Hidden Markov Models

Christopher Lee

Department of Chemistry and Biochemistry, UCLA

Announcements

- Reading assignment for today's lecture posted on CCLE.
- New mission training posted in CCLE Week 6.
- New programming project posted on Stepik (access from CCLE via Week 2 Stepik link).
- Today we begin looking at different bioinformatics application problems, such as sequence analysis, alignment, and evolution.
- We start with an important set of models for studying any kind of "sequential" problem, so-called *Hidden Markov Models*.

Fair or Loaded Dice?

Say a casino switches back and forth between a fair dice and a loaded dice (one outcome is favored). Look at the following sequence of rolls and predict when it's F vs. L, and state briefly your principle for how you can do this:

```
45313265124563666463163666316232645523626666666
```

Fair or Loaded Dice? Answer

Some segments of the sequence appear to have a lot more sixes than expected from a fair dice. In general, we are looking for *differences in the observation probabilities* in different parts of the sequence.
Here is the actual dice sequence:

```
4531326512456366646316366631623264552362666666
FFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLL
```

Your Mission: Casino Royalle Reloaded!

James Bond has to return to Casino Royalle to bankrupt another Bond villain by gambling, but M informed him that the casino is secretly switching between fair vs. loaded dice.
Your mission is to tell Bond when the loaded dice are put in play, and to give an exact confidence for that call, at each roll of the dice, simply by observing the rolls.

7 days to complete your Mission

By:

- **Wednesday**: complete crash course on what you need to know to solve this
- **Friday**: complete your mathematical solution of this problem.
- **Monday**: call the shots for James Bond at the Casino Royalle, by programming your solution and running it on a sequence of observed rolls.

Hidden State Sequences

- How to infer hidden state sequences from sequences of observations?
- How to measure the uncertainty of our inferences?
- How to train such Bayesian models from real data?

Markov Chains

- Simplest possible information graph structure: just a linear chain $X \to Y \to Z \to \cdots$.
- Obeys Markov property: joint probability factors into conditional probabilities that only depend on the *previous* variable

$$p(X, Y, Z, \cdots) = p(X)p(Y|X)p(Z|Y)\cdots$$

- Simple but useful for a remarkable range of problems in many fields.
- Widely used in bioinformatics: detecting features in sequences; sequence alignment; modeling evolution etc.

Markov = Locality

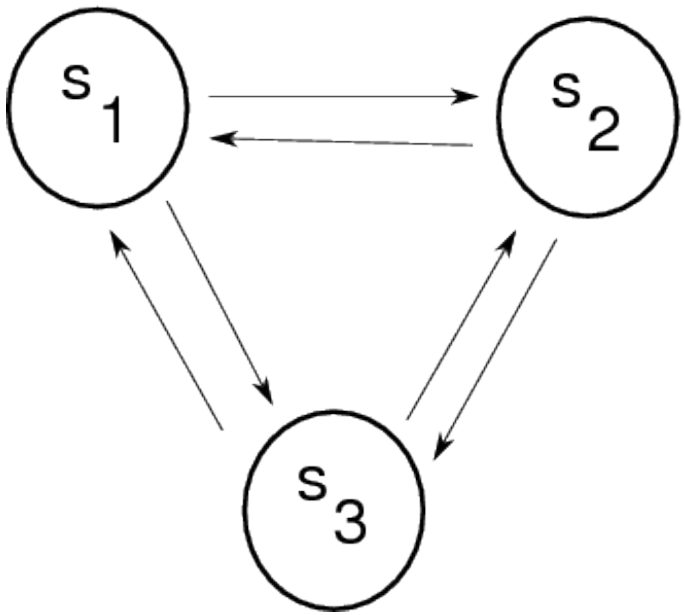
- Applicable to any sequential process with no "memory".
- I.e. what happens *next* depends only on *now* and not on anything in the *past*.
- Purely local algorithm means minimum computational complexity.

Discrete States, Discrete Time Steps

- Note that this formulation ("a sequence of random variables $X_1, X_2, \ldots X_n$") typically assumes a random variable that switches between *discrete* values, which are often referred to as its possible "states".
- For random variable X, call its possible states $\{s_1, s_2, \ldots s_m\}$. Interpret Markov chain $X_1, X_2, \ldots X_n$ as value of X at discrete time steps $t = 1, 2, \ldots n$.

Homogeneous Markov Chains

- If the set of allowed states s_i, and their transition probabilities $p(X_t = s_j | X_{t-1} = s_i)$ are the same for all X_t, then we call this a *homogeneous Markov chain*, which is representable by its *state graph*. Note that a "self-edge" must be explicitly drawn if a state can "transition to itself", i.e. remain in the same state.
- Here we show three states, each of which must transition to one of the other two:



Markov Model State Graphs

- Markov chains have a generic information graph structure: just a linear chain $X \to Y \to Z \to \cdots$.
- The more interesting aspect of how to build a Markov model is deciding *what states* it consists of, and what *state transitions* are allowed.
- This is represented by its state graph.
- Do not mix this up with an information graph!

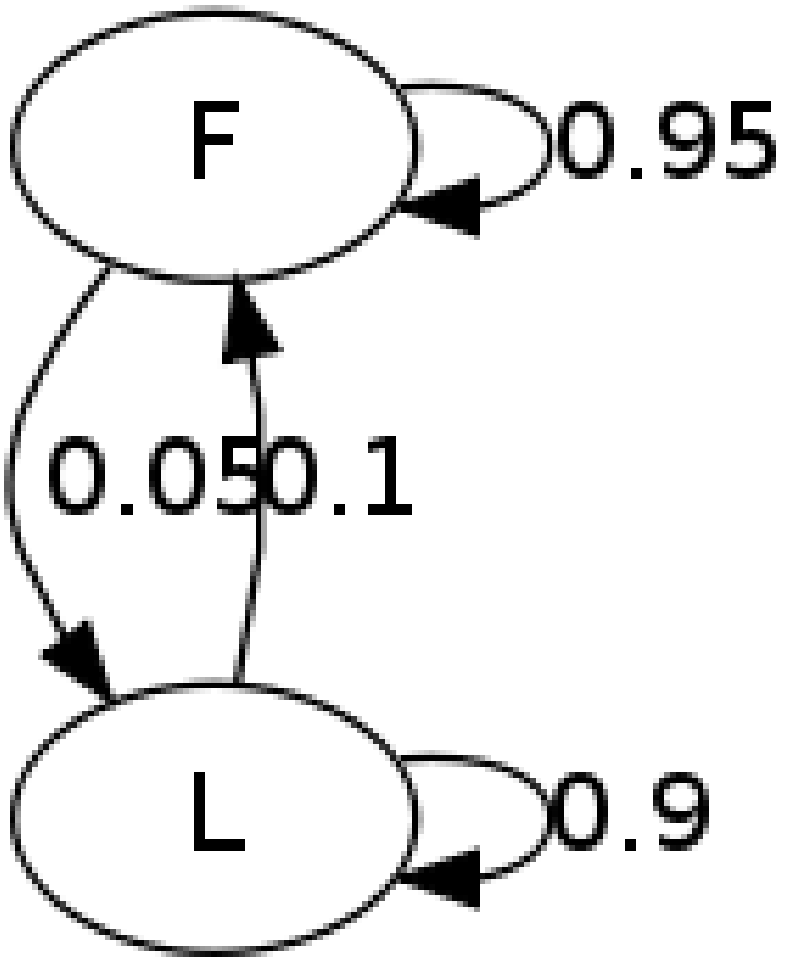
Example: Occasionally Dishonest Casino

In a dice game, a casino uses a fair dice (all rolls have equal probability) and a loaded dice, but switches between them randomly:

- after each roll of the fair dice, they switch to the loaded dice with 5% probability.
- after each roll of the loaded dice, they switch to the fair dice with 10% probability.

What's the state graph?

Example: Occasionally Dishonest Casino Answer



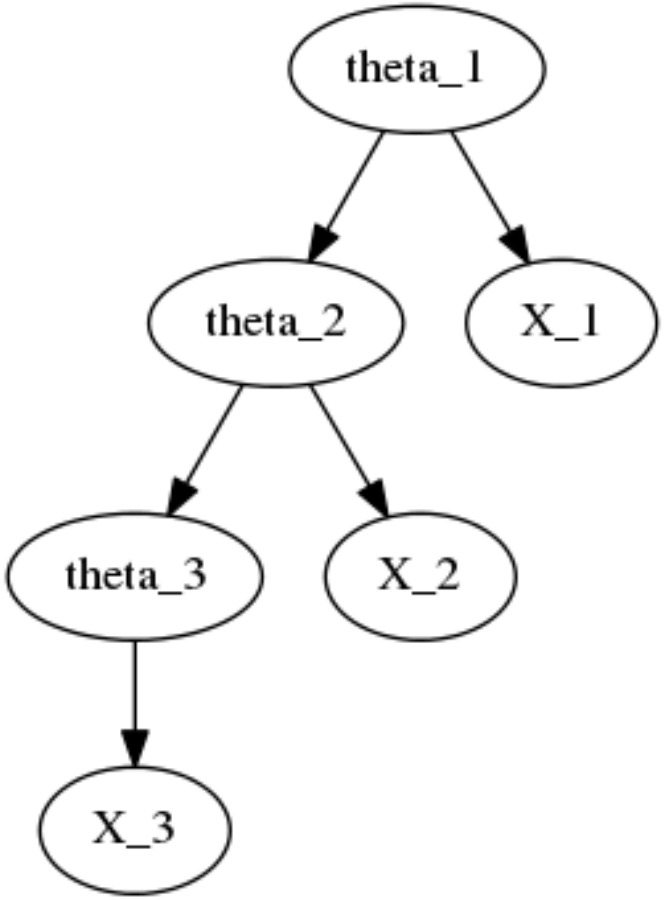
Occasionally Dishonest Casino Info Graph

Say we observe a series of dice roles at a casino that is following the "cheating" state graph in the previous problem.

- Draw an information graph structure representing a model of how the casino generates the sequence of observed dice rolls. (use our standard graph notation, e.g. `alpha -> beta;` ; and draw at least 2 observations, to make the sequence structure of your graph clear).
- Briefly define your variables.

Occasionally Dishonest Casino Info Graph Answer

We have hidden variables $\Theta_1, \Theta_2, \Theta_3, \ldots$ (F vs. L), and observations (dice rolls) $X_1, X_2, X_3, \ldots$:



Hidden Markov Models

- Consider a Markov chain of hidden variables $\Theta_1 \rightarrow \Theta_2 \rightarrow \cdots \rightarrow \Theta_n$ that each emit one observable variable X_t with some likelihood $p(X_t|\Theta_t)$, called the *emission probability*.
- **Problem**: Given a sequence of observations $\vec{X} = (X_1, X_2, ...X_n)$, infer the hidden variable sequence and its posterior probability, i.e. the hidden state path $\vec{\theta}^*$ such that $p(\vec{\theta}^*|\vec{X}) \geq p(\vec{\Theta}|\vec{X})$ for all possible paths $\vec{\Theta}$

Path vs. State Posteriors

Say we have an HMM with N variables each with M states. Now consider the very simplest possible case: all states have equal prior probability; all transition probabilities are equal; and all emission probabilities are equal. For a given observation sequence $\vec{X}$ and specific hidden state path $\vec{\theta}$, what values would you expect for

- $p(\vec{\theta}|\vec{X})$?
- $p(\theta_t = s_i|\vec{X})$?

Path vs. State Posteriors Answer

All possible paths through the HMM are equally probable. So we expect:

$$p(\vec{\theta}|\vec{X}) = M^{-N}$$

because there are M^N possible paths; and

$$p(\theta_t = s_i|\vec{X}) = \frac{1}{M}$$

because θ_t has M possible states.

Path vs. State Posteriors Errors

- Some people seem to think that there are N*M possible paths. No; the number goes up *exponentially* as M^N. This is a crucial principle to understand, e.g. the probability of a path goes down exponentially with its length -- even if it is the Viterbi maximum probability path!
- Some people answered this in terms of generic Markov chain calculations, e.g. transition, emission, etc. Implies they didn't think about the specific question, which eliminated all those issues by giving the simplest possible situation.
- Some people may not have understood that the problem is asking you about 1. the probability of an entire path (a specific path from beginning to end of the HMM); 2. the probability of a single hidden state. Implies they don't understand the basic nomenclature, since the question stated these things very explicitly.

How is Casino Royale Cheating?

James Bond and M have two very different models of how the Casino Royale is cheating, which they have encoded as HMMs M_1, M_2. You have a sequence of observations of $\vec{X}$ of the casino's actual dice rolls.

- Explain how you would compute a rigorous likelihood odds ratio for the two models.
- What is the main computational complexity challenge?
- Does the Markov property help solve the computational complexity challenge?

How is Casino Royale Cheating? Answer

We want to get the likelihood odds ratio

$$\frac{p(\vec{X}|M_1)}{p(\vec{X}|M_2)}$$

The computational complexity challenge arises from the need to sum over *all possible hidden state paths* through each HMM, i.e.

$$p(\vec{X}|M_1) = \sum_{\vec{\theta}} p(\vec{X}, \vec{\theta}|M_1)$$

The Markov property can greatly reduce the computational complexity of the calculation, because it means we can factor the summation into terms that we can compute separately.

Breaking up the Summation?

- Say we know $p(X_1, \theta_1)$.
- How would you compute $p(X_1, X_2, \theta_2)$ from that?

To answer this:

- Draw the information graph for these variables.
- Mark the variable(s) whose probability is already known with a check mark.
- Circle the new variable(s) whose probability must be calculated to get the answer.
- Mark the variable(s) that must be eliminated from the joint probability to get the answer, with an X.

Breaking up the Summation? Answer

These problems always have the same answer "Just follow the yellow brick road!" (information graph).

- We already know $p(X_1, \theta_1)$.
- The connection from what we've got, to what we want, is just $\theta_1 \rightarrow \theta_2$.
- this immediately gives us:

$$p(X_1, X_2, \theta_1, \theta_2) = p(X_1, \theta_1) p(\theta_2|\theta_1) p(X_2|\theta_2)$$

- Now all we have to do is get rid of θ_1:

$$p(X_1, X_2, \theta_2) = \sum_{\theta_1} p(X_1, \theta_1) p(\theta_2|\theta_1) p(X_2|\theta_2)$$

Generalizing the Summation?

- Say we know $p(\vec{X}_{[1,t]}, \theta_t)$ and want to extend this "one step" to compute $p(\vec{X}_{[1,t+1]}, \theta_{t+1})$.

Draw this computation on the information graph:

- draw a box around the region of variables whose probability is already known,
- circle the new variable(s) whose probability must be calculated,
- cross out variable(s) that must be eliminated from the joint probability.

Generalizing the Summation? Answer

(draw a picture on the board)

Completing the Summation?

- From this, how would we get $p(\vec{X}, \theta_n)$?
- Can we get $p(\vec{X})$ from this?
- How could you use this to settle M and Bond's disagreement?

Completing the Summation? Answer

- We simply perform the same calculation from each θ to the next:

$$p(\vec{X}_{[1,1]}, \theta_1) \rightarrow p(\vec{X}_{[1,2]}, \theta_2) \rightarrow p(\vec{X}_{[1,3]}, \theta_3) \rightarrow \ldots \rightarrow p(\vec{X}_{[1,n]}, \theta_n)$$

  where the notation $\vec{X}_{[1,t]} = X_1, X_2, \ldots X_t$. The general form of each step is:

$$p(\vec{X}_{[1,t]}, \theta_t) = \sum_{\theta_{t-1}} p(\vec{X}_{[1,t-1]}, \theta_{t-1}) p(\theta_t | \theta_{t-1}) p(X_t | \theta_t)$$

- This is called a **forward probability summation**, because it must be performed in *forward* order, i.e. $\theta_1, \theta_2, \ldots \theta_n$.
- Finally, we just eliminate θ_n:

$$p(\vec{X}) = \sum_{\theta_n} p(\vec{X}_{[1,n]}, \theta_n)$$

- We can use this procedure first using M's model M_1 and second using Bond's model M_2, to get their likelihood odds ratio.

Announcements

- See the new Mission Training in CCLE Week 6
- The new project is up on Stepik... any questions so far?
- quiz solutions will be posted today

Forward Probability Principles

A couple principles are at work here:

- *recursive definition*: in order to have a calculation that we can repeat over and over to solve for all θ_t, we must define an expression for θ_t that can be calculated from the exact same expression for θ_{t-1}. I.e.

$$p(\theta_{t-1}, \vec{X}^{t-1}) \rightarrow p(\theta_t, \vec{X}^t)$$

- *summation to eliminate the "nuisance variable"*: of course, now that we've brought θ_{t-1} into our equation, we need to get rid of it so that all that's left is θ_t. Note that this algorithm is the same as Viterbi except using summation instead of maximization.

Was the Casino Cheating at time t?

James Bond and M are having another violent argument about whether the casino was cheating at a critical moment t in the game. They agree on the HMM model M for how the casino was playing, and the observed dice rolls $X_1, X_2 \ldots X_n$, but disagree about whether the casino was cheating at time t (i.e. $\theta_t = L$).

- define the conditional probability that you would calculate to settle their argument.
- Using a picture of the information graph for these variables, circle the subject variable(s) of this conditional probability, and draw a box around the condition variable(s).

Was the Casino Cheating at time t? Answer

(draw a picture on the board)

A Forward Probability Solution?

Let's see whether we can solve this posterior probability using the forward probability calculation we developed, by drawing it on the information graph:

- draw a box around the variables whose joint probability must be calculated to get $p(\theta_t = L|\vec{X}_{[1,n]})$, and write the mathematical notation for their joint probability.
- draw another box around the variables included in the forward probability for time t, and write the mathematical notation for their joint probability.
- draw a dashed line around the "missing variable(s)" i.e. that are in the first box but not the second.
- write the chain rule for $p(\theta_t = L|\vec{X}_{[1,n]})$ (the first box) in terms of the second box and the third box.

A Forward Probability Solution? Answer

- $p(\theta_t = L, \vec{X}_{[1,n]})$
- $p(\theta_t = L, \vec{X}_{[1,t]})$
- $p(\theta_t = L, \vec{X}_{[1,n]}) = p(\theta_t = L, \vec{X}_{[1,t]})p(\vec{X}_{(t,n]}|\theta_t = L)$

where the notation $\vec{X}_{(t,n]}$ means "the observations after (and not including) time t".

Breaking up the Summation?

- Say we know $p(\vec{X}_{(n-1,n]}|\theta_{n-1}) = p(X_n|\theta_{n-1})$.
- How would you compute $p(\vec{X}_{(n-2,n]}|\theta_{n-2})$ from that?

Once again, draw the information graph, box the variable(s) whose probability is already known, circle the variable(s) whose probability must be calculated, and cross out any variable(s) that must be eliminated from the joint probability.

Breaking up the Summation? Answer

- Because of the Markov property, the connection between these two pieces is just $\theta_{n-2} \rightarrow \theta_{n-1}$.
- this immediately gives us:

$$p(\vec{X}_{(n-2,n]}, \theta_{n-1} | \theta_{n-2}) = p(\theta_{n-1} | \theta_{n-2}) p(X_{n-1} | \theta_{n-1}) p(\vec{X}_{(n-1,n]} | \theta_{n-1})$$

- Now all we have to do is get rid of θ_{n-1}:

$$p(\vec{X}_{(n-2,n]} | \theta_{n-2}) = \sum_{\theta_{n-1}} p(\theta_{n-1} | \theta_{n-2}) p(X_{n-1} | \theta_{n-1}) p(\vec{X}_{(n-1,n]} | \theta_{n-1})$$

Completing the Summation?

- From this, how would we get $p(\vec{X}_{(t,n]}|\theta_t)$?

Completing the Summation? Answer

- We simply perform the same calculation from each θ to the previous:

$$p(\vec{X}_{(n-1,n]}|\theta_{n-1}) \rightarrow p(\vec{X}_{(n-2,n]}|\theta_{n-2}) \rightarrow \ldots \rightarrow p(\vec{X}_{(t,n]}|\theta_t)$$

The general form of each step is:

$$p(\vec{X}_{(t,n]}|\theta_t) = \sum_{\theta_{t+1}} p(\theta_{t+1}|\theta_t) p(X_{t+1}|\theta_{t+1}) p(\vec{X}_{(t+1,n]}|\theta_{t+1})$$

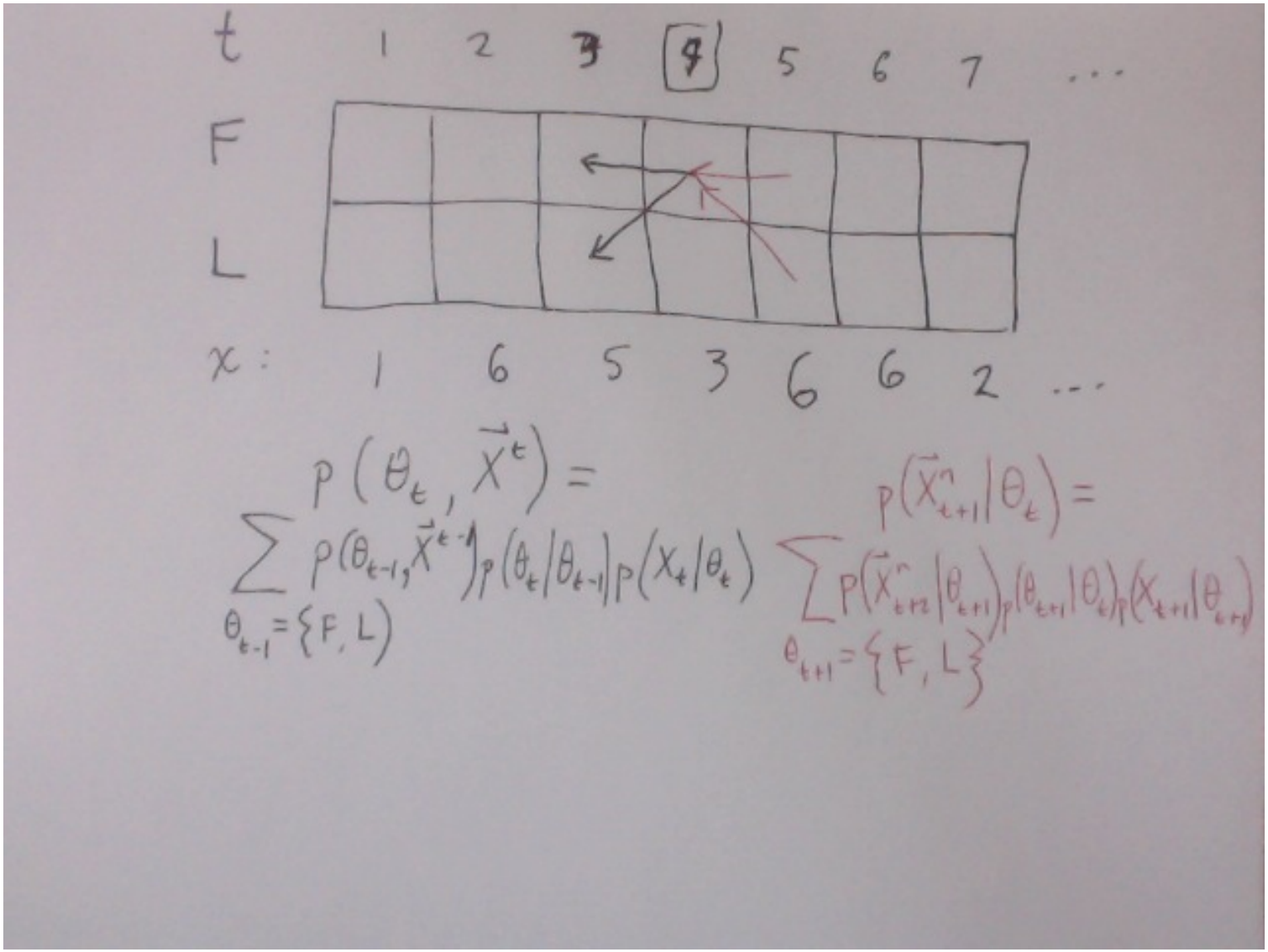
- This is called a **backward probability summation**, because it must be performed in *backward* order, i.e. $\theta_{n-1}, \theta_{n-2}, \ldots \theta_1$.

Draw the Forward-Backward Calculation

For the Occasionally Dishonest Casino, there are only two possible hidden states, F and L. To make the forward-backward algorithm concrete, let's draw the calculation:

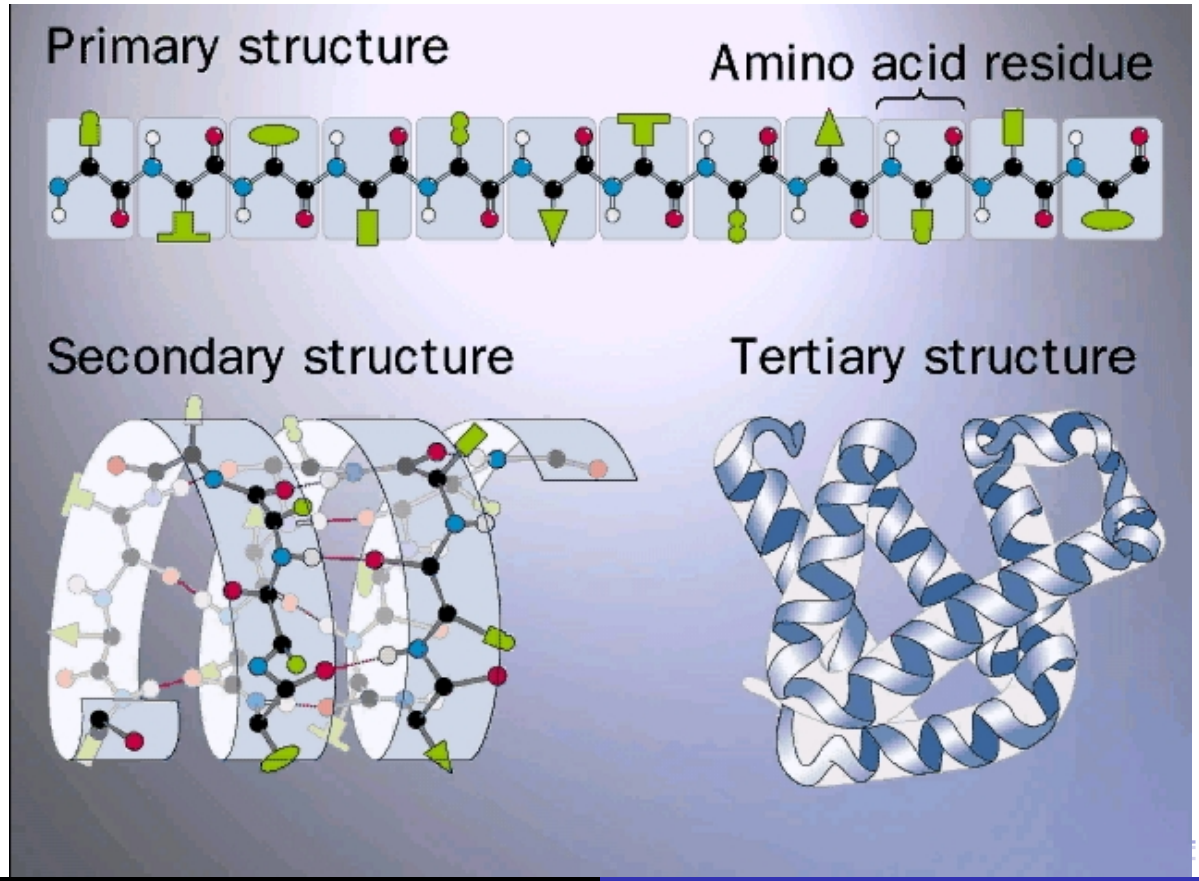
- draw a rectangular matrix whose rows are the possible hidden states s_i, and whose columns are the successive time steps t.
- For one time step t, draw the possible transitions for the forward calculation for $\theta_t = F$ as arrows on the matrix. Do the same for the backward calculation for $\theta_t = F$.
- Write the forward and backward probability sums for this state.

Draw the Forward-Backward Calculation Answer



Example: Secondary Structure Prediction

- Proteins are chains (strings) of a 20 letter alphabet (amino acids).
- The chain is flexible and automatically "folds" into three-dimensional structures based on its specific amino acid sequence.
- The simplest structural elements are α helix and β sheet.



Secondary Structure Prediction HMM

- When someone finds a new protein sequence, they often want to predict its secondary structure.
- Different secondary structures have different characteristic amino acid compositions, and length distributions.

What model structure would you propose? Give both an Information graph and state graph. Briefly define your variables.

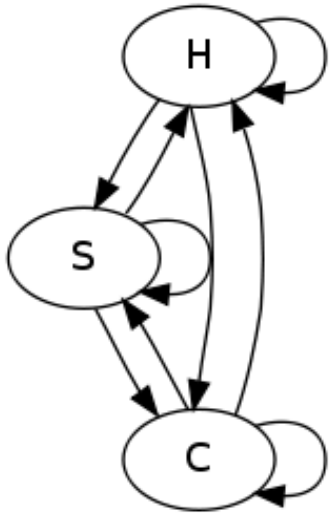
Secondary Structure Prediction HMM Answer

Given a protein sequence $\vec{X}^n$, we want to predict $\theta_t = H$ (helix) vs. S (sheet) vs. C ("random coil", which just means not helix or sheet) for each letter in the protein. Our info graph is just:

$$\theta_1 \rightarrow \theta_2 \rightarrow \ldots \rightarrow \theta_n$$

$$\theta_t \rightarrow X_t$$

And our state graph:



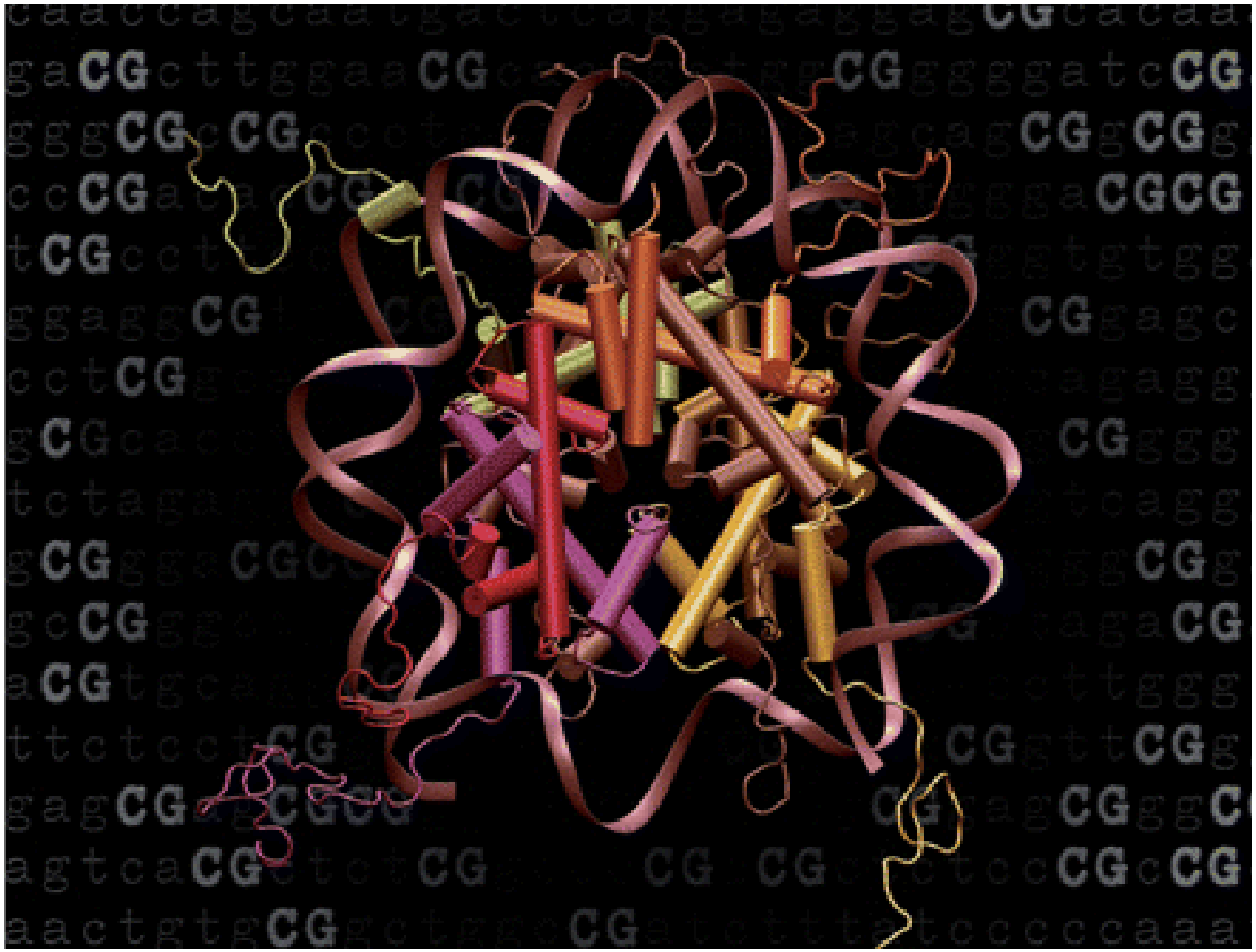
Secondary Structure Prediction Calculation

Say you are given all the necessary parameters for the HMM, and an amino acid sequence whose secondary structure we want to predict. To make predictions and measure confidence, what do we need to compute?

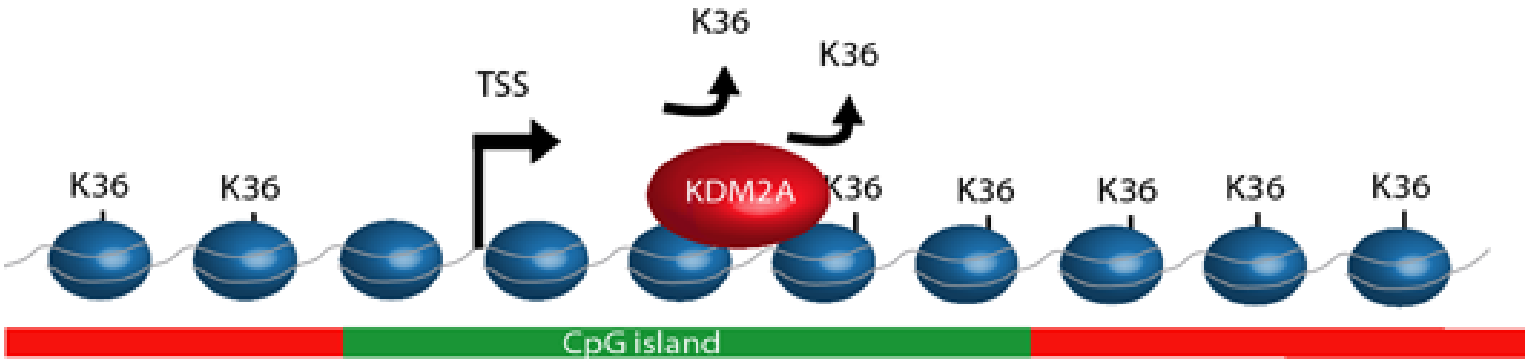
Secondary Structure Prediction Calculation Answer

To predict the most likely secondary structure for the amino acid sequence, we could just use the Viterbi algorithm to find the maximum probability path. However, since the question explicitly asked how we would calculate confidence on our prediction, we would be better off just computing the forward-backward posteriors for each position $p(\theta_t|\vec{X})$. This gives us both a prediction (i.e. the state with maximum posterior probability at each position), and a measure of confidence.

Nucleosome Binding To CpG Islands



Example: CpG Islands



- CpG islands are segments of DNA with unusually high frequencies of CG dinucleotide (i.e. a C followed by a G; the "p" refers to the phosphate linker on the DNA backbone).
- CpG islands are often binding sites for proteins that regulate the expression of a nearby gene, and *methylation* (a chemical modification of DNA that occurs in cells) of CpG islands helps control these regulatory interactions.
- CpG islands play an important role in tumors, which use them to turn off genes that block tumor growth.

Modeling CpG Islands as a Markov Chain

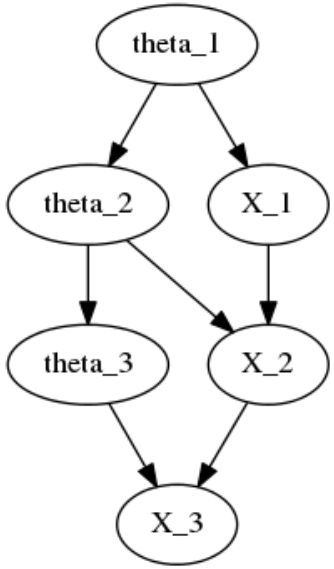
CpG islands show very different conditional probabilities $p(X_{t+1}|X_t)$ than non-CpG island sequence. This suggests we can use a Markov chain model to detect them in any sequence.

- Say we assume that any position in the genome can be in two possible states, *CpG* vs. *NOT-CpG*.
- We are given $p(\Theta_{t+1} = \text{CpG}|\Theta_t = \text{NOT-CpG})$ and $p(\Theta_{t+1} = \text{NOT-CpG}|\Theta_t = \text{CpG})$
- Say we are given two tables of conditional probabilities $p(X_{t+1}|X_t, \Theta_{t+1} = \text{CpG})$ and $p(X_{t+1}|X_t, \Theta_{t+1} = \text{NOT-CpG})$.

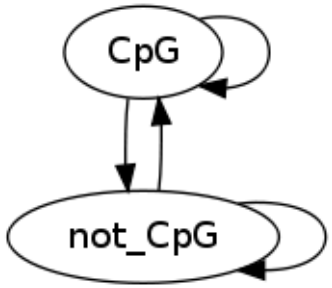
Draw the Info graph and state graph for modeling this problem (draw at least 2 observations, to make the sequence structure of your graph clear).

Modeling CpG Islands as a Markov Chain Answer

Information graph:



State Graph:



Announcements

- HMM Quiz Monday
- no further projects or quizzes
- grad term project feedback on CCLE, please take a look

Your Mission: Make Your First Solo Flight

- We have now trained on a variety of real-world HMM problems, including taking your test all the way to a real computational implementation (Casino Royale, DP alignment).
- Now it's time for you to apply your skills to solving new problems by formulating your own HMM models, on your own -- just like what you'll have to do in the real world.
- Friday: complete one more round of **mission training**, constructing an HMM for modeling a binding site "motif" or CpG islands, and getting immediate feedback.
- Monday: **quiz** where you'll be asked to formulate and apply an HMM for solving a problem (using the same principles) and scored on your ability to use the concepts soundly.

Monday's Quiz

- designed to be 30 minutes (but we'll give you 45 minutes).
- Your first solo flight solving problems with HMMs "for real money". If you can do it here, you'll be able to do it in the real world too.
- Bio glossary and equation sheet provided in the quiz -- memorization is not our focus, but rather whether you can use the ideas to figure out problems.

Maximum Probability Path

- Define a path variable $\vec{\theta} = (\theta_1, \theta_2, \ldots \theta_n)$ representing a path of n steps in a Markov chain.
- Find $\vec{\theta}^*$ such that $p(\vec{\theta}^*) \geq p(\vec{\theta})$ for all $\vec{\theta}$.
- Computational complexity if $p(\vec{\theta})$ followed the general chain rule?
- For a Markov chain?

Viterbi Algorithm

Define a Viterbi matrix **V** whose elements V_{ti} are determined as follows (we emphasize that i represents a state by writing s_i):

$$V_{ti} = \text{Max } p(\theta_t = s_i | \theta_{t-1} = s_j) V_{t-1,j}$$

where the maximimization is performed over all possible values of θ_{t-1} (indexed here by j). We also record the reverse-mapping $i \rightarrow j^*$, where j^* is the value of j that maximized V_{ti}:

$$R_t(i) = j^*$$

Then the *Viterbi path* $\vec{\theta}^*$ to $\theta_t = i$ is given by backtracking the reverse-mapping, i.e. for all $u < t$

$$\theta_u = R_{u+1}(\theta_{u+1})$$

HMM Viterbi

- Finds an optimal path (a sequence of hidden states) that maximizes the joint probability of the observable and hidden variables $p(\vec{X}, \vec{\theta})$.

$$V_{ti} = \text{Max}\left[p(X_t|\theta_t = s_i)p(\theta_t = s_i|\theta_{t-1} = s_j)V_{t-1,j}\right]$$

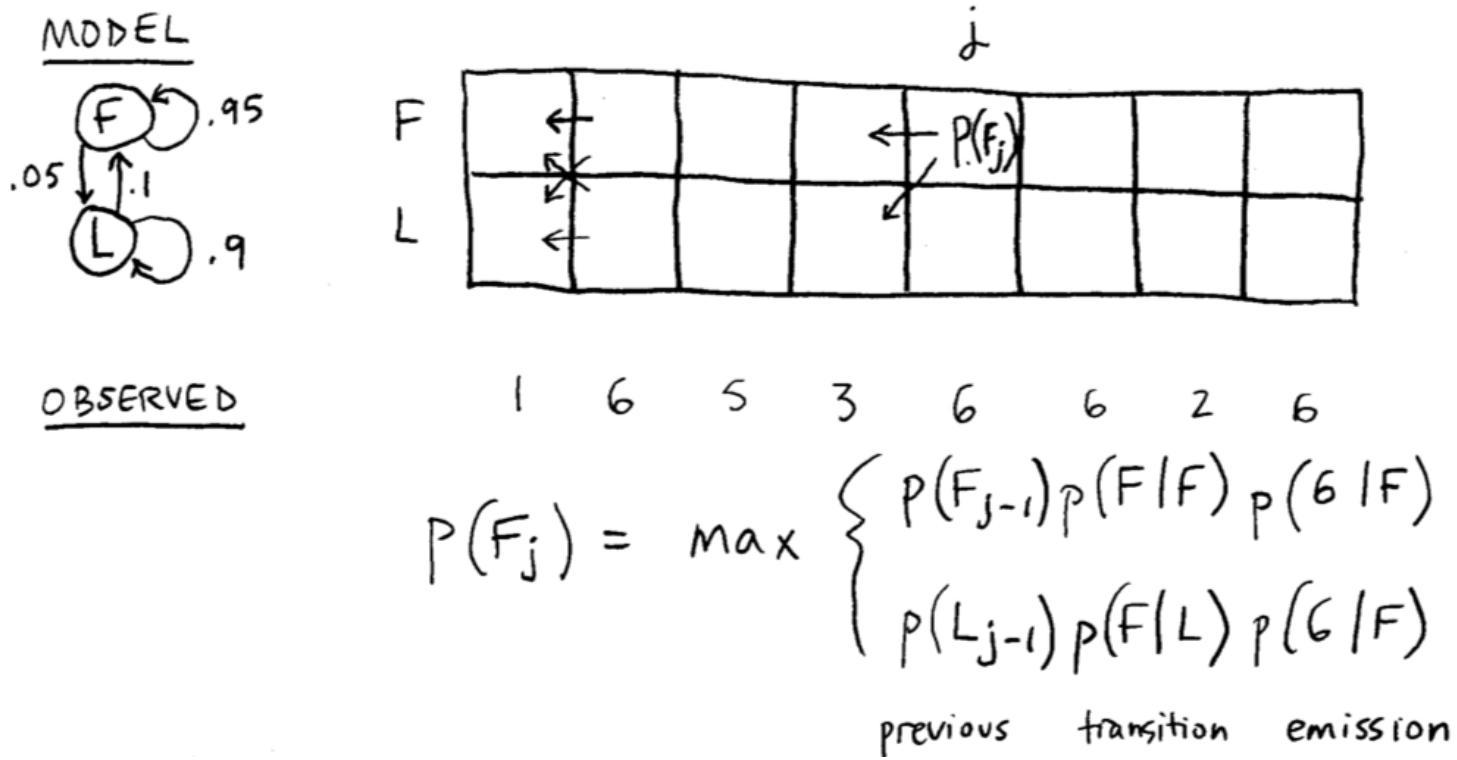
- Same as our first definition; we simply declare our Markov chain to be hidden variables.
- Can be implemented as a recursive algorithm, sometimes called "dynamic programming"

Occasionally Dishonest Casino Viterbi

To make the Viterbi algorithm concrete, let's draw the calculation:

- draw a rectangular matrix whose rows are the possible hidden states s_i, and whose columns are the successive time steps t.
- For one time step t, draw the possible transitions at that time step as arrows on the matrix.
- Write the Viterbi condition for choosing the optimal path probabiity to state $\theta_t = F$

Occasionally Dishonest Casino Viterbi Answer



- Shorthand notation: Interpret j as our time index.
- Interpret $p(F_j) \equiv p(\theta_1^*, \theta_2^*, ..., \theta_j^* = F, \vec{O}^j)$

Occasionally Dishonest Casino Viterbi Errors

- Some people left out the emission probability.
- Some people are thinking about Viterbi backwards, i.e. maximizing over multiple destination states θ_{t+1} from a specific *origin* θ_t. No; Viterbi maximizes over multiple origin states θ_{t-1} that transition to a specific *destination* θ_t.
- Some people included both a $p(\theta_{t-1})$ factor and a V_{t-1} factor. No; V_{t-1} already includes $p(\theta_{t-1})$.
- Some people are mixing up observable vs. hidden variables X_t, θ_t. In an HMM, the transitions are between *hidden states*.
- Some people seemed to think that the basic Viterbi step decides which state (F vs. L) is best at time *t*. No; for *each* state at time *t* (e.g. F), it decides which state at time *t-1* (F vs. L) is the best path to this state. Concretely, after we calculate $V_{t,F}$, $V_{t,L}$, we still don't know F or L will be on the Viterbi path at this time point. And we won't know until we find the maximimum probability endpoint at the end of the matrix, and traceback from it to time *t*.
- Some people left out the maximization step altogether. Don't

Proof By Induction

Assume the Viterbi path $\vec{X}^{t-1,j*}$ ending at $X_{t-1} = s_j$ is globally optimal, i.e. $p(\vec{X}^{t-1,j*}) = V_{t-1,j} \geq p(\vec{X}^{t-1,j})$ for all paths $\vec{X}^{t-1,j}$ ending at the specified $X_{t-1} = s_j$.
Then for *any* specific path $\vec{x}^{t,i}$ passing through $x_{t-1} = s_j$ and ending at $x_t = s_i$:

$$p(\vec{x}^{t,i}) = p(x_1, ..., x_{t-1} = s_j)p(x_t = s_i | x_{t-1} = s_j)$$

$$\leq V_{t-1,j}p(x_t = s_i | x_{t-1} = s_j)$$

$$\leq V_{t,i} = p(\vec{X}^{t,i*})$$

for the Viterbi path ending at $X_t = s_i$. I.e. it too is globally optimal. And since we know the optimal path to $X_1 = s_i$ (trivially), the induction holds for all t.

Computational Complexity?

What is the computational complexity of the Viterbi algorithm e.g. for the Occasionally Dishonest Casino optimal path? Express your answer in big-O notation, assuming that each hidden variable has m possible states, and the total length of the observation sequence is n.

Computational Complexity? Answer

The computational complexity for computing the optimal path is $O(nm^2)$ both in time and memory.

Transition Matrices

- For a homogeneous Markov chain, it's convenient to express it as its *transition matrix* T whose elements $\tau_{ij} = p(X_{t+1} = s_j | X_t = s_i)$ represent the *transition probabilities* for going from state $s_i \rightarrow s_j$.

$$T = \begin{pmatrix} p(s_1|s_1) & p(s_2|s_1) & \dots \\ p(s_1|s_2) & p(s_2|s_2) & \dots \\ \dots & & \\ p(s_1|s_m) & \dots & p(s_m|s_m) \end{pmatrix}$$

Note that *rows* sum to 1, not columns!

Stationary Distribution Definition

A distribution $p(s_i) = \pi_i$ is called *stationary* if for all s_i

$$\pi_i = \sum_j \pi_j \tau_{ji}; \text{ i.e. } \vec{\pi} = \vec{\pi}\mathbf{T}$$

which we can rearrange conveniently as follows:

$$\pi_i \sum_j \tau_{ij} = \sum_j \pi_j \tau_{ji}$$

$$\pi_i \sum_{j \neq i} \tau_{ij} = \sum_{j \neq i} \pi_j \tau_{ji}$$

This is called the *general balance equation*, because it represents a perfect balance of probability *exiting* state i vs. *entering* state i.

Uniform Probabilities?

Say we wish our model of the Occasionally Dishonest Casino to have the same hidden state probability $p(\Theta_t = F)$ at all times t. Assuming the transition probabilities are $\tau_{FL} = p(L|F) = 0.05$ and $\tau_{LF} = p(F|L) = 0.1$, how can we achieve this goal? (If this is not possible with the given data, just say "Insufficient data").

Uniform Probabilities? Answer

- If we set $p(\Theta_1 = s_i) = \pi_i$ to the *stationary distribution*, then the distribution at all times will just be $\vec{\pi}$.
- For this two-state case it's easy to get $\vec{\pi}$ straight from the transition probabilities:

$$\pi_F = \frac{\tau_{LF}}{\tau_{LF} + \tau_{FL}} = \frac{2}{3}$$

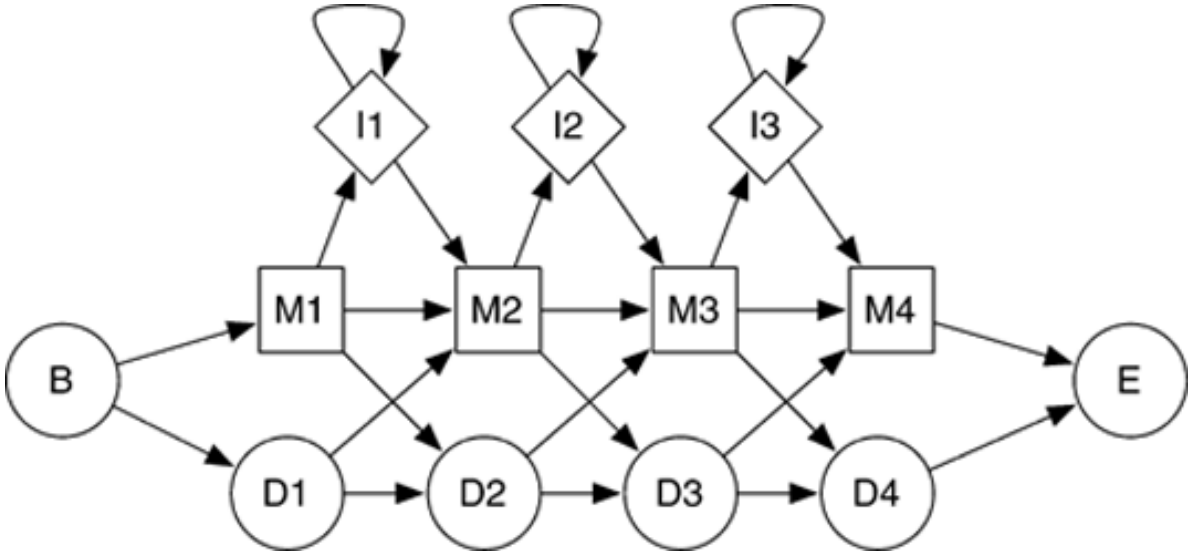
Uniform Probabilities? Errors

- Some people thought there was no way to maintain constant state probabilities over the Markov chain. But that is exactly what a stationary distribution is: a vector of state probabilities that are completely unchanged by our transition matrix.
- Some people correctly proposed the idea of the stationary distribution, but didn't derive the actual equation for the stationary distribution. That is a minor error.
- Some people proposed solving this by setting funny values for p(L|L) and p(F|F). No; those are already fixed by the data given in the problem.
- Some people proposed not permitting transitions to the L state. No; the problem explicitly assigned non-zero probability to those transitions.

Example: Profile Alignment HMM

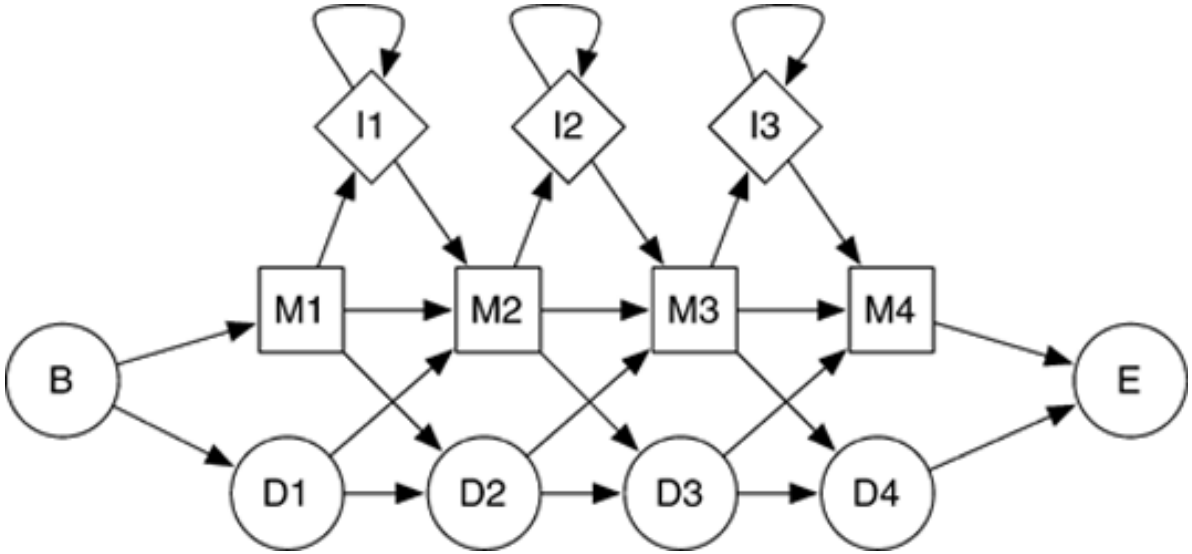
- Being able to assign a new sequence to a known protein family is crucial for infering its function.
- Being able to align it accurately to other members of the family indicates which parts of the new sequence carry out different aspects of the function of that family.
- Tools for aligning to individual sequences in the database (one at a time, e.g. BLAST) are often less sensitive than a "profile" that captures the detailed patterns of conservation of a protein family (in other words, the many sequences in that family).
- These patterns show you what's important vs. not, for recognizing a new member of the family.

Profile HMM



- Note this is a *heterogeneous* HMM: we use a different subset of states at each "step" of the HMM.
- **match** states (M) each store observed amino acid probabilities for a *specific* position in the protein. Think of these as the "consensus" positions in the family.
- **insert** states (I) allow for the possibility that a family member inserts extra amino acid(s) at a given position.
- **delete** states (D) are "silent", i.e. emit nothing.

State @ Time Notation

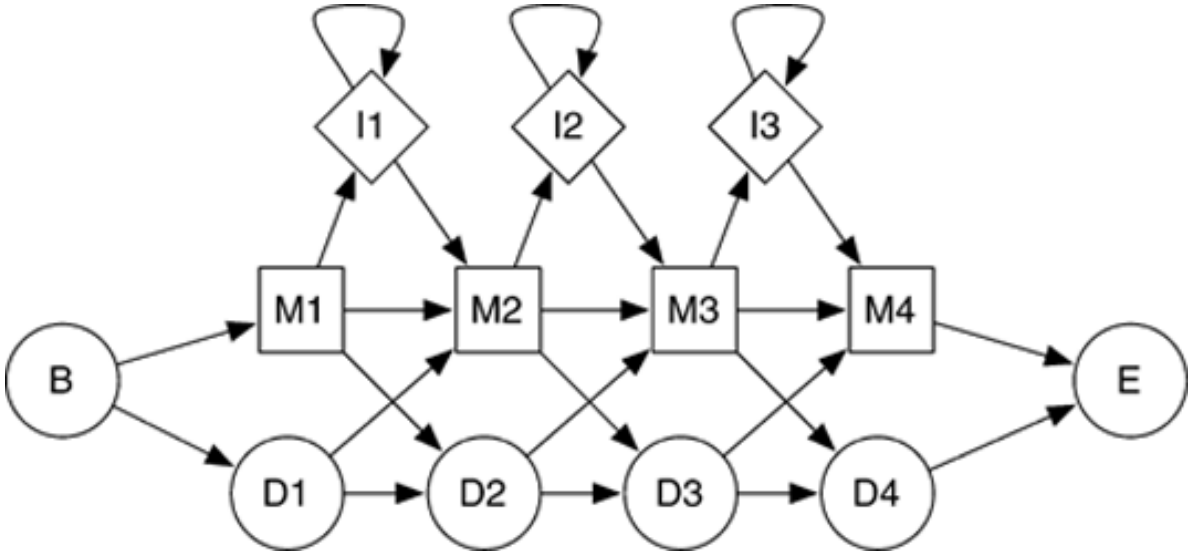


- Given the complexities of defining our hidden variables on variable length hidden sequences (due to insertion and deletion states, no longer guaranteed to match observation sequence length), it's a lot simpler to ensure we have a good notation for tracking *hidden states* vs. *observation time*.
- write this $s_i @ X_t$: "in state s_i at observation time t"
- So we can compute things like $p(s_i @ X_t | \vec{X}^n)$

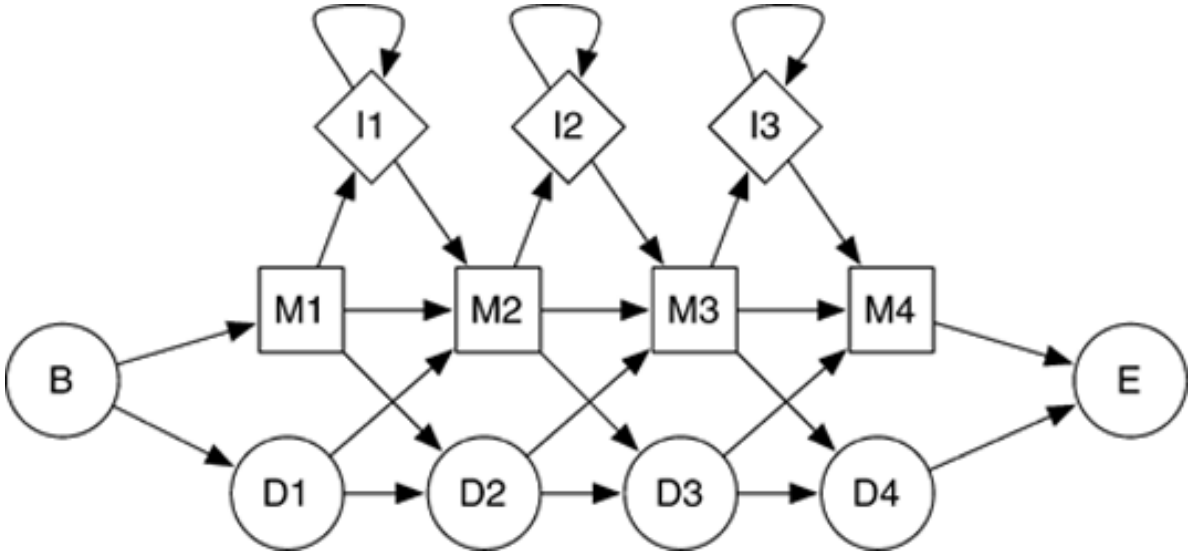
Emission of One Observation

Consider the following questions for this profile HMM:

- Give an example path that emits X_3 from state M3.
- Which of the following states could *emit* observation X_3?
- which of the following states could have non-zero $p(s_i @ X_3 | \vec{X}^n) > 0$



Emission of One Observation Answer



- $M1@X_1 \rightarrow M2@X_2 \rightarrow M3@X_3 \rightarrow \ldots.$
- States $I1, M2, I2, M3, I3, M4$ could emit X_3.
- States $I1, M2, I2, M3, D3, I3, D4, M4$ could have non-zero $p(s_i@X_3|\vec{X}^n)$. E.g. $M1@X_1 \rightarrow D2@X_2 \rightarrow M3@X_2 \rightarrow M4@X_3$.

Predecessor Notation

- It's also useful to be able to define a hidden variable $\pi_{s_i@X_t}$ that represents the set of all **predecessor** states @ time $s_j@X_u$ that can transition to a particular $s_i@X_t$, i.e.

$$\pi_{s_i@X_t} = \{\forall s_j@X_u | s_j@X_u \rightarrow s_i@X_t\}$$

- Note that when we use this in a Viterbi probability, to avoid piling up too many layers of subscripts, we can write the Viterbi probability for a particular predecessor $\pi_{s_i@X_t}$ as $V(\pi_{s_i@X_t})$.
- We also introduce the notation X_π to indicate "the observation (if any) emitted by $\pi_{s_i@X_t}$", and $\vec{X}_{[1,\pi]}$ to indicate "all the observations emitted up to (and including) $\pi_{s_i@X_t}$".

Writing Viterbi in State@Time notation?

Suggest a way of writing the Viterbi V and R matrix rules explicitly in terms of **state@time notation** and **predecessor notation**.

Writing Viterbi in State@Time notation? Answer

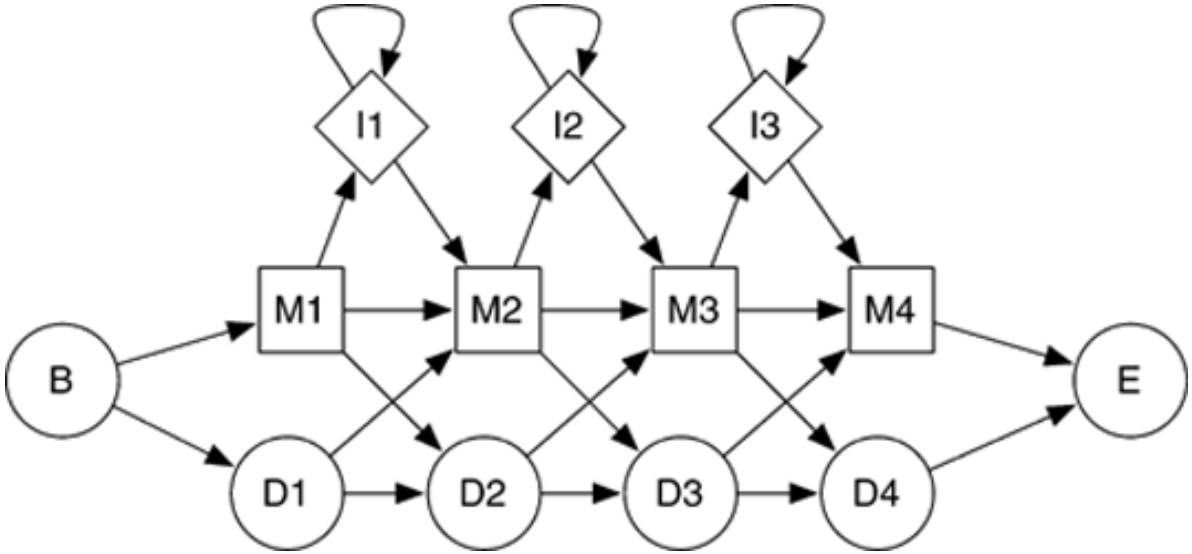
- First, let's recast our V matrix notation explicitly in terms of state@time $V(s_i@X_t)$.
  This is more explicit and thus less potentially confusing than using V_{ti} subscripts.
- We can now rewrite our Viterbi maximization rule in state@time notation:

$$V(s_i@X_t) = \max_{\pi}[V(\pi_{s_i@X_t})p(s_i@X_t|\pi_{s_i@X_t})p(X_t|s_i@X_t)]$$

- We write the R matrix rule similarly:

$$R(s_i@X_t) = \text{argmax}_{\pi}[V(\pi_{s_i@X_t})p(s_i@X_t|\pi_{s_i@X_t})p(X_t|s_i@X_t)]$$

Profile HMM Viterbi



Using the `state@time` notation $s_i@X_t$, indicate how you would apply the Viterbi algorithm to find the best possible (maximum likelihood) path up to $M4@X_4$ (i.e. that state M4 emits observation sequence letter X_4. Specifically, write the Viterbi maximization rule for arriving at $s_i@X_t$, showing specifically what "incoming" states @ time must be maximized over.

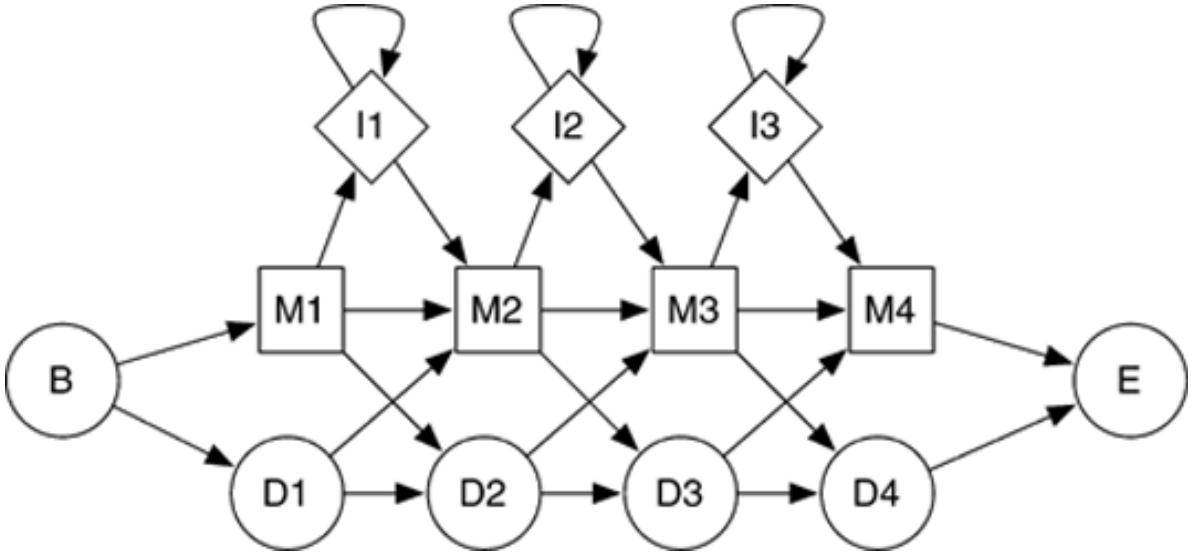
Profile HMM Viterbi Answer

We can arrive from $M3@X_3, I3@X_3, D3@X_4$ (note that D states emit no observation). So the Viterbi is just:

$$V_{4,M4} = p(X_4|M4)\max\{V_{3,M3}p(M4|M3), V_{3,I3}p(M4|I3), V_{4,D3}p(M4|D3)\}$$

Note that as usual the emission probability factor $p(X_4|M4)$ is the same for all three incoming edges.

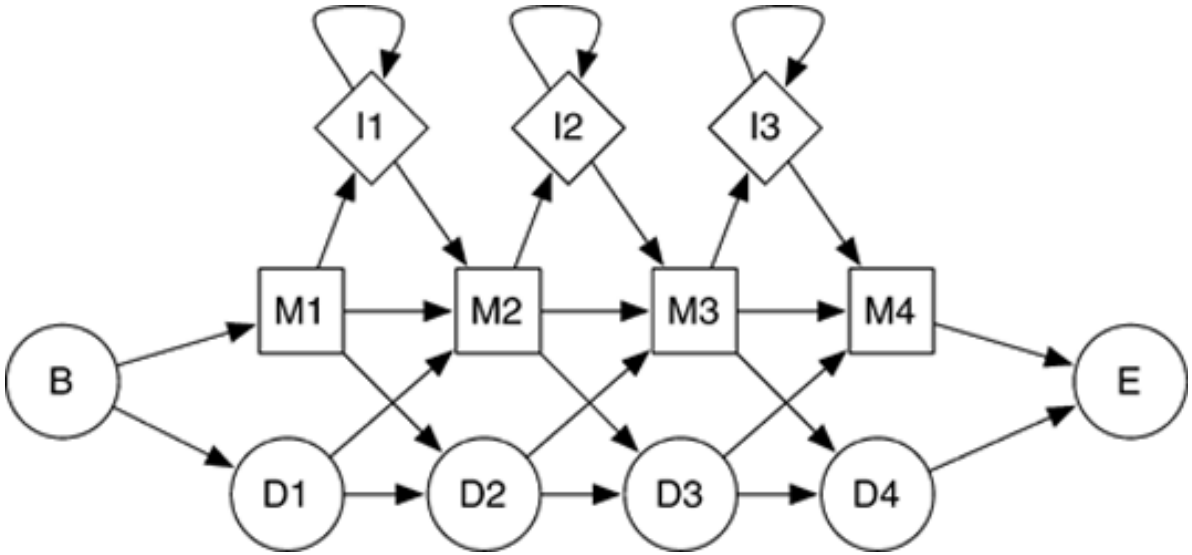
Storing the Profile HMM Viterbi matrix?



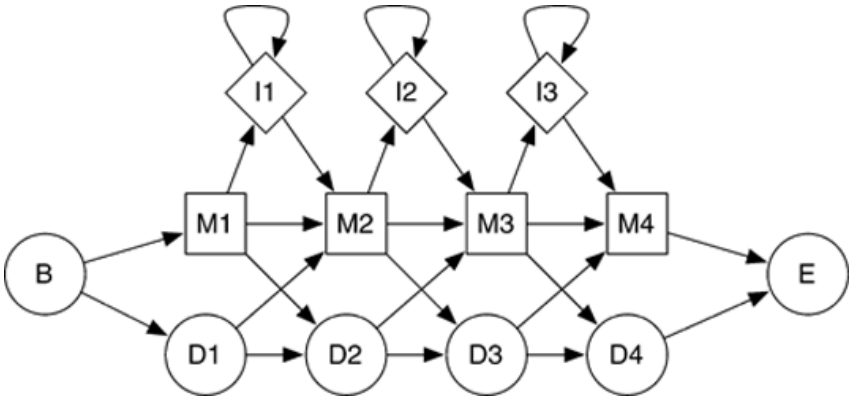
Suggest how you would actually store the Viterbi V matrix for the Profile HMM in a computer program, explicitly in terms of **state@time notation**.

Storing the Profile HMM Viterbi matrix? Answer

Concretely, we need to store $V(s_i@X_t)$, so in simple pseudocode terms we would store a matrix that is indexed by both state *i* and time *t*, i.e. `V[i][t]`, where *i* indexes every possible state `M1, I1, D1, ...` in the Profile HMM:



Possible Profile HMM Predecessor Times?



For a profile HMM (e.g. the above), given a predecessor variable

$$\pi_{s_i@X_t} = \{\forall s_j@X_u | s_j@X_u \rightarrow s_i@X_t\}$$

- What are the possible value(s) of the predecessor observation index u?
- If there's more than one possible value, what determines that value?

Possible Profile HMM Predecessor Times? Answer

For

$$\pi_{s_i@X_t} = \{\forall s_j@X_u | s_j@X_u \rightarrow s_i@X_t\}$$

- The possible values are $u = t - 1$ (for predecessor M or I states) or $u = t$ (for predecessor D states).
- It depends on whether the *predecessor* state emits anything or not. Note well: it does *not* depend on the destination state s_i.

Successor Notation

In the same way, we define a similar notation for successors:

- $\sigma_{s_i@X_t}$: a hidden variable that represents the set of all **successor** states @ time $s_j@X_u$ that $s_i@X_t$ can transition to, i.e.

$$\sigma_{s_i@X_t} = \{\forall s_j@X_u | s_i@X_t \rightarrow s_j@X_u\}$$

- X_σ: the observation (if any) emitted by $\sigma_{s_i@X_t}$.
- $\vec{X}_{[\sigma,n]}$: all the observations emitted from $\sigma_{s_i@X_t}$ and beyond.
- $\vec{X}_{(\sigma,n]}$: all the observations emitted *after* (and not including) $\sigma_{s_i@X_t}$. Note the use of standard open/closed interval notation, e.g. $(\sigma, n]$ means "the interval from (but not including) σ to (and including) n".

Profile HMM Posterior

Say we have a large profile HMM and apply it to an observation sequence $\vec{X}$ that we know belongs to this protein family. We want to know the posterior probability that a particular letter X_t "aligns to" (is emitted by) a particular state s_i.

- show how you would define that (in our usual Bayesian way).
- indicate how you would break that into "forward" and "backward" parts that you could calculate recursively.
- using our successor notation $\sigma_{s_i@X_t}$ write the recursion equation for computing the backward probability for computing this posterior.

Profile HMM Posterior Answer

- We just write Bayes' Law for our desired posterior in the usual way:

$$p(s_i@X_t|\vec{X}_{[1,n]}) = \frac{p(s_i@X_t, \vec{X}_{[1,n]})}{p(\vec{X}_{[1,n]})}$$

- We cut the numerator at $s_i@X_t$ into "forward" and "backward" components:

$$p(s_i@X_t, \vec{X}_{[1,n]}) = p(s_i@X_t, \vec{X}_{[1,s_i@X_t]})p(\vec{X}_{(s_i@X_t,n]}|s_i@X_t)$$

- We write the backward probability in terms of itself, shifted one step to the right, using successor notation:

$$p(\vec{X}_{(s_i@X_t,n]}|s_i@X_t) = \sum_{\sigma_{s_i@X_t}} p(\sigma_{s_i@X_t}|s_i@X_t)p(X_\sigma|\sigma_{s_i@X_t})p(\vec{X}_{(\sigma,n]}|\sigma_{s_i@X_t})$$

Profile HMM "non-emission" probability?

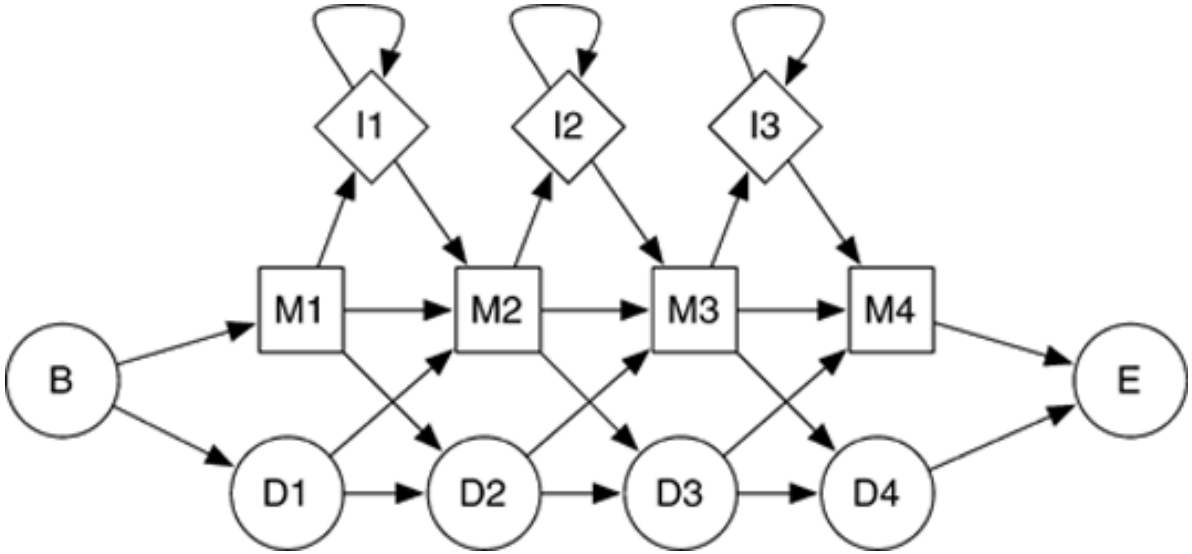
If the successor state $\sigma_{s_i@X_t}$ emits **nothing** (e.g. D state), what "effective value of $p(X_\sigma|\sigma_{s_i@X_t})$" would you use, e.g. in our standard equation for the backward probability?

Profile HMM "non-emission" probability? Answer

We need to use an effective value that gives the exact same result as if we hadn't included any emission term for this state, i.e.

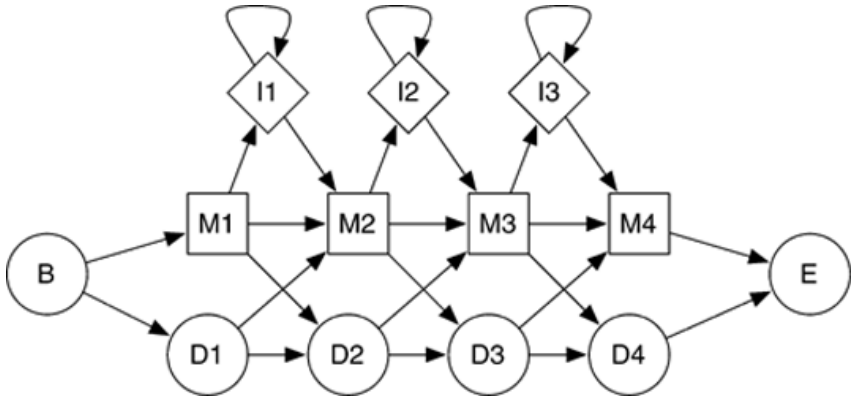
$$p(X_\sigma | \sigma_{s_i @ X_t} = D) = 1$$

Profile HMM probability of all observations?



How would you calculate the denominator $p(\vec{X}^n)$ on this profile HMM, e.g. using the forward probability?

Profile HMM probability of all observations? Answer



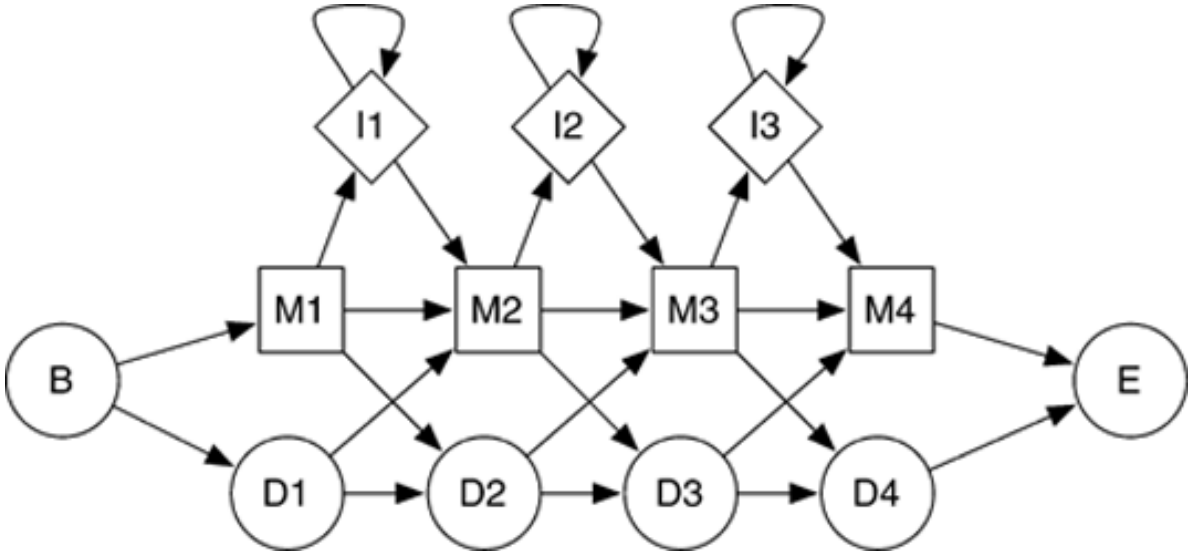
This is easiest to do at a point where all paths converge, e.g. calculate the forward probability at the END state **E**:

$$p(\vec{X}^n) = p(E@X_{n+1}, \vec{X}^n)$$

Note that the state @ time constraint *t=n+1* ensures we take into account *all* observations (up to and including X_n).
Note also this isn't the only way to do it, e.g. the probability flowing into **E** is equal to summing

$$p(\vec{X}^n) = p(M4@X_n, \vec{X}^n) + p(D4@X_{n+1}, \vec{X}^n)$$

Theta_t Hidden Variables?

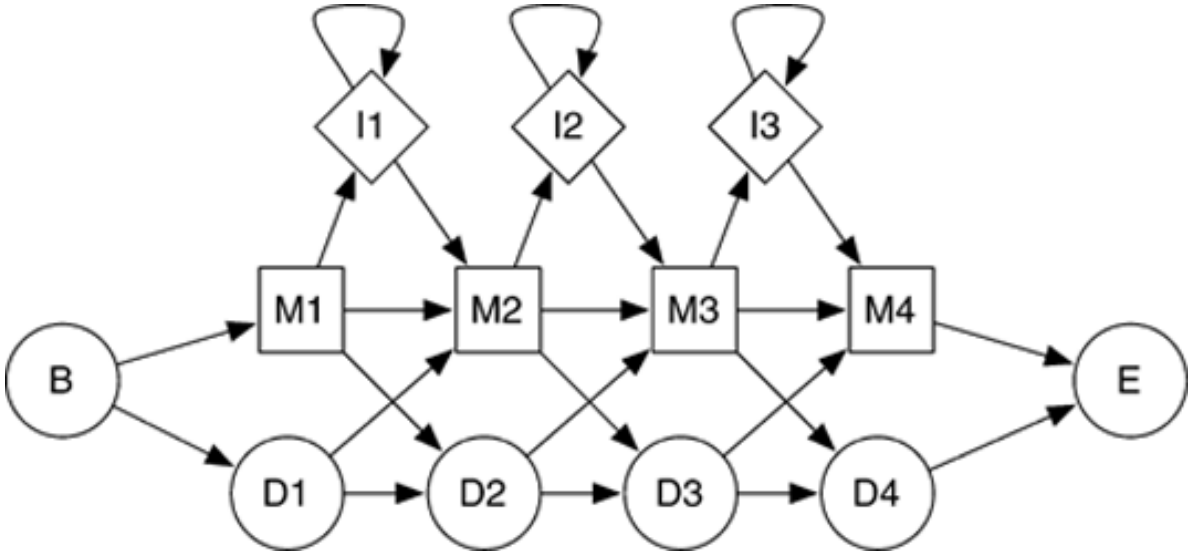


Say you wanted to describe our profile HMM in terms of the hidden variable sequence $\vec{\theta} = \theta_1 \rightarrow \theta_2 \rightarrow ....$ In general, would defining the hidden states for a given θ_t as

$$\theta_t = \{I_t, M_t, D_t\}$$

be a valid definition of its states?

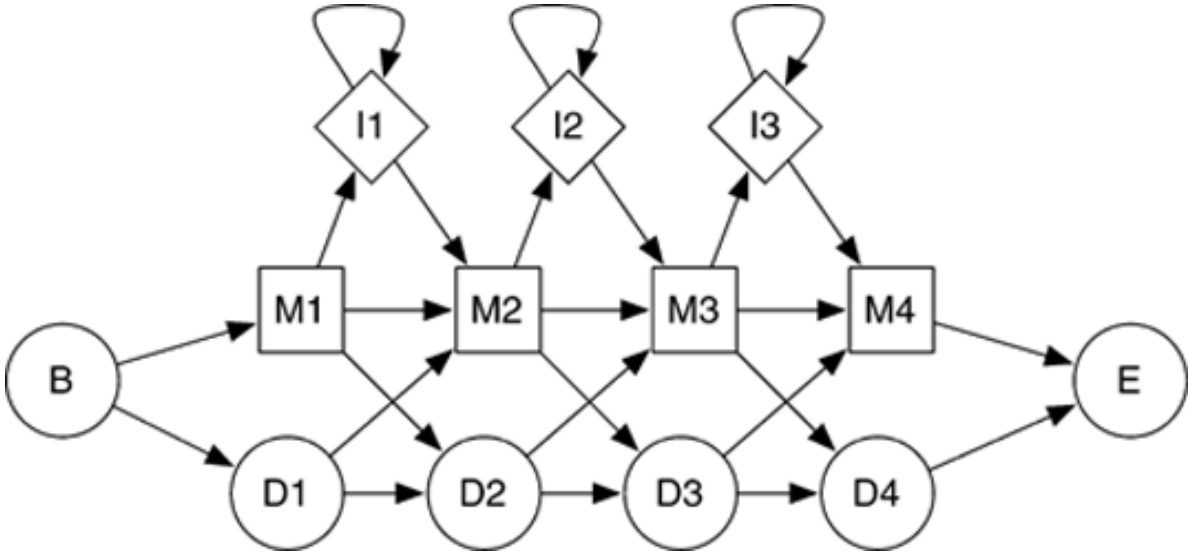
Theta_t Hidden Variables? Answer



No, the "numbering" of the HMM states has (almost) nothing to do with the θ_t numbering:

- Note that while M_t, D_t states are disjoint in any hidden state path (only one of them can be in a given path), I_t is *not* disjoint with M_t!
- The existence of self-edges for I states means that the hidden state sequence variable θ_t can get "out of sync" with the "state indices" of the profile HMM, e.g. we could have $\theta_4 = I1$. Strictly speaking, θ_t could be in just about *any* of the states in the HMM (just like for a homogeneous HMM!).

Profile HMM Information Graph?



Draw the information graph for this profile HMM (you only need to show enough of the information graph to make its structure clear). Use our standard `A -> B;` notation (the `graphviz dot` format).

Profile HMM Information Graph? Answer

- We have a hidden variable θ_u for each state the HMM passes through:

$$\theta_1 \rightarrow \theta_2 \rightarrow \ldots \rightarrow \theta_u \rightarrow \ldots$$

- The observable part of our information graph requires a bit more care, because we no longer have a one-to-one correspondence for each hidden state to emit one observation. We **cannot** just write $\theta_u \rightarrow X_u$; we actually have to track the "observation index" t separate from θ_u. We can write this as

$$\theta_u \rightarrow X_t \text{ iff } \theta_u = s_i @ X_t \text{ and } s_i \rightarrow X$$

- You could consider u to be "hidden time" and t to be "observable time" (which can get out of sync due to D states).

What Matters: Hidden State, Observable Time

- Generally what we *want to know* is **hidden state** as a function of **observable time**.
- Note: that is exactly what the **state @ time** $s_i@X_t$ notation captures.
- So every calculation you do (Viterbi, Forward-Backward etc.) will be in terms of $s_i@X_t$, its **predecessors** $\pi_{s_i@X_t}$ and **successors** $\sigma_{s_i@X_t}$.
- We generally do not even bother keeping track of "hidden time" θ_u.

Homogeneous HMM Parameter Matrices

Once the structure of hidden states s_i and their allowed transitions $s_i \rightarrow s_j$ is defined, the HMM is characterized by three matrices of parameters representing the **transition**, **emission**, and **prior** probabilities:

$$T : \tau_{ij} = p(s_j|s_i)$$

$$E : \varepsilon_{ix} = p(x|s_i)$$

$$P : p(\theta_1 = s_i)$$

Where do we get these parameters?

Homogeneous HMM Parameter Estimation

For a Markov chain $\theta_1 \rightarrow \theta_2 \rightarrow \ldots \theta_n$, by the Law of Large Numbers

$$\frac{1}{n}\sum_{t=1}^{n} Y_t \rightarrow E(Y) = \sum_{\theta} \pi(\theta) Y(\theta) \text{ as } n \rightarrow \infty$$

where $\pi(\theta)$ is the stationary distribution of θ. Defining an indicator function $I_x(X) = 1$ iff $X = x$, 0 otherwise

$$\frac{1}{n}\sum_{t=1}^{n} I_x(X_t) I_{s_i}(\theta_t) \rightarrow \sum_{X,\theta} \pi(X,\theta) I_x(X) I_{s_i}(\theta) = \pi(x, s_i) = \pi(s_i) p(x|s_i)$$

Finally, we estimate $I_{s_i}(\theta_t)$ by $p(\theta_t = s_i|\vec{X})$ again via a Law of Large Numbers convergence

$$\frac{1}{n}\sum_{t=1}^{n} I_x(X_t) p(\theta_t = s_i|\vec{X}) \rightarrow \sum_{X,\theta} \pi(X,\theta) I_x(X) I_{s_i}(\theta) = \pi(x, s_i) = \pi(s_i) p(x|s_i)$$

Emission Probability Estimation?

For a homogeneous Markov chain $\theta_1 \rightarrow \theta_2 \rightarrow \ldots \theta_n$, how could you obtain a Law of Large Numbers estimator of the emission probability $p(X|s_i)$?

Emission Probability Estimation? Answer

$$p(x|s_i) = \frac{\pi(s_i)p(x|s_i)}{\pi(s_i)} = \frac{\frac{1}{n}\sum_{t=1}^{n} I_x(X_t)p(\theta_t = s_i|\vec{X})}{\frac{1}{n}\sum_{t=1}^{n} p(\theta_t = s_i|\vec{X})}$$

Transition Probability Estimation?

For a homogeneous Markov chain $\theta_1 \rightarrow \theta_2 \rightarrow \ldots \theta_n$, how could you obtain a Law of Large Numbers estimator of the transition probability $p(s_j|s_i)$?

Transition Probability Estimation? Answer

We take our expectation value over all possible values of an edge in the Markov chain:

$$\frac{1}{n-1}\sum_{t=1}^{n-1} p(\theta_t = s_i, \theta_{t+1} = s_j|\vec{X}) \rightarrow \sum_{X,\theta} \pi(\theta \rightarrow \theta') I_{s_i,s_j}(\theta \rightarrow \theta')$$

$$= \pi(s_i \rightarrow s_j) = \pi(s_i)p(s_j|s_i)$$

Therefore

$$p(s_j|s_i) = \frac{\pi(s_i)p(s_j|s_i)}{\pi(s_i)} = \frac{\frac{1}{n-1}\sum_{t=1}^{n-1} p(\theta_t = s_i, \theta_{t+1} = s_j|\vec{X})}{\frac{1}{n}\sum_{t=1}^{n} p(\theta_t = s_i|\vec{X})}$$

Estimating Priors

$$\pi_i = p(\Theta_1 = s_i)?$$

Estimating Priors

Given N observation sequences $\vec{X}_1, \vec{X}_2, ...\vec{X}_N$

$$p(\Theta_1 = s_i) \approx \frac{1}{N} p(\Theta_1 = s_i | \vec{X}_j)$$

A Circular Problem

Given T, E, P we can infer posteriors for $\Theta_1, \Theta_2, \ldots \Theta_n$ from an observation sequence $\vec{X}$

$$T, E, P \xrightarrow{\vec{X}} p(\Theta_t | \vec{X})$$

Given posteriors for $\Theta_1, \Theta_2, \ldots \Theta_n$, we can estimate T, E, P

$$p(\Theta_t | \vec{X}) \rightarrow T, E, P$$

But in real life we are trying to infer *both*.

Self-consistency Condition

Given the correct T^*, E^*, P^* and a large enough set of observations $\vec{X}$, our estimation procedure should yield the same T^*, E^*, P^*:

$$T^*, E^*, P^* \xrightarrow{\vec{X}} p(\Theta_t|\vec{X}) \rightarrow T^*, E^*, P^*$$

Baum-Welch Training Algorithm

Given an initial estimate T_1, E_1, P_1 and a large enough set of observations $\vec{X}$, our estimation procedure should yield an improved estimate T_2, E_2, P_2:

$$T_1, E_1, P_1 \overset{\vec{X}}{\rightarrow} p(\Theta_t|\vec{X}) \rightarrow T_2, E_2, P_2 \rightarrow \ldots \rightarrow T^*, E^*, P^*$$

Global convergence is not guaranteed, but for well-behaved likelihood functions, it generally converges.

Baum-Welch Training Algorithm

0. choose initial parameters T_1, E_1, P_1

1. calculate posteriors via forward-backward

2. calculate T_2, E_2, P_2

3. repeat until $\Delta T, \Delta E, \Delta P$ fall below some threshold

Occasionally Dishonest Casino Baum-Welch?

Say you know that the casino switches surreptitiously between fair and loaded dice, but do not know either the emission probabilities of the loaded dice, or the transition probabilities for the switch. You observe a long history of rolls $\vec{X}$ from the casino, and notice there appear to be far more sixes overall than expected by random chance. How could you train an HMM on this observed history?

Occasionally Dishonest Casino Baum-Welch? Answer

We wish to estimate the emission probabilities $p(X|L)$ and transition probabilities $p(s_j|s_i)$, using Baum-Welch:

- we constrain $p(X|F) = 1/6$.
- we choose some initial guess $p(X|L)$, e.g. if we assume half the rolls are fair vs. loaded, we can estimate $\lambda = p(6|L)$ as

$$(\lambda + 1/6)/2 = n_6/n$$

- we choose some initial guess e.g. $p(F|L) = p(L|F) = 0.1$
- we compute posteriors $p(\theta_t|\vec{X})$.
- we compute new **T**, **E** from our posteriors.
- repeat Baum-Welch to convergence.

What If We Score Every Position the Same?

- If we are just aligning two sequences, we may not have "profile" information for either.
- We could still use our profile HMM, but it seems unnecessarily complicated.
- Instead, use same scoring function at every position: a *substitution matrix* (for *match* moves) and a "gap penalty" for insertions and deletions.
- Note profile-HMM alignment was "asymmetric": one seq treated as "observations", the other as a (heterogeneous) HMM sequence of hidden states. By contrast, we will now treat sequences $\vec{X}, \vec{Y}$ symetrically. E.g. "insertion" in one sequence equivalent to deletion in the other sequence, so just call it "indel".

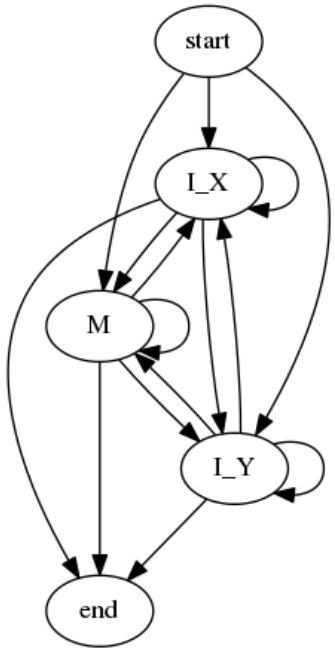
Pairwise Alignment Model

- Pairwise alignment of two sequences $\vec{X}^m, \vec{Y}^n$
- Index letter positions in the two sequences with t, u, e.g. X_t, Y_u.
- Hypothesis: these two sequences are related to each other by some evolutionary process, e.g. mutation from a common ancestor: $\vec{A} \rightarrow \vec{X}$ and $\vec{A} \rightarrow \vec{Y}$.
- Want to be able to assess this hypothesis and gain insights from it, e.g.
  - calculate posterior odds vs. null hypothesis that they are unrelated;
  - predict which parts of the two sequences are "the same", i.e. aligned, and likely to perform same function. Easy when the two sequences are very similar, much harder when similarity weak.

Affine Gap Penalties

- "simple" gap penalty charges same penalty for any indel of one letter, regardless of what came before it.
- By contrast, "affine" gap penalty charges different penalty if the previous move was a "match move" (this is referred to as the "gap opening" penalty), versus if the previous move was also an indel ("gap extension" penalty).
- Typically there is a bigger penalty for gap opening than gap extension.
- Consequence: a single matrix is no longer sufficient for performing Viterbi alignment. Instead, we must keep separate matrices for the **m**, **x** and **y** "states".
- Concretely, "gap opening" is a transition from the **m** matrix to either the **x** or **y** matrices; "gap extension" is a move within the **x** matrix or within the **y** matrix.

Homogeneous Alignment HMM



- We only need three states to represent **MATCH**, **INSERT-X** and **INSERT-Y** as a homogeneous HMM!
- But since we're aligning *two* observation sequences $\vec{X}, \vec{Y}$, now we need **two observation time indices**, i.e. our state @ time notation becomes $s_i @ X_t, Y_u$.

Homogeneous Alignment HMM Viterbi

Say we want to find the best possible alignment of sequences $\vec{X}, \vec{Y}$. Propose the basic Viterbi maximization rule for finding this best alignment using state @ time notation, to obtain both the maximum likelihood path $\vec{\theta}^*$ and the joint probability $p(\vec{X}, \vec{Y}, \vec{\theta}^*)$. (You only need to give the basic rule at one Viterbi path endpoint; you do not need to show how to get the complete alignment of $\vec{X}, \vec{Y}$).

Homogeneous Alignment HMM Viterbi Answer

First we write down the basic Viterbi maximization rule:

$$V(s_i@X_t, Y_u) = p(X_t, Y_u|s_i) \max_{\pi} V(\pi_{s_i@X_t,Y_u}) p(s_i|\pi_{s_i@X_t,Y_u})$$

- Note that the emission term might emit either *one* or *both* of X_t, Y_u.
- Note that the transition term depends only on the *hidden states* i.e. $p(s_i|s_j)$, not the observation times *t,u*.

Second, we also store the best predecessor state so that we can later reconstruct the best path:

$$R(s_i@X_t, Y_u) = \text{argmax}_{\pi} V(\pi_{s_i@X_t,Y_u}) p(s_i@X_t, Y_u|\pi_{s_i@X_t,Y_u})$$

Alignment Posterior

Given two sequences $\vec{X}^m$, $\vec{Y}^n$, we would like to compute the posterior probability that two specific letters X_t, Y_u are aligned to each other.

- assume that we have a three state alignment HMM model (**m**, **x**, **y**), with all emission and transition probabilities given.
- Propose a basic expression for computing $p(\mathbf{m}@t, u|\vec{X}, \vec{Y})$ that indicates how this could be broken into a separate "forward" and "backward" component.

Alignment Posterior Answer

We first apply Bayes' Law:

$$p(\mathbf{m}@t, u|\vec{X}, \vec{Y}) = \frac{p(\mathbf{m}@t, u, \vec{X}, \vec{Y})}{p(\vec{X}, \vec{Y})}$$

and then our usual forward-backward split:

$$= \frac{p(\mathbf{m}@t, u, \vec{X}_{[1,t]}, \vec{Y}_{[1,u]})p(\vec{X}_{[t+1,m]}, \vec{Y}_{[u+1,n]}|\mathbf{m}@t, u)}{p(\vec{X}, \vec{Y})}$$

Alignment Forward Probability

Propose a recursion for the forward probability $p(\mathbf{m}@t, u, \vec{X}_{[1,t]}, \vec{Y}_{[1,u]})$

Alignment Forward Probability Answer

Based on our state graph, we can arrive at **m** from either **m**, **x**, or **y** states. This gives us our forward calculation:

$$p(\mathbf{m}@t, u, \vec{X}_{[1,t]}, \vec{Y}_{[1,u]}) = [\ p(\mathbf{m}@t-1, u-1, \vec{X}_{[1,t-1]}, \vec{Y}_{[1,u-1]})p(\mathbf{m}|\mathbf{m})$$

$$+p(\mathbf{x}@t-1, u, \vec{X}_{[1,t-1]}, \vec{Y}_{[1,u]})p(\mathbf{m}|\mathbf{x})$$

$$+p(\mathbf{y}@t, u-1, \vec{X}_{[1,t]}, \vec{Y}_{[1,u-1]})p(\mathbf{m}|\mathbf{y})\ ]\ p(X_t, Y_u|\mathbf{m})$$

Forward-Backward Computational Complexity

Say we have an HMM with N variables each with M states, and wish to compute posterior probabilities $p(\theta_t = s_i|\vec{X})$ given an observation sequence $\vec{X}$.

- What is the computational complexity in big-O notation for computing $p(\theta_t = s_i|\vec{X})$ for a *single* variable θ_t?
- What is the computational complexity in big-O notation for computing $p(\theta_t = s_i|\vec{X})$ for *all* variables θ_t?

Forward-Backward Computational Complexity Answer

- for a single variable, we would have to compute the forward probability *up to* θ_t, and the backward probability *after* θ_t. This will be $O(NM^2)$.
- for all variables, we would have to compute the forward probability across the entire matrix, and the backward probability across the entire matrix. This will be $O(2NM^2)$, which in big-O terms is just $O(NM^2)$.

Concretely, it only takes twice as long to get *all* the posteriors as it does to calculate *one* posterior!

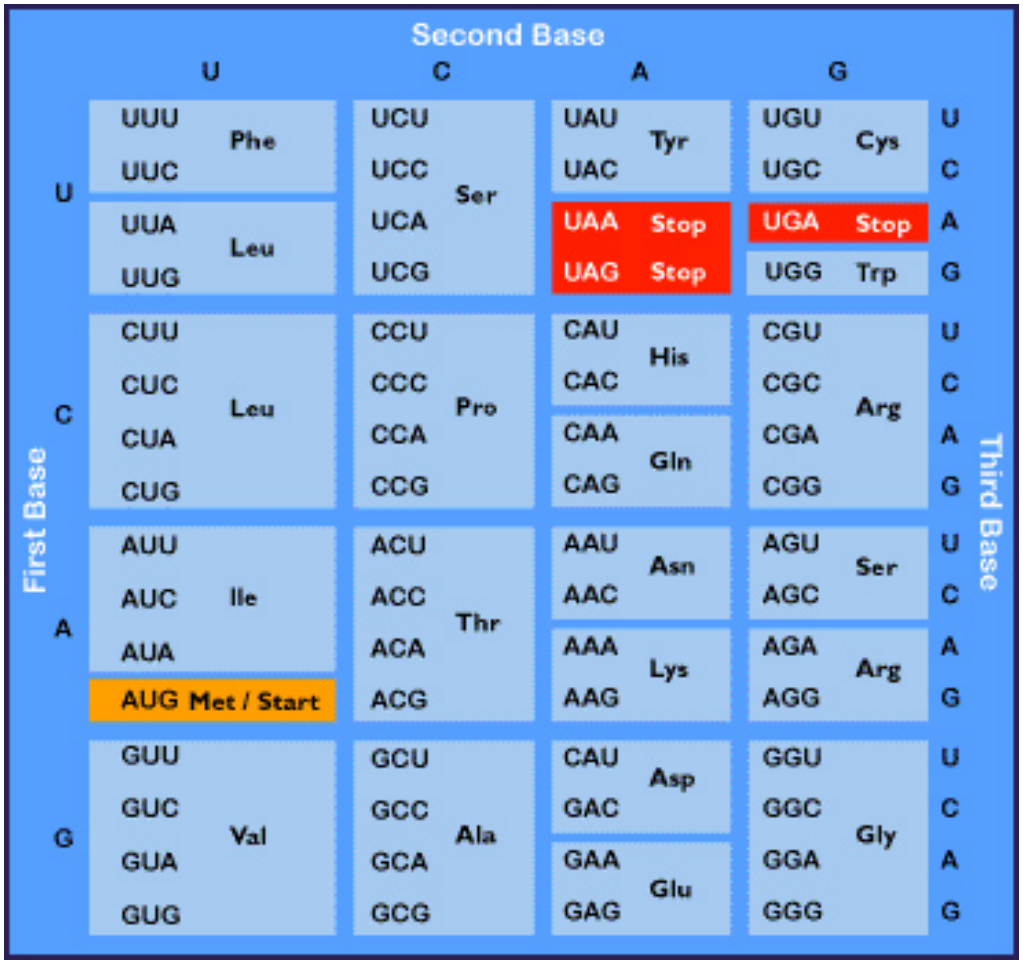
What About Longer Range Dependencies?

Say a hidden variable θ_t depends on more information than just θ_{t-1}.
Can this be handled in a Markov model?
Two ways of solving this:

- explicitly add more dependency edges, e.g. $p(\theta_t|\theta_{t-2}, \theta_{t-1})$ etc. Increases the computational complexity, but it also requires changes to the structure of your algorithm (think of this as additional "inner loops" in your algorithm).
- You can achieve the same thing without forcing changes to your algorithm, by creating "duplicate" states that represent the same state with *different* precursors. Then you can feed this state graph to a standard (first-order) HMM algorithm and it will work without requiring any code changes.

Example: Exon Prediction with Phase Information

A gene consists of a protein coding sequence consisting of three nucleotide "words" (called codons) encoding the 20 amino acids.

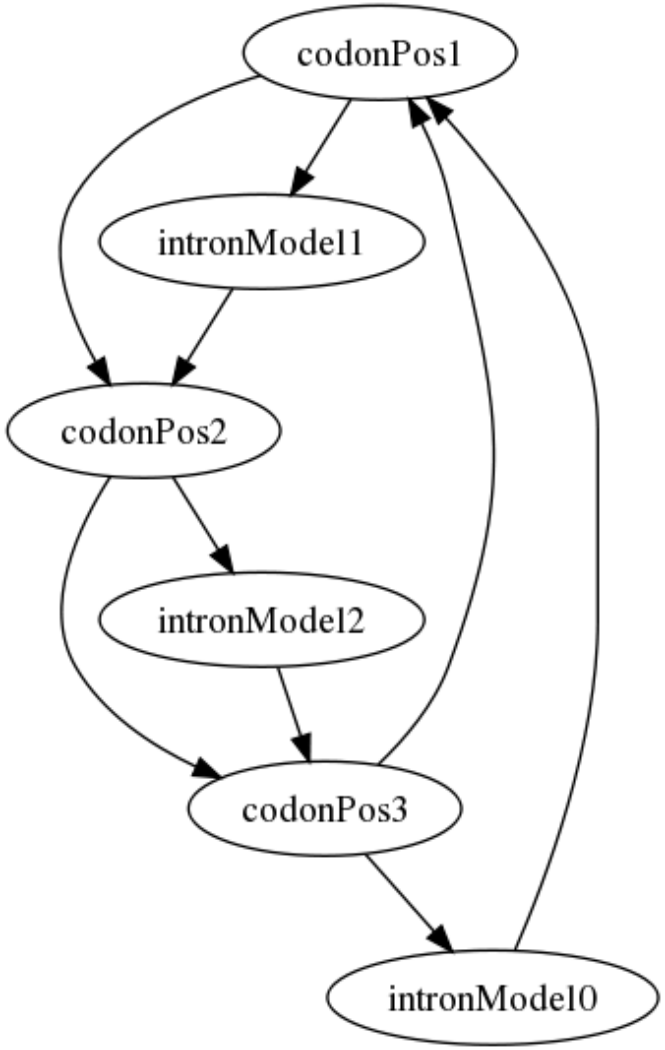


These *trinucleotide* frequencies cannot be encoded by a simple pairwise conditional probability $p(X_{t+1}|X_t)$.

Use Separate States for Different Codon Positions

- We can solve this by using a different set of states for codon position 1, codon position 2, and codon position 3.
- Position 1 states can only transition to position 2 states, $2 \rightarrow 3$ and $3 \rightarrow 1$.
- A further complication: genes of higher organisms are encoded by multiple segments ("exons") separated by "intron" segments that are *removed* (spliced) before translating to protein. An intron can occur anywhere in a coding sequence, e.g. between codon positions 1 and 2, 2 and 3, or 3 and 1. These are called a "phase 1 intron", phase 2 intron, or phase 0 intron.
- In this case, a gene prediction HMM must track *both* splicing signals that indicate a possible exon-intron transition, *and* the coding sequence phase from the end of one exon to the start of the next.

Duplicate Intron Models for Different Phases



Duplicate Intron Models for Different Phases

- since there are only three phases, an easy solution is just to duplicate the intron model for each separate phase.
- the three intron models are identical (e.g. intron start site, end site and other features).
- Having three of them simply tracks which phase our codon ended at, so that we start at the right codon position in the next exon.
- This captures very long-range dependencies, e.g. an intron could be 100,000 bp long!
- Note that even though this state graph structure will work in a standard (first-order) HMM algorithm, the computational complexity is *still* increased, because we are creating extra states.

Viterbi Limitations?

Which of the following are valid statements about limitations of the Viterbi algorithm?

1. Since it is based on a recursion rule, it must be computed using a recursive algorithm.
2. It is restricted to systems that obey a first-order Markov property.
3. It may not tell you anything meaningful about the hidden states.

Viterbi Limitations? Answer

- It's possible to calculate the Viterbi path with a non-recursive algorithm.
- The Viterbi algorithm can be applied to higher order Markov chains; the computational complexity goes up proportionally.
- The Viterbi path may be largely meaningless, if there is substantial uncertainty about hidden states. The problem is that it does not tell you a posterior confidence for any of its hidden state inferences.
- For example, what if *many* paths had almost as high probability as the Viterbi path?

Viterbi Limitations? Errors

- Some people asserted that Viterbi guarantees an optimal hidden state path, so it gives valuable information about hidden variables. But this doesn't consider exactly what's guaranteed: merely that there's no other *better* path. However, the probability could be spread out evenly over many (or even all possible) paths, in which case the "optimal" path is quite misleading, i.e. it implies a specific path when actually there is no such specificity.
- Some people asserted Viterbi is limited to first-order Markov chains. No; it's general, but obviously its computational complexity will grow for higher order chains.
- Some people assumed the "recursive form" of Viterbi restricts us to recursive implementations. No; we can implement it non-recursively.