PROFESSOR: Any questions from last time about Gibbs Sampling? No? So at the end, we introduced this concept of relative entropy. So I just wanted to briefly review this, and make sure it's clear to everyone.

So the relative entropy is a measure of distance between probability distributions-- can be written different ways, often with this D p q notation. And as you'll see, it's the mean bit-score, if you're scoring a motif with a foreground model, pk, and a background model, qk it's the average log odd score under the motif model. And I asked you to show that under the special case where qk is 1 over 4 to the w, that is uniform background, that the relative entropy of the motif ends up being simply 2w minus 8.

Did anyone have a chance to do this? It's pretty simple-- has anyone done this? Can anyone show this? Want me to do it briefly? How many would like to actually see this derivation? It's very, very quick. Few people, OK. So I'll just do that really quick.

So summation Pk log Pk over qk equals-- so you rewrite it as a difference, the log of a quotient is the difference of the log. Of summation log Pk plus summation Pk log qk.

OK, and then the special case that we're dealing with here is that qk is equal to a quarter, if we're dealing with the simplest case of a one-base motif. And so you recognize that that's minus H of Pp, right? H of p is defined as minus that, so it's minus H p. And this here, that's just a quarter. Log 2 of a quarter is minus 2. You can take the minus 2 outside of the sum, so you're ending up with minus 2-- I'm sorry.

How come Sally didn't correct me? Usually she catches these things. So that's a minus there, right, because we're taking the difference. And so then we have a minus 2 that we're pulling out from this, and you're left with summation Pk. And summation Pk, it sums to 1. So that's just 1. And so this equals minus minus 2, or 2 minus H of p.

And there are many other results of this type that can be shown in information theory. Often there are some simple results you can get simply by using this by splitting it into different terms, and summing. So another result that I mentioned earlier, without showing, is that if you have a motif, say, of length 2, that the information content of that motif model can be broken into the information content of each position if your model is such that the positions are independent.

So you would have, in that case-- let's just take the entropy of a model on [? dinucleotides. ?] It that would be minus summation Pi Pj log Pi Pj, if you have a model where the two are independent, and this sum would be taken over both i and j. And so if you want to show that this is equal to-- I claim that this is equal to the i-- Anyway, if you have different positions, in general-- this would be the more general term-- where you have two different compositions at the two positions for the motif. And then you can show that it's equal to basically the sum the entropies at the two positions.

OK, you do the same thing. You separate out the log of the sum, in terms of the sum of the logs, and then you do properties of summations until you get the answer. OK, so this is your homework, and obviously it won't be graded. But we'll check in next Thursday and see if anyone has questions with that.

So what is the use of relative entropy? the main use in bio-informatics is that it's a measure that takes into account non-uniform backgrounds. The standard definition of information basically works when the background is uniform, but falls apart when it's non-uniform. So if you have a very biased genome, like this one shown here which is 75% A T, then the information content using the standard method would be two bits of this motif, which is P C equals 1.

But then, that would predict, using the formula, that a motif occurs 2 to the information content-- once every 2 to the information content bases-- that would be 2 to the 2, which would be 4 bases, and that's clearly incorrect in this case. But the relative entropy, if you do it, there will be four terms, but three of them just have a 0. And then one of them has a 1, so it's 1 times log 1 over 1/8, in this case, and that's will be equal to 3. And so the relative entropy clearly gives you a more sensible version.

It's a good measure for non-uniform backgrounds. Questions about relative entropy?

All right, so then we said you can use a weight matrix, or a position-specific probability matrix, for a motif like this five-prime splice site motif, assuming independence between positions. But if that's not true, then a natural generalization would be an inhomogeneous Markov model. So now, we're going to say that the base at position k depends on the base at position k minus 1, but not on anything before that.

And so, the probability of generating a particular sequence, S1 to S9, is now given by this expression here, where you have for every base after the first, you have a conditional probability. This is the conditional probability of seeing the base, S2, at position minus 2, given that you saw S1 at position minus 3, and so forth. And again, you can take the log for convenience, if you like.

So I actually implemented both of these models. So just for thinking about it, if you want to implement this, you have parameters-- these conditional probability parameters-- and you estimate them as shown here. So remember, conditional probability of A given B is the joint probability divided by the probability of B. And so in this case, that would be the joint probability of seeing C A at minus 3, minus 2, divided by the probability of seeing C at minus 3. You could have the ratio of the frequencies, or, in this case, the counts, because the normalization constant will cancel. Is that clear?

So I actually implemented both the weight matrix model and a first-order Markov

model of five-prime splice sites, and scored some genomic sequence. And what you can see here, the units are in 1/10th-bit units, is that they both are partially successful in separating real five-prime splice sites-- shown in black from the background, shown in light bars-- but in both cases, it's not a perfect separation. There's some overlap here.

And if you zoom there, you can see that the Markov model is a little bit better. It has a tighter tail on the left. So it's generally separating the true from the decoys a little bit better. Not dramatically better, but slightly better. Yes, question?

AUDIENCE: From the previous slide, could you clarify what the letter R and the letter S are?

PROFESSOR: Yes, sorry about that. R would be the odd ratio-- so it's the ratio of the probability of generating that sequence under the foreground model-- the plus model, we're calling it-- divided by the probability under the background, or minus, model. And then, I think I pointed out last time that when you get products of probabilities, they tend to get very small. This can cause computational problems. And so if you just take the log, you convert it into a sum. And so we'll often use score, or S, for the log of the odds ratio. Sorry, should have marked that more clearly.

So Markov models can improve performance, when there is dependence, and when you have enough data to estimate the increased number of parameters. And it doesn't just have to be dependence on the previous base-- you can have a model where the probability of the next base depends on the two previous bases. That would be called a second-order Markov model, or in general, a K-order Markov model.

Sometimes, these dependencies actually occur in practice. With five-prime splice sites, it's a nice example, because there's probably a couple thousand of them in the human genome, and we know them very well, so you can make quite complex models and have enough data to train them. But in general, if you're thinking about modeling a transcription factor binding site, or something, often you might have dozens or, at best, hundreds of examples, typically. And so you might not have enough to train some of the larger model.

So how many parameters do you need to fit a K-order Markov model? So question first, yeah?

AUDIENCE: [INAUDIBLE] If you're comparing the first-order Markov models with W M M, what is W M M?

PROFESSOR: Weight matrix, or position-specific probability matrix. Just a model of independence between the two.

Coming back to this case. So let's suppose you are thinking about making a K-order Markov model, because you do some statistical tasks and you find there's some dependence between sets of positions in your motif. How many parameters would there be? So if you have an independence model, or weight matrix or position-specific probability matrix, there are four parameters at each position, the probabilities of the four bases. This will be only three free parameters, because the fourth one-- but let's just think about it as four, four parameters times the width of the motif.

So if I now go to a first-order Markov model, now there's more parameters, because I have these conditional probabilities at each position. So how many parameters are there? For a first-order Markov? How many do I need to estimate? Yeah, Kevin?

AUDIENCE: I think it would be 16 at each position.

PROFESSOR: Yeah, 16 at each position, except the first position, which has four. OK, and what about a second-order Markov model, where you condition on the two previous positions? 64, right? Because you have two possible bases you're conditioning on, that's 16 possibilities times 4. And so in general, the formula is 4 to the k plus 1.

This is really the issue-- if you have only 100 sequences, and you need to estimate 64 parameters at each position, you don't have enough data to estimate those. So you shouldn't use such a high order model.

All right, so let's think about this-- what could happen if you don't have enough data

to estimate parameters, and how can you get around that? So let's just take a very simple example. So suppose you were setting a new transcription factor. You had done some sort of pull-down assay, followed by, say, conventional sequencing, and identified 10 sequences that bind to that transcription factor.

And these are the 10 sequences, and you align them. You see there is sort of a pattern there-- there's usually an A at the first position, and usually a C at the second, and so forth. And so you consider making a weight matrix model. Then you tally up-- there's eight A's, one C, one G, and no T's at the first position. So how confident can you be that T is not compatible with binding of this transcription factor? Who thinks you can be very confident?

Most of you are shaking your head. So if you're not confident, why are you not confident? I think-- wait, were you shaking your head? What's the problem here?

It's just too small a sample, right? Maybe T occurs rarely. So suppose that T occurs at a frequency of 10%, what's the probability of that in natural sequences? And we just have a random sample of those. What's the probability we wouldn't see any T's in a sample of size 10? Anyone have an idea? Anyone have a ballpark number on this? Yeah, Simona?

AUDIENCE: 0.9 to the 10th.

PROFESSOR: 0.9 to the 10th, OK. And what is that?

AUDIENCE: 0.9 is the probability that you grab one, and don't see a T, and then you do that 10 times.

PROFESSOR: Yeah, exactly. In genera it's a binomial thing, but it works out to be 0.9 to the 10th. And that's roughly-- this is like a Poisson. There's a mean of 1, so it's roughly E to the minus 1, so about 35% chance that you don't see any T's. So we really shouldn't be confident. T probably doesn't have a frequency of 0.5, but it could easily have a frequency of 10% or even 5%, or even 15%. And you might have just not seen it.

So you don't want to assign a probability 0 to T. But what value should you assign

for something you haven't seen? Sally?

So, it turns out there is a principled way to do this called pseudocounts. So basically, if you use maximum likelihood estimation, you get-- maximum likelihood, it turns out, is equal to the observed frequency. But if you assume that the true frequency is unknown, but was sampled from all possible, reasonable frequencies-- so that's a Dirichlet distribution-- then you can calculate what the posterior distribution is in a Bayesian framework, given that you observed, for example, zero T's, what's the distribution of that parameter, frequency of T?

And it turns out, it's equivalent to adding a single count to each of your bins. I'm not going to go through the derivation, because it takes time, but it is well described in the appendix of a book called, *Biological Sequence Analysis*, published about 10, 15 years ago by a number of leaders in the field-- Durbin, Eddy, Krogh, and Mitchison. And there's also a derivation of this in the probability and statistics primer.

So basically you just do this poster calculation, and it turns out to be equivalent to adding 1 count. So when you add 1 count-- and then, of course, you re-normalize, and then you get a frequency-- what effectively it does is it will reduce the frequency of things that you observe most commonly, and boost up the things that you don't see, so that you actually end up assigning a probability of 0.07 to T.

Now, if you had a larger sample-- so let's imagine instead of 8 1 1 0, it was 80 10 10 0, you still at a single count. So you can see in that case, you're only going to be adding a very small, close to 1%, for T. So as you get more data, it converges to the maximum likelihood estimate. But it does something more reasonable, more open-minded, in a case where you're really limited in terms of data.

So the limitation-- you always want to be aware when you're considering going to a more complex model to get better predictability-- you want to be aware of how much data you have, and whether you have enough to accurately estimate parameters. And if you don't, you either simplify the model, or if you can't simplify it anymore, consider using pseudocounts. Sometimes you'll see smaller pseudocounts added-- like instead of 1 1 1 1, you might see a quarter, one pseudocount distributed across

the four bins. There's arguments pro and con, which I won't go into.

So for the remainder of today, I want to introduce hidden Markov models. We'll talk about some the terminology, some applications, and the Viterbi algorithm-- which is a core algorithm when using HMMs to predict things-- and then we'll give a couple examples. So we'll talk about the CpG Island HMM, which is about the simplest HMM I could think of, which is good for illustrating the mechanics of HMM. And then a couple later, probably coming into the next lecture, some examples of real-world HMMs, like one that predicts transmembrane helices.

So some background reading for today's lecture that's posted on the course website, there's a nature biotechnology primer on HMMs, there's a little bit in the textbook. But really, if you want to understand the guts of HMMs, you should read the Rabiner tutorial, which is really pretty well done.

For Thursday's lecture, I will post another of these nature biotechnology primers on RNA folding. This one is actually has a little bit more content, takes a little bit longer to absorb probably than some of the others, but still a good introduction to the topic. And then it turns out the text has a pretty good section on RNA folding, so take a look at chapter 11.

So hidden Markov models can be thought of as a general approach for modeling sequence labeling problems-- you have sequences, they might be genomic sequences, protein sequences, RNA sequences. And these sequences have features-- promoters, they may have domains, et cetera, linear motifs. And you want to label those features in an unknown sequence.

So a classical example would be gene finding. You have a genomic sequence, some parts are, say, exon, some are introns. You want to be able to label them, it's not known. But you might have a training set of known exons and introns, and you might learn what the sequence composition of each of those labels looks like, and then make a model that builds things together.

And what they allow you to do, though, with HMMs is to have transition probabilities

between the different states. You could model states, you can model the length of different types of states to some extent-- as we'll see-- and you can model which states need to follow other states. They're relatively easy to design, you can just simply draw a graph. It can even have cycles in it, that's OK.

And they've been described as the LEGOs of computational sequence analysis. They were developed originally in electrical engineering four or five decades ago for applications in voice recognition, and they're still used in voice recognition. So when you are calling up some large corporation, and instead of a person answering the phone, some computer answering the phone and attempting to recognize your voice, it could well be an HMM on the other end, which is either correctly recognizing what you're saying, or not. So you can thank them or blame them, as you wish.

All right, so Markov Model example-- we did this before, imagine the genotype at a particular locus, and successive generations is thought of as a Markov chain. Bart's genotype depends on Homer's, but is conditionally independent of Grandpa Simpson's, given Homer's. So now what's a hidden Markov model?

So imagine that our DNA sequencer is not working that week, we can't actually go in and measure the genotype. But instead, we're going to observe some phenotype that's dependent on genotype. But it's not dependent in a deterministic way, it's dependent in a more complex way, because there's an impact of environment, as well, let's say.

So we're imagining that your genotype at the apolipoprotein locus is correlated with cholesterol, but doesn't completely predict it. So you're homozygous, you tend to have higher LDL cholesterol than you are heterozygous. But there's a distribution depending on how many doughnuts you eat, or something like that. Imagine that we observe that grandpa had low cholesterol, 150, Homer had high cholesterol, and Bart's cholesterol is intermediate.

Now if we had just observed Bart's cholesterol, we would say, well, it could go either way. It could be homozygous or heterozygous. You would just look at the population

frequency of those two, and would use that to guess. But remember, we know his father's cholesterol, which was 250, makes it much more likely that his father was homozygous, and then that, in turn, biases the distribution [? of it. ?] So that'll make it a little bit more likely that Bart, himself, is homozygous, if you didn't know.

So this is the basic idea-- you have some observable phenotype, if you will, that depends, in a probabilistic way, on something hidden. And that hidden thing has some dependent structure to it. And you want to, then, predict those hidden states from the observable data. So we'll give some more examples coming up.

And the way to think about these models, or at least a handy way to think about them, is as generative models. And so this is from the Rabiner tutorial-- you imagine an HMM used in order to generate observable sequences. So there's these hidden states-- think of them as genotypes-- observable-- think of them as the cholesterol levels.

So the way that it works is you choose an initial state from one of your possible hidden states, according to some initial distribution, you set the time variable equal to 1. In this case, it's T, which will, in our case, often be the position in the sequence. And then you choose an observed value, according to some probability distribution, but it depends on what that hidden state was.

And then you transition to a new state, and then you emit another one. So we'll do an example. Let's say bacterial gene finding is our application, and we're going to model a bacterial gene-- these are protein coding genes, only it's got to have a start coat on, it's got to have an open reading frame, and then it's got to have a stop code on it.

So how many different states do we need in our HMM? What should our states be? Anyone? Do you want to make that-- Tim?

AUDIENCE: Maybe you need four states, because the start state, the orf state, the stop state, and the non-genic state.

PROFESSOR: OK, start, orf, stop and then intergenic, or non-genic. OK now, remember these are

the hidden states, so what are they going to emit? They emit observable data, what's that observable data going to be? Sequence, OK. And how many bases of sequence should each of them emit?

AUDIENCE: Well I guess we don't know.

PROFESSOR: You have a choice. You're the model builder, you can do anything you want. 1, 5, 10-- any number of bases you want. And they can emit different things, if you want. This is generative, you can do anything you want-- there will be consequences later, but for now-- I'm going to call this-- go ahead.

AUDIENCE: You could start with the start and the stop states maybe being three.

PROFESSOR: Three, OK. So this is going to emit three nucleotides. How about this state? What should this emit?

AUDIENCE: Any number.

PROFESSOR: Any number? Yeah, OK, Sally?

AUDIENCE: If you let it emit one number, and then add a self-cycle, then that would work.

PROFESSOR: So Sally wants to have this state emit one nucleotide, but she wants it to have a chance of returning to itself. So that then we can have strings of N's to represent intergenic. Does that make sense?

And these, I agree. Three is a good choice, here. If you had this one emit three, as well, then your genes would have to be a multiple of three apart from each other, which isn't realistic. You would miss out on some genes for that. So this has to be able to emit arbitrary numbers. So you could either have it emit an arbitrary number, but it's going to turn out to make the Viterbi algorithm easier if it just emits one and recurs, as Sally suggested.

And then we have our orf state. So how about here? What should we do here?

AUDIENCE: It can be three, and then you put the circle [INAUDIBLE].

PROFESSOR: So I'm going to change the name to Codon, because it's going to emit one codon-- three nucleotides. And then recur to itself. And now what transitions should we allow between states?

AUDIENCE: So start to four, orf to stop, then stop to N, and then to start.

PROFESSOR: Any others? Yeah?

AUDIENCE: N could go to stop, as well.

PROFESSOR: I'm sorry, N could go to stop?

AUDIENCE: Yeah, so that the gene [INAUDIBLE].

PROFESSOR: OK, so that's a question. We're thinking of a gene on the plus strand, a gene could well be on the opposite strand. And so we should probably make a model of where you would hit stop on the other strand, which would emit a triplet of the inverse complement of the stop code, [INAUDIBLE] et cetera. That's true, excellent point. And then you would traverse this whole circle in the opposite direction.

But it wouldn't be the same state. It would be stop-- because it would emit different things. So you'd have minus stop, stop minus strand. And then you'd have some other states there. And I'm not going to draw those, but that's a point. And you could have a teeny one-codon gene if you want, but probably not worth it.

All right, everyone have an idea about this HMM? So this is a model you have to specify in order for this to actually generate sequence. This model will actually generate annotations and sequence. You have to specify where to start, so you have to have some probability of starting, but the first base that you're going to generate is going to begin intergenic, or start, or codon, et cetera. And you might give it a high probability of this, and then it'll generate a label. So for example, let's say N. And then it'll generate a base, let's say G.

And then you look at these probabilities, so the transition probability here, versus this-- you either generate another N, or you generate a start. And let's say you go to

start, and then you'll generate three bases, so A T G. And then you would go to the codon state, you would emit some other triplet, and so forth. So this is a model that will generate strings of annotations with associated bases. Still doesn't predict gene structure yet, but at least it generates gene structures.

All right, so we are going to, for the sake of illustrating the Viterbi algorithm, we're going to use a simpler HMM in that. So this one only has two states, and its purpose is to predict CPG islands in a vertebrate genome. So what are CPG islands? Anyone remember? What is a CPG island? Anyone heard of this? I'm sure some of you.

Well, the definition here is going to be regions of high CNG content, and relatively high abundance of CPG dinucleotides, which are unmethylated. So what is the P here? The p means that the CG we're talking about is C followed by G along the particular DNA strand, just to distinguish it from C base paired with G. We're not talking about a base pair here, we're talking about C and G following each other along the strand.

So this dinucleotide is rare in vertebrate genomes, because CPG is the site of a methylase, and methylation of the C is mutogenic-- it lends to a much higher rate of mutations. so CPGs often mutate away, except for the ones that are necessary. But there are certain regions, often near promoters, that are unmethylated, and therefore, CPGs can accumulate to higher frequencies. And so you can actually look for these regions and use them to predict where promoters are. That's one application.

So they have higher CPG dinucleotide content, and also higher C and G content. The background of the human genome is about 40% C G only, so it's a bit AT rich, and so you see these patches of, say, 50% to 60% C G that are often associated with promoters, with promoters of roughly half of human genes. So we're going to-- I always drop that little clicker thing. Here it is.

We're going to make a model of these, and then run it to predict promoters in the gene. So here's our model. We have two states, we have a genome state-- this sort

of generic position in the genome-- and then we have an island state. We have the simplest possible transitions, you can go genome to genome, genome to island, island to genome, or island to island. So now you can generate islands of arbitrary size, interspersed with genomic regions of arbitrary size.

And then each of those hidden states is going to emit a single base. So a CPG island in this model is a stretch of I states in a row flanked by G states, if you will. Everyone clear on this set up? Good.

So here, in order to fully specify the model, you need to say what all the parameters are. And there are really three class of parameters. There are initiation probabilities-- so the green here is the notation used in the Rabiner tutorial, except they call them [? pi ?] [? j's. ?] So here, I'm going to say it's a 99% chance you start in the generic genome state, and a 1% chance you start in an island state, because islands are not that common.

And then you need to specify transition probabilities. So there's four possible transitions you could make, and you need to assign probabilities to them. So if the average length of an island were 1,000 bases, then a reasonable value for the I to I transition would be 0.999. You have 99.9% chance of making another island, and 0.1% chance of leaving that island state. If you just run that in this generative mode, it would generate a variety of lengths of islands, but on average, they'd be about one kb long, because the probability of terminating is one in 1,000.

And then if we imagine that those one kb islands are interspersed with genomic regions that are about, say, 100 kilo bases long on average, then you would get this [? five ?] [? nines ?] probability for P G G, and 10 to the minus fifth as a probability of going from genome to islands. That would generate widely spaced islands, that are on average 100 kb apart, that are about one kb in length. Is that making sense?

And now the third type of probability we need to specify are called emission probabilities, which are the B J K in Rabiner notation. And this is where the predictive power is going to come in. There has to be a difference in the emissions if you're going to have any ability to predict these features, and so we're going to

imagine that the genome is 40% C G, and islands are 60% C G. So it's a base composition that we're modeling. We're not doing the dinucleotides here, that would make it more complicated. We're just looking for patches of high G C content.

So now we've fully specified our model. The problem here is that the model is written from the hidden generating the observable, and the problem we're faced with, in practice, is that we have the observable sequence, and we want to go back to the hidden. So we need to reverse the conditioning that's in the model. So when you see this type of problem, how do you reverse conditioning? In general, what's a good way to do it?

You see P A, given B, but the model you have is written in P B, given A. What do you do? What's that?

AUDIENCE: Bayes theorem.

PROFESSOR: Yeah, Bayes theorem. Right, let's do base theorem here. Remember, definition of conditional probability, if we have P A given B-- so this might be the hidden states, given the observables-- we want to write that in terms of P B given A. So what we do first? How do we derive Bayes rule?

You first write the definition of conditional probability-- right, that's just the definition. And now what do I do? Split the top part into what?

AUDIENCE: A times P of B given [INAUDIBLE].

PROFESSOR: P B given A. That's just another way of writing joint probability of P A B, using the definition of conditional probability again. So now it's written the other way. That's basically the idea.

So this is the simple form. And like I said, I don't usually call it a theorem, because it's so simple-- it's something you can derive in 30 seconds. It should be called maybe a rule, or something. There is a more general form. This is where you have two states, basically B or not B that you're dealing with. And there's this more general form that's shown on the slide, which is when you have many states. And it

basically is the same idea, it's just that we've rewritten this term, this P B, and we split it up into all the possible states.

The slide starts from PB given A and goes the other-- anyway, so you rewrite the bottom term as a sum of all the possible cases. All right, so how does that apply to HMMs? So with HMMs, we're interested in the joint probability of a set of hidden states, and a set of observable states. So H, capital H, is going to be a vector that specifies the particular hidden state-- for instance, island or genome-- at position 1, that's H1 all the way to H N. So little h's are specific values for those hidden state.

And then, big O is a vector that describes the different bases in the genome. So O1 is the first base in the genome, up to O N. One can imagine comparing two H vectors, one of which, H versus the H primes, what's the what's the probability of this hidden state versus that? You could compare them in terms of their joint probabilities with this model, and perhaps favor those that have higher probabilities. Yeah?

AUDIENCE: [INAUDIBLE] of the two capital H's have any different notation? Like the second one being H prime, or something like that? Or are they supposed to be--

PROFESSOR: With the H, in this case, these are probability statements about random variables. So H is a random variable, which could assume any possible sequence of hidden states. The little h's are specific values. So for instance, imagine comparing what's the probability of H equals genome, genome, genome, versus the probability that H equals genome, genome, island. So the little h's, or the little h primes, are specific instances. The H's is a random variable, unknown. Does that help?

OK, so how do we apply Bayes' rule? So what we're interested in here is the probability that H, this unknown variable that represents hidden states, that it equals a particular set of hidden states, little h1 to h N, given the observables, little o1 to little oN, which is the actual sequence that we see. And we can write that using definition of conditional probability as the joint probability patient of H and O, over the probability of O.

And then Bayes' rule, we just apply conditional probability again. It's P H times P O, given H, over P O. So it turns out that this P O-- so what is P O equals O1 to O N in this model? Well, the model specifies how to generate the hidden states, and how the observables are generated from those hidden states. So P O is actually defined as the sum of P O comma H equals the first possible hidden state, plus the same term for the second. You have to sum over all the possible outcomes of the hidden states, every possible thing.

So if we have a sequence of length three, you have to sum over the possibility that H might be G G G or G G I, or G I G, or G I I, or I G G, et cetera. You have to sum over eight possibilities here. And if the sequence is a million long, you have to sum over 2 to the one millionth possibilities. That sounds complicated to calculate.

So it turns out that there's actually a trick, and you can calculate it. But you don't have to. That's one of the good things, is that we can just treat it as a constant. So notice that the denominator here is independent of the H's. So we'll just treat that as a constant, an unknown constant. And what we're interested in, which possible value of H has a higher probability? So we're just going to try to maximize P H equals H1 to H N-- find the optimal sequence of hidden states that optimizes that joint probability, the joint probability with the observable values, O1 to O N. Is that making sense?

Basically, we want to find the sequence of hidden states, we'll call it H opt. So now H opt here is a particular vector. Capital H, by itself, is a random vector. This is now a particular vector of hidden states, H1 opt through H N opt. And it's defined as the vector of hidden states that maximizes the joint probability with O equals O1 to O N, where that's the observed sequence that we're dealing with.

So now what I'm telling you is if we can find the vector of hidden states that maximizes the joint probability, then that will also maximize the conditional probability of H given O. And that's often the language of linguistics is used, and it's called the optimal parse of the sequence. You'll see that sometimes, I might say that.

So the solution is to define these variables, R I of H, which are defined as the probability of the optimal parse of the subsequence from one to I-- not the whole long sequence, but a little piece of it from the beginning to a particular place in the middle, that ends in state, H. And so first. We calculate R, R 1 1, the probability of generating the first base, given that it ended in hidden state 1.

And then we would do the hidden state 2, and then we basically have to figure out a way, a recursion, for getting the probabilities of the optimal parses ending at each of the states at position 2, given the values at position 1. And then we go, work our way all the way down to the end of the sequence. And then we'll figure out which is better, and then we'll backtrack to figure out what that optimal parse was. We'll do an example on the board, this is unlikely to be completely clear at this point. But don't worry.

So why is this called the Viterbi algorithm? Well, this is the guy who figured it out. He was actually an MIT alum. He did his bachelor's and master's in double E, I don't know, quite awhile ago, '50s or '60s. And later went on to found Qualcomm, and is now big philanthropist, who apparently supports USC. I don't know why he lost his loyalty to MIT, but maybe he'll come back and give us a seminar. I actually met him once.

Let's talk about his algorithm a little more. So what I want to do is I want to take a particular HMM, so we'll take our CPG island HMM, and then we'll go through the actual Viterbi algorithm on the board for a particular sequence. And you'll see that it's actually pretty simple. But then you'll also see that it's not totally obvious why it works. The mechanics of it are not that bad, but the understanding really how it is able to come up with the optimal parses, that's the more subtle part.

So let's suppose we have a sequence, A C G. Can anyone tell me what the optimal parse of this sequence is, without doing Viterbi? With this particular model, these initiation probabilities transitions and emissions? Do you know what it's going to be in advance? Any guesses?

AUDIENCE: How about genome, island, island.

PROFESSOR: Genome, island, island. Because, you're saying, that way the emissions will be optimized, right? Because you'll omit the C's and G's. OK, that's a reasonable guess. Sally's shaking her head, though.

AUDIENCE: The transitional probability from being in the genome is very, very small, and so it's more likely that it'll either only be in the genome or only be in the island.

PROFESSOR: So the transition from going from a genome to island or island to genome is very small, and so she's saying you're going to pay a bigger penalty for making that transition in there, that may not be offset by the emissions. Right, is that your point? Yeah, question?

AUDIENCE: Check here-- when we're talking about the optimal parse, we're saying let's maximize the probability of that letter--

PROFESSOR: The joint probability.

AUDIENCE: Sorry, the joint probability of that letter--

PROFESSOR: Of that [INAUDIBLE] state and that letter.

AUDIENCE: OK, so that means--

PROFESSOR: Or that set of [INAUDIBLE] states and that set of bases.

AUDIENCE: So when we're computing across this three-letter thing, we have to say the probability of the letter, then let's multiply if by the probability of the transition to the next letter, and then multiply it again [INAUDIBLE] and that letter.

PROFESSOR: So let's do this. If A C G is our sequence-- I'm just going to space it out a little bit. Here's our A at position one, here's our C at position two, and our G at position three. And then we have our hidden states. And so we'll write genome first, and then we have island here. And so what is the optimal parse of the sequence from base one to base one, that ends in genome? It's just the one that starts in genome, because it doesn't go-- right, so it's just genome. And what is its probability?

That's how this thing is defined here. This is this R I H thing I was talking about. This is H, and this is I here. So the probability of the optimal parse of the sequence, up to position one, that ends in genome, is just the one that starts in genome, and then emits that base. So what's the probability of starting in genome? It's five nines, right? So that's the initial probability-- 9 9 9 9 9.

And then what's the probability of emitting an A, given that we're in the genome state? 0.3. So I claim that this is the value of R1 of genome, of the genome state. OK, that's the optimal parse. There's only one parse, so there's nothing-- it is what it is. You start here, there's no transitions-- we started here-- and then you emit an A.

What's the probability of the optimal parse ending an island at position one of the sequence? Someone else? Yeah, question?

AUDIENCE: Why are we using the transition probability?

PROFESSOR: This is the initial-- Oh, I'm sorry. Correct. Thank you, thank you-- what was your name?

AUDIENCE: Deborah.

PROFESSOR: Deborah, OK, thanks Deborah. It should be the initial probability, which is 0.99, good. Initial probability. What about island? Deborah, you want to take this one?

AUDIENCE: 0.01.

PROFESSOR: 0.01 to be an island. And what about the emission probability? We have to start an island, and then emit an A, which is probably what?

AUDIENCE: 0.2.

PROFESSOR: 0.2, yeah, it's up on the screen. Should be, hopefully. Yeah, 0.02. So who's winning so far? If the sequences ended at position one? Genome, genome's winning.

This is a lot bigger than that, it's about 150 times bigger, right? Now what do when we go-- we said we have to do recursion, right? We have to figure out the

probability of the optimal parse ending at position two in each of these states, given the optimal parse ending at position one. How do we figure that out?

What are we going to write here, or what do we have to compare to figure out what to put here? There's two possible parses ending in genome at position two-- there's the one that started in genome at position one, and there's the one that started at island. So you have to compare this to this. So you compare what the probability of this parse was times the transition probability, and then the emission in that state.

So what would that be? What would this be, if we stay in genome? What's the transition? Now we've got our five nines, yeah, good. So five nines. And the emission is what?

AUDIENCE: Genome at 0.2.

PROFESSOR: 0.2, right. And times this. And what about this one? What are we going to multiply when we consider this island to genome transition?

AUDIENCE: [INAUDIBLE] 0.01? Because the genome is still 0.2.

PROFESSOR: It's still 0.2. So we take the maximum of these, right? We're doing optimal parse, highest probability. So which one of these two turns this bigger? Clearly, the top one is bigger. This one is already bigger than this, and we're multiplying by the same data, so clearly the answer here is 0.99 times 0.3 times-- my nines are going to have to get really skinny here-- 0.99999 and 0.2. That's the winner.

And the other thing we do, besides recording this number, is we circle this arrow. Does this ring a bell? This is sort of like Needleman-Wench or Smith-Waterman, where you don't just record what's the best score, but you remember how you got there. We're going to need that later.

And what about here? What's the optimal parse-- what's the probability of the optimal parse ending in island at position two? Or what do I have to do to calculate it? Sorry?

AUDIENCE: You have to calculate the [INAUDIBLE].

PROFESSOR: Right, you consider going genome to island here, and island to island. And who's going to win that race? Do you have an idea? Genome to island had a head start, but it pays a penalty for the transition. The transition is pretty small, that's 10 to the minus fifth.

And what about this one? This has a much higher transition probability of 0.999. And so even though you were starting from something, this is about 150 times smaller than this. But this is being multiplied by 10 to the minus fifth, and this is being multiplied by something that's around 1. So this one will win, island to island will win. Everyone agree on that?

And what will that value be? So it's whatever it was before times the transition, which is what? From island to island? 999. Times the emission which is? 0.3. Island is more likely [? to omit ?] a C. Everyone clear on that?

And then, we're not that until we circle this arrow here. That was the winner, the winner was coming from island, remaining on island. And then we keep going like this.

Do you want me to do one more base? How many people want me to do one more base, and how many people want me to stop this? I'll do one more base, but you guys will have to help me a little bit. Who is going to win-- now we want the probability of the optimal parse ending in G, ending at position three, which is a G, and ending in genome, or ending in island.

So for ending in genome, where is that one going to come from? Which is going to win? This one, or this one?

AUDIENCE: Stay in genome.

PROFESSOR: Yeah, stay in genome is going to win. This one is already bigger than this, and the transition probability here-- this is a 10 to minus 3 transition probability. And this is a probability that's near one, so the transitions are going to dominate here. And so you're going to have this term-- I'm going to call that R 2 of G. That's this notation

here. And times the transition probability, genome to genome, which is all these nines here. And then the emission probability of a G in the genome state is 0.2.

And who's going to win here for the optimal parse, ending at position three, in the island state? Is it going to be this guy, to island, or this one, changing from genome to island? Island to island, because, again, the transmission probability is prohibitive-- that's a 10 to the minus fifth penalty there.

So you're going to stay in island. So this one won here, and this one won here. And so this term here is R 2 of island times the transition probability, island to island, which is 0.999 times the emission probability, which is 0.3 of a G in the island state. Everyone clear on that?

Now, if we went out another 20 bases, what's going to happen? Probably not a lot. Probably the same kind of stuff that's happening. That seems kind of boring, but when would we actually get a cross? What would it take? To push you over and cause you to transition from one to the other?

AUDIENCE: Slowly stacked against you or long enough?

PROFESSOR: Yeah, that's a good way to put it. So let me give you an example. This is the Viterbi algorithm, written out mathematically. We can go over this in a moment, but I just want to try to stay with the intuition here. We did that, now I want to do this.

Suppose your sequence is A C G T, repeating 10,000 times. Can anyone figure out what the optimal parse of that sequence would be, without doing Viterbi in their head? Start and stay in genome. Can you explain why?

AUDIENCE: Because it's equal to the-- what are the [? widths? ?] Because it's homogeneous to the distributor, as opposed to enriched for C N G, and it just repeats without pattern. Or it repeats throughout without [? concentrating the C N Gs ?] anywhere.

PROFESSOR: Right, so the unit of the repeat, this A C G T unit, is not biased for either one. So there will be 2.3 emissions, and 2.2 emissions, whether you go through those in G G G G or in I I I I. Does that make sense? So the emissions will be the same, if

you're all in genome, or if you're all in island.

And the initial probabilities favor genome, and the transitions also favor staying in genome. Right, so all genome. Can everyone see that? So do you want to take a stab at this next one? This one's harder. Let me ask you, in the optimal parse, what state is it going to end in?

AUDIENCE: Genome.

PROFESSOR: Genome, you've got to run [? with a 1,000 ?] T's. And genome, the emissions favor emitting T's. So clearly, it's going to end in genome. And then, what about those runs of C's and G's in the middle there? Are any of those long enough to trigger a transition to island? What was your name again?

AUDIENCE: Daniel.

PROFESSOR: Daniel, so you're shaking your head. You think they're not long enough. So you think the winner's going to be genome all the way? Who thinks they're long enough? Or maybe some of them are? Go ahead, what was your name?

AUDIENCE: Michael.

PROFESSOR: Michael, yeah.

AUDIENCE: The ones at length 80 and 60 are long enough, but the one at length 20 is not.

PROFESSOR: OK, and why do you say that?

AUDIENCE: Just looking at power of 3 over 2, 3 times 10 to the 3 isn't enough to overcome the difference in transition probabilities between the island and genome. But 3 times 10 to the 10 and 1 times 10 to the 14 is, over the length of those sequences. The difference in probability of making that switch once at the beginning [INAUDIBLE].

PROFESSOR: OK, did everyone get what Michael was saying? So, Michael, can you explain why powers of 1.5 are relevant here?

AUDIENCE: Oh, that's the ratio of emission probability between the C states and the G states,

between island and genome. So in island it's 0.3, and in genome it's 0.2 over [INAUDIBLE].

PROFESSOR: Right, so when you're going through a run of C's, if you're in the island state, you get a power of 1.5, in terms of emissions at each position. What about the transitions? You're sort of glossing over those. Why is that?

AUDIENCE: Because that only has to happen once at the beginning. So the ratio between the transition probabilities is really high, but as long as the compounded ratio of the emission probability is high enough over a [INAUDIBLE] of sequences, that as long as that compound emission is greater than that one-off ratio at the beginning, then the island is more [INAUDIBLE].

PROFESSOR: Yeah, so if you think about the transitions, I to I, or G to G, as being close to 1-- so if you think of them as 1, then you can ignore them, and only focus on the cases where it transitions from G to I, and I to G.

So you say that 60 and 80 are long enough. So your prediction is at the optimal parse is G 1,000 I 80 G another 2,020-- you said that one wasn't going to-- and then I 60 G 1,000. Michael, is that what you're saying? Can you read this?

AUDIENCE: Yeah.

PROFESSOR: OK, so why do you say that 10 to the 10th is enough to flip the switch, and 10 to the 3rd is not?

AUDIENCE: If I remember the numbers from the previous slide correctly--

PROFESSOR: A couple of slides back?

AUDIENCE: So if you look at the ratio of the probability of staying in the genome, and the probability of going from the genome to the island, it's--

PROFESSOR: 10 to the 5th.

AUDIENCE: Yeah, 10 to the 5th. So whatever happens going over the next [? run ?] of

sequences has to overcome the difference in ratio for the switch to become more likely.

PROFESSOR: Right. So if everyone agrees that we're going to start in genome, we've got a run of 1,000 A's, and genome is favored anyway-- so that's clear, we're going to be in genome at the beginning for the first 1,000, and be in genome at the end, then if you're going to go to island, you have to pay two penalties, basically. You pay the penalty of starting an island, which is 10 to the minus 5th-- this is maybe a slightly different way than you were thinking about it, but I think it's equivalent-- 10 the minus 5th to switch island, and then you pay a penalty coming back, 10 to the minus 3.

And all the other transitions are near 1. So it's like a 10 to the minus 8 penalty for going from genome to island and back. And so if the emissions are greater than 10 to the 8th, favor island by a gradient of 10 to the 8th, it'll be worth doing that. Does that make sense?

AUDIENCE: I forgot about the penalty of [INAUDIBLE], but it's still the [INAUDIBLE].

PROFESSOR: Yeah, it's still true. Everyone see that? You have to pay a penalty of 10 to the 8gh to go from genome to island and back. But the emissions can make up for that. Even though it seems small, it seems like 60 bases is not enough-- it's multiplicative, and it adds up. Sally?

AUDIENCE: So it seems like the [INAUDIBLE] to me is going to return a lagging answer, because we're not going to actually switch to genome in our HMM until we hit the point where we should [? tip, ?] which would be about 60 G's into the run of 80.

PROFESSOR: So you're saying it's not actually going to predict the right thing?

AUDIENCE: Do you have to rerun the [INAUDIBLE] processing to get it actually in line to the correct thing?

PROFESSOR: What do people think about this? Yeah, comment?

AUDIENCE: That's not quite the case, because you [? pack it ?] or you [? stack it ?] both in the

genome and island possibilities, and your transition is the penalty. So it's the highest impact penalty. So when you island to island to island in that string of 80, the transition will only be valid starting at the first one. [INAUDIBLE]

PROFESSOR: OK, we're at position 1,000. I think you're on the right track here. So I'm going to claim that the Viterbi will transition at the right place, because it's actually proven to generate the optimal parse. So I'm right, but I totally get your intuition.

This is the key thing-- most people's intuition, my intuition, everyone's intuition when they first hear about this is that it seems like you don't transition soon enough. It seems like you have to look into the future to know to transition at that place, right? And obviously you can't look into the future, it's a recursion.

How does it work? Clearly, this is going to be the winner. So let's go to position 1,001, that's the first C. And this guy is going to come from here, this guy is the winner overall-- G 1,000 is clearly the winner.

But what about this guy? Where's it coming from? G 1,000, it's coming from there. And in fact, the previous guy came from G 1,000. I 1,000 came from G 999, and so forth.

Now, here's the interesting question. What happens at 1,002? Sally, I want you to tell me what happens at 1,002. Who wins here?

AUDIENCE: Genome.

PROFESSOR: Genome. Who wins here?

AUDIENCE: Island.

PROFESSOR: Island. It had been transitioning, genome has got a head start, so the best way to beat an island is to have been genome as long as possible, up until position 1,000. And that was still true at 1,001. It's no longer true after that. It was actually better to have transitioned back here to get that one extra emission, that one power of 1.5 from emitting that C in the island state.

If you're going to be in island anyway-- this is much lower than this, at this point. It's about 10 to the 5th lower. But that's OK, we still keep it. It's the best that ends in island. Do you see what I'm saying?

OK, there were all these-- island always had to come from genome at the latest possible time, up until this point, and now it's actually better to have made that transition there, and then stay in island. So you can see island is going to win for awhile, and then it'll flip back. And the question is going to be down here at 1,060, going to 1,061. Who's bigger here?

This guy was perhaps-- well, we don't even know exactly how we got here. But you can see that this parse here that stays in island is going to be optimal. And the question is, would it be just staying in genome? And the answer is yes, because it gained 10 to the 10th in emissions, overcomes the 10 to the 8th penalty that it paid.

Now what do you do at the end? How do you actually find the optimal parse overall? I go out to position whatever it is, 4,160. I've got a probability here, probability here, what do I do with those? Right, but what do I do first? You pick the bigger one, whichever one's bigger.

We decided that this one is going to be bigger, right? And then remember all the arrows that I circled? You just backtrack through and figure out what it was. Does that make sense? That's the Viterbi algorithm.

We'll do it a little bit more on this next time, or definitely field questions. It's a little bit tricky to get your head around. It's a dynamic programming algorithm, like Needleman-Wench or Smith-Waterman, but a little bit different.

The runtime-- what is the runtime, for those who were sleeping and didn't notice that little thing I flashed up there? Or, if you read it, can you explain where it comes from? How does the runtime depend on the number of hidden states and the length of the sequence?

I've got K states, sequence of length L, what is the runtime? So I'm going to put this up here. This might help.

So when you look at the recursion like this, when you want to think about the runtime-- forget about initialization and termination, that's not [INAUDIBLE]. It's what you do on the typical intermediate state that determines the runtime. That's what grows with sequence length. So what do you have to do at each-- to go from position I to position I plus 1? How many calculations?

AUDIENCE: You have to do N calculations for 33. Is that right?

PROFESSOR: Yeah, so 33. Yeah, the notation is a little bit different, but how many-- let me ask you this, how many transitions do you have to consider? If I have an HMM with K hidden states?

AUDIENCE: K squared.

PROFESSOR: K squared, right? So you're going to have to do K squared calculations, basically, to go from position I to I plus 1. So what is the overall dependence on K and L, the length of the sequence? OK, it's K squared L. It's linear in the sequence.

So is this good or bad? Yes, Sally?

AUDIENCE: Doesn't this assume that the graph is complete? And if you don't actually have [INAUDIBLE] you can get a little faster?

PROFESSOR: Yeah, it's a good point. So this is the worst case, or this is in the case where you can transition from any state to any other state. If you remember, the gene finding HMM-- I might have erased it, I think I erased it-- if you can see this subtle--

No, remember Tim designed an HMM for gene-finding here, which only had some of the arrows were allowed. So if that's true, if there's a bunch of zero probabilities for transitions, then you can ignore those, and it actually speeds it up. That's true. It's a good point.

Everyone got this? So this the worst case. K squared L-- is this good or bad? Fast or slow? Slow?

I mean, it depends on the structure of your HMM. For a simple HMM, like the CPG island HMM, this is like blindingly fast. K squared is 4, right? So it takes the same order of magnitude as just reading the sequence. So it'll be super, super fast. If you make a really complicated HMM, it can be slower. But the point is that for genomic sequence analysis, L is big. So as long as you keep K small, it'll run fast.

It's much better than sequence comparison, where you end up with these L squared types of things. So it's faster than sequence comparison. So that's really one of the reasons why Viterbi is so popular, is it's super fast.

So in the last couple minutes, I just want to say a few things about the midterm. You guys did remember there is a mid-term, right? So the midterm is a week from today, Tuesday, March 18. For everybody, it's going to be here, except for those who are in 6874-- and those people should go to 68 180. And because they're going to be given extra time, you should go there early. Go there at 12:40.

Everyone else, who's not in 6784, should come here to the regular class by 1:00 PM, just so you have a chance to get set up and everything. And then the exam will start promptly at 1:05, and will end at 2:25, an hour and 20 minutes.

It is closed book, open notes. So don't bring your textbook, but you can bring up to two pages-- they can be double sided if you want-- of notes. So why do we do this? Well, we think that the act of going through the lectures and textbook, or whatever other notes you have, and deciding what's most important, maybe helpful. And so this, hopefully, will be a useful studying exercise, so figure out what's most important and write it down on a piece of paper if you are likely to forget it-- maybe complicated equations, things like that, you might want to write down.

No calculators or other electronic aids. But you won't need them. If you get an answer that's e squared over 17 factorial-- you're asked to convert that into a decimal. Just leave it like that.

So what should you study? So you should study your lecture notes, readings and tutorials, and past exams. Past exams have been posted to the course website. p-

sets as well. The midterm exams from past years are posted. And there's some variation in topics from year to year, so if you're reading through a midterm from a past year, and you run across an unfamiliar phrase or concept, you have to ask yourself, was I just dozing off when that was discussed? Or was that not discussed this year? And act appropriately.

The content of the midterm will be all the lectures, all the topics up through today-- hidden Markov models. And I'll do a little bit more on hidden Markov models on Thursday. That part could be on the exam, but the next major topic-- RNA secondary structure-- will not be on the exam. It'll be on a p-set in the future. Any questions about the midterm?

And the TAs will be doing some review stuff in sections this week. OK, thank you. See you on Thursday.