Roadmap

- Probabilistic models (i.i.d.)
- ML and MAP parameters inference
- Probabilistic profiles (matrix profiles)
- Markov models
- Hidden Markov models
- From alignments to HMMs
Probabilistic models

• A model is a mathematical formulation of a hypothesis about the phenomenon under study
• The parameters $\theta$ of the model can be
  • probabilities associated with (groups of) symbols of the alphabet $\Sigma$
  • any other entity which completely characterizes the model
• Examples: i.i.d. model, Markov model
• While the true parameters $\theta_0$ are unknown, we might have a probabilistic model for them (prior)

Inference

• We observe a string generated by the unknown model $\theta_0$, say $x$="GTAAACTAGGC..."
• Statistical questions
  • How do we choose a “good” model?
  • How do we infer “good” estimates of $\theta_0$?
  • How much data do we need?
Inference

• Probabilities and/or other parameters of the model are typically estimated from a training set
• The training set can be composed positive and negative examples
• Limited training data can lead to overfitting
• An estimate $\theta$ is consistent, if, when $|x| \to \infty$, we have $\theta \to \theta_0$

Maximum likelihood

• Given training data $x$, a model $M$ with parameters $\theta$, the maximum likelihood estimate for $\theta$ is

$$\theta_{ML} = \arg\max_\theta P(x \mid M, \theta)$$

where $P(x \mid M, \theta)$ is the probability that the dataset $x$ has been generated by the model $M$ with parameters $\theta$

• ML is consistent, but can give poor results with limited data
Example (DNA)

- $x_1 = \text{CCACCCTTTTGTGGGCTTCTATTTCAAGG}$
- $x_2 = \text{TTGGTTCCTGCATGTGCGCGCAGTGC}$
- $x_3 = \text{TTCTAAAAGGGCATTATCAGAAAAAAG}$
- $x_4 = \text{GTGTAATTTGGTGTGCTACCTACCGTATTA}$
- $\Sigma = \{A, C, G, T\}$
- Maximum likelihood estimate for the parameter set $\{p_A, p_C, p_G, p_T\}$ for the Bernoulli (i.i.d.) model is
  - $p_A = f(A)/n = 28/120 = 0.233$
  - $p_C = 24/120 = 0.2$
  - $p_G = 28/120 = 0.233$
  - $p_T = 40/120 = 0.333$

Additional Issues

- How can we incorporate prior knowledge in the analysis process?
- How can we compare different models?
- Harder: How much data we need to be confident in our estimate?
Bayes Theorem

- Gives the posterior probability of an event

\[ P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} \]

- Useful to get better estimates of the parameters using prior knowledge
- Useful to compare different models and select the best

Example (alternative iid sources)

- Suppose we have two dice with 4 faces \{A, T, C, G\}

\[ D_1 = \{p_A=0.25, p_C=0.25, p_G=0.25, p_T=0.25\} \]
\[ D_2 = \{p_A=0.20, p_C=0.28, p_G=0.30, p_T=0.22\} \]
Example (alternative iid sources)

• The source generates a string as follows
  1. select randomly one die \( [P(D_1) = P(D_2) = 1/2] \)
  2. roll it, append the symbol to \( x \)
  3. repeat 2. until all symbols have been generated

• We observe a string generated by the model, say
  \( x = \text{“GTAAACTAC...”} \)

• What is the probability that \( x \) has been generated by, say, probability distribution \( D_1 \)?

\[
P(D_1 | x) = \frac{P(x | D_1)P(D_1)}{P(x)}
\]

where

\[
P(x) = P(x | D_1)P(D_1) + P(x | D_2)P(D_2)
\]
Bayesian Inference

- When parameters are in the continuous domain
\[ P(\theta \mid x) = \frac{P(x \mid \theta)P(\theta)}{P(x)} = \frac{P(x \mid \theta)P(\theta)}{\int P(x \mid \sigma)P(\sigma) \, d\sigma} \]
- \( P(\theta) \) represents any prior knowledge we have regarding the parameters
- \( P(\theta \mid x) \) represents the probability that \( \theta \) is correct given that we have observed data \( x \)

Bayesian Inference

- Given two alternative models \( \theta_1 \) and \( \theta_2 \), we can compare \( P(\theta_1 \mid x) \) to \( P(\theta_2 \mid x) \)
- We can define the “best” model, the one that maximizes \( P(\theta \mid x) \) (maximum a posteriori estimation, or MAP)
- Equivalent to minimizing
\[ -\log P(\theta \mid x) = -\log P(x \mid \theta) - \log P(\theta) + \log P(x) \]
- Note that \([\log P(x)]\) can be regarded as a constant
Bayesian Inference

• ML estimation

\[ \theta^{ML} = \arg \min_{\theta} \{-\log P(x | \theta)\} \]

• MAP estimation

\[ \theta^{MAP} = \arg \min_{\theta} \{-\log P(x | \theta) - \log P(\theta)\} \]
Profiles

- *Position weight matrices, or matrix profiles, are $|\Sigma| \times m$ matrices containing real numbers in the interval $[0,1]$, such that each column sums to 1,* e.g.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.26</td>
<td>0.22</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>B</td>
<td>0.17</td>
<td>0.18</td>
<td>0.59</td>
<td>0.00</td>
</tr>
<tr>
<td>C</td>
<td>0.09</td>
<td>0.15</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>D</td>
<td>0.48</td>
<td>0.45</td>
<td>0.41</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- Notion of consensus

Profiles

- To measure the distance between profiles one can use the relative entropy
- For example, one can measure the distance between a profile $P$, and the profile $Q=\{1/4\}$
- The higher the distance from the “null-profile” $Q$, the more informative is $P$
Entropy

The entropy $H(p)$ of a discrete probability distribution $p = \{p_1, p_2, \ldots, p_k\}$ is defined by

$$H(p) = E(-\log p) = -\sum_{i=1}^{k} p_i \log p_i$$

If the base of the log is 2, $H(p)$ is measured in bits. It reaches the maximum when $p_i = 1/k$ for all $i$.

Sequence Logos

A sequence logo showing the most conserved bases around the start codon from a set of human transcripts. The $y$-axis represents the information content, which is computed from the entropy: the taller is a nucleotide, the more informative it is.
Relative Entropy

The relative entropy $H(p \parallel q)$ between two discrete probability distributions $p = \{p_1, \ldots, p_k\}$ and $q = \{q_1, \ldots, q_k\}$ is defined by

$$H(p \parallel q) = \sum_{i=1}^{k} p_i \log \frac{p_i}{q_i}$$

Also called cross-entropy or Kullback-Liebler distance. It is easy to verify that $H(p, q) \geq 0$ with equality iff $p = q$.

Relative Entropy

Proof: First note that $\log x \leq x - 1$ for all real numbers $x$ with equality iff $x = 1$.

$$H(p \parallel q) = \sum_{i=1}^{k} p_i \log \frac{p_i}{q_i} \geq \sum_{i=1}^{k} p_i \left(1 - \frac{q_i}{p_i}\right) =$$

$$= \sum_{i=1}^{k} (p_i - q_i) = \sum_{i=1}^{k} p_i - \sum_{i=1}^{k} q_i = 1 - 1 = 0$$

Note that $H(p \parallel q) = 0$ iff $\frac{q_i}{p_i} = 1$ for all $i$. 
Example: CRP binding sites

- $S^+=\{TTGTGGC, ACGTGAT, ATTTATT, GTGTGAA, CTGTGAC, TTTTGAT, ATGTGAG, ATGAGAC, TTGTGAT, TTGTGAT, AAGTGTC, TTGTGAG, ATTTGAA, TTGTGAT, TTGTGAC, ATTTGCA, CTGTAAC, TTGTGAG, GTGTTAA, GCCTGAC, CTGTGCG, CTGTAAC, ATGCAAA\}$

Cyclic AMP receptor protein binding sites in *E.coli.* [Stormo & Hartzell, 89]

Training (CRP sites)

- Assume a Bernoulli model for each position

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.350</td>
<td>0.043</td>
<td>0.000</td>
<td>0.043</td>
</tr>
<tr>
<td>C</td>
<td>0.170</td>
<td>0.087</td>
<td>0.043</td>
<td>0.043</td>
</tr>
<tr>
<td>G</td>
<td>0.130</td>
<td>0.000</td>
<td>0.780</td>
<td>0.000</td>
</tr>
<tr>
<td>T</td>
<td>0.350</td>
<td>0.870</td>
<td>0.170</td>
<td>0.910</td>
</tr>
</tbody>
</table>

- Assume the uniform Bernoulli model for the non-sites $S^-$, that is $p_A=0.25$, $p_C=0.25$, $p_T=0.25$, $p_G=0.25$ for all the positions
Testing

• Suppose you have $x = \text{TGTGAC}$
  Is $x$ more likely to belong to $S+$ or to $S$-?
  I.e., is $x$ more likely to be generated from the Bernoulli model for $S+$ or from the Bernoulli model for $S$-?
  
• Let’s compute the probability

\[
P(x = \text{TGTGAC} | S+) = .35 \times .87 \times .78 \times .91 \times .83 \times .83 \times .3 = 0.045
\]
\[
P(x = \text{TGTGAC} | S-) = (.25)^7 = 0.0000061
\]

Likelihood ratio

Definition:

\[
LR(x) = \frac{P(x | S+)}{P(x | S-)}. 
\]

In the previous example

\[
LR(\text{TGTGAC}) = 732
\]

Idea: To test a sequence $x$, compare $LR(x)$ to a specified "threshold" $T$, and declare $x$ to be a site (i.e., positive) if $LR(x) > T$
Log-likelihood ratio

Definition:

\[ LLR(x) = \log_2 \frac{P(x \mid S+)}{P(x \mid S-)} \]

Tip: it is convenient to store the matrix profiles in log form, so that the one just add the entries (instead of multiplying) and we avoid underflow errors.

About the distribution for S-

• When we have no negative training set, one choice is to assume the uniform distribution
• A better choice is to estimate the background distribution from the entire genome, or large portion of it
• A uniform distribution would be OK for *E.coli*, but not for the malaria parasite *P. falciparum* which has \( p_A + p_T = 0.81 \)
**Information content of profiles**

- Remember that the relative entropy is zero only when $P$ and $Q$ are the same distribution
- In general, the higher the relative entropy between $P$ and $Q$, the better
- If one assume independence, the total relative entropy is the sum of the relative entropy for each position

**Information content for CRP**

- The information content of our previous example is shown below (in bits)

| 0.12 | 1.3 | 1.1 | 1.5 | 1.2 | 1.1 | 0.027 |

- It is interesting to go back to the original profiles and see why the positions in the middle are the most informative
A Markov model is defined by ...

- the order $h > 0$
- the states, which correspond to strings in $\Sigma^h$
- the stationary probabilities $P(x)$, which are the probabilities of the states in the limit
- the transition probabilities $a_{y,c}$, which represent the probabilities of observing/generating symbol $c$ having seen/generated the context $y$ in $\Sigma^h$
First order Markov model

- Let $y$ be a string of length $m$

$$P(y) = P(y[m]y[m-1]...y[1])
= P(y[m] | y[m-1]...y[1])P(y[m-1] | y[m-2]...y[1])...P(y[2] | y[1])P(y[1])
= P(y[m] | y[m-1])P(y[m-1] | y[m-2])...P(y[2] | y[1])P(y[1])
= \prod_{i=2}^{m} a_{y[i-1],y[i]} P(y[i])
= P(y[1]) \prod_{i=2}^{m} a_{y[i-1],y[i]}$$

Each arc has associated a transition probability. Outgoing arcs sum to 1.
First order Markov model

- A first order MM can be described by a $|\Sigma|+1 \times |\Sigma|$ transition matrix, as in the following example

\[
\begin{array}{cccc}
| & A & C & T & G \\
\hline
b & 0.25 & 0.25 & 0.25 & 0.25 \\
A & 0.05 & 0.40 & 0.30 & 0.25 \\
C & 0.19 & 0.06 & 0.29 & 0.46 \\
G & 0.25 & 0.26 & 0.32 & 0.17 \\
T & 0.33 & 0.31 & 0.05 & 0.31 \\
\end{array}
\]

Example: CpG islands

- CpG islands are stretches of CG dinucleotides repeating over and over
- Often occur adjacent to gene-rich areas, forming a barrier between the genes and “junk” DNA
- CpG island are believed to help regulate gene activity (via DNA methylation)
Example: CpG islands

- Training set
  - S+: 48 sequences putatively labeled as CpG islands
  - S-: other sequences from the rest of the genome
  - Total 60Kbases

- Estimate the transition probabilities for two Markov models of order one (ML estimates)

\[
a^+_{s,t} = \frac{f^+(st)}{\sum_{t' \in \Sigma} f^+(st')}
\]

\[
a^-_{s,t} = \frac{f^-(st)}{\sum_{t' \in \Sigma} f^-(st')}
\]
Example: CpG islands

- How do we use the two models?
- Given a string $y$ we compute the log likelihood ratio

$$S(y) = \log \frac{P(y | S^+)}{P(y | S^-)} = \sum_{i=2}^{m} \log \frac{a^+_y y_{i-1} y_i}{a^-_{y_{i-1} y_i}}$$

- If $S(y) > 0$ then $y$ is likely to be CpG island

Example: CpG islands

- How can we find a CpG island in a long string?
- We could use a sliding window, but how to choose the size of window?
- Can we combine the two models in one?
- The answer to all these questions is to use an HMM
Hidden Markov Models

- HMMs were introduced in the ‘70s for speech recognition
- HMMs have shown to be good models for biological sequences
- In Computational Biology, they are used mainly
  - for searching databases & alignment
  - for sequence analysis (classification)
Example: CpG islands

- Now we have two states for each symbol, for example “A” can be recognized/generated both by $A^+$ or $A^-$
- Within each group of states (+ or -), each group behave as the original MM model
- There is also a (small) probability to switch to the other state
- Since we can expect CpG island smaller than the “sea” then $P(S^-|S^+) > P(S^+|S^-)$
HMM state paths

• It is no longer possible to tell what state the system is by looking at the symbol
• We use $\pi$ to denote be the sequence of states (we use integers to denote the states)
• The chain of states follow the transition probabilities
  \[ a_{s,t} = P(\pi_i=t \mid \pi_{i-1}=s) \]
• As before, we introduce the probabilities $a_{0,l}$ from the start state and $a_{s,0}$ to the end state

HMM

• We can formalize the decoupling of symbol from states by introducing emission probabilities
• Definition: a state emits a symbol based on the following probability distribution
  \[ e_s(b) = P(y_{[i]}=b \mid \pi_i=s) \]
• In the CpG island examples, these probabilities are all 0/1
CpG islands with emission probabilities

Mixing source revisited

- Suppose we have a set of dice with four faces \{A, T, C, G\}
- 90% of them have distribution
  \{p_A=0.25, p_C=0.25, p_G=0.25, p_T=0.25\}
- 10% of them have distribution
  \{p_A=0.20, p_C=0.28, p_G=0.30, p_T=0.22\}
- Can we model this generative process with an HMM?
Mixing source revisited

HMMs

- Why “hidden”? Because if we use the previous HMM as a generative model for a string, for example GCAGTCGATA…, the states are kept hidden.
- How do we generate a string from an HMM?
  - Choose a new state based on the transition probabilities.
  - Choose a symbol based on the emission probabilities.
**HMMs**

- Given a string $y$ of length $m$ generated by a HMM using the path $\pi$, we have
  $$P(y, \pi) = a_{0, \pi_1} \prod_{i=1}^{m} e_{\pi_i}(y_{[i]}) \; a_{\pi_i, \pi_{i+1}}$$
  where we require $\pi_{m+1} = 0$, $\pi_0 = 0$
- For example, the sequence $\text{C+G-C-G+}$ for the CpG island example has probability $a_{\text{C}, \text{C+}} \cdot a_{\text{C+}, \text{G}} \cdot a_{\text{G}, \text{C-}} \cdot a_{\text{C-}, \text{G+}} \cdot a_{\text{G+}, 0}$ because the emission probabilities are 1
- In general, however, we do not know the path

**Most probable path (decoding)**

- **Problem**: Given a string $y$ generated by a given HMM find the most probable state path, that is
  $$\pi^* = \arg \max_{\pi} P(y, \pi)$$
- $\pi^*$ can be found recursively, using Viterbi algorithm

(next)
**Viterbi algorithm**

- Suppose the probability $v_s(i)$ of the most probable path ending in state $s$ at position $i$ for $y$ is known for all the states $s$, then we can compute $v_t(i+1)$ as follows

  $$v_t(i+1) = e_t(y_{i+1}) \max_s \{v_s(i) \cdot a_{s,t}\}$$

- Use dynamic programming to tabulate the values of matrix $v$
- Numerical instabilities can be solved using logarithms

---

**Viterbi dynamic programming table**

<table>
<thead>
<tr>
<th></th>
<th>$1$</th>
<th>$i$</th>
<th>$i+1$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>$v_{s_1}(i)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s$</td>
<td></td>
<td>$a_{i+1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>$v_t(i)$</td>
<td></td>
<td>$v_t(i+1)$</td>
<td></td>
</tr>
<tr>
<td>$s_k$</td>
<td></td>
<td>$a_{k+1}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$e_t(y_{i+1})$ represents the emission probability of observing $y_{i+1}$ in state $t$. $a_{s,t}$ represents the transition probability from state $s$ to state $t$. The dynamic programming table helps to compute the Viterbi path by filling in the probabilities at each state and position.
**Viterbi algorithm via dynamic programming**

- **Initialization**
  \[ v_0(0) = 1, \quad v_s(0) = 0 \quad \text{for} \quad s > 0 \]

- **Recursive step**
  \[ v_t(i) = e_t(y_{[i]}) \max_s \{ v_s(i-1) a_{s,t} \} \]
  remember the argmax with a back pointer

- **Termination**
  \[ P(y, \pi^*) = \max_s \{ v_s(m) a_{s,0} \} \]
  find \( \pi^* \) by traceback

**Computing \( P(y) \)**

- **Problem:** Given a string \( y \) and a hidden Markov model \( M \)
  find the probability that \( y \) has been generated by \( M \)

- **Recall the expression for first order MM**
  \[ P(y) = P(y_{[1]}) \prod_{i=2}^{m} a_{y_{[i-1]}, y_{[i]}} \]

- **What about HMMs, when we don’t know the path?**
Computing $P(y)$

• Because many state paths can generate to the same sequence, we must add them all

$$P(y) = \sum_{\pi} P(y, \pi) = \sum_{\pi} \left( a_{0,\pi_1} \prod_{i=1}^{m} e_{\pi_i}(y_{[i]}) a_{\pi_i,\pi_{i+1}} \right)$$

Computing $P(y)$ for an HMM

• How to compute $P(y)$?
• The number of paths can be exponential in the length of the string
• One approximation is to use only the most probable path $\pi^*$
• Alternatively, one can use the forward algorithm, which is a dynamic programming algorithm like Viterbi
“Forward” algorithm

• Suppose the probabilities $f_s(i) = P(y_{[1,i]}, \pi_i=s)$ are known for all states $s$, then we can compute $f_{t(i+1)}$ as follows
  $$f_{t(i+1)} = e_t(y_{[i+1]}) \sum_s \{ f_s(i) a_{s,t} \}$$

• Again, use dynamic programming to tabulate the matrix $f$

• Use log space

“Backward” algorithm

• Later we will need the probability of producing the entire string $y$ with the $i$-th symbol of $y$ is emitted moving through state $s$, that is
  $$P(y, \pi_i=s)$$
"Backward" algorithm

- We have \( P(y, \pi_i = s) = P(y_{[1,i]}, \pi_i = s) \cdot P(y_{[i+1,m]} | y_{[1,i]}, \pi_i = s) \)
  \[ = P(y_{[1,i]}, \pi_i = s) \cdot P(y_{[i+1,m]} | \pi_i = s) \]
  \[ = f_s(i) \cdot b_s(i) \]
  where we set \( b_s(i) = P(y_{[i+1,m]} | \pi_i = s) \)

- In fact, every symbol after \( y[i] \) depends only on the state \( \pi_i \)
- **Problem**: find \( b_s(i) \)

\[ P(U,V) = P(V)P(U|V) \]

"Backward" algorithm

- Suppose the probabilities \( b_s(i) = P(y_{[i+1,m]} | \pi_i = s) \) are known for all states \( s \), then we can compute \( b_t(i-1) \) as follows
  \[ b_t(i-1) = \sum_s a_{t,s} \cdot e_s(y_{[ii]} \cdot b_s(i) \]
- Use dynamic programming to tabulate the matrix \( b \) (this time starting at the end of the sequence)
- Use log space
Computing the posterior probabilities

Observe that

\[ P(\pi_i = s \mid y) = \frac{P(y, \pi_i = s)}{P(y)} = \frac{f_s(i) \ b_s(i)}{P(y)} \]

where \( P(y) \) can be obtained either in the forward or the backward algorithm.

Hyper-parameter and parameter estimation

- Two are the main issues:
  1) The design of the structure, e.g., order, states and how states are connected (not covered here)
  2) The assignment of the parameters values: transition probabilities \( a_{s,t} \) and emission probabilities \( e_s(b) \)
If the state paths are known

- When all the paths are known, estimation is simple: count the number of times a particular transition or emission is used in the training set
- The max likelihood estimation (h=1) are

\[
a_{s,t} = \frac{f(st)}{\sum_{t' \in \Sigma} f(st')}, \quad e_{s}(b) = \frac{f_{s}(b)}{\sum_{c \in \Sigma} f_{s}(c)}
\]

If paths are unknown

- A closed-form equation to estimate parameter values is not available anymore
- An iterative procedure must be used
- Several algorithms for optimization of continuous function can be used
- Baum-Welch algorithm [BW 72] is the “standard” learning method for HMMs
Baum-Welch

Key observations:
• If we knew the paths, we could compute transition and emission probabilities
• If we knew the transition and emission probabilities, we could compute the paths (e.g., the most probable paths)

Baum-Welch

Iterative process involving repeating 1&2:
1. Compute the most probable paths from the current values of $a_{s,t}$ and $e_s(b)$
2. Estimate the new parameters using

$$a_{s,t} = \frac{f(st)}{\sum_{t' \in \Sigma} f(st')}$$
$$e_s(b) = \frac{f_s(b)}{\sum_{c \in \Sigma} f_s(c)}$$
Baum-Welch

- It can be proved that at each iteration of overall likelihood of the model is increased
- We hope it will converge to the global maximum (likelihood)
- Unfortunately, one could get stuck in a local maximum: the one you end up with depends on the initial assignments of the parameters

Baum-Welch

- The “actual” counts $f(st)$ and $f_s(b)$ are not available because we don’t know the paths
- Baum-Welch replaces the actual counts $f(st)$ and $f_s(b)$ with the expected counts, based on the current parameters
The probability that \( a_{s,t} \) is used at position \( i \) in sequence \( y \) is

\[
P(\pi_i = s, \pi_{i+1} = t \mid y) = \frac{P(\pi_i = s, \pi_{i+1} = t, y)}{P(y)}
\]

\[
= \frac{P(y_{[1,i]}, y_{[i+1,m]}, \pi_i = s, \pi_{i+1} = t)}{P(y)}
\]

\[
= \frac{P(y_{[1,i]}, \pi_i = t \mid y_{[1,i]}, \pi_i = s) \cdot P(y_{[1,i]}, \pi_i = s)}{P(y)}
\]

\[
= \frac{P(y_{[i+1,m]}, \pi_{i+1} = t \mid \pi_i = s) \cdot P(y_{[1,i]}, \pi_i = s)}{P(y)}
\]

\[
= \frac{b_t(i+1) \cdot a_{s,t} \cdot e_f(y_{[i+1]}) \cdot f_s(i)}{P(y)}
\]

**Baum-Welch**

Then the expected number of times that \( a_{s,t} \) is used can be obtained summing over all positions and over all the \( k \) sequences \( \{x_1, x_2, ..., x_k\} \)

\[
A_{s,t} = \sum_{j=1}^{k} \frac{1}{P(x_j)} \sum_{i=1}^{|x_j|} f_s^j(i) \cdot a_{s,t} \cdot e_f(x_{j,[i+1]}^j) \cdot b_t^j(i+1)
\]

where \( f_s^j \) is the forward variable for the \( j \)-th sequence, and \( b_t^j \) is the corresponding backward variable.
**Baum-Welch**

Similarly, the expected number of times that symbol $b$ is emitted in state $s$ in all the $k$ sequences $\{x_1, x_2, ..., x_k\}$ is given by:

$$E_s(b) = \sum_{j=1}^{k} \frac{1}{P(x_j)} \sum_{i: |x_j|: x_{j,i} = b} f_s^j(i) b_s^j(i)$$

where the inner sum is only over the position of $x_j$ for which the emitted symbol is $b$.

**Baum-Welch**

- New model parameters are calculated using the usual formula, but this time based on the expected number of times, instead of the actual count:

$$a_{s,d} = \frac{A_{s,d}}{\sum_{t' \in \Sigma} A_{s,t'}} \quad e_s(b) = \frac{E_s(b)}{\sum_{c \in \Sigma} E_s(c)}$$

- Then we compute again $A_{s,t}$ and $E(b)$ based on the new parameters and iterate.
Baum-Welch

- Initialize the model with random probabilities
- Iterative step
  - Set all $A_{s,t}$ and $E_{s}(b)$ variables to zero (or to some pseudo-count $r$)
  - For each sequence $j=1...k$
    - Compute $f_{k}(i)$ for sequence $j$ using forward algorithm
    - Compute $b_{k}(i)$ for sequence $j$ using backward algorithm
    - Add the contribution of sequence $j$ to $A$ and $E$
  - Compute the new model parameters
- Compute the new log likelihood of the model
- Termination
  - Stop when the change in the log likelihood is less than some predefined threshold or max iterations reached
From alignments to HMMs

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Rigid pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A C A - - - A T G</td>
<td>[AT] [CG] [AC] [ACGT]* A [TG] [GC]</td>
</tr>
<tr>
<td>T C A A C T A T C</td>
<td></td>
</tr>
<tr>
<td>A C A C - - A G C</td>
<td></td>
</tr>
<tr>
<td>A G A - - - A T C</td>
<td></td>
</tr>
<tr>
<td>A C C G - - A T C</td>
<td></td>
</tr>
</tbody>
</table>

Avg length = (3+1+1)/3 = 1.666
Exp geom distr = 1/(1-.4) = 1.666

HMM structure

• A typical scenario is to model a stretch of DNA for which the distribution does not change for a certain length

• The simplest model implies that

\[ P(m \text{ symbols}) = (1-p)p^{m-1} \]

• The geometric distribution is not always appropriate to model the length
Profile HMMs

- Profile which allows insertions and/or deletions

Example

Shaded areas are conserved and therefore chosen for the main states

From A. Krogh [1998]
Example

The profile HMM trained on the previous alignment. Dashed lines are low probability transition. Probabilities times 100 are shown in insert states

Example

The same profile HMM but using a pseudo-count of one. We can use profile HMM to search in databases for similar sequences. To align a sequence to a HMM we use Viterbi’s algorithm.
HMMs for gene finding

HMMs for coding regions
HMMs for unspliced genes

From A. Krogh [1998]

An HMM for spliced genes

From A. Krogh [1998]
A sample of applications of HMMs

• Protein classification
• Alignment and searching (SAM)
• Gene finding (HMMer, HMMgene, Genie, Genscan)
• Splicing sites finding
• Promoter finding
• Translation initiation sites finding
• Prediction of secondary structure of proteins
• Modeling site dependence of evolutionary rates
• ...