed that this can be utilized for predicting the coding genes that are buried in the DNA. We also proposed efficient schemes for implementing the digital filters, which can identify the coding regions at a very low computational complexity. Secondly, we showed how we can predict the location of CpG islands using a filter bank. In the proposed approach, two Markov chains have been used to model the base transition probabilities inside the CpG islands and the probabilities outside the CpG islands. The sequence of log-likelihood ratios obtained from these models have been processed using a bank of IIR lowpass filters, where the output signals were used to predict the exact locations of the CpG islands. We demonstrated that the filter bank approach can yield reliable prediction results without sacrificing the resolution of the predicted start/end positions of the CpG islands.

The discussions on the application of signal processing techniques in biological sequence analysis presented in this thesis opens up many other interesting problems. First of all, as the profile-csHMM is a recent development, there are still many related issues that need to be addressed. For example, we do not yet have a training algorithm that can be used for finding the optimal parameters of a profile-csHMM. As the choice of the model parameters can have a significant impact on the performance of a profile-csHMM based application (e.g., ncRNA gene finder, RNA structural alignment tool), it is important to have an effective parameter re-estimation algorithm for profile-csHMMs. We also need an algorithm for finding the optimal adjoining order of general profile-csHMMs. Currently, our algorithm can find the adjoining order of only those profile-csHMMs whose correlation structure belongs to the Rivas&Eddy class [86], without guaranteeing optimality. As the adjoining order has a direct impact on the overall computational cost of the SCA algorithm, it is important to have an algorithm that can automatically find the best adjoining order

that minimizes the computational cost.

On the practical side, it would be interesting to build a BLAST-like homology search tool for RNAs. There has been an attempt to build such a tool based on CMs (covariance models) [57], but this tool was limited to RNAs without pseudoknots and it was also too slow to scan a large database. Based on profile-csHMMs, we can build a search tool that can find homologues of single structured RNAs including pseudoknots. We may use the pre-screening approach and also incorporate other heuristic methods to improve the search speed so that it becomes fast enough to be of practical use.

There exist also other interesting problems that are relevant to the current topic, although they have not been considered in this thesis. One such problem is the identification of novel ncRNAs in the genome. In this thesis, we mainly focused on finding new homologues of known ncRNA families. The computational identification of novel ncRNAs is a much more difficult problem, and the research on this topic is still at an early stage. It would be interesting to see how signal processing techniques could contribute to the prediction of novel ncRNAs. Another interesting problem is to build and analyze gene regulatory networks that include both ncRNA genes and protein-coding genes. As many ncRNAs play important regulatory roles within cells, such a network would be more realistic than the traditional gene regulatory networks solely based on protein-coding genes that have been considered till now. This may lead to a better understanding of the diverse control mechanisms in the cell machinery.

Appendix A

Example of a CFG That Is Not Representable by a csHMM

In this appendix, we give an example of a language that can be described by a context-free grammar but not by a csHMM. Let us consider a context-free grammar that has two non-terminal symbols S, T and three terminal symbols a, b, c. We begin with the start non-terminal S and apply the the following production rules

$$\begin{aligned} S &\longrightarrow aTSa \mid TaSa \mid TSaa \mid aTa \mid Taa, \\ T &\longrightarrow bT \mid bc. \end{aligned}$$

The grammar shown above can generate any symbol sequence of the form $\mathbf{x} = x_1 \dots x_{L-N} x_{L-N+1} \dots x_L$ for any given positive number N, where $x_{L-N+1} \dots x_L = a \dots a$ and $x_1 \dots x_{L-N}$ contains N number of "a's" and the same number of subsequences in the form of "$b \dots bc$." For example, we can generate the following sequences using this grammar

$$(N=3) \quad \underbrace{a\ bbbc\ a\ bc\ a\ bbc}_{x_1 \dots x_{L-N}} \quad \underbrace{aaa}_{x_{L-N+1} \dots x_L} \tag{A.1}$$

$$\underbrace{bbc\ bc\ a\ a\ a\ bbbbc}_{x_1 \dots x_{L-N}} \quad \underbrace{aaa}_{x_{L-N+1} \dots x_L} \tag{A.2}$$

$$(N=4) \quad \underbrace{bc\ a\ bbbbc\ a\ bbc\ a\ bc\ a}_{x_1 \dots x_{L-N}} \quad \underbrace{aaaa}_{x_{L-N+1} \dots x_L} \tag{A.3}$$

$$\underbrace{bbc\ bc\ bbbc\ bbc\ a\ a\ a\ a}_{x_1 \dots x_{L-N}} \quad \underbrace{aaaa}_{x_{L-N+1} \dots x_L} \tag{A.4}$$

It is *not* possible to construct a csHMM that generates *only* sequences in the above form. This

can be seen from the following. As shown in the above examples, the number of "a's" in the tail $x_{L-N+1} \dots x_L$ is always identical to the number of "a's" and the number of subsequences "$b \dots bc$" in the head part $x_1 \dots x_{L-N}$. As the transition probabilities at single-emission states S_n and pairwise-emission states P_n do not depend on past emissions, the only way to ensure the generation of specific number of "a's" in the tail is to use context-sensitive states C_n, which have variable transition probabilities that depend on the context. As the last N symbols are emitted at context-sensitive states, identical number of symbols in $x_1 \dots x_{L-N}$ have to be emitted at the corresponding pairwise-emission states. Since the number of "a's" and the number of "$b \dots bc$'s" in the head part are both N, we may consider the following two cases. Firstly, we may consider using the corresponding pairwise-emission states to generate the "a's" in the head part. As the emitted symbols at these states are used as the context for generating identical number symbols in the tail, the subsequences "$b \dots bc$" cannot make use of this context. Therefore, the only way to guarantee that the number of "$b \dots bc$'s" are also N is to construct the csHMM such that the states that generate "$b \dots bc$" always follow (or precede) the pairwise-emission states that generate "a's". This is illustrated in Figure A.1. Although this construction guarantees that the number of "a's" and the number of "$b \dots bc$'s" in $x_1 \dots x_{L-N}$ are both N, it cannot give rise to all possible orders of "a" and "$b \dots bc$." For example, such a csHMM cannot generate sequences in (A.2) and (A.4). Similar reasoning also holds when the pairwise-emission states, which correspond to the context-sensitive states used for generating the tail part, are used to generate (part of) the subsequence "$b \dots bc$." This leads to the conclusion that a construction which guarantees the emission of N "a's" and "$b \dots bc$" cannot generate sequences such as (A.2) and (A.4). Therefore, the given context-free language cannot be represented by a csHMM.

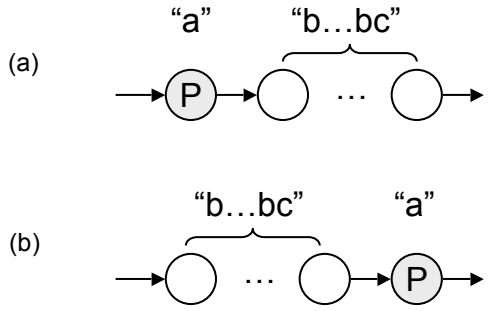


Figure A.1: Constructions which guarantee that the number of "a's" and the number of "$b \dots bc$'s" in $x_1 \dots x_{L-N}$ are identical.

Appendix B

Algorithms for Sequences with Single Nested Correlations

In this appendix, we present the csHMM alignment and scoring algorithms that can be used for analyzing sequences with a single-nested correlation structure. Figure B.1 shows examples of various correlation structures. For example, Figure B.1 (a) and Figure B.1 (b) show symbol sequences with single-nested correlations. Figure B.1 (c) and Figure B.1 (d) show symbol sequences with multiple-nested correlations. In biology, an RNA with a single hairpin (or a stem-loop) is an example of a sequence that has a single-nested correlation structure. An RNA with multiple stem-loops (e.g., a tRNA) is an example that has a multiple-nested correlation structure.

For symbol sequences with single-nested correlations, we can use simplified versions of the alignment and scoring algorithm that were presented in Chapter 2. In the following sections, we describe the simplified algorithms that can be used with these sequences. We use similar notations as in Section 2.4 and Section 2.5 unless specified otherwise.

The content of Section B.1 is mainly drawn from [127], and the content of Section B.2 is mainly drawn from [128].

B.1 Simplified alignment algorithm

In this section, we describe the simplified alignment algorithm. In addition to the variables defined in Section 2.4, we define the variable $\lambda(i, j, v, w)$ that will be used for tracing back the optimal path.

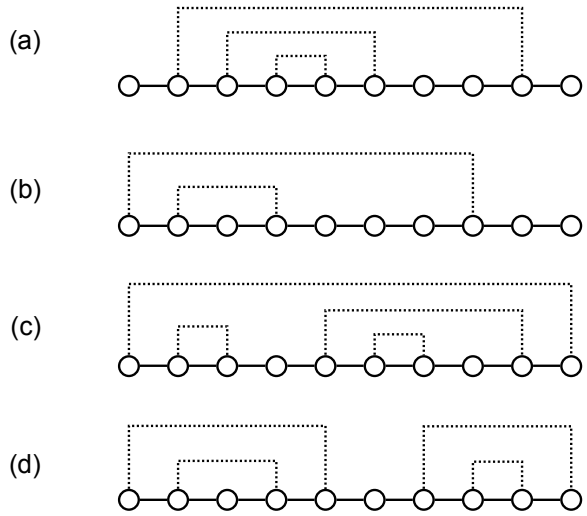


Figure B.1: Examples of symbol sequences with different correlation structures. Sequences with single-nested correlations are shown in (a) and (b), and sequences with multiple-nested correlations are shown in (c) and (d).

B.1.1 Computing the log-probability of the optimal path

We first compute the log-probability of the optimal path using the following algorithm.

(1) Initialization

For $i = 1, \ldots, L, v = 2, \ldots, M-1$.

$$\begin{aligned} \gamma(i,i,v,v) &= \begin{cases} \log e(x_i|v) & v \in \mathcal{S} \\ -\infty & \text{otherwise} \end{cases} \\ \lambda(i,i,v,v) &= (0,0,0,0) \end{aligned}$$

(2) Iteration

For $i = 1, \ldots, L-1, j = i+1, \ldots, L$ and $v = 2, \ldots, M-1, w = 2, \ldots, M-1$.

(i) $v = P_n, w = C_m (n \neq m)$, or $v \in \mathcal{C}$ or $w \in \mathcal{P}$

$$\begin{aligned} \gamma(i,j,v,w) &= -\infty \\ \lambda(i,j,v,w) &= (0,0,0,0) \end{aligned}$$

(ii) $v = P_n, w = C_n, j = i+1$

$$\begin{aligned} \gamma(i,j,v,w) &= \log e(x_i|v) + \log t(v,w) + \log e(x_j|w,x_i) \\ \lambda(i,j,v,w) &= (0,0,0,0) \end{aligned}$$

(iii) $v = P_n, w = C_n, j \neq i+1$

$$\begin{aligned} \gamma(i,j,v,w) &= \max_{u_1,u_2} \Big[\log e(x_i|v) + \log t(v,u_1) \\ &\qquad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i) \Big] \\ (u_1^*,u_2^*) &= \arg\max_{(u_1,u_2)} \Big[\log e(x_i|v) + \log t(v,u_1) \\ &\qquad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i) \Big] \\ \lambda(i,j,v,w) &= (i+1,j-1,u_1^*,u_2^*) \end{aligned}$$

(iv) $v \in \mathcal{P}, w \notin \mathcal{C}$

$$\begin{aligned} \gamma(i,j,v,w) &= \max_{u} \Big[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w) \Big] \\ u^* &= \arg\max_{u} \Big[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w) \Big] \\ \lambda(i,j,v,w) &= (i,j-1,v,u^*) \end{aligned}$$

(v) $v \notin \mathcal{P}, w \in \mathcal{C}$

$$\begin{aligned}
\gamma(i,j,v,w) &= \max_{u} \Big[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w) \Big] \\
u^* &= \arg\max_{u} \Big[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w) \Big] \\
\lambda(i,j,v,w) &= (i+1,j,u^*,w)
\end{aligned}$$

(vi) $v \notin \mathcal{P}, w \notin \mathcal{C}$

In this case, the variables $\gamma(i,j,v,w)$ and $\lambda(i,j,v,w)$ can be updated using any of the update formulae in (iii)–(v).

(3) Termination

$$\begin{aligned}
\log P(\mathbf{x}, \pi^*|\Theta) &= \max_{v,w} \Big[\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M) \Big] \\
(v^*, w^*) &= \arg\max_{(v,w)} \Big[\log t(0,v) + \gamma(1,L,v,w) + \log t(w,0) \Big] \\
\lambda^* &= (1,L,v^*,w^*)
\end{aligned}$$

■

The proposed algorithm starts from the inside of the observation sequence, and proceeds to the outward direction, to find the optimal path iteratively. It should be noted that every time there is an interaction between s_i and s_j, they are considered at the same time as shown in (ii) and (iii) of the **iteration** step. This informs us of the symbol x_i that was emitted by P_n, hence we can adjust the probabilities of the corresponding state C_n according to this value.

B.1.2 Trace-back

Let us define $\lambda_t = (i,j,v,w)$. We also need a stack T for trace-back. The optimal path is traced back as follows.

(1) Initialization

$s_i = 0$ $(i = 1, 2, \ldots, L)$.

Push λ^* onto T.

(2) Iteration

Pop $\lambda_t = (i, j, v, w)$ from stack T.

If $\lambda_t \neq (0, 0, 0, 0)$

If $s_i = 0$ then $s_i = v$.

If $s_j = 0$ then $s_j = w$.

$\lambda(\lambda_t)$ onto T.

If T is empty then go to **termination** step.

Otherwise, repeat the **iteration** step.

(3) Termination

The optimal path is $\mathbf{s}^* = s_1 s_2 \dots s_L$. ■

At the end of this procedure, we get the most probable path $\mathbf{s}^*$. It is not difficult to see that the computational complexity of the alignment algorithm is $O(L^2M^3)$, which is much better than $O(M^L)$ of the exhaustive search, and also smaller than the complexity $O(L^3M^3)$ of the more general alignment algorithm described in Section 2.4.

B.2 Simplified scoring algorithm

In this section, we describe the simplified scoring algorithm.

(1) Initialization

For $i = 1, \dots, L, v = 2, \dots, M-1$.

$$\alpha(i, i, v, v) = \begin{cases} e(x_i|v) & v \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

(2) Iteration

For $i = 1, \ldots, L-1, j = i+1, \ldots, L$ and $v = 2, \ldots, M-1, w = 2, \ldots, M-1$.

(i) $v = P_n, w = C_m (n \neq m)$, or $v \in \mathcal{C}$, or $w \in \mathcal{P}$

$$\alpha(i, j, v, w) = 0$$

(ii) $v = P_n, w = C_n, j = i+1$

$$\alpha(i, j, v, w) = e(x_i|v)t(v, w)e(x_j|w, x_i)$$

(iii) $v = P_n, w = C_n, j \neq i+1$

$$\alpha(i, j, v, w) = \sum_{u_1, u_2} \Big[e(x_i|v)t(v, u_1)\alpha(i+1, j-1, u_1, u_2)t(u_2, w)e(x_j|w, x_i) \Big]$$

(iv) $v \in \mathcal{P}, w \notin \mathcal{C}$

$$\alpha(i, j, v, w) = \sum_{u} \Big[\alpha(i, j-1, v, u)t(u, w)e(x_j|w) \Big]$$

(v) $v \notin \mathcal{P}, w \in \mathcal{C}$

$$\alpha(i, j, v, w) = \sum_{u} \Big[e(x_i|v)t(v, u)\alpha(i+1, j, u, w) \Big]$$

(vi) $v \notin \mathcal{P}, w \notin \mathcal{C}$

We can use either (iv) or (v).

(3) Termination

$$P(\mathbf{x}|\Theta) = \sum_{v, w} t(1, v)\alpha(1, L, v, w)t(w, M) \qquad \blacksquare$$

Note that $t(1, v)$ is the probability that the model will start at state v, and $t(w, M)$ is the probability that the model will terminate after state w. We can see that the probability $\alpha(i, j, v, w)$ is computed iteratively, starting from the inside to the outside. Every time there is a correlation between s_i and s_j, they are considered together as shown in (ii) and (iii) of the **iteration** step. In this way, we can know which symbol was emitted at the pairwise-emission state, and therefore we can

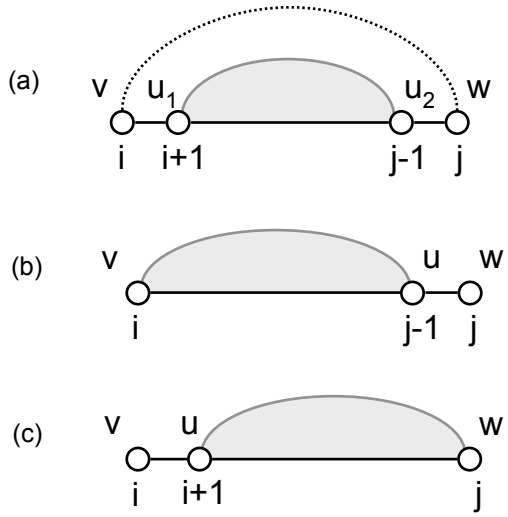


Figure B.2: Illustration of the iteration step of the simplified scoring algorithm.

decide the probabilities of the corresponding context-sensitive state. This is illustrated in Fig. B.2 (a). The dashed line denotes the interaction between $s_i = v$ and $s_j = w$. Since P_n and C_n exist in pairs inside $s_i \cdots s_j$, and as s_i is paired with s_j, all P_n and C_n states inside $s_{i+1} \cdots s_{j-1}$ must also exist in pairs. This is indicated by the shaded area in Figure B.2 (a). Therefore, the probability of $\alpha(i, j, v, w)$ can be computed as in (ii) of the **iteration** step. Similarly, Figs. B.2 (b) and (c) respectively illustrate (iv) and (v) of the **iteration** step. As the simplified alignment algorithm in Section B.1, the overall computational complexity of the simplified scoring algorithm is $O(L^2M^3)$.

Appendix C

Acronyms

In this appendix, we describe the acronyms that are used throughout the thesis.

A	adenine
C	cytosine
CFG	context-free grammar
CM	covariance model
CSG	context-sensitive grammar
csHMM	context-sensitive hidden Markov models
DFT	discrete Fourier transform
DNA	deoxyribonucleic acid
dsRNA	double-stranded RNA
ECG	electrocardiogram
EM algorithm	expectation-maximization algorithm
FIR	finite impulse response
FFT	fast Fourier transform
G	guanine
HMM	hidden Markov model
IDFT	inverse discrete Fourier transform
IIR	infinite impulse response
IRE	iron response element
IRP	iron regulatory protein
MRI	magnetic resonance imaging
mRNA	messenger RNA

ncRNA	noncoding RNA
ORF	open reading frame
PHMMTSs	pair hidden Markov models on tree structures
pre-mRNA	pre-messenger RNA
profile-csHMM	profile context-sensitive hidden Markov model
profile-HMM	profile hidden Markov model
profile-SCFG	profile stochastic context-free grammar
PSTAG	pair stochastic tree adjoining grammar
SCA algorithm	sequential component adjoining algorithm
SCFG	stochastic context-free grammar
SCSG	stochastic context-sensitive grammar
siRNA	small interfering RNA
snoRNA	small nucleolar RNA
SRG	stochastic regular grammar
T	thymine
tmRNA	transfer-messenger RNA
tRNA	transfer RNA
RNA	ribonucleic acid
RNAi	RNA interference
rRNA	ribosomal RNA
U	uracil

Bibliography

[1] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential Cell Biology: An Introducton to the Molecular Biology of the Cell*, New York, NY: Garland Publishing Inc., 1997.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403-410, 1990.

[3] D. Anastassiou, "Genomic signal processing," *IEEE Signal Processing Magazine*, vol. 18, pp. 8-20, 2001.

[4] A. Bairoch and P. Bucher, "PROSITE: Recent developments," *Nucleic Acids Research*, vol. 22, pp. 3583-3589, 1994.

[5] D. P. Bartel, "MicroRNAs: Genomics, biogenesis, mechanism, and function," *Cell*, vol. 116, pp. 281-297, 2004.

[6] L. E. Baum, "An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1-8, 1972.

[7] T. S. Bayer and C. D. Smolke, "Programmable ligand-controlled riboregulators of eykaryotic gene expression," *Nature Biotechnology*, vol. 23, pp. 337-343, 2005.

[8] S. B. Baylin, J. G. Herman, J. R. Graff, P. M. Vertino, and J. P. Issa, "Alterations in DNA methylation: A fundamental aspect of neoplasia," *Advances in Cancer Research*, vol. 72, pp. 141-196, 1998.

[9] A. Bird, "CpG islands as gene markers in the vertebrate nucleus," *Trends in Genetics*, vol. 3, pp. 342-347, 1987.

[10] A. Bird, "DNA methylation patterns and epigenetic memory," *Genes and Development*, vol. 16, pp. 6-21, 1999.

[11] T. A. Brown, *Genomes*, John Wiley and Sons, NY, USA, 2002.

[12] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[13] X. Cai and G. B. Giannakis, "Identifying differentially expressed genes in microarray experiments with model-based variance estimation," *IEEE Transactions on Signal Processing*, vol. 54, pp. 2418-2426, 2006.

[14] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transacations on Image Processing*, vol. 9, pp. 1532-1546, 2000.

[15] V. R. Chechektkin and A. Y. Turygin, "Size-dependence of three-periodicity and long-range correlations in DNA sequences," *Physics Letter A*, vol. 199, pp. 75-80, 1995.

[16] N. Chomsky, "On certain formal properties of grammars," *Information and Control*, vol. 2, pp. 137-167, 1959.

[17] A. Condon, B. Davy, B. Rastegari, S. Zhao, and F. Tarrant, "Classifying RNA pseudoknotted structures," *Theoretical Computer Science*, vol. 320, pp. 35-50, 2004.

[18] A. Coventri, D. J. Kleitman, and B. Berger, "MSARI: Multiple sequence alignments for statistical detection of RNA secondary structure," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, pp. 12102-12107, 2004.

[19] N. Dasgupta, S. Lin and L. Carin, "Sequential modeling for identifying CpG island locations in human genome," *IEEE Signal Processing Letters*, vol. 9, pp. 407-409, 2002.

[20] S. Datta and A. Asif, "DFT based DNA splicing algorithms for prediction of protein coding regions," *Proceedings of the 30th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Philadelphia, March 2005.

[21] D. di Bernardo, T. Down, and T. Hubbard, "ddbRNA: Detection of conserved secondary structures in multiple alignments," *Bioinformatics*, vol. 19, pp. 1606-1611, 2003.

[22] E. R. Dougherty and A. Datta, "Genomic signal processing: Diagnosis and therapy," *IEEE Signal Processing Magazine*, vol. 22, pp. 107-112, 2005.

[23] E. R. Dougherty, A. Datta, and C. Sima, "Research issues in genomic signal processing," *IEEE Signal Processing Magazine*, vol. 22, pp. 46-68, 2005.

[24] P. Duhamel and M. Vetterli, "Fast Fourier transforms: A tutorial review and a state of the art," *Signal Processing*, vol. 19, pp. 259-299, 1990.

[25] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*, Cambridge, UK: Cambridge University Press, 1998.

[26] S. R. Eddy and R. Durbin, "RNA sequence analysis using covariance models," *Nucleic Acids Research*, vol. 22, pp. 2079-2088, 1994.

[27] S. R. Eddy, "Multiple alignment using hidden Markov models," *Proceedings of the Third International Converence on Intelligent Systems for Molecular Biology*, pp. 112-120, 1995.

[28] S. R. Eddy, "Hidden Markov models," *Current Opinion in Structural Biology*, vol. 6, pp. 361-365, 1996.

[29] S. R. Eddy, "Profile hidden Markov models," *Bioinformatics*, vol. 14, pp. 755-763, 1998.

[30] S. R. Eddy, "Non-coding RNA genes and the modern RNA world," *Nature Reviews Genetics*, vol. 2, pp. 919-929, 2001.

[31] S. R. Eddy, "Computational genomics of noncoding RNA genes," *Cell*, vol. 109, pp. 127-140, 2002.

[32] S. R. Eddy, "How do RNA folding algorithms work?," *Nature Biotechnology*, vol. 22, pp. 1457-1458, 2004.

[33] Y. Ephraim and H. L. Van Trees, "A signal subspace approach for speech enhancement," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Minneapolis, April 1993.

[34] V. A. Erdmann, M. Z. Barciszewska, M. Szymanski, A. Hochberg, N. de Groot, and J. Barciszewski, "The non-coding RNAs as riboregulators," *Nucleic Acids Research*, vol. 29, pp. 189-193, 2001.

[35] S. Faisan, L. Thoraval, J.-P. Armspach, M.-N. Metz-Lutz, and F. Heitz, "Unsupervised learning and mapping of active brain functional MRI signals based on hidden semi-Markov event sequence models," *IEEE Transactions on Medical Imaging*, vol. 24, pp. 263-276, 2005.

[36] R. Feil and S. Khosla, "Genomic imprinting in mammals: an interplay between chromatin and DNA methylation?," *Trends in Genetics*, vol. 15, pp. 429-474, 1999.

[37] J. W. Fickett and C. S. Tung, "Assessment of protein coding measures," *Nucleic Acids Research*, vol. 20, pp. 6441-6450, 1992.

[38] J. W. Fickett, "The gene prediction problem: an overview for developers," *Computers and Chemistry*, vol. 20, pp. 103-118, 1996.

[39] A. Fire, S. Xu, M. K. Montgomery, S. A. Kostas, S. E. Driver, and C. C. Mello, "Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*," *Nature*, vol. 391, pp. 806-811, 1998.

[40] M. Gardiner-Garden and M. Frommer, "CpG islands in vertebrate genomes," *Journal of Molecular Biology*, vol. 196, pp. 261-282, 1987.

[41] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York, NY: W. H. Freeman, 1979.

[42] G. B. Giannakis, Y. Hua, P. Stoica, and L. Tong, *Signal Processing Advances in Wireless and Moblie Communications, Volume 1: Trends in Channel Estimation and Equalization*, Upper Saddle River, NJ: Prentice-Hall, 2000.

[43] G. B. Giannakis, Y. Hua, P. Stoica, and L. Tong, *Signal Processing Advances in Wireless and Moblie Communications, Volume 2: Trends in Single- and Multi-User Systems*, Upper Saddle River, NJ: Prentice-Hall, 2000.

[44] G. Storz, "An expanding universe of noncoding RNAs," *Science*, vol. 296, pp. 1260-1263, 2002.

[45] J. Gorodkin, L. J. Heyer, and G. D. Stormo, "Finding the most significant common sequence an structure motifs in a set of RNA sequences," *Nucleic Acids Research*, vol. 25, pp. 3724-3732, 1997.

[46] S. Gottesman, "Stealth regulation: Biological circuits with small RNA switches," *Genes and Development*, vol. 16, pp. 2829-2842, 2002.

[47] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy, "Rfam: An RNA family database," *Nucleic Acids Research*, vol. 31, pp. 439-441, 2003.

[48] R. Guan and J. Tuqan, "IIR filter deesign for gene identification," *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, Baltimore, MD, May 2004.

[49] Y. Guédon, "Estimating hidden semi-Markov chains from discrete sequences," *Journal of Computational and Graphical Statistics*, vol. 12, pp. 604-639, 2003.

[50] Y. Guédon, "Hidden hybrid Markov/semi-Markov chains," *Computational Statistics and Data Analysis*, vol. 49, pp. 663-688, 2005.

[51] M. A. Harrison, *Introduction to Formal Language Theory*, Reading, MA: Addison-Wesley, 1978.

[52] L. He and J. Hannon, "MicroRNAs: Small RNAs with a big role in gene regulation," *Nature Reviews Genetics*, vol. 5, pp. 522-531, 2004.

[53] T. M. Henkin and C. Yanofsky, "Regulation by transcription attenuation in bacteria: How RNA provides instructions for transcription termination/antitermination decisions," *Bioessays*, vol. 24, pp. 700-707, 2002.

[54] M. W. Hentze and L. C. Kuhn, "Molecular control of vertebrate iron metabolism: mRNA-basedregulatory circuits operated by iron, nitric oxide, and oxidative stress," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, pp. 8175-8182, 1996.

[55] P. A. Jones and P. W. Laird, "Cancer-epigenetics comes of age," *Nature Genetics*, vol. 21, pp. 163-167, 1999.

[56] F. Jelinek, *Statistical Methods for Speech Recognition*, Cambridge, MA: The MIT Press, 2001.

[57] R. J. Klein and S. R. Eddy, "RSEARCH: Finding homologs of single structured RNA sequences," *BMC Bioinformatics*, vol. 4, 44, 2003.

[58] A. Krogh, I. Saira Mian, D. Haussler, "A hidden Markov model that finds genes in *E. coli* DNA," *Nucleic Acids Research*, vol. 22, pp. 4768-4778, 1994.

[59] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden Markov models in computational biology: applications to protein modeling," *Journal of Molecular Biology*, vol. 235, pp. 1501-1531, 1994.

[60] K. Lari and S. J. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm," *Computer Speech and Language*, vol. 4, pp. 35-56, 1990.

[61] F. Larsen, G. Gundersen, R. Lopez, and H. Prydz, "CpG islands as gene markers in the human genome," *Genomics*, vol. 13, pp. 1095-1107, 1992.

[62] K. Lee, "Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 599-609, 1990.

[63] C. Li, C. Zheng, and C. Tai, "Detection of ECG characteristic points using wavelet transforms," *IEEE Transactions on Biomedical Engineering*, vol. 42, pp. 21-28, 1995.

[64] A. Ljolje, "High accuracy phone recognition using context clustering and quasi-triphonic models," *Computer Speech and Language*, vol. 8, pp. 129-151, 1994.

[65] M. Mandal and R. R. Breaker, "Gene regulation by riboswitches," *Nature Reviews Molecular Cell Biology*, vol. 5, pp. 451-463, 2004.

[66] H. Matsui, K. Sato, and Y. Sakakibara, "Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures," *Bioinformatics*, vol. 21, pp. 2611-2617, 2005.

[67] J. S. Mattick, "Challenging the dogma: The hidden layer of non-protein-coding RNAs in complex organisms," *BioEssays*, vol. 25, pp. 930-939, 2003.

[68] M. A. Matzke and J. A. Birchler, "RNAi-mediated pathways in the nucleus," *Nature Reviews Genetics*, vol. 6, 2005.

[69] M. T. McManus and P. A. Sharp, "Gene silencing in mammals by small interfering RNAs," *Nature Reviews Genetics*, vol. 3, 2002.

[70] M. D. Moore and M. I. Savic, "Speech reconstruction using a generalized HSMM (GHSMM)," *Digital Signal Processing*, vol. 14, pp. 37-53, 2004.

[71] V. Moulton, "Tracking down noncoding RNAs," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, pp. 2269-2270, 2005.

[72] Y. Neuvo, C.-Y. Dong, S. K. Mitra, "Interpolated finite impulse response filter," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, pp. 563-570, 1984.

[73] P. Noll, "MPEG digital audio coding," *IEEE Signal Processing Magazine*, vol. 14, pp. 59-81, 1997.

[74] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Procesing*, Englewood Cliffs, NJ: Prentice Hall, 1999.

[75] 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, pp. 2444-2448, 1988.

[76] 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, pp. 168-170, 1992.

[77] G. Pesole, S. Liuni, and M. D'Souza, "PatSearch: A pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance," *Bioinformatics*, vol. 16, pp. 439-450, 2000.

[78] G. Pesole, S. Liuni, G. Grillo, F. Licciulli, F. Mignone, and C. Saccone, "UTRdb and UTRsite: specialized databases of sequences and functional elements of 5' and 3' untranslated regions of eukaryotic mRNAs.," *Nucleic Acids Research*, vol. 30, pp. 335-340, 2002.

[79] W. Pieczynski, C. Hulard, and T. Veit, "Triplet Markov chains in hidden signal restoration," *Proceedings of the SPIE's International Symposium on Remote Sensing*, Crete, Greece, September 2002.

[80] W. Pieczynski, "Pairwise Markov chains," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 634-639, 2003.

[81] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989.

[82] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Upper Saddle River, NJ: Prentice-Hall, 1993.

[83] P. A. Regalia, S. K. Mitra, and P. P. Vaidyanathan, "The digital allpass filter: A versatile signal processing building block," *Proceeding of the IEEE*, vol. 76, pp. 19-37, 1988.

[84] M. Richards, *Radar signal processing*, McGraw-Hill Professional, New York, NY, 2005.

[85] E. Rivas and S. R. Eddy, "A dynamic programming algorithm for RNA structure prediction including pseudoknots," *Journal of Molecular Biology*, vol. 285, pp. 2053-2068, 1999.

[86] E. Rivas and S. R. Eddy, "The language of RNA: A formal grammar that includes pseudoknots," *Bioinformatics*, vol. 16, pp. 334-340, 2000.

[87] E. Rivas and S. R. Eddy, "Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs," *Bioinformatics*, vol. 16, pp. 583-605, 2000.

[88] E. Rivas and S. R. Eddy, "Noncoding RNA gene detection using comparative sequence analysis," *BMC Bioinformatics*, vol. 2, 8, 2001.

[89] S. M. Ross, *Introduction to Probability Models*, San Diego, CA: Academic Press, 2000.

[90] E. C. Rouchka, R. Mazzarella, and D. J. States, "Computational detection of CpG islands in DNA," *Technical Report*, Washington University, Department of Computer Science, WUCS-97-39, 1997.

[91] J. Ruan, G. D. Stormo, and W. Zhang, "An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots," *Bioinformatics*, vol. 20, pp. 58-66, 2004.

[92] A. Rushdi and J. Tuqan, "Gene identification using the Z-curve representation," *Proceedings of the 31st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, May 2006.

[93] G. Ruvkun, "Glimpses of a tiny RNA world," *Science*, vol. 294, pp. 797-799, 2001.

[94] J. S. Sahambi, S. N. Tandon, and R. K. P. Bhatt, "Using wavelet transforms for ECG characterization. An on-line digital signal processing system," *IEEE Engineering in Medicine and Biology Magazine*, vol. 16, pp. 77-83, 1997.

[95] Y. Sakakibara, "Pair hidden Markov models on tree structures," *Bioinformatics*, vol. 19, i232-i240, 2003.

[96] S. L. Salzberg, A. L. Delcher, S. Kasif, O. White, "Microbial gene identification using interpolated Markov models," *Nucleic Acids Research*, vol. 26, pp. 544-548, 1998.

[97] R. Schwartz, J. Klovstadt, L. Makhoul, and J. Sorensen, "Improved hidden Markov modeling of phonemes for continuous speech recognition," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 872-875, Denver, 1980.

[98] P. Y. Shi, M. A. Brinton, J. M. Veal, Y. Y. Zhong, and W. D. Wilson, "Evidence for the existence of a pseudoknot structure at the 3' terminus of the flavivirus genomic RNA," *Biochemistry*, vol. 35, pp. 4222-4230, 1996.

[99] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks," *Proceedings of IEEE*, vol. 90, pp. 1778-1792, 2002.

[100] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, pp. 36-58, 2001.

[101] J. L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Transactions on Image Processing*, vol. 11, pp. 670-684, 2002.

[102] N. Sudarsan, J. E. Barrick, and R. R. Breaker, "Metabolite-binding RNA domains are present in the genes of eukaryotes," *RNA*, vol. 9, pp. 644-647, 2003.

[103] 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, pp. 3740-3745, 2002.

[104] E. B. ten Dam, C. W. A. Pleij, and D. Draper, "Structural and functional aspects of RNA pseudoknots," *Biochemistry*, vol. 31, pp. 11665-11676, 1992.

[105] The Human Genome Sequencing Consortium, "Finishing the euchromatic sequence of the human genome," *Nature*, vol. 431, pp. 931-945, 2004.

[106] S. Tiwari, S. Ramachandran, A. Bhattacharya, S. Bhattacharya, and R. Ramaswamy, "Prediction of probable genes by Fourier analysis of genomic sequences," *Computer Applications in the Biosciences*, vol. 13, pp. 263-270, 1997.

[107] E. N. Trifonov and J. L. Sussman, "The pitch of chromatin DNA is reflected in its nucleotide sequence," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 77, pp. 3816-3820, 1980.

[108] B. J. Tucker and R. R. Breaker, "Riboswitches as versatile gene control elements," *Current Opinion in Structural Biology*, vol. 15, pp. 342-348, 2005.

[109] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Englewood Cliffs, NJ: Prentice Hall, 1993.

[110] P. P. Vaidyanathan and B.-J. Yoon, "Gene and exon prediction using allpass-based filters," *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, Raleigh, NC, October 2002.

[111] P. P. Vaidyanathan and B.-J. Yoon, "Digital filters for gene prediction applications," *Proceedings of the 36th Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA, November 2002.

[112] P. P. Vaidyanathan and B.-J. Yoon, "The role of signal-processing concepts in genomics and proteomics," *Journal of the Franklin Institute*, vol. 341, pp. 111-135, 2003.

[113] P. P. Vaidyanathan, "Genomics and proteomics: A signal processor's tour," *IEEE Circuits and Systems Magazine*, vol. 4, pp. 6-29, 2005.

[114] F. H. D. van Batenburg, A. P. Gultyaev, C. W. A. Pleij, J. Ng, and J. Oliehoek, "Pseudobase: a database with RNA pseudoknots," *Nucleic Acids Research*, vol. 28, pp. 201-204, 2000.

[115] H. Vikalo, B. Hassibi, and A. Hassibi, "A statistical model for microarrays, optimal estimation algorithms, and limits of performance," *IEEE Transactions on Signal Processing*, vol. 54, pp. 2444-2455, 2006.

[116] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260-267, 1967.

[117] R. F. Voss, "Evolution of long-range fractal correlations and $1/f$ noise in DNA base sequences," *Physical Review Letters*, vol. 68, pp.3805-3808, 1992.

[118] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 30-44, 1991.

[119] S. Washietl, I. L. Hofacker, and P. F. Stadler, "Fast and reliable prediction of noncoding RNAs," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, pp. 2454-2459, 2005.

[120] S. Washietl, I. L. Hofacker, M. Lukasser, A. Hüttenhofer and P. F. Stadler, "Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome," *Nature Biotechnology*, vol. 23, pp. 1383-1390, 2005.

[121] Z. Weinberg and W. L. Ruzzo, "Exploiting conserved structure for faster annotation of noncoding RNAs without loss of accuracy," *Bioinformatics*, vol. 20, Supplement 1, pp. i334-i340. 2004.

[122] Z. Weinberg and W. L. Ruzzo, "Sequence-based heuristics for faster annotation of noncoding RNA families," *Bioinformatics*, vol. 22, pp. 35-39, 2006.

[123] B.-J. Yoon and P. P. Vaidyanathan, "HMM with auxiliary memory: A new tool for modeling RNA secondary structures," *Proceedings of the 38th Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA, November 2004.

[124] B.-J. Yoon and P. P. Vaidyanathan, "Identification of CpG islands using a bank of IIR lowpass filters," *Proceedings of the 11th Digital Signal Processing Workshop*, Taos Ski Valley, New Mexico, August 2004.

[125] B.-J. Yoon and P. P. Vaidyanathan, "RNA secondary structure prediction using context-sensitive hidden Markov models," *Proceedings of the International Workshop on Biomedical Circuits and Systems (BioCAS)*, Singapore, December 2004.

[126] B.-J. Yoon and P. P. Vaidyanthan, "An overview of the role of context-sensitive HMMs in the prediction of ncRNA genes," *Proceedings of the IEEE Workshop on Statistical Signal Processing*, Bordeaux, France, July 2005.

[127] B.-J. Yoon and P. P. Vaidyanthan, "Optimal alignment algorithm for context-sensitive hidden Markov models," *Proceedings of the 30th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Philadelphia, March 2005.

[128] B.-J. Yoon and P. P. Vaidyanthan, "Scoring algorithm for context-sensitive HMMs with application to RNA secondary structure analysis," *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, Newport, RI, May 2005.

[129] B.-J. Yoon and P. P. Vaidyanathan, "Profile context-sensitive HMMs for probabilistic modeling of sequences with complex correlations," *Proceedings of the 31st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, May 2006.

[130] B.-J. Yoon and P. P. Vaidyanathan, "Computational identification and analysis of noncoding RNAs–Unearthing the buried treasures in the genome," *IEEE Signal Processing Magazine*, vol. 24, pp. 64-74, 2007.

[131] B.-J. Yoon and P. P. Vaidyanathan, "Context-sensitive hidden Markov models for modeling long-range dependencies in symbol sequences," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4169-4184, 2006.

[132] B.-J. Yoon and P. P. Vaidyanthan, "Modeling and identification of alternative folding in regulatory RNAs using context-sensitive HMMs ," *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, College Station, Texas, May 2006.

[133] B.-J. Yoon and P. P. Vaidyanathan, "Structural Alignment of RNAs Using Profile-csHMMs and Its Application to RNA Homology Search: Overview and New Results," *IEEE Transactions on Circuits and Systems: Part-I*, submitted.

[134] B.-J. Yoon and P. P. Vaidyanathan, "Fast search of sequences with complex symbol correlations using profile context-sensitive HMMs and pre-screening filters," *Proceedings of the 32nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, April 2007.

[135] Z.-G. Yu, V. V. Anh, and B. Wang, "Correlation property of length sequences based on global structure of the complete genome," *Physical Review E*, vol. 63, pp. 011903-1–011903-8, 2000.

[136] S.-Z. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden Markov model," *IEEE Signal Processing Letters*, vol. 10, pp. 11-14, 2003.

[137] V. B. Zhurkin, "Periodicity in DNA-primary structure is defined by secondary structure of the coded protein," *Nucleic Acids Research*, vol. 9, pp. 1963-1971, 1981.