CS262 Computational Genomics
Lecture 8
Pair HMMs for Sequence Alignment

Monday 02/04/2008

Scribed by: Chung Ng

This lecture was given by Prof. Tom Do (who substituted for Prof. Batzoglou) and was mainly taught on chalk board. He showed the class how to use HMMs to do the sequence alignment. The model used for sequence alignment is called Pair HMM. Pair HMM is a probabilistic model that represents the probability distribution over all possible alignments.

Three major topics covered in this lecture:

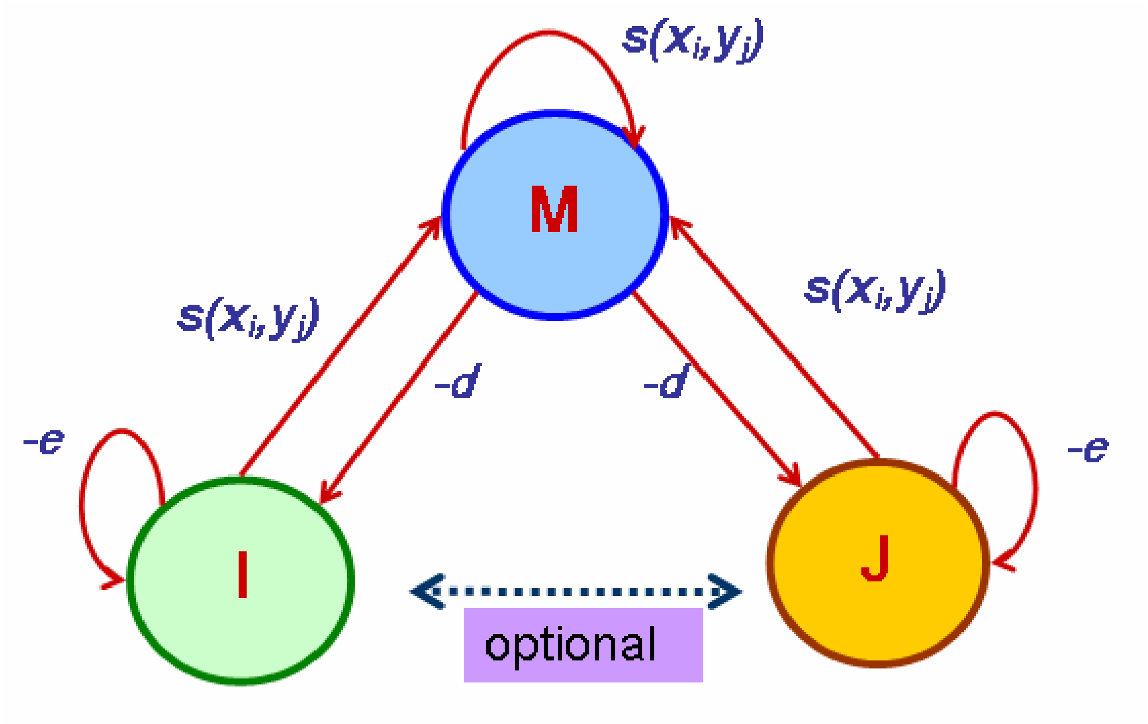
1. Representation
2. Inference
3. Learning

and a brief discussion on

4. EM algorithm

1. Representation of Pair HMMs

Recall that the standard Finite State Automaton (FSA) model for sequence alignment with affine gap penalty is follows:

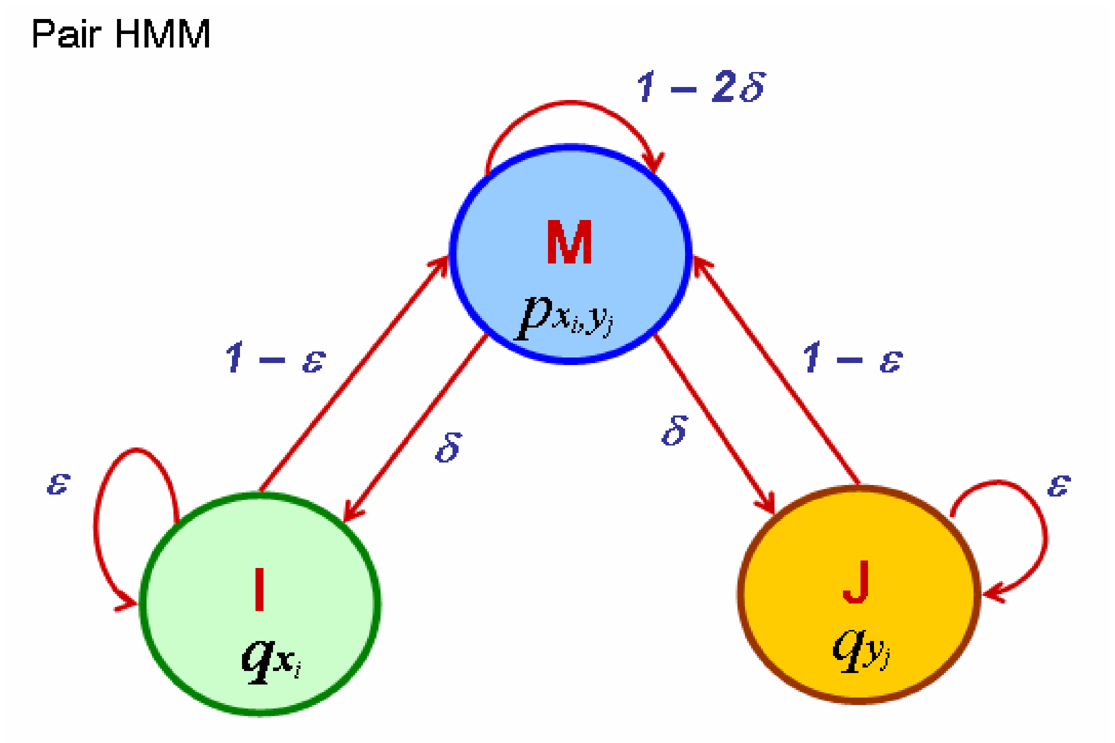


FSA model consists of the following parameters:

- Three states: *M, I, J*
  - One match/mismatch state *M* which represents matching a pair of letters or amino acids
  - Two insert states *I* and *J* which represents inserting a gap in either the first or second sequence
- Substitution matrix $s(x_i, y_i)$
- Opening gap penalty *d*
- Gap extension penalty *e*

The transition between state *I* and *J* is optional and usually is not considered.

This FSA model can be easily transformed to a HMM. The idea is to endow each state *M*, *I* and *J* with a probability distribution of emitting a column of a pairwise alignment, and replace the weights on the transitions between states by transition probabilities. This model is called Pair HMM.



Pair HMM consists of the following parameters:

- Three states: *M, I, J*

  State *M* matches one letter from each sequence

  State *I* inserts a gap to the second sequence

  State *J* inserts a gap to the first sequence

- Emission probabilities: $p_{x_i y_j}$, q_{x_i} and q_{y_j} where

  $p_{x_i y_j}$ = probability of emitting a pair of characters $\begin{bmatrix} x_i \\ y_j \end{bmatrix}$

  q_{x_i} = probability of emitting a pair of characters $\begin{bmatrix} x_i \\ - \end{bmatrix}$

  q_{y_j} = probability of emitting a pair of characters $\begin{bmatrix} - \\ y_j \end{bmatrix}$

- Transition probabilities: δ and ε where

  δ = probability of opening a gap and is set so that average length of conserved block = $\frac{1}{2\delta}$

  ε = probability of extending a gap and is set so that mean length of a gap = $\frac{1}{1-\varepsilon}$

  The transition probabilities for all transitions leaving each state sum to one.

Note that a full model would also contain a start state and a final state. From the start state, there is initial transition probability δ from the start state to state *I* and *J*, and $1-2\delta$ to *M*. In addition, in order to terminate, there is transition probability τ from state *M, I* and *J* to the final state and from final state back to itself. In the case that we have to take into account of τ, the transition probabilities between states might have to adjust accordingly. The full model is described in Durbin book. However, the full model is not used in the class discussion and the following sections.

Similarly to FSA model, each alignment corresponds to a path of states. In Pair HMM, each alignment corresponds to a path in the HMM states emitting the pair of letters in the alignment. A probability is calculated for each step of generating each possible alignment. The total probability, which is the product of probabilities in each step, is kept for each alignment. Therefore this model gives a probability distribution over all possible alignment of any two sequences.

2. Inference

Let $P(x, y, \pi) =$ probability of observing a particular pair of sequences *x* and *y* along with its pairwise alignment π.

Given a Pair HMM and two sequences *x* and *y*, find the most likely sequence alignment out of all possible alignments. In other words, we want to find an alignment π^* such that the term "Viterbi Inference"

$$\arg\max_{\pi} P(x, y, \pi)$$

is maximum.

Algorithm: Viterbi algorithm for Pair HMM

This optimization problem can be solved by dynamic programming similar to Viterbi algorithm for HMM. First we define three matrices:

$V^M(i, j) = P(x_1..x_i, y_1..y_j, \pi_1..\pi_t = M)$

= probability of a path from start state aligning $x_1..x_i$ and $y_1..y_j$ ending in match state M

$V^I(i, j) = P(x_1..x_i, y_1..y_j, \pi_1..\pi_t = I)$

= probability of a path from start state aligning $x_1..x_i$ and $y_1..y_j$ ending in gap state I

$V^J(i, j) = P(x_1..x_i, y_1..y_j, \pi_1..\pi_t = J)$

= probability of a path from start state aligning $x_1..x_i$ and $y_1..y_j$ ending in gap state J

Then we define the following recurrence relations:

$$V^M(i, j) = \max \begin{cases} V^M(i-1, j-1) \cdot (1-2\delta) \cdot p_{x_i y_j} \\ V^I(i-1, j-1) \cdot (1-\varepsilon) \cdot p_{x_i y_j} \\ V^J(i-1, j-1) \cdot (1-\varepsilon) \cdot p_{x_i y_j} \end{cases}$$

$$V^I(i, j) = \max \begin{cases} V^M(i-1, j) \cdot \delta \cdot q_{x_i} \\ V^I(i-1, j) \cdot \varepsilon \cdot q_{x_i} \end{cases}$$

$$V^J(i, j) = \max \begin{cases} V^M(i, j-1) \cdot \delta \cdot q_{y_j} \\ V^J(i, j-1) \cdot \varepsilon \cdot q_{y_j} \end{cases}$$

These recurrence relations can be transformed into the form similar to those used in Needleman-Wunsch algorithm. Take the log function on both sides of the recurrence relations, we have:

$$\log V^M(i,j) = \max \begin{cases} \log V^M(i-1,j-1) + \log(1-2\delta) + \log p_{x_i y_j} \\ \log V^I(i-1,j-1) + \log(1-\varepsilon) + \log p_{x_i y_j} \\ \log V^J(i-1,j-1) + \log(1-\varepsilon) + \log p_{x_i y_j} \end{cases}$$

$$\log V^I(i,j) = \max \begin{cases} \log V^M(i-1,j) + \log\delta + \log q_{x_i} \\ \log V^I(i-1,j) + \log\varepsilon + \log q_{x_i} \end{cases}$$

$$\log V^J(i,j) = \max \begin{cases} \log V^M(i,j-1) + \log\delta + \log q_{y_j} \\ \log V^J(i,j-1) + \log\varepsilon + \log q_{y_j} \end{cases}$$

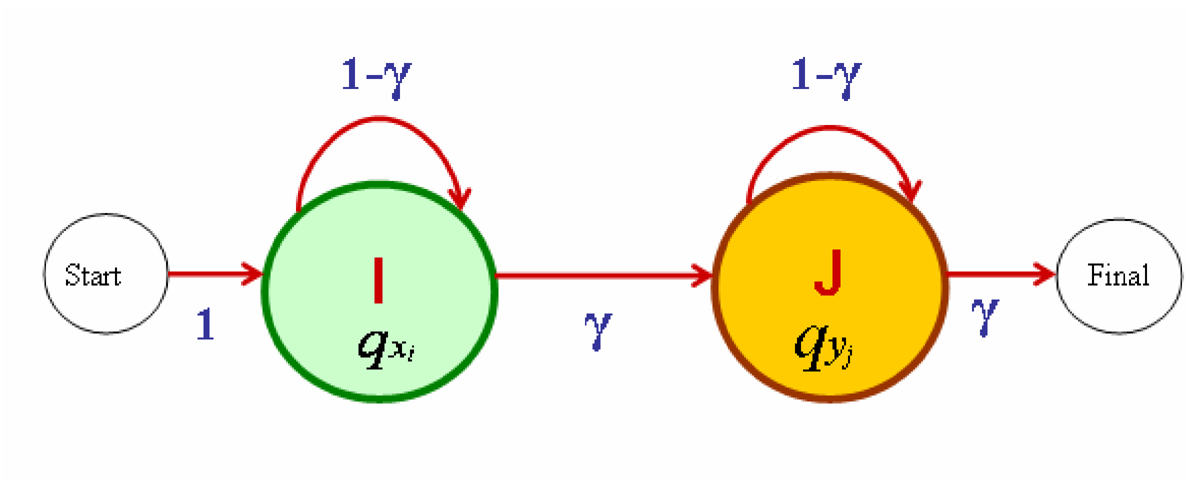
To simplify the solving of the recurrence relations, we could, instead of optimizing $\arg\max_{\pi} P(x,y,\pi)$, optimize the term

$$\arg\max_{\pi} \frac{P(x,y,\pi)}{C(x,y)}$$

where *C(x,y)* is a constant depends only on sequences x an y, but not the alignment π.

The trick is to choose a specific form of *C(x,y)* so that some of the terms will be cancelled. This is done by using a random model.

Random Model



This random model emits all letters from one sequence and then emits all letters from another sequence.
Denote this probability as $P_R(x,y)$ which is independent of π.

Replace *C(x,y)* by $P_R(x,y)$ in our objective function, we have the probability ratio with respect to the random model:

$$\arg\max_{\pi} \frac{P(x,y,\pi)}{P_R(x,y)}$$

We can either define new symbols for the matrices V^M, V^I and V^J, or overload the matrices $V^M(i,j), V^I(i,j), V^J(i,j)$ so that these matrices store the probability ratio with respect to the random model, rather than the actual probability. This can be done by dividing each term of recurrence relations by $P_R(x,y)$ at appropriate time of the sequence.

For example, we align sequences

```
x:      ACGTCCA
y:      AC__CCA

state:  MMIIMMM
```

For each *M* state, divide by $q_{x_i} q_{y_j} (1-\gamma)^2$.

For each *I* state, divide by $q_{x_i}(1-\gamma)$

For each *J* state, divide by $q_{y_j}(1-\gamma)$.

Algorithm: Optimal log-odds Alignment

By applying the probability ratio adjustment, the recurrence relations in the Viterbi algorithm for Pair HMM become:

$$V^M(i,j) = \max \begin{cases} V^M(i-1,j-1)\cdot(1-2\delta)\cdot p_{x_i y_j} / q_{x_i} q_{y_i} (1-\gamma)^2 \\ V^I(i-1,j-1)\cdot(1-\varepsilon)\cdot p_{x_i y_j} / q_{x_i} q_{y_i} (1-\gamma)^2 \\ V^J(i-1,j-1)\cdot(1-\varepsilon)\cdot p_{x_i y_j} / q_{x_i} q_{y_i} (1-\gamma)^2 \end{cases}$$

$$V^I(i,j) = \max \begin{cases} V^M(i-1,j)\cdot\delta\cdot q_{x_i} / q_{x_i}(1-\gamma) \\ V^I(i-1,j)\cdot\varepsilon\cdot q_{x_i} / q_{x_i}(1-\gamma) \end{cases}$$

$$V^J(i,j) = \max \begin{cases} V^M(i,j-1)\cdot\delta\cdot q_{y_j} / q_{y_i}(1-\gamma) \\ V^J(i,j-1)\cdot\varepsilon\cdot q_{y_j} / q_{y_i}(1-\gamma) \end{cases}$$

$$\text{Let } s(xi, yi) = \log\left[(1-2\delta)\cdot p_{x_i y_j} / q_{x_i} q_{y_i} (1-\gamma)^2\right]$$
$$= \log\left[\frac{p_{x_i y_j}}{q_{x_i} q_{y_i}}\right] + \log\left[\frac{(1-2\delta)}{(1-\gamma)^2}\right]$$
$$\approx \log\left[\frac{p_{x_i y_j}}{q_{x_i} q_{y_i}}\right]$$
$$d = -\log\left[\frac{\delta}{(1-\gamma)}\right]$$
$$e = -\log\left[\frac{\varepsilon}{(1-\gamma)}\right]$$

We have

$$\log V^M(i,j) = \max\begin{cases}\log V^M(i-1, j-1) + s(x_i, y_j)\\ \log V^I(i-1, j-1) + s(x_i, y_j)\\ \log V^J(i-1, j-1) + s(x_i, y_j)\end{cases}$$

$$\log V^I(i,j) = \max\begin{cases}\log V^M(i-1, j) - d\\ \log V^I(i-1, j) - e\end{cases}$$

$$\log V^J(i,j) = \max\begin{cases}\log V^M(i, j-1) - d\\ \log V^J(i, j-1) - e\end{cases}$$

Therefore, for any Pair HMM, we can derive an equivalent FSA for obtaining the most probable alignment. There are implicit substitution matrix $s(x_i, y_j)$, gap opening penalty *d* and gap extension penalty *e* associated with a Pair HMM, which is equivalent to what Needleman-Wunsch algorithm does. Therefore the Viterbi algorithm for Pair HMMs is exactly finding the best global alignment with affine gaps.

3. Learning Parameters from HMM

Once the type of model is chosen, the parameters such as the initial probabilities, transition and emission probabilities of the model have to be inferred from data. The goal is to identify the set of parameters that performs well for the pairwise alignments. We need a training set T which consists of ordered triples of the form:

$$T = \left\{\left(x^{(i)}, y^{(i)}, \pi^{(i)}\right)_1^m\right\}$$

where $x^{(i)}$ is the ith sequence of x in the training set T

$y^{(i)}$ is the ith sequence of y in the training set T
$\pi^{(i)}$ is the alignment of $x^{(i)}$ and $y^{(i)}$ in the training set T

In particular, we want the probability

$$P(x^{(i)}, y^{(i)}, \pi^{(i)})$$

to be large for every single example in the training set.

By using a statistical approach known as "Maximum Likelihood Learning", we can identify the solution vector $\overrightarrow{\theta^*}$, which is the set of all parameters for a Pair HMM, such that:

$$\overrightarrow{\theta^*} = \arg\max_{\vec{\theta}} \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}, \pi^{(i)}, \vec{\theta})$$

Since log is a monotonic function, the maximization of above problem is equivalent to maximization of

$$\overrightarrow{\theta^*} = \arg\max_{\vec{\theta}} \sum_{i=1}^{m} \log P(x(i), y(i), \pi(i), \vec{\theta})$$

Solving this problem corresponds to finding solution to the parameters. Solution occurs when

Emission probability q_c to emit a generic letter c:

$$q_c = \frac{\text{\# of c's emitted in I or J state}}{\text{\# of times we're in I or J state}}$$

Transition probability of continuing a gap state:

$$\varepsilon = \frac{\text{\# of times I} \rightarrow \text{I or J} \rightarrow \text{J}}{\text{\# of times in I or J state}}$$

4. EM (Expectation Maximization) algorithm:

This corresponds to the situation when we have some data, but the training set is incomplete. The training set consists of only sequences x and y and there is no example of any sequence alignment, i.e. the training set is of the form

$$T = \left\{ \left(x^{(i)}, y^{(i)} \right)_1^m \right\}$$

Using the same principle, we want to $\overrightarrow{\theta^*}$ that maximizes

$$\overrightarrow{\theta^*} = \arg\max_{\vec{\theta}} \sum_{i=1}^{m} \log P(x(i), y(i), \vec{\theta})$$

Notice that this problem is a lot harder than the one with complete training set because each of the term

$$P(x^{(i)}, y^{(i)}, \vec{\theta})$$

is equivalent to a summation of terms

$$\sum_{\pi} P(x(i), y(i), \pi, \vec{\theta})$$

over all possible alignment π. There is no nice simple form solution. The details of this algorithm will be discussed in the next lecture.