Genome Annotation

The Viterbi algorithm

Outline

- The most probable path task for HMMs
- The Viterbi algorithm

HMM most probable path task

- Given: *x*
- Do: find a hidden path of states π that maximizes the joint probability of *x* and π

- Mathematically:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, P(x, \pi)$$

- Note that this is equivalent to computing

$$\underset{\pi}{\operatorname{argmax}}\, P(\pi | x)$$

Example

- Consider an candidate CpG island

CGCGC

- Considering our HMM for CpG island model, some possible paths that are consistent with this CpG island are

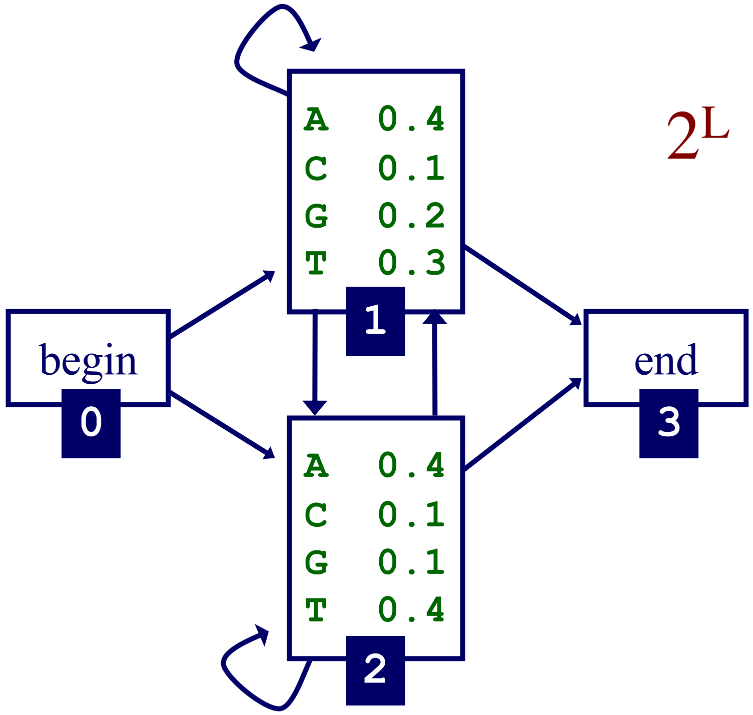
$C^+G^+C^+G^+C^+$

$C^-G^-C^-G^-C^-$

$C^-G^+C^-G^+C^-$

Number of paths

- for a sequence of length *L*, how many possible paths through this HMM are there?



2^L

- the **Viterbi** algorithm enables us to compute the most likely path by efficiently taking a maximum over all possible paths

Finding the Most Probable Path: The Viterbi Algorithm

- Dynamic programming!
- subproblem: define $v_k(i)$ to be the probability of the most probable path accounting for the first i characters of x and ending in state k

$$v_k(i) = \max_{\pi_1, \ldots, \pi_i : \pi_i = k} P(x_1, \ldots, x_i, \pi_1, \ldots, \pi_i)$$

- we want to compute $v_N(L)$, the probability of the most probable path accounting for all of the sequence and ending in the end state
- can define recursively
- can use DP to find $v_N(L)$ efficiently

Derivation of Viterbi recurrence

$$\begin{aligned}
v_\ell(i) &= \max_{\pi_1,\ldots,\pi_i:\pi_i=\ell} P(x_1,\ldots,x_i,\pi_1,\ldots,\pi_i) \\
&= \max_{\pi_1,\ldots,\pi_i:\pi_i=\ell} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1})P(\pi_i|\pi_{i-1})P(x_i|\pi_i) \\
&= \max_k \max_{\pi_1,\ldots,\pi_i:\pi_{i-1}=k,\pi_i=\ell} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1})P(\pi_i|\pi_{i-1})P(x_i|\pi_i) \\
&= \max_k \max_{\pi_1,\ldots,\pi_i:\pi_{i-1}=k,\pi_i=\ell} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1})a_{kl}e_\ell(x_i) \\
&= e_\ell(x_i)\max_k a_{kl} \max_{\pi_1,\ldots,\pi_{i-1}:\pi_{i-1}=k} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1}) \\
&= e_\ell(x_i)\max_k a_{kl}\underline{v_k(i-1)}
\end{aligned}$$

Finding the Most Probable Path: The Viterbi Algorithm

- initialization:

$$v_0(0) = 1 \qquad \text{(0 is the begin state)}$$

$$v_k(0) = 0, \quad \text{for } k \text{ that are not silent states}$$

The Viterbi Algorithm

- recursion for emitting states ($i = 1 \ldots L$):

$$v_l(i) = e_l(x_i) \max_k \left[v_k(i-1) a_{kl} \right]$$

$$\mathrm{ptr}_l(i) = \arg\max_k \left[v_k(i-1) a_{kl} \right]$$

keep track of most probable path

- recursion for silent states:

$$v_l(i) = \max_k \left[v_k(i) a_{kl} \right]$$

$$\mathrm{ptr}_l(i) = \arg\max_k \left[v_k(i) a_{kl} \right]$$

The Viterbi Algorithm

- termination:

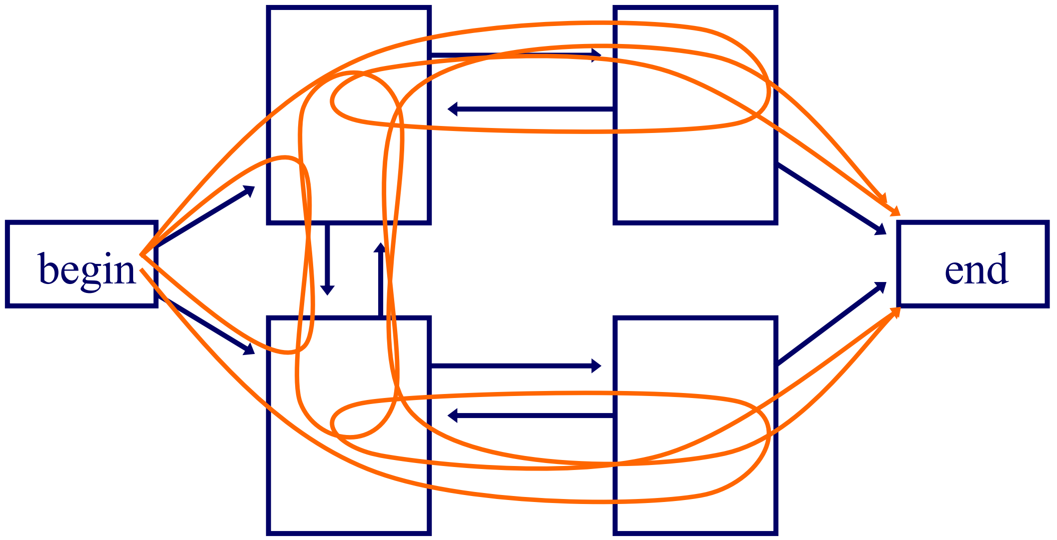
$$\Pr(x,\pi^*) = \underline{\max_k \left(v_k(L)a_{kN}\right)}$$

$$\pi_L^* = \arg\max_k \left(v_k(L)a_{kN}\right)$$

- traceback: follow pointers back starting at π_L^*

Implicitly considering all possible paths

- The Viterbi algorithm effectively considers all possible paths for a sequence
- consider a sequence of length 4…



Numerically stable Viterbi

- Use log probabilities instead

$$\underline{V_k(i) = \log v_k(i)} \qquad \underline{\tilde{e}_k(c) = \log e_k(c)} \qquad \underline{\tilde{a}_{k\ell} = \log a_{k\ell}}$$

- Initialization

$$\underline{V_0(0) = 0} = \log(1)$$

$$V_\ell(0) = -\infty \xleftarrow{\log(0)} \text{ for all other (non-silent) states}$$

- Recurrence

$$V_\ell(i) = \tilde{e}_\ell(x_i) + \max_k \left(V_k(i-1) + \tilde{a}_{k\ell} \right)$$

Using HMM to detect CpG islands

- Recall the 8-state HMM for our CpG island
- Apply the Viterbi algorithm to a DNA sequence on this HMM
- Contiguous assignments of '+' states will correspond to CpG islands

Summary

- Viterbi algorithm efficiently solves the most probable path problem
- Viterbi algorithm is a dynamic programming algorithm
  - subproblem: $v_k(i)$: probability of most probable path for a prefix (i) of the sequence that ends in a given state (k)
- Log-transformed equations are needed for numerical stability when computing on long sequences