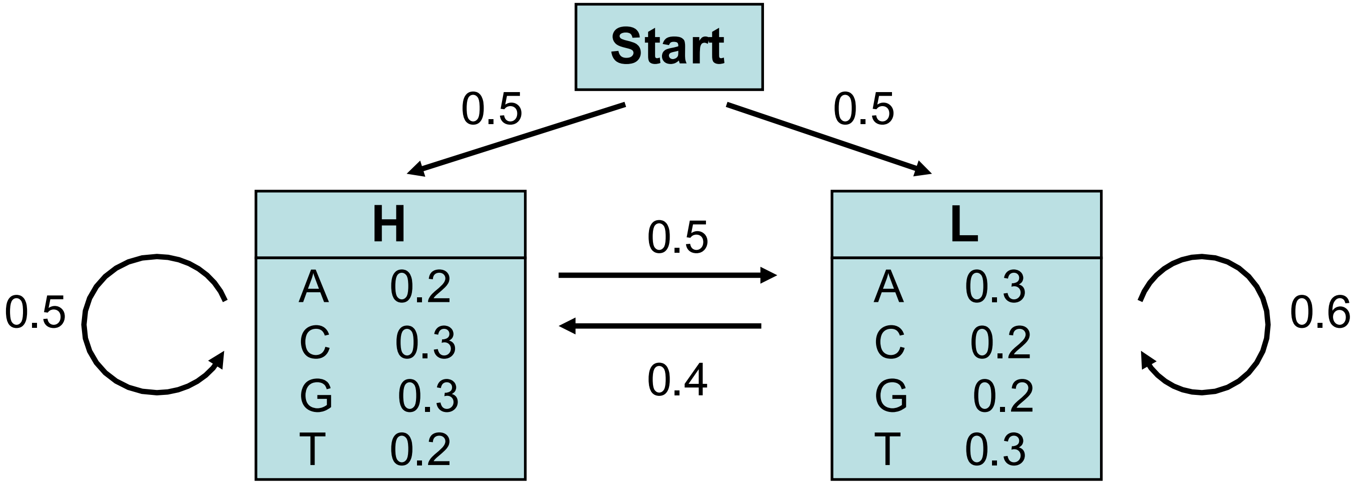
HMM : Viterbi algorithm - a toy example

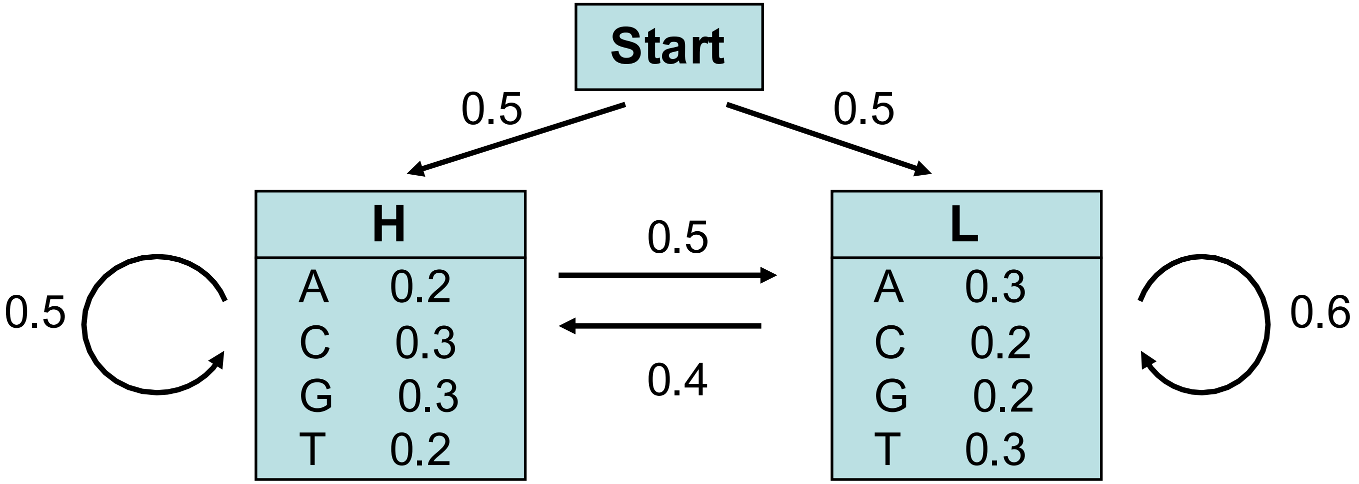


Let's consider the following simple HMM. This model is composed of 2 states, **H** (high GC content) and **L** (low GC content). We can for example consider that state H characterizes coding DNA while L characterizes non-coding DNA.

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

HMM : Viterbi algorithm - a toy example



Consider the sequence S= **GGCACTGAA**

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

Example: P = **LLHHHHLLL**

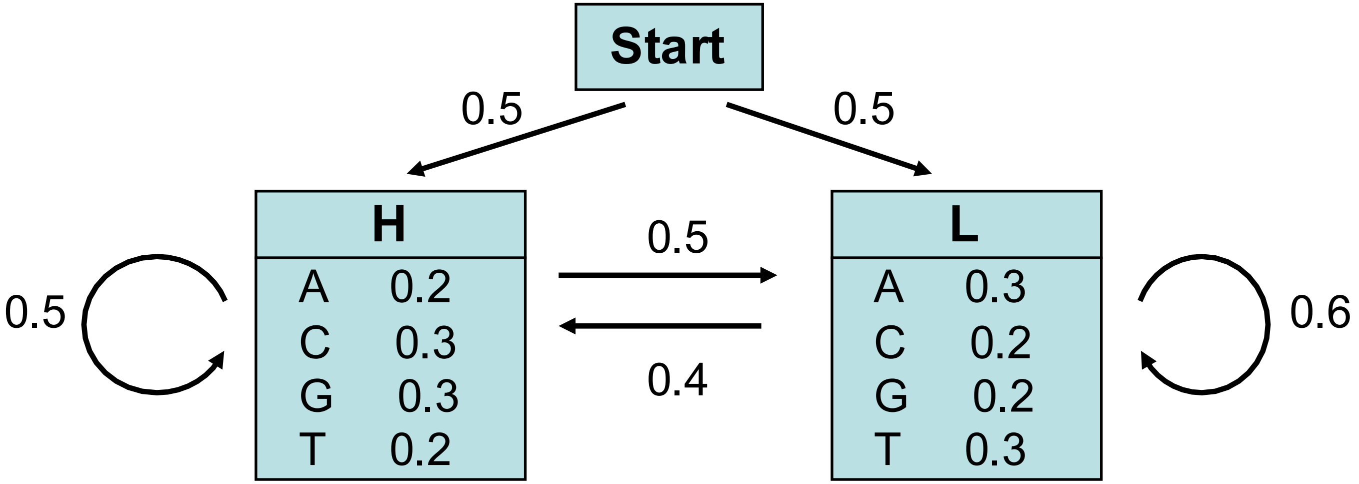
The probability of the HMM to produce sequence S through the path P is:

$$p = p_L(0) * p_L(G) * p_{LL} * p_L(G) * p_{LH} * p_H(C) * \ldots$$

$$= 0.5 * 0.2 * 0.6 * 0.2 * 0.4 * 0.3 * \ldots$$

$$= \ldots$$

HMM : Viterbi algorithm - a toy example



GGCACTGAA

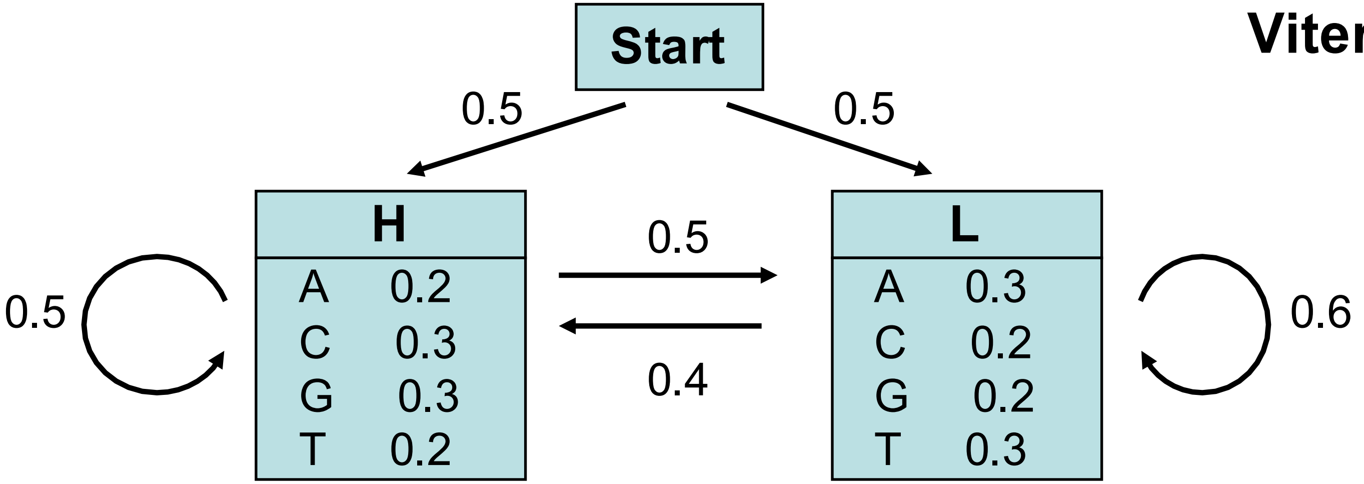
There are several paths through the hidden states (H and L) that lead to the given sequence, but they do not have the same probability.

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

HMM : Viterbi algorithm - a toy example

Viterbi algorithm: principle



G G C A C T G A A

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

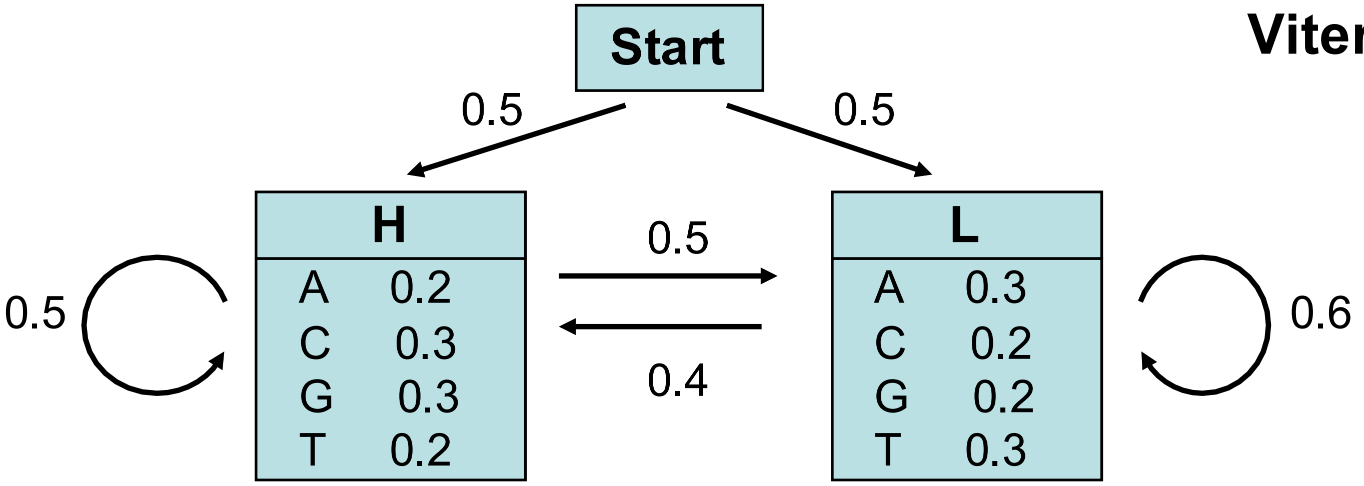
$$p_l(i,x) = e_l(i)\max_k\left(p_k(j,x-1).p_{kl}\right)$$

probability to observe element *i* in state *l*

probability of the most probable path ending at position x-1 in state *k* with element *j*

probability of the transition from state *l* to state *k*

HMM : Viterbi algorithm - a toy example



Viterbi algorithm: principle

G G C A C T G A A

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

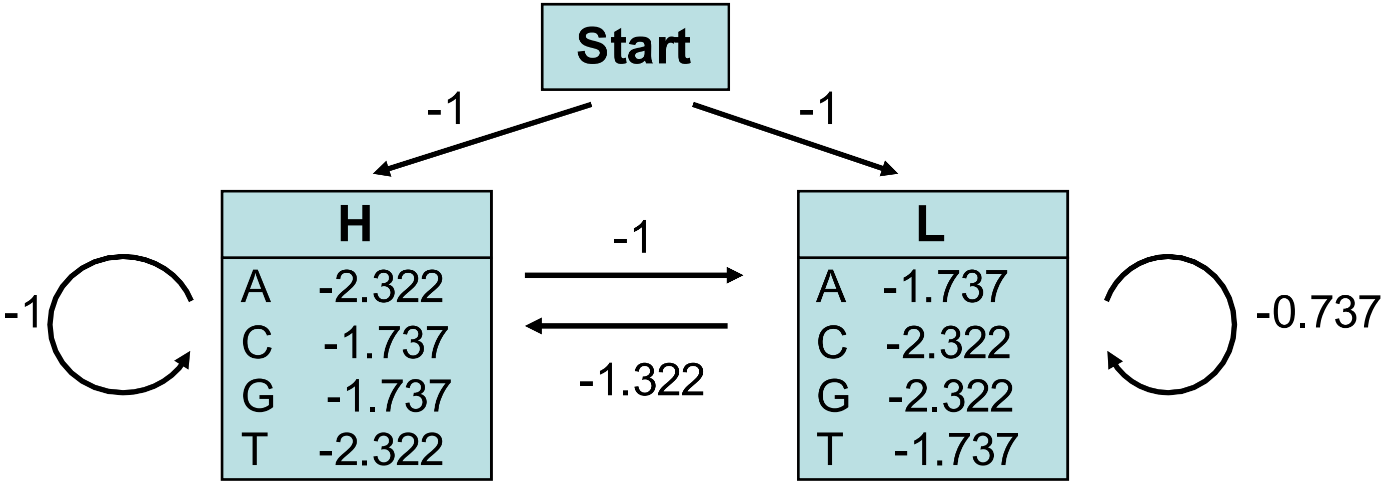
$$p_l(i,x) = e_l(i)\max_k\left(p_k(j,x-1).p_{kl}\right)$$

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

$$p_H(A,4) = e_H(A)\max\left(p_L(C,3)p_{LH}, p_H(C,3)p_{HH}\right)$$

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

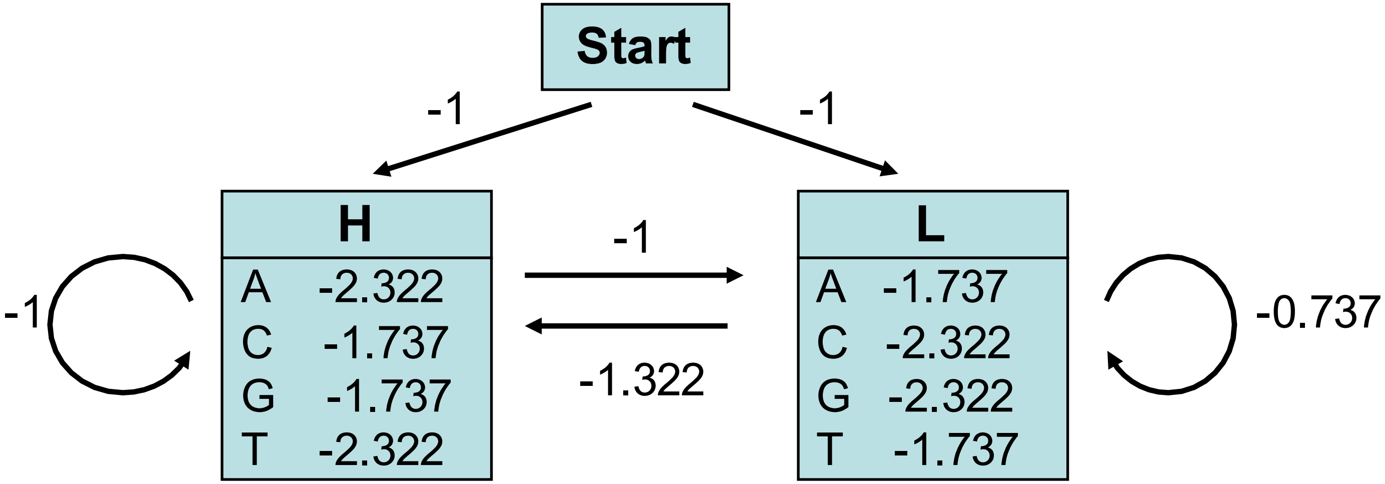
HMM : Viterbi algorithm - a toy example



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 $\log_2(p)$.

HMM : Viterbi algorithm - a toy example



Start

-1

-1

H	
A	-2.322
C	-1.737
G	-1.737
T	-2.322

-1

-1.322

L	
A	-1.737
C	-2.322
G	-2.322
T	-1.737

-1

-0.737

GGCACTGAA

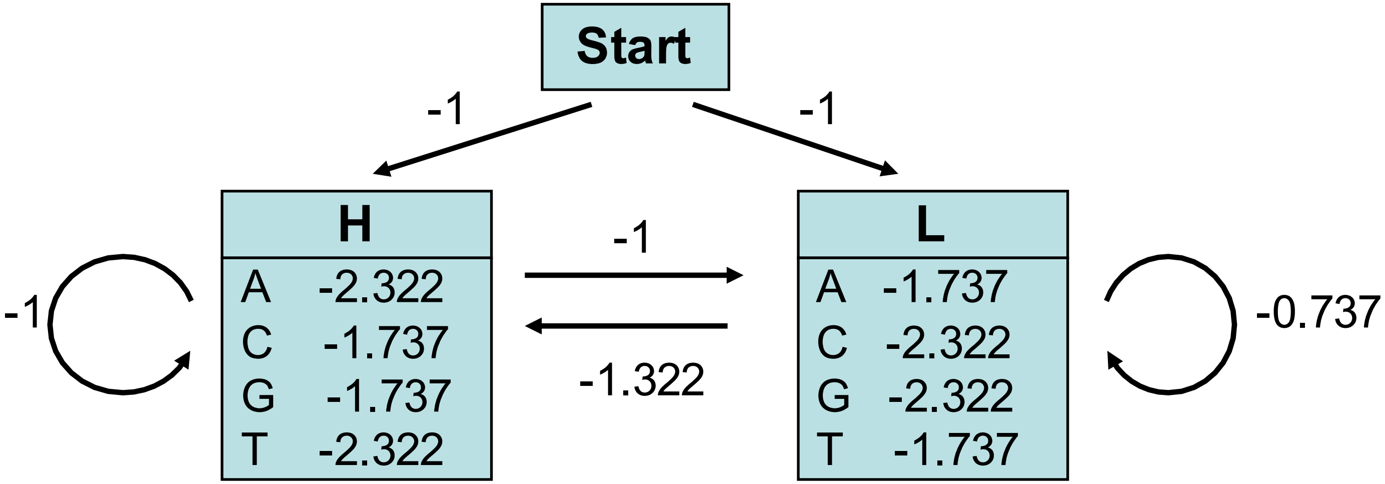
Probability (in $\log_2$) that G at the first position was emitted by state **H**

$p_H(G,1) = -1 -1.737 = -2.737$

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

$p_L(G,1) = -1 -2.322 = -3.322$

HMM : Viterbi algorithm - a toy example



Start

-1

-1

H	
A	-2.322
C	-1.737
G	-1.737
T	-2.322

L	
A	-1.737
C	-2.322
G	-2.322
T	-1.737

-1

-1.322

-1

-0.737

GGCACTGAA

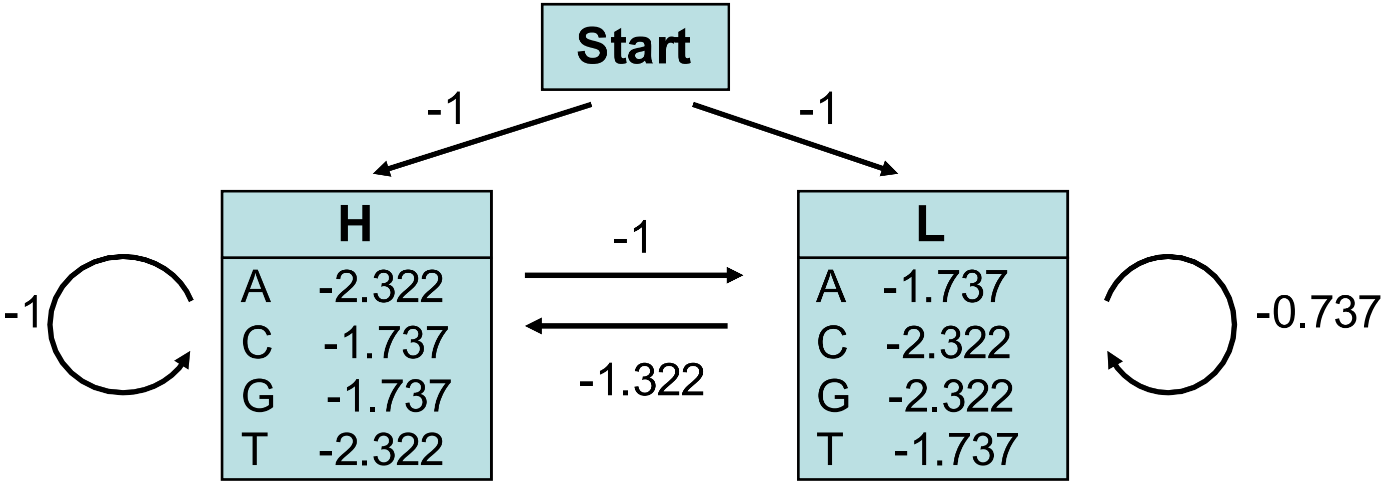
Probability (in $\log_2$) that G at the 2nd position was emitted by state **H**

$$\begin{aligned} p_H(G,2) &= -1.737 + \max\,(p_H(G,1)+p_{HH},\ p_L(G,1)+p_{LH}) \\ &= -1.737 + \max\,(-2.737 - 1 \ ,\ -3.322 - 1.322) \\ &= -5.474 \text{ (obtained from } p_H(G,1)\text{)} \end{aligned}$$

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

$$\begin{aligned} p_L(G,2) &= -2.322 + \max\,(p_H(G,1)+p_{HL},\ p_L(G,1)+p_{LL}) \\ &= -2.322 + \max\,(-2.737 - 1.322 \ ,\ -3.322 - 0.737) \\ &= -6.059 \text{ (obtained from } p_H(G,1)\text{)} \end{aligned}$$

HMM : Viterbi algorithm - a toy example



Start

-1

-1

H	
A	-2.322
C	-1.737
G	-1.737
T	-2.322

-1

-1.322

L	
A	-1.737
C	-2.322
G	-2.322
T	-1.737

-1

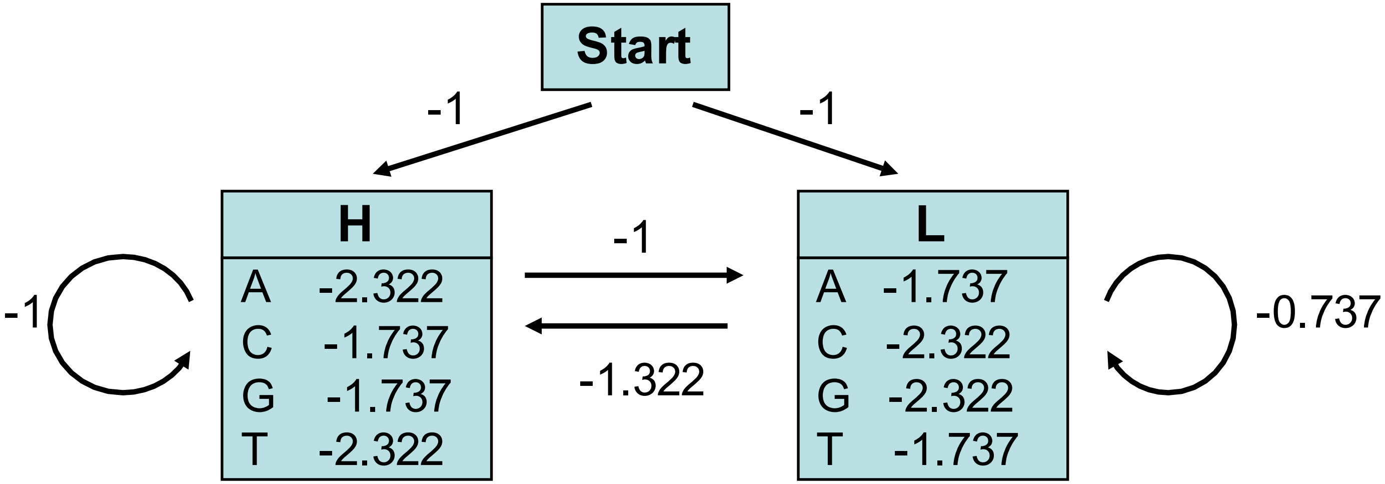
-0.737

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 $p_H(i,x)$ and $p_L(i,x)$ that nucleotide *i* at position *x* 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.

HMM : Viterbi algorithm - a toy example



GGCACTGAA

back-tracking

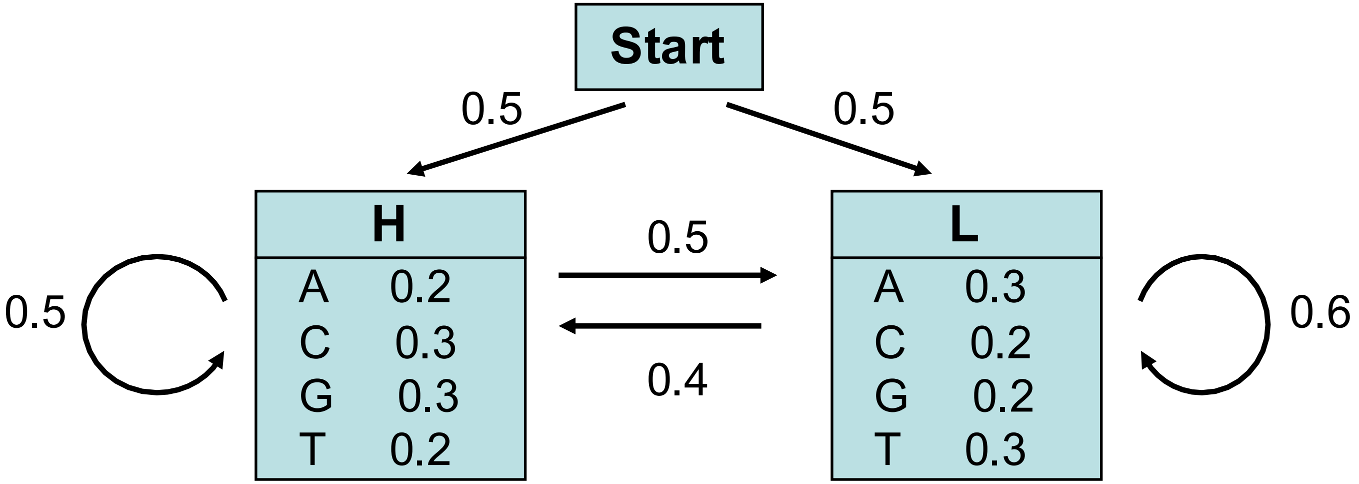
(= finding the path which corresponds to the highest probability, -24.49)

	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-8 (remember that we used $\log_2(p)$)

HMM : Forward algorithm - a toy example



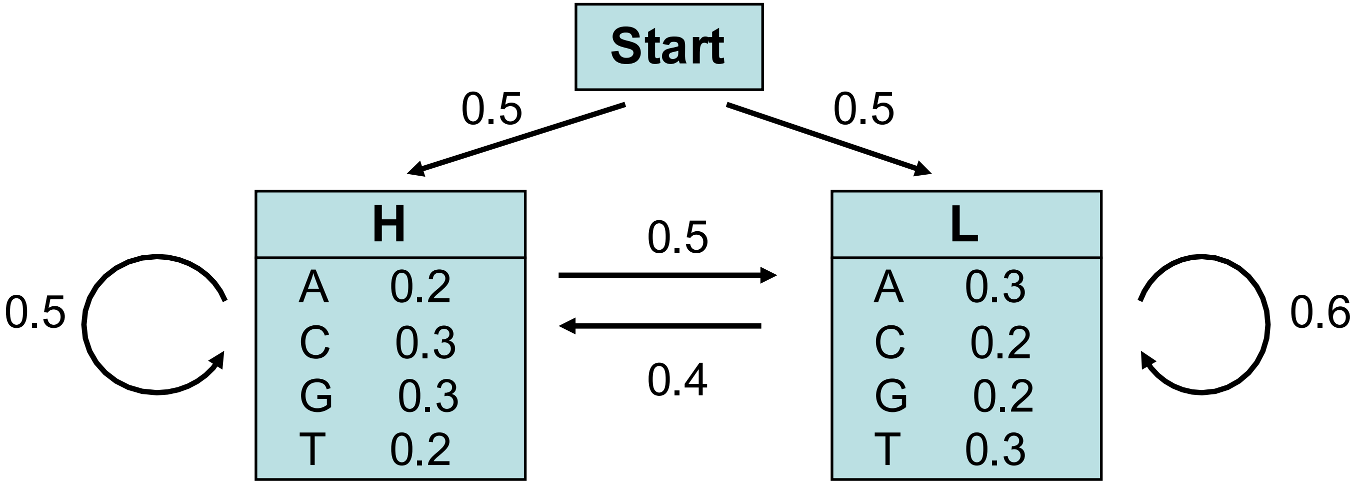
Consider now the sequence S= **GGCA**

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.

HMM : Forward algorithm - a toy example

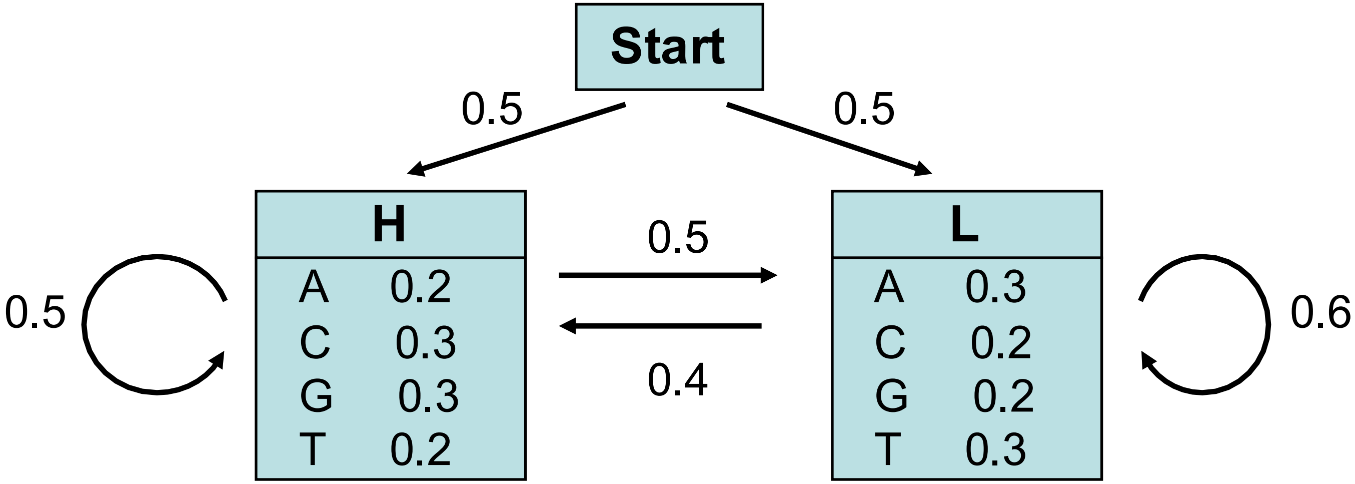


Forward algorithm

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			

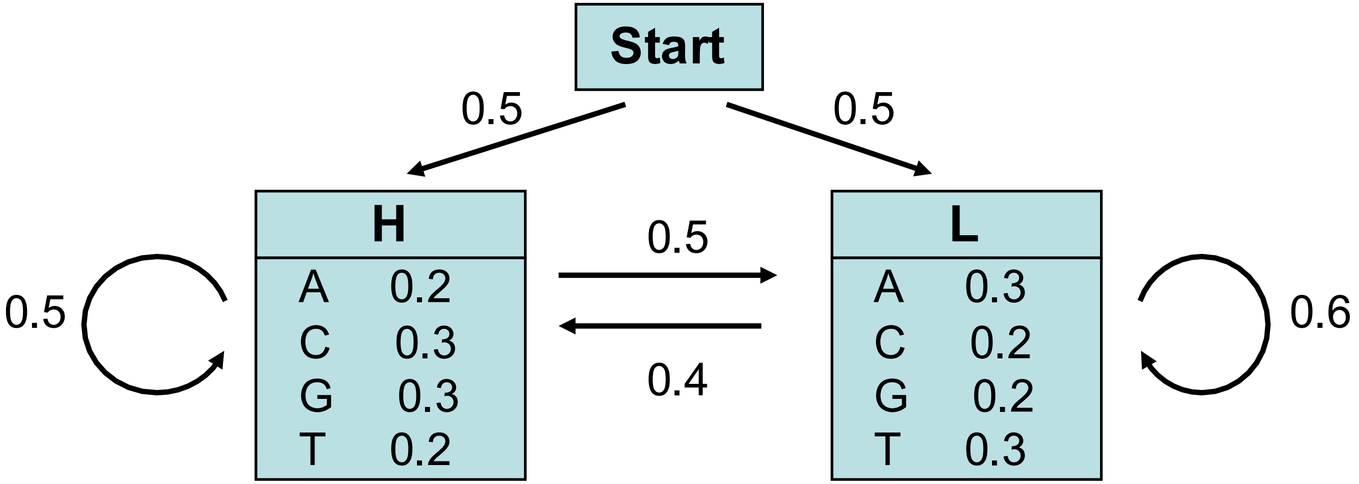
HMM : Forward algorithm - a toy example



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			

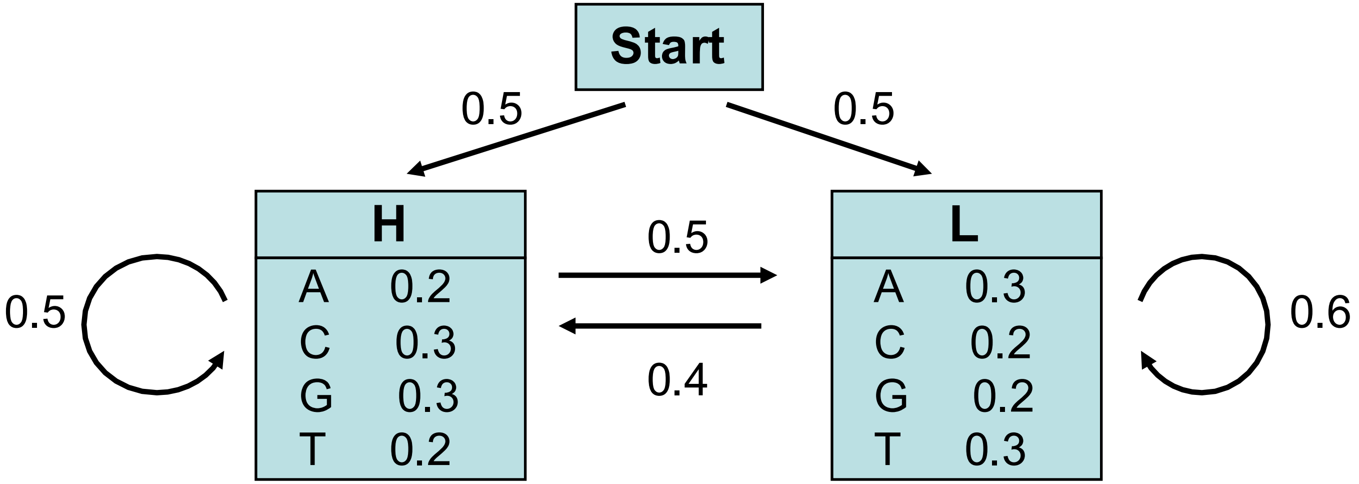
HMM : Forward algorithm - a toy example



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		

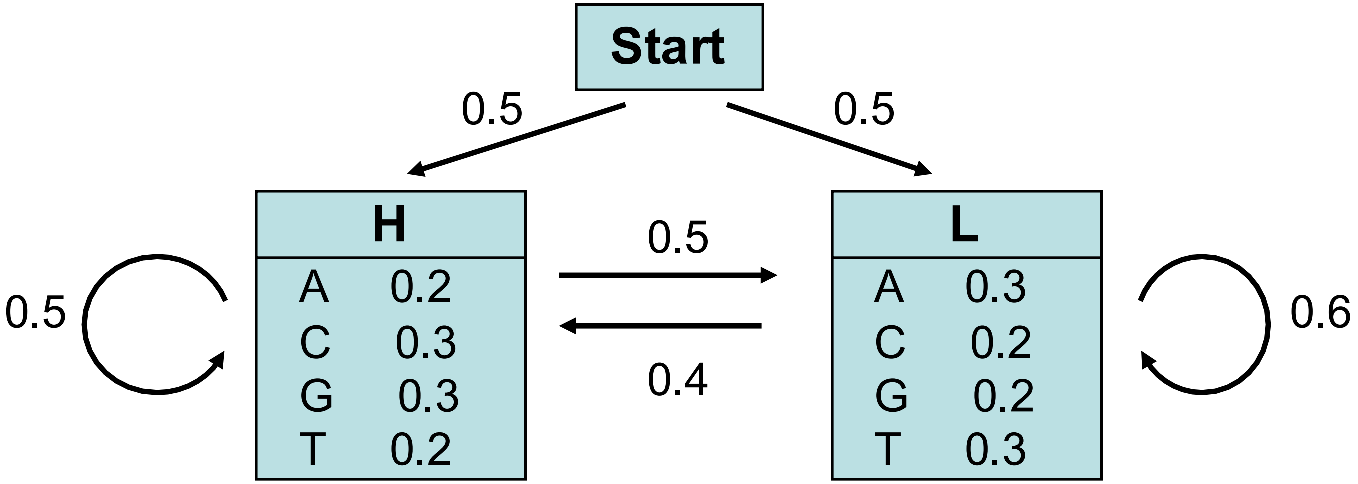
HMM : Forward algorithm - a toy example



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	... + ...	

HMM : Forward algorithm - a toy example



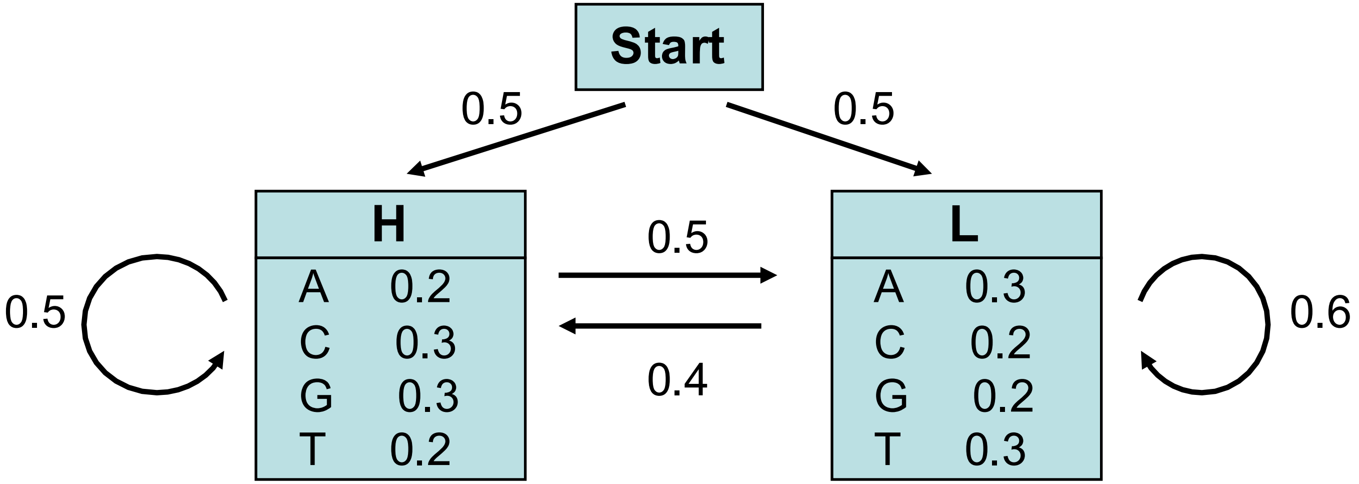
Consider now the sequence S= **GGCA**

	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

Σ = 0.0038432

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

HMM : Forward algorithm - a toy example



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.

HMM : Summary

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** (or the **Backward 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). A HMM motif can represent a TF binding site for ex.

HMM : 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)

HMMER

The HUMMER3 package contains a set of programs (developed by S. Eddy) to build HMM models (from a set of aligned sequences) and to use HMM models (to align sequences or to find sequences in databases). These programs are available at the Mobyle plateform (http://mobyle.pasteur.fr/cgi-bin/MobylePortal/portal.py)