Introducción a la Bioinformática:

Problems in Hidden Markov Models (HMMs)

Luis Garreta

Doctorado en Ingeniería
Pontificia Universidad Javeriana – Cali

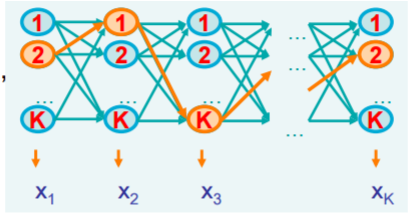
March 29, 2017

What we know

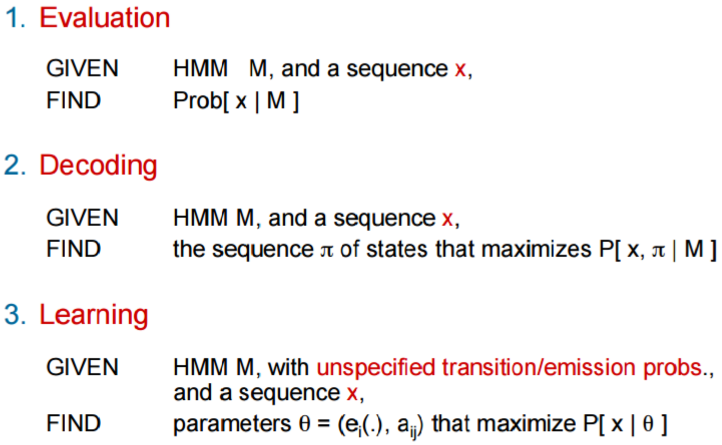
Given a sequence $x = x_1, \ldots, x_N$
and a parse $\pi = \pi_1, \ldots, \pi_N$,

we know how to compute
how likely the parse is:

$P(x, \pi)$



What we would know



1. Evaluation

   GIVEN HMM M, and a sequence x,
   FIND Prob[x | M]

2. Decoding

   GIVEN HMM M, and a sequence x,
   FIND the sequence π of states that maximizes P[x, π | M]

3. Learning

   GIVEN HMM M, with unspecified transition/emission probs., and a sequence x,
   FIND parameters $\theta = (e_i(.), a_{ij})$ that maximize P[x | θ]

Problem 2: Decoding

Find the best parse of a sequence

Viterbi algorithm: *Notation*

Step i	i=0	i=1		i		i=L
Observation X:		x_1	...	x_i	...	x_l

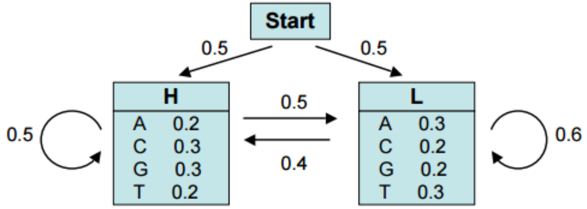
x_i : Observation at step i

$a_{k,l}$: Probability of the transition from state l to k

$e_k(x_i)$: Probability to observe element x_i in state k

$V_k(i)$: Probability of the most probable path ending in state k at position i with observation x_i

Viterbi algorithm Example: *Finding CpG Islands*

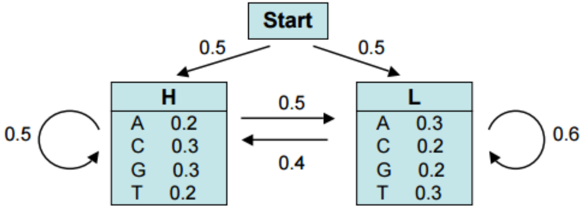


Let's consider the following simple HMM. This model is composed of 2 states, **H** (high GC content) and **L** (low GC content). We can for example consider that state H characterizes coding DNA while L characterizes a non-coding DNA.

The model can then be used to predict the region of coding DNA from a given sequence.

Sources: For the theory, see Durbin *et al* (1998);
For the example, see Borodovsky & Ekisheva (2006), pp 80-81

Viterbi algorithm: *Several Paths*



Consider the sequence S= **GGCACTGAA**

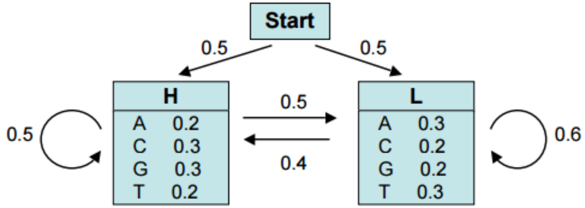
There are several paths through the hidden states (H and L) that lead to the given sequence.

Example: P = **LLHHHHLLL**

The probability of the HMM to produce sequence S through the path P is:

$$p = a_{0L} * e_L(G) * a_{LL} * e_L(G) * a_{LH} * e_H(C) * \ldots$$
$$p = 0.5 * 0.2 * 0.6 * 0.2 * 0.4 * 0.3 * \ldots$$
$$p = \ldots$$

Viterbi algorithm: *A dynamical Programming algorithm*



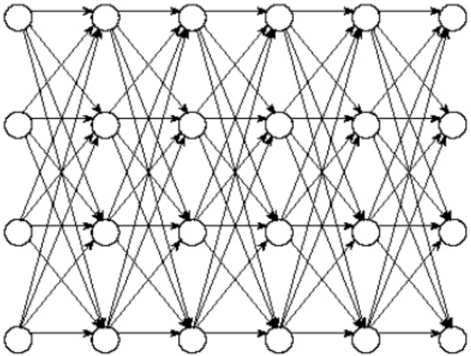
GGCACTGAA

There are several paths through the hidden states (H and L) that lead to the given sequence, but they do not have the same probability.

The **Viterbi algorithm** is a dynamical programming algorithm that allows us to compute the most probable path. Its principle is similar to the DP programs used to align 2 sequences (i.e. Needleman-Wunsch)

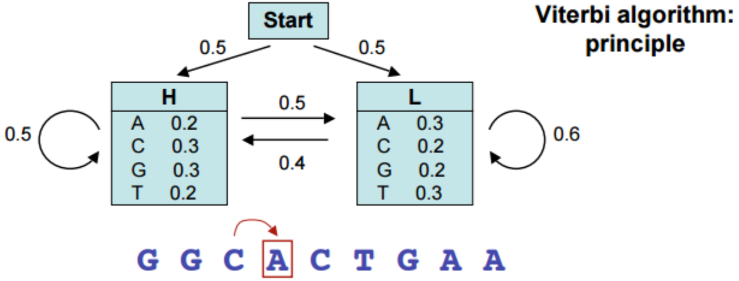
Source: Borodovsky & Ekisheva, 2006

The edit graph for the decoding problem



- The Decoding Problem is essentially finding a longest path in a *directed acyclic graph (DAG)*

Viterbi algorithm: *The most probable path $V_k(i)$*



Suppose the probability $V_k(i)$ of the most probable path ending in state k with observation i is known for all states k, then:

$$V_l(i+1) = e_l(x_{i+1}) \max_k (V_k(i) * a_{kl})$$

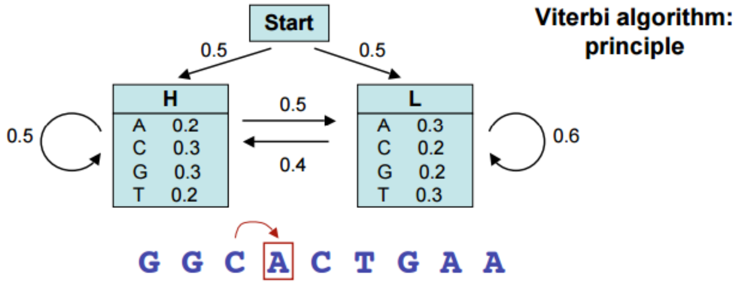
Probability of the most probable path at the end state *l*

Probability to observed element x_{i+1} in state *l*

Probability of the most probable path ending in state *k at* position *i*

Probability of the transition from state *k to state li*

Viterbi algorithm: *The probability of $V_H(4)$*



The probability of the most probable path ending in state **k** with observation "i" is

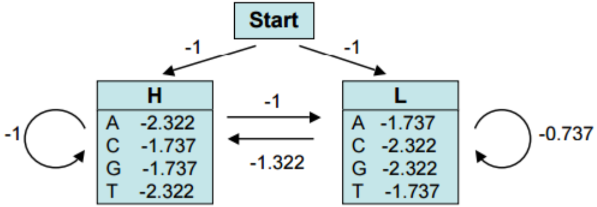
$$V_k(i)=e_k(x_i)\max_s(V_s(i-1)*a_{sl})$$

In our example, the probability of the most probable path ending in state **H** with observation "A" at the 4th position is:

$$V_H(4)=e_H(A)\max_s(V_L(3)*a_{LH},V_H(3)*a_{HH})$$

We can thus compute recursively (from the first to the last element of our sequence) the probability of the most probable path.

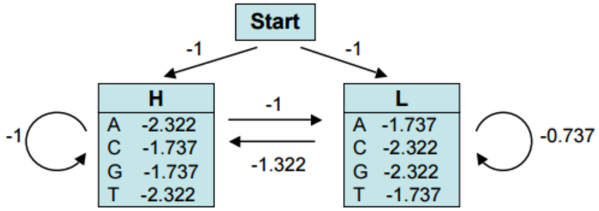
Viterbi algorithm: *Logs instead of Probabilities*



Remark: for the calculations, it is convenient to use the log of the probabilities (rather than the probabilities themselves). Indeed, this allows us to compute *sums* instead of *products*, which is more efficient and accurate.

We used here $\log_2(p)$.

Viterbi algorithm: *Maximum at the first position*



GGCACTGAA

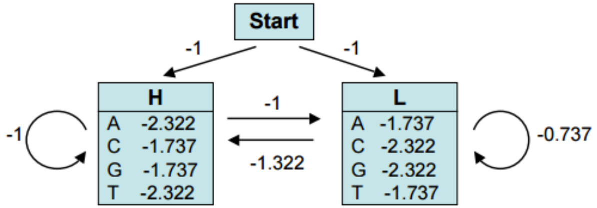
Probability (in $\log_2$) that G at the first position was emitted by state **H**

$$\boxed{V_H(1) = -1 - 1.737 = -2.737 \quad \text{(Maximum)}}$$

Probability (in $\log_2$) that G at the first position was emitted by state **L**

$$V_L(1) = -1 - 2.322 = -3.322$$

Viterbi algorithm: *Maximum at the second position*



GGCACTGAA

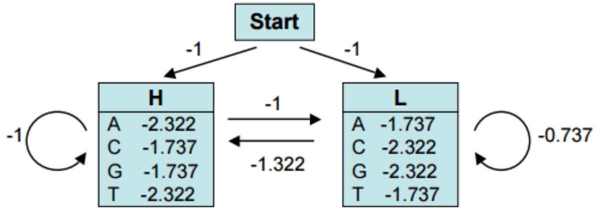
Probability (in $\log_2$) that G at the 2nd position was emitted by state **H**

$$\begin{aligned} V_H(2) &= -1.737 + max\left(V_H(1)+a_{HH}, V_L(1)+a_{LH}\right) \\ &= -1.737 + max\left(-2.737-1, -3.322-1.322\right) \\ &= -5.474 \text{ (obtained from } V_H(1) \text{)} \qquad \text{(Maximum)} \end{aligned}$$

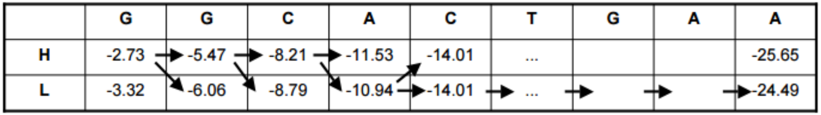
Probability (in $\log_2$) that G at the 2nd position was emitted by state **L**

$$\begin{aligned} V_L(2) &= -2.322 + max\left(V_H(1)+a_{HL}, V_L(1)+a_{¿}\right) \\ &= -2.322 + max\left(-2.737-1, -3.322-0.737\right) \\ &= -6.059 \text{ (obtained from } V_H(1) \text{)} \end{aligned}$$

Viterbi algorithm: *Compute Iteratively the Probabilities*



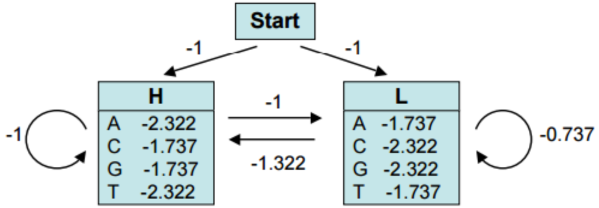
GGCACTGAA



	G	G	C	A	C	T	G	A	A
H	-2.73	-5.47	-8.21	-11.53	-14.01	...			-25.65
L	-3.32	-6.06	-8.79	-10.94	-14.01	...			-24.49

We then compute iteratively the probabilities $V_H(i)$ and $V_L(i)$ that nucleotide x_i at position i was emitted by state H or L, respectively. The highest probability obtained for the nucleotide at the last position is the probability of the most probable path. This path can be retrieved by back-tracking.

Viterbi algorithm: *Backtracking*



GGCACTGAA

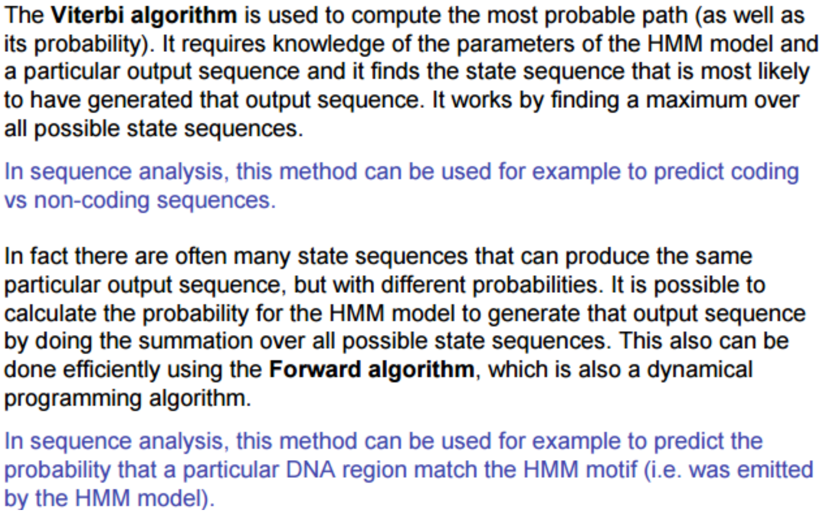
back-tracking
(= finding the path which corresponds to the highest probability, -24.49)

	G	G	C	A	C	T	G	A	A
H	**-2.73**	**-5.47**	**-8.21**	-11.53	-14.01	...			-25.65
L	-3.32	-6.06	-8.79	**-10.94**	-14.01	...			**-24.49**

The most probable path is: **HHHLLLLLL**

Its probability is $2^{-24.49} = 4.25E\text{-}8$
(remember that we used $\log_2(p)$)

Viterbi algorithm: *Remarks*



The **Viterbi algorithm** is used to compute the most probable path (as well as its probability). It requires knowledge of the parameters of the HMM model and a particular output sequence and it finds the state sequence that is most likely to have generated that output sequence. It works by finding a maximum over all possible state sequences.

In sequence analysis, this method can be used for example to predict coding vs non-coding sequences.

In fact there are often many state sequences that can produce the same particular output sequence, but with different probabilities. It is possible to calculate the probability for the HMM model to generate that output sequence by doing the summation over all possible state sequences. This also can be done efficiently using the **Forward algorithm**, which is also a dynamical programming algorithm.

In sequence analysis, this method can be used for example to predict the probability that a particular DNA region match the HMM motif (i.e. was emitted by the HMM model).

The Viterbi algorithm

Input: $x = x_1, \ldots, x_N$

Initialization: $V_0(0) = 1$ (0 is the imaginary first position)

$V_k(0) = 0$, for all $k > 0$

Iteration: $V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i-1)$

$Ptr_j(i) = \text{argmax}_k \, a_{kj} V_k(i-1)$

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

Traceback: $\pi_N^* = \text{argmax}_k V_k(N)$

$\pi_{i-1}^* = Ptr_{\pi i}(i)$

Problem 1: Evaluation

Finding the probability a sequence is generated by the model

Forward algorithm: *Notation*

Step i	i=0	i=1		i		i=L
Observation X:		x_1	...	x_i	...	x_l

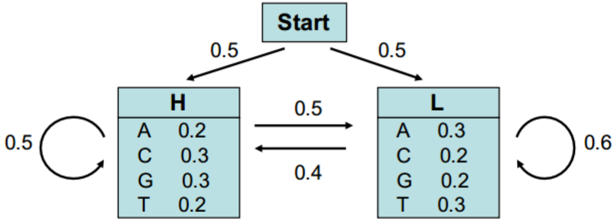
x_i : Observation at step i

$a_{k,l}$: Probability of the transition from state l to k

$e_k(x_i)$: Probability to observe element x_i in state k

$f_k(i)$: Probability of the observed sequence up to and including x_i and ending in state k

Forward Algorithm: *CpG Example*



Consider now the sequence S= **GGCA**

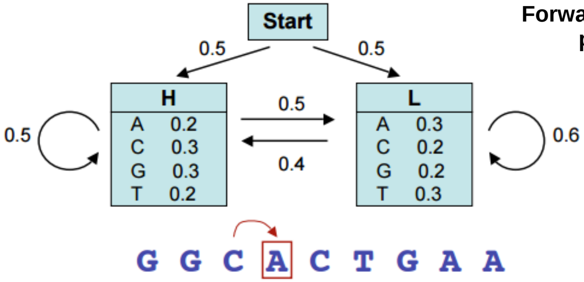
What is the probability P(S) that this sequence S was generated by the HMM model?

This probability P(S) is given by the sum of the probabilities $p_i(S)$ of each possible path that produces this sequence.

The probability P(S) can be computed by dynamical programming using either the so-called **Forward** or the **Backward** algorithm.

Forward algorithm: $f_k(i)$, *Probabilty of x_i ending in state k*

Forward algorithm: principle



Suppose the probability $f_k(i)$ of the observed sequence ut to x_i, ending in state k

$$f_l(i+1)=e_l(x_{i+1})\sum_k f_k(i)*a_{kl}$$

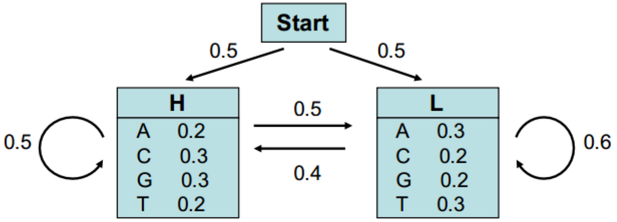
Probability to observed sequence up to x_{i+1} ending in state *l*

Probability to observed element x_i in state *l*

Probability to observed sequence up to x_i ending in state *k*

Probability of the transition from state *k to state li*

Forward Algorithm: *First Step: $f_H(G)$ and $f_L(G)$*

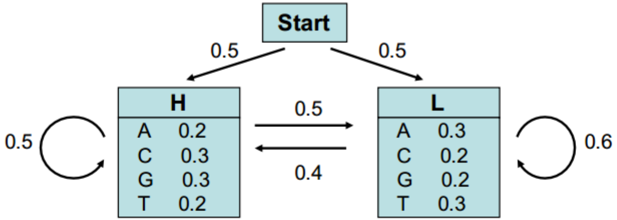


Forward algorithm

Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15			
L	0	0.5*0.2=0.1			

Forward Algorithm: *Second Step: $f_H(G)$*

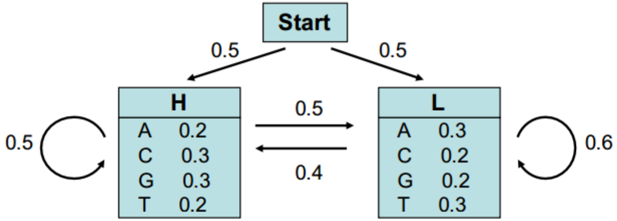


Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15 →	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345		
L	0	0.5*0.2=0.1 →			

S

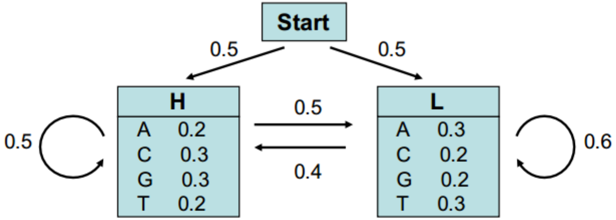
Forward Algorithm: *Second Step: $f_H(G)$ and $f_L(G)$*



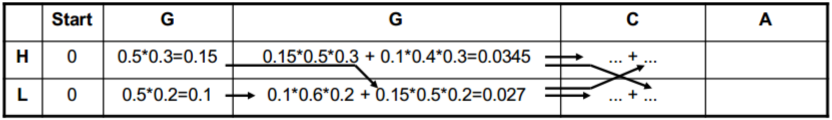
Consider now the sequence S= **GGCA**

	Start	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345		
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027		

Forward Algorithm: *All steps: $f_H(i)$ and $f_L(i)$*

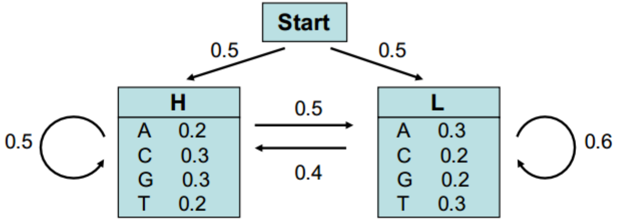


Consider now the sequence S= **GGCA**

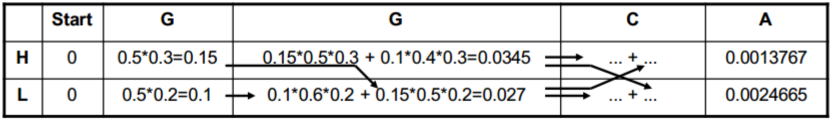


	Start	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345	... + ...	
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027	... + ...	

Forward Algorithm: *Last Step:* $f_H(L)$ *and* $f_L(L)$



Consider now the sequence S= **GGCA**

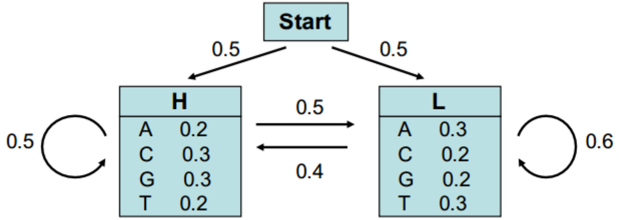


	Start	G	G	C	A
H	0	0.5*0.3=0.15	0.15*0.5*0.3 + 0.1*0.4*0.3=0.0345	... + ...	0.0013767
L	0	0.5*0.2=0.1	0.1*0.6*0.2 + 0.15*0.5*0.2=0.027	... + ...	0.0024665

$\Sigma = 0.0038432$

=> The probability that the sequence S was generated by the HMM model is thus P(S)=0.0038432.

Forward Algorithm: *Significance of the probabilty*



The probability that sequence S="GGCA" was generated by the HMM model is $P_{HMM}(S)$ = 0.0038432.

To assess the significance of this value, we have to compare it to the probability that sequence S was generated by the background model (i.e. by chance).

Ex: If all nucleotides have the same probability, p_{bg}=0.25; the probability to observe S by chance is: $P_{bg}(S) = p_{bg}^4 = 0.25^4 = 0.00396$.

Thus, for this particular example, it is likely that the sequence S does not match the HMM model ($P_{bg} > P_{HMM}$).

NB: Note that this toy model is very simple and does not reflect any biological motif. If fact both states H and L are characterized by probabilities close to the background probabilities, which makes the model not realistic and not suitable to detect specific motifs.

Problem 3: Learning

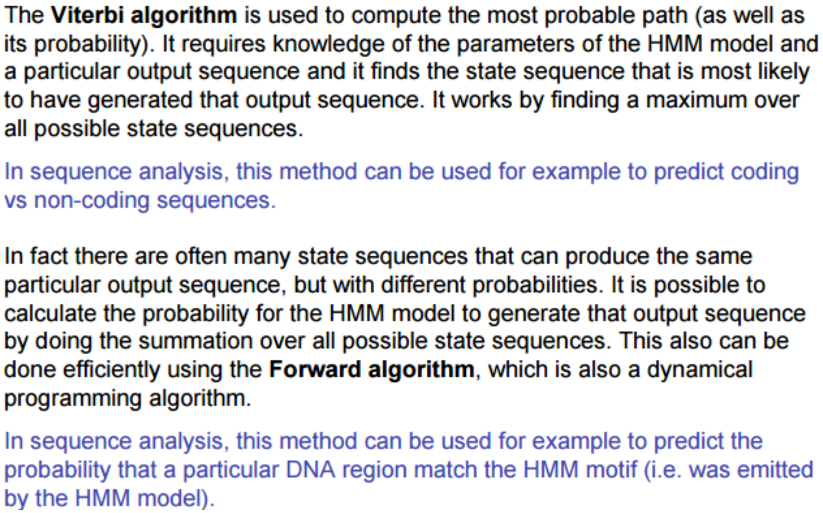
Estimation of the HMM Parameters when state sequence is known

Estimation of the HMM Parameters when state sequence is known

Counting:

- When all the paths are known, we can count the number of times each particular transition or emision is used in the set of training sequences.
- Let be A_{kl} and $E_k(b)$:
    - $a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$
    - $e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$

Summary



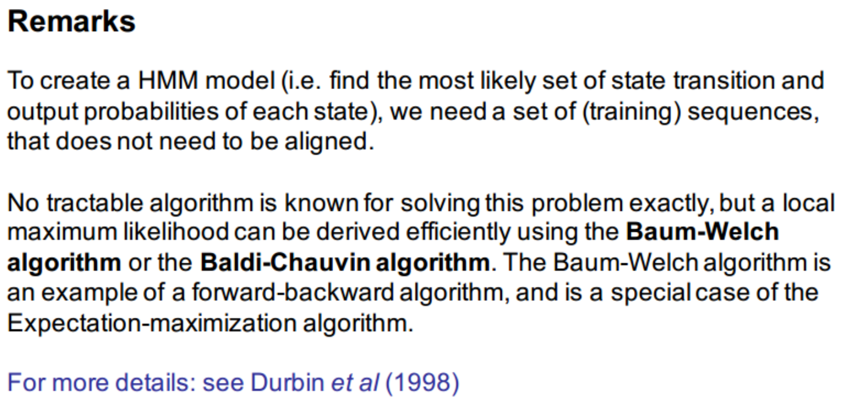
The **Viterbi algorithm** is used to compute the most probable path (as well as its probability). It requires knowledge of the parameters of the HMM model and a particular output sequence and it finds the state sequence that is most likely to have generated that output sequence. It works by finding a maximum over all possible state sequences.

In sequence analysis, this method can be used for example to predict coding vs non-coding sequences.

In fact there are often many state sequences that can produce the same particular output sequence, but with different probabilities. It is possible to calculate the probability for the HMM model to generate that output sequence by doing the summation over all possible state sequences. This also can be done efficiently using the **Forward algorithm**, which is also a dynamical programming algorithm.

In sequence analysis, this method can be used for example to predict the probability that a particular DNA region match the HMM motif (i.e. was emitted by the HMM model).

Summary



Remarks

To create a HMM model (i.e. find the most likely set of state transition and output probabilities of each state), we need a set of (training) sequences, that does not need to be aligned.

No tractable algorithm is known for solving this problem exactly, but a local maximum likelihood can be derived efficiently using the **Baum-Welch algorithm** or the **Baldi-Chauvin algorithm**. The Baum-Welch algorithm is an example of a forward-backward algorithm, and is a special case of the Expectation-maximization algorithm.

For more details: see Durbin *et al* (1998)