### Recall: Training with Labeled Sequences

The sequence of DNA is shown below along with the corresponding labels:

```
CGATATTGATTCTACGCGCGTATAACTAGCTTATCTGATC
```

The labels are represented as:

```
011111112222222111111222211111122221111110
```

#### Transitions from State to State

<table>
<thead>
<tr>
<th>from state</th>
<th>to state</th>
<th>0 (0%)</th>
<th>1 (100%)</th>
<th>2 (0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1 (4%)</td>
<td>21 (84%)</td>
<td>3 (12%)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 (0%)</td>
<td>3 (20%)</td>
<td>12 (80%)</td>
<td></td>
</tr>
</tbody>
</table>

#### Emissions in Symbol

<table>
<thead>
<tr>
<th>in state</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6 (24%)</td>
<td>7 (28%)</td>
<td>5 (20%)</td>
<td>7 (28%)</td>
</tr>
<tr>
<td>2</td>
<td>3 (20%)</td>
<td>3 (20%)</td>
<td>2 (13%)</td>
<td>7 (47%)</td>
</tr>
</tbody>
</table>

The equation for computing the state transition probability is:

\[
\alpha_{i,j} = \frac{A_{i,j}}{\sum_{h=0}^{|Q|-1} A_{i,h}}
\]

And the equation for computing the emission probability is:

\[
e_{i,k} = \frac{E_{i,k}}{\sum_{h=0}^{|\alpha|-1} E_{i,h}}
\]
The Problem:

We have training sequences, but not the associated paths (state labels). Therefore, we have to do unsupervised training.

The Solution:

**EM Training**: sum over all possible paths to estimate expected counts $A_{i,j}$ and $E_{i,k}$; then use the same formulas as for labeled sequence training on these expected counts:

$$a_{i,j} = \frac{A_{i,j}}{\sum_{h=0}^{Q-1} A_{i,h}}$$

$$e_{i,k} = \frac{E_{i,k}}{\sum_{h=0}^{|\alpha|-1} E_{i,h}}$$

Doing this iteratively is guaranteed to maximize the likelihood...
The Baum-Welch Algorithm ("EM")

1. **Initial Submodel** $M$
2. **Forward-Backward**
3. **Expected Counts**
   - $a_{i,j} = \frac{A_{i,j}}{\sum_{h=0}^{|Q|-1} A_{i,h}}$
   - $e_{i,k} = \frac{E_{i,k}}{\sum_{h=0}^{|Q|-1} E_{i,h}}$
4. **Repeat $n$ Times**
5. **New Submodel** $M'$
6. **Final Submodel** $M^*$

**Training Features**

**S** = $x_0 \ldots \ldots x_{k-1} x_k \ldots \ldots x_{L-1}$

**$F(i,k)$** = $P(x_0 \ldots x_{k-1}, q_i) = P(M \text{ emits } x_0 \ldots x_{k-1} \text{ by any path ending in state } q_i, \text{ with } x_{k-1} \text{ emitted by } q_i)$.

**$B(i,k)$** = $P(x_k \ldots x_{L-1} | q_i) = P(M \text{ emits } x_k \ldots x_{L-1} \text{ and then terminates, given that } M \text{ is in state } q_i, \text{ which has emitted } x_{k-1})$. 
Forward-Backward: Summing over All Paths

\[ F(i,k)B(i,k) \]

\[ = P(M \text{ emits } x_0...x_{k-1}...x_{L-1}, \text{ with } x_{k-1} \text{ being emitted by state } q_i) \]

\[ \Rightarrow \text{anchoring the path at a particular emission.} \]

\[ F(i,k)P_t(q_j|q_i)P_e(x_k|q_j)B(j,k+1) \]

\[ = P(M \text{ emits } x_0...x_{k-1}x_k...x_{L-1} \text{ and uses transition } q_i \rightarrow q_j \text{ at time } k-1) \]

\[ \Rightarrow \text{anchoring the path at a particular transition.} \]
Recall: The Forward Algorithm

\[ F(i,k) = \begin{cases} 
1 & \text{for } k = 0, i = 0 \\
0 & \text{for } k = 0, i > 0 \\
0 & \text{for } k > 0, i = 0 \\
\sum_{j=0}^{\lfloor Q \rfloor - 1} F(j,k-1)P_t(q_i|q_j)P_e(x_{k-1}|q_i) & \text{for } 1 \leq k \leq |S|, \\
& 1 \leq i < |Q| 
\end{cases} \]

\[ F(i,k) \text{ represents the probability } P(x_0 \ldots x_{k-1}, q_i) \text{ that the machine emits the subsequence } x_0 \ldots x_{k-1} \text{ by any path ending in state } q_i \text{—i.e., so that symbol } x_{k-1} \text{ is emitted by state } q_i. \]

\[ P(S|M) = \sum_{i=0}^{\lfloor Q \rfloor - 1} F(i,|S|)P_t(q_0|q_i) \]
The Backward Algorithm

\[ B(i,k) = \begin{cases} \sum_{j=1}^{|Q|-1} P_t(q_j|q_i)P_e(x_k|q_j)B(j,k+1) & \text{if } k < L, \\ P_t(q_0|q_i) & \text{if } k = L. \end{cases} \] recurrence

\[ B(i,k) = \text{probability that the machine } M \text{ will emit the subsequence } x_k...x_{L-1} \text{ and then terminate, given that } M \text{ is currently in state } q_i \text{ (which has already emitted } x_{k-1}). \]

THEREFORE:

\[ P(S|M) = B(0,0) \]
The Forward-Backward Algorithm

\[ F(i,k) = P(x_0...x_{k-1}, q_i) = P(M \text{ emits } x_0...x_{k-1} \text{ by any path ending in state } q_i, \text{ with } x_{k-1} \text{ emitted by } q_i). \]

\[ B(i,k) = P(x_k...x_{L-1}|q_i) = P(M \text{ emits } x_k...x_{L-1} \text{ and then terminates, given that } M \text{ is in state } q_i, \text{ which has emitted } x_{k-1}). \]

\[ F(i,k)B(i,k) = P(x_0...x_{k-1}, q_i)P(x_k...x_{L-1}|q_i) = P(x_0...x_{L-1}, q_i@k-1)* \]

\[ F(i,k)B(i,k)/P(S) = P(q_i@k-1 \mid S) \]

**expectation** \[ E_{q_i,S} \]

\[ \text{expectation} = \sum_{k \in x_k=s} \left( \sum_{c=0}^{1} P_{\text{count}}(c \mid S)c \right) = \sum_{k \in x_k=s} P(q_i \mid k \mid S) = \sum_{k \in x_k=s} \frac{F(i,k+1)B(i,k+1)}{P(S)} \]

**expectation** \[ A_{i,j} \]

\[ \text{expectation} = \sum_k \frac{F(i,k)P_t(q_j \mid q_i)P_e(x_k \mid q_j)B(j,k+1)}{P(S)} \]

*because \( x_k...x_{L-1} \) is conditionally independent of \( x_0...x_{k-1} \), given \( q_i@k-1 \)*
The Baum-Welch Algorithm (EM)

**Procedure** `baumWelch(ref M, T, n)`
1. \((Q, \alpha, P_t, P_e) \leftarrow M;\)
2. for \(h \leftarrow 1\) up to \(n\) do
   3. \(\forall i \in [0, |Q|-1], \forall j \in [0, |Q|-1]\) \(A[i][j] \leftarrow 0;\)
   4. \(\forall i \in [0, |Q|-1], \forall k \in [0, |q|-1]\) \(E[i][k] \leftarrow 0;\)
   5. foreach \(S \in T\) do
      6. \(F \leftarrow \text{forwardAlgorithm}(M, S);\)
      7. \(B \leftarrow \text{backwardAlgorithm}(M, S);\)
      8. \(\text{updateCounts}(A, E, F, B, Q, S);\)
      9. \(\text{updateModel}(M, A, E);\)

**Procedure** `updateCounts(ref A, ref E, F, B, Q, S)`
1. \(P \leftarrow B[0][0];\)
2. for \(i \leftarrow 1\) up to \(|Q|-1\) do
   3. for \(k \leftarrow 0\) up to \(|S|-1\) do
      4. \(E[i][S[k-1]] \leftarrow E[i][S[k-1]] + F[i][k] \times B[i][k]/P;\)
      5. for \(j \leftarrow 0\) up to \(|Q|-1\) do
         6. \(A[j][i] \leftarrow A[j][i] + F[j][k] \times P_t(q_i | q_j) \times P_e(S[k] | q_i) \times B[i][k+1]/P;\)
      7. \(A[i][0] \leftarrow F[i][|S|] \times P_t(q_0 | q_i) / P;\)

\[
\begin{align*}
a_{i,j} &= \frac{A_{i,j}}{\sum_{h=0}^{|Q|-1} A_{i,h}} \\
e_{i,k} &= \frac{E_{i,k}}{\sum_{h=0}^{|Q|-1} E_{i,h}}
\end{align*}
\]

- compute Fwd & Bkwd DP matrices
- accumulate expected counts for E & A
In the log-space version of these algorithms, we can replace the raw probabilities $p_i$ with their logarithmic counterparts, $\log p_i$, and apply the above equation whenever the probabilities are to be summed. Evaluation of the $e^{\log p_i - \log p_0}$ term should generally not result in numerical underflow in practice, since this term evaluates to $p_i/p_0$, which for probabilities of similar events should not deviate too far from unity.

(due to Kingsbury & Rayner, 1971)

Tip: watch out for zero probabilities!
Monotonic Convergence - To a Local Maximum
Another use of Forward-Backward: *Posterior Decoding*

\[
P(E \text{ is an exon} \mid S) = \sum_{\text{all } \phi \text{ containing } E} P(\phi \mid S)
\]

\[
P(E = [i, j] \text{ is an exon} \mid S) = \sum F(y, i) P_t(q_{\text{exon}} \mid q_y) \left( \prod_{k=i}^{j} P_e(x_k \mid q_{\text{exon}}) \right)
\]

\[
\times P_t(q_{\text{exon}} \mid q_{\text{exon}})^{j-i} P_t(z \mid q_{\text{exon}}) P_e(x_{j+1} \mid z) B(z, j + 2) / P(S)
\]
Define $L(\theta) = \log \sum_Y P(X, Y\mid \theta)$ for observables $X$, unobservables $Y$, and model $\theta$.

Dempster et al. (1977) introduced the $Q$ function:

$$Q(\theta_{k+1}\mid \theta_k) = \sum_Y P(Y\mid X, \theta_k) \log P(X, Y\mid \theta_{k+1})$$

from which we can derive another function, $V$, having several useful properties:

$$V(\theta_{k+1}\mid \theta_k) = L(\theta_k) + Q(\theta_{k+1}\mid \theta_k) - Q(\theta_k\mid \theta_k)$$

(Borman, 2006). The two useful properties of $V$ are:

$$L(\theta_{k+1}) \geq V(\theta_{k+1}\mid \theta_k) \quad \text{and} \quad L(\theta_k) = V(\theta_k\mid \theta_k)$$

The first property follows from *Jensen’s inequality* and the concavity of the log function; the second follows directly from the definition of $V$. 
Observe that $c \geq b \geq a$. Thus, by stepping from $\theta_k$ to $\theta_{k+1}$ we will increase $L$:

$$L(\theta_{k+1}) \geq V(\theta_{k+1} | \theta_k) \geq V(\theta_k | \theta_k) = L(\theta_k)$$

$$\therefore L(\theta_{k+1}) \geq L(\theta_k)$$
Thus, by increasing $V$ we are guaranteed to increase $L$. By iteratively increasing $V$ (for successive values of $k$) we will be effectively hill-climbing on $L$. Furthermore, if we iteratively maximize $V$ rather than just increasing it, we will be hill-climbing with the largest possible step size at each iteration:

$$
\theta_{k+1} = \arg \max_{\theta} V(\theta|\theta_k)
$$

Since any term not dependent on $\theta$ drops out of the optimization:

$$
\arg \max_{\theta} V(\theta|\theta_k) = \arg \max_{\theta} \left( L(\theta_k) + Q(\theta|\theta_k) - Q(\theta_k|\theta_k) \right),
$$

we arrive at the **EM update equation**:

$$
\theta_{k+1} = \arg \max_{\theta} Q(\theta|\theta_k)
$$
The $Q$ Function for HMMs

In the case of HMMs, we have:

$$Q(\theta_{k+1} | \theta_k) = \sum_{\phi} P(\phi | S, \theta_k) \log P(S, \phi | \theta_{k+1})$$

$$= \sum_{i=1}^{N-1} \sum_{x \in \alpha} E_{i,x} \log e_{k,x} + \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} A_{i,j} \log a_{i,j}$$

(Durbin et al., 1998). It can be shown (see next slide) that $Q$ is maximized by simply choosing:

$$a_{i,j}^{k+1} = \frac{A_{i,j}}{\sum_{h=0}^{LQ-1} A_{i,h}} \quad e_{i,j}^{k+1} = \frac{E_{i,j}}{\sum_{h=0}^{L\alpha-1} E_{i,h}} \quad \theta_{k+1} = \{e_{\cdot,j}^{k+1}, a_{i,\cdot}^{k+1}\}$$

“EM update equations”

where $A_{i,j}$ and $E_{i,j}$ are the expected emission and transition counts computed via Forward-Backward.
The $Q$ Function for HMMs

To see that this choice of parameters will indeed maximize $Q$, consider the difference between the chosen $\theta_{k+1}$ and an alternative $\theta$:

$$Q(\theta_{k+1}|\theta_k) - Q(\theta|\theta_k) =$$

$$\sum_{i=1}^{N-1} \left( \sum_{y \in \alpha} E_{i,y} \right) \sum_{x \in \alpha} e^{k+1}_{i,x} \log \frac{e^{k+1}_{i,x}}{e_{i,x}} + \sum_{i=0}^{N-1} \left( \sum_{j=0}^{N-1} A_{i,j} \right) \sum_{h=0}^{N-1} a^{k+1}_{i,h} \log \frac{a^{k+1}_{i,h}}{a_{i,h}}$$

(Durbin et al., 1998). The log-sum terms are relative entropies, which are always non-negative; the $E_{i,y}$ and $A_{i,j}$ terms are also non-negative, and are constant for any fixed $\theta_k$. Thus, $Q(\theta_{k+1}|\theta_k) \geq Q(\theta|\theta_k)$ for any $\theta$ other than $\theta_{k+1}$, so $Q(\theta_{k+1}|\theta_k)$ is maximal.
We want to maximize the likelihood, $L$.

We can do this by iteratively maximizing $V$.

Maximizing $V$ is equivalent to maximizing $Q$:

$$\arg \max_{\theta} V(\theta|\theta_k) = \arg \max_{\theta} \left( L(\theta_k) + Q(\theta|\theta_k) - Q(\theta_k|\theta_k) \right)$$

Using the normalized, expected counts maximizes $Q$:

$$a_{i,j}^{k+1} = \frac{A_{i,j}}{\sum_{h=0}^{Q-1} A_{i,h}} \quad e_{i,j}^{k+1} = \frac{E_{i,j}}{\sum_{h=0}^{\alpha-1} E_{i,h}}$$

Therefore, these formulas maximize the likelihood.
Continuous, Multivariate HMMs

(continuous values, discrete time)
Types of Emissions

- **alphabetical:** DNA/RNA – e.g., ATTCGATCGCGATA
- **numeric:**
  - **discrete:** read counts – e.g., 1,2,3,4,5,6,7,...
  - **continuous:** anything – e.g., 1.372, -11234.8876, 7.42e-10

Examples of numeric sequential data:
- Read counts from deep sequencing
- DNAse I hypersensitivity
- Chromatin marks
- DNA methylation assays
- Raw sequencer output
Gaussian mixture model:

\[ D \] variates ("signals", "dimensions")

\[ m \] mixture components

\[ \lambda_i = \text{a single mixture weight} \]

\[ \mu_i = \text{vector of } D \text{ means (one mean per variate)} \]

\[ C_i = D \times D \text{ covariance matrix (one)} \]

For each component \( i \) we have:

\[ e_q(s) = \sum_{j=0}^{m-1} \lambda_{qj} \mathcal{N}(s; \mu_j, C_j) \]

"mixture of Gaussians"

\[ \mathcal{N}(s; \mu, C) = \frac{1}{\sqrt{\det(2\pi C)}} e^{-\frac{1}{2} (s-\mu)^T C^{-1} (s-\mu)} \]

"Gaussian density"
Posterior probability of mixture component $j$ for observation $i$:

$$
\rho_{ij} = \frac{\lambda_j \mathcal{N}(S_i;\mu_j,C_j)}{\sum_{j=0}^{m-1} \lambda_j \mathcal{N}(S_i;\mu_j,C_j)}
$$

EM update formulas:

$$
\mu_j \leftarrow \frac{\sum_{i=0}^{L-1} S_i \rho_{ij}}{\sum_{i=0}^{L-1} \rho_{ij}}
$$

$$
C_j \leftarrow \frac{\sum_{i=0}^{L-1} (S_i - \mu_j)(S_i - \mu_j)^T \rho_{ij}}{\sum_{i=0}^{L-1} \rho_{ij}}
$$

$$
\lambda_j \leftarrow \frac{\sum_{i=0}^{L-1} \rho_{ij}}{\sum_{j=0}^{m-1} \sum_{i=0}^{L-1} \rho_{ij}}
$$
Training a Gaussian HMM with EM

State emission probability:

\[ e_q(s) = \sum_{j=0}^{m-1} \lambda_{qj} \mathcal{N}(s; \mu_j, C_j) \]

Posterior probability of state \( q \) & component \( j \) for observation \( i \):

\[
\rho_{qij} = \frac{F_{qi} B_{qi}}{\sum_{q \in Q} F_{qi} B_{qi}} \left( \frac{\lambda_{qj} \mathcal{N}(S_i; \mu_j, C_j)}{\sum_{j=0}^{m-1} \lambda_{qj} \mathcal{N}(S_i; \mu_j, C_j)} \right)
\]

EM update formulas:

\[
\begin{align*}
\mu_j &\leftarrow \frac{\sum_{q \in Q} \sum_{i=0}^{L-1} S_i \rho_{qij}}{\sum_{q \in Q} \sum_{i=0}^{L-1} \rho_{qij}} \\
C_j &\leftarrow \frac{\sum_{q \in Q} \sum_{i=0}^{L-1} (S_i - \mu_j)(S_i - \mu_j)^T \rho_{qij}}{\sum_{q \in Q} \sum_{i=0}^{L-1} \rho_{qij}} \\
\lambda_{qj} &\leftarrow \frac{\sum_{i=0}^{L-1} \sum_{j=0}^{m-1} \rho_{qij}}{\sum_{j=0}^{m-1} \sum_{i=0}^{L-1} \rho_{qij}}
\end{align*}
\]
A tends to emit high values for marks $v_1$-$v_3$,
B tends to emit high values for marks $v_4$-$v_6$,
C tends to emit high values for marks $v_7$-$v_9$
bg (background) is unbiased
Training a Model from the Simulated Data

Generator Model

Trained Model

emit

training sequences

EM

emit

test

test sequences
• Training an HMM with unlabeled sequences can be accomplished using the **Baum-Welch algorithm**.

• Baum-Welch estimates transition & emission events by computing expectations via **Forward-Backward**, by summing over all paths containing a given event. It works because it is an instance of the **EM algorithm**.

• EM finds a **local maximum** of the likelihood function.

• **Posterior decoding** can be used to estimate the probability that a given symbol or substring was generated by a particular state.

• **Continuous-emission** HMMs can be used to model various epigenetic sequence data, and they can be trained using EM


http://www.jstor.org/stable/2984875