
WHY HAVE BIOLOGISTS STILL NOT DEVELOPED AN HIV VACCINE?
Hidden Markov Models
CHŌ!
HAN!
Owning a Bioinformatics Time Machine isn't always fun and games. One night, we made a fateful decision to travel back to early 19th Century Tokyo to eat at the first sushi restaurant.
We took a wrong turn and found ourselves outside a gambling house. Always up for a new adventure, we decided to try our hands at a game or two.
How were we supposed to know that the casino was run by yakuza?!?!..

Classifying the HIV Phenotype

How does HIV evade the human immune system?

In 1984, US Health and Human Services Secretary Margaret Heckler declared that an HIV vaccine would be available within two years, stating, "Yet another terrible disease is about to yield to patience, persistence and outright genius."

In 1997, Bill Clinton established a new research center at the National Institutes of Health with the goal of developing an HIV vaccine. In his words, "It is no longer a question of *whether* we can develop an AIDS vaccine, it is simply a question of *when*."

In 2005, Merck began clinical trials of an HIV vaccine but discontinued them two years later after learning that the vaccine actually *increased* the risk of HIV infection in some recipients.

Today, despite enormous investment and ongoing clinical trials, we are still far from an HIV vaccine, and 35 million people are living with the disease. Scientists have made great progress in developing a successful **antiretroviral therapy**, a drug cocktail that stabilizes an infected patient's symptoms. However, this therapy does not cure AIDS and cannot prevent the spread of HIV, and so it does not hold the promise of a true vaccine for containing the AIDS epidemic.

Classical vaccines against viruses are often made from the surface proteins of a virus. These vaccines stimulate the human immune system to recognize viral envelope proteins as foreign, destroy them, and keep a record of it, so that the immune system can identify and eradicate the virus in a later encounter.

However, HIV viral envelope proteins are extremely variable because the virus must mutate rapidly in order to survive (see DETOUR: The Red Queen Effect). The HIV population in a *single* infected individual rapidly evolves to evade the human immune system (Figure 10.1), not to mention that HIV strains taken from *different* patients represent multiple highly diverged subtypes. Therefore, a successful HIV vaccine must be broad enough to account for this variability.

PAGE 229

In an effort to counteract HIV's variability, we could create a single peptide that contains the least variable segments of the envelope proteins taken from all known HIV strains and use this peptide as the basis for a universal vaccine fighting all HIV strains. However, not only do HIV envelope proteins mutate fast, but they are also "masked" by **glycosylation**, a post-translational modification that often makes these proteins invisible to the human immune system (see DETOUR: Glycosylation). As a result, all attempts at developing an HIV vaccine have thus far failed.

PAGE 229

HIV has just nine genes, and in this chapter we will focus on the rapidly mutating *env* gene, which has a mutation rate of 1 to 2% per nucleotide per year. The protein

```
VKKLGEQFR-NKTIIFNQPSGGDLEIVMHSFNCGGEFFYCNTTQLFN----------NSTES------DTITL
VKKLGEQFR-NKTIIFNQPSGGDLEIVMHSFNCGGEFFYCNTTQLFN----------NSTDNG-----DTITL
VKKLGEQFR-NKTIIFNQPSGGDLEIVMHSFNCGGEFFYCNTTQLFD----------NSTESNN----DTITL
VDKLREQFGKNKTIIFNQPSGGDLEIVMHTFNCGGEFFYCNTTQLFNSTWNS---TGNGTESYNGQENGTITL
VDKLREQFGKNKTIIFNQPSGGDLEIVMHTFNCGGEFFYCNTTQLFNSTWNG---TNTT--GLDG--NDTITL
VDKLREQFGKNKTIIFNQSSGGDLEIVTHTFNCGGEFFYCNTTQLFNSNWTG---NSTE--GLHG--DDTITL
VKKLGEQFG-NKTIIFNQSSGGGLEIVMHSFNCGGEFFYCNTTQLFNN--TR-----NSTESNNGQGNDTTTL
VKKLREQFGKNKTIIFKQSSGGDLEIVTHTFNCAGEFFYCNTTQLFNSNWTE-----NSITGLDG--NDTITL
VGKLREQFGK-KTIIFNQPSGGDLEIVMHSFNCQGEFFYCNTTRLFNSTWDNSTWNSTGKDKENGN-NDTITL
```

FIGURE 10.1 A multiple alignment of a short region of gp120 proteins sampled from a single HIV-positive patient at nine different time points. Almost half of the columns (shown in darker text) are not conserved across all time points, illustrating how quickly HIV evolves, even within an individual host. Amino acids differing from the most common symbol in a column are shown in blue.

encoding the *env* gene then gets cut into **glycoprotein gp120** (approx. 480 amino acids) and **glycoprotein gp41** (approx. 345 amino acids). Together, gp120 and gp41 form the **envelope spike**, which mediates entry of the HIV virus into human cells.

Since HIV mutates so fast, different HIV isolates may have different phenotypes, thus requiring different drug cocktails. For example, HIV viruses can be divided into fast-replicating **syncytium-inducing (SI)** isolates and slow-replicating **non-syncytium-inducing (NSI)** isolates. During infection, viral proteins like gp120 that are used by HIV to enter the cell are transported to the cell surface, where they can cause the host cell membrane to fuse with neighboring cells. This causes dozens of human cells to fuse their cell membranes into a giant, nonfunctional **syncytium**, or abnormal multinucleate cell (Figure 10.2). This mechanism allows an SI virus to kill many human cells by infecting only one.

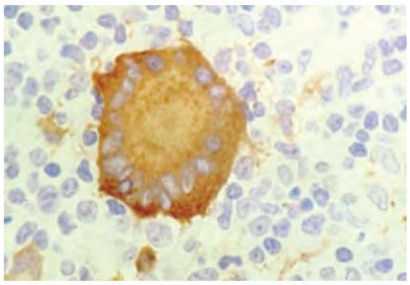

FIGURE 10.2 A syncytium with multiple nuclei in an HIV patient.

Because gp120 is important in classifying a virus as SI or NSI, biologists are interested in determining which amino acids in gp120 can be used for this classification.

In 1992, Jean-Jacques De Jong analyzed a multiple alignment of the **V3 loop** region in gp120 (Figure 10.3 (top)) and devised the **11/25 rule**, which asserts that an HIV strain is more likely to have an SI phenotype if the amino acid at either positions 11 or 25 of its V3 loop is arginine or lysine. It later was shown that many other positions influence the SI/NSI phenotype.

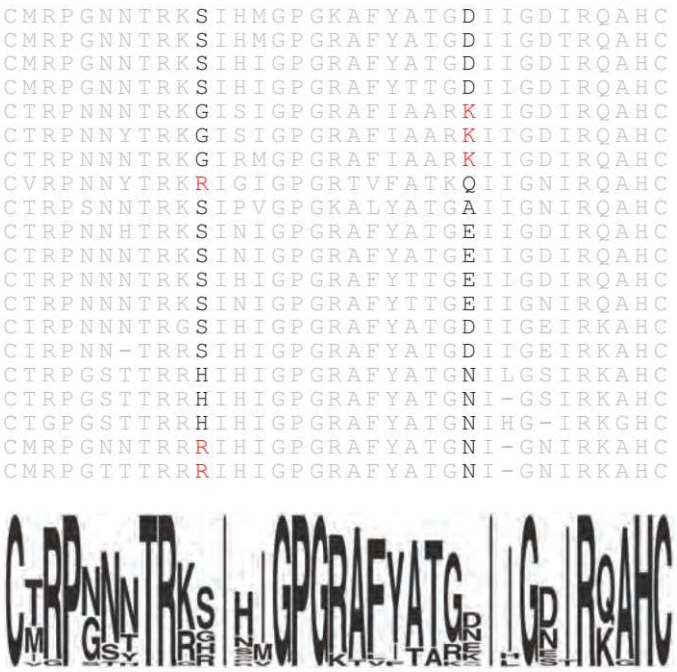

FIGURE 10.3 A multiple alignment of the V3 loop region taken from twenty HIV patients, and the motif logo of this alignment. The alignment's 11th and 25th columns are shown in darker text; occurrences of arginine (R) or lysine (K) in these columns are shown in red. The 11/25 rule will classify six of the patients as infected with an SI isolate. Although the V3 loop is an important and rather conserved segment of gp120, the level of conservation varies across different positions. For example, whereas the first and last positions are extremely conserved, positions 11 and 25 exhibit high variation.

Limitations of sequence alignment

Before biologists could even start to examine the question of predicting HIV phenotypes from gp120 sequences, they faced the problem of constructing accurate multiple alignments of these sequences. Indeed, a single misalignment, placing an incorrect amino

acid at a position influencing the SI/NSI phenotype, could lead to a faulty classification of HIV phenotypes. And we already know from Chapter 5 that constructing a multiple alignment of highly diverged sequences is a difficult algorithmic problem.

Figure 10.3 (bottom) shows a motif logo from the V3 loop of gp120 and illustrates that some positions in gp120 are relatively conserved, whereas others are extremely variable. Furthermore, this motif logo does not account for insertions and deletions, which are prevalent in other regions of gp120 that are less conserved than the V3 loop. These insertions and deletions make analyzing gp120 even more complex.

Because the columns of the multiple alignment in Figure 10.3 have varying levels of conservation, we question the wisdom of using the *same* amino acid scoring matrix (as well as indel penalties) across different columns of an alignment. A better approach would use a different scoring approach at *different* columns. For example, an amino acid differing from R in position 3 of the alignment in Figure 10.3 should incur a larger penalty than an amino acid deferring from S in position 11.

In other words, the *problem formulation* of multiple sequence alignment introduced in Chapter 5 does not offer an adequate translation of the biological problem of HIV classification into an algorithmic problem. We must therefore devise a new problem formulation for sequence alignment that will lead to a statistically solid analysis of gp120 proteins. But first, we will ask you to join us in our time machine for one more trip.

Gambling with Yakuza

The Japanese crime syndicates called yakuza descend from groups of 18th Century traveling gamblers called bakuto. (In fact, "yakuza" is the name of a losing hand in a Japanese card game.) One of the most popular games that the bakuto would host in their makeshift casinos is called **Chō-Han**. In this game, which literally translates as "even-odd", the dealer rolls two dice, and players wager on whether the sum of the dice will be even or odd.

Although playing Chō-Han in a yakuza gambling house would undoubtedly make for a fun evening, we can play an equivalent — albeit less exciting — game called "Heads or Tails" by flipping a coin and wagering on the outcome. Assume that for some strange reason, more people wager on tails than on heads in this game. Then a crooked dealer might use a biased coin that is more likely to result in heads than tails. We will assume that this biased coin results in heads with probability 3/4.

STOP and Think: Say that you play Heads or Tails 100 times, and the coin produces heads 63 times. Is the dealer cheating? Was the coin fair or biased?

This question is not well-formulated, since either coin could have produced any sequence of flips. But can we determine which coin is *more likely* to have been used?

We write the probabilities of tails ("T") and heads ("H") for the fair coin (F) as

$$\Pr_F(\text{"H"}) = 1/2 \qquad \Pr_F(\text{"T"}) = 1/2$$

and the probabilities for the biased coin (B) as

$$\Pr_B(\text{"H"}) = 3/4 \qquad \Pr_B(\text{"T"}) = 1/4$$

Since coin flips are independent events, the probability that n flips of the fair coin will generate a given sequence $x = x_1 x_2 \ldots x_n$ with k occurrences of "H" is

$$\Pr(x|F) = \prod_{i=1}^{n} \Pr_F(x_i) = (1/2)^n .$$

On the other hand, the probability that the biased coin will generate the same sequence is

$$\Pr(x|B) = \prod_{i=1}^{n} \Pr_B(x_i) = (1/4)^{n-k} \cdot (3/4)^k = 3^k/4^n.$$

If $\Pr(x|F) > \Pr(x|B)$, then the dealer more likely used a fair coin, and if $\Pr(x|F) < \Pr(x|B)$, then the dealer more likely used a biased coin. The numbers $(1/2)^n$ and $3^k/4^n$ are so small for large n that in order to compare them, we will use their **log-odds ratio**,

$$\log_2 \left(\frac{\Pr(x|F)}{\Pr(x|B)} \right) = \log_2 \left(\frac{2^n}{3^k} \right) = n - k \cdot \log_2 3.$$

EXERCISE BREAK: Show that $\Pr(x|F)$ is larger than $\Pr(x|B)$ when the log-odds ratio is positive (i.e., when $k/n < 1/\log_2 3$) and smaller than $\Pr(x|B)$ when the log-odds ratio is negative (i.e., when $k/n > 1/\log_2 3$).

Returning to our example of witnessing $k = 63$ heads in $n = 100$ flips, the log-odds ratio is positive, since

$$k/n = 0.63 < 1/\log_2 3 \approx 0.6309.$$

It follows that $\Pr(x|F) > \Pr(x|B)$, and so the dealer most likely used a fair coin, even though 63 is closer to 75 than it is to 50.

Two Coins up the Dealer's Sleeve

In bakuto gambling houses, a Chō-Han dealer would remove his shirt during play, displaying a tattooed chest, in order to reduce any suspicions of dice tampering. (These tattoos would later become a yakuza tradition.) We will assume, however, that in Heads or Tails, the crooked dealer is wearing a shirt and keeps both coins up a sleeve, secretly changing them back and forth whenever he likes during the sequence of flips. Since he does not want to be caught switching coins, he does so only occasionally.

We will assume that the crooked dealer switches coins with probability 0.1 after each flip. Given a sequence of coin flips, we must determine when the dealer used the biased coin and when he used the fair coin.

Casino Problem:
Given a sequence of coin flips, determine when the crooked dealer used a fair coin and when he used a biased coin.

> **Input**: A sequence $x = x_1 x_2 \ldots x_n$ of coin flips made by two possible coins (F and B).
> **Output**: A sequence $\pi = \pi_1 \pi_2 \ldots \pi_n$, with each π_i being equal to either F or B and indicating that x_i is the result of flipping the fair or biased coin, respectively.

Unfortunately, this problem is poorly stated, since either coin can generate any outcome. Instead, we need to determine the *most likely* sequence of coins used by the dealer.

STOP and Think: Can you reformulate the Casino Problem so that it makes sense?

A well-defined computational problem for finding the most likely sequence of coins used by the dealer should somehow grade different sequences π as better answers than others. One approach to guessing the most likely coin the dealer used for each flip would be to slide a window (of length $t < n$) along the sequence of flips $x = x_1 \ldots x_n$ and then calculate the log-odds ratio under each window. If the log-odds ratio of the window falls below zero, then the dealer most likely used the biased coin inside this window; otherwise, the dealer most likely used the fair coin.

STOP and Think: Do you see any problems with this method?

There are two issues with the window-sliding approach. First, we have no apparent choice for the length of the window. Second, overlapping windows may classify the same outcome as caused by both the fair and biased coins. For example, if $x =$ "HHHHHTTHHHTTTTT", then the window $x_1 \dots x_{10} =$ "HHHHHTTHHH" has a negative log-odds ratio, and the window $x = x_6 \dots x_{15} =$ "TTHHHTTTTT" has a positive log-odds ratio. So which coin did the dealer use on the flips $x_6 \dots x_{10} =$ "TTHHH"?

Finding CG-Islands

In the next section, we will improve our method of grading sequences of coin flips. The solution will lead us to a computational paradigm that has been successfully applied to a wide array of bioinformatics problems, including HIV comparison. For now, however, you may still not believe how coin flipping could possibly relate to sequence comparison. Thus, we will briefly describe a different biological problem that more clearly relates to our coin flipping analogy.

In the early 20th Century, Phoebus Levene discovered the four nucleotides making up DNA. At this time, little was known about DNA (Watson & Crick's double helix paper was still half a century away). As a result, Levene doubted that DNA could store genetic information using just a four-letter alphabet, and he hypothesized that DNA comprised nearly equal amounts of adenine, cytosine, guanine, and thymine.

A century later, we know that complementary nucleotides on *opposing* strands of DNA have equal frequencies because of base pairing — ignoring extremely rare base-pairing errors. However, it is not true that nucleotide frequencies are approximately equal on a *single* strand of DNA. For example, different species have widely varying **GC-content**, or the percentage of cytosine and guanine nucleotides in a genome. For example, the human genome's GC-content is approximately 42%.

After accounting for the human genome's skewed GC-content, we might expect that each of the dinucleotides `CC`, `CG`, `GC`, and `GG` would occur in the human genome with frequency $0.21 \cdot 0.21 = 4.41\%$. However, the frequency of `CG` in the human genome is only about 1%! This dinucleotide is so rare because of **methylation**, the most common DNA modification, which typically adds a methyl group (CH_3) to the cytosine nucleotide within a `CG` dinucleotide. The resulting methylated cytosine has the tendency to further deaminate into thymine (see DETOUR: DNA Methylation). As a result of methylation, `CG` is the least frequent dinucleotide in many genomes.

PAGE 229

Nevertheless, methylation is often suppressed around genes in areas called **CG-islands**, where `CG` appears relatively frequently (Figure 10.4). If you were to sequence a

mammalian genome that you knew nothing about, perhaps one of the first things you might do in order to find genes in this genome is look for CG-islands.

STOP and Think: How would you identify CG-islands in a genome?

	A	C	G	T
A	0.053	0.079	0.127	0.036
C	0.037	0.058	0.058	0.041
G	0.035	0.075	0.081	0.026
T	0.024	0.105	0.115	0.050

	A	C	G	T
A	0.087	0.058	0.084	0.061
C	0.067	0.063	0.017	0.063
G	0.053	0.053	0.063	0.042
T	0.051	0.070	0.084	0.084

FIGURE 10.4 Dinucleotide frequencies for a collection of CG-islands (left) and non-CG-islands (right) in the human genome computed for a single strand of the X chromosome. Frequencies of CG are shown in red.

A naive approach to search for CG-islands in a genome would slide a window down the genome, declaring windows with higher frequencies of CG as potential CG-islands. The disadvantages of this method are analogous to those of using a sliding window to determine which coin the crooked dealer most likely used at any given point in time. We do not know how long the window should be, and overlapping windows may simultaneously classify the same genomic position as belonging to a CG-island and as not belonging to a CG-island.

Hidden Markov Models

From coin flipping to a Hidden Markov Model

Our goal is to develop a single concept that models both the crooked dealer and the search for CG-islands in a genome. To this end, we will think about the crooked dealer not as a human but as a primitive machine. We do not know how this machine is constructed, but we do know that it proceeds in a sequence of steps; in each step, it is in one of two hidden states, F and B, and it emits a symbol, "H" or "T".

After each step, the machine makes two decisions:

- Which hidden state will I move to next?
- Which symbol will I emit in that state?

The machine answers the first question by choosing randomly among the F and B states, with probability 0.9 of remaining in its current state and probability 0.1 of changing states. The machine answers the second question by choosing between the symbols "H" and "T" with probabilities that depend on the state it is in. In our coin flipping example, the probabilities for state F (0.5 and 0.5) differ from the probabilities for state B (0.75 and 0.25). Our goal is to infer the machine's most likely sequence of states by analyzing the sequence of symbols that it emits.

We have just transformed the dealer into an abstract machine called a **Hidden Markov Model (HMM)**. The only difference between our specialized "coin flipping machine" and the general concept of an HMM is that the latter can have an arbitrary number of states and may have arbitrary probability distributions governing which state to move into and which symbols to emit. In general, an HMM ($\sum$, *States*, *Transition*, *Emission*) is defined by a set of four objects:

- an alphabet Σ of emitted symbols;
- a set *States* of **hidden states**;
- a $|States| \times |States|$ matrix $Transition = (transition_{l,k})$ of **transition probabilities**, where $transition_{l,k}$ represents the probability of moving from state l to state k;
- a $|States| \times |\Sigma|$ matrix $Emission = (emission_k(b))$ of **emission probabilities**, where $emission_k(b)$ represents the probability of emitting symbol b from alphabet Σ when the HMM is in state k.

For each state l,

$$\sum_{\text{all states } k} transition_{l,k} = 1$$

and

$$\sum_{\text{all symbols } b \text{ from } \Sigma} emission_l(b) = 1\,.$$

EXERCISE BREAK: What are Σ, *States*, *Transition*, and *Emission* for the HMM modeling the crooked dealer?

The HMM diagram

As illustrated in Figure 10.5, an HMM can be visualized using an **HMM diagram**, a graph in which every state is represented by a solid node. Solid directed edges connect every pair of nodes, as well as every node to itself. Each such edge is labeled with the transition probability of moving from one state to the other (or remaining in the same state). In addition, the HMM diagram has dashed nodes representing each possible symbol from the alphabet Σ and dashed edges connecting each state to each dashed node. Each such edge is labeled by the probability that the HMM will emit this symbol while in the given state.

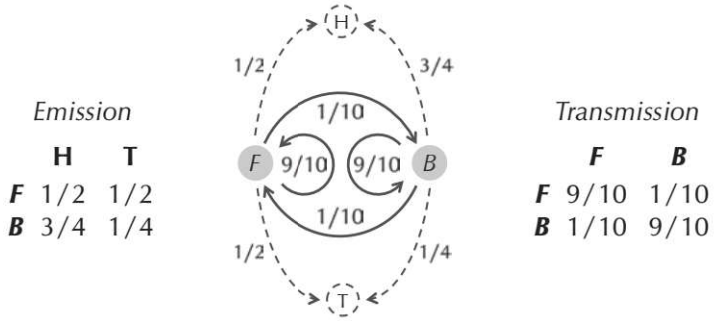

FIGURE 10.5 The transition and emission probability matrices for the crooked dealer HMM described by the HMM diagram shown in the center. This HMM has two states (gray nodes), F and B. In each state, the HMM can emit one of two symbols (dashed nodes), heads ("H") or tails ("T"), with the probabilities shown along dashed edges. Transition probabilities are shown on solid edges; the crooked dealer HMM transitions between states F and B with probability $1/10$ and remains in the same state with probability $9/10$.

A **hidden path** $\pi = \pi_1 \dots \pi_n$ in an HMM is the sequence of states that the HMM passes through; such a path corresponds to a path of solid edges in the HMM diagram. Figure 10.6 presents an example in which the crooked dealer HMM produces a sequence of flips x = "THTHHHTHTTH" with hidden path $\pi = FFFBBBBBFFF$, i.e., the fair coin is used for the first three flips and last three flips, and the biased coin is used for the five intermediate flips.

Reformulating the Casino Problem

We can now rephrase the improperly formulated Casino Problem as finding the most likely hidden path π for a string x of symbols emitted by an HMM. To solve this problem,

i	1	2	3	4	5	6	7	8	9	10	11
x	T	H	T	H	H	H	T	H	T	T	H
π	F	F	F	B	B	B	B	B	F	F	F
$\Pr(\pi_i \rightarrow \pi_{i+1})$	$\frac{1}{2}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{1}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	$\frac{1}{10}$	$\frac{9}{10}$	$\frac{9}{10}$
$\Pr(x_i \mid \pi_i)$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

FIGURE 10.6 A sequence x of emitted symbols along with a hidden path π for the crooked dealer HMM. $\Pr(\pi_i \rightarrow \pi_{i+1})$ denotes the probability $transition_{\pi_i, \pi_{i+1}}$ of transitioning from state π_i to π_{i+1}. $\Pr(\pi_0 \rightarrow \pi_1)$ is set equal to $1/2$ to comply with the assumption that in the beginning, the dealer is equally likely to use the fair or biased coin. $\Pr(x_i|\pi_i)$ denotes the probability that the dealer produced symbol x_i from state π_i and is equal to $emission_{\pi_i}(x_i)$.

we will first consider the simpler problem of computing the probability $\Pr(x, \pi)$ that an HMM follows the hidden path $\pi = \pi_1 \ldots \pi_n$ and emits the string $x = x_1 \ldots x_n$. Note that

$$\sum_{\text{all strings of emitted symbols } x} \; \sum_{\text{all hidden paths } \pi} \Pr(x, \pi) = 1.$$

STOP and Think: What is $\Pr(x, \pi)$ for the x and π in Figure 10.6?

Each emitted string x has probability $\Pr(x)$, which is independent of the hidden path taken by the HMM:

$$\Pr(x) = \sum_{\text{all hidden paths } \pi} \Pr(x, \pi).$$

Each hidden path π has probability $\Pr(\pi)$, which is independent of the string that the HMM emits:

$$\Pr(\pi) = \sum_{\text{all strings of emitted symbols } x} \Pr(x, \pi).$$

The event "the HMM follows the hidden path π and emits x" can be thought of as a combination of two consecutive events:

- The HMM follows the path π. The probability of this event is $\Pr(\pi)$.
- The HMM emits x, given that the HMM follows the path π. We refer to the probability of this event as the **conditional probability** of x given π, denoted $\Pr(x|\pi)$.

Both of these events must occur for the HMM to follow path π and emit string x, which implies that

$$\Pr(x, \pi) = \Pr(x|\pi) \cdot \Pr(\pi).$$

PAGE 230 To learn more about this formula, see **DETOUR: Conditional Probability**.

To compute $\Pr(x, \pi)$, we will first compute $\Pr(\pi)$. As shown in Figure 10.6, we write $\Pr(\pi_i \rightarrow \pi_{i+1})$ to denote the transition probability of the HMM transitioning from state π_i to π_{i+1}. For simplicity, we assume that in the beginning, the dealer is equally likely to use the fair or biased coin, an assumption that is modeled by setting $\Pr(\pi_0 \rightarrow \pi_1) = 1/2$ in Figure 10.6, where π_0 is a "silent" **initial state** that does not emit any symbols. The probability of π is therefore equal to the product of its transition probabilities (purple elements in Figure 10.6),

$$\Pr(\pi) = \prod_{i=1}^{n} \Pr(\pi_{i-1} \rightarrow \pi_i) = \prod_{i=1}^{n} transition_{\pi_{i-1}, \pi_i}.$$

Probability of a Hidden Path Problem:
Compute the probability of an HMM's hidden path.

Input: A hidden path π in an HMM (Σ, *States*, *Transition*, *Emission*).
Output: The probability of this path, $\Pr(\pi)$.

Note that we have already computed $\Pr(x|\pi)$ for the crooked dealer HMM when the dealer's hidden path consisted only of B or F, which we wrote as $\Pr(x|B)$ and $\Pr(x|F)$, respectively. To compute $\Pr(x|\pi)$ for a general HMM, we will write $\Pr(x_i|\pi_i)$ to denote the emission probability $emission_{\pi_i}(x_i)$ that symbol x_i was emitted given that the HMM was in state π_i (Figure 10.6). As a result, for a given path π, the HMM emits a string x with probability equal to the product of emission probabilities along that path,

$$\begin{aligned} \Pr(x|\pi) &= \prod_{i=1}^{n} \Pr(x_i|\pi_i) \\ &= \prod_{i=1}^{n} emission_{\pi_i}(x_i)\,. \end{aligned}$$

Probability of an Outcome Given a Hidden Path Problem:
Compute the probability that an HMM will emit a string given its hidden path.

Input: A string $x = x_1 \dots x_n$ emitted by an HMM (Σ, *States*, *Transition*, *Emission*) and a hidden path $\pi = \pi_1 \dots \pi_n$.
Output: The conditional probability $\Pr(x|\pi)$ that x will be emitted given that the HMM follows the hidden path π.

Returning to our formula for $\Pr(x, \pi)$, the probability that an HMM follows path π and emits string x can be written as a product of emission and transition probabilities,

$$\begin{aligned}\Pr(x, \pi) &= \Pr(x|\pi) \cdot \Pr(\pi) \\ &= \prod_{i=1}^{n} \Pr(x_i|\pi_i) \cdot \Pr(\pi_{i-1} \rightarrow \pi_i) \\ &= \prod_{i=1}^{n} \mathit{emission}_{\pi_i}(x_i) \cdot \mathit{transition}_{\pi_{i-1}, \pi_i}.\end{aligned}$$

EXERCISE BREAK: Compute $\Pr(x, \pi)$ for the x and π in Figure 10.6. Can you find a better explanation for x = "THTHHHTHTTH" than $\pi = FFFBBBBBFFF$?

STOP and Think: Now that you have learned about HMMs, try designing an HMM that will model searching for CG-islands in a genome. What barriers do you encounter?

The Decoding Problem

The Viterbi graph

As we stated in the previous section, in both the crooked dealer and CG-island HMMs, we are looking for the most likely hidden path π for an HMM that emits a string x. In other words, we would like to maximize $\Pr(x, \pi)$ among all possible hidden paths π.

Decoding Problem:
Find an optimal hidden path in an HMM given a string of its emitted symbols.

Input: A string $x = x_1 \dots x_n$ emitted by an HMM (Σ, *States*, *Transition*, *Emission*).
Output: A path π that maximizes the probability $\Pr(x, \pi)$ over all possible paths through this HMM.

In 1967, Andrew Viterbi used an HMM-inspired analog of a Manhattan-like grid to solve the Decoding Problem. For an HMM emitting a string of n symbols $x = x_1 \dots x_n$, the nodes in the HMM's **Viterbi graph** are divided into $|States|$ rows and n columns (Figure 10.7 (middle)). That is, node (k, i) represents state k and the i-th emitted symbol. Each node is connected to all nodes in the column to its right; the edge connecting $(l, i-1)$ to (k, i) corresponds to transitioning from state l to state k (with probability $transition_{l,k}$) and then emitting symbol x_i (with probability $emission_k(x_i)$). As a result, every path connecting a node in the first column of the Viterbi graph to a node in the final column corresponds to a hidden path $\pi = \pi_1 \dots \pi_n$.

We assign a weight of

$$\text{WEIGHT}_i(l, k) = transition_{\pi_{i-1}, \pi_i} \cdot emission_{\pi_i}(x_i)$$

to the edge connecting $(l, i-1)$ to (k, i) in the Viterbi graph. Furthermore, we define the **product weight** of a path in the Viterbi graph as the product of its edge weights. For a path from the leftmost column to the rightmost column in the Viterbi graph corresponding to the hidden path π, this product weight is equal to the product of $n-1$ terms,

$$\prod_{i=2}^{n} transition_{\pi_{i-1}, \pi_i} \cdot emission_{\pi_i}(x_i) = \prod_{i=1}^{n-1} \text{WEIGHT}_i(l, k).$$

STOP and Think: How does this expression differ from the formula for $\Pr(x, \pi)$ that we derived in the previous section?

The only difference between the above expression and the expression that we obtained for $\Pr(x, \pi)$,

$$\prod_{i=1}^{n} transition_{\pi_{i-1}, \pi_i} \cdot emission_{\pi_i}(x_i),$$

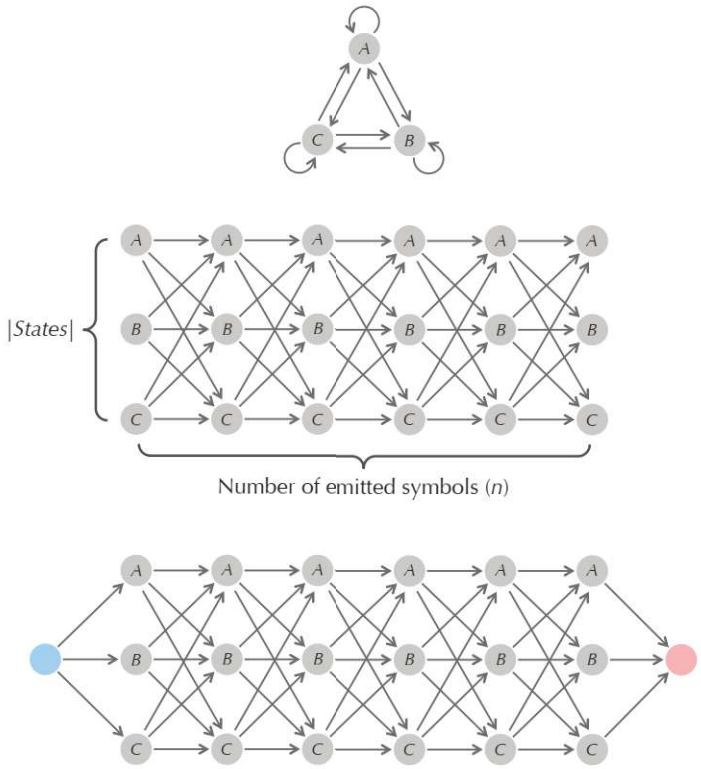

FIGURE 10.7 (Top) The diagram of an HMM with three states (emission/transmission probabilities as well as nodes corresponding to emitted symbols are omitted). (Middle) Given a string of n symbols $x = x_1 \ldots x_n$ emitted by an HMM, Viterbi's Manhattan is a grid with $|States|$ rows and n columns in which each node is connected to every node in the column to its right. The weight of the edge connecting $(l, i-1)$ to (k, i) is $\text{WEIGHT}_i(l,k) = transition_{l,k} \cdot emission_k(x_i)$. Unlike the alignment graphs from Chapter 5, in which the set of valid directions was restricted to south, east, and southeast edges, every node in a column is connected by an edge to every node in the column to its right in the Viterbi graph. (Bottom) The Viterbi graph with additional source node (blue) and sink node (red). A path of largest product weight connecting the source to the sink corresponds to an optimal hidden path solving the Decoding Problem.

is the single factor $transition_{\pi_0,\pi_1} \cdot emission_{\pi_1}(x_1)$, which corresponds to transitioning from the initial state π_0 to π_1 and emitting the first symbol. To model the initial state, we will add a source node *source* to the Viterbi graph and then connect *source* to each node $(k, 1)$ in the first column with an edge of weight $\text{WEIGHT}_0(source, k) = transition_{\pi_0,k} \cdot emission_k(x_1)$. We will also assume that the HMM has another silent

terminal state that the HMM enters when it has finished emitting symbols. To model the terminal state, we add a sink node *sink* to the Viterbi graph and connect every node in the last column to *sink* with an edge of weight 1 (Figure 10.7 (bottom)).

Every hidden path π in the HMM now corresponds to a path from *source* to *sink* in the Viterbi graph with product weight $\Pr(x, \pi)$. Therefore, the Decoding Problem reduces to finding a path in the Viterbi graph of largest product weight over all paths connecting *source* to *sink*.

EXERCISE BREAK: Find the maximum product weight path in the Viterbi graph for the crooked dealer HMM when x= "HHTT".

The Viterbi algorithm

We will apply a dynamic programming algorithm to solve the Decoding Problem. First, define $s_{k,i}$ as the product weight of an optimal path (i.e., a path with maximum product weight) from *source* to the node (k, i). The **Viterbi algorithm** relies on the fact that the first $i-1$ edges of an optimal path from *source* to (k, i) must form an optimal path from *source* to $(l, i-1)$ for some (unknown) state l. This observation yields the following recurrence:

$$\begin{aligned} s_{k,i} &= \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot (\text{weight of edge between nodes} (l, i-1) \text{ and } (k,i)) \right\} \\ &= \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot \text{WEIGHT}_i(l,k) \right\} \\ &= \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot \mathit{transition}_{\pi_{i-1},\pi_i} \cdot \mathit{emission}_{\pi_i}(x_i) \right\} \end{aligned}$$

Since *source* is connected to every node in the first column of the Viterbi graph,

$$\begin{aligned} s_{k,1} &= s_{source} \cdot (\text{weight of edge between } source \text{ and } (k,1)) \\ &= s_{source} \cdot \text{WEIGHT}_0(source, k) \\ &= s_{source} \cdot \mathit{transition}_{source,k} \cdot \mathit{emission}_k(x_1) \end{aligned}$$

In order to initialize this recurrence, we set s_{source} equal to 1. We can now compute the maximum product weight over all paths from *source* to *sink* as

$$s_{sink} = \max_{\text{all states } l} s_{l,n} .$$

STOP and Think: How can we adapt our algorithm for finding a longest path in a DAG to find a path with maximum product weight?

How fast is the Viterbi algorithm?

We can interpret the Decoding Problem as yet another instance of the Longest Path in a DAG Problem from Chapter 5 because the path π maximizing the product weight $\prod_{i=1}^{n} \text{WEIGHT}_i(\pi_{i-1}, \pi_i)$ also maximizes the logarithm of this product, which is equal to $\sum_{i=1}^{n} \log\left(\text{WEIGHT}_i(\pi_{i-1})\right)$. Thus, we can substitute the weights of all edges in the Viterbi graph by their logarithms. Finding a longest path (i.e. a path maximizing the *sum* of edge weights) in the resulting graph will correspond to a path of maximum *product* weight in the original Viterbi graph. For this reason, the runtime of the Viterbi algorithm, which you are now ready to implement, is linear in the number of edges in the Viterbi graph. The following exercise shows that the number of these edges is $\mathcal{O}(|States|^2 \cdot n)$, where n is the number of emitted symbols.

EXERCISE BREAK: Show that the number of edges in the Viterbi graph of an HMM emitting a string of length n is $|States|^2 \cdot (n-1) + 2 \cdot |States|$.

EXERCISE BREAK: Apply your solution for the Decoding Problem to find CG-islands in the first million nucleotides from the human X chromosome. To help you design an HMM for this application, you may assume that transitions from CG-islands to non-CG-islands are rare, occurring with probability 0.001, and that transitions from non-CG-islands to CG-islands are even more rare, occurring with probability 0.0001. How many CG-islands do you find?

In practice, many HMMs have **forbidden transitions** between some states. For such transitions, we can safely remove the corresponding edges from the HMM diagram (Figure 10.8 (left)). This operation results in a sparser Viterbi graph (Figure 10.8 (right)), which reduces the runtime of the Viterbi algorithm, since the runtime of the algorithm for finding the longest path in a DAG is linear in the number of edges in the DAG.

EXERCISE BREAK: Let *Edges* denote the set of edges in the diagram of an HMM that may have some forbidden transitions. Prove that the number of edges in the Viterbi graph for this HMM is $|Edges| \cdot (n-1) + 2 \cdot |States|$.

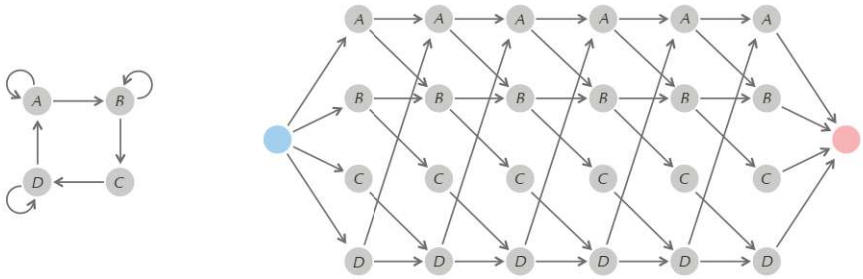

FIGURE 10.8 (Left) An HMM diagram for an HMM that has four states with some forbidden transitions, such as from A to D and from C to itself. Edges corresponding to forbidden transitions between states are not included in the HMM diagram. (Right) The Viterbi graph for this HMM emitting a string of length 6.

Finding the Most Likely Outcome of an HMM

Dynamic programming allows us to answer questions about HMMs extending beyond the most likely hidden path. For example, we have already computed the probability $\Pr(\pi)$ of a hidden path π. But what about computing $\Pr(x)$, the probability that the HMM will emit a string x?

EXERCISE BREAK: Which outcome is more likely in the crooked casino: "HHTT" or "HTHT"? How would you find the most likely sequence of four coin flips?

Outcome Likelihood Problem:
Find the probability that an HMM emits a given string.

Input: A string $x = x_1 \dots x_n$ emitted by an HMM (Σ, *States*, *Transition*, *Emission*).

Output: The probability $\Pr(x)$ that the HMM emits x.

STOP and Think: To solve the Outcome Likelihood Problem, you can make a slight change to the Viterbi recurrence $s_{k,i} = \max_{\text{all states } l}\{s_{l,i-1} \cdot \text{WEIGHT}_i(l,k)\}$. What is the change?

We have already observed that $\Pr(x)$ is equal to the sum of $\Pr(x, \pi)$ over all hidden paths π. However, the number of paths through the Viterbi graph is exponential in the length of the emitted string x, and so we will use dynamic programming to develop a faster approach to compute $\Pr(x)$.

We denote the total product weight of all paths from *source* to node (k, i) in the Viterbi graph as $forward_{k,i}$; note that $forward_{sink}$ is equal to $\Pr(x)$. To compute $forward_{k,i}$, we will divide all paths connecting *source* to (k, i) into $|States|$ subsets, where each subset contains those paths that pass through node $(l, i-1)$ (with product weight $forward_{l,i-1}$) before reaching (k, i) for some l between 1 and $|States|$. Therefore, $forward_{k,i}$ is the sum of $|States|$ terms,

$$\begin{aligned} forward_{k,i} &= \sum_{\text{all states } l} forward_{l,i-1} \cdot (\text{weight of edge connecting } (l, i-1) \text{ and } (k, i)) \\ &= \sum_{\text{all states } l} forward_{l,i-1} \cdot \text{WEIGHT}_i(l, k) \, . \end{aligned}$$

Note that the only difference between this recurrence and the Viterbi recurrence,

$$s_{k,i} = \max_{\text{all states } l} \left\{ s_{l,i-1} \cdot \text{WEIGHT}_i(l, k) \right\} ,$$

is that the maximization in the Viterbi algorithm has changed into a summation symbol. We can now solve the Outcome Likelihood Problem by computing $forward_{sink}$, which is equal to

$$\sum_{\text{all states } k} forward_{k,n} \, .$$

10D

Now that we can compute $\Pr(x)$ for an emitted string x, a natural question is to find the most likely such string. In the crooked dealer example, this corresponds to finding the most likely sequence of flips over all possible sequences of fair and biased coins that the dealer could use.

Most Likely Outcome Problem:
Find a most likely string emitted by an HMM.

Input: An HMM $(\Sigma, States, Transition, Emission)$ and an integer n.
Output: A most likely string $x = x_1 \ldots x_n$ emitted by this HMM, i.e., a string maximizing the probability $\Pr(x)$ that the HMM will emit x.

EXERCISE BREAK: Solve the Most Likely Outcome Problem (Hint: You may need to build a 3-dimensional version of Viterbi's Manhattan).

Profile HMMs for Sequence Alignment

How do HMMs relate to sequence alignment?

You may still be wondering what in the world HMMs have to do with our original problem of aligning sequences using a column-specific score. As we will see, HMMs offer an elegant solution to this problem.

Given a family of related proteins, we can check whether a newly sequenced protein belongs to this family by constructing pairwise alignments between the newly sequenced protein and each member of the family. If one of the resulting alignments scores above some stringent threshold, then we can assume that the new protein belongs to the family. However, this approach may fail to identify distantly related proteins, such as gp120 proteins taken from different HIV isolates, since these proteins may have scores falling below the threshold. If a sequence has weak similarities with many family members, then it most likely belongs to the family.

The problem, then, is to align a new protein to *all* members of the family at once. To do so, we must assume that we have already constructed a multiple alignment of a family of proteins. Fortunately, it will often be obvious that two proteins come from the same family (e.g., if the proteins are taken from closely related species). Accordingly, biologists often start by constructing an alignment of undeniably related proteins, which are typically easy to align even using the simple multiple alignment methods that we covered in Chapter 5.

Figure 10.9 (first panel) shows a 5×10 alignment *Alignment* representing a hypothetical family of proteins. Note that the sixth and seventh columns of this alignment contain many space symbols and likely do not represent meaningful characteristics of the family. Accordingly, biologists often ignore columns for which the fraction of space symbols is greater than or equal to a **column removal threshold** θ. Column removal results in a 5×8 **seed alignment** (Figure 10.9 (second panel)).

Given a seed alignment $Alignment^*$ representing a family of related proteins, our aim is to build an HMM that realistically models the propensities of symbols in $Alignment^*$ represented by the profile matrix $\text{PROFILE}(Alignment^*)$ (Figure 10.9 (third panel)). Rather than thinking about aligning the existing seed alignment to a given string *Text* (representing a new protein), we will instead think about computing the probability

		1	2	3	4	5	6	7	8
Alignment		A	C	D	E	F	A C	A	D	F
		A	F	D	A	–	– –	C	C	F
		A	–	–	E	F	D –	F	D	C
		A	C	A	E	F	– –	A	–	C
		A	D	D	E	F	A A	A	D	F
*Alignment**		A	C	D	E	F		A	D	F
		A	F	D	A	–		C	C	F
		A	–	–	E	F		F	D	C
		A	C	A	E	F		A	–	C
		A	D	D	E	F		A	D	F
PROFILE(*Alignment**)	A	1	0	0	1/5	0	3/5	0	0
	C	0	2/4	0	0	0	1/5	1/4	2/5
	D	0	1/4	3/4	0	0	0	3/4	0
	E	0	0	0	4/5	0	0	0	0
	F	0	1/4	0	0	1	1/5	0	3/5

FIGURE 10.9 A 5×10 multiple alignment *Alignment* (first panel), its 5×8 seed alignment *Alignment** (second panel), the profile matrix PROFILE(*Alignment**) of the seed alignment (third panel), and the diagram of a simple HMM that models this profile (fourth panel). The seed alignment is obtained from the original alignment by ignoring poorly conserved columns (shaded gray); in this case, we ignore columns for which the fraction of space symbols is greater than or equal to the threshold $\theta = 0.35$. To better illustrate the relationship between the alignment and its seed alignment, we have separated the first five columns in the seed alignment from its last three columns and numbered these columns above the original alignment. The match states MATCH(i) are abbreviated as M_i. The HMM only has one possible path; it is initially in state MATCH(1), the transition probability from state MATCH(i) to state MATCH($i+1$) is equal to 1 for all i, and all other transitions are forbidden. Emission probabilities are equal to frequencies in the profile, e.g., emission probabilities for M_2 are 0 for A, 2/4 for C, 1/4 for D, 0 for E, and 1/4 for F.

that the HMM emits *Text*. If the HMM is designed well, then the more similar *Text* is to the strings in *Alignment**, the more likely it will be emitted by the HMM.

We will first construct a simple HMM that treats the columns of *Alignment** as k sequentially linked states called **match states** (Figure 10.9 (fourth panel)), denoted

MATCH(1), ..., MATCH(k). When the HMM enters state MATCH(i), it emits symbol x_i with probability equal to the frequency of this symbol in the i-th column of PROFILE(*Alignment**). The HMM then moves into state MATCH($i+1$) with transition probability equal to 1.

The **similarity score** between *Alignment** and *Text* is the probability Pr(*Text*) that the HMM for *Alignment** emits *Text*. This score is equal to the product of frequencies in PROFILE(*Alignment**) corresponding to each symbol of *Text*. For example, the probability that the HMM in Figure 10.9 emits ADDAFFDF is

$$1 \cdot \tfrac{1}{4} \cdot \tfrac{3}{4} \cdot \tfrac{1}{5} \cdot 1 \cdot \tfrac{1}{5} \cdot \tfrac{3}{4} \cdot \tfrac{3}{5} = 0.003375.$$

STOP and Think: What are the limitations of the HMM in Figure 10.9?

The HMM that we have proposed does score each column in Figure 10.9 differently, and to a degree, the more similar *Text* is to *Alignment**, the higher its similarity score. However, this HMM it is not in keeping with the spirit of HMMs because it has only one hidden path. Furthermore, it offers a simplistic view of multiple alignment because it does not account for insertions and deletions. Finally, it can only "align" *Text* against *Alignment** if the length of *Text* is exactly equal to the number of columns in *Alignment** (Figure 10.10). Yet we will use this limited HMM as the foundation of a more powerful HMM.

	A	C	D	E	F	A	D	F
	A	F	D	A	-	C	C	F
Alignment*	A	-	-	E	F	F	D	C
	A	C	A	E	F	A	-	C
	A	D	D	E	F	A	D	F
Text	A	D	D	A	F	F	D	F
emission probability	1	1/4	3/4	1/5	1	1/5	3/4	3/5

FIGURE 10.10 Aligning *Text* = ADDAFFDF against the seed alignment *Alignment** represented as a simple HMM in Figure 10.9. This HMM is limited because we are not able to align a string of length other than 8. Indeed, there is no way to add space symbols to *Text* or to add symbols of *Text* "between" columns of *Alignment**.

Building a profile HMM

The improved HMM that we propose is called a **profile HMM**. Given a multiple alignment *Alignment* and a column removal threshold θ used to obtain a seed alignment *Alignment**, we will denote this profile HMM as HMM(*Alignment*, θ). Because the profile HMM will be constructed from the seed alignment, we will also informally refer to it as HMM(*Alignment**). Given a string *Text* to align against the existing seed alignment, our goal is to find an optimal hidden path in the profile HMM by solving the Decoding Problem for this HMM and the emitted string *Text*.

As with our first attempt at an HMM from Figure 10.9, the profile HMM will still traverse its states in an order consistent with traversing the columns of *Alignment** from left to right. However, to align strings *Text* of varying lengths, we will need more states in addition to the k match states.

First, we add $k+1$ **insertion states**, denoted INSERTION$(0), \ldots,$ INSERTION(k) (Figure 10.11). Entering INSERTION(i) allows the profile HMM to emit an additional symbol after visiting the i-th column of PROFILE(*Alignment**) and before entering the $(i+1)$-th column. Thus, we will connect MATCH(i) to INSERTION(i) and INSERTION(i) to MATCH$(i+1)$. Furthermore, to allow for multiple inserted symbols between columns of PROFILE(*Alignment**), we will connect INSERTION(i) to itself.

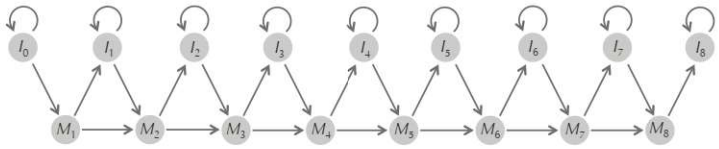

FIGURE 10.11 An HMM diagram for the seed alignment in Figure 10.9 with match and insertion states, abbreviated as M and I, respectively. The states I_0 and I_8 model insertions of symbols occurring before the beginning and end of *Alignment**, respectively.

STOP and Think: Can we use the HMM in Figure 10.11 to align a string *Text* of length less than 8?

After modeling insertions of new symbols in PROFILE(*Alignment**), we should also model "deletions" allowing the profile HMM to skip columns of PROFILE(*Alignment**). One way of modeling these deletions is to add edges connecting every state in the profile HMM to every state on its right (Figure 10.12).

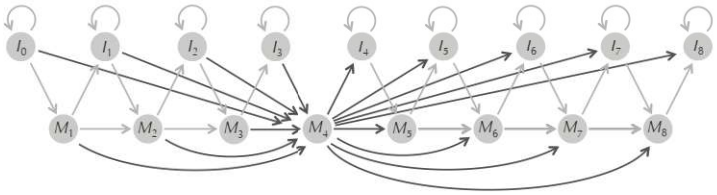

FIGURE 10.12 By adding edges connecting each state in the profile HMM from Figure 10.11 to every state on its right, we can skip columns of *Alignment* when comparing *Text* against this alignment. The above HMM diagram highlights all edges leading into and out of MATCH(4).

STOP and Think: Revisit the Exercise Break on page 195 to recall that the running time of the Viterbi algorithm is proportional to the number of edges (with nonzero transition probabilities) in the HMM diagram. How many edges will the diagram in Figure 10.12 have? How can we reduce the number of edges in the HMM diagram?

Instead of skipping states as in Figure 10.12, we can reduce the number of edges in the HMM diagram by introducing k silent **deletion states** DELETION$(1), \ldots,$ DELETION(k) (Figure 10.13). For example, instead of jumping from MATCH$(i-1)$ to MATCH$(i+1)$, we can make the transition MATCH$(i-1) \rightarrow$ DELETION$(i) \rightarrow$ MATCH$(i+1)$. Entering DELETION(i) allows the HMM to skip over a column of the alignment without emitting a symbol.

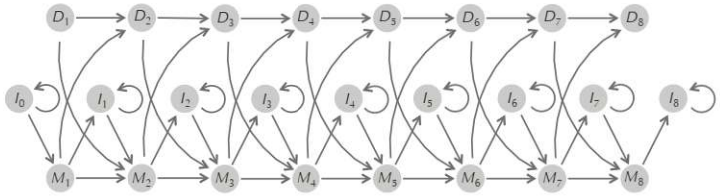

FIGURE 10.13 Adding silent deletion states (abbreviated as D_i) to the profile HMM diagram.

STOP and Think: Is the HMM in Figure 10.13 now adequate, or is there anything else that we have forgotten to add?

We can now transition back and forth between match states and insertion states, as well as back and forth between match states and deletion states, but we cannot transition between insertion states and deletion states. The profile HMM diagram should therefore include edges connecting INSERTION(i) to DELETION$(i+1)$ and connecting DELETION(i) to INSERTION(i) for each i. As a result, the profile HMM can move from any match/insertion state to any other match/insertion state on its right by sidetracking through intermediate deletion states. We obtain the complete profile HMM diagram shown in Figure 10.14 after connecting the initial state (S) to the first match/insertion/deletion states and connecting the final match/insertion/deletion states to the terminal state (E).

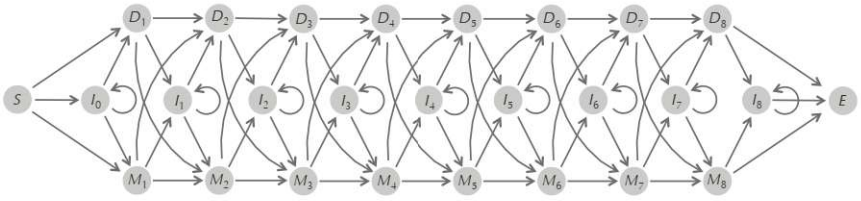

FIGURE 10.14 Adding transitions from insertion states to deletion states and vice-versa completes the profile HMM diagram for the profile matrix from Figure 10.9. Silent initial and terminal states are shown by S and E, respectively.

STOP and Think: Consider the following questions.

- How many edges does the HMM diagram in Figure 10.14 have? How does this compare to the HMM diagram in Figure 10.12?
- What does the Viterbi graph of the profile HMM in Figure 10.14 look like? How many nodes and edges does it have?

Transition and emission probabilities of a profile HMM

In Figure 10.15, we return to the multiple alignment *Alignment* from Figure 10.9 and represent each of the five colored rows of this alignment as a path in the diagram of HMM$(Alignment^*)$. Symbols in the seed alignment $Alignment^*$ (non-shaded columns) correspond to either a match state (non-space symbols) or a deletion state (space symbols). As for symbols not present in the seed alignment (shaded columns), space symbols are ignored, and non-space symbols are emitted from insertion states.

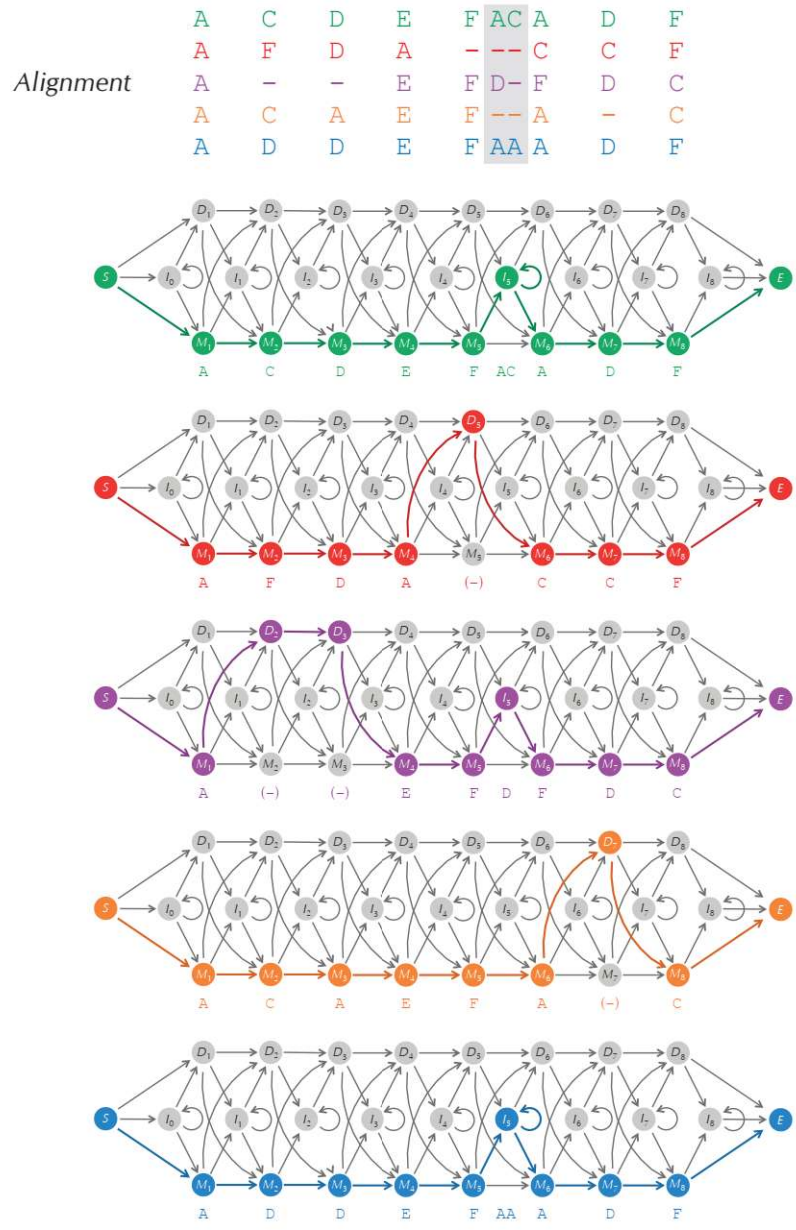

FIGURE 10.15 Five paths through the profile HMM corresponding to the five rows in the alignment from Figure 10.9. Space symbols below an HMM diagram correspond to deletion states and are shown in parentheses to indicate that they are not emitted by the HMM.

STOP and Think: How would you assign transition and emission probabilities for the profile HMM of the alignment in Figure 10.15?

To assign the transition probability $transition_{l,k}$, we simply take the frequency of transitions from state l to state k made by these colored paths with respect to all paths that visited state l. For example, in Figure 10.15, four of the colored paths visit MATCH(5). Three of these paths then transition to INSERTION(5), and one transitions to MATCH(6). Thus, we set the following transition probabilities leaving MATCH(5):

$$transition_{\text{MATCH(5)},\text{INSERTION(5)}} = 3/4$$
$$transition_{\text{MATCH(5)},\text{MATCH(6)}} = 1/4$$
$$transition_{\text{MATCH(5)},\text{DELETION(6)}} = 0$$

We can define the transition probabilities from the initial state analogously. For the multiple alignment in Figure 10.15, we enter MATCH(1) with probability 1; for a general profile HMM, the only other states we could enter from the initial state are INSERTION(0) and DELETION(1). The complete matrix of transmission probabilities is shown in Figure 10.16.

STOP and Think: Due to the small number of strings in the alignment from Figure 10.15, many of the transition probabilities in the gray cells in Figure 10.16 are equal to zero. What are the possible negative consequences of these zeroes, and how would you address these consequences?

To assign the emission probability $emission_k(b)$, we divide the number of times that symbol b was emitted from state k by the total number of symbols emitted from state k. For example, in Figure 10.15, there are three occurrences of A, one occurrence of C, and one occurrence of D emitted from the state INSERTION(5). Also, there are two occurrences of C, one occurrence of D, and one occurrence of F emitted from MATCH(2). We can therefore infer the following emission probabilities for these two states:

$$emission_{\text{INSERTION(5)}}(\text{A}) = 3/5 \qquad emission_{\text{MATCH(2)}}(\text{A}) = 0$$
$$emission_{\text{INSERTION(5)}}(\text{C}) = 1/5 \qquad emission_{\text{MATCH(2)}}(\text{C}) = 2/4$$
$$emission_{\text{INSERTION(5)}}(\text{D}) = 1/5 \qquad emission_{\text{MATCH(2)}}(\text{D}) = 1/4$$
$$emission_{\text{INSERTION(5)}}(\text{E}) = 0 \qquad emission_{\text{MATCH(2)}}(\text{E}) = 0$$
$$emission_{\text{INSERTION(5)}}(\text{F}) = 0 \qquad emission_{\text{MATCH(2)}}(\text{F}) = 1/4$$

	S	I_0	M_1	D_1	I_1	M_2	D_2	I_2	M_3	D_3	I_3	M_4	D_4	I_4	M_5	D_5	I_5	M_6	D_6	I_6	M_7	D_7	I_7	M_8	D_8	I_8	E
S			1																								
I_0																											
M_1						.8	.2																				
D_1																											
I_1																											
M_2									1																		
D_2										1																	
I_2																											
M_3												1															
D_3												1															
I_3																											
M_4															.8	.2											
D_4																											
I_4																											
M_5																	.75	.25									
D_5																		1									
I_5																	.4	.6									
M_6																					.8	.2					
D_6																											
I_6																											
M_7																								1			
D_7																											
I_7																								1			
M_8																											1
D_8																											
I_8																											
E																											

FIGURE 10.16 The 27×27 matrix of transition probabilities for HMM(*Alignment*, 0.35), where *Alignment* is the multiple alignment from Figure 10.9. All values in empty cells are equal to zero. Cells shaded gray correspond to edges in the HMM diagram from Figure 10.14; cells shaded white correspond to forbidden transitions.

EXERCISE BREAK: Construct the 27×20 emission probability matrix for HMM(*Alignment*, 0.35) derived from *Alignment* in Figure 10.9.

You are now ready to construct the profile HMM for an arbitrary multiple alignment.

Profile HMM Problem:
Construct a profile HMM from a multiple alignment.

Input: A multiple alignment *Alignment* and a threshold θ.
Output: HMM(*Alignment*, θ).

EXERCISE BREAK: Construct a profile HMM for the HIV sequences shown in Figure 10.1 with $\theta = 0.35$.

Classifying proteins with profile HMMs

Aligning a protein against a profile HMM

Given a protein family, represented by *Alignment*, we can now return to the problem of deciding whether a newly sequenced protein, represented by *Text*, belongs to the family. We first form HMM(*Alignment*, θ) for some parameter θ. As shown in Figure 10.17, a hidden path through HMM(*Alignment*, θ) corresponds to a sequence of match, insertion, and deletion states for aligning *Text* against *Alignment*.

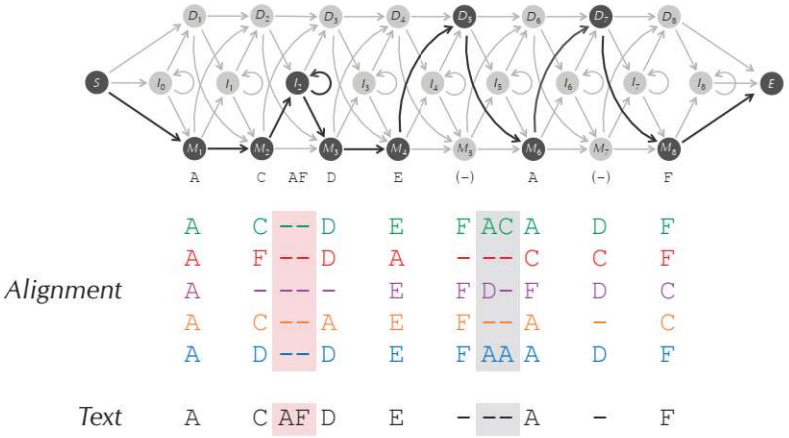

FIGURE 10.17 (Top) A path through HMM(*Alignment*, 0.35) for the multiple alignment from Figure 10.9 and the emitted string *Text* = ACAFDEAF. (Bottom) The emitted symbols correspond to aligning *Text* against *Alignment*. Specifically, the first two symbols are emitted from two match states and belong in the first two positions of the alignment. The next two symbols are emitted from an insertion state and belong in columns of their own (shown in pink). The space symbols in the seventh and eleventh columns above correspond to deletion states; these symbols are not emitted by the HMM. The space symbols in the gray columns do not correspond to any states and are passed over. The non-shaded columns form an augmented 6×8 seed alignment for comparison against newly sequenced proteins.

To find the "best" alignment of *Text* against *Alignment*, we simply need to apply the Viterbi algorithm to find an optimal hidden path in HMM(*Alignment*, θ). If the product weight of this optimal hidden path exceeds a predetermined threshold, then we may conclude that *Text* belongs to the protein family, in which case we augment the existing seed alignment with an additional row corresponding to *Text*. In this way, we can recruit more and more distant family members to a seed alignment, adding these new proteins to the growing multiple alignment, and thus making the resulting profile HMM more and more suitable for analyzing the protein family of interest.

STOP and Think: If the product weight for a new protein exceeds a threshold for more than one protein family, how would you classify this protein?

Profile HMMs have finally helped us achieve our original goal of scoring different columns of a multiple alignment differently based on the frequency of symbols in each column. For example, say that the seventh column of *Alignment** contains more occurrences of A than C, and the ninth column of *Alignment** contains more occurrences of C than A. A hidden path passing through MATCH(7) would be rewarded more for emitting A than C, whereas a hidden path passing through MATCH(9) would be rewarded more for emitting C than A.

The return of pseudocounts

The majority of transition probabilities in the gray cells of Figure 10.16 are equal to zero. (The same is true of emission probabilities.) These zeroes may cause problems; for example, the path in Figure 10.17 seems perfectly reasonable for *Text* = ACAFDEAF, and yet $\Pr(x, \pi)$ is equal to zero because the transition probability from MATCH(2) to INSERTION(2) for this profile HMM is zero.

As in Chapter 2, we will introduce pseudocounts by adding a small value σ to entries in the transition matrix that correspond to edges of the HMM diagram in Figure 10.14 (i.e., only the gray elements of Figure 10.16). Note that white cells in Figure 10.16, corresponding to forbidden transitions, are not affected by pseudocounts. The resulting matrix will then need to be normalized so that the elements in each row sum to 1.

EXERCISE BREAK: Compute the normalized matrix for the matrix in Figure 10.16 after adding the pseudocount $\sigma = 0.01$.

We will also add pseudocounts to the matrix of emission probabilities and normalize the resulting matrix. We refer to the profile HMM defined by the resulting normalized

matrices of transition and emission probabilities as HMM$(Alignment, \theta, \sigma)$.

Profile HMM with Pseudocounts Problem:
Construct a profile HMM with pseudocounts from a multiple alignment.

Input: A multiple alignment *Alignment*, a threshold value θ, and a pseudocount value σ.
Output: HMM$(Alignment, \theta, \sigma)$.

STOP and Think: Since the HMM diagram in Figure 10.17 has 25 nodes — not including the start and end states — the Viterbi graph for the string emitted in this figure has 25 rows. How many columns does this Viterbi graph have?

We are now ready to align a string *Text* to a multiple alignment by constructing the Viterbi graph for this string (Figure 10.18) and solving the Decoding Problem to find the most likely hidden path.

STOP and Think: Find paths through the Viterbi graph corresponding to the bottom four hidden paths in Figure 10.15. What happens?

The troublesome silent states

If you reached this point without any questions about Figure 10.18, then we have successfully concealed from you that solving the Decoding Problem for HMMs with silent states is not as simple as it may appear: the graph in Figure 10.18 is not a Viterbi graph! To see why not, consider the path in Figure 10.19, which emits the same string as Figure 10.18 but passes through one fewer silent deletion state, thus reducing the number of columns by one. But the Viterbi graph is not allowed to change depending on the hidden path π, since we know nothing about the hidden path in advance! Instead, the number of columns in the Viterbi graph must equal the length of the *emitted string*, a condition that is violated in both Figure 10.18 and Figure 10.19.

STOP and Think: How can we modify the notion of the Viterbi graph for HMMs with silent states?

More generally, the Viterbi algorithm does not tolerate silent states other than the initial and terminal states. In other words, this algorithm assumes that node (k, i) in the

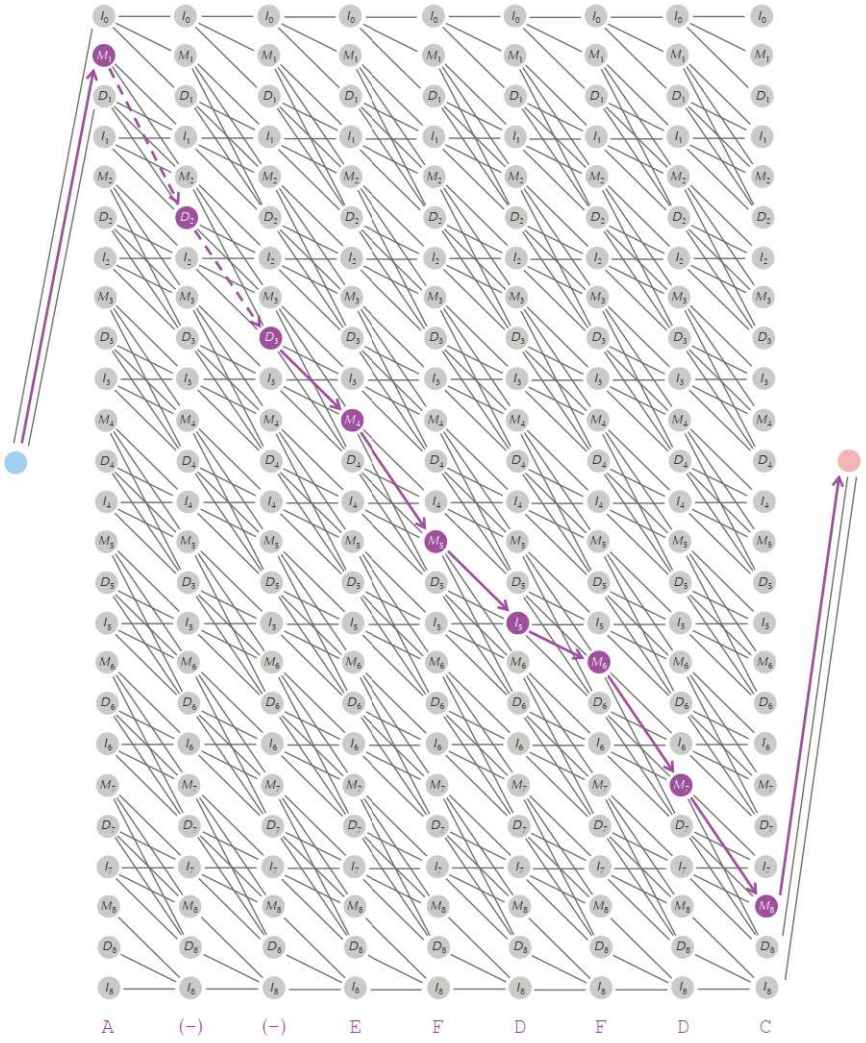

FIGURE 10.18 The Viterbi graph for HMM($Alignment, \theta$) and a path in this graph (shown in purple) corresponding to the hidden path for the emitted string **AEFDFDC** from Figure 10.15. Edges between columns correspond to allowed transitions in the HMM diagram from Figure 10.14 and have an implied rightward orientation. Edges entering nodes corresponding to deletion states are dashed. Emitted symbols are shown beneath each column.

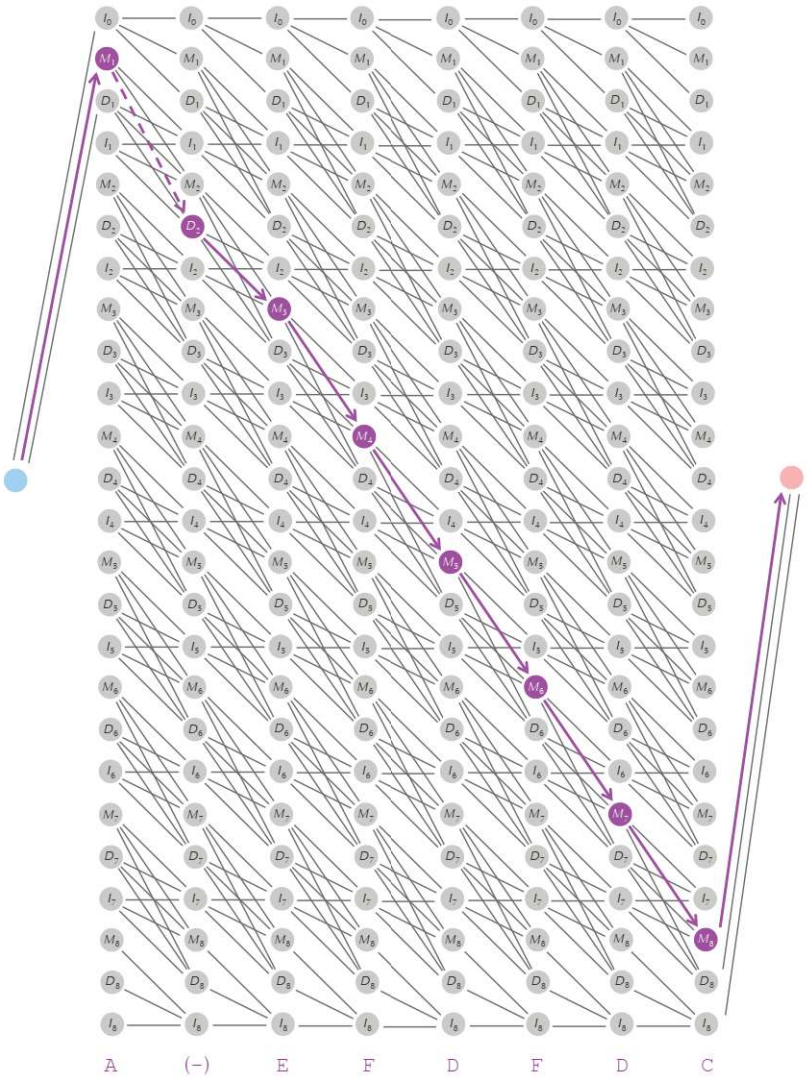

FIGURE 10.19 Another path through another "Viterbi graph" emitting the same string **AEFDFDC** as in Figure 10.18.

Viterbi graph describes the event "the HMM emitted symbol x_i when it was in state k". However, if k is a silent state, then the role of the node (k, i) in the Viterbi graph is poorly defined, as it is unclear how to define the weight of edges entering this node.

Fortunately, we can fix this issue in the case of profile HMMs by defining the Viterbi graph with $|States|$ rows and $|Text|$ columns (Figure 10.20). Every time the HMM moves into a deletion state, rather than crossing over to the next column of the Viterbi graph (as in Figure 10.18 and Figure 10.19), we will move *within* the same column. When the HMM moves into a match or insertion state, we will move to the next column. As a result, every column of the Viterbi graph corresponds to a single emitted symbol, even though a path can pass through more than one state in a given column.

EXERCISE BREAK: Show that the vertical edge connecting (i, l) to (i, k), where k is a deletion state, should be assigned weight equal to $transition_{l,k}$.

STOP and Think: Are there any remaining issues with the graph in Figure 10.20?

There is still a minor flaw with the graph in Figure 10.20. If the HMM moves from the initial state into DELETION(1), then the HMM will move through the first column without emitting a symbol. We will therefore transform the initial state into a column of silent states containing the initial state and all deletion states (Figure 10.21). This way, if the HMM enters DELETION(1) from the initial state, it can move downward through deletion states before transitioning to a match or insertion state in the first column.

You are now ready to use the profile HMM to align a sequence against a seed alignment. The only remaining snare is that when computing $s_{k,i}$ — or, equivalently, $\log(s_{k,i})$ — we must make sure that all incoming scores have been computed. We therefore suggest the top-down, column-by-column topological ordering for the profile HMM shown in Figure 10.22.

Sequence Alignment with Profile HMM Problem:
Align a new sequence to a family of sequences using a profile HMM.

Input: A multiple alignment *Alignment*, a threshold θ, a pseudocount value σ, and a string *Text*.
Output: An optimal hidden path emitting *Text* in HMM$(Alignment, \theta, \sigma)$.

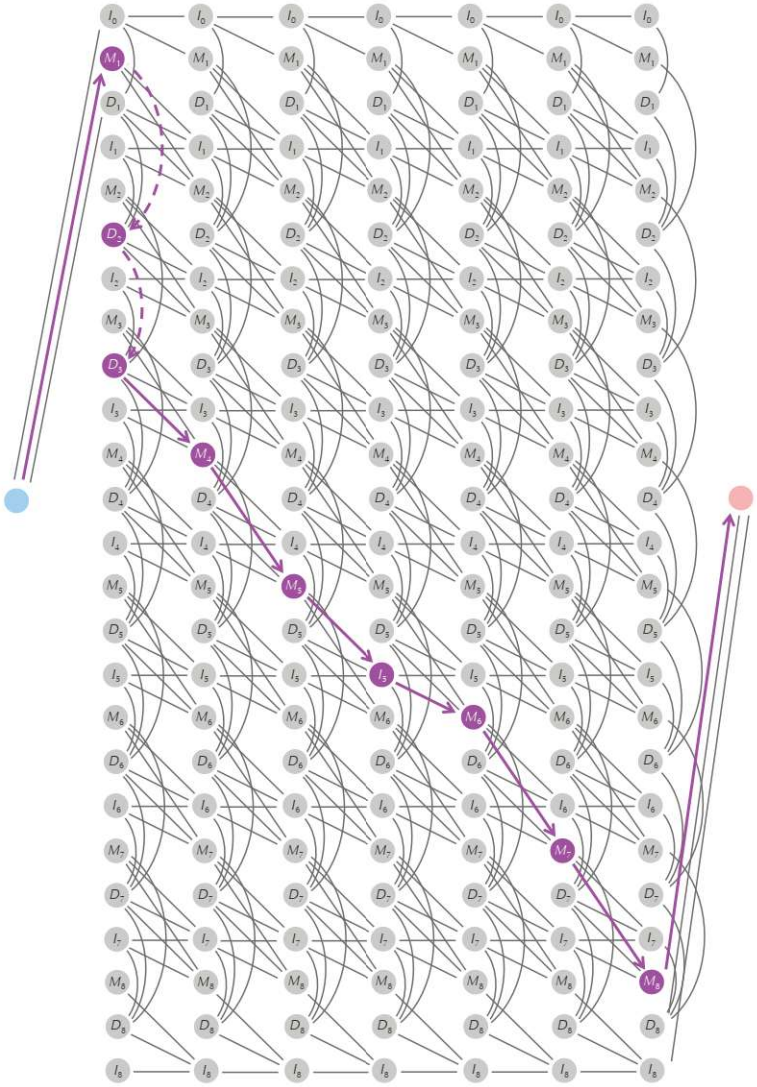

FIGURE 10.20 The Viterbi graph with |*States*| rows and |*Text*| columns for the profile HMM from Figure 10.14 emitting a string *Text* of length 7 so that edges entering deletion states are drawn downward within the same column instead of between columns as in Figure 10.18 and Figure 10.19. The purple path corresponds to the path through the HMM in Figure 10.15 emitting **AEFDFDC**.

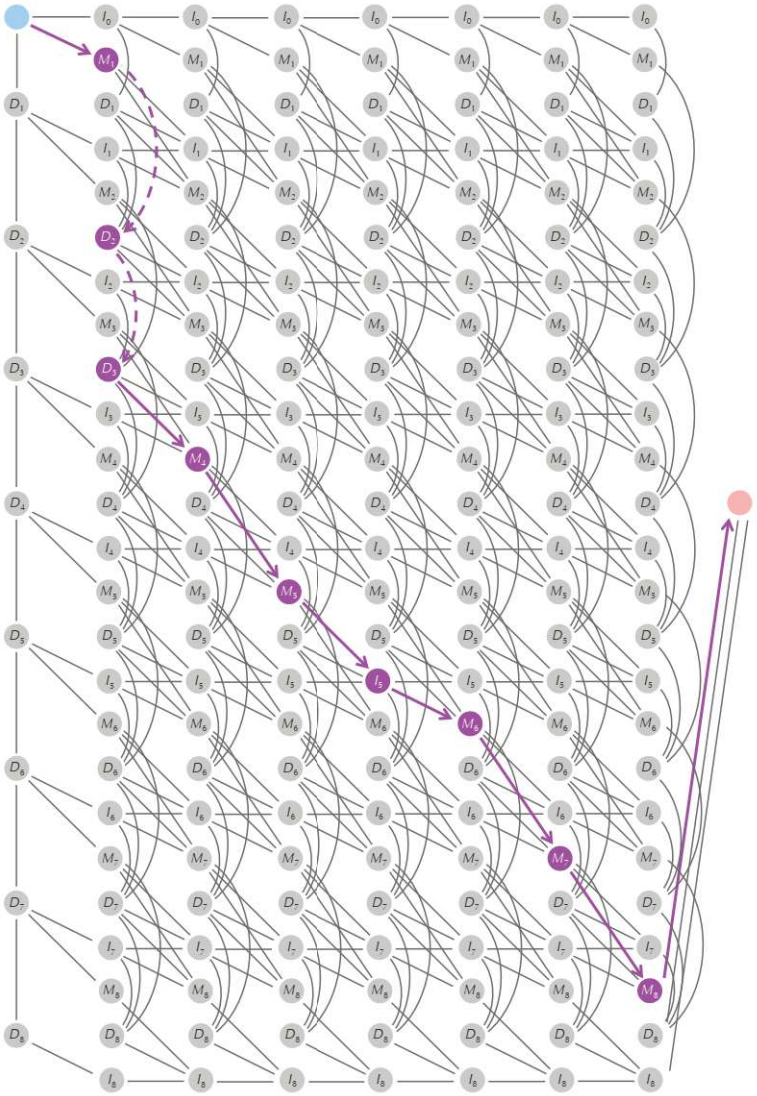

FIGURE 10.21 The final Viterbi graph for the profile HMM in Figure 10.14 emitting a string of length 7. Edges within the same column have a downward orientation; edges between columns have a rightward orientation. Once again, the purple path corresponds to the path through the HMM in Figure 10.15 emitting **AEFDFDC**.

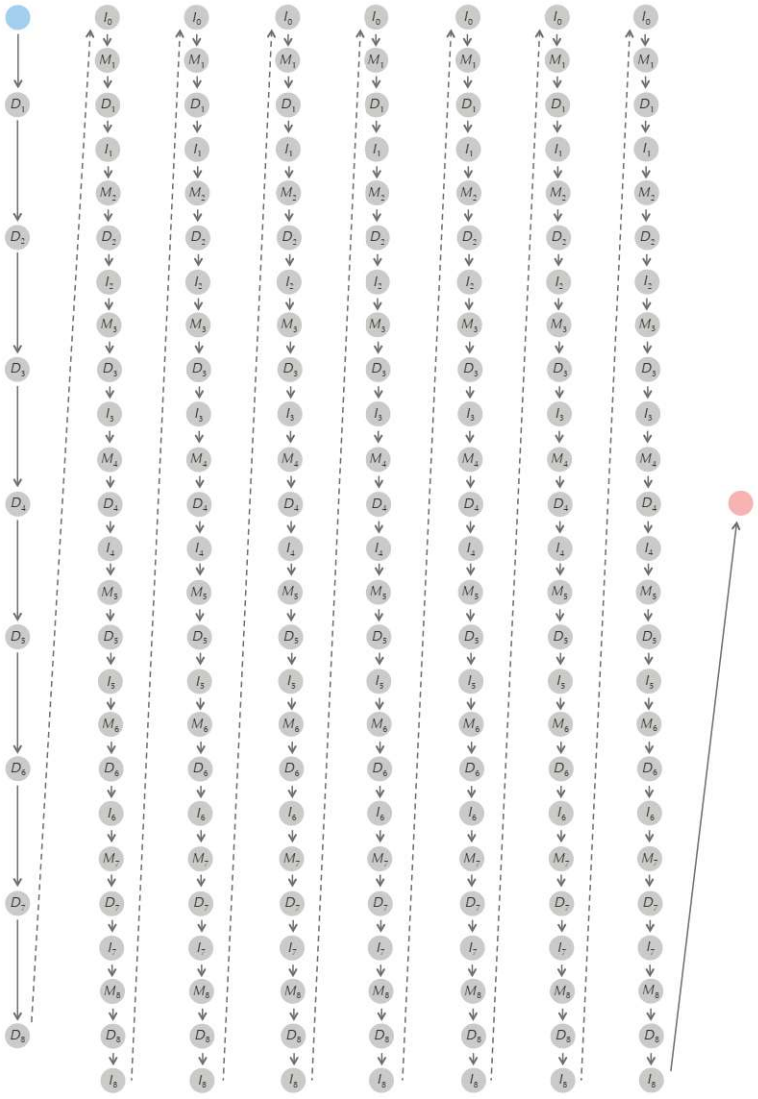

FIGURE 10.22 The topological order of the Viterbi graph from Figure 10.21, which proceeds top-down and column-by-column.

EXERCISE BREAK: Solve the Sequence Alignment with Profile HMM Problem for the profile HMM you constructed for the multiple alignment in Figure 10.1 along with a gp120 protein taken from chimpanzee Simian Immunodeficiency Virus (SIV).

STOP and Think: How would you build the Viterbi graph for an arbitrary HMM with silent states? In which situations will it be impossible to construct a Viterbi graph of an HMM with silent states?

Are profile HMMs really all that useful?

The Viterbi algorithm applies for any HMM, but we will describe how it works for profile HMMs in order to make an important point. Define $s_{\text{MATCH}(j),i}$ as the probability of the most likely hidden path for the prefix $x_1 \ldots x_i$ of x that ends at state $\text{MATCH}(j)$, and define $s_{\text{INSERTION}(j),i}$ and $s_{\text{DELETION}(j),i}$ analogously. Because there are only three edges entering $\text{MATCH}(j)$, the Viterbi recurrence states that

$$s_{\text{MATCH}(j),i} = \max \begin{cases} s_{\text{MATCH}(j-1),i-1} \cdot \text{WEIGHT}_i(\text{MATCH}(j-1), \text{MATCH}(j)) \\ s_{\text{INSERTION}(j-1),i-1} \cdot \text{WEIGHT}_i(\text{INSERTION}(j-1), \text{INSERTION}(j)) \\ s_{\text{DELETION}(j-1),i-1} \cdot \text{WEIGHT}_i(\text{DELETION}(j-1), \text{DELETION}(j)) \end{cases}$$

After taking the logarithm of both sides, the resulting recurrence is very similar to the standard recurrence relation for global pairwise alignment because it is the maximum of three sums:

$$\log\left(s_{\text{MATCH}(j),i}\right) =$$
$$\max \begin{cases} \log\left(s_{\text{MATCH}(j-1),i-1}\right) + \log\left(\text{WEIGHT}_i(\text{MATCH}(j-1), \text{MATCH}(j))\right) \\ \log\left(s_{\text{INSERTION}(j-1),i-1}\right) + \log\left(\text{WEIGHT}_i(\text{INSERTION}(j-1), \text{INSERTION}(j))\right) \\ \log\left(s_{\text{DELETION}(j-1),i-1}\right) + \log\left(\text{WEIGHT}_i(\text{DELETION}(j-1), \text{DELETION}(j))\right) \end{cases}$$

Figure 10.23 shows how a path in a Manhattan-like alignment graph corresponds to a path through the profile HMM. Diagonal edges, vertical edges, and horizontal edges in the Manhattan-like graph correspond to match states, insertion states, and deletion states, respectively.

Figure 10.23 may make it seem that we have wasted your time introducing HMMs, since it appears that a profile HMM is somehow equivalent to pairwise sequence alignment. However, keep in mind that the choice of edges in Figure 10.23 is based on

varying transition and emission probabilities. By deriving individual scoring parameters for each column in the alignment matrix, profile HMMs allow us to capture subtle similarities that can fly under the radar of the simple scoring approaches from Chapter 5.

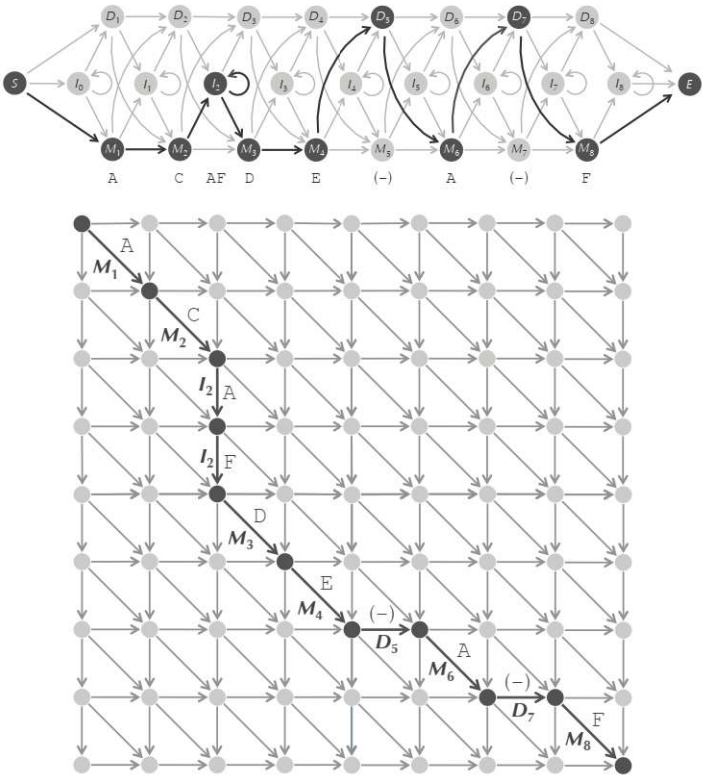

FIGURE 10.23 (Top) The hidden path through the profile HMM in Figure 10.17 (top) emitting ACAFDEAF. (Bottom) The path through a Manhattan-like graph corresponding to this hidden path.

Learning the Parameters of an HMM

Estimating HMM parameters when the hidden path is known

Thus far, our analysis has assumed that we know the parameters of an HMM, i.e., its transition and emission probabilities. We have described a naive — and not necessarily

optimal — heuristic for choosing these parameters for a profile HMM, but it is not clear how to select parameters for an arbitrary HMM.

Indeed, the largest complication when modeling biological problems with an HMM is estimating the HMM's parameters from data. In terms of the crooked casino, imagine that you know that the dealer is using two coins to cheat, but you don't know the bias of the coins or the probability that the dealer switches coins at any given time. Can you infer these parameters just from the sequence of coin flips?

STOP and Think: Say that you observe the sequence of coin flips "HHTHH-HTHHTTTH". What is your best guess for the biases of the two coins and for the probabilities of switching from one coin to the other? Would your best guess change if you knew that the hidden path were $\pi = FFFBBFFFFFBBB$?

We will collectively refer to the matrices *Transition* and *Emission* as *Parameters*. Our goal is to find *Parameters* and π when we are only given the emitted string x. We will work toward this goal by assuming that we are given x as well as either *Parameters* or π, and we must infer the remaining component. If x and *Parameters* are known, then we can find the most likely hidden path π using the Viterbi algorithm. However, we have not yet considered how to estimate *Parameters* if we know x and the hidden path π.

HMM Parameter Estimation Problem:
Find optimal parameters explaining the emitted string and the hidden path of an HMM.

Input: A string $x = x_1 \ldots x_n$ emitted by an HMM with unknown transition and emission probabilities following a known hidden path $\pi = \pi_1 \ldots - \pi_n$.
Output: A transition matrix *Transition* and an emission matrix *Emission* that maximize $\Pr(x, \pi)$ over all possible transition and emission matrices.

If we know both x and π, then we can compute empirical estimates for the transition and emission probabilities using a method similar to one we used for estimating parameters for profile HMMs. If $T_{l,k}$ denotes the number of transitions from state l to state k in the hidden path π, then we can estimate the probability $transition_{l,k}$ by computing the ratio of $T_{l,k}$ to the total number of transitions leaving state l,

$$transition_{l,k} = \frac{T_{l,k}}{\sum_{\text{all states } j} T_{l,j}}.$$

Likewise, if $E_k(b)$ denotes the number of times symbol b is emitted when the hidden path π is in state k, then we can estimate the probability $emission_k(b)$ as the ratio of $E_k(b)$ to the total number of emitted symbols from state k,

$$emission_k(b) = \frac{E_k(b)}{\sum_{\text{all symbols } c \text{ in the alphabet}} E_k(c)}.$$

It turns out that the above two formulas for computing *Transition* and *Emission* result in parameters solving the HMM Parameter Estimation Problem.

Viterbi learning

If we know x and *Parameters*, then we can construct the most likely path π by applying the Viterbi algorithm to solve the Decoding Problem:

$$(x, ?, Parameters) \rightarrow \pi$$

On the other hand, if we know x and π, then reconstructing *Parameters* amounts to solving the HMM Parameter Estimation Problem:

$$(x, \pi, ?) \rightarrow Parameters$$

STOP and Think: What do the expressions $(x, \pi, ?) \rightarrow Parameters$ and $(x, ?, Parameters) \rightarrow \pi$ remind you of?

HMM Parameter Learning Problem:
Estimate the parameters of an HMM explaining an emitted string.

Input: A string $x = x_1 \ldots x_n$ emitted by an HMM with unknown transition and emission probabilities.
Output: A transition matrix *Transition* and an emission matrix *Emission* that maximize $\Pr(x, \pi)$ over all possible transition and emission matrices and over all hidden paths π.

Unfortunately, the HMM Parameter Learning Problem is intractable, and so we will instead develop a heuristic that is analogous to the Lloyd algorithm for k-means clustering from Chapter 8. In that algorithm, illustrated in Figure 8.12, we iterated two steps, "From Centers to Clusters",

$$(Data, ?, Centers) \rightarrow HiddenVector,$$

and "From Clusters to Centers",

$$(Data, HiddenVector, ?) \rightarrow Centers.$$

As for HMM parameter estimation, we begin with an initial random guess for *Parameters*. Then, we use the Viterbi algorithm to find the optimal hidden path π:

$$(x, ?, Parameters) \rightarrow \pi$$

Once we know π, we will question our original choice of *Parameters* and apply our solution to the HMM Parameter Estimation Problem to update *Parameters* based on x and π:

$$(x, \pi, ?) \rightarrow Parameters'$$

We then iterate over these two steps, hoping that the estimated parameters are getting closer and closer to the parameters solving the HMM Parameter Learning Problem:

$$\begin{aligned}(x, ?, Parameters) &\rightarrow (x, \pi, Parameters) &&\rightarrow (x, \pi, ?)\\ &\rightarrow (x, \pi, Parameters') &&\rightarrow (x, ?, Parameters')\\ &\rightarrow (x, \pi', Parameters') &&\rightarrow (x, \pi', ?)\\ &\rightarrow (x, \pi', Parameters'') &&\rightarrow \ldots\end{aligned}$$

This approach to learning the HMM's parameters is called **Viterbi learning**.

STOP and Think: Can $\Pr(x, \pi)$ decrease during Viterbi learning? When would you decide to stop the Viterbi learning algorithm?

Note that we have not specified how Viterbi learning should terminate. In practice, there are various stopping rules to control its running time. For example, the algorithm can be stopped if the number of iterations exceeds a predefined threshold or if $\Pr(x, \pi)$ changes very little from one iteration to another.

Also, because Viterbi learning is dependent on the initial guess for *Parameters*, it may become stuck in a local optimum. Like other heuristics, it is often run many times, retaining the best choice of *Parameters*.

EXERCISE BREAK: Apply Viterbi learning to learn parameters for an HMM modeling CG-islands as well as for the profile HMM for the gp120 HIV alignment in Figure 10.1.

Soft Decisions in Parameter Estimation

The Soft Decoding Problem

In Chapter 8, we introduced a "soft" clustering algorithm, based on the more general expectation maximization algorithm, that relaxed the Lloyd algorithm's rigid assignment of points to clusters. Analogously, by generating a single optimal hidden path, the Viterbi algorithm provides a rigid "yes" or "no" answer to the question of whether an HMM was in state k at time i. But how certain are we that this was the case?

Returning to the crooked casino analogy once more, say that the i-th coin flip is heads. If this flip occurs in the middle of ten consecutive heads, then you should be relatively confident that the biased coin was used. But what if, of the ten flips surrounding the i-th flip, six are heads and four are tails? In this case, you should be less certain that the biased coin was used.

In the case of an arbitrary HMM, we would like to compute the conditional probability $\Pr(\pi_i = k|x)$ that the HMM was in state k at time i given that it emitted string x.

Soft Decoding Problem:
Find the probability that an HMM was in a particular state at a particular moment given its emitted string.

Input: A string $x = x_1 \ldots x_n$ emitted by an HMM.
Output: The conditional probability $\Pr(\pi_i = k|x)$ that the HMM was in state k at step i given that it emitted x.

The unconditional probability that a hidden path will pass through state k at time i and emit x can be written as the sum

$$\Pr(\pi_i = k, x) = \sum_{\text{all paths } \pi \text{ with } \pi_i = k} \Pr(x, \pi)\,.$$

The conditional probability $\Pr(\pi_i = k|x)$ is equal to the proportion of paths that pass through state k at time i and emit x with respect to all paths emitting x:

$$\begin{aligned}\Pr(\pi_i = k|x) &= \frac{\Pr(\pi_i = k, x)}{\Pr(x)} \\ &= \frac{\sum_{\text{all paths } \pi \text{ with } \pi_i = k} \Pr(x, \pi)}{\sum_{\text{all paths } \pi} \Pr(x, \pi)}\,.\end{aligned}$$

STOP and Think: If the Viterbi algorithm for the crooked casino emits a path $\pi = \pi_1\pi_2\ldots\pi_n$ with $\pi_i = B$, is the dealer more likely to have used a biased coin at step i? Is it possible that $\pi_i = B$ but that $\Pr(\pi_i = B|x)$ is smaller than $\Pr(\pi_i = F|x)$?

The forward-backward algorithm

We note that $\Pr(\pi_i = k, x)$ is equal to the sum of product weights $\Pr(\pi, x)$ of all paths π through the Viterbi graph for x that pass through the node (k, i). As shown in Figure 10.24 (top), we can break each such path into a blue subpath from *source* to (k, i), which we denote π_{blue}, and a (red) subpath from (k, i) to *sink*, which we denote π_{red}. Writing $\text{WEIGHT}(\pi_{\text{blue}})$ and $\text{WEIGHT}(\pi_{\text{red}})$ as the respective product weights of these subpaths yields the recurrence

$$\begin{aligned}\Pr(\pi_i = k, x) &= \sum_{\text{all paths } \pi \text{ with } \pi_i = k} \Pr(x, \pi) \\ &= \sum_{\text{all paths } \pi_{\text{blue}}} \sum_{\text{all paths } \pi_{\text{red}}} \text{WEIGHT}(\pi_{\text{blue}}) \cdot \text{WEIGHT}(\pi_{\text{red}}) \\ &= \sum_{\text{all paths } \pi_{\text{blue}}} \text{WEIGHT}(\pi_{\text{blue}}) \cdot \sum_{\text{all paths } \pi_{\text{red}}} \text{WEIGHT}(\pi_{\text{red}}) .\end{aligned}$$

We have already computed the sum of product weights of all blue subpaths; it is just $forward_{k,i}$, which we encountered when solving the Outcome Likelihood Problem. Now we would like to compute the sum of product weights of all red subpaths, which we denote as $backward_{k,i}$, so that the preceding equation becomes

$$\Pr(\pi_i = k, x) = forward_{k,i} \cdot backward_{k,i} .$$

The name of $backward_{k,i}$ derives from the fact that to compute this value, we can simply *reverse* the directions of all edges in the Viterbi graph (Figure 10.24 (bottom)) and apply the same dynamic programming algorithm used to compute $forward_{k,i}$. Since the reversed edge connecting $(l, i+1)$ to (k, i) has weight $\text{WEIGHT}_i(k, l) = transition_{k,l} \cdot emission_l(x_{i+1})$, we have that

$$backward_{k,i} = \sum_{\text{all states } l} backward_{l,i+1} \cdot \text{WEIGHT}_i(k, l).$$

STOP and Think: How should this recurrence be initialized?

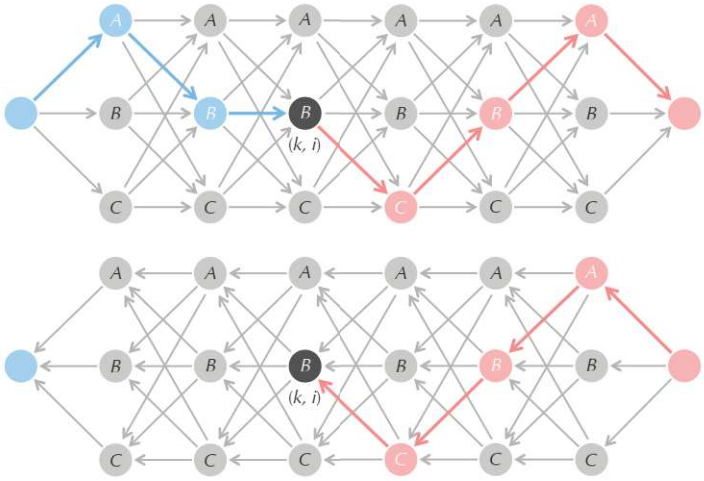

FIGURE 10.24 (Top) Each path from *source* to *sink* passing through the (black) node (k, i) in the Viterbi graph from Figure 10.7 (bottom) can be partitioned into two subpaths, one from *source* to (k, i) (shown in blue) and another from (k, i) to *sink* (shown in red). (Bottom) A "reversed Viterbi graph" in which all edges have been reversed, with a path from sink to (k, i) highlighted in red. The recurrence for $backward_{k,i}$ is based on computing $backward_{l,i+1}$ for every state l.

The resulting dynamic programming approach for computing $\Pr(\pi_i = k, x)$ is called the **forward-backward algorithm**. Combining the forward-backward algorithm with our solution to the Outcome Likelihood Problem for computing $\Pr(x)$ yields that

$$\Pr(\pi_i = k|x) = \frac{\Pr(\pi_i = k, x)}{\Pr(x)} = \frac{forward_{k,i} \cdot backward_{k,i}}{forward(sink)},$$

and so we are ready to solve the Soft Decoding Problem.

EXERCISE BREAK: Consider the following questions.

- For the crooked dealer HMM, compute $\Pr(\pi_i = k|x)$ for x = "THTHH-HTHTTH" and each value of i. How does your answer change if x = "HHHHHHHHHHH"?
- Apply your solution for the Soft Decoding Problem to find CG-islands in the first million nucleotides from the human X chromosome. How does your answer differ from the solution given by the Viterbi algorithm?

We have just seen how to compute the conditional probability $\Pr(\pi_i = k|x)$ that the HMM passes through node (k, i) in the Viterbi graph given that the HMM emits x. But what about the conditional probability $\Pr(\pi_i = l, \pi_{i+1} = k|x)$ that the HMM passes through the edge connecting (l, i) to $(k, i+1)$ given that the HMM emits x? As with the forward-backward algorithm, we can divide every path through the edge in question into a blue path from *source* to this edge and a red path from this edge to *sink* (Figure 10.25).

EXERCISE BREAK: Prove that $\Pr(\pi_i = l, \pi_{i+1} = k|x)$ is equal to $\mathit{forward}_{l,i} \cdot \text{WEIGHT}_i(l, k) \cdot \mathit{backward}_{k,i+1} / \mathit{forward}(\mathit{sink})$.

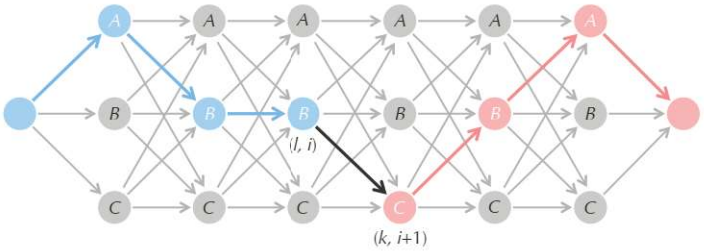

FIGURE 10.25 Each path from *source* to *sink* in the Viterbi graph passing through the (black) edge $(l, i) \rightarrow (k, i+1)$ in the Viterbi graph can be partitioned into two subpaths, one from *source* to (l, i) (shown in blue) and another from $(k, i+1)$ to *sink* (shown in red).

The probabilities $\Pr(\pi_i = k|x)$ can be put into a $|\mathit{States}| \times n$ responsibility matrix Π^*, where $\Pi^*_{k,i}$ corresponds to a *node* in the Viterbi graph and is equal to $\Pr(\pi_i = k|x)$. Figure 10.26 (top) shows the "responsibility" matrix Π^* for the crooked casino.

The probabilities $\Pr(\pi_i = l, \pi_{i+1} = k|x)$ can be put into another $|\mathit{States}| \times |\mathit{States}| \times (n-1)$ responsibility matrix Π^{**}, where $\Pi^{**}_{l,k,i}$ corresponds to an *edge* in the Viterbi graph and is equal to $\Pr(\pi_i = l, \pi_{i+1} = k|x)$ (Figure 10.26 (bottom)). For brevity, we use Π to collectively refer to the matrices Π^* and Π^{**}.

EXERCISE BREAK: What is the complexity of an algorithm computing the matrices Π^* and Π^{**}?

	T	H	T	H	H	H	T	H	T	T	H
F	0.636	0.593	0.600	0.533	0.515	0.544	0.627	0.633	0.692	0.686	0.609
B	0.364	0.407	0.400	0.467	0.485	0.456	0.373	0.367	0.308	0.314	0.391

	1	2	3	4	5	6	7	8	9	10
FF	0.562	0.548	0.507	0.473	0.478	0.523	0.582	0.608	0.643	0.588
FB	0.074	0.045	0.093	0.059	0.037	0.022	0.045	0.025	0.049	0.098
BF	0.031	0.053	0.025	0.042	0.066	0.104	0.051	0.084	0.043	0.022
BB	0.333	0.354	0.374	0.426	0.418	0.351	0.322	0.282	0.265	0.293

FIGURE 10.26 (Top) The responsibility matrix Π^*, where x = "THTHHHTHTTH" and the emission/transition matrices *Parameters* are taken from the crooked dealer HMM in Figure 10.5. $\Pi^*_{k,i}$ is equal to $\Pr(\pi_i = k|x)$. (Bottom) The responsibility matrix Π^{**}, where $\Pi^{**}_{l,k,i} = \Pr(\pi_i = l, \pi_{i+1} = k|x)$ for the same emitted string and emission/transition matrices.

Baum-Welch Learning

The expectation maximization algorithm for parameter estimation, called **Baum-Welch learning**, alternates between two steps. In the E-step, it estimates the responsibility profile Π given the current parameters:

$$(x, ?, Parameters) \rightarrow \Pi$$

Then, in the M-step, it re-estimates the parameters from the responsibility profile:

$$(x, \Pi, ?) \rightarrow Parameters$$

We have already implemented the E-step of the expectation maximization algorithm, but the question remains how to design the M-step.

When we know the hidden path, the previously defined estimators for *Parameters*, reproduced below, define optimal choices for a given hidden path π:

$$transition_{l,k} = \frac{T_{l,k}}{\sum_{\text{all states } j} T_{l,j}} \qquad emission_k(b) = \frac{E_k(b)}{\sum_{\text{all symbols } c \text{ in the alphabet}} E_k(c)}.$$

Here, $T_{l,k}$ is the number of transitions from state l to state k in the hidden path π, and $E_k(b)$ is the number of times symbol b is emitted when the hidden path π is in state k.

EXERCISE BREAK: How would you redefine these estimators when the hidden path is unknown?

To see how to estimate $transition_{l,k}$ and $emission_k(b)$ when the hidden path is unknown, we will compute $T_{l,k}$ and $E_k(b)$ for a known path π in a slightly different way to make the transition from hard to soft choices more apparent. First, define the following binary variables:

$$T^i_{l,k} = \begin{cases} 1 \text{ if } \pi_i = l \text{ and } \pi_{i+1} = k \\ 0 \text{ otherwise} \end{cases} \qquad E^i_k(b) = \begin{cases} 1 \text{ if } \pi_i = k \text{ and } x_i = b \\ 0 \text{ otherwise} \end{cases}$$

With this notation, the formulas computing $T_{l,k}$ and $E_k(b)$ can be rewritten as

$$T_{l,k} = \sum_{i=1}^{n-1} T^i_{l,k} \qquad E_k(b) = \sum_{i=1}^{n} E^i_k(b)$$

When the hidden path is unknown, we will substitute the binary variables $T_{l,k}$ and $E_k(b)$ for new variables $T^i_{l,k}$ and $E^i_k(b)$ that are computed in terms of the conditional probabilities that a hidden path will pass through a given node or edge of the Viterbi graph:

$$T^i_{l,k} = \Pr(\pi_i = l, \pi_{i+1} = k | x) \qquad E^i_k(b) = \Pr(\pi_i = k | x)$$
$$= \Pi^{**}_{l,k,i} \qquad = \Pi^*_{k,i} \text{ if } x_i = b \text{ and 0 otherwise}$$

Armed with these probabilities computed in the previous section, we can compute new estimates for *Parameters* that often perform better in practice than the estimates provided by Viterbi learning:

$$transition_{l,k} = \sum_{i=1}^{n-1} \Pi^{**}_{l,k,i} \qquad emission_k(b) = \sum_{i=1}^{n} \Pi^*_{k,i}$$

STOP and Think: Should we normalize the transition and emission probabilities in the above equations? For example, do the above equations imply that all transition probabilities leaving state l must sum to 1?

EXERCISE BREAK: Use Baum-Welch learning to learn parameters for the HMM modeling CG-islands and for the HIV profile HMM. Compare these parameters with parameters derived by applying Viterbi learning.

The Many Faces of HMMs

Profile HMMs for multiple sequence alignment and HMMs finding CG-islands are just two examples of many applications of HMM in bioinformatics. Furthermore, applications of HMMs to HIV analysis are not limited to profile HMMs but also include analysis of HIV resistance against antiviral drug therapies.

Early in the chapter, we mentioned that patients infected with HIV are treated with a cocktail of several drugs. These drugs attempt to suppress replication of the virus, but HIV often mutates into drug-resistant strains that eventually dominate the virus population in a host, making a drug cocktail progressively ineffective. HIV viruses are often sequenced after drug therapy has failed in order to decide how to reformulate the drug cocktail. Thus, understanding HIV's pathways to drug resistance is important to design an effective cocktail.

Yet modeling HIV resistance pathways is a difficult task. Two mutations that are advantageous for the virus may interact synergistically, causing the double mutation to be fixed more often than we might predict from the frequencies of the individual substitutions. Mutations can also interact antagonistically, resulting in mutants that are less fit than we might predict.

In 2007, Niko Beerenwinkel and Mathias Drton introduced an HMM-based model for HIV evolution and developing drug resistance. However, their HMM is far too complex to explain here. We nevertheless mention it here in order to emphasize the power of HMMs. Even though they may seem like simple machines that flip coins and emit symbols, HMMs can be applied to tackle complex bioinformatics problems ranging from gene prediction to regulatory motif finding.

Epilogue: Nature is a Tinkerer and not an Inventor

The sequence of amino acids in a protein encodes its 3-D structure, which often defines the biological function of a protein. For example, a **zinc finger** is an element of the 3-D structure of **zinc finger proteins** (Figure 10.27). By arranging two cysteines and two histidines close to each other in a zinc finger protein's amino acid sequence, the protein is able to "grab" a zinc ion and fold tightly around it. Zinc fingers are so useful that they are found in thousands of human proteins. Furthermore, zinc finger proteins are used for more than binding zinc, as many of these proteins bind to other metals or even non-metals.

Over 100,000 experimentally determined protein structures are currently known, but many of them represent very similar structures or share segments with very similar

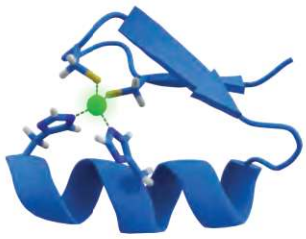

FIGURE 10.27 A zinc ion (shown in green) is held in place by two histidine residues and two cysteine residues within a zinc finger.

structure. A **protein domain** is a conserved part of a protein that can function independently of the rest of the protein. Domains vary in length, but the average domain length is approximately 100 amino acids (zinc finger domains are only 20-30 amino acids long.) Many proteins consist of several domains, and the same domain may appear (with variations) in many proteins.

Nobel Laureate François Jacob famously said in 1977: "Nature is a tinkerer and not an inventor." In accordance with this principle, nature uses domains as building blocks, shuffling them into different arrangements to create **multi-domain proteins**. Most domains once existed as independent proteins; for example, many domains belonging to human multi-domain proteins can be found as single-domain proteins in bacteria. Multi-domain proteins occur naturally when a genome rearrangement creates a new protein-coding sequence containing parts of the coding sequences from two different genes. Association of two domains into a single protein often provides an evolutionary advantage, such as when both domains are enzymes, in which case it may be beneficial for the cell to ensure a fixed one-to-one ratio of the enzymes' activities.

Since proteins are often built from multiple domains with different structures and functions, biologists commonly analyze individual domains instead of entire proteins in order to understand evolutionary relationships. Since sequence similarities between domains with similar structures can be extremely low, classifying domains into structural families can be difficult. The **Pfam database**, which contains over 10,000 HMM-derived multiple alignments of protein domain families, can be used to analyze new protein sequences.

CHALLENGE PROBLEM: Using the Pfam HMM for gp120 (constructed from a seed alignment of just 24 gp120 proteins), construct alignments of all known gp120 proteins and identify the "most diverged" gp120 sequence.

Detours

The Red Queen Effect

The **Red Queen Effect** is the hypothesis that evolution is necessary not only to equip organisms with an advantage in a *fixed* environment, but also to help them survive in response to *changing* environments. Its name derives from a statement that the Red Queen made to Alice in Lewis Carroll's *Through the Looking-Glass*:

> *Now, here, you see, it takes all the running you can do, to keep in the same place.*

The Red Queen Effect is often seen in predator-prey relationships. For example, an adaptation may help wolves run a little faster, and caribou must evolve in turn to survive. The result is that wolves and caribou appear to run at the same speed, with the slowest wolves are starving and the slowest caribou being eaten.

Glycosylation

Cells have a dense coating of sugar chains, called **glycans**, on their surface. Glycans are often post-translational modifications of **glycoproteins**, which modulate interactions with other cells in a multicellular organism or between the cell and another organism (e.g., between human cells and a virus). For example, influenza infection begins with an interaction between the proteins on the virus's surface and glycans on the host cell's surface.

Glycans are constructed from a family of building blocks called **monosaccharides**. Each monosaccharide can be linked with other monosaccharides to form complex, tree-like structures (Figure 10.28).

DNA methylation

DNA methylation results in the addition of a methyl group (CH_3) to a cytosine or guanine nucleotide (Figure 10.29), which often alters the expression of nearby genes. Genes that acquire a high concentration of methylated residues in their upstream regions have suppressed expression. DNA methylation is vital to development, and both DNA hypermethylation and hypomethylation have been linked to various cancers.

DNA methylation is important in the process of **cell differentiation**, in which embryonic stem cells become specialized tissues. The change is often permanent, preventing a cell from reverting to a stem cell or converting to a different cell type. Methylation is inherited during cell division but is usually removed during zygote formation.

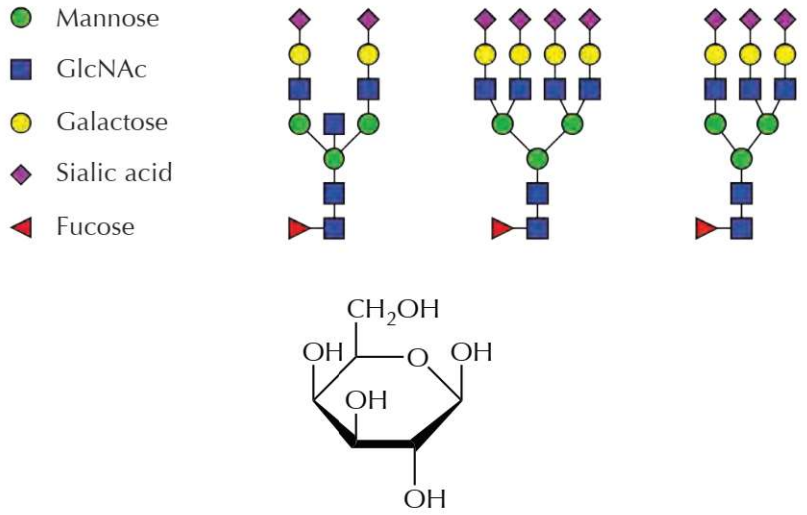

FIGURE 10.28 (Top) Five types of monosaccharides along with three examples of how these monosaccharides are assembled into glycans in humans. (Bottom) The chemical formula for the monosaccharide galactose.

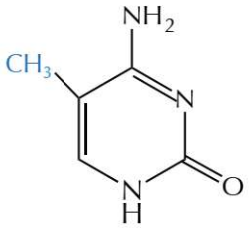

FIGURE 10.29 DNA methylation of a cytosine nucleotide base, with the methyl group shown in blue.

Conditional probability

Let's return to the game of Chō-Han and analyze the sum s of two standard six-sided dice. Let A be the event that s is odd and B be the event that s is larger than 10. The probability of A is equal to $1/2$ because half of the 36 possible outcomes for rolling two dice produce an odd sum. The probability of B is equal to $3/36$ because there are three outcomes ($5+6$, $6+5$, and $6+6$) for which $s > 10$.

STOP and Think: If we tell you that s is larger than 10 (but you cannot see the dice), is s more likely to be odd or even?

The **conditional probability** of event A given event B, denoted $\Pr(A|B)$, is the probability that event A will occur given that event B has occurred. For the dice example, since B corresponds to two tosses with s odd ($6+5$ and $5+6$) and one toss with s even ($6+6$), $\Pr(A|B) = 2/3$. Note that $\Pr(A|B)$ is completely different from the probability $\Pr(A, B)$ that both events A and B will occur, which is equal to $2/36$ (A and B only occur together for the sums $6+5$ and $5+6$).

More generally, the conditional probability $\Pr(A|B)$ is often defined using the following formula:

$$\Pr(A|B) = \frac{\Pr(A, B)}{\Pr(B)}.$$

EXERCISE BREAK: To test your knowledge of conditional probability, consider the following puzzle, called the "Monty Hall Problem", which originally appeared in a letter to *American Statistician* in 1975 and has stumped many aspiring mathematicians over the years:

> *Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?*

Bibliography Notes

The decoding algorithm was developed by Viterbi, 1967. The first algorithms for HMM parameter estimation were developed by Baum et al., 1970. Churchill, 1989, Krogh et al., 1994, and Baldi et al., 1994 pioneered the application of HMMs in computational biology. Bateman et al., 2002 described applications of profile HMM alignments for developing a database of protein domain families Pfam. De Jong et al., 1992 discovered the 11/25 rule. Beerenwinkel and Drton, 2007 constructed an HMM for analyzing HIV resistance.