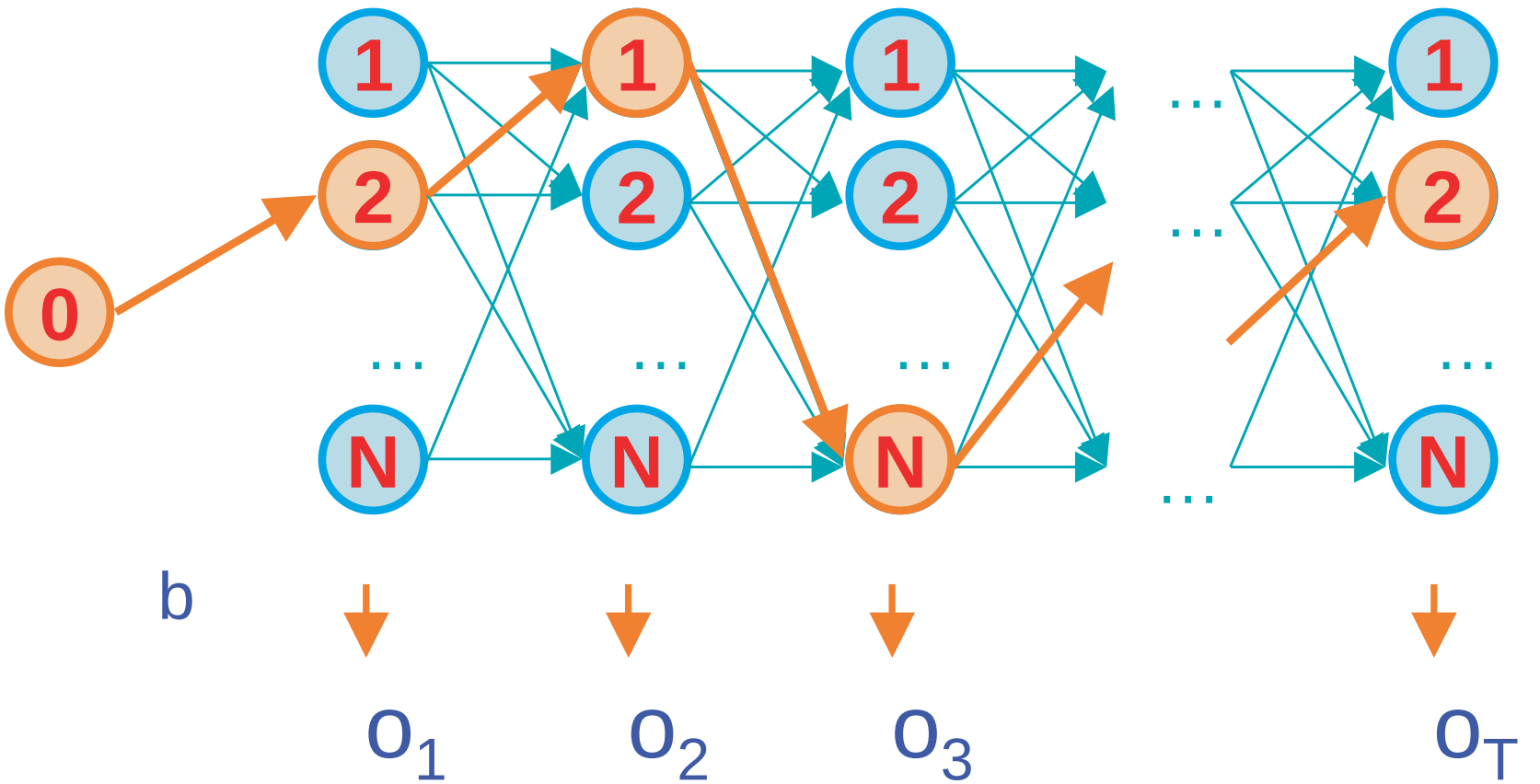
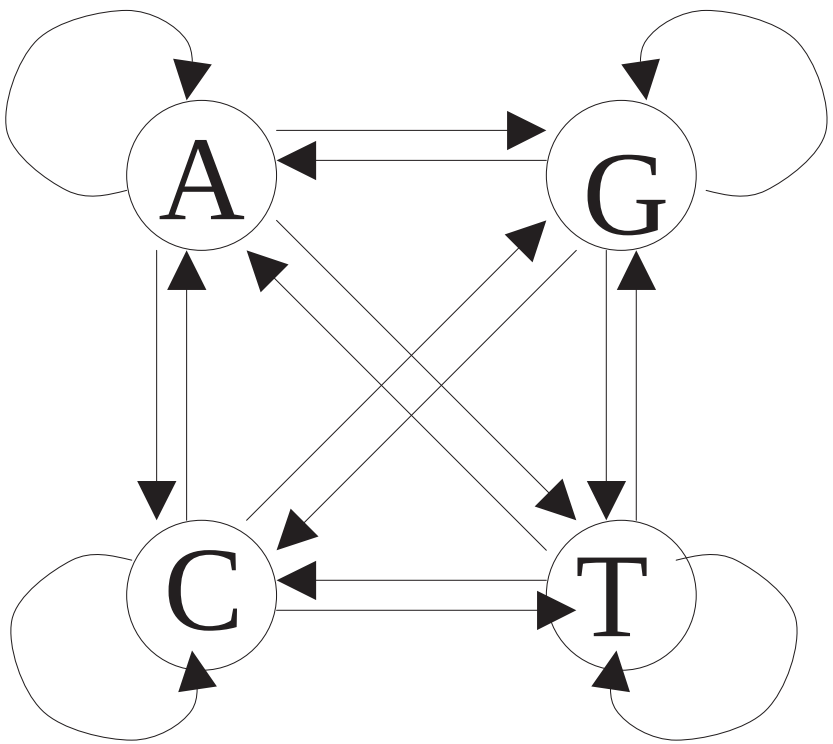
Hidden Markov Models for biological systems



- We would like to identify stretches of sequences that are actually functional (code for proteins or have regulatory functions) from non-coding or junk sequences.
- In prokaryotic DNA we have only two kinds of regions, ignore regulatory sequences which are coding (+) and non-coding (-) and the four letters A,C,G,T.



- This simulates a very common phenomenon:

  there is some underlying dynamic system running along according to simple and uncertain dynamics, but we cannot see it.

- All we can see are some noisy signals arising from the underlying system. From those noisy observations we want to do things like predict the most likely underlying system state, or the time history of states, or the likelihood of the next observation

What are Hidden Markov Models good for?

- useful for modeling protein/DNA sequence patterns
- probabilistic state-transition diagrams
- Markov processes - independence from history
- hidden states

History

- voice compression, information theory
- speech = string of phonemes
- probability of what follows what

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77:257-286.

Where does one use HMM's

- protein families
- DNA patterns
- secondary structure (helix, strand, coil (each has 20x20 table with transition frequencies between neighbors $a_i \rightarrow a_{i+1}$))
- protein fold recognition
- fold classification
- gene silencing
- ...

- **Example: CpG Islands**

  - Regions labeled as CpG islands $\longrightarrow$ + model
  - Regions labeled as non-CpG islands $\longrightarrow$ - model
  - Maximum likelihood estimators for the transition probabilities for each model

    $$a_{st} = \frac{c_{st}}{\sum_{t'} c_{st'}}$$

    and analogously for the - model. c_{st} is the number of times letter t followed letter s in the labeled region

- A Hidden Markov Model is a two random variable process, in which one of the random variables is hidden, and the other random variable is observable.
- It has a finite set of states, each of which is associated with a probability distribution.
- Transitions among the states are governed by transition probabilities.
- In a particular state an observation can be generated, according to the associated probability distribution.
- It is only the observation, not the state visible to an external observer, and therefore states are "hidden" from the observer.

- **Example:**
  - For DNA, let + denote coding and - non-coding. Then a possible observed sequence could be

$$O = AACCTTCCGCGCAATATAGGTAACCCCGG$$

and

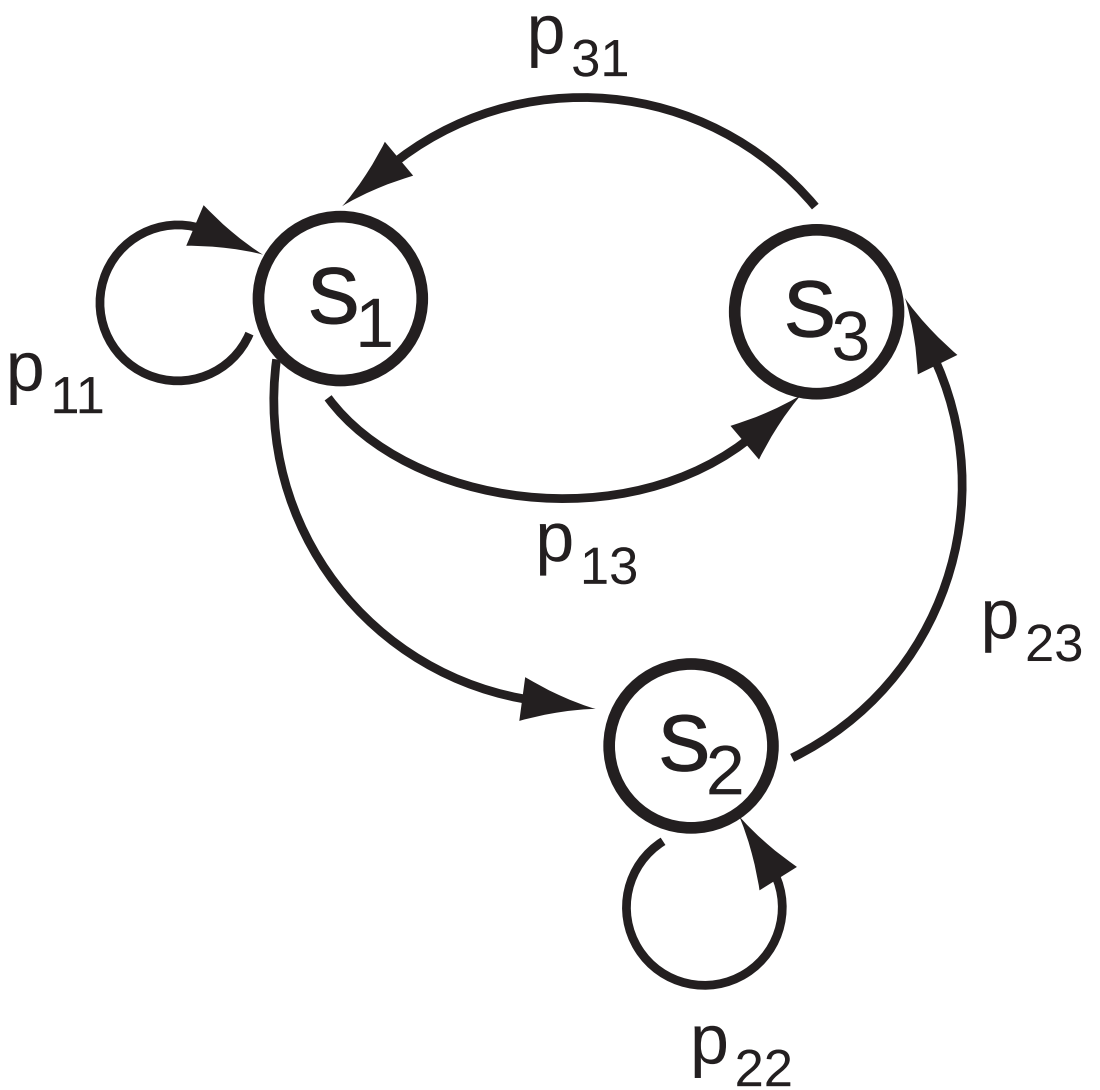
$$Q = --+++++++++++++++++++---------$$

- Question: How can one find CpG - islands in a long chain of nucleotides?
- Merge both models into one model with small transition probabilities between the chains.

A Hidden Markov Model (HMM) $\lambda =< Y, S, A, B >$ consists of:

- an output alphabet $Y = \{1, ..., b\}$
- a state space $S = \{1, ..., c\}$ with a unique initial state s_0
- a transition probability distribution $A(s'|s)$
- an emission probability distribution $B(y|s, s')$

- HMMs are equivalent to (weighted) finite state automata with outputs on transitions
- Unlike MMs, constructing HMMs, estimating their parameters and computing probabilities are not so straightforward


p31
s1
s3
p11
p13
p23
s2
p22

Given a HMM λ and a state sequence $S = (s_1, ..., s_{t+1})$, the probability of an output sequence $O = (o_1, ..., o_t)$ is

$$P(O|S, \lambda) = \prod_{i=1}^{t} P(o_i|s_i, s_{i+1}, \lambda) = \prod_{i=1}^{t} B(o_i|s_i, s_{i+1}). \tag{1}$$

Given λ, the probability of a state sequence $S = (s_1, ..., s_{t+1})$ is

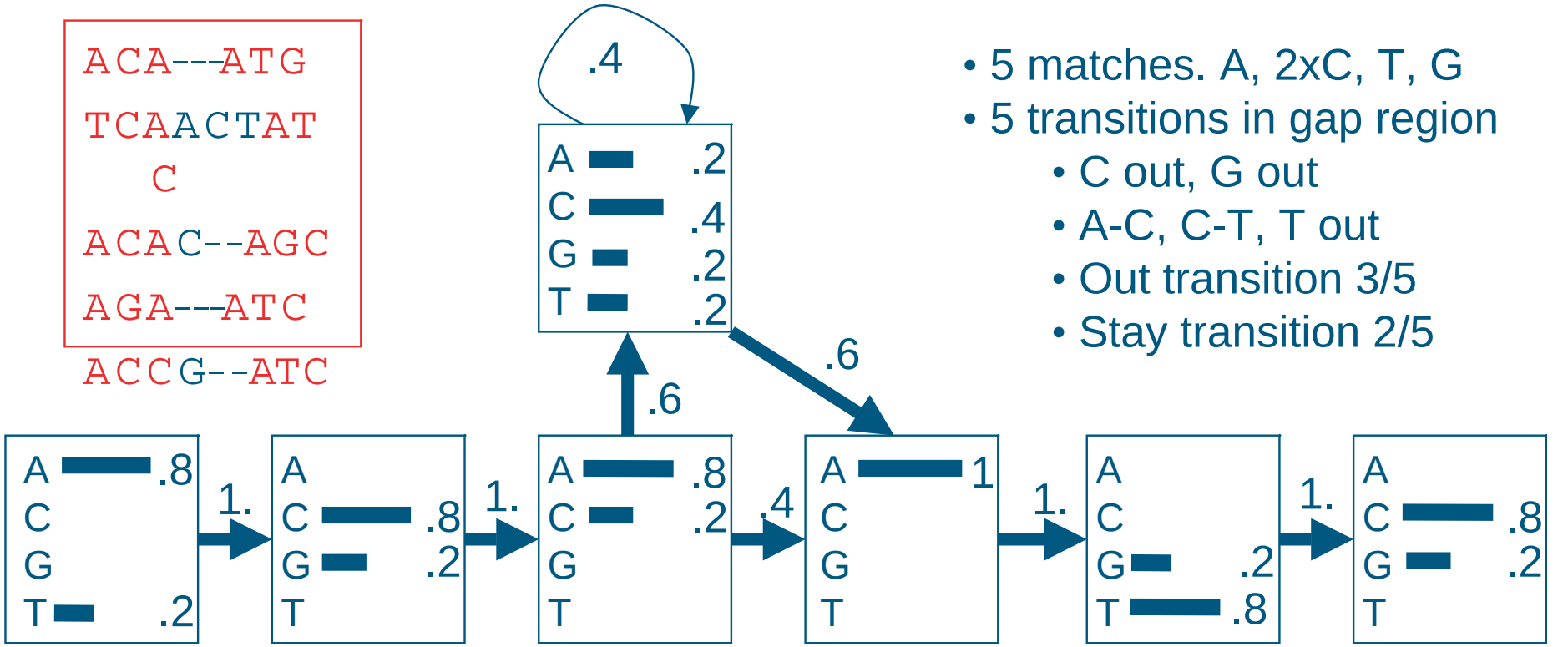
$$P(S|\lambda) = \prod_{i=1}^{t} P(s_{i+1}|s_i) = \prod_{i=1}^{t} A(s_{i+1}|s_i). \tag{2}$$

Of importance is the probability of an output sequence $O = (o_1, ..., o_t)$ under a given λ. It is easy to show that the straightforward computation yields

$$P(O|\lambda) = \sum_{S} \prod_{i=1}^{t} A(s_{i+1}|s_i) B(o_i|s_i, s_{i+1}) \tag{3}$$

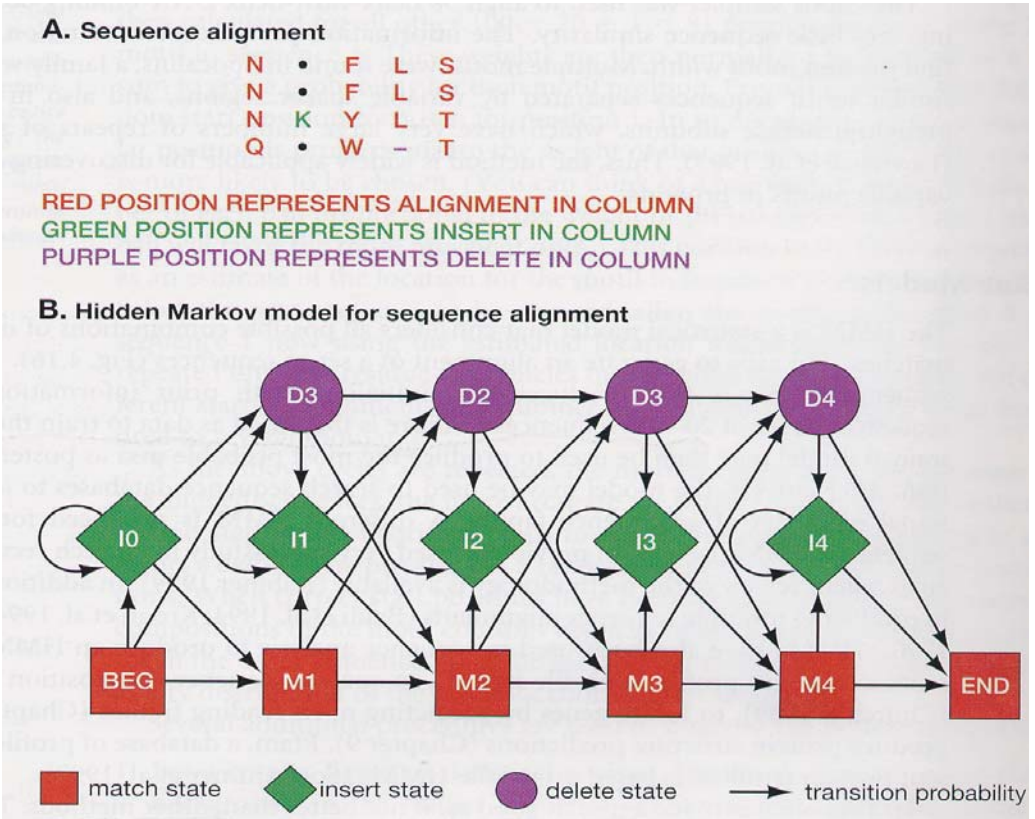
with a computational complexity of $(2c + 1) * c^{t+1}$ multiplications.

- Multiple Sequence Alignments
  - In theory, making an optimal alignment between two sequences is computationally straightforward (Smith-Waterman algorithm), but aligning a large number of sequences using the same method is almost impossible (e.g. $O(t^N)$).
  - The problem increases exponentially with the number of sequences involved (the product of the sequence lengths).
  - Statistical Methods:
    * Expectation Maximization Algorithm (deterministic)
    * Gibbs Sampler (stochastic)
    * Hidden Markov Models (stochastic)
  - Advantages for HMM: theoretical explanation, no sequence ordering, no insertion and deletion penalties, using prior information
  - Disadvantage for HMM: large number of sequences for training



- 5 matches. A, 2xC, T, G
- 5 transitions in gap region
  - C out, G out
  - A-C, C-T, T out
  - Out transition 3/5
  - Stay transition 2/5

ACA---ATG 0.8x1x0.8x1x0.8x0.4x1x0.8x1x0.2 = 3.3×10^{-2}



By Forward-backward, Viterbi, Baum-Welch (or EM) algorithms, (1) find Prob. of the observation sequence given the model; (2) find the most likely hidden state sequence given the model and observations; (3) adjust parameters to maximize their Prob. given observations.

- There are three basic problems:
  1. Given a model, how likely is a specific sequence of observed values (evaluation problem).
  2. Given a model and a sequence of observations, what is the most likely state sequence in the model that produces the observations (decoding problem).
  3. Given a model and a set of observations, how should the model's parameters be updated so that it has a high probability of generating the observations (learning problem).

Forward algorithm

- We define $\alpha_s(i)$ as the probability being in state s at position i:

$$\alpha_s(i) = P(o_1, ..., o_i, s_i = s|\lambda) \tag{4}$$

- Base case: $\alpha_s(1)$ if $s = s_0$ and $\alpha_s(0) = 0$ otherwise
- Induction:

$$\alpha_s(i+1) = \max_{s \in S} A(s|s')B(o_i|s', s)\alpha_s(i) \tag{5}$$

- Finally, at the end:

$$P(o_1, ..., o_k|\lambda) = \sum_{s \in S} \alpha_s(k) \tag{6}$$

Forward algorithm

- Partial sums could as well be computed right to left (backward algorithm), or from the middle out
- In general, for any position i:

$$P(O|\lambda) = \sum_{s \in S} \alpha_s(i)\beta_s(i) \tag{7}$$

- This algorithm could be used, e.g. to identify which λ is most likely to have produced an output sequence O

What is the most probable path given observations (decoding problem)?

- given $o_1, ..., o_t$ what is

$$\text{argmax}_S P(s, o_1, ...o_t|\lambda)$$

- slow and stupid answer:

$$\text{argmax}_S \frac{P(o_1, ..., o_t|s)P(s)}{P(o_1, ..., o_t)}$$

Viterbi algorithm

- We define $\delta_s(i)$ as the probability of the most likely path leading to state s at position i:

$$\delta_s(i) = \max_{s_1,...,s_{i-1}} P(s_1, ..., s_{i-1}, o_1, ..., o_{i-1}, s_i = s|M) \quad (8)$$

- Base case: $\delta_s(1)$ if $s = s_0$ and $\delta_s(0) = 0$ otherwise

Viterbi algorithm

- Again we proceed recursively:

$$\delta_s(i+1) = \max_{s \in S} A(s|s')B(o_i|s', s)\delta_s(i) \tag{9}$$

  and since we want to know the identity of the best state sequence and not just its probability, we also need

$$\Psi(i+1) = \text{argmax}_{s \in S} A(s|s')B(o_i|s', s)\delta_s(i) \tag{10}$$

- Finally, we can follow Ψ backwards from the most likely final state

Viterbi algorithm

- The Viterbi algorithm efficiently searches through $|S|^T$ paths for the one with the highest probability in $O(T|S|^2)$ time
- In practical applications, use log probabilities to avoid underflow errors
- Can be easily modified to produce the n best paths
- A beam search can be used to prune the search space further when $|S|$ is very large (n-gram models)

n-gram models

- Predicting the next state s_n depending on $s_1, ..., s_{n-1}$ results in

$$P(s_n | s_1, ..., s_{n-1})$$

- Markov Assumption $(n-1)$th order : last $n-1$ states are in the same equiv. class

Parameter estimation

- Given an HMM with a fixed architecture, how do we estimate the probability distributions A and B?
- If we have labeled training data, this is not any harder than estimating non-Hidden Markov Models (supervised training):

$$A(s'|s) = \frac{C(s \rightarrow s')}{\sum_{s''} C(s \rightarrow s'')} \tag{11}$$

$$B(o|s, s') = \frac{C(s \rightarrow s', o)}{C(s \rightarrow s')} \tag{12}$$

Forward-Backward algorithm

- Also known as the *Baum-Welch algorithm*
- Instance of the *Expectation Maximization (EM) algorithm*:
  1. Choose a model at random
  2. E: Find the distribution of state sequences given the model
  3. M: Find the most likely model given those state sequences
  4. Go back to 2.

Forward-Backward algorithm

- Our estimate of A is:

$$A(s'|s) = \frac{E[C(s \rightarrow s')]}{E[C(s \rightarrow ?)]} \tag{13}$$

- We estimate $E[C(s \rightarrow s')]$ via $\tau_t(s, s')$, the probability of moving from state s to state s' at position t given the output sequence O:

$$\begin{aligned}
\tau_t(s, s') &= P(s_t = s, s_{t+1} = s'|O, \lambda) && (14)\\
&= \frac{P(s_t = s, s_{t+1} = s', O|\lambda)}{P(O|\lambda)} && (15)\\
&= \frac{\alpha_s(t) A(s|s') B(o_{t+1}|s, s') \beta_{s'}(t+1)}{\sum_{s''} \alpha_{s''}} && (16)
\end{aligned}$$

Forward-Backward algorithm

- This lets us estimate A:

$$A(s'|s) = \frac{\sum_t \tau_t(s, s')}{\sum_t \sum_{s''} \tau_t(s, s'')} \tag{17}$$

- We can estimate B along the same lines, using $\sigma_t(o, s, s')$, the probability of emitting o while moving from state s to state s' at position t given the output sequence O
- Alternate re-estimating A from τ, then τ from A, until estimates stop changing
- If the initial guess is close to the right solution, this will converge to an optimal solution

- A. Krogh, M. Brown, I. S. Mian, K. Sjander, and D. Haussler, Hidden Markov models in computational biology: Applications to protein modeling. Journal of Molecular Biology, 235, 1501-1531 (1994)
- R. Durbin, S.R. Eddy, A. Krogh and G. Mitchison Biological sequence analysis, Cambridge University Press, 1998