Hidden Markov Models for biological sequence analysis I

Master in Bioinformatics UPF
2014-2015

Eduardo Eyras
Computational Genomics
Pompeu Fabra University - ICREA
Barcelona, Spain
Wherever there is a CG (CpG) in the genome, C use to change chemically by methylation, which is then likely to mutate into a T. Thus CpG dinucleotides are less frequent than expected (less than 1/16)

This methylation is sometimes suppressed in some regions (e.g. promoter regions of genes), hence they contain a high content of CpG. These regions are called CpG islands

CpG islands are of variable length, between hundreds to thousands of base pairs.

We would like to answer the following questions:

• given a DNA sequence, Is it part of a CpG island?
• Given a large DNA region, can we find CpG islands in it?
Example: CpG Islands

\[ a^+_{st} \] Transition probability between two adjacent positions in CpG islands

\[ a^-_{st} \] Transition probability between two adjacent pos. outside CpG islands

Given a sequence S the log-likelihood ratio is:

\[ \sigma = \log \frac{P(S|+)}{P(S|-)} = \sum_{i=1}^{N} \log \frac{a^+_{i-1,i}}{a^-_{i-1,i}} \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0.180</td>
<td>0.274</td>
<td>0.426</td>
<td>0.120</td>
<td>-</td>
<td>0.300</td>
<td>0.205</td>
<td>0.285</td>
<td>0.210</td>
</tr>
<tr>
<td>A</td>
<td>0.171</td>
<td>0.368</td>
<td>0.274</td>
<td>0.188</td>
<td>C</td>
<td>0.322</td>
<td>0.298</td>
<td>0.078</td>
<td>0.302</td>
</tr>
<tr>
<td>C</td>
<td>0.161</td>
<td>0.339</td>
<td>0.375</td>
<td>0.125</td>
<td>G</td>
<td>0.248</td>
<td>0.246</td>
<td>0.298</td>
<td>0.208</td>
</tr>
<tr>
<td>G</td>
<td>0.079</td>
<td>0.355</td>
<td>0.384</td>
<td>0.182</td>
<td>T</td>
<td>0.177</td>
<td>0.239</td>
<td>0.292</td>
<td>0.292</td>
</tr>
</tbody>
</table>

The larger the value of sigma, the more likely is to be a CpG island
Example: CpG Islands

Given a large stretch of DNA of length \( L \), we extract windows of \( l \) nucleotides:

\[
S^{(k)} = (s_{k+1}, ..., s_{k+l}) \quad \text{for} \quad 1 \leq k \leq L - l \quad \text{and} \quad l << L
\]

For each window we calculate \( \sigma(S^{(k)}) \)

\[
\sigma = \log \frac{P(S^+)}{P(S^-)} = \sum_{i=1}^{N} \log \frac{a_{i-1,i}^+}{a_{i-1,i}^-}
\]

Windows with \( \sigma(S^{(k)}) > 0 \) (or above a set threshold) are possible CpG islands

**Limitation:** the Markov chain model cannot determine the lengths of the CpG islands
Example: CpG Islands

Can we build a single model for the entire sequence that incorporates both Markov chains?
Include the probability to switch from one model to the other

For every symbol, in addition to the transitions to symbols of the same type (state), we also consider the transitions to symbols of the other type (state).

But this approach will require the estimation of many probabilities.
Example: CpG Islands

Simplify by defining the probability to switch from one model to the other

We summarize the transitions between + and – symbols into single transition probabilities

The change between states “+” and “-” determines the limits of the CpG island
Example: CpG Islands

We have transition probabilities between states: \( a_{++}, a_{--}, a_{+-}, a_{-+} \)

And emission probabilities in each state: \( e_+(a), e_+(ab), e_-(a), e_-(ab) \)
In fact, we can redraw the CpG island model in the following way:

We have transitions between states and each state can generate symbols.
Example: The coin tossing problem

A simpler version of the CpG island problem

Game: to flip coins, which results in two possible outcomes: Head or Tail

Two coins:

Fair coin (N): Head (a) or Tail (z) with same probability = 1/2

\[
P(a \mid N) = e_N(a) = 0.5 \\
P(z \mid N) = e_N(z) = 0.5
\]

Loaded coin (T): Head (a) with probability 3/4 (tail (z) with probability 1/4)

\[
P(a \mid T) = e_T(a) = 0.75 \\
P(z \mid T) = e_T(z) = 0.25
\]
The coin tossing problem

The crooked dealer changes between Fair and loaded coins 10% of the time (90% of the time uses the same coin)

We can define some sort of probability transition between coin “states”:

\[
P(N \mid N) = a_{NN} = 0.9
\]
\[
P(T \mid T) = a_{TT} = 0.9
\]
\[
P(T \mid N) = a_{NT} = 0.1
\]
\[
P(N \mid T) = a_{TN} = 0.1
\]
The coin tossing problem

We can represent this model graphically

We have two states: fair and loaded

Each state has emission probabilities (head or tail)

There are transition probabilities between states (changing the coin)
HMM definition

A Hidden Markov Model (HMM) is a triplet: \( M = (\Sigma, Q, \Theta) \)

- \( \Sigma \) is an alphabet of symbols
- \( Q \) is a set of states that can emit symbols from the alphabet
- \( \Theta \) is a set of probabilities
  - \( a_{kl} \) transition probabilities between states \( k, l \in Q \)
  - \( e_k(b) \) emission probabilities \( k \in Q, b \in \Sigma \)
Hidden Markov Model with two states: fair (N) and loaded (T)

Symbols: head and tail

There are transition probabilities between states and emission probabilities in each state
HMM definition

A path in the model $M$ is a sequence of states $\Pi = (\pi_1, \ldots, \pi_L)$

The path of states is unknown (hidden)

We have added two states initial and final $\pi_0 \pi_{N+1}$

Given a sequence $S = s_1 s_2 s_3 \ldots s_N$ of symbols

The transition probabilities between states ($k$ and $l$), and the emission probabilities (of character $b$ at state $k$) are written as:

$$a_{kl} = P(\pi_i = l \mid \pi_{i-1} = k)$$
$$e_k(b) = P(s_i = b \mid \pi_i = k)$$

The probability that sequence $S$ is generated by model $M$ given the path

$$P(S \mid \Pi) = P(s_1 \ldots s_N \mid \pi_1 \ldots \pi_N)$$

$$= P(\pi_1 \mid \pi_0)P(s_1 \mid \pi_1) \cdot P(\pi_2 \mid \pi_1)P(s_2 \mid \pi_1) \cdot \ldots \cdot P(\pi_N \mid \pi_{N-1})P(s_N \mid \pi_N) \cdot P(\pi_{N+1} \mid \pi_N)$$

$$= a_{\pi_0, \pi_1} e_{\pi_1}(s_1) \cdot a_{\pi_1, \pi_2} e_{\pi_2}(s_2) \cdot \ldots \cdot a_{\pi_{N-1}, \pi_N} e_{\pi_N}(s_N) = a_{\pi_0, \pi_1} \prod_{i=1}^{N} a_{\pi_i, \pi_{i+1}} e_{\pi_i}(s_i)$$
HMM example

$a_{13}$  Probability of transition from state 1 to state 3

e_2(A)  Probability of emitting character A in state 2

begin/end states are silent states on the path: do not emit symbols
A path \( \pi = \pi_1 \ldots \pi_n \) in the HMM is defined as sequence of states

The probability that sequence \( x \) was generated by the path \( \pi \):

\[
P(s | \pi) = P(\pi_1 | \pi_0) \prod_{i=1}^{n} P(s_i | \pi_i) P(\pi_{i+1} | \pi_i)
\]

\[
= a_{\pi_0 \pi_1} \prod_{i=1}^{n} e_{\pi_i} (s_i) a_{\pi_i \pi_{i+1}}
\]

\[
= \prod_{i=0}^{n} e_{\pi_i} (s_i) a_{\pi_i \pi_{i+1}}
\]

- Probability of transition between states \( \pi_i \) and \( \pi_{i+1} \)
- Probability that \( s_i \) was emitted from state \( \pi_i \)
- No emission at the begin state
Why Hidden?

In Markov chains, each state accounts for each part of the sequence.

In HMMs we distinguish between the **observable** parts of the problem (emitted characters) and the **hidden** parts (states):

In a Hidden Markov model, there are multiple states that could account for the same part of the sequence: that actual state is hidden. That is, there is no one-to-one correspondence between states and observations.

E.g. The toss of a coin is an observable, the coin being used is the state, which we cannot observe.

E.g. given a biological sequence, interesting features are hidden (genes, splice-sites, etc...)

We will have to calculate the probability of being in a particular state.
HMMs are memory-less

At each time (or position) step, the only thing that affects future (or downstream) states and emissions is the current state

\[ P(\pi_i = l \mid "\text{what ever happened so far}"
\]
\[ = P(\pi_i = l \mid \pi_1, \pi_2, ..., \pi_{i-1}, s_1, s_2, ..., s_{i-1}) = P(\pi_i = l \mid \pi_{i-1} = k) = a_{kl} \]

At each time/position, the emission of a character only depends on the current state:

\[ P(s_i = b \mid "\text{what ever happened so far}"
\]
\[ = P(s_i = b \mid \pi_1, \pi_2, ..., \pi_i, s_1, s_2, ..., s_{i-1}) = P(s_i = b \mid \pi_i = k) = e_k(b) \]
HMMs can be seen as generative models

We can generate a path of states (the orange path).

This orange path in the HMM generates a sequence:

\[ \Pi = 0113 \]

\[ S = \text{AAC} \]

There is a probability associated to this sequence, given by the transition and emission probabilities:

\[
P(S \mid \Pi) = a_{\pi_0, \pi_1} \cdot \prod_{i=1}^{L} e_{\pi_i}(s_i) \cdot a_{\pi_i, \pi_{i+1}}
\]

\[
P(\text{AAC} \mid \Pi) = a_{01} \times e_1(A) \times a_{11} \times e_1(A) \times a_{13} \times e_3(C) \times a_{35}
\]

\[
= 0.5 \times 0.4 \times 0.2 \times 0.4 \times 0.8 \times 0.3 \times 0.6
\]
This is not necessarily the most probable sequence of states.

This is also not the total probability of the sequence of symbols, since other combination of states can give the same sequence of symbols.
HMM questions

The LEARNING problem

Goal: find the transition and emission probabilities of the model: learn the model

The EVALUATION problem

what is the probability that this sequence has been generated by the crooked dealer?

The DECODING problem

We must find the underlying sequence of states that led to the observed sequence of results: we must label the results by its state

The POSTERIOR DECODING problem

Goal: find the probability that the dealer was using a biased coin at a particular time
HMM questions (algorithms)

The LEARNING problem
What are the parameters of the model?
Maximum likelihood estimation, Baum-Welch, EM

The EVALUATION problem
How likely is a given sequence
The Forward algorithm

The DECODING problem
What is the most probable path for generating a given sequence?
The Viterbi algorithm

The POSTERIOR DECODING problem
Which one is the most likely state in a given position?
The Forward-Backward algorithm
HMM questions (algorithms)

The LEARNING problem
What are the parameters of the model?
Maximum likelihood estimation, Baum-Welch, EM

The EVALUATION problem
How likely is a given sequence
The Forward algorithm

The DECODING problem
What is the most probable path for generating a given sequence?
The Viterbi algorithm

The POSTERIOR DECODING
Which one is the most likely state in a given position?
The Forward-Backward algorithm
Given a series of labelled sequences

Goal: find the transition and emission probabilities of the model: learn the model

HMM questions
The LEARNING problem

loaded

fair

loaded

fair

Goal: find the transition and emission probabilities of the model: learn the model

\[
a_{NN} = 0.9
\]

\[
e_M(a) = 0.5
\]

\[
e_M(z) = 0.5
\]

\[
a_{NT} = 0.1
\]

\[
e_M(a) = 0.75
\]

\[
e_M(z) = 0.25
\]

\[
a_{TN} = 0.1
\]

\[
a_{TT} = 0.9
\]
HMM Parameter estimation

The simplest way to obtain the parameters of the model is from a set of labelled observations. E.g. a number of sequences for which we know the actual sequence of states.

\[ A_{kl} \quad \text{Count the number of transitions between states } k \text{ and } l \]

\[ E_k(b) \quad \text{Count the number of times the symbol } b \text{ is emitted by state } k \]

We can estimate the probabilities as follows:

\[ a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}, \quad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')} \]
HMM Parameter estimation

This estimation is called “Maximum likelihood”, since the probability of all sequences given the parameters is maximal:

\[ P(s^1 \ldots s^N \mid \theta) = \prod_{j=1}^{N} P(s^j \mid \theta) \]

Is maximal for parameters estimated from frequencies and assuming that the sequences are independent

To avoid overfitting, use pseudocounts:

\[ A_{kl} \rightarrow A_{kl} + r_{kl} \]
\[ E_k(b) \rightarrow E_k(b) + r_k(b) \]

Pseudocounts reflect our prior knowledge
The DECODING problem

What is the most probable path for generating a given sequence?
The Viterbi algorithm

The POSTERIOR DECODING

Which one is the most likely state in a given position?
The Forward-Backward algorithm
Given a sequence of heads (H) and tails (T)

HTTHHHHHHHTTTTHTHTHTHTTHTHTHHTTHHHTHHHHHTTTTHTHTHHTT

Is it possible to reproduce the coin types the dealer has used?

Goal: to determine which coin produced each outcome

We must find the underlying sequence of states that led to the observed sequence of results: we must label the results by its state. For instance:

Observations States
HHTTHHHHHHTTTTHTHTHTHTTHTHTHHTTHHHTHHHHHTTTTHTHTHHTT
LLLLLLLLLLLLFFFFFFFFFFFLLLLLLLLLLLLFFFFFFFFFFF
loaded fair loaded fair

The label assignment has associated a probability
Finding the most probable path: Viterbi algorithm (the decoding problem)

We want to find the optimal path $\Pi^*$ that maximizes the probability of the observed sequence $P(S \mid \Pi^*)$ is maximal.

We can express this as follows:

$$\Pi^* = \arg \max_{\Pi} \{ P(S \mid \Pi) \}$$

We solve this problem using Dynamic programming:

The most likely parse (path through the model) up to character $s_i$ ending in a specific state $k$ is part of the optimal parse for the whole sequence $S$.
Finding the most probable path: Viterbi algorithm (the decoding problem)

We can define a dynamic programming table of size $|\Pi| \times L$, where $|\Pi|$ is the number of states and $L$ the length of the sequence.

Each element $v_k(i)$, $k=1,\ldots,|\Pi|$, $i=1,\ldots,L$ represents the probability of the most probable path for the prefix $s_1s_2\ldots s_i$ and ending in state $k$.

$$v_k(i) = \max_{\pi_1,\ldots,\pi_{i-1}} P(s_1\ldots s_{i-1},\pi_1\ldots \pi_{i-1}, s_i, \pi_i = k)$$

We want to calculate $v_{\text{end}}(L)$ the probability of the most probable path accounting for all the sequence and ending in the end state.
Finding the most probable path: Viterbi algorithm
(the decoding problem)

We can define this probability recursively

\[ v_k(i + 1) = \max_{\{\pi_1, \ldots, \pi_i\}} P(s_1 \ldots s_i, \pi_1, \ldots, \pi_i, s_{i+1}, \pi_{i+1} = k) \]

\[ = \max_{\{\pi_1, \ldots, \pi_i\}} P(s_{i+1}, \pi_{i+1} = k \mid s_1 \ldots s_i, \pi_1 \ldots \pi_i) P(s_1 \ldots s_i, \pi_1 \ldots \pi_i) \]

\[ = \max_{\{\pi_1, \ldots, \pi_i\}} \left[ P(s_{i+1}, \pi_{i+1} = k \mid \pi_i) P(s_1 \ldots s_{i-1}, \pi_1 \ldots \pi_{i-1}, s_i, \pi_i) \right] \]

\[ = \max_l \left[ P(s_{i+1}, \pi_{i+1} = k \mid \pi_i = l) \max_{\{\pi_1, \ldots, \pi_i\}} P(s_1 \ldots s_{i-1}, \pi_1 \ldots \pi_{i-1}, s_i, \pi_i = l) \right] \]

\[ = \max_l \left[ P(s_{i+1} \mid \pi_{i+1} = k) P(\pi_{i+1} = k \mid \pi_i = l) v_l(i) \right] \]

\[ = e_k(s_{i+1}) \max_l \left[ v_l(i) a_{lk} \right] \]
Finding the most probable path: Viterbi algorithm (the decoding problem)

1.- Initialization

\[ v_{\text{begin}}(0) = 1 \]
\[ v_k(0) = 0, \ \forall k \neq \text{begin} \]

2.- Recursion: for positions \( i = 0, \ldots, L - 1 \) and states \( l \in Q \)

\[ v_l(i + 1) = e_l(s_{i+1}) \cdot \max_{k \in Q} \{ v_k(i) \cdot a_{kl} \} \]

3.- Termination

\[ P(S \mid \Pi^*) = \max_{k \in Q} \{ v_k(L) \cdot a_{k,\text{end}} \} \]
Finding the most probable path: Viterbi algorithm (the decoding problem)

(keep the pointers for back-tracking of states)

1.- Initialization

\[ v_{\text{begin}}(0) = 1 \]
\[ v_k(0) = 0, \ \forall k \neq \text{begin} \]

2.- Recursion: for positions \( i = 0, \ldots, L - 1 \) and states \( l \in Q \)

\[ v_l(i + 1) = e_l(s_{i+1}) \cdot \max_{k \in Q} \{ v_k(i) \cdot a_{kl} \} \]
\[ \text{ptr}_{i+1}(l) = \arg \max_{k \in Q} \{ v_k(i) \cdot a_{kl} \} \]

3.- Termination

\[ P(S \mid \Pi^*) = \max_{k \in Q} \{ v_k(L) \cdot a_{k,\text{end}} \} \]
\[ \pi_L^* = \arg \max_{k \in Q} \{ v_k(L) \cdot a_{k,\text{end}} \} \]

Find optimal path following pointers back, starting at \( \pi_L \) using the recursion:

\[ \pi_i^* = \text{ptr}(\pi_{i+1}^*) \]
Finding the most probable path: Viterbi algorithm
(the decoding problem)

Can use logarithms to avoid underflow problems:

1.- Initialization

\[ v_{\text{begin}}(0) = 1 \]
\[ v_k(0) = 0, \ \forall \ k \neq \text{begin} \]
\[ v_{\text{begin}}(0) = 0 \]
\[ v_k(0) = -\infty, \ \forall \ k \neq \text{begin} \]

2.- Recursion: for positions \( i = 0, \ldots, L - 1 \) and states \( l \in Q \)

\[ v_l(i+1) = e_l(s_{i+1}) \cdot \max_{k \in Q} \{ v_k(i) \cdot a_{kl} \} \]
\[ v_l(i+1) = \log(e_l(s_{i+1})) + \max_{k \in Q} \{ v_k(i) + \log(a_{kl}) \} \]
\[ \text{ptr}_{i+1}(l) = \arg \max_{k \in Q} \{ v_k(i) \cdot a_{kl} \} \]
\[ \text{ptr}_{i+1}(l) = \arg \max_{k \in Q} \{ v_k(i) + \log(a_{kl}) \} \]

3.- Termination

\[ P(S \mid \Pi^*) = \max_{k \in Q} \{ v_k(L) \cdot a_{k,\text{end}} \} \]
\[ PL(S, \Pi^*) = \max_{k \in Q} \{ v_k(L) + \log(a_{k,\text{end}}) \} \]
\[ \pi_L^* = \arg \max_{k \in Q} \{ v_k(L) \cdot a_{k,\text{end}} \} \]
\[ \pi_L^* = \arg \max_{k \in Q} \{ v_k(L) + \log(a_{k,\text{end}}) \} \]
\[ \pi_i^* = \text{ptr}(\pi_{i+1}^*) \]

Find optimal path using the same recursion
HMM questions

The LEARNING problem

- What are the parameters of the model?
  - Maximum likelihood estimation, Baum-Welch, EM

The EVALUATION problem

- How likely is a given sequence
  - The Forward algorithm

The DECODING problem

- What is the most probable path for generating a given sequence?
  - The Viterbi algorithm

The POSTERIOR DECODING

- Which one is the most likely state in a given position?
  - The Forward-Backward algorithm
HMM questions

The EVALUATION problem

Given a sequence of heads (H) and tails (T)

HTTHHHHHHHTTTHTHHTTTTHTHTTHHTHHTHHHHHHHTTTTHTHHTHT

Assuming that we have modelled the coins correctly,

what is the probability that this sequence has been generated by the crooked dealer?

HTTHHHHHHHTTTHTHHTTTTHTHTTHHTHHTHHHHHHHTTTTHTHHTHT

e.g. P(S)= 1.3x10^{-35}
How likely is a sequence?

But that is the probability of the sequence, given a specific path.

We want the probability of the sequence over all paths, i.e.:

\[ P(S) = P(s_1 \ldots s_L) = \sum_{\pi} P(s_1 \ldots s_L, \pi_0 \ldots \pi_N) \]

But the number of possible paths can grow exponentially:
there are \( N^L \) possible paths for the case of \( N \) states and a sequence of length \( L \).

The Forward algorithm allows to calculate this probability efficiently.

This algorithm will allow us to calculate also the probability of
being in a specific state \( k \) in a position \( i \) of the sequence.
The Forward algorithm

We define the probability of generating the first $i$ letters of $S$ ending up in state $k$ (forward probability):

$$f_k(i) = P(s_1...s_i,\pi_i = k)$$

$$f_k(i) = P(s_1...s_i,\pi_i = k)$$

$$= \sum_{\pi_1...\pi_{i-1}} P(s_1...s_{i-1},\pi_1...\pi_{i-1},\pi_i = k) e_k(s_i)$$

$$= \sum_l \sum_{\pi_1...\pi_{i-1}} P(s_1...s_{i-1},\pi_1...\pi_{i-2},\pi_{i-1} = l) a_{lk} e_k(s_i)$$

$$= \sum_l P(s_1...s_{i-1},\pi_{i-1} = l) a_{lk} e_k(s_i)$$

$$= e_k(s_i) \sum_l f_l(i-1) a_{lk}$$

We obtain a recursion formula for this probability: we can apply a dynamic programming approach.
The Forward algorithm

1.- Initialization

\[ f_{\text{begin}}(0) = 1 \]
\[ f_k(0) = 0, \quad \forall k \neq \text{begin} \]

2.- Recursion: for positions \( i = 0, \ldots, L - 1 \) and states \( l \in Q \)

\[ f_l(i+1) = e_l(s_{i+1}) \sum_{k \in Q} f_k(i) \cdot a_{kl} \]

3.- Termination

\[ P(S) = \sum_{k \in Q} f_k(L) \cdot a_{k,\text{end}} \]

Forward algorithm is very similar to Viterbi. The only difference is that here we do a sum instead of a maximisation.
The Forward algorithm: Example

S = TAGA
The Forward algorithm: Example

- given the sequence $x = \text{TAGA}$
- initialization
  
  $$f_0(0) = 1 \quad f_1(0) = 0 \quad \ldots \quad f_5(0) = 0$$
- computing other values
  
  $$f_1(1) = e_1(T) \times (f_0(0) \times a_{01} + f_1(0) a_{11}) = 0.3 \times (1 \times 0.5 + 0 \times 0.2) = 0.15$$
  
  $$f_2(1) = 0.4 \times (1 \times 0.5 + 0 \times 0.8)$$
  
  $$f_1(2) = e_1(A) \times (f_0(1) \times a_{01} + f_1(1) a_{11}) = 0.4 \times (0 \times 0.5 + 0.15 \times 0.2)$$
  
  $$\cdots$$

  $$\Pr(\text{TAGA}) = f_5(4) = (f_3(4) \times a_{35} + f_4(4) a_{45})$$
In some cases the algorithm can be more efficient by taking into account the minimum number of steps that must be taken to reach a state.

Eg. For this model, we do not need to initialize or calculate the values:

\[ f_3(0), f_4(0), f_5(0), f_5(1) \]
The DECODING problem

What is the most probable path for generating a given sequence?
The Viterbi algorithm

The POSTERIOR DECODING

Which one is the most likely state in a given position?
The Forward-Backward algorithm
Given a sequence of heads and tails

HTTHHHHHHHTTHTHHTHTTTHTHHTHHTHHTTHHHTHHHHHHTTTTHTHHTHT

Is it possible to catch him cheating?

Goal: find the probability that the dealer was using a biased coin at a particular time

P(Loaded)=80%
The posterior probability of a state 
(Backward algorithm)

We want to calculate the most probable state given an observation $s_i$

For instance, we want to catch the coin dealer cheating

For this, we calculate the probability distribution of the $i^{th}$ position: the probability that a given position in the sequence has an underlying state:

$$P(\pi_i = k \mid S)$$

We can compute the posterior probability in the following way:

$$P(\pi_i = k \mid S) = \frac{P(S, \pi_i = k)}{P(S)}$$
The posterior probability of a state
(Backward algorithm)

We can compute the posterior probability in the following way:

\[ P(\pi_i = k \mid S) = \frac{P(S, \pi_i = k)}{P(S)} \]

Consider the following join probability

\[ P(S, \pi_i = k) = \]
\[ P(s_1...s_i, \pi_i = k) P(s_{i+1}...s_L \mid s_1...s_i, \pi_i = k) = \]
\[ P(s_1...s_i, \pi_i = k) P(s_{i+1}...s_L \mid \pi_i = k) \]

Probability of generating the rest of the sequence \((s_{i+1}...s_L)\), starting from a given state \((k)\) and ending at the end state.
The posterior probability of a state
(Backward algorithm)

\[ P(\pi_i = k \mid S) = \frac{P(S, \pi_i = k)}{P(S)} \]

Given by the Forward and Backward algorithms

\[ b_k(i) = P(s_{i+1} \ldots s_L \mid \pi_i = k) \]

\[ = \sum_{\pi_{i+1} \ldots \pi_N} P(s_{i+1}s_{i+2} \ldots s_L, \pi_{i+1} \ldots \pi_N \mid \pi_i = k) \]

\[ = \sum_l \sum_{\pi_{i+1} \ldots \pi_N} P(s_{i+1}s_{i+2} \ldots s_L, \pi_{i+1} = l, \pi_{i+2} \ldots \pi_N \mid \pi_i = k) \]

\[ = \sum_l e_l(s_{i+1}) a_{kl} \sum_{\pi_{i+1} \ldots \pi_N} P(s_{i+2} \ldots s_L, \pi_{i+2} \ldots \pi_N \mid \pi_{i+1} = k) \]

\[ = \sum_l e_l(s_{i+1}) a_{kl} b_l(i + 1) \]

We can again apply dynamic programming to calculate the backward probabilities
The posterior probability of a state  
(Backward algorithm)

1.- Initialization  
\[ i = L \]

\[ b_k(L) = a_{k,end}, \ \forall k \in Q \]

2.- Recursion: for positions  
\[ i = L - 1, \ldots, 1 \]  
and states  \( l \in Q \)

\[ b_k(i) = \sum_{l \in Q} a_{kl} \cdot e_l(s_{i+1}) \cdot b_l(i+1) \]

3.- Termination

\[ P(S) = \sum_{l \in Q} a_{\text{begin},l} \cdot e_l(s_1) \cdot b_l(1) \]

Not usually necessary, as we normally use Forward to derive  \( P(S) \)
Posterior decoding

We can ask at each position, what is the most likely state that gave that character in the sequence. (I.e. the probability that a symbol is produced by a given state, given the full sequence)

E.g.: we want to know the probability for each G to be emitted by the state D (donor site):

$$P(\pi_i = k, S) = f_k(i)b_k(i)$$

$$P(\pi_i = k, S) = P(\pi_i = k \mid S)P(S)$$

The posterior probability:

$$P(\pi_i = k \mid S) = \frac{f_k(i)b_k(i)}{P(S)}$$
References

**Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids**
Cambridge University Press, 1999

**Problems and Solutions in Biological Sequence Analysis**
Mark Borodovsky, Svetlana Ekisheva
Cambridge University Press, 2006

**Bioinformatics and Molecular Evolution**
Paul G. Higgs and Teresa Attwood.

**An Introduction to Bioinformatics Algorithms**
(Computational Molecular Biology)
Exercise
Viterbi algorithm - Example

From a DNA sequence, we want to predict whether a region is likely to be exonic or intronic:

AGGGCTAGGGTCTTCTGGTGCAACTCAGCCCCTCTGTACCAGCCCTAGCTATGTATGTTCG
|--exon---|<------intron-------->|-exon-|<---intron--->|--exo--|

This identification of functional elements can be seen as a labelling of the observed bases in terms of a number of states (E: exon, I: intron)

AGGGCTAGGGTCTTCTGGTGCAACTCAGCCCCTCTGTACCAGCCCTAGCTATGTATGTTCG
EEEEEEEEEEEEEEEEEEEEEEEEEIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

Provided a series of transition probabilities between states and emission probabilities for symbols (nucleotides), we want to determine the most probable labeling of an input sequence (the most probable path/parse)
Viterbi algorithm - Example

Simple HMM for 5’ splice-site recognition (Eddy 2004)

This model describes sequences where the initial part is exonic (homogeneous composition), then there is a nucleotide that will form the splice-site (mostly G), and the rest of the sequence belongs to the intron (rich in A and T).

The problem consists in finding the 5´splice-site (donor) in an unknown DNA sequence.
Viterbi algorithm - Example

Given the path $\Pi = E \ E \ D \ I \ I$  
(E: exon, I: intron, D: donor)

The probability of observing the sequence: $S= C \ A \ G \ T \ A$

$$P(S|\Pi) = a_{BE} \times e_E(C) \times a_{EE} \times e_E(A) \times a_{ED} \times e_D(G) \times a_{DI} \times e_I(T) \times a_{II} \times e_I(A)$$

$$= 1.0 \times 0.25 \times 0.9 \times 0.25 \times 0.1 \times 0.95 \times 1.0 \times 0.4 \times 0.9 \times 0.4$$

$$= 0.0007695$$

For this other path of $E \ D \ I \ I \ I$

$C \ A \ G \ T \ A$

$$P(S|\Pi) = a_{BE} \times e_E(C) \times a_{ED} \times e_D(A) \times a_{DI} \times e_I(G) \times a_{II} \times e_I(T) \times a_{II} \times e_I(A)$$

$$= 1.0 \times 0.25 \times 0.1 \times 0.05 \times 1.0 \times 0.1 \times 0.9 \times 0.4 \times 0.9 \times 0.4$$

$$= 0.0001620$$

It is smaller
Viterbi algorithm - Example

The logs of the probabilities of the different sequence of states for the same DNA sequence

Sequence: C T T C A T G T G A A A G C A G A C G T A A G T C A
State path: E E E E E E E E E E E E E E E E E E E 5 I I I I I I I
Parsing: (sequence of states)

log P
-41.22
-43.90
-43.45
-43.94
-42.58
-41.71
Viterbi algorithm - Exercise

Fill in the Viterbi dynamic programming matrix and obtain the most probable path

<table>
<thead>
<tr>
<th>x</th>
<th>no nucleotide</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>G</th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin (0)</td>
<td>η₀(0)</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
<tr>
<td>donor (2)</td>
<td>η₂(0)</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-8.24</td>
<td>-12.69</td>
<td>-11.24</td>
<td>-∞</td>
<td>-17.19</td>
</tr>
<tr>
<td>intron (3)</td>
<td>η₃(0)</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-9.16</td>
<td>-11.57</td>
<td>-12.16</td>
<td>-13.19</td>
<td>-14.22</td>
</tr>
<tr>
<td>end (4)</td>
<td>η₄(0)</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
<td>-∞</td>
</tr>
</tbody>
</table>
Viterbi algorithm - Exercise

For instance, using the log-form of the algorithm:

\[ v_3(8) = v_I(T) \]
\[ = e_I(8) + \max_{k \in \{D,I\}} \{ v_k(7) + a_{kI} \} \]
\[ = e_I(8) + \max\{v_I(7) + a_{II}; v_D(7) + a_{DI} \} \]
\[ = -0.92 + \max\{\{-11.57 - 0.11; -11.24 + 0\} \]
\[ = -0.92 - 11.24 = -12.16 \]

The maximum is given by the transition from D to I:

\[ ptr_8(3) = ptr_8(I) = D \]

only need to use the possible transitions. Not allowed transitions can be given a 0 probability (-\(\infty\) in log-form)
What is a hidden Markov model?
Sean R. Eddy (2004)
Nature Biotechnology 22(10)1315