Pair Hidden Markov Models

Scribe: Rishi Bedi
Lecturer: Serafim Batzoglou

January 29, 2015

1 Recap of HMMs

- alphabet: $\Sigma = \{b_1, ...b_M\}$
- set of states: $Q = \{1, ..., K\}$
- transition probabilities: $A = [a_{ij}]$
- initial state probabilities: a_{0i}
- emission probabilities: $e_i(b_k)$

2 Pair Hidden Markov Models

Pair HMMs give a probability distribution over certain sequences of pairs of observations. With respect to sequence alignment, we have 2 sequences in our pairwise alignment case.

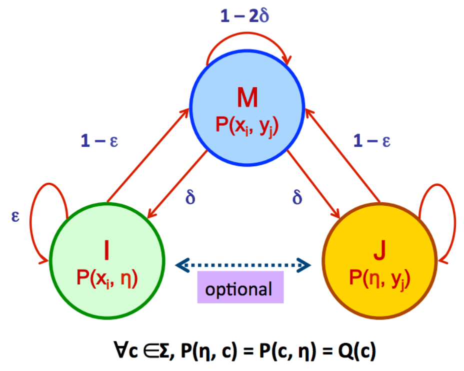
Question: What is our sequence of pairs of observations? Each pair of observations is either letter / letter, or letter / gap, or gap / letter. Consider the following HMM :

$$((\Sigma_1 \cup \{\eta\})x(\Sigma_2 \cup \{\eta\}), Q, A, a_o, e)$$

So in every state, we have the choice of emitting two letters, or a letter paired with the empty string (denoted by η).

2.1 Example Model

Let's walk through the following pair-HMM (given in the lecture slides):



We always emit a pair of letters. Define P as a table of letters (for every pair of letters X, Y, define P(X,Y)), with each probability summing to 1. M, I, J are similarly defined (M corresponds to emissions consisting of pairs of letters, I corresponds to a letter and a gap, and J consists of a gap and a letter).

We define transitions between the states as shown, noting that δ and ϵ are very small. Furthermore, $\delta < \epsilon$ (so opening the gap happens with a smaller probability, and thus incurs a "larger penalty", than extending the gap).

2.2 Stepping through the example model

What does this have to do with sequences? Let's look at an example alignment using the pair HMM defined above. Consider this alignment:

AG-TCA
AGGTC-

The state path for this alignment is: MMJMMI.

Given the state path, we can assign a probabiity to this alignment

$$P(A,A) * (1-2\delta) * P(G,G) * \delta * P_J(-,G) * \epsilon * P(T,T) * (1-2\delta) * P(C,C) * \delta * P_I(A,-)$$

We get a different probability for every alignment. This procedure can always be used to go from alignment to state path and then to probability of the alignment.

2.3 Adapting Viterbi Decoding

Now, we consider a harder question: can we find optimal alignment using this state model? What is the connection between this state model and the traditional alignment algorithms we've been talking about? Also, how does the probability of a pair of letters correspond to matches and mismatches?

With traditional HMMs, we defined the Viterbi algorithm to do this process (i.e., finding a state sequence that occurs with maximum probability and thus defines the best alignment). This was a straight-forward algorithm, using dynamic programming.

It is tempting to apply the same Viterbi algorithm to pair HMMs. The naive adaptation would be, for every position i, letting $V_k(i)$ be the optimal way to emit the first i letters of x and the first i letters of y, ending in state k. But that doesn't quite work, because we dont know that the first i letters of x align precisely with the first i letters of y.

Let's assume inductively that for smaller numbers of x and y, that Viterbi is correctly computed, we can now compute for a given position in x and y, and a given state, what is the optimal Viterbi path.

Denote $Q(x_i) = P(x_i, \eta)$, and $Q(y_j) = P(\eta, y_j)$. Now we can show a fixed Viterbi, for pair-HMMs:

$$V_M(i, j) = P(x_i, y_j) \max \begin{cases} (1 - 2\delta)\, V_M(i - 1, j - 1) \\ (1 - \varepsilon)\, V_I(i - 1, j - 1) \\ (1 - \varepsilon)\, V_J(i - 1, j - 1) \end{cases}$$

$$V_I(i, j) = Q(x_i) \max \begin{cases} \delta\, V_M(i - 1, j) \\ \varepsilon\, V_I(i - 1, j) \end{cases}$$

$$V_J(i, j) = Q(y_j) \max \begin{cases} \delta\, V_M(i, j - 1) \\ \varepsilon\, V_J(i, j - 1) \end{cases}$$

We can define forward and backward algorithms similarly.

Forward example:

$$F_M(i,j) = \text{Prob}(x_1 ... x_i, y_1 ... y_j, x_i \sim y_j)$$

In other words, the probability that the first i letters of x align to first j letters of y, and x_i aligns to y_j is given by $P(x_i, y_j)$ * (the sum of the three probabilities given for $V_M(i,j)$ that are maxed over for the pair-HMM Viterbi algorithm).

2.4 Probabilistic Interpretation

Consider the product over all the sequence characters $Q(x_i)Q(y_j)$. This is a very small constant. This is the probability of emitting every letter with a gap (i.e., nothing matches...) Emit x by itself ... and then emit y by itself. Imagine taking the probability of an alignment, and dividing it by this quantity. It's a constant - so it doesn't favor any one alignment over another. The optimal alignment will be the same, if you divide the probability of every alignment by this number. But we have something interesting now – a probabilistic interpretation of alignment. This new "denominator" is the baseline theory – that the two sequences were generated independently of each other. (the null hypothesis, in a sense?) If you take the alignment probability, and divide it by this baseline probability, and that quotient is > 1, it is more likely than not that the sequences were generated by the pair HMM (and have characters in common)... but if it's < 1, it is more likely than not that the sequences are in fact independent and were emitted randomly, not by the pair HMM. Rather than dividing by everything at the end, we'll divide along the way (every time we see an x_i, divide by $Q(x_i)$, etc...).

2.5 Connection to Needleman-Wunsch

Can we find a connection between Viterbi for pair HMMs and Needleman-Wunsch? Given a NW scoring function, how do you set pair HMM viterbi parameters to make their outputs match? Maximizing the multiplication of probabilities as we do here in the Viterbi algorithm is the same maximizing the sum of the logs of the probabilities. Then our DP looks very similar and we can connect the scoring parameters to the Viterbi parameters.

2.5.1 Transforming pair Viterbi to NW with affine gaps

1. Convert products to sum of logs.
2. The log of a number close to 1 is very close to 0, so remove small terms.
3. Every time you open a gap you pay $\log(\delta)$
4. Every time you extend a gap you pay $\log(\epsilon)$

Viterbi has a few extra small terms we dropped but the DP is virtually identical.

2.5.2 Setting N-W Parameters

We can use the following procedure to set the parameters of Needleman-Wunsch to have real meaning now:

1. Hand-curate (true) alignments that we trust.

2. Count up how frequently given transitions (between states) occur, and the expected gap length.

3. Compute the maximum likelihood parameters using those counts.

If we don't have necessarily true alignments at our disposal, but we have sequences we think that align, use Baum-Welch as discussed in a previous lecture.

2.5.3 Scoring Example

- $S(x_i, y_j) = \log \dfrac{P(x_i, y_j)}{Q(x_i)Q(y_j)}$
- $P(AA) = P(CC) = P(GG) = P(TT) = 0.2$
- $P(x \neq i) = 0.2/12$ (the 12 different ways you could have a mismatch)
- We can set $Q(A) = 0.25$
- Now $m = \log \frac{0.2}{0.25^2} = 1.16$ and $s = \log \frac{0.2/12}{0.25^2} = -1.32$
- Consider 80%, 20% mismatch. That yields $+66$. Clearly more aligned than not.
- $.8 + (.25/2) = .525 =$ break-even point (score of 0).
- Implication: Need a little bit more matches than mismatches (52.5%) for the sequences to be considered a positive alignment.

2.6 Conditional Random Fields

We don't really care about generating x. So let's lose that ability in favor of a much richer model for $P(\pi|x)$, the probability of a parse given a sequence.

$$P(\pi|x) = \frac{exp(\sum_{i=1...|x|} w^T F(\pi_i, \pi_{i-1}, x, i))}{\sum_{\pi'} exp(\sum_{i=1...|x| w^T F(\pi', \pi'_{i-1}, x, i))}}$$

The denominator of the above expression gives the sum over scores of all possible parses (forces the quantity to become a probability); i.e., the sum over all possibilities of numerators.

We are really only interested in the exponent because the proability is monotonic with respect to the exponent.

- w is a vector - a set of weights of different parameters of the model
- F is a given set of feature counts (e.g., a pair of states (current state, previous state), sequence of observations, and the position we're in)
- $w^T F$ is dot product of w and F.

2.6.1 CRFs versus HMMs

CRFs are richers than HMMs. In HMMs, parameters must obey the constraint of being probabilities, while this is not so for CRFs. CRFs can contain features that are not possible to include in HMMs.

In HMMs, we were only allowed to consider x_i (cannot look at $x_i - 1$ or $x_i - 2$ because that would be a double emission). With a CRF, there are no longer emissions, so this restriction no longer applies, so you can look at how many ever previous events you want.

CRFs require training data with knowledge of underlying alignments; i.e., there is no equivalent of the Baum-Welch algorithm for CRFs.

CRF feature selection ultimately relies on intuition. One example: might look for codon pattern in genes (i.e., MMS MMS MMS).

There is a connection between HMMs and CRFs: to convert a HMM into a CRF, the CRF's feature vector would be a 1 indicator function for the specific transition and emission, and 0 for everything else.