Markov Chains

Markov models and HMM are used to answer questions such as:

- Does this sequence belong to a particular family?
- Assuming the sequence does come from some family, what can we say about its internal structure? That is finding alpha helix, beta sheet, etc. in a protein sequence.

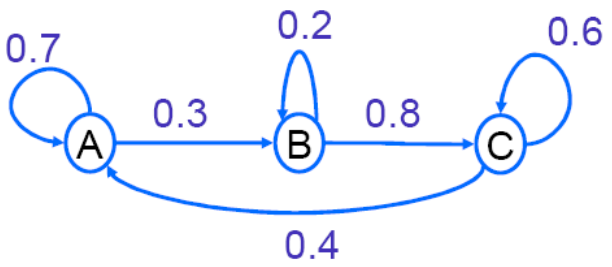
A markov model contains states and arrows. For example, in DNA each nucleotide can be considered a state. In a protein sequence, each amino acid can be considered a state. A probability parameter is associated with each arrow. So an arrow going from C to G represents the probability of having a G after a C. These probability parameters are called transition probabilities.

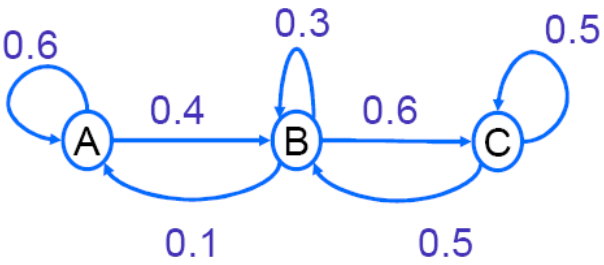
$$a_{st} = P(x_i = t \mid x_{i-1} = s) \qquad (1)$$

If you look a this equation closely, it says the probability of reaching a t given s, P(t|s). A key property of Markov chain is that the probability of each symbol Xi depends only on the preceding symbol Xi-1. Therefore, the following equation:

$$P(x_1) = \prod_{i=2}^{L} a_{x_{i-1}x_i} \qquad (2)$$

In other words, you multiply transition probabilities from i=2 to L.





Q = { A, B, C }

P = (1, 0, 0)

unique starting state A

$$T = \begin{pmatrix} .7 & .3 & 0 \\ 0 & .2 & .8 \\ .4 & 0 & .4 \end{pmatrix}$$

P(AABBCCC | M) =

A→A→B→B→C→C→C

$7 \cdot 3 \cdot 2 \cdot 8 \cdot 6 \cdot 6 \cdot 10^{-6} = 1.2\ 10^{-2}$

vs

$6 \cdot 4 \cdot 3 \cdot 6 \cdot 5 \cdot 5 \cdot 10^{-6} = 2.2\ 10^{-2}$

Modeling the beginning and end of sequences

Note that the previous equation requires us to begin with X1. This can be replaced by a begin state. We can also define an end state. Both these state are silent states. The begin states allows us to say that the probability of have an A in the first position is x.

Using Markov chains for discrimination

The primary use of equation 2 is to calculate the values for a likelihood ratio test. Lets look at this for CpG islands. We first need to extract the regions containing CpG islands. Then we derive two Markov chain models. The First, +, for regions containing CpG islands. The second, -, for regions not containing CpG islands. Let a^+_{st} denote the transition probability of $s, t \in$ □ inside a CpG island and let a□ st denote the transition probability outside a CpG island. We compute the logarithmic likelihood score by for a sequence X by:

$$Score(X) = \log \frac{P(X|\text{CpG island})}{P(X|\text{non CpG island})} = \sum_{i=1}^{L} \log \frac{a^+_{x_{i-1},x_i}}{a^-_{x_{i-1},x_i}}$$

The higher the score, the more likely it is that sequence X is a CpG island.

$$a^+_{st} = c^+_{st} / \sum_{t'} c^+_{st'} \qquad (3)$$

c refers to the number of times t is followed by s. These are maximum likelihood (ML) estimators for the transition probabilities. If the number of nucleotides were quite small, we would have used Bayesian estimation process.

In a table of F(i,j), i is the current nucleotide and j the following nucleotide. So the cell F(i,j) notes the probability of having a j after i. Using information in this table, we calculate the log-odds ratio. See equation of page 51. S(x) generated by the log-odds ratio shows a clear discrimination between regions labeled CpG islands and other regions.

Hidden Markov Models

The log-odds ratio window which we have created requires a window of a fixed size. A better model would be to build a single model for the entire sequence that incorporates both Markov chains. This means that we need twice as many states a before. A+, C+, G+, T+, A-, C-, G-, and T-.

The transition probabilities in this model are so that within each group they are close to the transition probabilities of the original component model, but there is also a small but finite chance of switching into the other component. Overall there is more chance of switching from '+' to '-' than vice versa, so if left to run free, the model will spend more of its time in the '-' non-island states than in the island states.

The essential difference between Markov chain and HMM is that for a HMM there is not a 1-1 correspondence between states and symbols. Therefore, by looking at C, you will not be able to tell whether it was emitted by state C+ or C-.

Formal definition of an HMM

A **hidden Markov model** (**HMM**) is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a *hidden* Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states.

Definition: A HMM is a triplet M = (S, Q, T) where:

- S is an alphabet of symbols
- Q is a finite set of states, capable of emitting symbols form the alphabet S
- T is a set of probabilities, comprised of:
    - *State transition probabilities*, denoted by *akl* for each $k, l \in Q$.
    - *Emission probabilities*, denoted by *ek*(*b*) for each $k \in Q$ and $b \in$ S.

We now need to distinguish the sequence of states from the sequence of symbols. The path pi is a sequence of states. The path itself follows a simple Markov chain, so the probability of a state depends only on the previous state. As in the markov chain model, we can defne the state transition probabilities in terms of pi:

$$a_{kl} = P(\pi_i = l \mid \pi_{i-1} = k) \qquad (4)$$

Since there is no longer a 1-1 correspondence between states and symbols, we must introduce a new set of parameters for the model:

$$e_k(b) = P(\pi_i = b \mid \pi_{i-1} = k) \qquad (5)$$

ek(b) is the probability that the symbol b is seen in state k.

$$P(X, \Pi) = a_{\pi_0,\pi_1} \cdot \prod_{i=1}^{L} e_{\pi_i}(x_i) \cdot a_{\pi_i,\pi_{i+1}}$$

Where for our convenience we denote π_0 = *begin* and π_{L+1} = *end*.

In practice, this is not very useful since very often we do not know the path. So we have to estimate the path either by finding the most likely path or using an a posteriori distribution over states.

Using HMM

There are 3 canonical problems associated with HMMs:

- Given the parameters of the model, compute the probability of a particular output sequence. This problem is solved by the forward-backward procedure.
- Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence. This problem is solved by the Viterbi algorithm.
- Given an output sequence or a set of such sequences, find the most likely set of state transition and output probabilities. In other words, train the parameters of the HMM given a dataset of sequences. This problem is solved by the Baum-Welch algorithm.

Most probable state path: the Viterbi algorithm

Although it is no longer possible to tell what state the system is in by looking at the corresponding symbol, it is often the sequence of underlying states that we are interested in. **Decoding** is the act of finding out the meaning of a sequence by considering the underlying states. A commonly used method is a dynamic programming algorithm, Viterbi.

In general, many different states can give rise to any particular sequence of symbols. For example the following 3 states give result in CGCG sequence of symbols.

1. C+, G+, C+, G+
2. C-, G-, C-, G-
3. C+, G-, C+. G-

The probability of have the first is larger than the second which is larger than the third. So if we are to choose only one path, it is likely that the first one will be chosen.

We can calculate the most probable path in a hidden Markov model using a dynamic programming algorithm.

$$\pi^* = \arg\max_{\pi} P(x,\pi)$$

X is a sequence of length L. Vk(i) is the probability of the most probable path that ends in state k.

$$v_k(i) = \max_{\{\Pi|\Pi_i=k\}} P(x_1, \ldots, x_i, \Pi)$$

1. Initialize:

$$\begin{aligned} v_{begin}(0) &= 1 \\ \forall_{k \neq begin} \quad v_k(0) &= 0 \end{aligned}$$

2. For each $i = 0, \ldots, L-1$ and for each $l \in Q$ recursively calculate:

$$v_l(i+1) = e_l(x_{i+1}) \cdot \max_{k \in Q} \{v_k(i) \cdot a_{kl}\}$$

3. Finally, the value of $P(X, \Pi^*)$ is:

$$P(X, \Pi^*) = \max_{k \in Q} \{v_k(L) \cdot a_{k,end}\}$$

We can reconstruct the path pi* itself by keeping back pointers during the recursive stage and tracing them.

When this algorithm is applied to a longer sequence, the derived optimal path pi* will switch between '+' and '-' components of the model, thereby giving the precise boundaries of the predicted CpG island regions.

The forward & backward algorithm

We calculated the probability of a sequence, P(x), with equation 2. Now we want to be able to calculate the probability for an HMM. Because many different state paths can give rise to the same sequence x, we must add the probabilities for all possible paths to obtain the full probability of x.

$$P(x) = \sum_{\pi} P(x, \pi) \qquad (9)$$

The number of possible paths pi increases exponentially with the length of the sequence, so brute force evaluation with this equation is not possible. One possible solution is to replace P(x) by pi*, the most probable path. An even better technique is to replace the maximization steps of Viterbi algorithm with sum, in other words use the forward algorithm.

Given a sequence X = (x1…xL) let us denote by fk(i) the probability of emitting the prefix (x1…xL) and eventually reaching state pii = k:

$$f_k(i) = P(x_1,...,x_i,\pi_i = k) \quad (10)$$

We terminate by:

$$P(X) = \sum_{k \in Q} f_k(L) \cdot a_{k,end}$$

which is essentially equation 9

In a complementary manner we denote by bk(i) the probability of the suffix (xi+1,…,xL) given pii = k.

To avoid underflow problem, Viterbi, forward, and backward algorithms are implemented in log space.

Posterior decoding and Viterbi algorithms are useful when many different paths have almost the same probability as the most probable one.