6.047/6.878 Lecture 06 : Hidden Markov Models I

Sumaiya Nazeen (Sep 25, 2012)
Chrisantha Perera (Sep 27, 2011)
Gleb Kuznetsov, Sheida Nabavi (Sep 28, 2010)
Elham Azizi (Sep 29, 2009)

October 12, 2012

Contents

List of Figures

1 Introduction

Hidden Markov Models (HMMs) are some of the most widely used methods in computational biology. They allow us to investigate questions such uncovering the underlying model behind certain DNA sequences. By representing data in rich probabilistic ways, we can ascribe meaning to sequences and make progress in endeavors including, but not limited to, Gene Finding.

This lecture is the first of two on HMMs. It covers Evaluation and Parsing. The next lecture on HMM's will cover Posterior Decoding and Learning with an eventual lead into the Expectation Maximization (EM) algorithm. We will eventually cover both supervised and unsupervised learning. 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.

We find many parallels between HMMs and sequence alignment. Dynamic programming lies at the core of both methods, as solutions to problems can be viewed as being composed of optimal solutions to sub-problems.

2 Modeling

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 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.

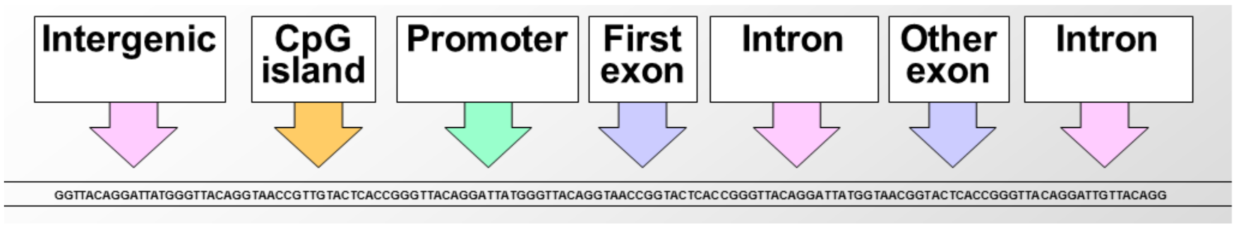

Intergenic
CpG island
Promoter
First exon
Intron
Other exon
Intron
GGTTACAGGATTATGGGTTACAGGTAACCGTTGTACTCACCGGGTTACAGGATTATGGGTTACAGGTAACCGGTACTCACCGGGTTACAGGATTATGGTAACGGTACTCACCGGGTTACAGGATTGTTACAGG

Figure 1: Modeling biological sequences

Building this framework will give us the ability to:

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

2.2 Why probabilistic sequence modeling?

- Biological data is noisy.
- 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.

3 Motivating Example: The Dishonest Casino

3.1 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:

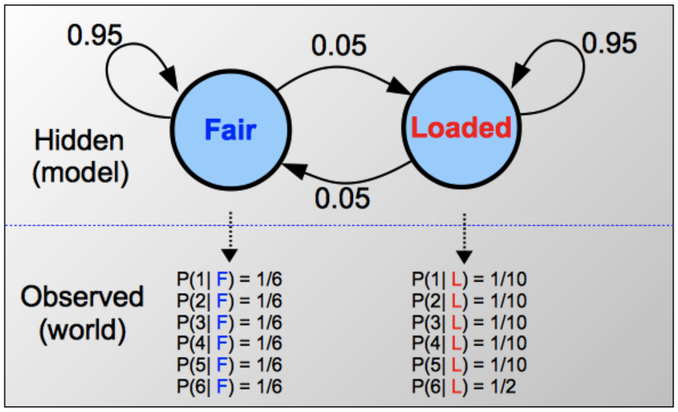


Figure 2: State of a casino die represented by a Hidden Markov model

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.

3.2 Staying in touch with biology: An analogy

For comparison, the figure 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:

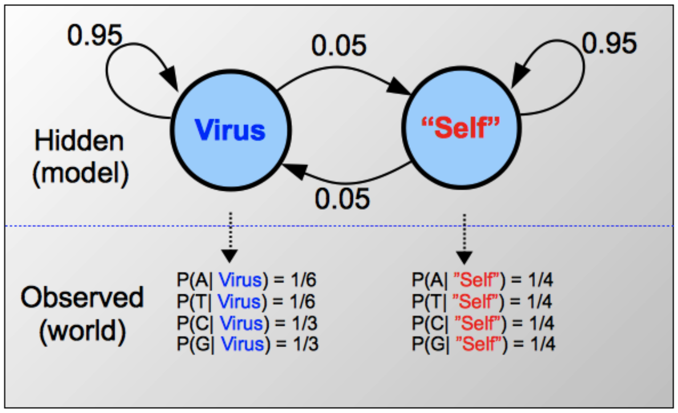


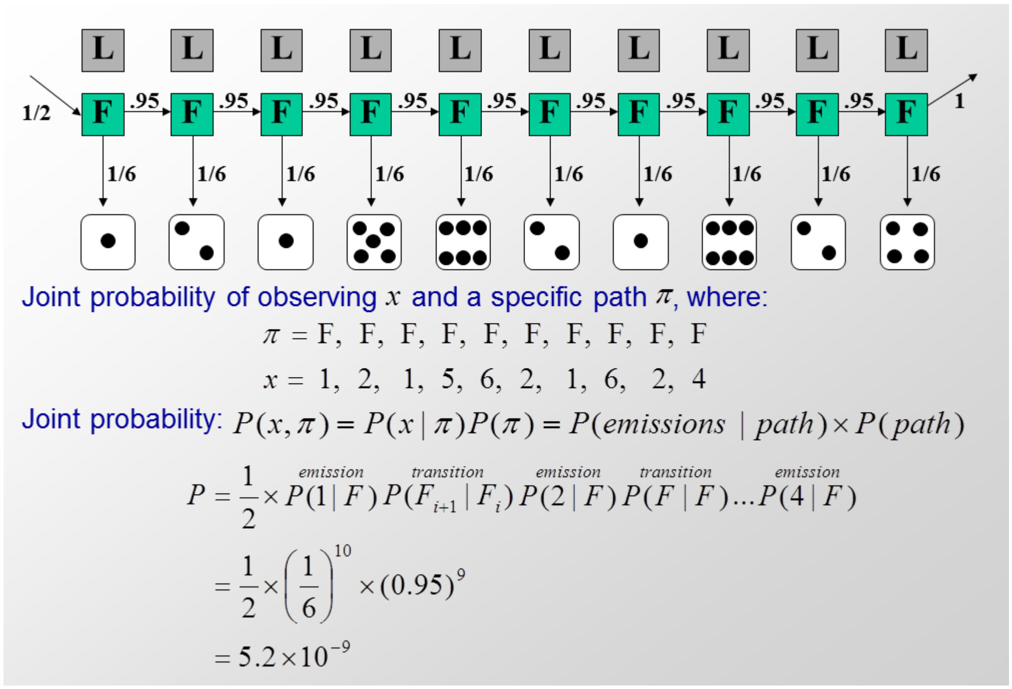
Figure 3: Potential DNA sources: viral injection vs. normal production

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.

3.3 Running the Model

Let's look at a particular sequence of rolls and pick for the underlying model the two extreme cases: 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 run the model on each to understand the implications. We'll build up to introducing HMMs by first trying to understand the likelihood calculations from the standpoint of fundamental probability principles of independence and conditioning.



Joint probability of observing x and a specific path π, where:

$$\pi = F,\ F,\ F,\ F,\ F,\ F,\ F,\ F,\ F,\ F$$

$$x = 1,\ 2,\ 1,\ 5,\ 6,\ 2,\ 1,\ 6,\ 2,\ 4$$

Joint probability: $P(x,\pi) = P(x \mid \pi)P(\pi) = P(emissions \mid path) \times P(path)$

$$P = \frac{1}{2} \times \overset{emission}{P(1 \mid F)}\, \overset{transition}{P(F_{i+1} \mid F_i)}\, \overset{emission}{P(2 \mid F)}\, \overset{transition}{P(F \mid F)} \ldots \overset{emission}{P(4 \mid F)}$$

$$= \frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times (0.95)^9$$

$$= 5.2 \times 10^{-9}$$

Figure 4: Running the model: probability of a sequence, given path consists of all fair dice

In the first case, where we assume the dealer is always using a fair die, we calculate the probability as shown in Figure 4. The product term has three components: 1/2 - the probability of starting with the fair die model, $(1/6)^{10}$ - the probability of the roll given the fair model (6 possibilities with equal chance), and lastly $(0.95)^9$ - the transition probabilities that keep us always in the same state.

The opposite extreme, where the dealer always uses a loaded die, 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 based on the assumption of a loaded die underlying model, whiletwo rolls of 6 have a probability of 1/2 of occurring. Again we multiply all of these probabilities together according to principles of independence and conditioning. This is shown in Figure 5.

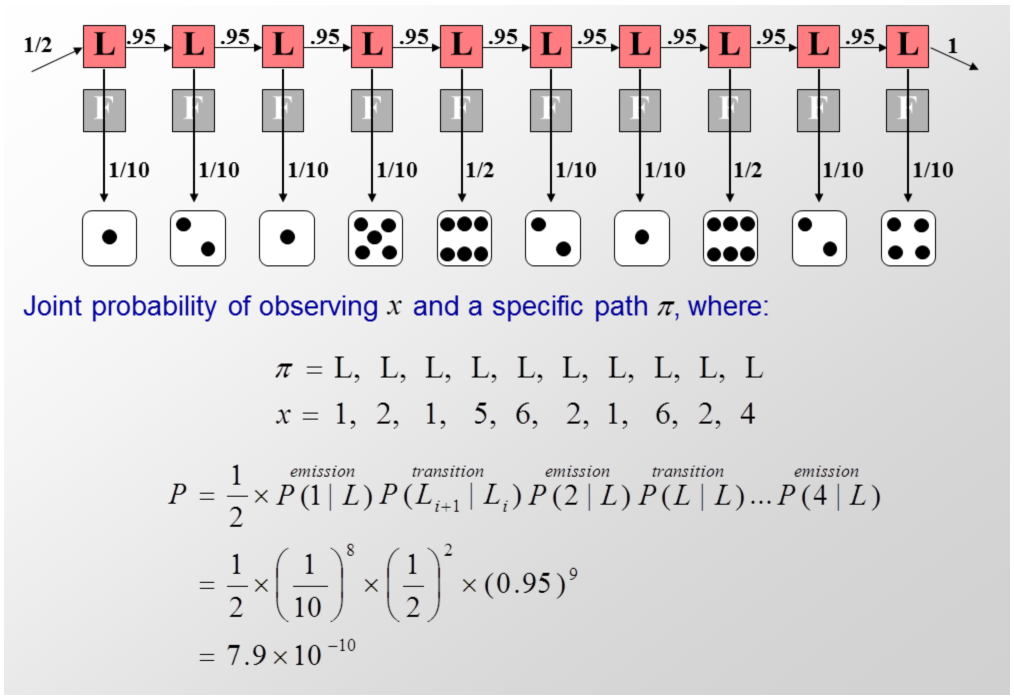
Finally, in order to make sense of the likelihoods, we compare them by calculating a likelihood ratio, as shown in Figure 6.

We find that, it is significantly more likely for the sequence to result from a fair die. Now we step back and ask, does this make sense? Well, of course. Two occurrences of rolling 6 in ten rolls doesn't seem out of the ordinary, so our intuition matches the results of running the model.

3.4 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:

Again, we can calculate the likelihood using fundamental probabilistic methods. From Figure 7, we can see that, our guessed path is far less likely than the previous two cases. Now, it begins to dawn on us that we will need more rigorous 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.

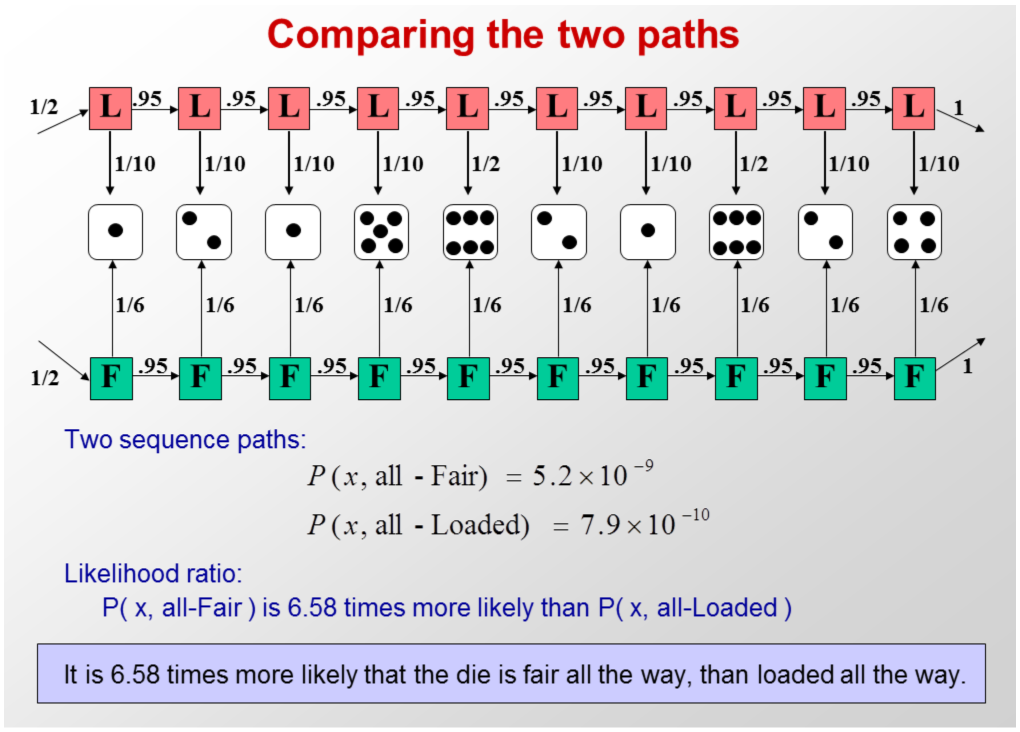


Joint probability of observing x and a specific path π, where:

$$\pi = \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L},\ \mathrm{L}$$
$$x = 1,\ 2,\ 1,\ 5,\ 6,\ 2,\ 1,\ 6,\ 2,\ 4$$

$$P = \frac{1}{2} \times \overset{emission}{P(1 \mid L)}\, \overset{transition}{P(L_{i+1} \mid L_i)}\, \overset{emission}{P(2 \mid L)}\, \overset{transition}{P(L \mid L)} \ldots \overset{emission}{P(4 \mid L)}$$
$$= \frac{1}{2} \times \left(\frac{1}{10}\right)^8 \times \left(\frac{1}{2}\right)^2 \times (0.95)^9$$
$$= 7.9 \times 10^{-10}$$

Figure 5: Running the model: probability of a sequence, given path consists of all loaded dice



Comparing the two paths

1/2
L .95 L .95 L .95 L .95 L .95 L .95 L .95 L .95 L .95 L
1
1/10 1/10 1/10 1/10 1/2 1/10 1/10 1/2 1/10 1/10
1/6 1/6 1/6 1/6 1/6 1/6 1/6 1/6 1/6 1/6
1/2
F .95 F .95 F .95 F .95 F .95 F .95 F .95 F .95 F .95 F
1

Two sequence paths:

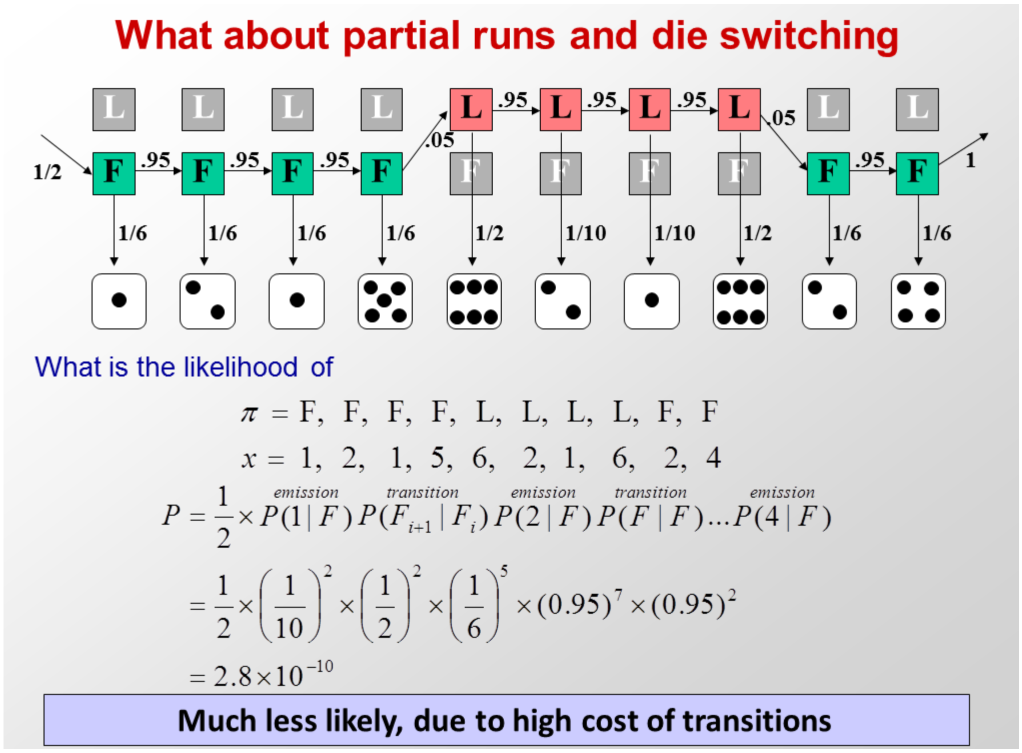
$$P(x, \text{all - Fair}) = 5.2 \times 10^{-9}$$
$$P(x, \text{all - Loaded}) = 7.9 \times 10^{-10}$$

Likelihood ratio:

P(x, all-Fair) is 6.58 times more likely than P(x, all-Loaded)

It is 6.58 times more likely that the die is fair all the way, than loaded all the way.

Figure 6: Comparing the two paths: all-Fair vs. all-loaded.



What about partial runs and die switching

What is the likelihood of

$$\pi = \text{F, F, F, F, L, L, L, L, F, F}$$
$$x = 1,\ 2,\ 1,\ 5,\ 6,\ 2,\ 1,\ 6,\ 2,\ 4$$
$$P = \frac{1}{2} \times \overset{\text{emission}}{P(1|F)}\, \overset{\text{transition}}{P(F_{i+1}|F_i)}\, \overset{\text{emission}}{P(2|F)}\, \overset{\text{transition}}{P(F|F)} \ldots \overset{\text{emission}}{P(4|F)}$$
$$= \frac{1}{2} \times \left(\frac{1}{10}\right)^2 \times \left(\frac{1}{2}\right)^2 \times \left(\frac{1}{6}\right)^5 \times (0.95)^7 \times (0.95)^2$$
$$= 2.8 \times 10^{-10}$$

Much less likely, due to high cost of transitions

Figure 7: Partial runs and die switching

4 Formalizing Markov Chains and HMMS

4.1 Markov Chains

A Markov Chain reduces a problem space to a finite set of states and the transition probabilities between them. At every time step, we observe the state we are in and simulate a transition, independent of how we got that state. More formally, a Markov Chain consists of:

- A set of states, Q.
- A transition matrix, A whose elements correspond to the probabilities of transition from state i to state j.
- A vector of initial state probabilities , p.

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:

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

Therefore, the columns of A have to sum up to 1. In this way, the probability of the sequence can be decomposed into:

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

$P(x_1)$ can also be calculated from the transition probabilities: If we multiply the initial state probabilities at time $t = 0$ by the transition matrix, we get the probabilities of states at time $t = 1$ and therefore we also have them for time $t = n$.

4.2 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 consists of the following parameters:

- A series of states, Q.
- A transition matrix, A: For each s, t, in Q the transition probability is: $a_{st} = P(x_i = t|x_{i-1} = s)$
- A vector of initial state probabilities , p.
- Set of observation symbols, V, for example {A, T, C, G} or, 20 amino-acids or utterances in human language.
- 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 memory-less-ness is inherited from Markov Models: the emissions and transitions are only dependent on the current state and not on the past history.

5 Back to Biology

Now that we have formalized HMMs, we want to use them HMMs in solving 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.

5.1 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 both regions could have emited it at different probabilities. 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). It was noted again that Markov chains are absolutely memory-less, this means that for example if you have lost in a Casino in the last 7 days, it won't mean that you would most probably win today. This is also true in the example of rolling a die where you might have repetitions of a number.

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 emissions transition probabilities (by dynamic programming).

HMMs are generative models, in that an HMM gives the probability of emission given a state (with Bayes' Rule), essentially telling you how likely the state is to generate those sequences. So we can always run a generative model for transition between states and start anywhere. In Markov Chains, the next state

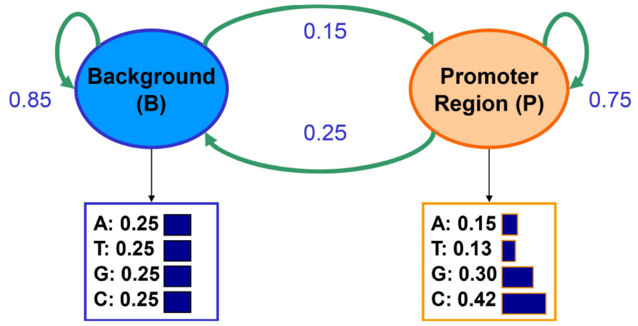
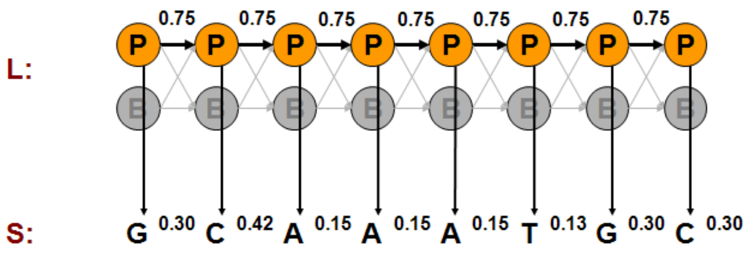


Figure 8: HMMS as a generative model for finding GC-rich regions.

will give different outcomes with different probabilities. No matter which the next state is, at that next state, the next symbol will still come out with different probabilities. HMMs are similar: You can pick an initial state based on initial probability vector. In the example above, you pick B since there are more background than promoter. Then draw an emission from the P(X—B). Since all are 0.25, you can pick any symbol, for example G. Given this emission, the probability of the next transition state does not change. So we have transition to B again is with probability 0.85 and to P with 0.15 so we go with B 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. For the example above we have the probability of three different paths computed as in Figures 9, 10, and 11.

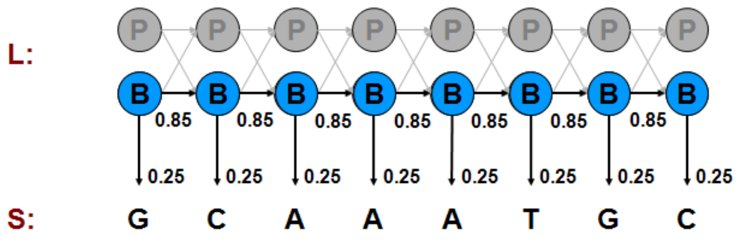


$$\mathbf{P(x,\pi)=a_P*e_P(G)*a_{PP}*e_P(G)*a_{PP}*e_P(C)*a_{PP}*e_P(A)*a_{PP}*\ldots}$$
$$\mathbf{=a_p*(0.75)^7*(0.15)^3*(0.13)^1*(0.30)^2*(0.42)^2}$$
$$\mathbf{=9.3*10^{-7}}$$

Figure 9: Probability of seq, path if all promoter

Now the question is, what path is most likely to generate the given sequence?

One brute force approach may be looking at all paths, trying all possibilities, and calculating their joint probability $P(x,\pi)$. 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 11, $P(x,\pi) = 6.7 \times 10^{-7}$; 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. In the coming sections we will see how can we mathematically support our intuition.



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

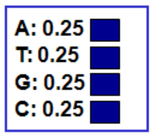
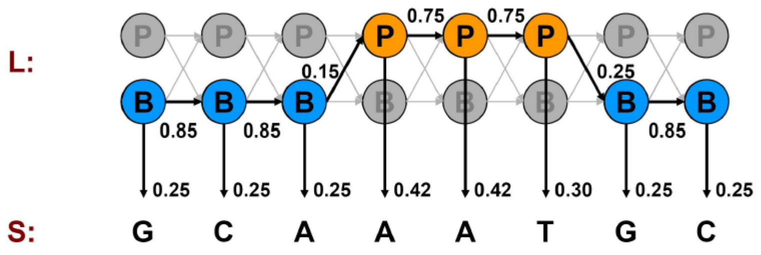


Figure 10: Probability of seq, path if all background



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

Figure 11: Probability of seq, path sequence if mixed

5.2 Application of HMMs in Biology

HMMs are used in answering many interesting questions. Some biological application of HMMs are summarized in the table shown in Figure 12.

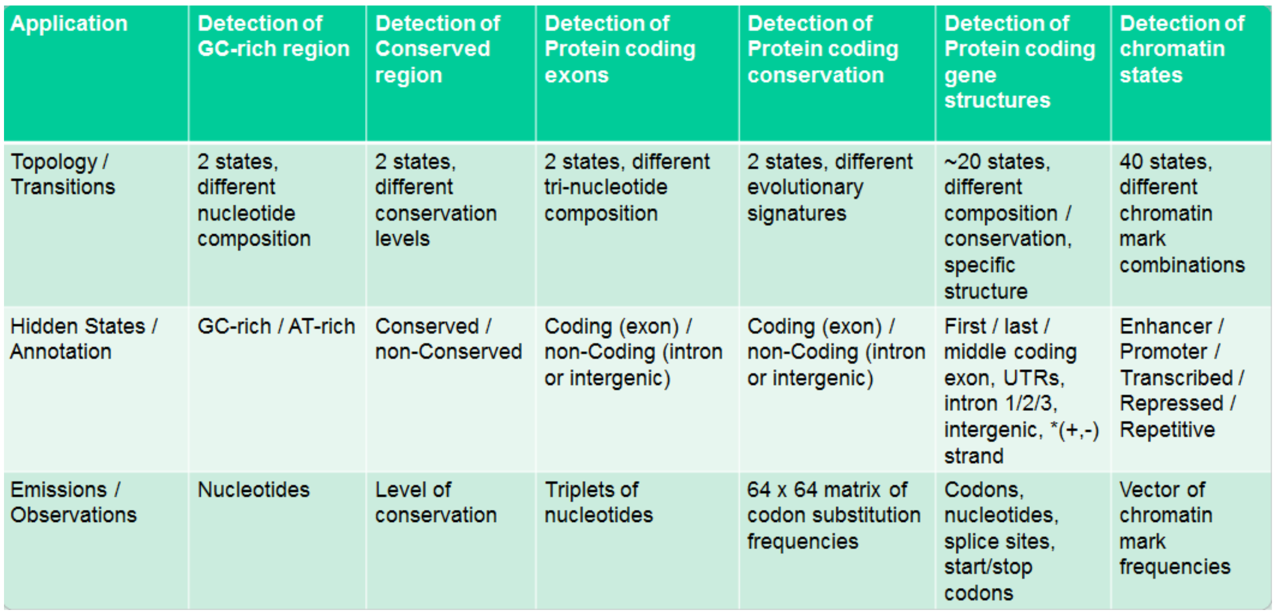
6 Algorithmic Settings for HMMs

We use HMMs for three types of operation: **scoring**, **decoding**, and **learning**. We 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, the most likely path. However in all paths operations we are interested in a sequence of observations or emissions regardless of its corresponding paths.

6.1 Scoring

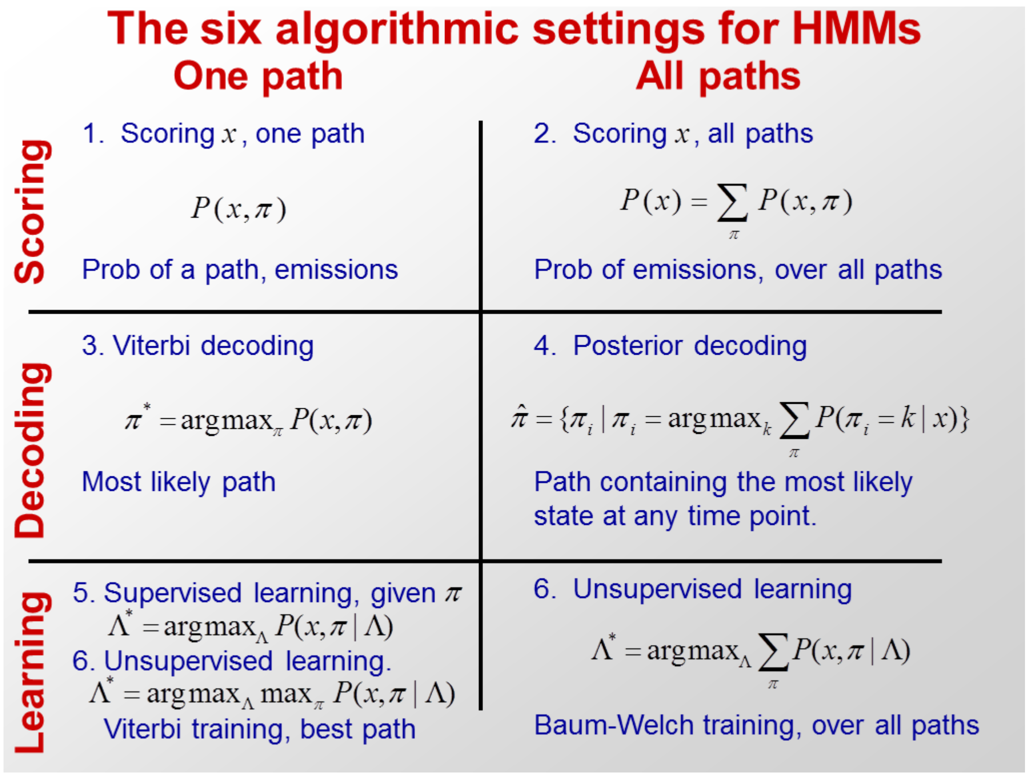
6.1.1 Scoring over a single path

For a single path we define the Scoring problem as follows:



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 12: Some biological applications of HMM



The six algorithmic settings for HMMs

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

Figure 13: The six algorithmic settings for HMMS

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

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.

6.1.2 Scoring over all paths

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

- **Input:** A sequence of observations $x = x_1 x_2 \ldots x_n$ generated by an HMM $M(Q, A, p, V, E)$.
- **Output:** Joint probability, $P(x, \pi)$ of observing x over all possible hidden state sequences, πs.

we do all operations on all possible paths and states and score the sequence over all paths π using the following formula,

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

In this case the score is calculated for just a given sequence of observations or emissions regardless of the paths. We use this score when we are interested in knowing the likelihood of particular sequence for a given HMM.

6.2 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 $x = x_1 x_2 \ldots x_N$ generated by an HMM $M(Q, A, p, V, E)$.
  - Output: The most probable path of states, $\pi^* = \pi_1^* \pi_2^* \ldots \pi_N^*$
- **Decoding over all paths:**
  - Input: A sequence of observations $x = x_1 x_2 \ldots x_N$ generated by an HMM $M(Q, A, p, V, E)$.
  - Output: The path of states, $\pi^* = \pi_1^* \pi_2^* \ldots \pi_N^*$ that contains the most likely states at any 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 - there are exponential number of paths and using such a brute force search for the maximum likelihood among all the paths is very time consuming and impractical. To solve this problem **Dynamic Programming** can be used. Let us formulate the problem in the dynamic programming approach.

Through decoding we would like to find out the most likely sequence of states based on the observation. For the decoding operation, the model parameters $e_i s$ (the emission probabilities given their states) and $a_{ij}s$, (the state transition probabilities) are given. Also, the sequence of emissions x is given. The goal is to find the sequence of hidden states, π^*, which maximizes the joint probability of a given emission and its path $P(x, \pi)$, i.e.,

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

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:

- Best path to previous state
- Best transition from previous state to this state
- 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, a dynamic programming algorithm, is commonly used to obtain the best path.

6.2.1 Most probable state path: the Viterbi algorithm

Suppose, the probability $v_k(i)$, of the most likely path ending at state k at position (or time instance) i in the path (which can be shown as $\pi_i = k$) is known for all the states k. Then we can compute this probability at time $i+1$, as follows:

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

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 $i-1$, now 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 penalty of transition to the new states (transition probability), and
- 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 we studied in previous lectures.

The steps of the Viterbi algorithm **??** are summarized below:

1. Initialization ($i = 0$): $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$.
2. Recursion ($i = 1 \ldots N$): $v_l(i) = e_l(x_i) max_k(a_{kl}v_k(i-1))$; $ptr_i(l) = \arg\max_k(a_{kl}v_k(i-1))$.
3. Termination: $P(x, \pi^*) = max_k(v_k(N)a_{k0})$; $\pi^*_N = \arg\max_k(v_k(N)a_{k0})$.
4. Traceback ($i = N \ldots 1$): $\pi^*_{i-1} = ptr_i(\pi^*_i)$.

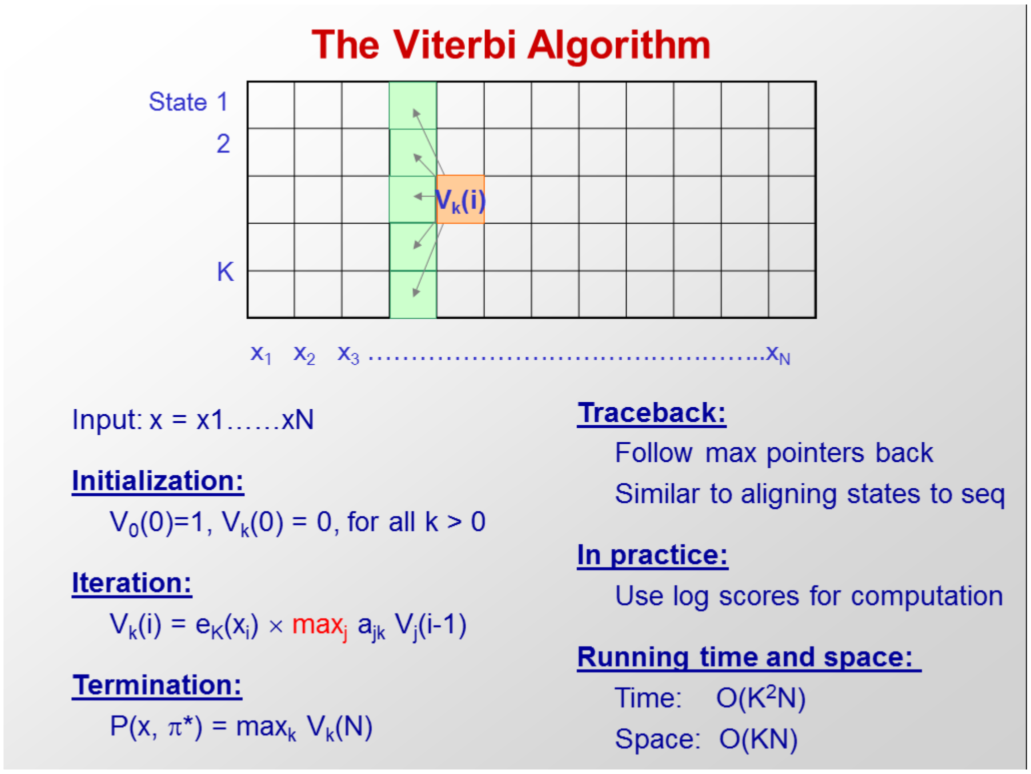
As we can see in Figure 14, we fill the matrix from left to right and trace back. Each position in the matrix has K states to consider and there are KN cells in the matrix, so, the required computation time is $O(K^2N)$ and the required space is $O(KN)$ to remember the pointers. Note that, the running time has reduced from exponential to polynomial.

6.3 Evaluation

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

Let us formally define the Evaluation problem first.

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


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

The Viterbi Algorithm

Input: x = x1……xN

Initialization:
$V_0(0)=1$, $V_k(0) = 0$, for all $k > 0$

Iteration:
$V_k(i) = e_K(x_i) \times \max_j a_{jk} V_j(i-1)$

Termination:
$P(x, \pi^*) = \max_k V_k(N)$

Traceback:
Follow max pointers back
Similar to aligning states to seq

In practice:
Use log scores for computation

Running time and space:
Time: $O(K^2N)$
Space: $O(KN)$

Figure 14: The Viterbi algorithm

We know that, if we are given an HMM, we can generate a sequence of length n easily 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 there are too many paths (exponential number of paths). Each path has an associated probability. Some paths are more likely, some are less likely, and we need to sum them up all. One approach may be using just the Viterbi path and ignoring the others. We already have shown how to obtain the Viterbi path. But its probability is very small; it is just a small fraction of the probability mass of all possible paths. It is a good approximation only if it has high probability density. In other cases, it will give us an inaccurate approximation. So, the correct approach calculating the exact sum iteratively by using dynamic programming. The algorithm that does this is known as **Forward Algorithm**.

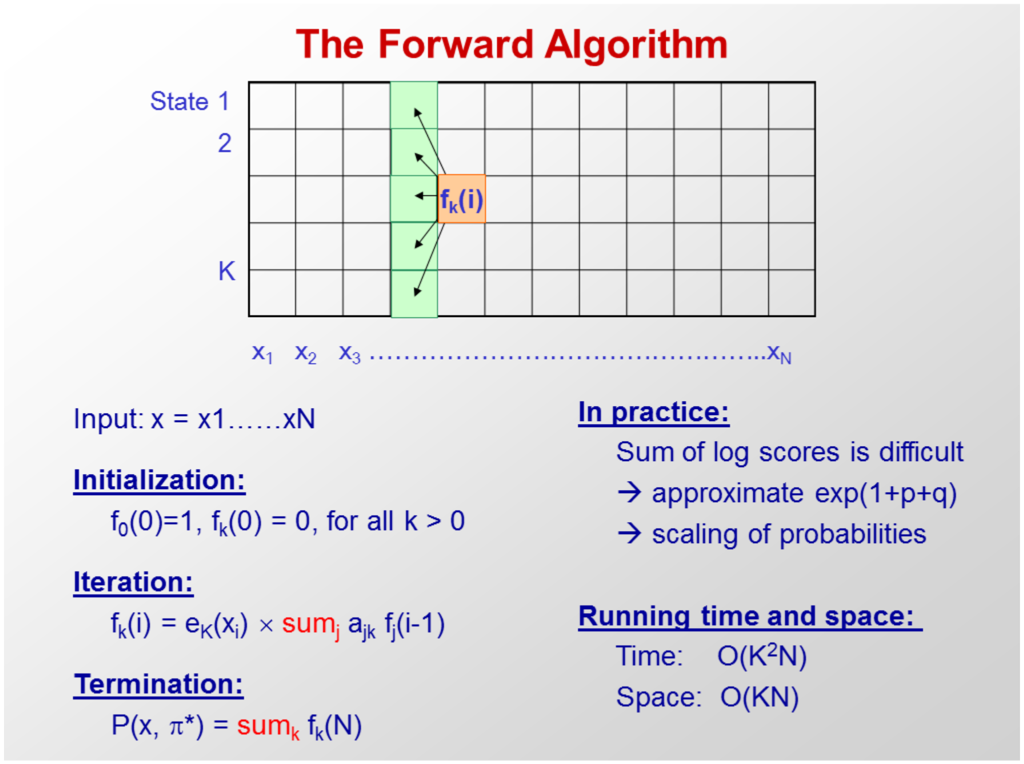
6.3.1 The Forward Algorithm

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

$$\begin{aligned}
f_l(i) &= P(x_1 \ldots x_i, \pi = l) && (1)\\
&= \sum_{\pi_1 \ldots \pi_{i-1}} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1}, \pi_i = l) e_l(x_i) && (2)\\
&= \sum_k \sum_{\pi_1 \ldots \pi_{i-2}} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1} = k) a_{kl} e_l(x_i) && (3)\\
&= \sum_k f_k(i-1) a_{kl} e_l(x_i) && (4)\\
&= e_l(x_i) \sum_k f_k(i-1) a_{kl} && (5)\\
& && (6)
\end{aligned}$$

The full algorithm[2] is summarized below:

- Initialization $(i = 0)$: $f_0(0) = 1$, $f_k(0) = 0$ for $k > 0$.
- Recursion $(i = 1 \ldots N)$: $f_l(i) == e_l(x_i) \sum_k f_k(i-1) a_{kl}$.
- Termination: $P(x) == \sum_k f_k(N) a_{k0}$



The Forward Algorithm

Input: x = x1……xN

Initialization:

$f_0(0)=1$, $f_k(0) = 0$, for all k > 0

Iteration:

$f_k(i) = e_K(x_i) \times \text{sum}_j\, a_{jk}\, f_j(i-1)$

Termination:

$P(x, \pi^*) = \text{sum}_k\, f_k(N)$

In practice:

Sum of log scores is difficult

→ approximate exp(1+p+q)

→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$

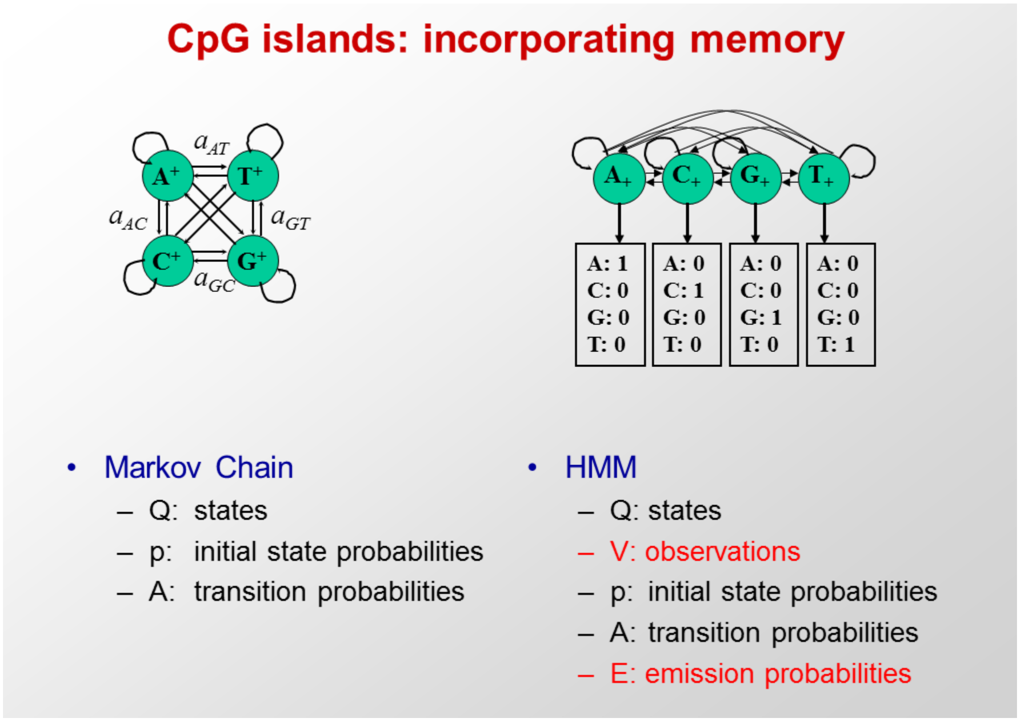
Space: $O(KN)$

Figure 15: The Forward algorithm

From Figure 15, 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 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, the thing that we can do to store additional information is to augment the number of states. Now, look back to the biological example we gave in Section 5.1. In our model, state emissions were dependent only on 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. This is illustrated in Figures 16 and .17.



CpG islands: incorporating memory

- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities
- HMM
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities

Figure 16: CpG Islands - Incorporating Memory

Here we show an HMM for CpG islands. Recall that, in our first biological example we used only two states - Background (non GC-rich) and Promoter (GC-rich) region, but here we have eight states to remember what nucleotide we have seen before, so that we can count the di-nucleotide frequencies. More discussion on this will follow in the next lecture.

8 Further Reading

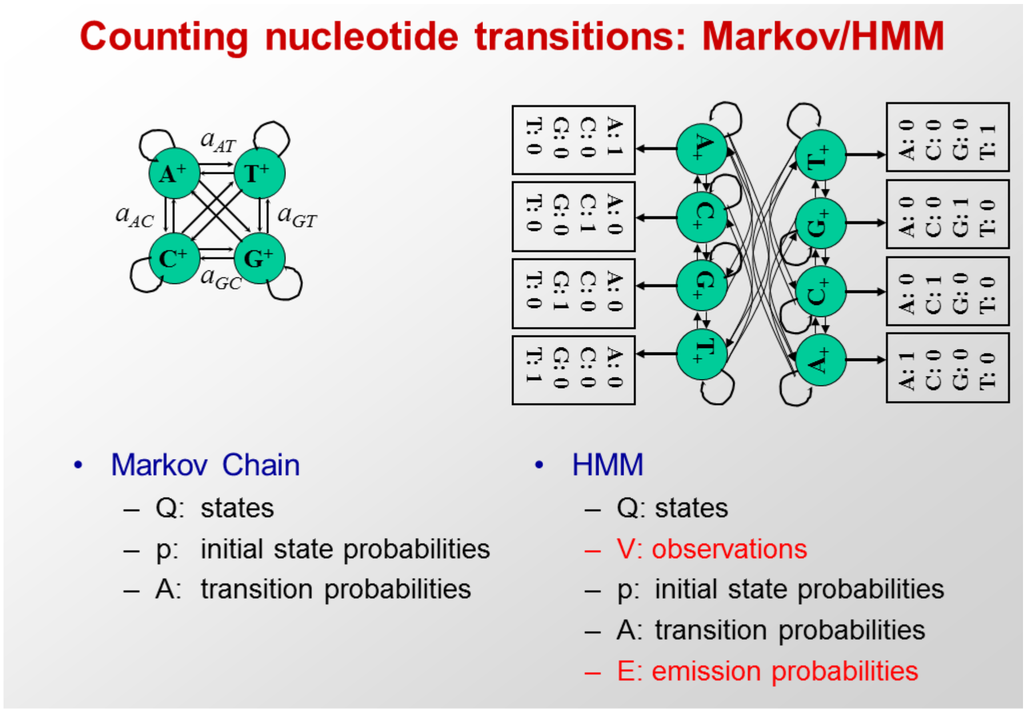
8.1 Length Distributions of States and Generalized Hidden Markov Models

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

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



Counting nucleotide transitions: Markov/HMM

- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities
- HMM
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities

Figure 17: Counting Nucleotide Transitions

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 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.

8.2 Conditional random fields

Conditional random field model is an alternative to HMMs. It is a discriminative undirected probabilistic graphical model. 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

9 Current Research Directions

10 Tools and Techniques

11 What Have We Learned?

References

[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.