CHAPTER
SEVEN

HIDDEN MARKOV MODELS I

Anastasiya Belyaeva, Justin Gullingsrud (Sep 26, 2015)
William Leiserson, Adam Sealfon (Sep 22, 2014)
Haoyang Zeng (Sep 28, 2013)
Sumaiya Nazeen (Sep 25, 2012)
Chrisantha Perera (Sep 27, 2011)
Gleb Kuznetsov, Sheida Nabavi (Sep 28, 2010)
Elham Azizi (Sep 29, 2009)

Figures

7.1 Introduction

Hidden Markov Models (HMMs) are a fundamental tool from machine learning that is widely used in computational biology. Using HMMs, we can explore the underlying structure of DNA or polypeptide sequences, detecting regions of especial interest. For instance, we can identify conserved subsequences or uncover regions with different distributions of nucleotides or amino acids such as promoter regions and CpG islands. Using this probabilistic model, we can illuminate the properties and structural components of sequences and locate genes and other functional elements.

This is the first of two lectures on HMMs. In this lecture we will define Markov Chains and HMMs, providing a series of motivating examples. In the second half of this lecture, we wil discuss scoring and decoding. We will learn how to compute the probability of the combination of a particular combination of observations and states. We will introduce the Forward Algorithm, a method for computing the probability of a given sequence of observations, allowing all sequences of states. Finally, we will discuss the problem of determining the most likely path of states corresponding to the given observations, a goal which is achieved by the Viterbi algorithm.

In the second lecture on HMMs, we will continue our discussion of decoding by exploring posterior decoding, which allows us to compute the most likely state at each point in the sequence. We will then explore how to learn a Hidden Markov Model. We cover both supervised and unsupervised learning, explaining how to use each to learn the model parameters. In supervised learning, we have training data available that labels sequences with particular models. In unsupervised learning, we do not have labels so we must seek to partition the data into discrete categories based on discovered probabilistic similarities. In our discussion of unsupervised learning we will introduce the general and widely applicable Expectation Maximization (EM) algorithm.

7.2 Motivation:

7.2.1 We have a new sequence of DNA, now what?

1. **Align it:**
    - with things we know about (database search).
    - with unknown things (assemble/clustering)
2. **Visualize it:** *"Genomics rule #1": Look at your data!*
    - Look for nonstandard nucleotide compositions.
    - Look for k-mer frequencies that are associated with protein coding regions, recurrent data, high GC content, etc.
    - Look for motifs, evolutionary signatures.
    - Translate and look for open reading frames, stop codons, etc.
    - Look for patterns, then develop machine learning tools to determine reasonable probabilistic models. For example by looking at a number of quadruples we decide to color code them to see where they most frequently occur.
3. **Model it:**
    - Make hypothesis.
    - Build a generative model to describe the hypothesis.
    - Use that model to find sequences of similar **type**.

    We're not looking for sequences that necessarily have common ancestors. Rather, we're interested in sequences with similar properties. We actually don't know how to model whole genomes, but we can model small aspects of genomes. The task requires understanding all the properties of genome regions and computationally building generative models to represent hypotheses. For a given sequence, we want to annotate regions whether they are introns, exons, intergenic, promoter, or otherwise classifiable regions.

Building this framework will give us the ability to:

- Emit (generate) sequences of similar type according to the generative model
- Recognize the hidden state that has most likely generated the observation

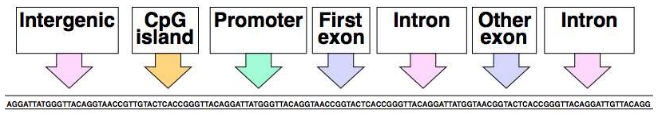


Figure 7.1: Modeling biological sequences

- Learn (train) large datasets and apply to both previously labeled data (supervised learning) and unlabeled data (unsupervised learning).

In this lecture we discuss algorithms for emission and recognition.

7.2.2 Why probabilistic sequence modeling?

- Biological data is noisy.
- Update previous knowledge about biological sequences.
- Probability provides a calculus for manipulating models.
- Not limited to yes/no answers, can provide degrees of belief.
- Many common computational tools are based on probabilistic models.
- Our tools: Markov Chains and HMM.

7.3 Markov Chains and HMMS: From Example To Formalizing

7.3.1 Motivating Example: Weather Prediction

Weather prediction has always been difficult, especially when we would like to forecast the weather many days, weeks or even months later. However, if we only need to predict the weather of the next day, we can reach decent prediction precision using some quite simple models such as Markov Chain and Hidden Markov Model by building graphical models in Figure 7.2.

For the Markov Chain model on the left, four kinds of weather (Sun, Rain, Clouds and Snow) can directly transition from one to the other. This is a "what you see is what you get" in that the next state only depends on the current state and there is no memory of the previous state. However for HMM on the right, all the types of weather are modeled as the emission(or outcome) of the hidden seasons (Summer, Fall, Winter and Spring). The key insight behind is that the hidden states of the world (e.g. season or storm system) determines emission probabilities while state transitions are governed by a Markov Chain.

7.3.2 Formalizing of Markov Chain and HMMS

To take a closer look at Hidden Markov Model, let's first define the key parameters in Figure 7.3. Vector x represents sequence of observations. Vector π represents the hidden path, which is the sequence of hidden states. Each entry a_{kl} of Transition matrix A denotes the probability of transition from state k to state l. Each entry $e_k(x_i)$ of emission vector denotes the probability of observing x_i from state k. And finally with these parameters and Bayes's rule, we can use $p(x_i|\pi_i = k)$ to estimate $p(\pi_i = k|x_i)$.

Markov Chains

A Markov Chain is given by a finite set of states and transition probabilities between the states. At every time step, the Markov Chain is in a particular state and undergoes a transition to another state. The probability of transitioning to each other state depends only on the current state, and in particular is independent of how the current state was reached. More formally, a Markov Chain is a triplet (Q, p, A) which consists of:

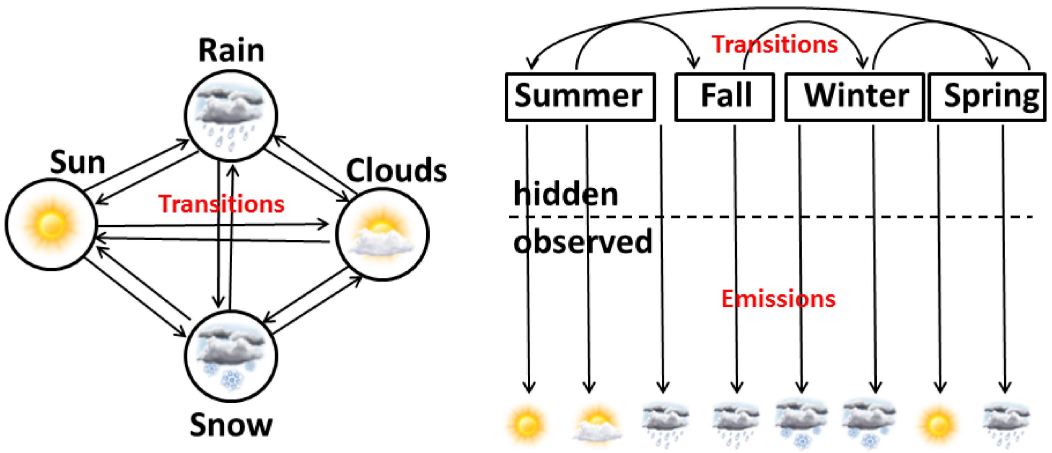


Figure 7.2: Prediction Models Using Markov Chain and HMM

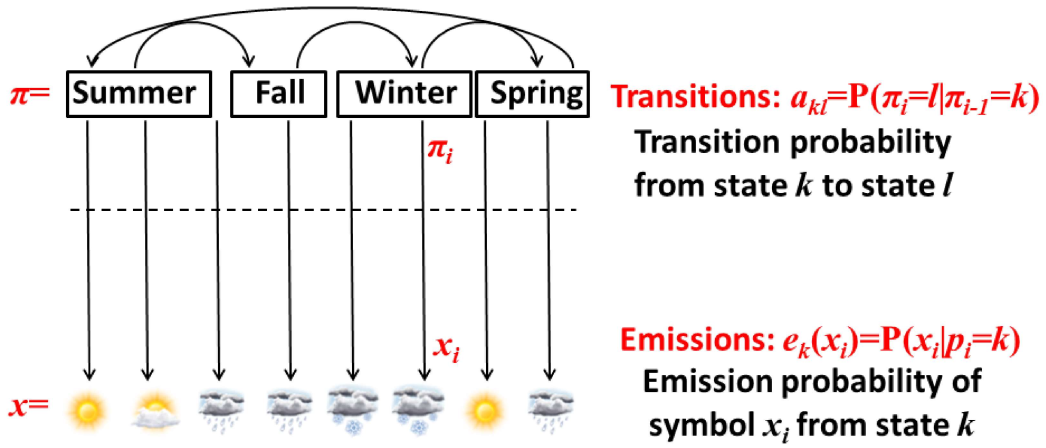

π=
Summer
Fall
Winter
Spring
π_i
Transitions: a_kl=P(π_i=l|π_i-1=k)
Transition probability from state k to state l
x_i
x=
Emissions: e_k(x_i)=P(x_i|p_i=k)
Emission probability of symbol x_i from state k

Figure 7.3: Parameterization of HMM Prediction Model

- A set of states Q.
- A transition matrix A whose elements correspond to the probability A_{ij} of transitioning from state i to state j.
- A vector p of initial state probabilities.

The key property of Markov Chains is that they are memory-less, i.e., each state depends only on the previous state. So we can immediately define a probability for the next state, given the current state:

$$P(x_i|x_{i-1},...,x_1) = P(x_i|x_{i-1})$$

In this way, the probability of the sequence can be decomposed as follows:

$$P(x) = P(x_L, x_{L-1}, ..., x_1) = P(x_L|x_{L-1})P(x_{L-1}|x_{L-2})...P(x_2|x_1)P(x_1)$$

$P(x_L)$ can also be calculated from the transition probabilities: If we multiply the initial state probabilities at time $t = 0$ by the transition matrix A, we get the probabilities of states at time $t = 1$. Multiplying by the appropriate power A^L of the transition matrix, we obtain the state probabilities at time $t = L$.

Hidden Markov Models

Hidden Markov Models are used as a representation of a problem space in which observations come about as a result of states of a system which we are unable to observe directly. These observations, or emissions, result from a particular state based on a set of probabilities. Thus HMMs are Markov Models where the states are hidden from the observer and instead we have observations generated with certain probabilities associated with each state. These probabilities of observations are known as emission probabilities.

Formally, a Hidden Markov Model is a 5-tuple (Q, A, p, V, E) which consists of the following parameters:

- A series of states, Q.
- A transition matrix, A
- A vector of initial state probabilities , p.
- A set of observation symbols, V, for example {A, T, C, G} or the set of amino acids or words in an English dictionary.
- A matrix of emission probabilities, E: For each s, t, in Q, the emission probability is

$$e_{sk} = P(v_k \text{ at time } t | q_t = s)$$

The key property of memorylessness is inherited from Markov Models. The emissions and transitions depend only on the current state and not on the past history.

7.4 Apply HMM to Real World: From Casino to Biology

7.4.1 The Dishonest Casino

The Scenario

Imagine the following scenario: You enter a casino that offers a dice-rolling game. You bet \$1 and then you and a dealer both roll a die. If you roll a higher number you win \$2. Now there's a twist to this seemingly simple game. You are aware that the casino has two types of dice:

1. Fair die: $P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6$
2. Loaded die: $P(1) = P(2) = P(3) = P(4) = P(5) = 1/10$ and $P(6) = 1/2$

The dealer can switch between these two dice at any time without you knowing it. The only information that you have are the rolls that you observe. We can represent the state of the casino die with a simple Markov model:

The model shows the two possible states, their emissions, and probabilities for transition between them. The transition probabilities are educated guesses at best. We assume that switching between the states doesn't happen too frequently, hence the .95 chance of staying in the same state with every roll.

Staying in touch with biology: An analogy

For comparison, Figure 7.5 below gives a similar model for a situation in biology where a sequence of DNA has two potential sources: injection by a virus versus normal production by the organism itself:

Given this model as a hypothesis, we would observe the frequencies of C and G to give us clues as to the source of the sequence in question. This model assumes that viral inserts will have higher CpG prevalence, which leads to the higher probabilities of C and G occurrence.

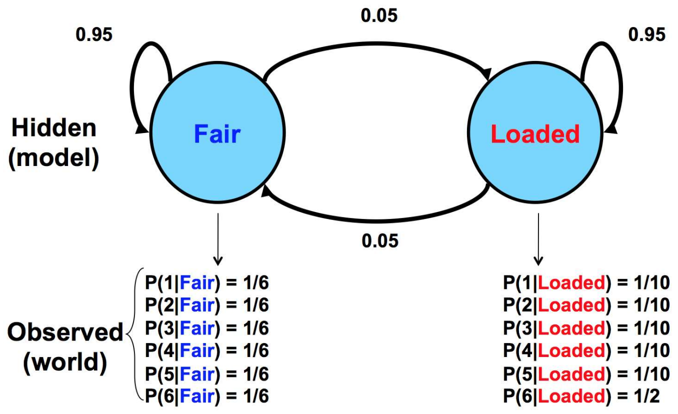


Figure 7.4: State of a casino die represented by a Hidden Markov model

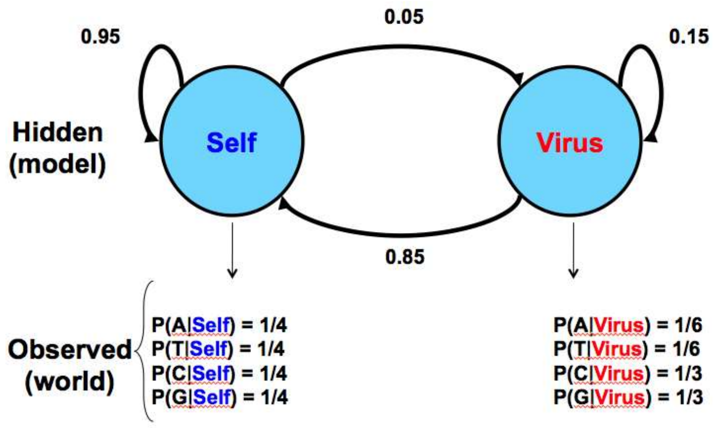


Figure 7.5: Potential DNA sources: viral injection vs. normal production

Running the Model

Say we are at the casino and observe the sequence of rolls given in Figure 7.6. We would like to know whether it is more likely that the casino is using the fair die or the loaded die.

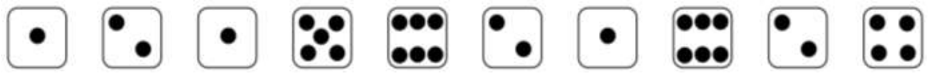


Figure 7.6: A possible sequence of observed die rolls.

Let's look at a particular sequence of rolls.

Therefore, we will consider two possible sequences of states in the underlying HMM, one in which the dealer is always using a fair die, and the other in which the dealer is always using a loaded die. We consider each execution path to understand the implications. For each case, we compute the joint probability of an observed outcome with that sequence of underlying states.

In the first case, where we assume the dealer is always using a fair die, the transition and emission probabilities are shown in Figure 7.7. The probability of this sequence of states and observed emissions is a product of terms which can be grouped into three components: 1/2, the probability of starting with the fair die; $(1/6)^{10}$, the probability of the sequence of rolls if we always use the fair die; and lastly $(0.95)^9$, the

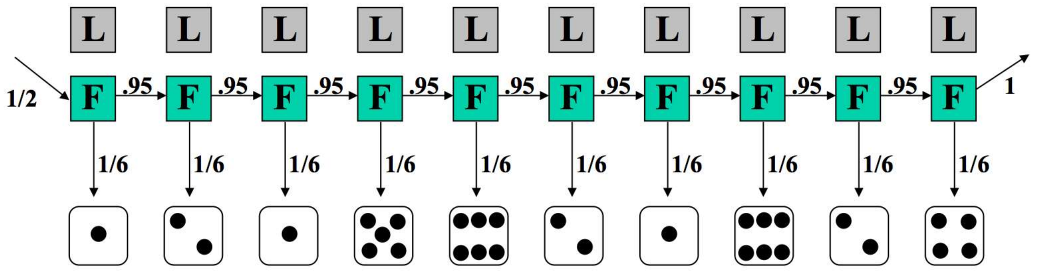


Figure 7.7: Running the model: probability of a sequence, given path consists of all fair dice

probability that we always continue to use the fair die.

In this model, we assume $\pi = \{F, F, F, F, F, F, F, F, F, F\}$, and we observe $x = \{1, 2, 1, 5, 6, 2, 1, 6, 2, 4\}$. Now we can calculate the joint probability of x and π as follows:

$$\begin{aligned} P(x, \pi) &= P(x|\pi)P(\pi) \\ &= \frac{1}{2} \times P(1|F) \times P(F|F) \times P(2|F) \cdots \\ &= \frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times (0.95)^9 \\ &= 5.2 \times 10^{-9} \end{aligned}$$

With a probability this small, this might appear to be an extremely unlikely case. In actuality, the probability is low because there are many equally likely possibilities, and no one outcome is *a priori* likely. The question is not whether this sequence of hidden states is likely, but whether it is *more* likely than the alternatives.

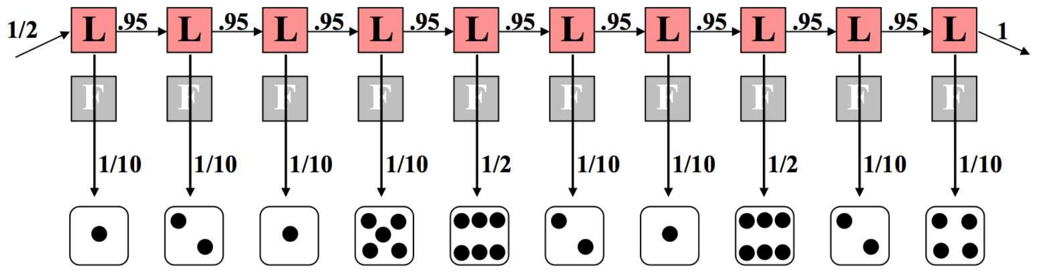


Figure 7.8: Running the model: probability of a sequence, given path consists of all loaded dice

Let us consider the opposite extreme where the dealer always uses a loaded die, as depicted in Figure 7.8. This has a similar calculation except that we note a difference in the emission component. This time, 8 of the 10 rolls carry a probability of 1/10 because the loaded die disfavors non-sixes. The remaining two rolls of six have each a probability of 1/2 of occurring. Again we multiply all of these probabilities together according to principles of independence and conditioning. In this case, the calculations are as follows:

$$\begin{aligned} P(x, \pi) &= \frac{1}{2} \times P(1|L) \times P(L|L) \times P(2|L) \cdots \\ &= \frac{1}{2} \times \left(\frac{1}{10}\right)^{8} \times \left(\frac{1}{2}\right)^{2} \times (0.95)^9 \\ &= 7.9 \times 10^{-10} \end{aligned}$$

Note the difference in exponents. If we make a direct comparison, we can say that the situation in which a fair die is used throughout the sequence is 52×10^{-10} (as compared with 7.9×10^{-10} with the loaded die).

Therefore, it is six times more likely that the fair die was used than that the loaded die was used. This is not too surprising—two rolls out of ten yielding a 6 is not very far from the expected number 1.7 with the fair die, and farther from the expected number 5 with the loaded die.

Adding Complexity

Now imagine the more complex, and interesting, case where the dealer switches the die at some point during the sequence. We make a guess at an underlying model based on this premise in Figure 7.9.

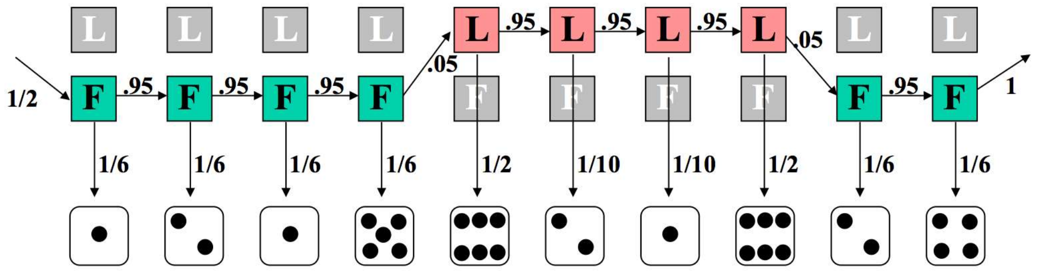


Figure 7.9: Partial runs and die switching

Again, we can calculate the likelihood of the joint probability of this sequence of states and observations. Here, six of the rolls are calculated with the fair die, and four with the loaded one. Additionally, not all of the transition probabilities are 95% anymore. The two swaps (between fair and loaded) each have a probability of 5%.

$$\begin{aligned}P(x,\pi) &= \frac{1}{2} \times P(1|L) \times P(L|L) \times P(2|L) \cdots \\ &= \frac{1}{2} \times \left(\frac{1}{10}\right)^2 \times \left(\frac{1}{2}\right)^2 \times \left(\frac{1}{6}\right)^6 \times (0.95)^7 \times (0.05)^2 \\ &= 4.67 \times 10^{-11}\end{aligned}$$

Clearly, our guessed path is far less likely than either of the previous two cases. But if we are looking for the most likely scenario, we cannot possibly calculate *all* alternatives in this way. We need new techniques for inferring the underlying model. In the above cases we more-or-less just guessed at the model, but what we want is a way to systematically derive likely models. Let's formalize the models introduced thus far as we continue toward understanding HMM-related techniques.

7.4.2 Back to Biology

Now that we have formalized HMMs, we want to use them to solve some real biological problems. In fact, HMMs are a great tool for gene sequence analysis, because we can look at a sequence of DNA as being emitted by a mixture of models. These may include introns, exons, transcription factors, etc. While we may have some sample data that matches models to DNA sequences, in the case that we start fresh with a new piece of DNA, we can use HMMs to ascribe some potential models to the DNA in question. We will first introduce a simple example and think about it a bit. Then, we will discuss some applications of HMM in solving interesting biological questions, before finally describing the HMM techniques that solve the problems that arise in such a first-attempt/native analysis.

A simple example: Finding GC-rich regions

Imagine the following scenario: we are trying to find GC rich regions by modeling nucleotide sequences drawn from two different distributions: background and promoter. Background regions have uniform distribution of 0.25 for each of A, T, G, C. Promoter regions have probabilities: A: 0.15, T: 0.13, G: 0.30, C: 0.42. Given one nucleotide observed, we cannot say anything about the region from which it was originated, because either region will emit each nucleotide at some probability. We can learn these initial state probabilities based in steady state probabilities. By looking at a sequence, we want to identify which regions originate from a background distribution (B) and which regions are from a promoter model (P).

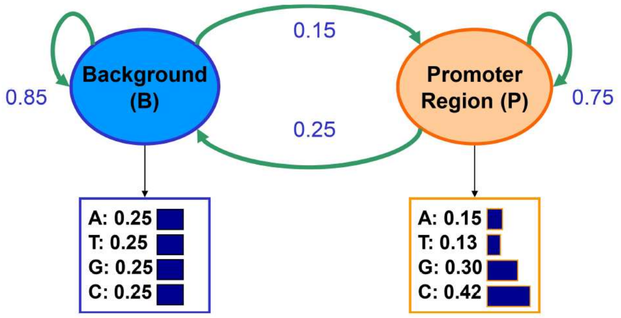
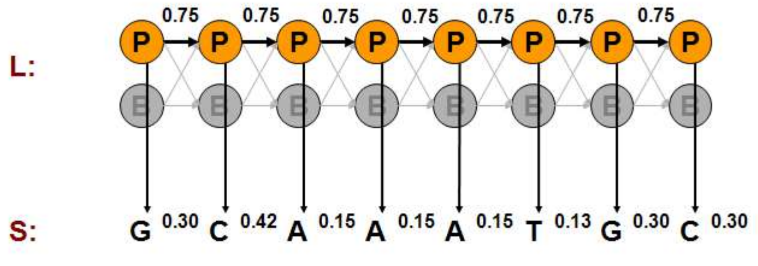


Figure 7.10: HMMS as a generative model for finding GC-rich regions.

We are given the transition and emission probabilities based on relevant abundance and average length of regions where x = vector of observable emissions consisting of symbols from the alphabet {A,T,G,C}; π = vector of states in a path (e.g. BPPBP); π^* = maximum likelihood of generating that path. In our interpretation of sequence, the max likelihood path will be found by incorporating all emission and transition probabilities by dynamic programming.

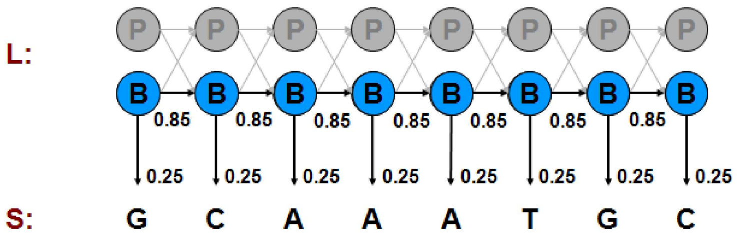
HMMs are generative models, in that an HMM gives the probability of emission given a state (using Bayes' Rule), essentially telling you how likely the state is to generate those sequences. So we can always run a generative model for transitions between states and start anywhere. In Markov Chains, the next state will give different outcomes with different probabilities. No matter which state is next, at the next state, the next symbol will still come out with different probabilities. HMMs are similar: You can pick an initial state based on the initial probability vector. In the example above, we will start in state B with high probability since most locations do not correspond to promoter regions. You then draw an emission from the $P(X|B)$. Each nucleotide occurs with probability 0.25 in the background state. Say the sampled nucleotide is a G. The distribution of subsequent states depends only on the fact that we are in the background state and is independent of this emission. So we have that the probability of remaining in state B is 0.85 and the probability of transitioning to state P is 0.15, and so on.

We can compute the probability of one such generation by multiplying the probabilities that the model makes exactly the choices we assumed. Consider the examples shown in Figures 7.11, 7.12, and 7.13.



$$P(x,\pi)=a_P*e_P(G)*a_{PP}*e_P(G)*a_{PP}*e_P(C)*a_{PP}*e_P(A)*a_{PP}*\ldots$$
$$=a_p*(0.75)^7*(0.15)^3*(0.13)^1*(0.30)^2*(0.42)^2$$
$$=9.3*10^{-7}$$

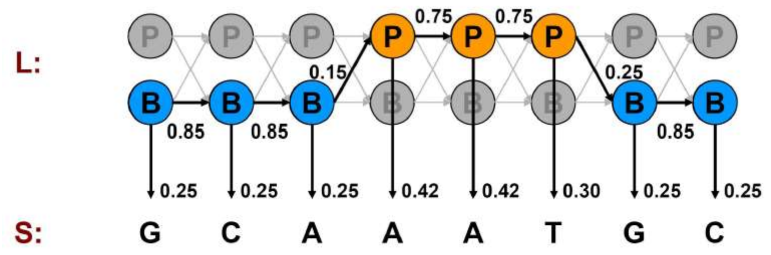
Figure 7.11: Probability of seq, path if all promoter



$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(B_3 \mid B_2)\dots P(C \mid B_7)$$
$$= (0.85)^7 \times (0.25)^8$$
$$= 4.9 \times 10^{-6}$$

A: 0.25	■
T: 0.25	■
G: 0.25	■
C: 0.25	■

Figure 7.12: Probability of seq, path if all background



$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(P_3 \mid B_2)\dots P(C \mid B_7)$$
$$= (0.85)^3 \times (0.25)^6 \times (0.75)^2 \times (0.42)^2 \times 0.30 \times 0.15$$
$$= 6.7 \times 10^{-7}$$

Figure 7.13: Probability of seq, path sequence if mixed

We can calculate the joint probability of a particular sequence of states corresponding to the observed emissions as we did in the Casino examples:

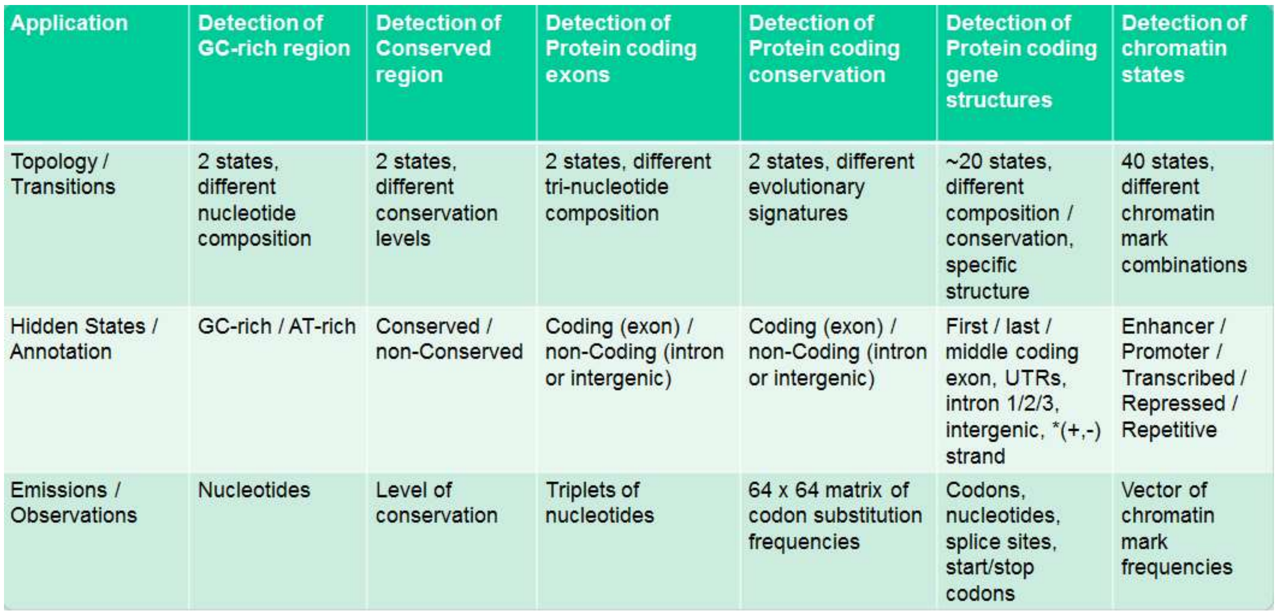
$$\begin{aligned}
P(x, \pi_P) &= a_P \times e_P(G) \times a_{PP} \times e_P(G) \times \cdots \\
&= a_P \times (0.75)^7 \times (0.15)^3 \times (0.13) \times (0.3)^2 \times (0.42)^2 \\
&= 9.3 \times 10^{-7} \\
P(x, \pi_B) &= (0.85)^7 \times (0.25)^8 \\
&= 4.9 \times 10^{-6} \\
P(x, \pi_{mixed}) &= (0.85)^3 \times (0.25)^6 \times (0.75)^2 \times (0.42)^2 \times 0.3 \times 0.15 \\
&= 6.7 \times 10^{-7}
\end{aligned}$$

The pure-background alternative is the most likely option of the possibilities we have examined. But how do we know whether it is the option most likely out of all possibile paths of states to have generated the observed sequence?

The brute force approach is to examine at all paths, trying all possibilities and calculating their joint probabilities $P(x, \pi)$ as we did above. The sum of probabilities of all the alternatives is 1. For example, if all states are promoters, $P(x, \pi) = 9.3 \times 10^{-7}$. If all emissions are Gs, $P(x, \pi) = 4.9 \times 10^{-6}$. If we have use the mixture of B's and P's as in Figure 7.13, $P(x, \pi) = 6.7 \times 10^{-7}$; which is small because a lot of penalty is paid for the transitions between B's and P's which are exponential in length of sequence. Usually, if you observe more G's, it is more likely to be in the promoter region and if you observe more A and Ts, then it is more likely to be in the background. But we need something more than just observation to support our belief. We will see how can we mathematically support our intuition in the following sections.

Application of HMMs in Biology

HMMs are used in answering many interesting biological questions. Some biological application of HMMs are summarized in Figure 7.14.



Application	Detection of GC-rich region	Detection of Conserved region	Detection of Protein coding exons	Detection of Protein coding conservation	Detection of Protein coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition / conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved / non-Conserved	Coding (exon) / non-Coding (intron or intergenic)	Coding (exon) / non-Coding (intron or intergenic)	First / last / middle coding exon, UTRs, intron 1/2/3, intergenic, *(+,-) strand	Enhancer / Promoter / Transcribed / Repressed / Repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	64 x 64 matrix of codon substitution frequencies	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

Figure 7.14: Some biological applications of HMM

7.5 Algorithmic Settings for HMMs

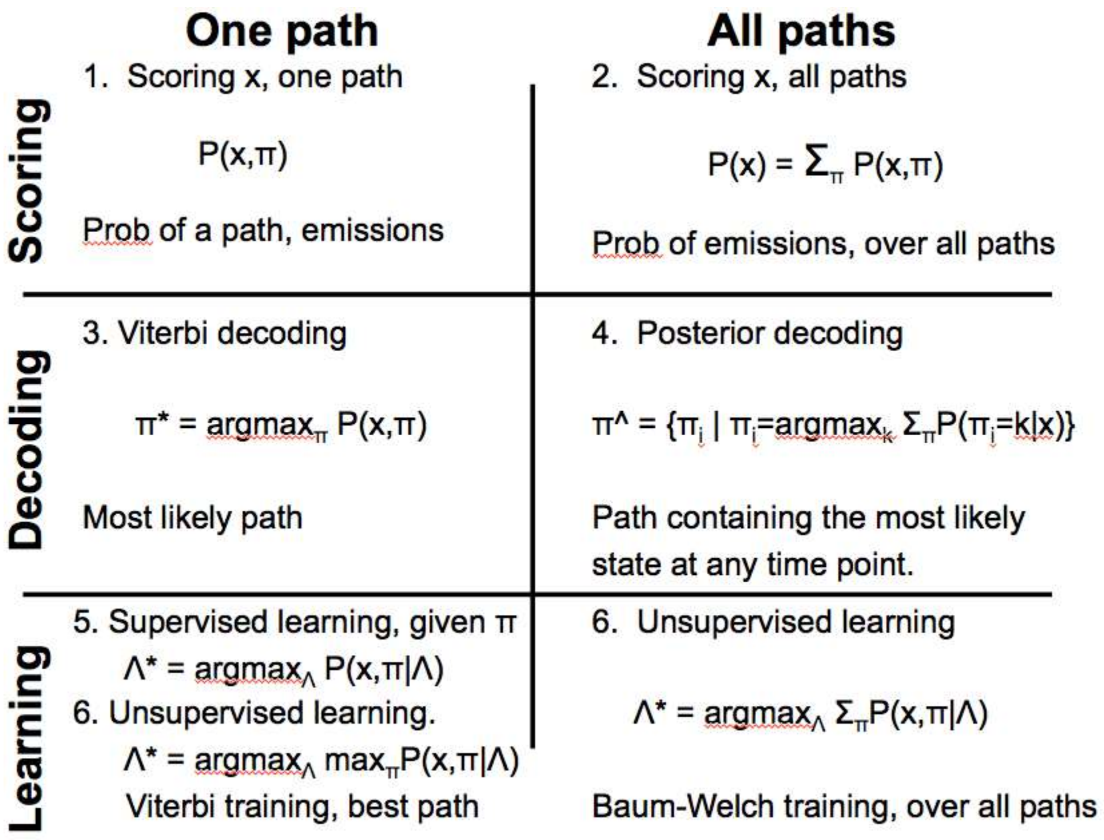
We use HMMs for three types of operation: **scoring**, **decoding**, and **learning**. We will talk about scoring and decoding in this lecture. These operations can happen for a single path or all possible paths. For the single path operations, our focus is on discovering the path with maximum probability. However, we are interested in a sequence of observations or emissions for all path operations regardless of its corresponding paths.

7.5.1 Scoring

Scoring over a single path

The Dishonest Casino problem and Prediction of GC-rich Regions problem described in section 7.4 are both examples of finding the probability score corresponding to a single path. For a single path we define the scoring problem as follows:

- **Input:** A sequence of observations $x = x_1 x_2 \dots x_n$ generated by an HMM $M(Q, A, p, V, E)$ and a path of states $\pi = \pi_1 \pi_2 \dots \pi_n$.
- **Output:** Joint probability, $P(x, \pi)$ of observing x if the hidden state sequence is π.



	One path	All paths
Scoring	1. Scoring x, one path P(x,π) Prob of a path, emissions	2. Scoring x, all paths P(x) = Σ_π P(x,π) Prob of emissions, over all paths
Decoding	3. Viterbi decoding π* = argmax$_\pi$ P(x,π) Most likely path	4. Posterior decoding π^ = {π$_i$ | π$_i$=argmax$_k$ Σ_πP(π$_i$=k|x)} Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π Λ* = argmax$_\Lambda$ P(x,π|Λ) 6. Unsupervised learning. Λ* = argmax$_\Lambda$ max$_\pi$P(x,π|Λ) Viterbi training, best path	6. Unsupervised learning Λ* = argmax$_\Lambda$ Σ_πP(x,π|Λ) Baum-Welch training, over all paths

Figure 7.15: The six algorithmic settings for HMMS

The single path calculation is essentially the likelihood of observing the given sequence over a particular path using the following formula:

$$P(x, \pi) = P(x|\pi)P(\pi)$$

We have already seen the examples of single path scoring in our Dishonest Casino and GC-rich region examples.

Scoring over all paths

We define the all paths version of scoring problem as follows:

- **Input:** A sequence of observations $x = x_1x_2\dots x_n$ generated by an HMM $M(Q, A, p, V, E)$.
- **Output:** The joint probability, $P(x, \pi)$ of observing x over all possible sequences of hidden states π.

The probability over all paths π of hidden states of the given sequence of observations is given by the following formula.

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

We use this score when we are interested in knowing the likelihood of a particular sequence for a given HMM. However, naively computing this sum requires considering an exponential number of possible paths. Later in the lecture we will see how to compute this quantity in polynomial time.

7.5.2 Decoding

Decoding answers the question: Given some observed sequence, what path gives us the maximum likelihood of observing this sequence? Formally we define the problem as follows:

- **Decoding over a single path:**
  - Input: A sequence of observations $x = x_1x_2 \dots x_N$ generated by an HMM $M(Q, A, p, V, E)$.
  - Output: The most probable path of states, $\pi^* = \pi_1^*\pi_2^* \dots \pi_N^*$
- **Decoding over all paths:**
  - Input: A sequence of observations $x = x_1x_2 \dots x_N$ generated by an HMM $M(Q, A, p, V, E)$.
  - Output: The path of states, $\pi^* = \pi_1^*\pi_2^* \dots \pi_N^*$ that contains the most likely state at each time point.

In this lecture, we will look only at the problem of decoding over a single path. The problem of decoding over all paths will be discussed in the next lecture.

For the single path decoding problem, we can imagine a brute force approach where we calculate the joint probabilities of a given emission sequence and all possible paths and then pick the path with the maximum joint probability. The problem is that there are an exponential number of paths and using such a brute force search for the maximum likelihood path among all possible paths is very time consuming and impractical. **Dynamic Programming** can be used to solve this problem. Let us formulate the problem in the dynamic programming approach.

We would like to find out the most likely sequence of states based on the observation. As inputs, we are given the model parameters $e_i(s)$,the emission probabilities for each state, and $a_{ij}s$, the transition probabilities. The sequence of emissions x is also given. The goal is to find the sequence of hidden states, π^*, which maximizes the joint probability with the given sequence of emissions. That is,

$$\pi^* = \arg\max_{\pi} P(x, \pi)$$

Given the emitted sequence x we can evaluate any path through hidden states. However, we are looking for the best path. We start by looking for the optimal substructure of this problem.

For a best path, we can say that, the best path through a given state must contain within it the following:

- The best path to previous state
- The best transition from previous state to this state
- The best path to the end state

Therefore the best path can be obtained based on the best path of the previous states, i.e., we can find a recurrence for the best path. The **Viterbi** algorithm is a dynamic programming algorithm that is commonly used to obtain the best path.

Most probable state path: the Viterbi algorithm

Suppose $v_k(i)$ is the known probability of the most likely path ending at position (or time instance) i in state k for each k. Then we can compute the corresponding probabilities at time $i + 1$ by means of the following recurrence.

$$v_l(i+1) = e_l(x_{i+1}) \max_k (a_{kl} v_k(i))$$

The most probable path π^*, or the maximum $P(x, \pi)$, can be found recursively. Assuming we know $v_j(i-1)$, the score of the maximum path up to time $i-1$, we need to increase the computation for the next time step. The new maximum score path for each state depends on

- The maximum score of the previous states
- The transition probability
- The emission probability.

In other words, the new maximum score for a particular state at time i is the one that maximizes the transition of all possible previous states to that particular state (the penalty of transition multiplied by their maximum previous scores multiplied by emission probability at the current time).

All sequences have to start in state 0 (the begin state). By keeping pointers backwards, the actual state sequence can be found by backtracking. The solution of this Dynamic Programming problem is very similar to the alignment algorithms that were presented in previous lectures.

The steps of the Viterbi algorithm [2] are summarized below:

1. Initialization ($i = 0$): $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$.
2. Recursion ($i = 1 \dots N$): $v_k(i) = e_k(x_i) \max_j(a_{jk}v_j(i-1))$; $ptr_i(l) = \arg\max_j(a_{jk}v_j(i-1))$.
3. Termination: $P(x, \pi^*) = \max_k v_k(N)$; $\pi^*_N = \arg\max_k v_k(N)$.
4. Traceback ($i = N \dots 1$): $\pi^*_{i-1} = ptr_i(\pi^*_i)$.

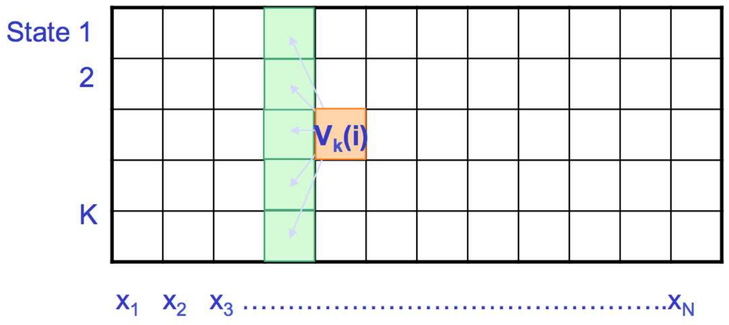

State 1
2
K
$V_k(i)$
x_1 x_2 x_3 ....x_N

Figure 7.16: The Viterbi algorithm

As we can see in Figure 7.16, we fill the matrix from left to right and trace back. Each position in the matrix has K states to consider and there are KN cells in the matrix, so, the required computation time is $O(K^2N)$ and the required space is $O(KN)$ to remember the pointers. In practice, we use log scores for the computation. Note that the running time has been reduced from exponential to polynomial.

7.5.3 Evaluation

Evaluation is about answering the question: How well does our model of the data capture the actual data? Given a sequence x, many paths can generate this sequence. The question is how likely is the sequence given the model? In other words, is this a good model? Or, how well does the model capture the exact characteristics of a particular sequence? We use evaluation of HMMs to answer these questions. Additionally, with evaluation we can compare different models.

Let us first provide a formal definition of the Evaluation problem.

- Input: A sequence of observations $x = x_1x_2 \dots x_N$ and an HMM $M(Q, A, p, V, E)$.
- Output: The probability that x was generated by M summed over all paths.

We know that if we are given an HMM we can generate a sequence of length n using the following steps:

- Start at state π_1 according to probability $a_{0\pi_1}$ (obtained using vector, p).
- Emit letter x_1 according to emission probability $e_{\pi_1}(x_1)$.
- Go to state π_2 according to the transition probability $a_{\pi_1|\pi_2}$
- Keep doing this until emit x_N.

Thus we can emit any sequence and calculate its likelihood. However, many state sequence can emit the same x. Then, how do we calculate the total probability of generating a given x over all paths? That is, our goal is to obtain the following probability:

$$P(x|M) = P(x) = \sum_{\pi} P(x,\pi) = \sum_{\pi} P(x|\pi)P(\pi)$$

The challenge of obtaining this probability is that there are too many paths (an exponential number) and each path has an associated probability. One approach may be using just the Viterbi path and ignoring the others, since we already know how to obtain this path. But its probability is very small as it is only one of the many possible paths. It is a good approximation only if it has high probability density. In other cases, the Viterbi path will give us an inaccurate approximation. Alternatively, the correct approach for calculating the exact sum iteratively is through the use of dynamic programming. The algorithm that does this is known as **Forward Algorithm**.

The Forward Algorithm

First we derive the formula for forward probability $f(i)$.

$$\begin{aligned}
f_l(i) &= P(x_1 \dots x_i, \pi = l) && (7.1)\\
&= \sum_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1}, \pi_i = l) e_l(x_i) && (7.2)\\
&= \sum_k \sum_{\pi_1 \dots \pi_{i-2}} P(x_1 \dots x_{i-1}, \pi_1, \dots, \pi_{i-2}, \pi_{i-1} = k) a_{kl} e_l(x_i) && (7.3)\\
&= \sum_k f_k(i-1) a_{kl} e_l(x_i) && (7.4)\\
&= e_l(x_i) \sum_k f_k(i-1) a_{kl} && (7.5)\\
& && (7.6)
\end{aligned}$$

The full algorithm[2] is summarized below:

- Initialization ($i = 0$): $f_0(0) = 1$, $f_k(0) = 0$ for $k > 0$.
- Iteration ($i = 1 \dots N$): $f_k(i) = e_k(x_i) \sum_j f_j(i-1) a_{jk}$.
- Termination: $P(x, \pi^*) = \sum_k f_k(N)$

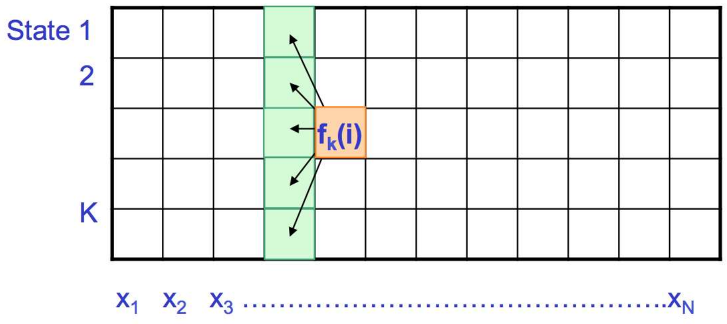


Figure 7.17: The Forward algorithm

From Figure 7.17, it can be seen that the Forward algorithm is very similar to the Viterbi algorithm. In the Forward algorithm, summation is used instead of maximization. Here we can reuse computations of the previous problem including penalty of emissions, penalty of transitions and sums of previous states. The required computation time is $O(K^2N)$ and the required space is $O(KN)$. The drawback of this algorithm is that in practice, taking the sum of logs is difficult; therefore, approximations and scaling of probabilities are used instead.

7.6 An Interesting Question: Can We Incorporate Memory in Our Model?

The answer to this question is - Yes, we can! But how? Recall that, Markov models are memoryless. In other words, all memory of the model is enclosed in states. So, in order to store additional information, we must increase the number of states. Now, look back to the biological example we gave in Section 7.4.2. In our model, state emissions were dependent only on the current state. And, the current state encoded only one nucleotide. But, what if we want our model to count di-nucleotide frequencies (for CpG islands[1]), or, tri-nucleotide frequencies (for codons), or di-codon frequencies involving six-nucleotide? We need to expand number of states.

For example, the last-seen nucleotide can be incorporated into the HMM's "memory" by splitting the plus and minus states from our High-GC/Low-GC HMM into multiple states: one for each nucleotide/region combination, as in Figure 7.18.

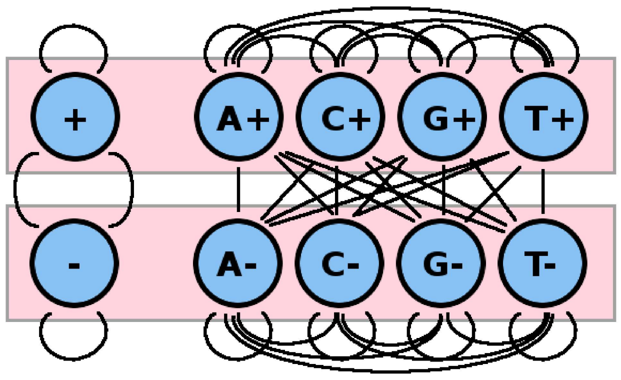


Figure 7.18: CpG Islands - Incorporating Memory

Moving from two to eight states allows us to retain memory of the last nucleotide observed, while also distinguishing between two distinct regions. Four new states now correspond to each of the original two states in the High/Low-GC HMM. Whereas the transition weights in the smaller HMM were based purely on the frequencies of individual nucleotides, now in the larger one, they are based on di-nucleotide frequencies.

With this added power, certain di-nucleotide sequences, such as CpG islands, can be modeled specifically: the transition from C+ to G+ can be assigned greater weight than the transition from A+ to G+. Further, transitions between + and - can be modeled more specifically to reflect the frequency (or infrequency) of particular di-nucleotide sequences within one or the other.

The process of adding memory to an HMM can be generalized and more memory can be added to allow the recognition of sequences of greater length. For instance, we can detect codon triplets with 32 states, or di-codon sextuplets with 2048 states. Memory within the HMM allows for increasingly tailored specificity in scanning.

[1]CpG stands for C-phosphate-G. So, CpG island refers to a region where GC di-nucleotide appear on the same strand.

7.7 Further Reading

7.7.1 Length Distributions of States and Generalized Hidden Markov Models

Given a Markov chain with the transition from any state to the end state having probability τ, the probability of generating a sequence of length L (and then finishing with a transition to the end state) is given by:

$$\tau(1-\tau)^{L-1}$$

Similarly, in the HMMs that we have been examining, the length of states will be exponentially distributed, which is not appropriate for many purposes. (For example, in a genomic sequence, an exponential distribution does not accurately capture the lengths of genes, exons, introns, etc). How can we construct a model that does not output state sequences with an exponential distribution of lengths? Suppose we want to make sure that our sequence has length exactly 5. We might construct a sequence of five states with only a single path permitted by the transition probabilities. If we include a self loop in one of the states, we will output sequences of minimum length 5, with longer sequences exponentially distributed. Suppose we have a chain of n states, with all chains starting with state π_1 and transitioning to an end state after π_n. Also assume that the transition probability between state π_i and π_{i+1} is $1-p$, while the self transition probability of state π_i is p. The probability that a sequence generated by this Markov chain has length L is given by:

$$\binom{L-1}{n-1} p^{L-n}(1-p)^n$$

This is called the negative binomial distribution.

More generally, we can adapt HMMs to produce output sequences of arbitrary length. In a *Generalized Hidden Markov Model* [1] (also known as a *hidden semi-Markov model*), the output of each state is a string of symbols, rather than an individual symbol. The length as well as content of this output string can be chosen based on a probability distribution. Many gene finding tools are based on generalized hidden Markov models.

7.7.2 Conditional random fields

The conditional random field model a discriminative undirected probabilistic graphical model that is used alternatively to HMMs. It is used to encode known relationships between observations and construct consistent interpretations. It is often used for labeling or parsing of sequential data. It is widely used in gene finding. The following resources can be helpful in order to learn more about CRFs:

- Lecture on Conditional Random Fields from Probabilistic Graphical Models course: `https://class.coursera.org/pgm/lecture/preview/33`. For background, you might also want to watch the two previous segments, on pairwise Markov networks and general Gibbs distributions.
- Conditional random fields in biology: `http://www.cis.upenn.edu/~pereira/papers/crf.pdf`
- Conditional Random Fields tutorial: `http://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf`

7.8 Current Research Directions

7.9 Tools and Techniques

7.10 What Have We Learned?

In this section, the main contents we covered are as following:

- First, we introduced the motivation behind adopting Hidden Markov Models in our analysis of genome annotation.

- Second, we formalized Markov Chains and HMM under the light of weather prediction example.
- Third, we got a sense of how to apply HMM in real world data by looking at Dishonest Casino and CG-rich region problems.
- Fourthly, we systematiclly introduced algorithmic settings of HMM and went into detail of three of them:
  - Scoring: scoring over single path
  - Scoring: scoring over all paths
  - Decoding: Viterbi coding in determing most likely path
- Finally, we discussed the possibility of introducing memory in the analysis of HMM and provided further readings for interested readers.

Bibliography

[1] Introduction to GHMMs: www.cs.tau.ac.il/~rshamir/algmb/00/scribe00/html/lec07/node28.html.

[2] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis.* eleventh edition, 2006.