Computational Biology
Lecture 11: Pairwise alignment using HMMs
Saad Mneimneh

We looked at various alignment algorithms with different scoring schemes. We argued that the score of an alignment is related to the relative likelihood that the two sequences are related compared to being unreleated, and we used the log-odds ratio to express this relative likelihood while maintaining an additive scoring scheme.

Therefore, maximizing the score of an alignment was in some sense equivalent to maximizing the log-odds ratio, with the exception that gaps are scored separately and are not related to the log-odds ratio. Recall that we have considered only ungapped alignments in deriving the scores that relate to the log-odds ratio, and assumed a separate model for scoring gaps; for instance, an affine gap penalty function.

Now we will unify both models into a single probabilistic model and see how the score of a gapped alignment of the Needleman-Wunsch algorithm can be viewed as a maximum log-odds ratio obtained by a Viterbi algorithm for an HMM that generates two sequences simultaneously.

Needleman-Wunsch as a Finite State Machine

Recall the Needleman-Wunsch algorithm with an affine gap penalty function.

$$A(i,j) = s(i,j) + \max \begin{cases} A(i-1,j-1) \\ B(i-1,j-1) \\ C(i-1,j-1) \end{cases}$$
$$B(i,j) = \max \begin{cases} A(i-1,j) - e \\ B(i-1,j) - d \end{cases}$$
$$C(i,j) = \max \begin{cases} A(i,j-1) - e \\ C(i,j-1) - d \end{cases}$$

Any alignment generated by this algorithm can be represented as a path starting in state A in the following finite state machine:

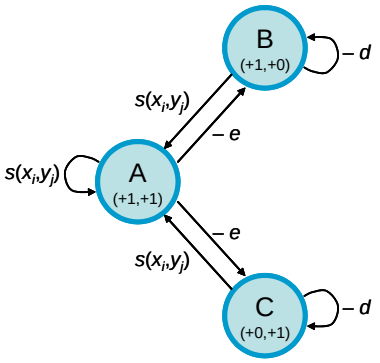

Figure 1: Needleman-Wunsch as FSM

In each state there is an indication to which character is emitted. For instance, in state A, two aligned characters from x and y respectively are emitted. In state B only one character from x aligned with a gap is emitted. In state C only one character from y aligned with a gap is emitted.

The score of the alignment is obtained as the sum of weights of the edges on its corresponding path. A transition from state A to either state B or state C adds a score of $-e$ (the opening of a gap), while a transition from state B to state B or a transition from state C to state C adds a score of $-d$ (a continuing gap). A transition to state A adds the corresponding score for aligning the two characters of x and y respectively.

Needleman-Wunsch as HMM

Let's convert the above finite state machine into an HMM.

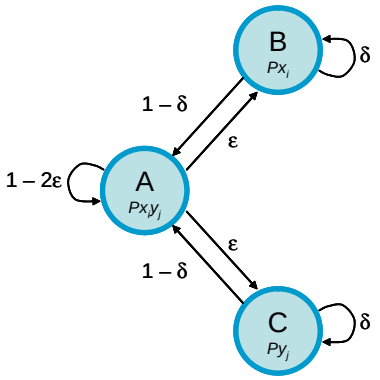

Figure 2: Needleman-Wunsch HMM

We replace the scores on the edges with transition probabilities. For each state, we also assign an emission probability for the corresponding emission. For instance, since state A emits two characters x_i and y_j, we have an emission probability of $p_{x_i y_j}$. Recall that $p_{x_i y_j}$ is the probability of seeing x_i aligned with y_j. Similarly, state B has an emission probability p_{x_i} for emitting x_i. Recall that p_{x_i} is the probability of seeing an x_i. State C is similar to state B. Let's not worry about the meaning of the transition probabilities for now, but obviously ϵ is related to the opening of a gap and δ is related to the continuing gap.

We can always model the length of an alignment by adding an explicit end state as the following figure shows, but let's not worry about this.

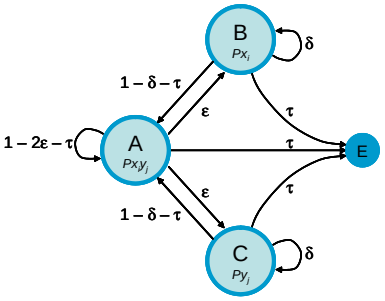

Figure 3: Needleman-Wunsch HMM with end state

Now given an alignment, we can ask the following: What is the probability of obtaining this alignment? Here's an example:

Consider the following alignment:

```
VLSPAD-K
HL--AESK
```

Starting at A, the probability of obtaining this alignment is:

$$(1-2\epsilon)p_{VH}.(1-2\epsilon)p_{LL}.\epsilon p_S.\delta p_P.(1-\delta)p_{AA}.(1-2\epsilon)p_{DE}.\epsilon p_S.(1-\delta)p_{KK}$$

We can also ask the following question: Given two sequences, what is their most probable alignment? This of course can be solved by a Viterbi algorithm. But we have to be careful here since there is an extra dimension in the search space because of the extra emitted sequence. Therefore, instead of using $v_k(i)$ as before, we will have to use $v_k(i,j)$. Note that i and j are not synchronized because x and y are not generated in a synchronized way (there might be gaps in the alignment and regardless of that x and y might have different lengths). Nevertheless, $v_k(i,j)$ can advance only in certain ways: In state A both i and j advance, in state B only i advances, and in state C only j advances. With these restrictions, the Viterbi algorithm will be as follows:

$$
\begin{aligned}
&v_A(0,0) = 1, v_A(i,0) = v_A(0,i) = 0 \\
&v_B(0,0) = 0, v_B(i,0) = \epsilon\delta^{i-1}p_{x_1}\dots p_{x_i}, v_B(0,j) = 0 \\
&v_C(0,0) = 0, v_C(i,0) = 0, v_C(0,j) = \epsilon\delta^{j-1}p_{y_1}\dots p_{y_j} \\
&\textbf{for } i = 1...m, j = 1..n \\
&\quad v_A(i,j) = p_{x_i y_j}.\max\begin{cases}(1-2\epsilon)v_A(i-1,j-1)\\(1-\delta)v_B(i-1,j-1)\\(1-\delta)v_C(i-1,j-1)\end{cases} \\
&\quad v_B(i,j) = p_{x_i}.\max\begin{cases}\epsilon v_A(i-1,j)\\\delta v_B(i-1,j)\end{cases} \\
&\quad v_C(i,j) = p_{y_j}.\max\begin{cases}\epsilon v_A(i,j-1)\\\delta v_C(i,j-1)\end{cases} \\
&\textbf{return } \max[v_A(m,n), v_B(m,n), v_C(m,n)]
\end{aligned}
$$

Of course, by keeping pointers as before, we obtain the most probable alignment itself.

Now this Viterbi algorithm looks very similar the the Needleman-Wunsch algorithm. Performing Viterbi in log space will definitly make it even more similar to Needleman-Wunsch (transforming multiplications to additions). Let's do even more. We will modify the Viterbi algorithm to directly compute the log-odds ratio of the maximum probability of an alignment (path in HMM) to the probability of obtaining the sequences at radom; therefore, to compute the maximum log-odds ratio. Of course we can do this my simply dividing the result of the Viterbi algorithm by the probability of obtaining the sequences at random, and then taking the log. But we want to see how the Viterbi algorithm will look like if it is made to compute this directly.

log-odds Viterbi

In order to obtain the log-odds Viterbi algorithm, we will constrcut a model for two random sequences and then perform Viterbi in log space with both models in mind: the previous HMM and the random model.

Let us first define a model for two random sequences. All we need is for the model to generate the two sequences independently at random. The following HMM captures this aspect.

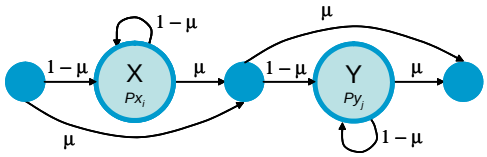

Figure 4: A model for two random seuqnences

The random model generates sequence x first, then seuquence y indendently of x. While generating x, each character x_i is emitted according to the probability x_i. Same for y. Note that the model allows empty sequences for both x and y.

The probability of two sequences x and y being generated by the random model is:

$$p(x,y) = \mu^2(1-\mu)^{m+n}\prod_{i=1\dots m}p_{x_i}\prod_{j=1\dots n}p_{y_j}$$

We will obtain the log-odds Viterbi as follows. Assume that an alignment of x and y is obtained optimally by a path with probability $p = p_1.p_2.p_3.....$ Assume also that x and y can be obtained by the random model with probability $q = q_1.q_2.q_3.....$ Then we will modify Viterbi to use $(p_1/q_1).(p_2/q_2).(p_3/q_3)....$ as the probability of the path. If we succeed in doing this, then performing Viterbi in log space will compute what we want, the log-odds ratio $\log(p_1/q_1) + \log(p_2/q_2) + \log(p_3/q_3) + ...$

How can we do that? We will identify "steps" in each model with a probability for each step. Then we will associate steps together in a way to make both models generate x and y at the same time, i.e. if the alignment model generates a character in a sequence, the random model will generate the same character in the same sequence. For each step with probability p in Viterbi, we replace p by p/q, where q is the probability of the corresponding step in the random model. Therefore, we need to identify steps with their contributed probabilities in both models, then associate steps together.

- In the alignment model
  - each match step contributes $(1-2\epsilon)p_{x_i y_j}$ (well, it depends on how state A is reached, but let us assume for now that we always reach A from A).
  - each start gap step contributes ϵp_{x_i} or ϵp_{y_j}. To correct for above, replace ϵ with $\epsilon\frac{1-\delta}{1-2\epsilon}$.
  - each continuing gap step contributes δp_{x_i} or δp_{x_i}.
- In the random model
  - each character step contributes $(1-\mu)p_{x_i}$ or $(1-\mu)p_{y_j}$.
  - all terms are counted except for a μ^2.

$$
\begin{array}{l}
v_A(0,0)=1, v_A(i,0)=v_A(0,i)=0 \\
v_B(0,0)=0, v_B(i,0)=\epsilon\frac{1-\delta}{1-2\epsilon}\delta^{i-1}p_{x_1}\dots p_{x_i}, v_B(0,j)=0 \\
v_C(0,0)=0, v_C(i,0)=0, v_C(0,j)=\epsilon\frac{1-\delta}{1-2\epsilon}\delta^{j-1}p_{y_1}\dots p_{y_j} \\
\textbf{for } i=1...m, j=1..n \\
\quad v_A(i,j)=p_{x_i y_j}.\max\left\{\begin{array}{l}(1-2\epsilon)v_A(i-1,j-1)\\(1-2\epsilon)v_B(i-1,j-1)\\(1-2\epsilon)v_C(i-1,j-1)\end{array}\right. \\
\quad v_B(i,j)=p_{x_i}.\max\left\{\begin{array}{l}\epsilon\frac{1-\delta}{1-2\epsilon}v_A(i-1,j)\\\delta v_B(i-1,j)\end{array}\right. \\
\quad v_C(i,j)=p_{y_j}.\max\left\{\begin{array}{l}\epsilon\frac{1-\delta}{1-2\epsilon}v_A(i,j-1)\\\delta v_C(i,j-1)\end{array}\right. \\
\textbf{return } \max[v_A(m,n), v_B(m,n)\frac{1-2\epsilon}{1-\delta}, v_C(m,n)\frac{1-2\epsilon}{1-\delta}]
\end{array}
$$

The reason for having the extra term $\frac{1-\delta}{1-2\epsilon}$ is the following: we assumed that a match (state A) is always reached from state A and hence always contributes a probability $1-2\epsilon$. However, it is possible that state A is reached from either state B or state C, in which case it will contribute a probability of $1-\delta$ and not $1-2\epsilon$. Therefore, we anticipate this before it happens, and as soon as we start a gap, we multiply the probability by an adjusting factor $\frac{1-\delta}{1-2\epsilon}$. When we come back to A from the gap, we (mistakingly) multiply the probability by $1-2\epsilon$, which in presence of the previously adjusting factor $\frac{1-\delta}{1-2\epsilon}$, produces the correct probability $1-\delta$.

But what if we do not come back to A after the last transition out of A? In that case we end up in state B or C; therefore, we have to cancel this adjusting factor at the end.

Now let us associate steps together:

alignment model	random model
match x_i with y_j	generate x_i and y_j
start gap in y with x_i	generate x_i
start gap in x with y_j	generate y_j
continuing gap in y with x_i	generate x_i
continuing gap in x with y_j	generate y_j

Now let us define $\log(p/q)$ for each step of the alignment model:

match step: call this $s(x_i,y_j)=\log\frac{p_{x_i y_j}(1-2\epsilon)}{(1-\mu)p_{x_i}(1-\mu)p_{y_j}}=\log\frac{p_{x_i y_j}(1-2\epsilon)}{p_{x_i}p_{y_j}}+\log\frac{1}{(1-\mu)^2}\approx\log\frac{p_{x_i y_j}(1-2\epsilon)}{p_{x_i}p_{y_j}}$

continuing gap step: call this $-d=\log\frac{\delta p_{x_i}}{(1-\mu)p_{x_i}}=\log\frac{\delta}{1-\mu}\approx\log\delta$

start gap step: call this $-e=\log(\frac{\epsilon p_{x_i}}{(1-\mu)p_{x_i}}\frac{1-\delta}{1-2\epsilon})\approx\log\epsilon\frac{1-\delta}{1-2\epsilon}$

Rewriting the Viterbi algorithm in log space and using the modified probabilities we get:

$$
\begin{aligned}
&V_A(0,0) = -2\log\mu, V_A(i,0) = V_A(0,i) = -\infty \\
&V_B(0,0) = -\infty, V_B(i,0) = -e-(i-1)d, V_B(0,j) = -\infty \\
&V_C(0,0) = -\infty, V_C(i,0) = -\infty, V_C(0,j) = -e-(j-1)d \\
&\textbf{for } i = 1...m, j = 1..n \\
&\quad V_A(i,j) = s(x_i,y_j) + \max\begin{cases} V_A(i-1,j-1) \\ V_B(i-1,j-1) \\ V_C(i-1,j-1) \end{cases} \\
&\quad V_B(i,j) = \max\begin{cases} V_A(i-1,j)-e \\ V_B(i-1,j)-d \end{cases} \\
&\quad V_C(i,j) = \max\begin{cases} V_A(i,j-1)-e \\ V_C(i,j-1)-d \end{cases} \\
&\textbf{return } \max[V_A(m,n), V_B(m,n)-c, V_C(m,n)-c]
\end{aligned}
$$

$V_A(0,0)$ is initialized to $-2\log\mu$ instead of 0 to account for the $\frac{1}{\mu^2}$ term that was not included in any step of the random model. The term $c = \log\frac{1-\delta}{1-2\epsilon}$ is substracted from $V_B(m,n)$ and $V_C(m,n)$ since, as explained above, these indicate that we did not come back to state A after the last transition out of A; therefore, we have to remove this factor that we previously added.

The log-odds Viterbi is now the same as the Needleman-Wunsch algorithm. In fact, Needleman-Wunsch is just Viterbi with an appropriate transformation! We knew from before that the score is related to the log-odds ratio of probabilities. Now Viterbi gives more justification of this (we are actually after the most likely alignment compared to just a random instance). We have a clear relation between the scores (even for gaps now) of Needleman-Wunsch and the parameters of a probabilistic model.

The question now is: what do the model parameters ϵ, δ, and μ really represent? To answer this question, consider a state X with a transition probability $a_{XX} = 1-p$. What is the expected duration in state X? This is a geometric distribution. Each time we have a probability p of leaving X. Therefore, the expected duration in state X is $\frac{1}{p}$. Applying this to our three parameters we have:

- $2\epsilon = \frac{1}{L_u}$, where L_u is the expected length of ungapped regions (expected duration in state A). Note that if $L_u \nearrow$, then $\epsilon \searrow$, then $e \nearrow$. Therefore, the start gap penalty is high if ungapped regions are large, which makes sense.
- $1-\delta = \frac{1}{L_g}$, where L_g is the expected length of a gap (expected duration in state B or state C). Note that if $L_g \nearrow$, then $\delta \nearrow$, then $d \searrow$. Therefore, the additional gap penalty is small if gaps are large, which makes sense.
- $\mu = \frac{1}{L}$, where L is the expected length of an arbitrary sequence. Therefore, μ is very small.

References

Durbin R. et al, Biological Sequence Analysis, Chapter 4.