ECE 208 - Homework 5
Profile HMM for Sequence Matching

Feb 2021

In this document, we assume you are familiar with the concepts of a profile HMM model, including the graphical representation, the states, and the emission/transition distributions/probabilities. Throughout this document we will use ‘M’, ‘I’, ‘D’, ‘B’, and ‘E’ to refer to the matching, insertion, deletion, begin, and end states, respectively.

1 The HMM file structure

We follow the conventions in the HMMER software to store a HMM model in a file. We provide to you all the routines to parse the HMM file and load it to a Python data structure (see the next section). In principle, you do not need to understand the HMM file structure to complete this homework. However, if you want to learn about the file structure to make your own tests, you can read the HMMER’s manual, pages 210-215. We will ignore all the headers in the .hmm files except for the LENG attribute, so you can skip to the main part on pages 214 and 215.

2 The HMM data structure

We build a class **HMM** to represent a HMM model in Python. It has the following attributes:

- `HMM.alphabet`: the alphabet of the HMM model. This is a list of characters.
- `HMM.nstate`: the number of M states in the model. This is an integer.
- `HMM.eM`: emission distributions of the M states. This is a list of dictionaries; `eM[j][c]` is the *log* of the probability that state M_j emits c, where c is a letter in the alphabet. We treat the B state as M_0 so `eM[0]` is always an empty dictionary.
- `HMM.eI`: emission distributions of the I states; it has the same data structure as `HMM.eM`.

- `HMM.t`: transition distributions. This is a list of dictionaries, each has 9 keys (`‘MM’`,`‘MI’`,`‘MD’`,`‘IM’`,`‘II’`,`‘ID’`,`‘DM’`,`‘DI’`, and `‘DD’`) corresponding to the 9 possible transitions of the model. See table 1 below for more details.

Concept	Math notation	HMM class accession
Emission probabilities	$\log e_j^M(c)$	`eM[j][c]`
	$\log e_j^I(c)$	`eI[j][c]`
Transitions from M state	$\log a_{M_jM_{j+1}}$	`t[j][‘MM’]`
	$\log a_{M_jI_j}$	`t[j][’MI’]`
	$\log a_{M_jD_{j+1}}$	`t[j][‘MD’]`
Transitions from I state	$\log a_{I_jM_{j+1}}$	`t[j][‘IM’]`
	$\log a_{I_jI_j}$	`t[j][’II’]`
	$\log a_{I_jD_{j+1}}$	`t[j][‘ID’]`
Transitions from D state	$\log a_{D_jM_{j+1}}$	`t[j][‘DM’]`
	$\log a_{D_jI_j}$	`t[j][‘DI’]`
	$\log a_{D_jD_{j+1}}$	`t[j][‘DD’]`

Table 1: Accession of HMM model parameters in the HMM data structure

In addition, the HMM class also has 4 *methods*, 2 of which is fully provided to you and the other 2 are left for you to implement. Refer to table 2 below for details.

Method	Purpose	Status
`load(self,file)`	read HMM file	provided
`compute_llh(self,seq_path)`	compute log likelihood of a sequence path	provided
`Viterbi(self,query)`	find the most probable path	**to be implemented**
`Forward(self,query)`	compute marginal probability	**to be implemented**

Table 2: Methods in the HMM data structure

3 Viterbi recursive formula to compute log-likelihood

Let $V_j^M(i)$ be the log-likelihood of the best path matching subsequence $x_{0..i}$ to the submodel up to state j, ending with x_i being emitted by state M_j. Similarly, let $V_j^I(i)$ be the log-likelihood of the best path ending in x_i being emitted by I_j, and $V_j^D(i)$ is the log-likelihood of the best path ending in state D_j to matching the subsequence up to x_i. Then we can write:

$$V_j^M(i) = \log e_j^M(x_i) + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j} \end{cases} \quad (1)$$

$$V_j^I(i) = \log e_j^I(x_i) + \max \begin{cases} V_j^M(i-1) + \log a_{M_j I_j} \\ V_j^I(i-1) + \log a_{I_j I_j} \\ V_j^D(i-1) + \log a_{D_j I_j} \end{cases} \tag{2}$$

$$V_j^D(i) = \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1} D_j} \\ V_{j-1}^I(i) + \log a_{I_{j-1} D_j} \\ V_{j-1}^D(i) + \log a_{D_{j-1} D_j} \end{cases} \tag{3}$$