CSCE 970 Lecture 3: HMM Application: Biological Sequence Analysis

Stephen D. Scott

Introduction

- Idea: Given a collection S of related biological sequences, build a (profile) hidden Markov model M to generate the sequences
- Then test new sequence X against M using (which algorithm?) to predict X's membership in S
- Can also align X against M using (which algorithm?) to see how X matches up position by position against sequences in S

Introduction (cont'd)

- Will build M based on a multiple alignment of sequences in S:

```
... V G A - - H A G E Y ...
... V - - - - N V D E V ...
... V E A - - D V A G H ...
... V K G - - - - - - D ...
... V Y S - - T Y E T S ...
... F N A - - N I P K H ...
... I A G A D N G A G V ...
```

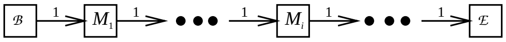
- In alignments, will differentiate matches, insertions, and deletions

Outline

- Ungapped regions
- Insert and delete states
- Deriving profile HMMs from multiple alignments
- Searching with profile HMMs
- Variants for non-global alignments
- Estimating probabilities

Organization of a Profile HMM

- Start with a trivial HMM M (not really hidden at this point)

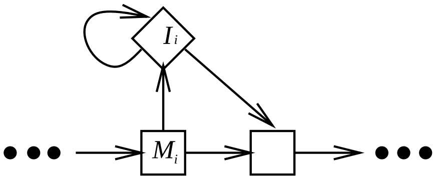


- Each match state has its own set of emission probs, so we can compute prob of a new sequence x being part of this family:

$$P(x \mid M) = \prod_{i=1}^{L} e_i(x_i)$$

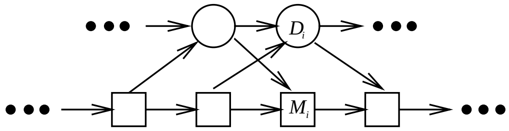
Organization of a Profile HMM (cont'd)

- But this assumes ungapped alignments!
- To handle gaps, consider insertions and deletions
  - Insertion: part of x that doesn't match anything in multiple alignment (use insert states)

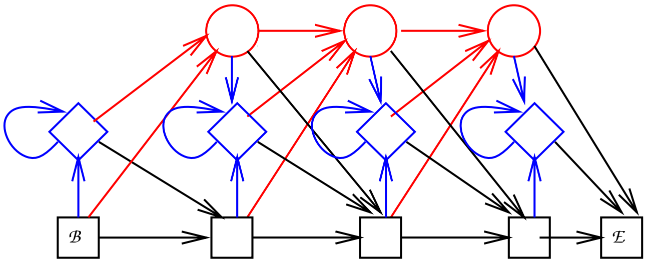


Organization of a Profile HMM
(cont'd)

- Deletion: parts of multiple alignment not matched by any residue (symbol) in x (use silent delete states)

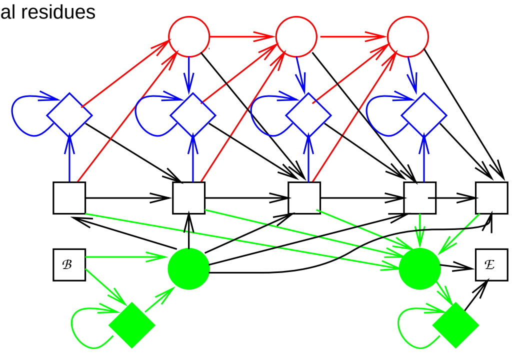


General Profile HMM Structure



Handling non-Global Alignments

- Original profile HMMs model entire sequence
- Add flanking model states (or free insertion modules) to generate non-local residues



Building a Model

- Given a multiple alignment, how to build an HMM?
  - General structure defined, but how many match states?

```
... V G A - - H A G E Y ...
... V - - - - N V D E V ...
... V E A - - D V A G H ...
... V K G - - - - - - D ...
... V Y S - - T Y E T S ...
... F N A - - N I P K H ...
... I A G A D N G A G V ...
```

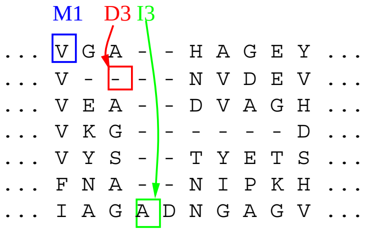
Building a Model
(cont'd)

- Given a multiple alignment, how to build an HMM?
  - General structure defined, but how many match states?
  - Heuristic: if more than half of characters in a column are non-gaps, include a match state for that column

```
... V G A - - H A G E Y ...
... V - - - - N V D E V ...
... V E A - - D V A G H ...
... V K G - - - - - - D ...
... V Y S - - T Y E T S ...
... F N A - - N I P K H ...
... I A G A D N G A G V ...
```

Building a Model
(cont'd)

- Now, find parameters
- Multiple alignment + HMM structure → state sequence



```
... V G A - - H A G E Y ...
... V - - - - N V D E V ...
... V E A - - D V A G H ...
... V K G - - - - - - D ...
... V Y S - - T Y E T S ...
... F N A - - N I P K H ...
... I A G A D N G A G V ...
```

Non-gap in match column -> match state

Gap in match column -> delete state

Non-gap in insert column -> insert state

Gap in insert column -> ignore

Durbin Fig 5.4, p. 109

Building a Model
(cont'd)

- Count number of transitions and emissions and compute:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

- Still need to beware of some counts $= 0$

Weighted Pseudocounts

- Let c_{ja} = observed count of residue a in position j of multiple alignment

$$e_{M_j}(a) = \frac{c_{ja} + Aq_a}{\sum_{a'} c_{ja'} + A}$$

- q_a = background probability of a, A = weight placed on pseudocounts (sometimes use $A \approx 20$)

- Also called a prior distribution

Dirichlet Mixtures

- Can be thought of a mixture of pseudocounts

- The mixture has different components, each representing a different context of a protein sequence
  - E.g. in parts of a sequence folded near protein's surface, more weight (higher q_a) can be given to hydrophilic residues (ones that readily bind with water)

- Will find a different mixture for each position of the alignment based on the distribution of residues in that column

Dirichlet Mixtures
(cont'd)

- Each component k consists of a vector of pseudocounts $\vec{\alpha}^k$ (so α^k_a corresponds to Aq_a) and a mixture coefficient (m_k, for now) that is the probability that component k is selected

- Pseudocount model k is the "correct" one with probability m_k

- We'll set the mixture coefficients for each column based on which vectors best fit the residues in that column
  - E.g. first column of our example alignment is dominated by V, so any vector $\vec{\alpha}^k$ that favors V will get a higher m_k

Dirichlet Mixtures
(cont'd)

- Let $\vec{c}_j$ be vector of counts in column j

$$e_{M_j}(a) = \sum_k P\left(k \mid \vec{c}_j\right) \frac{c_{ja} + \alpha^k_a}{\sum_{a'}\left(c_{ja'} + \alpha^k_{a'}\right)}$$

- $P\left(k \mid \vec{c}_j\right)$ are the posterior mixture coefficients, which are easily computed [Sjölander et al. 1996], yielding:

$$e_{M_j}(a) = \frac{X_a}{\sum_{a'} X_{a'}} \ ,$$

where

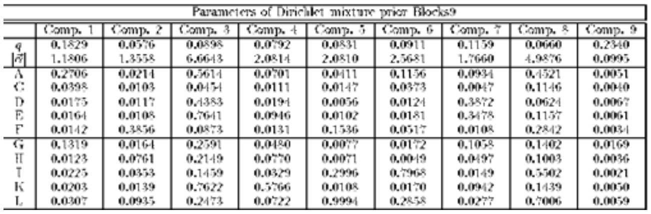
$$X_a = \sum_k m_{k0} \ \mathsf{exp}\left(\mathsf{ln}\, B\left(\vec{\alpha}^k_a + \vec{c}_j\right) - \mathsf{ln}\, B\left(\vec{\alpha}^k_a\right)\right) \frac{c_{ja} + \vec{\alpha}^k_a}{\sum_{a'}\left(c_{ja'} + \alpha^k_{a'}\right)} \ ,$$

$$\mathsf{ln}\, B(\vec{x}) = \sum_i \mathsf{ln}\, \Gamma(x_i) - \mathsf{ln}\, \Gamma\left(\sum_i x_i\right)$$

Dirichlet Mixtures
(cont'd)

- Γ is gamma function, and $\mathsf{ln}\,\Gamma$ is computed via `lgamma` and related functions in C

- m_{k0} is prior probability of component k ($= q$ in Sjölander Table 1):



Parameters of Dirichlet mixture prior Blocks9

	Comp. 1	Comp. 2	Comp. 3	Comp. 4	Comp. 5	Comp. 6	Comp. 7	Comp. 8	Comp. 9
q	0.1829	0.0576	0.0898	0.0792	0.0831	0.0911	0.1159	0.0660	0.2340
$\|\vec{\alpha}\|$	1.1806	1.3558	6.6643	2.0814	2.0810	2.5681	1.7660	4.9876	0.0995
A	0.2706	0.0214	0.5614	0.0701	0.0411	0.1156	0.0934	0.4521	0.0051
C	0.0398	0.0103	0.0454	0.0111	0.0147	0.0373	0.0047	0.1146	0.0040
D	0.0175	0.0117	0.4383	0.0194	0.0056	0.0124	0.3872	0.0624	0.0067
E	0.0164	0.0108	0.7641	0.0946	0.0102	0.0181	0.3478	0.1157	0.0061
F	0.0142	0.3856	0.0873	0.0131	0.1536	0.0517	0.0108	0.2842	0.0034
G	0.1319	0.0164	0.2591	0.0480	0.0077	0.0172	0.1058	0.1402	0.0169
H	0.0123	0.0761	0.2149	0.0770	0.0071	0.0049	0.0497	0.1003	0.0036
I	0.0225	0.0353	0.1459	0.0329	0.2996	0.7968	0.0149	0.5502	0.0021
K	0.0203	0.0139	0.7622	0.5766	0.0108	0.0170	0.0942	0.1439	0.0050
L	0.0307	0.0935	0.2473	0.0722	0.9994	0.2858	0.0277	0.7006	0.0059

⋮

Searching for Homologues

- Score a candidate match x by using log-odds:
  - $P(x, \pi^* \mid M)$ is probability that x came from model M via most likely path π^*
    $\Rightarrow$ Find using Viterbi
  - $Pr(x \mid M)$ is probability that x came from model M summed over all possible paths
    $\Rightarrow$ Find using forward algorithm
  - $score(x) = \log(P(x \mid M)/P(x \mid \phi))$
    - ϕ is a "null model", which is often the distribution of amino acids in the training set or AA distribution over each individual column
    - If x matches M much better than ϕ, then score is large and positive

Viterbi Equations

- $V_j^M(i)$ = log-odds score of best path matching $x_{1\ldots i}$ to the model, where x_i emitted by state M_j (similarly define $V_j^I(i)$ and $V_j^D(i)$)
- Rename $\mathcal{B}$ as M_0, $V_0^M(0) = 0$, rename $\mathcal{E}$ as M_{L+1} (V_{L+1}^M = final)

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

$$V_j^I(i) = \log\left(\frac{e_{I_j}(x_i)}{q_{x_i}}\right) + \max\begin{cases} V_j^M(i-1) + \log a_{M_j I_j} \\ V_j^I(i-1) + \log a_{I_j I_j} \\ V_j^D(i-1) + \log a_{D_j I_j} \end{cases}$$

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

Forward Equations

$$F_j^M(i) = \log\left(\frac{e_{M_j}(x_i)}{q_{x_i}}\right) + \log\Big[a_{M_{j-1}M_j}\exp\left(F_{j-1}^M(i-1)\right) + a_{I_{j-1}M_j}\exp\left(F_{j-1}^I(i-1)\right) + a_{D_{j-1}M_j}\exp\left(F_{j-1}^D(i-1)\right)\Big]$$

$$F_j^I(i) = \log\left(\frac{e_{I_j}(x_i)}{q_{x_i}}\right) + \log\Big[a_{M_j I_j}\exp\left(F_j^M(i-1)\right) + a_{I_j I_j}\exp\left(F_j^I(i-1)\right) + a_{D_j I_j}\exp\left(F_j^D(i-1)\right)\Big]$$

$$F_j^D(i) = \log\Big[a_{M_{j-1}D_j}\exp\left(F_{j-1}^M(i)\right) + a_{I_{j-1}D_j}\exp\left(F_{j-1}^I(i)\right) + a_{D_{j-1}D_j}\exp\left(F_{j-1}^D(i)\right)\Big]$$

- $\exp(\cdot)$ needed to use sums and logs

Aligning a Sequence with a Model (Multiple Alignment)

- Given a string x, use Viterbi to find most likely path π^* and use the state sequence as the alignment
- More detail in Durbin, Section 6.5
  - Also discusses building an initial multiple alignment and HMM simultaneously via Baum-Welch