CHAPTER

Identifying *CpG* Islands: Sliding Window and Hidden Markov Model Approaches 9

Raina Robeva[*], **Aaron Garrett**[†], **James Kirkwood**[*] **and Robin Davies**[‡]

[*]*Department of Mathematical Sciences, Sweet Briar College, Sweet Briar, VA, USA*
[†]*Department of Mathematical, Computing, and Information Sciences, Jacksonville State University, Jacksonville, AL, USA*
[‡]*Department of Biology, Sweet Briar College, Sweet Briar, VA, USA*

9.1 INTRODUCTION

The dinucleotide CpG (a cytosine followed by a guanine on a single DNA strand) has a pattern of non-homogeneity along the genome. Regions of relatively low CpG frequencies are interrupted by clusters with markedly higher C and G content, known as CpG islands (CGIs). CGIs are often associated with the promoter regions of genes, and methylation of the promoter CGI is associated with the transcriptional silence of the gene. Conversely, promoter-associated CGIs in constitutively expressed housekeeping genes are unmethylated. Appropriate methylation of CGIs is required for normal development, and inappropriate methylation of CGIs in tumor suppressor promoters has been associated with the development of numerous human cancers.

9.1.1 Biochemistry Background

In higher multicellular organisms the genetic composition of an individual is determined by the fusion of sperm and egg nuclei following fertilization. With a few exceptions[1] all cells of a multicellular organism have the same DNA sequence. However, the cells of the multicellular organism have very different patterns of gene expression and thus make very different groups of proteins. During the process of development, cells become differentiated and take on their mature pattern of gene expression. The following question is thus important: Once a tissue is produced during development, how is the tissue-specific pattern of gene expression maintained? Part of the answer lies with the production of specific proteins involved in the transcription of specific genes, part involves the histones which package the DNA, and another part of the answer lies in the chemical alteration of the DNA itself.

In complex organisms a fraction of the cytosine DNA bases may be methylated, with the degree of cytosine methylation varying considerably among fungi, plants, invertebrate and vertebrate animals [1]. Methylation of cytosine occurs on the #5

[1]Those include red blood cells, which have no DNA, the lymphocytes of the immune system, in which DNA has been rearranged, and gametes, which have half of the adult's DNA.

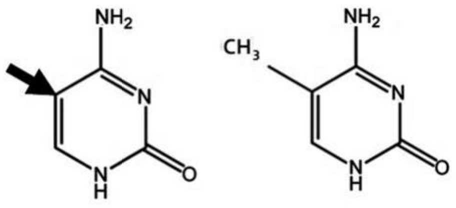


FIGURE 9.1

Comparison of unmethylated (left panel) and methylated (right panel) cytosines. The arrow in the left panel marks the #5 position. (The carbons and nitrogens are numbered, in this case, counter-clockwise beginning with the nitrogen on the bottom.)

position (see Figure 9.1) and the resulting entity is called 5-methyl cytosine. If we were to compare the DNA of a pair of differentiated cells (e.g., liver cells or skin cells), we would observe that they had different patterns of methylation. Patterns of methylation are correlated with patterns of gene expression in an inverse relationship, in which silent (non-expressed) genes are methylated. In a particular differentiated cell type, the pattern of methylation is maintained through successive mitoses by the action of enzymes called maintenance methylases.

In vertebrate animals, methylated cytosines occur in the dinucleotide sequence CpG. This dinucleotide is interesting in that its complement on the other strand of DNA is also CpG, and if the C on one strand is methylated, the C on the other strand is too. This state of affairs enables the pattern of DNA methylation to be perpetuated through successive rounds of replication. When a DNA sequence containing methylated C in a CpG dinucleotide is replicated, the two daughter strands will each have the C on the template strand methylated and the C on the new strand unmethylated. DNA in this state of half-methylation is the substrate for the maintenance methylase, which will methylate the unmethylated C on the new strand and thus restore the methylation pattern of the parent DNA strand. The methylation pattern, and the pattern of gene expression, will be inherited through subsequent mitoses.

Methylation of CpG dinucleotides is required for normal embryonic development and patterns of CpG methylation must be established following a generalized demethylation that occurs early in embryonic development [2]. The new methylation patterns are established by *de novo* methylases and appear to contribute to lineage restriction during development [3,4]. In other words, when pluripotent stem cells give rise to tissue-specific stem cells with more limited differentiation potential, the promoters of a subset of genes which were formerly active in the pluripotent stem cells become methylated and transcriptionally silent [5].

9.1.2 *CpG* Islands

Over time, both methylated and unmethylated cytosines may undergo random deamination reactions. The impact of these deamination reactions differs depending upon

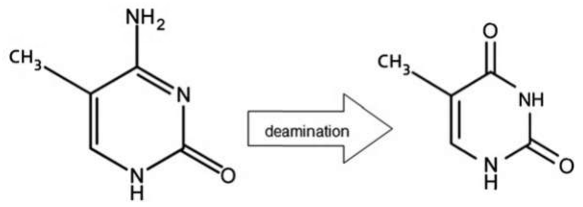


FIGURE 9.2
5-Methyl cytosine is deaminated to thymine.

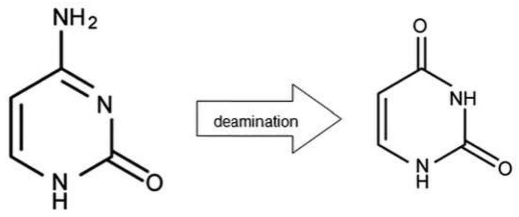


FIGURE 9.3
Cytosine is deaminated to uracil.

the methylation state of the cytosine. Methylated cytosine is deaminated to thymine, while unmethylated cytosine is deaminated to uracil (see Figures 9.2 and 9.3). Uracil bases do not belong in DNA and are recognized and removed by a repair enzyme, which restores the uracil to a *C* in its position. The *T* represents a mismatch (as it does not pair with *G*) and many of these Ts may also be repaired through the action of thymine-DNA glycosylase [6], but many will escape repair and the mutation will be propagated by subsequent rounds of replication. This gives rise to a *C* to *T* transition and the loss of a *CpG*. This means that, on an evolutionary timescale, unmethylated *C*s tend to be preserved and methylated *C*s tend to be eliminated.

Because of this systemic depletion of methylated cytosines, vertebrates have many fewer *CpG* dinucleotides than would be predicted by chance. Interestingly, when the DNA of vertebrates is examined, sequence information reveals that many of the *CpG* dinucleotides that do remain tend to occur in CGIs. The length of these CGIs varies, but they have been reported to range from several hundred to several thousand nucleotides. CGIs are found before genes (i.e., in their promoter regions), in the coding sequences themselves, and after the coding region as well.

The cytosines in CGIs in promoters are normally unmethylated, and many of these un-methylated CGIs are found in the promoters of important housekeeping genes—the genes that must function in order to support life, including the genes for enzymes involved in aerobic respiration, transcription, and translation [7]. Since the *C*s in the *CpG* islands are unmethylated, any deamination events would be detected and repaired and they would not be lost over evolutionary time. Since all cells, even those which give rise to sperm and eggs, must express their housekeeping genes, promoters of housekeeping genes would retain their CpG dinucleotides and appear as CGIs [7].

However, CGIs are not confined to the promoters of housekeeping genes. CGIs are found in the promoters and protein coding regions (exons) of about 40% of mammalian genes [8]. Since CGIs seem to be located at the promoter regions of many known genes, identifying CGIs may be a useful method for identifying new genes.

9.1.3 DNA Methylation in Cancer

DNA methylation is a powerful mechanism for gene silencing. Genes which are methylated and repressed in vertebrate organisms are effectively shut off. For example, hemoglobin genes should be methylated (and silent) in skin cells but unmethylated (and actively expressed) in red blood cell precursors.

The generation of cancer requires multiple changes in gene structure and function. A single mutation is not enough to transform a normal cell into a cancerous cell; between three and seven mutations or other genetic insults are required [9,10]. The affected genes—those which are involved in the progression from normal cell to cancer cell—fall into two different categories: oncogenes and tumor suppressors. Oncogenes must be activated, and tumor suppressors must be inactivated, in order for cancer to develop. Tumor suppressors may be inactivated through mutation or through gene silencing.

Mutations of tumor suppressors were identified first. In hereditary retinoblastoma, a cancer of the retina, affected children have inherited a defective copy of the gene for the tumor suppressor *Rb* from one of their parents. Tumors often arise in both eyes of these children following the loss of the second copy of *Rb* (inherited from the other parent) [11]. Additional tumor suppressor gene mutations were discovered by examining the DNA of many different tumors and comparing that DNA to that of normal tissue. For example, the tumor suppressor *p16*, a cell cycle control protein, was found to be deleted or to have suffered mutations in many different types of cancer [12]. Tumor suppressor genes do not need to be mutated to contribute to the genesis of cancer, however. They merely need to be silenced. If the CGIs in the promoters of *p16* or *Rb* genes are inappropriately methylated and the genes are turned off, then the cell in question will be one step closer to the development of a cancer.

9.2 QUANTITATIVE CHARACTERISTICS OF THE *CpG* ISLAND REGIONS AND SLIDING WINDOWS ALGORITHMS

CGIs were first characterized quantitatively 25 years ago by Gardiner-Garden and Frommer [13] and their definition is still widely in use today. The terms *percent combined* $C + G$ *content* ($\%C + G$) and *observed over expected CpG ratio* ($O/E\ CpG$) introduced by the authors provide a way to identify genomic regions with higher frequencies of C and G nucleotides and CpG dinucleotides. The $\%C + G$

of a sequence is the fraction of the combined number of Cs and Gs in the sequence divided by the total number of nucleotides in the sequence. To define $O/E\ CpG$, we note that if dinucleotides in a DNA sequence were formed by random independent choices of two nucleotides, the expected number of *CpG* dinucleotides in a sequence of length *l* would be

$$(\text{number of } C\text{s in the sequence}) * (\text{number of } G\text{s in the sequence})/l.$$

The observed CpG would be the actual count of CpG dinucleotides found in the sequence of length *l*. The observed over the expected CpG ratio O/E is defined as the ratio of these two numbers (and, unlike the quantity $\%C+G$, may assume values greater than 1).

In the original study published in 1987 [13], Gardiner-Garden and Frommer defined CGIs in the vertebrate genome as sequences that have: (1) length of at least 200 bp, (2) $\%C+G \geqslant 50\%$, and (3) $O/E\ CpG \leqslant 0.6$. This definition is still commonly used today but it serves more as a guideline since there is no universal standard for the cutoff values. For instance, Takai and Jones [8] used a more stringent criterion to analyze CGIs in human chromosome 21 and 22: (1) length $\geqslant$ 500 bp, (2) $\%C+G \geqslant 0.55$, and (2) $O/E\ CpG \geqslant .65$ motivated by reducing the number of CGIs found within *Alus*.[2]

Algorithms for extracting CGIs often utilize a sliding windows approach that has been implemented by many web-based software systems including CpGPlot/CpGReport [14], CpGProd [15], CpGIS [8], and CpGIE [16]. The method calculates the $\%C+G$ and $O/E\ CpG$ for subsequences of fixed length *l* that differ from one another only by 1 bp (the new subsequence is offset by 1 bp to the right from the previous one). One can visualize the process as sliding a "window" of length *l* along the genome. If the subsequence in the window meets the specific cutoff values for $\%C+G$ and $O/E\ CpG$, it will be included in a (possibly larger) CpG island region. The details of the specific algorithm implemented by CpGIS are shown in Figure 9.4. An animated version of a sliding windows algorithm is available in the *CpG Educate* suite that has been developed for this chapter and is available at http://inspired.jsu.edu/~agarrett/cpg/.

The project *Investigating Predicted Genes* available online from the volume's website as part of this chapter utilizes sliding windows software to search for the presence of CGIs in the vicinity of predicted genes. The existence of a CGI in the area of the sequence where the promoter for the predicted gene should be found would be an additional piece of evidence suggesting that the predicted gene may be, in fact, an actual gene.

Sliding windows algorithms are not based on any specific assumptions of structural mechanisms (mathematical or biological) that can explain the differences in CpG density between the island and non-island regions. As such, they do not utilize any mathematical models, theory, or specialized tools to make the questions of CGI identification more tractable. Sliding windows algorithms often differ from one

[2] *Alu* sequences (named for the restriction endonuclease AluI, which cuts in these sequences) are short repetitive sequences with a relatively high $C+G$ content and $O/E\ CpG$ ratio.

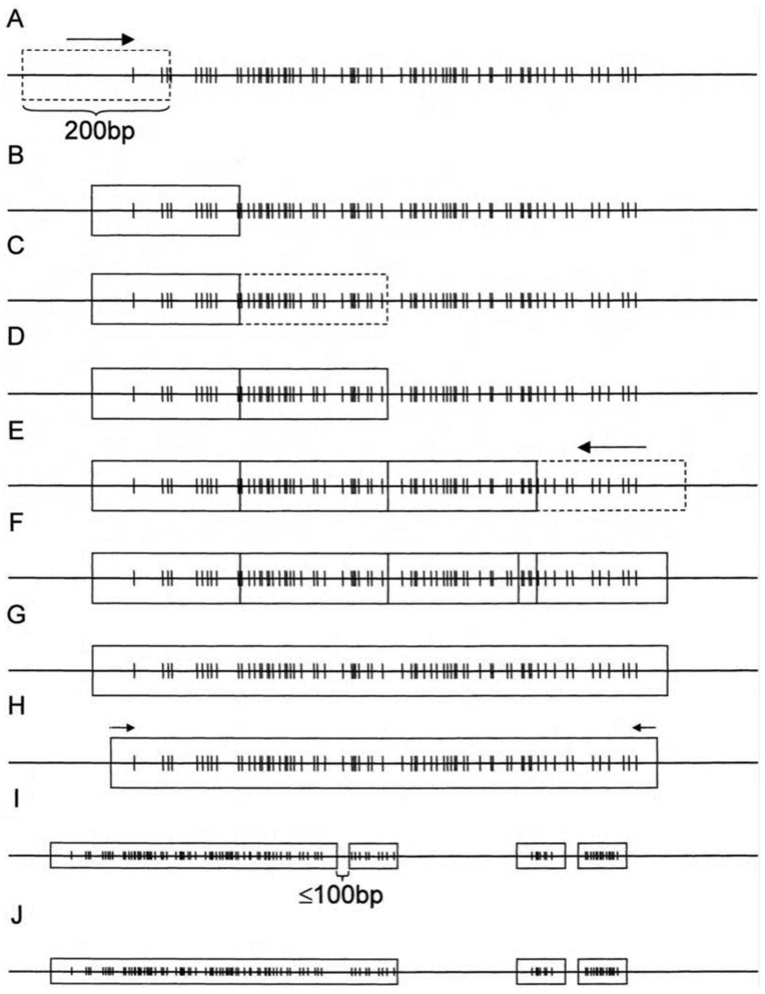


FIGURE 9.4

Schematics for the sliding window algorithms for CGI extraction from human genome sequences from [8]. (A) Set a 200-base window in the beginning of a contig (sequence), compute $C + G$ content and O/E *CpG* ratio. Shift the window 1 bp after evaluation until the window meets the criteria of a CGI. (B) If the window meets the criteria, shift the window 200 bp and then evaluate again. (C and D) Repeat these 200 bp shifts until the window does not meet the criteria. (E) Shift the last window 1 bp toward the 5′ end until it meets the criteria. (G) Evaluate $C + G$ content and O/E *CpG* ratio. (H) If this large CGI does not meet the criteria, trim 1 bp from each side until it meets the criteria. (I) Two individual CGIs were connected if they were separated by less than 100 bp. (J) Values for O/E *CpG* and $C + G$ content were recalculated to remain within the criteria. Reprinted with permission from Takai, D., Jones, P. Comprehensive analysis of CpG islands in human chromosomes 21 and 22. *PNAS* 2002 99 (6) 3740–3745. Copyright (2002) National Academy of Sciences, USA.

another not only in the choice of the threshold values for the length, $\%C + G$, and $O/E\ CpG$ ratio for a sequences to be recognized as a CGI but also in whether islands that are separated by small gaps would be merged if they still meet the cutoff criteria. In addition, some CGI searcher sites use a modified criterion for CGIs, requiring that the $\%C + G$ and $O/E\ CpG$ thresholds are met over an average of several windows (e.g., EMBOSS uses an average of 10 windows). For the same DNA sequences, these differences in the algorithms would generally lead to different regions identified as CGIs. More importantly, it has been shown that the sliding windows method does not guarantee an exhaustive search and that it may fail to identify all regions on the genome that meet the established criteria [17]. The use of alternative methods for CGI identification, such as Hidden Markov Models (HMMs) [18] and clustering methods [19–22] would therefore be preferable.

For the rest of this chapter we consider the HMM approach to locating CGIs. Such models are based on assumptions about the distribution of the nucleotides and dinucleotide in the genome and provide a convenient mathematical framework within which the question of locating the island regions translates into well-understood mathematical problems. We begin with some examples.

The frequencies in Tables 9.1 and 9.2 present an example of nucleotide frequencies obtained from a sequence of annotated human DNA of about 60,000 nucleotides with known locations and lengths of the islands [23]. There are notable differences in the distributions, in agreement with the expectations that island regions would have elevated $\%C + G$ content and higher frequencies of the CpG dinucleotide. For unannotated sequences those frequencies would be unknown and we would want

Table 9.1 Sample dinucleotide frequencies (from [23]). The first row represents the frequencies of the transitions from A to A, C, T, and G in island and non-island regions and similarly for the other rows. Note that G is a lot more likely to follow C in island regions. The transition frequencies have been computed from annotated DNA as follows. If a_{ij}^{+} stands for the transition frequency (transition probability) from letter i to letter j in the island region, where $i,j \in Q, Q = \{A,C,T,G\}$, then a_{ij}^{+} is computed as the ratio $a_{ij}^{+} = \frac{c_{ij}^{+}}{\sum_{k\in Q} c_{ik}}$, where $c_{ij}{}^{+}$ is the number of times the letter i followed by the letter j in the annotated island regions. The transition probabilities a_{ij}^{-} for the non-island regions are computed in the same way.

	Island ("+")				**Non-Island ('–')**			
	Dinucleotide Frequencies				***Dinuceotide Frequencies***			
	A	***C***	***T***	***G***	***A***	***C***	***T***	***G***
A	0.180	0.274	0.120	0.426	0.300	0.205	0.210	0.285
C	0.171	0.368	0.188	0.274	0.322	0.298	0.302	0.078
T	0.161	0.339	0.125	0.375	0.248	0.246	0.208	0.298
G	0.079	0.355	0.182	0.384	0.177	0.239	0.292	0.292

Table 9.2 Sample nucleotide frequencies from [23]. The entries are computed by averaging the columns of Table 9.1 for the ("+") and for the ("−") regions.

	A	*C*	*T*	*G*
Island ("+") Frequencies:	0.15	0.33	0.16	0.36
Non-Island ("−") Frequencies:	0.27	0.24	0.26	0.23

to have a way of obtaining some estimates for those from the data. This may initially seem to be an impossible task. After all, how can we estimate those frequencies when the boundaries between the regions are unknown? As we will see below, there is a way to solve this problem, if we view the DNA data as generated by a known underlying mechanism, formally described by a HMM.

The rest of the chapter is organized as follows: Section 3 contains a brief review of Markov chains and HHMs, presenting some examples, highlighting some main concepts, and introducing the notation. Readers without prior experience with Markov chains or HMMs may need to consider a more detailed introduction to these topics such as [25,26]. In Section 4 we present CGI identification methods based on HMMs. The section focuses on the questions of evaluation, decoding, and parameter estimation for HMM. Some final comments and notes on generalizations and ongoing work involving HMMs for CGI identification are gathered in Section 5. A suite of web applications, *CpG Educate*, has been developed by the authors to illustrate the algorithms and is used for many of the examples and exercises in the chapter. *CpG Educate* is freely available at http://inspired.jsu.edu/∼agarrett/cpg/.

9.3 DEFINITION AND BASIC PROPERTIES OF MARKOV CHAINS AND HIDDEN MARKOV MODELS

9.3.1 Finite State Markov Chains

A Markov chain is a "process" that at any particular time is in one of a finite number of "states" from a set $Q = \{s_1, \dots, s_n\}$. We will only consider discrete time, which means that the process can be visualized as running according to a digital timer, possibly changing states at each time step. The *Markov property* assumes the state of the process at the next step depends only on its present state and not on the history of how the process arrived at that state. Thus, the process has no memory beyond the "present" and the only factors that govern the process are (i) its current state, and (ii) the probabilities with which the process moves from its current state to other states.

Assume that $\pi_t \in Q$ denotes the state of the process at time t, where $t = 1, 2, 3, \dots$ As time goes on, the process transitions from one state to another generating a *path* $\pi = \pi_1\pi_2 \cdots \pi_l$ (the number l is the *length* of the path). If we have observed the path of the process up to time t, the formal definition of the Markov property is that

$$P\{\pi_{t+1}|\pi_1\pi_2 \cdots \pi_t\} = P\{\pi_{t+1}|\pi_t\}, \quad t = 1, 2, 3, \dots,$$

stating that the probability for the process transitioning to π_{t+1} at time $t+1$ does not depend on the entire "history" of the process $\pi_1\pi_2\cdots\pi_t$ but only on its current state π_t.

For any two states $i, j \in Q$, we consider the transitional probabilitiy $a_{ij} = P\{\pi_{t+1} = j|\pi_t = i\}$, defined as the probability that the process moves from state i to state j when the (discrete) time changes from t to $t+1$. When $i = j, a_{ii}$ is the probability that the process remains in state i at time $t+1$. The transition probabilities are the same for all values of t, meaning that the transitions depend only on the state of the process and not on when the process visits that state. The transition probabilities are commonly organized in a *transition matrix*

$$p = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, \quad a_{ij} \geqslant 0,$$

with $\sum_{j\in Q} a_{ij} = 1,$ for all $i \in Q$.[3] The initial state for the process is determined by the *initial distribution* $p_i = P(\pi_1 = i), i \in Q$, where $\sum_{i\in Q} p_i = 1$.

It is often convenient to introduce notation that would allow for the initial distribution and for the ending of the sequence to be treated as transition probabilities and included in the notation, $a_{ij}, i, j \in Q$. To accomplish this, a hypothetical "beginning" state B and an "ending" state E are introduced with the assumption that the process begins at state B at time $t = 0$ with probability 1 and it transitions to E with probability 1 at the end of each path. The probability for transitioning into B after time $t = 0$ is zero, and the probability of transitioning out of E is zero. With these additions, each path $\pi = \pi_1\pi_2\cdots\pi_l$ of the Markov chain can be expanded to $\pi = B\pi_1\pi_2\cdots\pi_l E = \pi_0\pi_1\pi_2\cdots\pi_l\pi_{l+1}$. We can append a superficial state denoted by 0 to Q, $Q = \{0, s_1, \ldots, s_n\}$ and write $a_{0i} = p_i = P(\pi_1 = i|\pi_0 = \text{B})$ for the initial distribution, for all $i \in Q$, and $a_{i0} = P(\pi_{l+1} = \text{E}\,|\pi_l = i) = 1$, for all $i \in Q$ (at the end of each path $\pi = \pi_0\pi_1\pi_2\cdots\pi_l$, the process moves to E with probability 1). In what follows we will not explicitly append the symbols B and E at the beginning and at the end of all paths but, when it is necessary, the transition probabilities will be interpreted in this generalized sense.

For any observed path $\pi = \pi_0\pi_1\pi_2\cdots\pi_l$, we apply the Markov property to compute its probability as follows:

$$\begin{aligned} P(\pi) &= P(\pi_0\pi_1\pi_2\cdots\pi_l) = P\{\pi_l|\pi_{l-1}, \pi_{L-2}, \ldots, \pi_0\}P(\pi_0\pi_1\cdots\pi_{l-1}) \\ &= P\{\pi_l|\pi_{l-1}\}P(\pi_0\pi_1\pi_2\cdots\pi_{l-1}) = \ldots = \end{aligned}$$

[3]This condition simply reflects the fact the process will be in some state from Q at the next time step, unless it terminates.

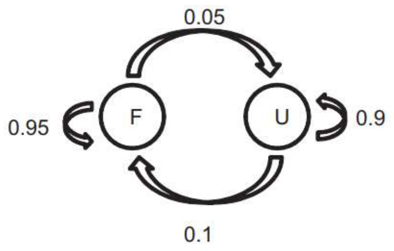


FIGURE 9.5

Directed graph representation of a discrete Markov chain with a state space $Q = \{f, u\}$. The vertices of the digraph correspond to the states of the process and the labels on the directed edges stand for the transition probabilities. The beginning and ending states are not pictured.

$$
\begin{aligned}
&= P\{\pi_l|\pi_{l-1}\}P\{\pi_{l-1}|\pi_{l-2}\}P\{\pi_{l-2}|\pi_{l-3}\}\cdots P\{\pi_1|\pi_0\}\\
&= a_{\pi_{l-1}\pi_l}a_{\pi_{l-2}\pi_{l-1}}\cdots a_{\pi_0\pi_1}\\
&= \prod_{i=1}^{l} a_{\pi_{i-1}\pi_i}.
\end{aligned}
$$

Example 9.1. Assume that a game is played by successively rolling a die. Two dice are available, one fair and one unfair. Either die is equally likely to be chosen for the first game. After that, the process of switching between the dice can be depicted by the transition diagram in Figure 9.5: if the fair die is used in the current game, it will be retained for the next game with probability 0.95 and switched with the unfair die with probability 0.05. If the unfair die is used for the current game, we continue to use it for the next game with probability 0.9 and switch to the fair die with probability 0.1. The process π of switching between the dice can be described by a Markov chain with a state space of two elements $Q = \{F, U\}$, with transition probabilities $a_{FF} = P(\pi_{t+1} = F|\pi_t = F) = 0.95, a_{FU} = P(\pi_{t+1} = F|\pi_t = U) = 0.05$, $a_{UF=P}(\pi_{t+1} = F|\pi_t = U) = 0.10$, and $a_{UU} = P(\pi_{t+1} = U|\pi_t = U) = 0.90$. The transition matrix is then $P = \begin{pmatrix} a_{FF} & a_{FU} \\ a_{UF} & a_{UU} \end{pmatrix} = \begin{pmatrix} 0.95 & 0.05 \\ 0.10 & 0.9 \end{pmatrix}$ and the initial distribution is $a_{0F} = 0.5$ and $a_{0U} = 0.5$.

We can now easily compute the probability of any path of this process. For example, if $\bar{\pi} = UFUF$ and $\bar{\bar{\pi}} = UUFF$, we obtain $P(\bar{\pi}) = a_{0U}a_{UF}a_{FU}a_{UF} = (0.5) * (0.1) * (0.05) * (0.1) = 0.0025$ and $P(\bar{\bar{\pi}} = a_{0U}a_{UU}a_{UF}a_{FF} = (0.5) * (0.9) * (0.1) * (0.95) = 0.0428$.

Exercise 9.1. For the Markov chain from Example 9.1, compute the probabilities of the following paths: (a) $\bar{\pi} = FUUUU$; (b) $\bar{\bar{\pi}} = UUFFUF$; (c) $\tilde{\pi} = FFFFFU$. ▽

Exercise 9.2. If a Markov chain has a nonzero probability for transitioning from a state to itself, it is possible that a path of the process will feature "runs" of the same symbols, indicating that the process remains in that same state for more than one step. For the Markov chain from Figure 9.5, those will be runs of Us and Fs. Generalizing, assume that the transition matrix in Example 9.1 is $P = \begin{pmatrix} a_{FF} & a_{FU} \\ a_{UF} & a_{UU} \end{pmatrix} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$, where $0 < p, q < 1$.

Show that the distribution of the lengths of the runs is as follows:

1. For runs of length of at least l that do not start at beginning of the sequence

$$P(\text{a run of } Fs \text{ of length of at least } l) = (1-q)p^{l-1},$$
$$P(\text{a run of } Us \text{ of length of at least } l) = (1-p)q^{l-1}.$$

2. For runs of length at least l that start at beginning of the sequence

$$P(\text{a run of } Fs \text{ of length of at least } l) = a_{0F}\,p^{l-1},$$
$$P(\text{a run of } Us \text{ of length of at least } l) = a_{0U}\,q^{l-1}.$$

▽

Exercise 9.3. In the context of Exercise 9.2, show the following: (1) When p and q are fixed, the probability for runs of length at least l decreases exponentially as l increases. (2) When is fixed, the probability to observe a run of length at least l increases as the probabilities p and q increase. ▽

9.3.2 Hidden Markov Models (HMMs)

HMMs generalize Markov chains by assuming that the process described by the Markov chain is not readily observable (it is *hidden*). According to some rules, each hidden state generates (emits) a symbol and only the sequence of emitted symbols is observed. The following example, inspired by the Occasionally Dishonest Casino example by Durbin et al. [23], illustrates the idea.

Example 9.2. The outcome of a game is determined by the roll of a die. If the number of points on the die is 1, 2, 3, or 4, the player wins, and if the number of points is 5 or 6, the player loses. The casino uses a fair or an unfair die for each game. For the fair die, all outcomes are equally likely but the unfair die is heavily biased toward 6 with $p(i) = 0.1$ for $i = 1, 2, 3, 4, 5$ and $p(6) = 0.5$. The process of switching between the dice is a Markov chain with state space $Q = \{F, U\}$, transition matrix $P = \begin{pmatrix} 0.95 & 0.05 \\ 0.10 & 0.9 \end{pmatrix}$, and initial distribution $a_{0F} = 0.5$, $a_{0U} = 0.5$, just as in Example 9.1. The player is unaware which die is being used for each game or when a switch between the fair and unfair dice occurs. Thus the paths π of this Markov process are hidden.

The player records wins and losses from consecutive games in a sequence of Ws and Ls (e.g., $x = WWLLLLLWLL$), where W stands for a win and L stands for a loss. This way, for each sequence of games, the player generates a record

$x = x_1x_2x_3 \cdots x_l$ of wins and losses with $x_t \in M = \{W, L\}$, denoting the outcome of game t, $t = 1, 2, \ldots, l$. The chances of winning or losing a game depend on the choice of die used for that game. We can think of each x_t as generated (emitted) by the hidden state π_t with a certain probability. Since a fair die gives the player a win with probability $\frac{2}{3} = 0.67$ and a loss with probability $\frac{1}{3} = 0.33$, we will say that the fair die *emits* a W with probability $e_F(W) = 0.67$ and that it emits an L with probability $e_F(L) = 0.33$. Similarly, the unfair die emits a W with probability $e_U(W) = 0.4$ and an L with probability $e_U(L) = 0.6$. The set $M = \{W, L\}$ is the set of *emitted states* for the hidden process.

The main difference between Markov chains and hidden Markov chains is that the observed sequence x cannot be mapped to a unique path of state-to-state transitions for the hidden process. Multiple hidden paths π can generate x and, as our next example illustrates, they do so with different probabilities.

Example 9.3. In the context of Example 9.2, assume the following sequence of rolls is generated for four consecutive games: $z = 2561$. The recorded (observed) sequence of wins and losses is then $x = WLLW$. Any hidden sequence $\pi = \pi_1\pi_2\pi_3\pi_4$, $\pi_t \in \{F, U\}$, could have generated this sequence but different sequences will do so with very different probabilities. To illustrate, we will compute the probabilities $P(x, \bar{\pi})$ and $P(x, \bar{\bar{\pi}})$ that x is generated by each of the following hidden sequences: $\bar{\pi} = FUUU$ and $\bar{\bar{\pi}} = FUFU$. For $P(x, \bar{\pi})$, we obtain $P(x, \bar{\pi}) = 0.5 \cdot e_F(W) \cdot a_{FU} \cdot e_U(L) \cdot a_{UU} \cdot e_U(L) \cdot a_{UU} \cdot e_U(W) = 0.000012$. The rationale is as follows. For x and $\bar{\pi}$ to occur, a series of events should take place: the fair die is chosen for the first game, emitting a *W*; a switch from the fair to the unfair die follows and the unfair die emits an *L*; the unfair die is retained for the next game, emitting an *L*, and so on. The probability $P(x, \bar{\pi})$ is then the product of the transition and emission probabilities. Similarly, $P(x, \bar{\bar{\pi}}) = 0.5 \cdot e_F(W) \cdot a_{FU} \cdot e_U(L) \cdot a_{UF} \cdot e_F(L) \cdot a_{FU} \cdot e_U(W) = 0.0000067$, which is different from $P(x, \bar{\pi})$. In principle, if we were to guess which hidden path π generated the observed sequence x, the smart way to do so would be to find the path π^* for which the probability $P(x, \pi)$ is the largest.[4] The probability $P(x)$ of emitting the sequence x, is the sum over all $P(x, \pi)$ probabilities for all hidden sequences π: $P(x) = \sum_\pi P(x, \pi)$.

Example 9.4. (The Occasionally Dishonest Casino example from [23]). Assume the hidden process that switches between the fair and unfair die is as described in Example 9.2 but now the player bets on the specific number of points rolled at each game. It would then make sense to record the sequence $z = z_1z_2 \cdots z_l$ where $z_t \in \{1, 2, 3, 4, 5, 6\}$ stands for the number of points rolled in game t, $t = 1, 2, 3, \ldots, l$. The hidden states F and U now emit the symbols from the set $M = \{1, 2, 3, 4, 5, 6\}$ with probabilities $e_F(1) = e_F(2) = e_F(3) = e_F(4) = e_F(5) = e_F(6) = \frac{1}{6} = 0.1667$ and $e_U(1) = e_U(2) = e_U(3) = e_U(4) = e_U(5) = 0.1$ and $e_U(6) = 0.5$. We will use this HMM for several of the examples and exercises in Section 4.

[4]We will show how this can be done in a computationally efficient way in Section 4.

With this background, we can now ask the following questions: Given an observed sequence $x = x_1x_2x_3 \cdots x_l$, how do we determine the hidden sequence $\pi = \pi_1\pi_2\pi_3 \cdots \pi_l$ that maximizes the probability of observing x? Can we estimate the parameters of the HMM (the transition matrix P and the emission probabilities) from the observed sequence x?

The main reason for considering these gambling examples is that the problem of locating CGIs is in many ways mathematically analogous and may be stated similarly. A DNA sequence x composed of the symbols A, C, T, and G may be viewed as generated by a mechanism switching between two hidden states $Q = \{+, -\}$, analogous to the honest and dishonest dice. One state is that of CGIs (the "+" region), the other is the non-island region (the "−" region). The states A, C, T, and G are the same for both regions, but the transition matrices and/or the emission probabilities will be different.[5] We cannot observe the state-to-state transitions $\pi = \pi_1\pi_2\pi_3 \cdots \pi_l$ of the process directly but we observe the sequences $x = x_1x_2x_3 \cdots x_l$ emitted by the process where each x_t is a nucleotide from the set $M = \{A, C, T, G\}$. The questions above for the casino games are now immediately translated into the following questions: Given a long DNA sequence $x = ACTGTC \cdots TCAC$, how do we determine which hidden path is the most likely to have emitted this sequence? Can we estimate the transition and the emission probabilities of the HMM from the observed sequence? To examine these questions in detail, we first need to introduce the general notation for HMMs.

In general, a HMM consists of a hidden Markov chain with a finite state space Q and a finite set of emission symbols M. The state of the hidden process at time is denoted by π_t. The transition probabilities are $a_{ij} = P(\pi_{t+1} = j|\pi_t = i)$, and the initial distribution is $p_i = a_{0i} = P(\pi_1 = i), i \in Q, \sum_{i \in Q} p_i = 1$. The process emits a symbol $k \in M$ from each of the hidden states $j \in Q$ that it visits. The emission probabilities are denoted by $e_j(k) : e_j(k) = P(x_t = k|\pi_t = j), j \in Q, k \in M$, where (since each of the hidden states emits exactly one symbol) $\sum_{k \in M} e_j(k) = 1$. The set of transition, emission, and initial probabilities forms *the set of parameters* of the HMM.

A convenient way to summarize the parameters of a HMM is to organize them in a table where the transition matrix, emission probabilities, and the initial distribution are given in the columns and where the rows are labeled by the hidden states. Table 9.3 contains the parameters of the HMM from Example 9.3.

Exercise 9.4. Give the table for the parameters of the HMM from Example 9.4. ▽

9.3.3 Viewing DNA Sequences As Outputs of a HMM

Tables 9.1 and 9.2 demonstrate two different ways of viewing the quantitative differences between the "+" and the "−" regions in the genome, each one of which can

[5]For example, within the "+" and the "−" regions, the transition probabilities could be similar to those in Table 9.1 (with the modification that there should also be small nonzero probabilities of switching between the "+" and the "−" regions).

Table 9.3 The parameter set for the HMM from Example 9.3 in tabular form.

	Transitions		Emissions		Initial Distribution
	F	*U*	*W*	*L*	
F	0.95	0.05	0.67	0.33	0.5
U	0.1	0.9	0.40	0.60	0.5

Table 9.4 A set of probabilities (parameters) of a HMM for a DNA sequence where the model is only concerned with the frequencies of the individual nucleotides. The transition matrix of the HMM is under the "Transitions" heading. Each of the hidden states emits a symbol from the set $M = \{A,C,T,G\}$ with emission probabilities listed under the "Emissions" heading. The hidden process is equally likely to begin in the "+" and "−" state, as stated under the "Initial Distribution" heading.

	Transitions		Emissions				Initial Distribution
	+	−	*A*	*C*	*T*	*G*	
+	0.90	0.10	0.15	0.33	0.16	0.36	0.5
−	0.05	0.95	0.27	0.24	0.26	0.23	0.5

be used to construct a HMM. When we look only at the nucleotide frequencies as in Table 9.2 we can consider a HMM with a state space $Q = \{+, -\}$, where each of these states can emit a symbol from the set $M = \{A, C, T, G\}$ with emission probabilities as those in Table 9.2. Assuming that hidden process transitions between the "+" and "−" states are as in Figure 9.5 (where in this case, we will identify the state U with "+" and the state F with "−") the parameters for the HMM will be those in Table 9.4.

If we want the model to incorporate information about dinucleotides, as in the case of Table 9.1, the set of emitted symbols is again $M = \{A, C, T, G\}$ but now the emission events at each step are not independent from one another. If, say, the process is in the hidden state "+," the probability for emitting a symbol C will depend upon the symbol emitted by the previous state and whether this symbol was emitted from the "+" or from the "−" hidden state. We can think of it as emitted from one of two hidden states C_+ or C_-. Thus, for each of the emission symbols $k \in M$ we should have states k_+ and k_- in Q, leading to a state space $Q = \{A_+, A_-, C_+, C_-, T_+, T_-, G_+, G_-\}$ for the hidden process. The matrix for the transitions within the subsets of the "+" and "−" states should be close to those in the transition matrices in Table 9.5 but switching between the "+" and "−" subsets $Q_+ = \{A_+, C_+, T_+, G_+\}$ and $Q_- = \{A_-, C_-, T_-, G_-\}$ of Q should also be allowed with some small probability. Table 9.5 presents this scenario.

Exercise 9.5. The HMM from Table 9.4 could be considered to be a special case of the general model from Table 9.5 with state space $Q = \{A_+, A_-, C_+, C_-, T_+, T_-, G_+, G_-\}$. Give a set of HMM parameters for the general HMM from

Table 9.5 The set of parameters of a HMM describing a DNA sequence. The model incorporates information about dinucleotide frequencies. Here the block matrices P^+ and P^- are the "+" and "−" transition matrices from Table 9.1 The process remains in the the "+" and "−" region with probabilities p and q, respectively. When switching between the "+" and "−" regions, this model assumes that all states in the new region are equally likely. For each of the emission symbols $k \in M$, the model assumes that only the states k_+ and k_- from can Q emit k. The initial distribution is uniform.

	Transitions								Emissions				Initial
	A_+	C_+	T_+	G_+	A_-	C_-	T_-	G_-	A	C	T	G	
A_+					$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	1	0	0	0	0.125
C_+					$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	0	1	0	0	0.125
T_+		$(P^+) * p$			$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	0	0	1	0	0.125
G_+					$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	$(1-p)/4$	0	0	0	1	0.125
A_-	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$					1	0	0	0	0.125
C_-	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$					0	1	0	0	0.125
T_-	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$		$(P^-) * q$			0	0	1	0	0.125
G_-	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$	$(1-q)/4$					0	0	0	1	0.125

Table 9.5 to obtain a HMM with transition and emission probabilities such as those in Table 9.4. ▽

The parameters in Table 9.5 present a very special case among all possible HMMs with $Q = \{A_+, A_-, C_+, C_-, T_+, T_-, G_+, G_-\}$ and $M = \{A, C, T, G\}$ and we have included it here since it provides a natural way to include the information from Table 9.1. Ultimately, though, the parameters of the HMM would need to be estimated from the DNA data, and it would be more appropriate to consider the model in its full generality. Thus, any transition matrix P under the Transitions heading, any emission matrix E under the Emissions heading, and any initial distribution could be used as model parameters. In [27], the authors consider such a general model and estimate the parameters from a set of 1000 DNA sequences.

9.4 THREE CANONICAL PROBLEMS FOR HMMs WITH APPLICATIONS TO CGI IDENTIFICATION

We now turn to the mathematical solutions of the questions raised above and describe how they can be solved in a computationally efficient way. After discussing each problem, we provide examples and exercises with connections to CGI identification. We begin with restating the questions in the context of HMMs:

1. *Decoding Problem*: Given an observed path $x = x_1x_2x_3\cdots x_l$ generated by a HMM with known parameters, what is the most likely hidden path $\pi = \pi_1\pi_2\pi_3\cdots\pi_l$ to emit x? In other words, how do we find $\pi_{\max} = \arg\max_\pi P(\pi|x)$?[6] From Example 9.3 we know how to compute $P(x, \pi)$ for any hidden path π and it can be shown (see Exercise 9.6 below) that

$$\pi_{\max} = \arg\max_{\pi} P(\pi|x) = \arg\max_{\pi} P(x, \pi).$$

   Since there are finitely many hidden paths, one may be tempted to answer the question by computing $P(x, \pi)$ for all paths π, compare their values, and pick the largest. Such a "brute force" approach is not practical, however, since the number of all hidden paths grows exponentially with the length of the paths l. The number $|Q|^l$ of all such paths is astronomical for large values of l, making the task impossible even for the fastest computers.[7]
2. *Evaluation Problem*: Given an observed path x, what is the probability of this path $P(x)$? Mathematically, this probability can be expressed as

$$P(x) = \sum_{\pi} P(x, \pi), \tag{9.1}$$

[6]The notation is for the argument of the maximum. We need the path(s) π for which $P(x, \pi)$ is maximal.
[7]For an observed sequence of just 90 nucleotides, the number of paths is a bit over 10^{80}, which is the estimated number of atoms in the observable universe. The brute force approach would therefore require years of computing time even if large computing resources are applied. In principle, we would be interested in sequences of tens of thousands of nucleotides.

where the sum is over all possible hidden sequences $\pi = \pi_1\pi_2\pi_3\cdots\pi_l$ with $\pi_t \in Q$ and where

$$P(x,\pi) = a_{0\pi_1}\prod_{i=1}^{l} e_{\pi_i}(x_i)a_{\pi_i\pi_{i+1}}. \tag{9.2}$$

As with the decoding problem, the mathematical solution from Eq. (9.1) has no practical value since the number of such paths grows exponentially as the length of the path l grows.

3. *Training (Learning) Problem:* Given an observed sequence x or a set of observed sequences, what are the HMM parameters that make the sequence x most likely to occur? The answer to this question provides estimates for the parameters of the HMM from a data set of observed sequences.

9.4.1 Decoding: The Viterbi algorithm

The Viterbi algorithm is a recursive and computationally efficient method for computing the most likely state sequence $\pi_{\max}$ for a given observed sequence $x = x_1x_2x_3\cdots x_l$ (see [28] and also [29] for an interesting account of the history by Andrew Viterbi himself).

To understand the recursive step, let's assume that for any state $j \in Q$ we have somehow computed the hidden sequence $\pi_1\pi_2\pi_3\cdots\pi_{t-2}\pi_{t-1}$ of highest probability among those emitting the observations $x_1x_2x_3\cdots x_{t-1}$ up to time $t-1$ with $\pi_{t-1} = j$. That is, we assume that for each $j \in Q$, we have determined the most probable path of length $t-1$ for the data $x_1x_2x_3\cdots x_{t-1}$, ending in state j. Denote the probability of this path by $v_j(t-1)$:

$$v_j(t-1) = \max_{\pi=\pi_1\pi_2\pi_3\cdots\pi_{t-1}} P(\pi_{t-1}=j, x_{t-1}), \quad j \in Q.$$

Next, we can use $v_j(t-1)$ to compute the most probable path of length t ending in each of the states $k \in Q$ and emitting the sequence $x_1x_2x_3\cdots x_t$:

$$v_k(t) = \max_{\pi=\pi_1\pi_2\pi_3\cdots\pi_t} P(\pi_t = k, x_t).$$

The probability $v_k(t)$ of the most likely path of length t ending in k and emitting x_t will be the largest among the probabilities for hidden sequences that get into j at time $t-1$ with a maximal probability, transition from j to k at time t, and emit x_t. Thus

$$v_k(t) = \max_{j\in Q}\{v_j(t-1)a_{jk}e_k(x_t)\} = e_k(x_t)\max_{j\in Q}\{v_j(t-1)a_{jk}\}.$$

Iterating this argument for all $t = 1, 2, \ldots, l$ will allow us to compute the probability of the most likely path $\pi = \pi_1\pi_2\pi_3\cdots\pi_l$ for the data $x = x_1x_2x_3\cdots x_l$. To be able to recover the path $x = x_1x_2x_3\cdots x_l$ itself, for each time t and for each state $k \in Q$, we keep pointers (ptr) to remember the state $r \in Q$ from which the maximal probability path ending in k came. For each $t = 1, 2, \ldots, l$ and $k \in Q$ we record k's predecessor

$\text{ptr}_t(k) = r$, where $v_r(t-1)a_{rk} = \max_{j \in Q}\{v_j(t-1)a_{jk}\}$. The notation $\text{ptr}_t(k) = r$ means that the path with the highest probability ending in state k at time t came from state r at time $t-1$.

Utilizing the agreement that the process always starts in the beginning state B, Viterbi's algorithms can be summarized as follows. To find the most probable path $\pi = \pi_1\pi_2\pi_3 \cdots \pi_l$ for the observed data $x = x_1x_2x_3 \cdots x_l$, perform the following steps:

- Initialization $t = 0$: $v_B(0) = 1$, $v_j(0) = 0$, $j \in Q$.
- Recursion (repeat for $t = 1, 2, \ldots, l$: $v_k(t) = e_k(x_t) \max_{j \in Q}\{v_j(t-1)a_{jk}\}$; $\text{ptr}_t(k) = r$ where $r = \arg\max_j\{v_j(t-1)a_{jk}\}$.
- Termination: $P(x, \pi^*) = \max_\pi P(x, \pi) = \max_{j \in Q}\{v_j(l)\}$; $\text{ptr}_l(k) = \pi_l^*$.
- The maximal probability path π^* can now be found by backtracking through the recorded pointers.

Our next example illustrates the method.

Example 9.5. Consider the game from Example 9.3 with modified transition probabilities between the hidden states *F* and *U* as in Table 9.6. Assume we have observed the sequence $x = x_1x_2x_3 = LWW$. We will apply the Viterbi algorithm to compute the most likely hidden path $\pi = \pi_1\pi_2\pi_3$ for this observed sequence.

$t = 0$:

We begin with probability 1 at the beginning state and thus $v_B(0) = 1$, $v_F(0) = 0$, $v_U(0) = 0$. As explained earlier, $v_B(0) = 1$ is used just for initialization. It stands for the most probable path ending in B at time $t = 0$. Since all paths of the process start from B, $v_B(0) = 1$. Since no path can be in either F or U at time $t = 0$, $v_F(0) = 0$, $v_U(0) = 0$.

$t = 1$:

$$\begin{aligned} v_F(1) &= \max_{\pi_1} P(\pi_1 = F, x_1 = L) = \max\{v_B(0)a_{0F}e_F(L)\} \\ &= (0.5)(0.33) = 0.1667. \end{aligned}$$

The maximum is taken over all paths that start at the beginning state and end at F. There is only one such path. Thus $v_F(1)$ is the probability that the process is at state F at time 1 (which occurs with probability 0.5) and that it emits the symbol $x_1 = L$ from that state (which happens with probability 0.33). Similarly, $v_U(1) = \max_{\pi_1} P(\pi_1 = U, x_1 = L) = \max\{v_B(0)a_{0U}e_U(L)\} = v_B(0)a_{0U}e_U(L) = (0.5)(0.6) = 0.30$.

Table 9.6 HMM parameters for Example 9.5.

	Transitions		**Emissions**		**Initial Distribution**
	F	*U*	*W*	*L*	
F	0.7	0.3	0.67	0.33	0.5
U	0.4	0.6	0.40	0.60	0.5

$t = 2:$

$$\begin{aligned}
v_F(2) &= \max_{\pi_1\pi_2} P(\pi_2 = F, x_2 = W) \\
&= \max\{v_F(1)a_{FF}e_F(W), v_U(1)a_{UF}e_F(W)\} \\
&= e_F(W) \max\{v_F(1)a_{FF}, v_U(1)a_{UF}\} \\
&= (0.67) \max\{(0.1667) * (0.7), (0.3) * (0.4)\} \\
&= (0.67) \max\{0.1167, 0.12\} = (0.67)*(0.12) = 0.0804.
\end{aligned}$$

Since the max was achieved for $v_U(1)a_{UF}$, the most likely path into F at $t = 2$ came from U and we have $\text{ptr}_2(F) = U$. Similarly,

$$\begin{aligned}
v_U(2) &= \max_{\pi_1\pi_2} P(\pi_2 = U, x_2 = W) \\
&= \max\{v_F(1)a_{FU}e_U(W), v_U(1)a_{UU}e_U(W)\} \\
&= e_U(W) \max\{v_F(1)a_{FU}, v_U(1)a_{UU}\} \\
&= (0.4) \max\{(0.1667)*(0.3), (0.3)*(0.6)\} \\
&= (0.4) \max\{0.05, 0.18\} = (0.4)*(0.18) = 0.072.
\end{aligned}$$

Since the max was achieved for $v_U(1)a_{UU}$, the most likely path into U at $t = 2$ came from U and we have $\text{ptr}_2(U) = U$.

$t = 3:$

$$\begin{aligned}
v_F(3) &= \max_{\pi_1\pi_2\pi_3} P(\pi_3 = F, x_3 = W) \\
&= \max\{v_F(2)a_{FF}e_F(W), v_U(2)a_{UF}e_F(W)\} \\
&= e_F(W) \max\{v_F(2)a_{FF}, v_U(2)a_{UF}\} \\
&= (0.67) \max\{(0.0804) * (0.7), (0.072)*(0.4)\} \\
&= (0.67) \max\{0.0563, 0.0296\} = (0.67) * (0.0563) = 0.038.
\end{aligned}$$

Since the max was achieved at $v_F(2)a_{FF}$, the most likely path into F at $t = 3$ came from F and we have $\text{ptr}_3(F) = F$. Finally,

$$\begin{aligned}
v_U(3) &= \max\nolimits_{\pi_1\pi_2\pi_3} P(\pi_3 = U, x_3 = W) \\
&= \max\{v_F(2)a_{FU}e_U(W), v_U(2)a_{UU}e_U(W)\} \\
&= e_U(W) \max\{v_F(2)a_{FU}, v_U(2)a_{UU}\} \\
&= (0.4) \max\{(0.0804) * (0.3), (0.072) * (0.6)\} \\
&= (0.67) \max\{0.0241, 0.0432\} \\
&= (0.4) * (0.0432) = 0.017.
\end{aligned}$$

The max was achieved for $v_U(2)a_{UU}$, thus the most likely path into U at $t = 3$ came from U and we have $\text{ptr}_3(U) = U$.

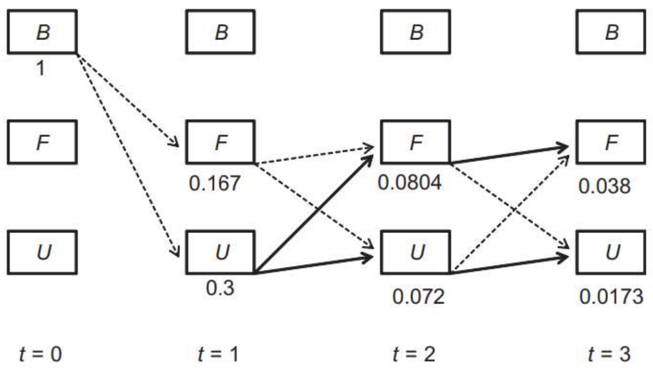


FIGURE 9.6

Trellis diagram for Example 9.5. *B* is the beginning state (the ending state is not pictured). The numbers under each node indicate the maximal probability for all paths ending in this node and emitting sequences in agreement with the data. The dashed arrows indicate possible transitions. For each node, the solid arrow into the node is the pointer indicating where the maximal probability path comes from.

We have reached the end of the sequence $x = x_1x_2x_3 = LWW$ and the algorithm terminates. The probability of the most likely path is therefore

$$\begin{aligned} P(x, \pi^*) &= \max_{\pi} P(x, \pi) = \max_{j \in Q}\{v_j(l)\} = \max\{v_F(3), v_U(3)\} \\ &= \max\{0.038, 0.017\} = 0.038. \end{aligned}$$

The maximum was achieved at $v_F(3)$, thus the ending state for the most likely path is $\pi_3^* = F$. Since $\text{ptr}_3(F) = F$, the most likely path into F at time $t = 3$ came from F, and thus $\pi_2^* = F$. Since $\text{ptr}_2(F) = U$, the most likely path into F at time $t = 2$ came from U, and thus $\pi_1^* = U$. Therefore, the hidden sequence of maximal probability 0.038 is $\pi^* = \pi_1^*\pi_2^*\pi_3^* = UFF$.

It is convenient to visualize the Viterbi algorithm by using a *trellis diagram* that makes it easier to follow the process. In a trellis diagram, the columns are labeled with the values of t and the rows correspond to the states of the Markov chain (see Figure 9.6). Starting from the state B and following the arrows can generate all possible paths $\pi = \pi_1\pi_2\pi_3$. The number under each node is the maximal probability for the path ending in the node and agreeing with the observed data up to that time (these are the v-values computed by the Viterbi algorithm). The solid arrow into each node is the pointer that keeps track of the origin of the maximal probability path. Locating the largest probability in the last column and backtracking using the solid lines, generates the maximal probability path $\pi^* = \pi_1^*\pi_2^*\pi_3^*$.

It can be shown that the computational complexity of the Viterbi algorithm is no worse than $O(n^2 l)$ in time[8] and $O(nl)$ in space, where $n = |Q|$ is the number of states and l is the length of the sequence. Thus, the complexity of the algorithm is quadratic in time, which, although still computationally demanding, is a huge improvement over the exponential rate of the brute force approach.

Since the Viterbi algorithm involves multiplying many probabilities, the results could get very small, resulting in essentially zero probabilities and causing underflow problems. To avoid this, all calculations are performed on the logarithms of the probabilities $v_k(t), k \in Q, t = 1, 2, \ldots$, using addition instead of multiplication.

Exercise 9.6. Show that the hidden path found by the Viterbi algorithm is exactly the path of maximal probability, given the observed sequence x. That is, show that

$$\pi_{max} = \underset{\pi}{\operatorname{argmax}}\, P(x, \pi) = \underset{\pi}{\operatorname{argmax}}\, P(\pi | x). \qquad \triangledown$$

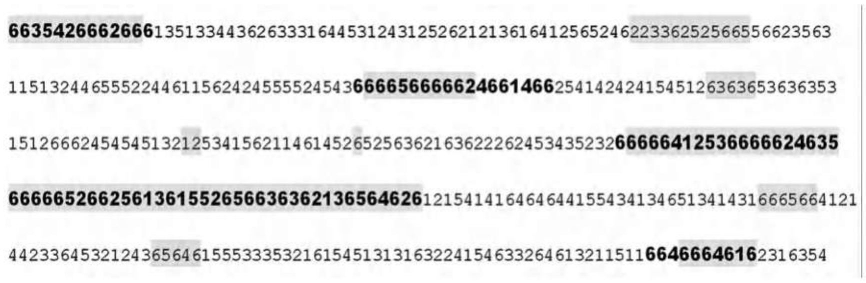
Obviously, Example 9.5 shows that performing the Viterbi computations by hand is tedious even for very short paths. From now on, we will use the *CpG Educate* suite that implements the Viterbi algorithm for HMMs corresponding to the Dishonest Casino problem from Example 9.4 and for the CGI identification problem.

The applications in the *CpG Educate* suite have the capability to generate an output from a HMM and record the hidden path that produced the emitted sequence. This *simulated data* can then be used to test the "accuracy" of the Viterbi algorithm by comparing the predicted states from the Viterbi decoding with the actual ones from the simulations.

Example 9.6. We used the *Dishonest Casino* application in the *CpG Educate* suite to generate a sequence of length 400 for the Dishonest Casino HMM from Example 9.4. We then applied the Viterbi algorithm to the emitted sequence and compared the output with the known hidden sequence from the simulations. Figure 9.7 depicts the outcome. The gray background identifies rolls made with the unfair die. The outcomes in bold are the predicted unfair die state from the path of maximal probability found by the Viterbi algorithm. Although the overlap is not perfect, Viterbi appears to have recovered the states fairly well. In practice, the Viterbi algorithm will often miss short sequences generated by either die. Since switching between the dice happens with small probabilities, the path of maximal probability would generally tend to "bridge" short gaps and preserve the trends set by longer runs.

When using a HMM for identifying CGIs, the outcome of the Viterbi algorithm will produce the predicted path through the hidden "+" and "−" states. The start of the island will be where the predicted hidden sequence switches from the subset $Q_- = \{A_-, C_-, T_-, G_-\}$ to the subset $Q_+ = \{A_+, C_+, T_+, G_+\}$ and the end of the island will be where the sequence switches back from the "+" to the "−" states.

[8]The "big O" notation here means that for very large values of n and l the number of operations required by the Viterbi algorithm has the same order of magnitude as the number $n^2 l$.



663542666266 61351334436263331644531243125262121361641256524622336252566556623563

11513244655522446115624245555245436**66656666624661466**2541424241545126363653636353

151266624545451321253415621146145265256362163622262453435232**66666412536666624635**

66666526625613615526566363621365646 261215414164646441554341346513414316665664121

442336453212436564615553335321615451313163224154633264613211511**664666461 6**2316354

FIGURE 9.7

Viterbi decoding on simulated data of 400 rolls for the Dishonest Casino Example 9.4. The gray background highlights the rolls generated with the unfair die in the simulation. Digits in bold indicate the unfair die states predicted by the Viterbi algorithm.

Exercise 9.7.*[9] Use the *CpG Islands* application in the *CpG Educate* suite to generate simulated sequences of length of 600 bp with HMM parameters from Table 9.5 with $p = 0.9$ and $q = 0.95$. (The file *Exercise_9.7.csv* containing the parameters from Table 9.5 for these values of p and q is provided for you to load into the *CpG Islands* application). Run the Viterbi algorithm on the simulated data and examine how well the decoded sequence matches the states of the HMM from the simulations. Generate several sequences and comment on the outcome. Notice specifically how the maximal probability path generated by the Viterbi algorithm tends to ignore small gaps between the island and non-island regions, as already observed at the end of Example 9.6. ▽

Exercise 9.8. Experiment with applying the Viterbi algorithm to sequences generated by the *Dishonest Casino* application in the *CpG Educate* suite. Consider relatively long sequences of (e.g., 5000 or longer) for different values of the transition probabilities of the HMM (the four probabilities under the Transitions heading). Specifically, consider the following types of transition probabilities: a) transition probabilities are close to uniform (that is, the elements of transition matrix $P = \begin{pmatrix} a_{FF} & a_{FU} \\ a_{UF} & a_{UU} \end{pmatrix}$ are close to 0.5), and b) distributions for which the process retains its current state with a large probability (that is, transition matrices $P = \begin{pmatrix} a_{FF} & a_{FU} \\ a_{UF} & a_{UU} \end{pmatrix} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$ for which p and q are very close to 1). Consider values as large as 0.9999 for p and q.

For each generated sequence run the Viterbi algorithm and note the general quality of its performance: Is the decoded sequence generally close to the hidden sequence generating the simulated data? If there are misses, are those false negatives (switches to the unfair die in the simulation that have remained undetected by the Viterbi algorithm) or false positives (predicted switches to the unfair die in places

[9]Exercises marked with an asterisk indicate that their execution requires downloads from the volume's website.

where the simulations have been generated by using the fair die). Are there any emerging patterns? Be mindful that due to the stochastic nature of the process, you should generate several sequences for each set of parameters. Try to summarize your observations, by answering the following questions:

1. Did you detect any difference in the performance of the Viterbi algorithm for the sets of parameters from part (a) and (b)?
2. If you answered yes in (1), is the performance of the algorithm to decode the hidden sequence better for distributions like those in part a) or for distributions like those in part (b)?
3. Does the performance of the Viterbi algorithm improve when the values of and are very close to 1?
4. What explanations can you provide for your findings for questions (1)? It may be helpful to revisit Exercises 9.1 and 9.2. ▽

Exercise 9.9.* Repeat Exercise 9.8, this time experimenting with the *CpG Islands* application in the *CpG Educate* suite. Consider distributions such as in Table 9.5 and a) values for p and q close to 0.5; b) values for p and q close to 1. Experiment with both short and long sequences and notice that when p or q are very close to 1, the hidden process and/or the maximal probability path may never switch between the "+" and "−" states for relatively short sequence. Use the file *Table_9.5.xlsx* available from the volume's website to generate sets of HMM parameters for different values of p and q. Be sure to save the generated sets of HMM parameters in Comma Separated Values (CSV) format and use the CSV fileto load the parameters into the *CpG Islands* application (refer to the *CpG Educate* tutorial [30] for more details). ▽

9.4.2 Evaluation and Posterior Decoding: The Forward-Backward Algorithm

In this section we outline a computationally efficient algorithm for computing $P(x)$, the probability for an observed sequence $x = x_1x_2\cdots x_l$. We also discuss an alternative decoding method called *posterior decoding.*

The forward algorithm is similar in idea to the Viterbi algorithm. Instead of keeping track of the hidden sequences of maximal probability in state j at time t, the forward algorithm computes the probabilities of being in state j, having observed the first t symbols $x_1x_2\cdots x_t$ of the sequence x. Denote these probabilities by $f_j(t) = P(x_1x_2\cdots x_t, \pi_t = j), j \in Q$. Recursively, as in the Viterbi algorithm, we can now compute

$$f_k(t+1) = P(x_1x_2\cdots x_{t+1}, \pi_{t+1} = k),$$

the probability that the process is in state k at time $t+1$ with emitted sequence $x_1x_2\cdots x_{t+1}$:

$$f_k(t+1) = P(x_1x_2\cdots x_{t+1}, \pi_{t+1} = k) = \sum_{j\in Q} f_j(t)a_{jk}e_k(x_{t+1}), \quad k \in Q.$$

The rationale is straightforward. For the process to be in state k at time $t+1$ with emitted sequence $x_1 x_2 \cdots x_{t+1}$, it must be in one of the states $j \in Q$ at time t with emitted sequence $x_1 x_2 \cdots x_t$ (with probability $f_j(t)$), then transition to state k at time $t+1$ (with probability a_{jk}) and emit x_{t+1} (with probability $e_k(x_{t+1})$). The probability $f_k(t+1)$ is then the sum of the product of those probabilities over all states j.

At time $t = 0$, the process is in the beginning state with probability 1 and has emitted no output sequence yet. Thus, we initialize the algorithm by $f_B(0) = 1$, $f_k(0) = 0$, $k \in Q$.

The complete algorithm is:

- Initialization ($t = 0$): $f_B(0) = 1$, $f_j(0) = 0$, $j \in Q$.
- Recursion (repeat for $t = 1, 2, \ldots, l$): $f_k(t) = e_k(x_t) \sum_{j \in Q} f_j(t-1) a_{jk}$, $k \in Q$.
- Termination: $P(x) = \sum_{k \in Q} f_k(l)$.

Example 9.7. Consider again the HMM from Example 9.5. Use the forward algorithm to compute the probability of the sequence $x = LWW$.

Solution:

$t = 0:$

We begin with probability 1 at the beginning state B and, thus, $f_B(0) = 1$, $f_F(0) = 0$, $f_U(0) = 0$.

$t = 1:$

$$f_F(1) = P(x_1 = L, \pi_1 = F) = f_B(0) a_{0F} e_F(L) = (0.5)(0.33) = 0.165.$$
$$f_U(1) = P(x_1 = L, \pi_1 = U) = f_B(0) a_{0U} e_U(L) = (0.5)(0.6) = 0.30.$$

$t = 2:$

$$\begin{aligned} f_F(2) &= P(x_1 = L, x_2 = W, \pi_2 = F) = e_F(W)(f_F(1)a_{FF} + f_U(1)a_{UF}) \\ &= (0.67)((0.165) * (0.7) + (0.3) * (0.4)) = 0.1578. \\ f_U(2) &= P(x_1 = L, x_2 = W, \pi_2 = U) = e_U(W)(f_F(1)a_{FU} + f_U(1)a_{UU}) \\ &= (0.4)((0.1667) * (0.3) + (0.3) * (0.6)) = 0.0918. \end{aligned}$$

$t = 3:$

$$\begin{aligned} f_F(3) &= P(x_1 = L, x_2 = W, x_3 = W, \pi_3 = F) \\ &= e_F(W)(f_F(2)a_{FF} + f_U(2)a_{UF}) \\ &= (0.67)((0.1578) * (0.7) + (0.092) * (0.4)) = 0.0986. \\ f_U(3) &= P(x_1 = L, x_2 = W, x_3 = W, \pi_3 = U) \\ &= e_U(W)(f_F(2)a_{FU} + f_U(2)a_{UU}) \\ &= (0.4)((0.1578) * (0.3) + (0.0918) * (0.6)) = 0.041. \end{aligned}$$

Finally, we have $P(x) = P(LWW) = 0.0986 + 0.041 = 0.1396$.

For our main problem of CpG identification, the probability $P(x)$ of a DNA sequence of nucleotides $x = x_1x_2\cdots x_l$ is not of much interest by itself but we will need it to compute the *posterior probabilities* $P(\pi_t = k|x), k \in Q, t = 1, 2, \ldots, l$, that symbol x_t in the observed sequence was emitted from state k, which can then be used for decoding.

Since

$$P(\pi_t = k|x) = \frac{P(x, \pi_t = k)}{P(x)}, \tag{9.3}$$

we need a recursive algorithm for computing $P(x, \pi_t = k)$. The Markov property of the hidden process makes this possible:

$$\begin{aligned} P(x, \pi_t = k) &= P(x_1x_2\cdots x_t, \pi_t = k)P(x_{t+1}x_{t+2}\cdots x_l|x_1x_2\cdots x_t, \pi_t = k) \\ &= P(x_1x_2\cdots x_t, \pi_t = k)P(x_{t+1}x_{t+2}\cdots x_l|\pi_t = k). \end{aligned} \tag{9.4}$$

The first line is a direct application of the conditional probability formula where the probability that x is generated with $\pi_t = k$ at time t is given as the product of the probabilities of the following events: (1) symbols $x_1x_2\cdots x_t$ are emitted up to time t and the process is in state k at time t, and (2) conditioned upon the event (1), the rest of the emitted sequence is $x_{t+1}x_{t+2}\cdots x_l$. The second line follows from the Markov property of the hidden process and restates that the probability to emit the sequence $x_{t+1}x_{t+2}\cdots x_l$ depends only on the state of the process at time t.

Notice that the $P(x_1x_2\cdots x_t, \pi_t = k)$ are exactly the probabilities $f_k(t)$, which are computed from the forward algorithm. Denote $b_k(t) = P(x_{t+1}x_{t+2}\cdots x_l|\pi_t = k)$. Equation (9.4) can now be re-written as

$$P(x, \pi_t = k) = f_k(t)b_k(t). \tag{9.5}$$

The probabilities $b_k(t)$ are computed by the *backward algorithm*. We begin by initializing the algorithm for $t = l$ where, since x_l is the last observed symbol, $\pi_{l+1} = E$ is the end state (that does not emit a symbol). Thus $b_k(l) = P(\pi_{l+1} = E|\pi_l = k) = 1, k \in Q$, since the sequence goes to the end state E with probability 1.

Once we know $b_k(l)$ for all $k \in Q$, for any of the values $t = l-1, l-2, \ldots, 1$ we can compute

$$\begin{aligned} b_j(t) &= P(x_{t+1}x_{t+2}\cdots x_l|\pi_t = j) \\ &= \sum_{k\in Q} P(\pi_{t+1} = k|\pi_t = j)e_k(x_{t+1})P(x_{t+2}\cdots x_l|\pi_{t+1} = k) \\ &= \sum_{k\in Q} a_{jk}e_k(x_{t+1})b_k(t+1). \end{aligned}$$

The justification is as follows: At time $t+1$ the process can transition from j to any other state $k \in Q$ (this happens with probability a_{jk}), emit x_{t+1} (this happens with probability $e_k(x_{t+1})$), and, being in state k at time $t+1$, emit the rest of the sequence $x_{t+2}\cdots x_l$ (which happens with probability $b_k(t+1)$).

Since at time $t = 0$ the process is in the beginning state B with probability 1, $b_0(0) = P(x_1x_2 \cdots x_l|\pi_0 = B) = P(x_1x_2 \cdots x_l) = P(x)$. This shows that the probability $P(x)$ can also be computed from the backward algorithm as $b_0(0) = P(x) = \sum_{k\in Q} a_{0k}e_k(x_1)b_k(1)$. If this probability is already computed from the forward algorithm, the backward algorithm will terminate once $b_k(1)$ are computed for all $k \in Q$.

The complete algorithm is:

- Initialization ($t = l$): $b_k(l) = 1, k \in Q$.
- Recursion (repeat for $t = l-1, l-2, \dots, 1$): $b_j(t) = \sum_{k\in Q} a_{jk}e_k(x_{t+1})b_k(t+1)$.
- Termination: $P(x) = \sum_{k\in Q} a_{0k}e_k(x_1)b_k(1)$.

Combining now Eqs. (9.3) and (9.5), we obtain the following equation for the posterior probabilities

$$P(\pi_t = k|x) = \frac{P(x, \pi_t = k)}{P(x)} = \frac{f_k(t)b_k(t)}{P(x)}, \quad k \in Q, \quad t = 1, 2, \dots, l. \qquad (9.6)$$

Since we use both the forward and the backward algorithms, we say that the posterior probabilities are computed by the *forward-backward* algorithm.

Example 9.8. Consider again the HMM from Example 9.5. Use the forward-backward algorithm to compute the posterior probabilities of the sequence $x = LWW$.

Solution: $P(x)$ and the probabilities $f_k(t)$ were computed in Example 9.7. We now compute $b_k(t)$ for $t = 3, 2, 1$.

$$\begin{aligned}
t = 3: \quad & b_F(3) = 1; b_U(3) = 1.\\
t = 2: \quad & b_F(2) = a_{FF}e_F(W)b_F(3) + a_{FU}e_U(W)b_U(3)\\
& \qquad\quad = (0.7) * (0.67) + (0.3) * (0.4) = 0.589,\\
& b_U(2) = a_{UF}e_F(W)b_F(3) + a_{UU}e_U(W)b_U(3)\\
& \qquad\quad = (0.4) * (0.67) + (0.6) * (0.4) = 0.508.\\
t = 1: \quad & b_F(1) = a_{FF}e_F(W)b_F(2) + a_{FU}e_U(W)b_U(2) = 0.3372,\\
& b_U(1) = a_{UF}e_F(W)b_F(2) + a_{UU}e_U(W)b_U(2) = 0.2798.
\end{aligned}$$

The posterior probabilities are now:

$$P(\pi_1 = F|x) = \frac{f_F(1)b_F(1)}{P(x)} = \frac{(0.165) * (0.3372)}{0.1396} = 0.3986,$$
$$P(\pi_1 = U|x) = \frac{f_U(1)b_U(1)}{P(x)} = \frac{(0.3) * (0.2798)}{0.1396} = 0.6014.$$

The process continues similarly until all posterior probabilities are determined (see Exercise 9.10).

Exercise 9.10. Complete the previous example to calculate $P(\pi_2 = F|x)$, $P(\pi_2 = U|x)$, $P(\pi_3 = F|x)$, and $P(\pi_3 = U|x)$. ▽

Exercise 9.11. Continue the application of the forward algorithm from Example 9.8 to carry out the termination step and compute $P(x)$. Verify that this termination step produces the same value for $P(x)$ as the value computed by the forward algorithm in Example 9.7. ▽

The posterior probabilities computed here can be used to complement the results from the Viterbi decoding or as a possible alternative decoding method referred to as *posterior decoding*. They could prove useful when there are multiple sequences with probabilities close to the maximal probability sequence(s) generated by the process of Viterbi decoding, in which case it may not be justified to only consider the sequence(s) of maximal probability. The posterior probabilities give the likelihood (based on the entire observed sequence $x = x_1x_2 \cdots x_l$) that the symbol x_t in position t has been emitted by the hidden state k. Posterior decoding can be quite useful for decoding a hidden process with two states (or two groups of states), which is exactly the case we are concerned with in this chapter. To see this, in the case of the Dishonest Casino example, given the sequence x, we plot the probabilities $P(\pi_t = U|x)$ for each $t = 0, 1, \ldots, l$. The "hills" in the resulting graph would indicate segments in the sequence x that are likely to be emitted from the U state. The remaining segments are more likely to have been emitted by the F state. Analytically, the decoded sequence will be

$$\widetilde{\pi}_t = \underset{k}{\operatorname{argmax}}\, P(\pi_t = k|x). \tag{9.7}$$

Figure 9.8 exemplifies this approach for a simulated sequence of 500 symbols. The gray areas highlight the time steps at which the unfair die was used for the simulations.

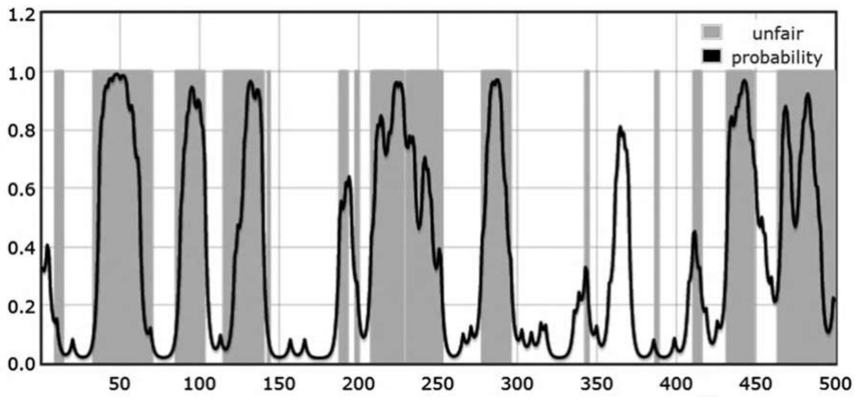


FIGURE 9.8

A plot of the posterior probabilities of being in a state generated by the unfair die for a sequence of 500 simulated runs. The gray highlight identifies the runs obtained from the simulation by using the unfair die. The transition probabilities of the HMM are $a_{UU} = 0.95$; $a_{UF} = 0.05$; $a_{FU} = 0.04$, and $a_{UU} = 0.96$. The default values for the emission probabilities (chosen to match those from Example 9.4) were used.

The condition from Eq. (9.7) in the case of only two hidden states means that if we consider a threshold of 0.5 for Figure 9.8, the hidden states with posterior probabilities plotted above the horizontal line at 0.5 will be predicted as U states. The overlap is not perfect, of course, but, as our next exercise shows, the separation between the states can be even better for hidden processes that only switch between states with very small probability.

Exercise 9.12. Use the *Dishonest Casino* application in the *CpG Educate* suite to experiment with the evaluation algorithms to plot the posterior probabilities of being in a state generated by the unfair die for sequences x of various lengths.[10] Do the same for several sets of transition probabilities, focusing specifically on the two extremes, as in Exercise 9.8: (a) transition probabilities that are close to uniform, and (b) distributions for which the process retains its current state with a large probability. Consider values as large as 0.999 for p and q in this case. Summarize your observations, by answering the following questions: (1) Did you detect any improvement when the sets of parameters are less like those from part (a) and more like those in part (b)? If so, in what sense?; (2) Consider several sets of transition probabilities to illustrate that when the probabilities for switching between the two hidden states get smaller, the performance of posterior decoding improves. ▽

Exercise 9.13. Repeat Exercise 9.12 but, this time, try different values for the emission probabilities. Experiment with posterior decoding to get a sense that its performance improves as the emission distributions for the different states become "more different." If, for instance, both emission distributions are nearly or exactly uniform (e.g., $\{0.1167, 0.1167, 0.1167, 0.1167, 0.1167, 0.1167\}$ and $\{0.15, 0.15, 0.2, 0.15, 0.15, 0.2\}$) the decoding into U and F states will be generally poor. As the emissions distribution for the unfair die becomes more skewed, the performance of the posterior decoding method improves. ▽

Exercise 9.14.* Use the *CpG Islands* application in the *CpG Educate* suite to simulate sequences of various lengths and compare the performance of Viterbi decoding vs. Posterior Decoding. Use HMM parameters in the form of Table 9.5 first, then experiment with general sets of HMM parameters. Use the file *Table_9.5.xlsx* from the volume's website to generate sets of HMM parameters in the format of Table 9.5 for different values of p and q. (Do not forget that the file needs to be saved in CSV format before loading the parameters into the *CpG Islands* application. See Exercise 9.9 and [30] for more details.) ▽

Exercise 9.15. Repeat Exercise 9.14 for the *Dishonest Casino* application. Compared to the HMM from Exercise 9.14, the *Dishonest Casino* application has fewer parameters, so you will not need to upload them from a file - just change the parameter values by typing over the default values that are provided. ▽

[10] For optimal viewing of the posterior probabilities, use sequences with lengths between 300 and 1200.

9.4.3 Training: The Baum-Welch Algorithm

For all computations until now, we always assumed the parameters of the HMM are known. Those parameters, however, are usually only initial estimates and, in most cases, we may not even have those estimates to begin with. In this section we will discuss how to estimate the HMM parameters from the data. This process is called *training* (or sometimes *learning*). The algorithm presented here was first introduced in [31] (see also [32]). We begin with an observed sequence x or a set of observed sequences $x^1, x^2, \ldots, x^m$ for which we would want to adjust the HMM model parameters to ensure the best possible fit. Once a set of parameters is determined for those sequences, we apply the Viterbi algorithm or use the posterior probabilities to other similar sequences for decoding. The sequences $x^1, x^2, \ldots, x^m$ are called *training sequences* and we say that we will train the model to best fit those sequences.

More formally, this means that given the data $x^1, x^2, \ldots, x^m$ and the HMM, we need to find the values of the model parameters (the transition probabilities a_{jk} and the emission probabilities $e_k(b)$ for $j, k \in Q$ and $b \in M$) that maximize the probability of the data $P(x)$. If we use θ to refer to the whole set of parameters for the model, in this section we will sometimes write $P(x) = P(x|\theta) = P(x|\theta, HMM)$ to emphasize that the likelihood of the data depends on the set of parameters θ and on the model. The goal is to find

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\{P(x|\theta)\}.$$

If the training sequences are annotated and we know the exact paths of the process emitting the training sequences, the estimates for the model parameters will be computed by determining the frequencies of each transition and emission. Assume that A_{jk} is the number of transitions from j to k in the set of training sequences and $E_k(b)$ is the number of times the state $k \in Q$ emitted the symbol $b \in M$. Then the frequencies

$$a_{jk} = \frac{A_{jk}}{\sum_{r\in Q} A_{jr}} \text{ and } e_k(b) = \frac{E_k(b)}{\sum_{c\in M} E_k(c)}, \quad \text{for } j, k \in Q, \quad b \in M \tag{9.8}$$

are maximum likelihood estimates for the HMM [23].

When the paths that generate the training data are unknown, we can no longer determine the counts A_{jk} and $E_k(b)$ but they can be replaced with the expected counts for each transition/emission in the HMM. Obtaining the maximum likelihood estimates in this case can be done by an iterative process known as the Baum-Welch algorithm [31]. This method is no longer guaranteed to find a global maximum for the probability of the data but the iterative steps will converge to a local maximum. The Baum-Welch algorithm is a special case of a more general class of methods known as *Expectation Maximization* algorithms or *EM* algorithms. Below we will outline the computational aspect of the Baum-Welch method but will omit the theoretical proofs of convergence to a maximum. Those proofs together with the general theory of the EM algorithms can be found in [33]. In the description of the Baum-Welch algorithm below we will assume that the HMM is being trained on a single data sequence x. We will comment on the straightforward generalization to multiple sequences $x^1, x^2, \ldots, x^m$ afterwards.

The Baum-Welch algorithm generates a sequence of approximations $\theta_0, \theta_1, \theta_2, \ldots$ for the set of HMM parameters, with each new set of parameters θ improving the value of $P(x|\theta)$ over the previous iteration. The process terminates when two successive iterations produce the same values for $P(x|\theta)$ or values that are closer than any previously chosen tolerance value.

To initialize the Baum-Welch algorithm, choose any set of model parameters, incorporating any prior information that may be available. Absent such information, a uniform or any other arbitrary distribution may be chosen. Denote the set of those initial parameters by θ_0. To obtain improved estimates, we still want to use Eq. (9.8) but this time the counts A_{jk} and $E_k(b)$ will be replaced with the expected counts computed from the data. To compute those expected counts, we will first compute $P(\pi_t = j, \pi_{t+1} = k|x, \theta_0)$—the probability that the hidden process will transition from state j to state k at time t.

The inclusion of the parameter set θ_0 in the notation indicates that these probabilities will be computed based on the initial set of parameters θ_0. We will omit this from the notation from now on for simplicity but all of the expressions below assume that we use this parameter distribution for the computations. Using conditional probabilities, we obtain

$$\begin{aligned} &P(\pi_t = j, \pi_{t+1} = k|x) \\ &\quad = \frac{P(x_1, x_2, \ldots, x_t, \pi_t = j) P(\pi_{t+1} = k, x_{t+1}, x_{t+2}, \ldots, x_l | x_1, x_2, \ldots, x_t, \pi_t = j)}{P(x)} \\ &\quad = \frac{f_j(t) P(\pi_{t+1} = k, x_{t+1}, x_{t+2}, \ldots, x_l | \pi_t = j)}{P(x)}, \end{aligned} \tag{9.9}$$

the last equation following from the Markov property.[11] We have also used the notation $f_j(t) = P(x_1, x_2, \ldots, x_t, \pi_t = j)$, introduced earlier for the forward algorithm.

The probability $P(\pi_{t+1} = k, x_{t+1}, x_{t+2}, \ldots, x_l|\pi_t = j)$ can further be expressed as

$$\begin{aligned} &P(\pi_{t+1} = k, x_{t+1}, x_{t+2}, \ldots, x_l|\pi_t = j) \\ &\quad = P(\pi_{t+1} = k|\pi_t = j) P(x_{t+1}|\pi_{t+1} = k) P(x_{t+2}, \ldots, x_l|\pi_{t+1} = k) \\ &\quad = a_{jk} e_k(x_{t+1}) b_k(t+1), \end{aligned}$$

where, as in the backward algorithm, we use $b_k(t+1) = P(x_{t+2}, \ldots, x_l|\pi_{t+1} = k)$. The justification is as follows: Once the process is in state j at time t, for the event $\pi_{t+1} = k, x_{t+1}, x_{t+2}, \ldots, x_l$ to take place, the process needs to transition into k, emit the symbol x_{t+1}, and generate the rest of the sequence $x_{t+1}, \ldots, x_l$. Combining this with Eq. (9.9), we obtain

$$P(\pi_t = j, \pi_{t+1} = k|x) = \frac{f_j(t) a_{jk} e_k(x_{t+1}) b_k(t+1)}{P(x)}. \tag{9.10}$$

[11]The likelihood of the data, given the set of model parameters θ, $P(x) = P(x|\theta)$, is computed by the forward algorithm.

The average number of transitions from j to k in the training sequence x will then be the sum of the probabilities of this transition occurring exactly at position t, over all positions in the sequence x:

$$\begin{aligned} A_{jk} &= \sum_{t=1}^{l} \frac{f_j(t)a_{jk}e_k(x_{t+1})b_k(t+1)}{P(x)} \\ &= \frac{1}{P(x)} \sum_{t=1}^{l} f_j(t)a_{jk}e_k(x_{t+1})b_k(t+1). \end{aligned} \tag{9.11}$$

In a similar way, since the posterior probabilities can be computed from the forward-backward algorithm as $P(\pi_t = k|x, \theta_0) = \frac{f_k(t)b_k(t)}{P(x)}$ (see Eq. (9.6)), the average number of states emitting the symbol $b \in M$ will be

$$E_k(b) = \sum_{\{t=1,\ldots,l|x_t=b\}} \frac{f_k(t)b_k(t)}{P(x)} = \frac{1}{P(x)} \sum_{\{t=1,\ldots,l|x_t=b\}} f_k(t)b_k(t). \tag{9.12}$$

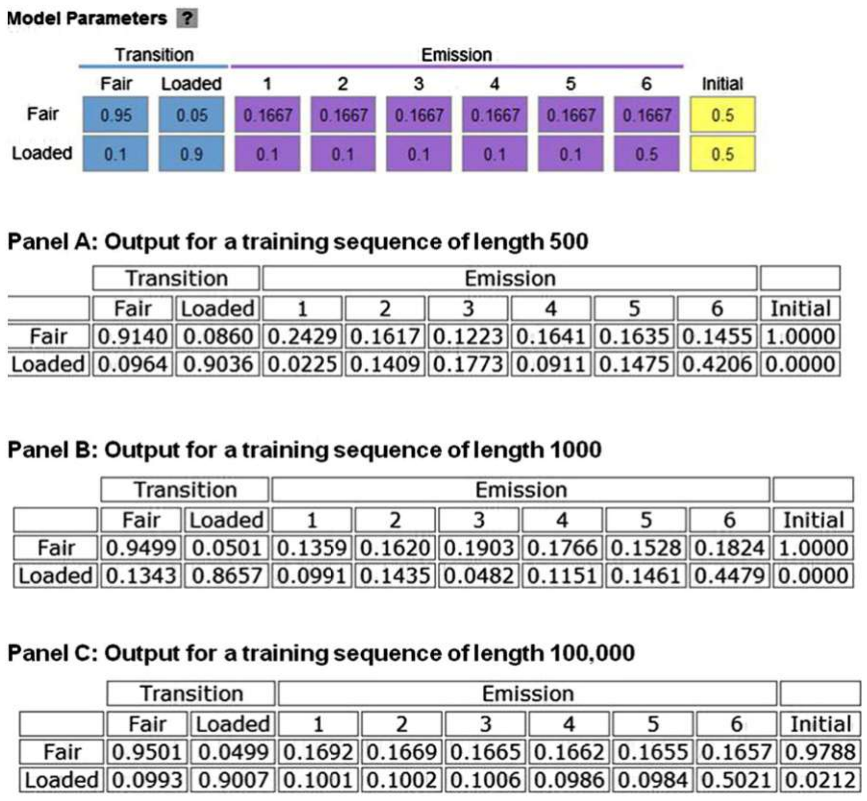
Next, the expected counts from Eqs. (9.11) and (9.12) are substituted in the Eqs. (9.8), generating improved estimates for the transition and emission probabilities. This is the set of parameters that we have denoted by θ_1. Now, repeating the computations given by Eqs. (9.11) and (9.12) with the new values of the parameters from the parameter set θ_1 and substituting those values in Eqs. (9.8) will generate the set of parameters θ_2 and so on. The process is guaranteed to increase the likelihood of the data, that is, for the sequence of parameter approximations $\{\theta_i\}, i = 0, 1, 2, \ldots, P(x|\theta_i) \leqslant P(x|\theta_{i+1})$ [33]. The process terminates when two successive values are either identical or sufficiently close.

As with the other algorithms we have considered in this chapter, to avoid underflows from multiplying small probabilities, the computations are usually carried out in logarithm space. If several sequences $x^1, x^2, \ldots, x^m$ are used for training, Eqs. (9.11) and (9.12) should be modified to include the sum of the expected counts from all sequences:

$$\begin{aligned} A_{jk} &= \sum_{i=1}^{m} \frac{1}{P(x^i)} \sum_{t=1}^{l} f_j^i(t)a_{jk}e_k(x_{t+1}^i)b_k^i(t+1), \\ E_k(b) &= \sum_{i=1}^{m} \frac{1}{P(x^i)} \sum_{\{t=1,\ldots,l|x_t=b\}} f_k^i(t)b_k^i(t), \end{aligned} \tag{9.13}$$

where the superscripts i indicates that the respective probability is computed for the sequence x^i.

Example 9.9. Once again we will illustrate the method on simulated data from the *Dishonest Casino* application in the *CpG Educate* suite with known parameters. We used the algorithm three times for a simulated sequence of length 500, length



Model Parameters ?

	Transition		Emission						
	Fair	Loaded	1	2	3	4	5	6	Initial
Fair	0.95	0.05	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667	0.5
Loaded	0.1	0.9	0.1	0.1	0.1	0.1	0.1	0.5	0.5

Panel A: Output for a training sequence of length 500

	Transition		Emission						
	Fair	Loaded	1	2	3	4	5	6	Initial
Fair	0.9140	0.0860	0.2429	0.1617	0.1223	0.1641	0.1635	0.1455	1.0000
Loaded	0.0964	0.9036	0.0225	0.1409	0.1773	0.0911	0.1475	0.4206	0.0000

Panel B: Output for a training sequence of length 1000

	Transition		Emission						
	Fair	Loaded	1	2	3	4	5	6	Initial
Fair	0.9499	0.0501	0.1359	0.1620	0.1903	0.1766	0.1528	0.1824	1.0000
Loaded	0.1343	0.8657	0.0991	0.1435	0.0482	0.1151	0.1461	0.4479	0.0000

Panel C: Output for a training sequence of length 100,000

	Transition		Emission						
	Fair	Loaded	1	2	3	4	5	6	Initial
Fair	0.9501	0.0499	0.1692	0.1669	0.1665	0.1662	0.1655	0.1657	0.9788
Loaded	0.0993	0.9007	0.1001	0.1002	0.1006	0.0986	0.0984	0.5021	0.0212

FIGURE 9.9

Output from the Baum-Welch algorithm for simulated data for a sequence of length 500 (Panel A), 1000 (Panel B), and 100,000 (Panel C). The HMM model parameters used for the simulation are presented above Panel A.

1000, and length 10,000, respectively. The results are presented in Figure 9.9. The example illustrates what should be intuitively clear: the longer training sequences yield more accurate estimates for the HMM parameters. Due to the stochastic nature of the process, shorter sequences are more likely to not contain the necessary number of transitions and emissions to accurately estimate the HMM parameters. Notice that since we use a single training sequence in all cases, the estimates for the initial distribution cannot be accurate. Using a set of training sequences vs. a single very long training sequence would be advantageous if obtaining estimates for the initial distribution is important.

Exercise 9.16. Experiment with the Baum-Welch algorithm for the *Dishonest Casino* application in the *CpG Educate* suite. Simulate sequences of various lengths using HMM models with different sets of parameters. How well in your opinion is the Baum-Welch algorithm capable of recovering the HMM parameters for (a) short sequences? (b) long sequences? ▽

Exercise 9.17. *Repeat Exercise 9.16, this time using the Baum-Welch algorithm for the *CpG Islands* application in *CpG Educate* (a template *CpG_HMM.csv* to save and upload the HMM parameters from a file is available for download from the volume's website). ▽

9.4.4 Post-Processing

The decoding methods described in this section are purely mathematical and they may not produce completely accurate results when applied to CGIs. Both Viterbi and the posterior decoding methods impose no restrictions on the length of the identified islands or check whether biologically important conditions such as high $\%C + G$ content or high $O/E\ CpG$ ratio (see Section 2) are met. When HMMs are used for CGI identification, the consideration of these properties is done during the *post-processing* stage. At this stage we turn back to the genomic properties of the CGIs that have not been modeled by the HMM. This stage usually includes performing one or more of the following refinements:

- *Combine CGIs separated by short gaps*: Neighboring CGIs that are separated by small gaps of non-island regions are merged into a single larger island. A minimal distance threshold between islands is set in advance and neighboring islands closer than this threshold value are merged. The selection of the threshold values used in the reported literature varies from about 15–20 [27] to up to 100 [8].
- *Check for minimal $\%C + G$ content and $O/E\ CpG$ ratio*: Check to see if the islands identified by the decoding methods meet the biologically relevant thresholds for $\%C + G$ content and $O/E\ CpG$ as described in Section 2. If the identified CGIs do not meet those threshold value requirements, those states will be relabeled as non-islands.
- *Check for minimal length*: As discussed in Section 2, short sequences labeled as CGIs are not of biological interest. Different length-threshold values are used in the literature but those are usually in the range 140–500 bp [8,27]. If the length of a predicted CpG island is less than the threshold value, those states will be relabeled as non-island.

Post-processing is then applied to filter out the regions that do not meet the biological criteria for CGIs with cutoffs.

Example 9.10. In [27] the authors use the general HMM with states $Q = \{A_+, A_-, C_+, C_-, T_+, T_-, G_+, G_-\}$ and emission symbols $M = \{A, C, T, G\}$ that we considered earlier, with no restrictions on the transition or the emission matrices, training it on a set of 1000 sequences from the database embl173hum of all

Table 9.7 Learned HMM parameters used in the *CpG Discover* System [27]. See the text for details.

	Transitions								Emissions				Initial
	A_+	C_+	T_+	G_+	A_-	C_-	T_-	G_-	*A*	*C*	*T*	*G*	
A_+	0.2233	0.202	0.1745	0.3106	0.0229	0.0377	0.0196	0.0164	0.8358	0.0607	0.0501	0.0564	0.0742
C_+	0.2667	0.2511	0.3138	0.0746	0.023	0.0465	0.0226	0.0087	0.0382	0.9006	0.0266	0.0376	0.1005
T_+	0.093	0.252	0.245	0.3459	0.0093	0.0184	0.0251	0.0182	0.0624	0.0486	0.8376	0.0544	0.0309
G_+	0.1846	0.2703	0.1815	0.2326	0.0399	0.0272	0.0178	0.0531	0.0344	0.0451	0.0331	0.8904	0.1181
A_-	0.0323	0.0154	0.0154	0.0342	0.2684	0.1679	0.1496	0.3238	0.8699	0.0434	0.0452	0.0445	0.077
C_-	0.0243	0.0392	0.0146	0.0089	0.3052	0.2486	0.3013	0.065	0.0499	0.8827	0.0283	0.0422	0.2492
T_-	0.021	0.0274	0.0209	0.0372	0.0966	0.2415	0.2114	0.3511	0.0729	0.0389	0.8552	0.036	0.0692
G_-	0.048	0.0303	0.0187	0.0287	0.2575	0.207	0.1737	0.2432	0.0382	0.0307	0.0406	0.8935	0.2877

predicted human CpG islands in the EMBL database at the time the work was done. The parameters resulting from the training are provided in Table 9.7.[12]

Post-processing is then applied to filter out the regions that do not meet the biological criteria for CGIs with cutoffs as follows: If the distance between neighboring islands is less than 20 bp, the regions are merged. The newly extended CGIs and all other predicted islands are then tested to be of length $\geqslant$ 140 bp, have $\%C + G \geqslant 60\%$, and $O/E\ CpG \geqslant 60\%$. If those combined conditions are not met, the states previously recognized as CGIs will be relabeled as non-islands. Table 1 in [27] gives the final results for several test sequences used by the authors and compares those with results obtained from using other CGI locating systems.

9.5 CONCLUSIONS AND DISCUSSION

CGIs are of great interest in genomic analysis and are often used as markers for cancer and gene identification, as well as to investigate methylation profiles [34,35]. Following the original definition and algorithm for CGIs identification by Gardiner-Garden and Frommer [13], a number of sliding window algorithms have been developed with the ability to implement different interpretations and cutoff values for $\%C + G$, $O/E\ CpG$, the island length, gaps between the islands, and size of the sliding window [8,14–16]. It has been shown, however, that such methods do not provide an exhaustive search and that they may miss a large percentage of CGIs [17]. In addition, since sliding windows do not make use of any underlying mathematical structure, alternative approaches based on mathematical models are preferable in order to make some of the problems arising in the context of CGI identification more standardized and tractable.

In this chapter we focused mainly on the use of HMM for CGI identification. Viewing DNA data as output from a HMM elucidates the search for CGIs by placing the problems within a well-developed mathematical framework where efficient methods for decoding, evaluation, and training are readily available. In general, HMMs are widely used to analyze sequences and time series of data under the assumption that they have been generated by a process that cannot be directly observed. Instead, information about the process must be inferred from the observed sequences. In addition to CGI identification, this broad-spectrum approach has also been used for speech recognition [36], protein modeling [37], peptide sequencing [38], multiple DNA sequence alignment [39], gene prediction [40], and many others.

The models introduced in this chapter can also be used as a basis for further extensions and improved HMMs for CGI identification. Some work in this direction includes the development of an extensible approach that summarizes the evidence of CpG islands as probability scores [41], a hybrid visualization HMM method [18], a modified HMM approach with Poisson emission probabilities [42], HMM approaches

[12]The *CpG Islands* application in the *CpG Educate* suite uses these values as default parameters for the CGI HMM.

to improving the power of pattern detection [43], improved performance Viterbi and EM algorithms [44], and methods for speeding up HMM decoding [45].

The chapter includes a number of exercises of both applied and theoretical nature and an online project *Investigating "Predicted" Genes* available from the volume's website. We encourage the reader to attempt each exercise/project as soon as the relevant material for its execution has been introduced. Most of the applied exercises use the *CpG Educate* suite of web applications developed for this chapter and available at http://inspired.jsu.edu/∼agarrett/cpg/[30]. *CpG Educate* uses the General Hidden Markov Model library (http://ghmm.org/) for the implementation of all HMM algorithms. A key feature of *CpG Educate* is that it provides the option to simulate data for most of the HMMs used in the chapter and compare the results from decoding and training with those from the simulations. After completing the chapter exercises, we invite the reader to undertake further experimentations with the decoding and learning methods for various simulated and actual data.

Acknowledgments

Authors Davies, Kirkwood, and Robeva gratefully acknowledge the support of the National Science Foundation under DEU award #0737467.

9.6 SUPPLEMENTARY MATERIALS

A supplementary project and additional files and data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/B978-0-12-415780-4.00019-3 and from the volume's website http://booksite.elsevier.com/9780124157804.

References

[1] Bird A. DNA methylation patterns and epigenetic memory. Genes Dev 2002;16:6–21.
[2] Klose RJ, Bird AP. Genomic DNA methylation: the mark and its mediators. Trends Biochem Sci 2006;31:89–97.
[3] Sorensen AL, Timoskainen S, West FD, Vekterud K, Boquest AC, Ahrlund-Richter L, et al. Lineage-specific promoter DNA methylation patterns segregate adult progenitor cell types. Stem Cells Dev 2010;19:1257–66.
[4] Isagawa T, Nagae G, Shiraki N, Fujita T, Sato N, Ishikawa S, et al. DNA methylation profiling of embryonic stem cell differentiation into the three germ layers. PLoS One 2011;6:e26052.
[5] Collas P. Programming differentiation potential in mesenchymal stem cells. Epigenetics 2010;5:476–82.

[6] Neddermann P, Jiricny J. The purification of a mismatch-specific thymine-DNA glycosylase from HeLa cells. Journal Biol Chem 1993;268:21218–24.
[7] Straussman R, Nejman D, Roberts D, Steinfeld I, Blum B, Benvenisty N, et al. Developmental programming of CpG island methylation profiles in the human genome. Nature Struct Mol Biol 2009;16:564–71.
[8] Takai D, Jones PA. Comprehensive analysis of CpG islands in human chromosomes 21 and 22. Proc Natl Acad Sci USA 2002;99:3740–5.
[9] Ashley DJB. The two hit and multiple hit theories of carcinogenesis. Br J Cancer 1969;23:313–28.
[10] Renan MJ. How many mutations are required for tumorigenesis? Implications from human cancer data. Mol Carcinog 1993;7:139–46.
[11] Schappert-Kimmijser J, Hemmes JGD, Nijland R. The heredity of retinoblastoma. Ophthalmologica 1966;151:197–213.
[12] Noburi T, Miura K, Wu DJ, Lois A, Takabayashi K, Carson D. Deletions of the cyclin dependent kinase-4 inhibitor gene in multiple human cancers. Nature 1994;368:753–6.
[13] Gardiner-Garden M, Frommer M. CpG Islands in Veribrate Genome. J Mol Biol 1987;196:261–82.
[14] Rice P, Longden I, Bleasby A. EMBOSS: The European Molecular Biology Open Software Suite. TIG 2000;16:276–7.
[15] Ponger L, Mouchiroud D. CpGProD: identifying CpG islands associated with transcription start sites in large genomic mammalian sequences. Bioinformatics 2002;18:631–3.
[16] Wang Y, Leung FCC. An evaluation of new criteria for CpG islands in the human genome as gene markers. Bioinformatics 2004;20:1170–7.
[17] Hsieh F, Chen SC, Pollard K. A nearly exhaustive search for CpG islands on whole chromosomes. Int J Biostatistics 2009;5(1), Art. 14.
[18] Rambally G, Rambally R. A hybrid visualization Hidden Markov Model approach to identifying CG-islands in DNA sequences, In Southeastcon, 2008. IEEE; 3–6 April 2008. p. 1–6.
[19] Hackenberg M, Previti C, Luque-Escamilla P, Carpena P, Martinez-Aroza J, Oliver J. CpGcluster: a distance-ased algorithm for CpG-island detection. BMC Bioinform 2006;7:446.
[20] Hackenberg M, Barturen G, Carpena P, Luque-Escamilla PL, Previti C, Oliver JL. Prediction of CpG-island function: CpG clustering vs. sliding-window methods. BMC Genom 2010;26:327.
[21] Sujuan Y, Asaithambi A, Liu Y. CpGIF: an algorithm for the identification of CpG islands. Bioinformation 2008;2:335–8.
[22] Chuang LY, Huang HC, Lin MC, Yang CH. Particle swarm optimization with reinforcement learning for the prediction of CpG islands in the human genome. PLoS One 2011;6:e21036.
[23] Durbin R, Eddy S, Krogh A, Mitchison G. Biological sequence analysis. Probabilistic models of proteins and nucleic acids. Cambridge, UK, Cambridge University Press; 1998.

[24] Pahter L, Sturmfels B. Algebraic statistics for computational biology. Cambridge, UK: Cambridge University Press; 2005.
[25] Norris JR. Markov chains. Cambridge: Cambridge University Press; 1997.
[26] Elliot JR, Aggoun L, Moore JB. Hidden markov models: estimation and control (corrected 3rd printing). New York: Springer; 2008.
[27] Lan M, Xu Y, Li L, Wang F, Zuo Y, Tan CL, et al. CpG-Discover: a machine learning approach for CpG island identification from human DNA sequence. In: Proceedings of international joint conference on neural networks, Atlanta, Georgia, USA; June 14–19, 2009. p. 1702–7.
[28] Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans Inform Theory 1967;IT-13:260–9.
[29] Viterbi A. A personal history of the viterbi algorithm. IEEE Signal Process Mag 2006;23:120–42.
[30] Garrett, A. CpG EducateSoftware tutorial; 2012, <http://inspired.jsu.edu/~agarrett/cpg/CpGEducate.pdf>.
[31] Baum LE, Petrie T, Soules G, Weiss NA. Maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann Math Stat 1970;41:164–71.
[32] Welch LR. The Shannon lecture: hidden Markov models and the Baum-Welch algorithm. IEEE Inform Soc Newslett 2003;53(4), December 2003.
[33] McLachlan G, Krishnan T. The EM algorithms and extensions. 2nd ed. Hoboken, NJ: Wiley; 2008.
[34] Illingworth PR, Bird AP. CpG islands – a rough guide. FEBS Lett 2009;583: 1713–20.
[35] Bobbie PO, Reams R, Suther S, Brown CP. Finding molecular signature of prostate cancer: an algorithmic approach. In: Proceedings of the 2006 international conference on bioinformatics & computational biology, BIOCOMP'06, Las Vegas, Nevada, USA, June 26–29, 2006, p. 265–9.
[36] Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 1989;77:257–85.
[37] Krogh A, Brown M, Mian IS, Sjlander K, Haussler D. Hidden Markov models in computational biology. Application to protein modeling. J Mol Biol 1994;235:1501–31.
[38] Fischer B, Roth V, Roos F, Grossmann J, Baginsky S, Widmayer P, et al. NovoHMM: a hidden Markov model for de novo peptide sequencing. Anal Chem 2005;77:7265–73.
[39] Do CB, Mahabhashyam MS, Brudno M, Batzoglou S. ProbCons: probabilistic consistency-based multiple sequence alignment. Genome Res 2005;15:330–40.
[40] Bernal A, Crammer K, Hatzigeorgiou A, Pereira F. Global discriminative learning for higher-accuracy computational gene prediction. PLoS Comput Biol 2007;16:e54.
[41] Wu H, Caffo B, Jaffee HA, Irizarry RA, Feinberg AP. Redefining CpG islands using hidden Markov models. Biostatistics 2010;11:499–514.

[42] Irizarry RA, Wu H, Feinberg AP. A species-generalized probabilistic model-based definition of CpG islands. Mamm Genome 2009;20:674–80.
[43] Zhai Z, Ku SY, Luan Y, Reinert G, Waterman MS, Sun F. The power of detecting enriched patterns: an HMM approach. J Comput Biol 2010;17:581–92.
[44] Lam T, Mayer A. Efficient algorithms for training the parameters of hidden Markov models using stochastic expectation maximization (EM) training and Viterbi training. Alg Mol Biol 2010;5:38.
[45] Lifshits Y, Mozes S, Weimann O, Ziv-Ukelson M. Speeding up HMM decoding and training by exploiting sequence repetitions. Algorithmica 2009;54:379–99.