CS262 Lecture 7 Notes
Instructor: Serafim Batzoglou
Scribe: Qianying Lin

Lecture 7: HMMs continued

January 26, 2016

Key Concepts & Useful Applications

1. Learning Hidden Markov models
2. Re-estimate the parameters based on the training data
3. Pair Markov Models

Applications:

1. Learning Markov model allows us to estimate the parameters of a Markov model based on the training data available.
   According to the paper *Hidden Markov Models and their Applications in Biological Sequence Analysis*, HMMs have been shown to be very effective in representing biological sequences, as they have been successfully used for modeling speech signals. As a result, HMMs have become increasingly popular in computational molecular biology, and many state-of-the-art sequence analysis algorithms have been built on HMMs.
2. Pair Markov model could be used to do sequence alignment, as in we can put (character, character), (character, blank), (blank, character) as three different states.

Lecture Content

<u>Part I. Hidden Markov Model</u>

1. Viterbi, forward and backward algorithms are very similar.

$v_k(i)$: The probability of having the optimal way (i.e. highest probability) of π_1 up to π_{i-1} so as to generate $x_1 \dots x_{i-1}$ from π_1 up to $\pi_{i-1}, \pi_i = k$.

$$v_k(i) = max_{\{\pi_1,\dots,\pi_{i-1}\}} P(x_1, \dots x_{i-1}, \pi_1, \dots, \pi_{i-1}, x_i, \pi_i = k)$$

$f_k(i)$: The probability of any π_1 up to π_{i-1} generating $x_1 \dots x_i$ with $x_i = k$

$$f_k(i) = P(x_1, \dots x_{i,}\ \pi_i = k)$$

Question: which one is smaller?

Answer: $v_k(i)$ since it is only the max of the terms. Yet, $f_k(i)$ is the sum of the exponentially large terms.

$b_k(i) = P(x_{i+1}, \dots x_{n,}\ \pi_i = k)$

2. Two learning scenarios:

➔ Estimation when the right answer is known

1) A genomic region x = $x_1 \dots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands
2) The casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls.

➔ Estimation when the right answer is unknown

1) The porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition.
2) 10,000 rolls of the casino player, but we don't see when he changes dice.

$P(x|\theta)$: The best θ for our observed sequences

3. When the states are known:

Maximum likelihood parameters: the number of 6s/ total number for loaded dice

Define:

A_{kl} = number of times k→l transition occurs in π

$E_k(b)$ = number of times state k in π emits b in x

The maximum likelihood parameters:

$a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}}$

$e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)}$

Possible drawback: Little data might lead to over-fitting (0 probability) -> stuck in 0, never recovers -> solution: pseudo-counts

Pseudo-counts:

A_{kl} = # times k⟶l transition occurs in π + r_{kl}

$E_k(b)$ = # times state k in π emits b in x + $r_k(b)$

r_{kl}, $r_k(b)$ are pseudo counts representing our prior belief

Larger pseudo counts ⇒ Strong prior belief

Small pseudo counts (ε < 1): just to avoid 0 probabilities

4. When the states are hidden:

   i.e. We don't know the true A_{kl}, $E_k(b)$.

   Idea:

   - Estimate our "best guess" on what A_{kl}, $E_k(b)$ are

     Or, we start with random / uniform values

   - Update the parameters of the model, based on our guess

   - Repeat

   The principle is Expectation Maximization.

   1. Estimate A_{kl}, $E_k(b)$ in the training data

   2. Update θ according to A_{kl}, $E_k(b)$

   3. Repeat 1 & 2, until convergence

5. So the question boils down to how to estimate the new parameters.

   To estimate $\mathbf{A_{kl}}$, the transition probability.

   $$P(\pi_i = k, \pi_{i+1} = l \mid x) = \frac{Q}{P(x)}$$

   Where

   $$\begin{aligned} Q &= P(x_1, \dots, x_N, \pi_i = k, \pi_{i+1} = l) \\ &= P(\pi_{i+1} = l, x_{i+1}, .., x_N \mid \pi_i = k) P(x_1, \dots, x_i, \pi_i = k) \\ &= P(\pi_{i+1} = l, x_{i+1}, .., x_N \mid \pi_i = k)\, f_k(i) \\ &= P(x_{i+2}, .., x_N \mid \pi_{i+1} = l) P(x_{i+1} \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) f_k(i) \\ &= b_l(i+1) e_l(x_{i+1}) a_{kl} f_k(i) \end{aligned}$$

   Hence,

   $$A_{kl} = P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{b_l(i+1) e_l(x_{i+1}) a_{kl} f_k(i)}{P(x \mid \theta)}$$

6. The Baum-Welch Algorithm

   Complexity for one iteration: $O(nk^2)$ because it is dominated by forward and backward pass.

   Initialization:

   Pick the best guess for model parameters (or arbitrary)

   Iteration:

1. Forward
2. Backward
3. Calculate: A_{kl}, $E_k(b)$ given current θ
4. Calculate new model parameters: update θ based on A_{kl}, $E_k(b)$.
5. Calculate new log-likelihood $P(x|\theta)$

Note that this algorithm is not guaranteed to find globally best parameters

It will converge to local optimum, depending on initial conditions.

7. Alternative: Viterbi training.

Initialization: Same as Baum-Welch Algorithm.

Iteration:

1. Perform Viterbi, to find π*
2. Calculate A_{kl}, $E_k(b)$ according to π* + pseudocounts
3. Calculate the new parameters A_{kl}, $E_k(b)$

Until convergence

Part II. Pair HMMs

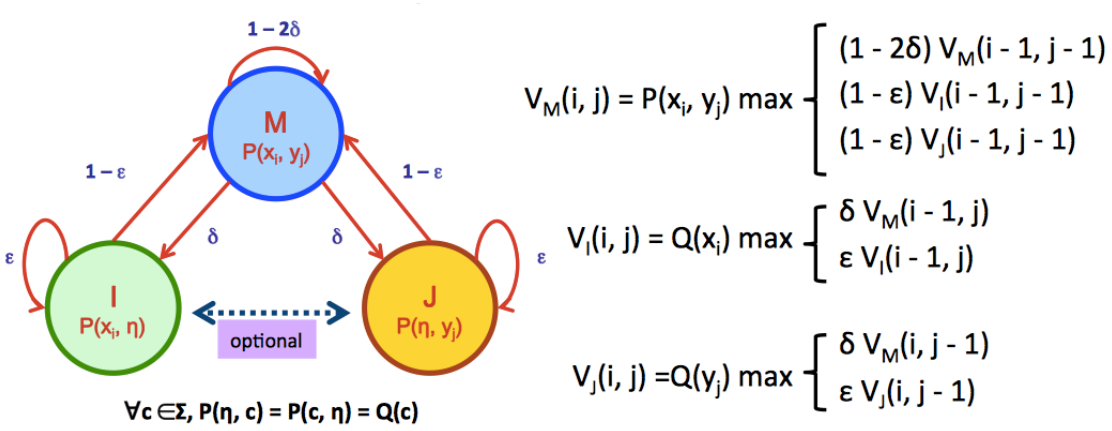

Figure 1: Pair HMM model for sequence alignment

8. Definition of Pair HMMs

$((\Sigma_1 \cup \{\eta\}) \times (\Sigma_2 \cup \{\eta\}), Q, A, a_0, e)$

Where an HMM = (Σ, Q, A, a_0, e).

Alphabet: $\Sigma = \{b_1, \ldots, b_M\}$

Set of states: $Q = \{1, \ldots, K\}$

Transition probabilities: $A = [a_{ij}]$

Initial state probabilities: a_{0i}

Emission probabilities: $e_i(b_k)$

9. Figure 1 is a pair HMM for sequence alignment. Details about the Model:
   1) Transition from I to J: allow A_ matching to _C. This is optional because usually it is better just to match A with C.
   2) Connection with an alignment: **AGT[ ][ ]CAT** VS **AGTTTCA[ ]** : MMMJJMMI. Hence, there is a one-to-one correspondence between paths and alignments.
   3) Ending in state M, I, J: modeling the affine gaps alignment

   δ: Open the gap. Larger than ϵ.

10. The time complexity is quadratic for Viterbi for pair HMMs.
11. Connection to NW: NW is positive, addition. This model multiplies, not necessarily positive.
12. Q: Prior probability of a given letter, assuming independent generation. We take the two models, if the numerator greater then alignment more likely. If the denominator is greater, then independent generation is more likely. Essentially: divide by independent generation. For any alignment, the product of Q is constant. Optimal alignment will not change.

Example for sequence alignment using Pair Markov Model

Assume Sequence identity 88%.

P(A, A) + ... + P(T, T) = 0.88, therefore P(A, A) = 0.22

P(A,C)+...+P(G,T)=0.12,thereforeP(x,y,x!=y)=0.01

$$P(match) = \log\left(\frac{0.22}{0.25^2}\right) = 1.25846$$

$$P(mismatch) = \log\left(\frac{0.01}{0.25^2}\right) = -1.83258$$

Score of ungapped aligned region = 0:

1.25846f – 1.83258(1 – f) = 0

Therefore, f = 1.83258/(1.25846 + 1.83258) = 0.5929

Thus anything smaller 0.5929 will be likely from independent generation. ☺

CS262 Lecture 7 Notes
Instructor: Serafim Batzoglou
Scribe: Qianying Lin

Reference:

1. Lecture Slide 7
2. Hidden Markov Models and their Applications in Biological Sequence Analysis, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766791/