Hidden Markov Models

part 2

CBB 261

B. Majoros

Duke
UNIVERSITY

Recall: Training with Labeled Sequences

CGATATTCGATTCTACGCGCGTATACTAGCTTATCTGATC

0111111122222211111122221111111222211111 0

transitions

		to state		
		0	**1**	**2**
from state	**0**	0 (0%)	1 (100%)	0 (0%)
	1	1 (4%)	21 (84%)	3 (12%)
	2	0 (0%)	3 (20%)	12 (80%)

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

emissions

		symbol			
		A	**C**	**G**	**T**
in state	**1**	6 (24%)	7 (28%)	5 (20%)	7 (28%)
	2	3 (20%)	3 (20%)	2 (13%)	7 (47%)

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

Duke UNIVERSITY

Training with Unlabeled Sequences

The Problem:

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

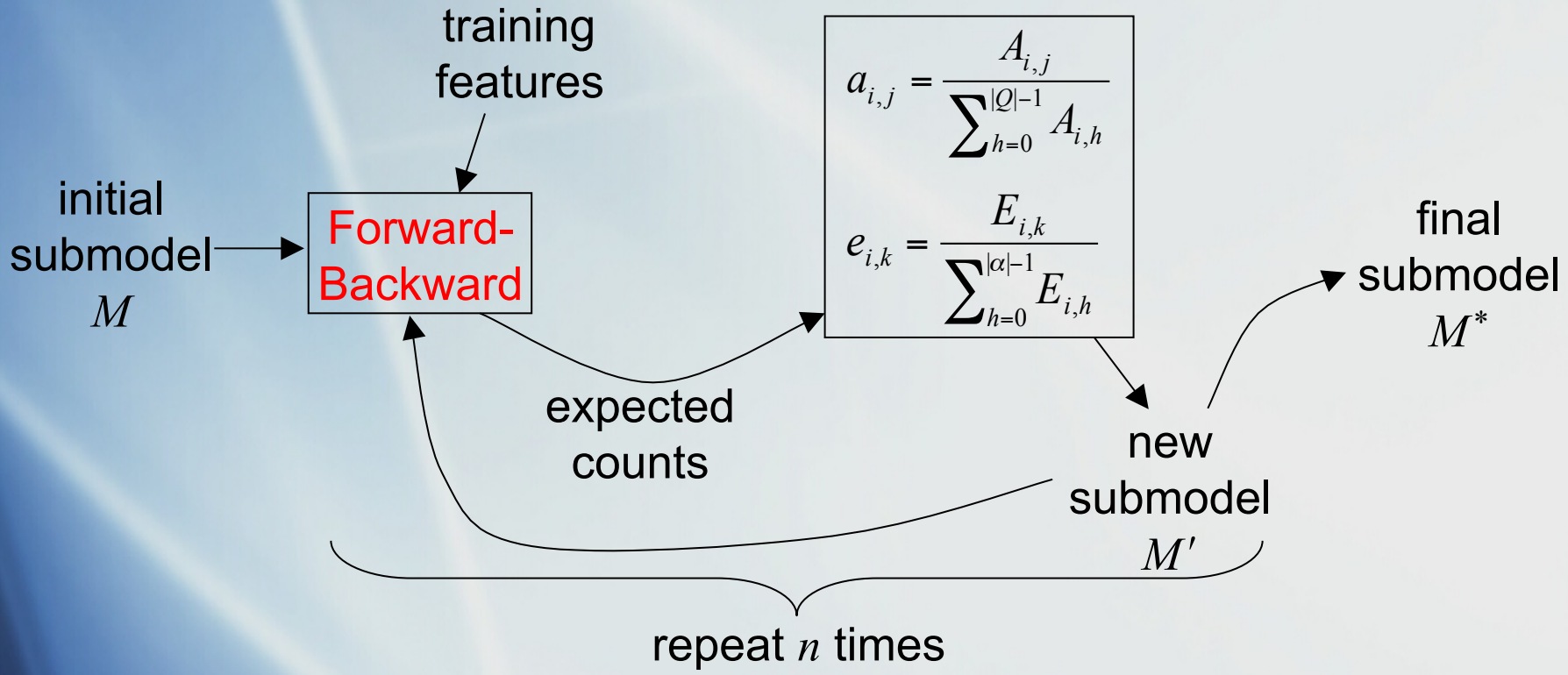
The Solution:

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:

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

Doing this iteratively is guaranteed to maximize the likelihood...

The Baum-Welch Algorithm("EM")

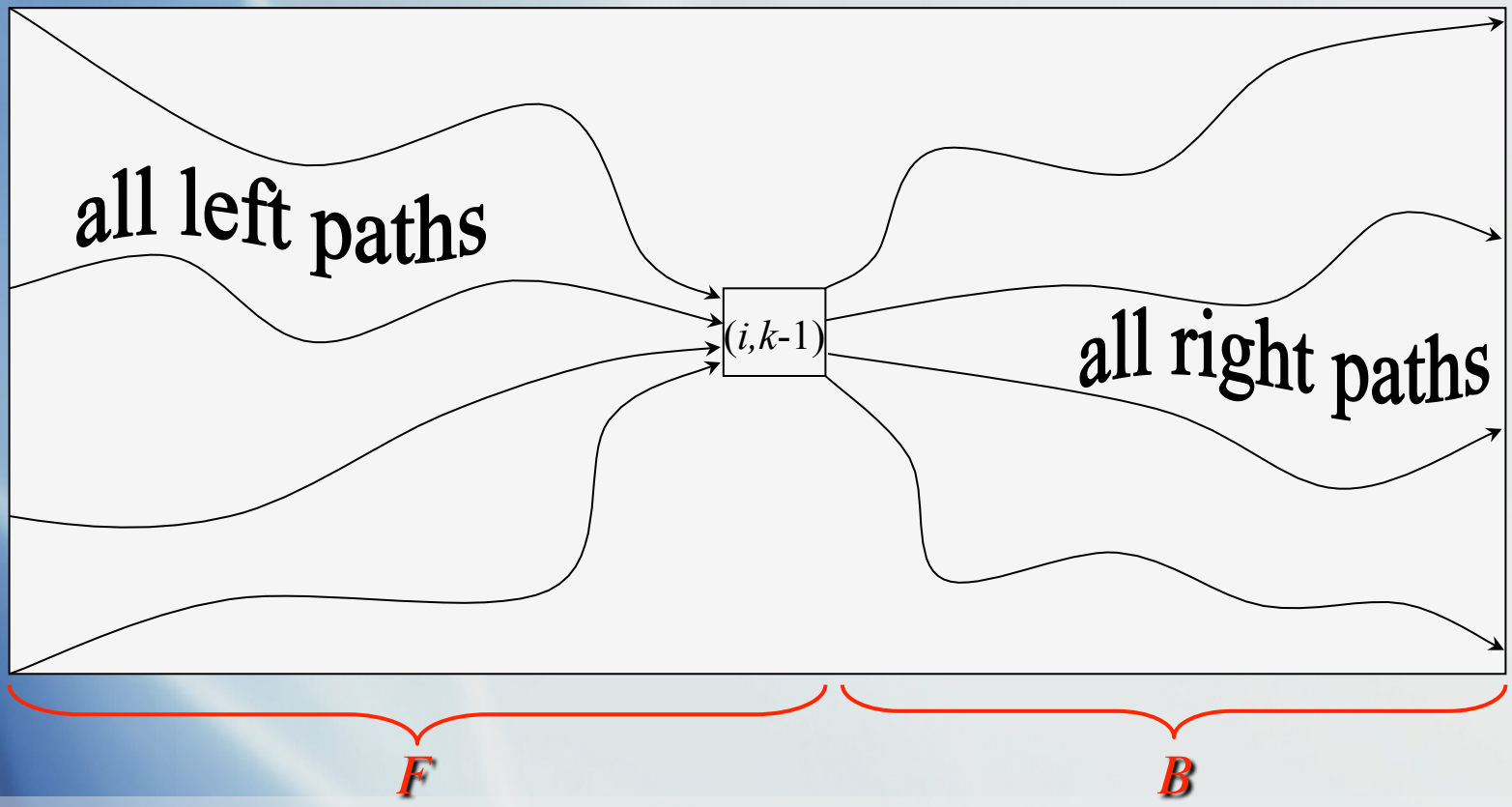


$$S = x_0 \ldots\ldots\ldots x_{k-1}\, x_k \ldots\ldots\ldots x_{L-1}$$

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

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

Forward-Backward: Summing over All Paths



$F(i,k)B(i,k)$

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

=> *anchoring the path at a particular emission*.

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

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

=> *anchoring the path at a particular transition*.

Recall: The Forward Algorithm

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

start in state q_0

q_0 is silent

recurrence

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

$$P(S|M) = \sum_{i=0}^{|Q|-1} F(i, |S|) P_t(q_0 | q_i)$$

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

end in state q_0

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

THEREFORE:

$$P(S|M) = B(0,0)$$

The Forward-Backward Algorithm

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

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

$F(i,k)B(i,k) = P(x_0 \ldots x_{k-1}, q_i)P(x_k \ldots x_{L-1} | q_i) = P(x_0 \ldots x_{L-1}, q_i @ k-1)^*$

$F(i,k)B(i,k)/P(S) = P(q_i @ k-1 \mid S)$ } this is the "Forward-Backward Algorithm"

$$expectation(E_{q_i,s}) = \sum_{k \ni x_k = s} \left(\sum_{c=0}^{1} P_{count}(c|S)c \right) = \sum_{k \ni x_k = s} P(q_i @ k | S) = \sum_{k \ni x_k = s} \frac{F(i,k+1)B(i,k+1)}{P(S)}$$

$$expectation(A_{i,j}) = \sum_k \frac{F(i,k)P_t(q_j|q_i)P_e(x_k|q_j)B(j,k+1)}{P(S)}$$

recall: $expectation[f(x)] = \sum_x P(x)f(x)$

*because $x_k \ldots x_{L-1}$ is conditionally independent of $x_0 \ldots x_{k-1}$, given $q_i @ k-1$

Duke UNIVERSITY

The Baum-Welch Algorithm (EM)

```
procedure baumWelch(ref M,T,n)
1.  (Q,α,P_t,P_e)←M;
2.  for h←1 up to n do
3.    ∀_{i∈[0,|Q|-1]} ∀_{j∈[0,|Q|-1]} A[i][j]←0;
4.    ∀_{i∈[0,|Q|-1]} ∀_{k∈[0,|α|-1]} E[i][k]←0;
5.    foreach S∈T do
6.      F←forwardAlgorithm(M,S);
7.      B←backwardAlgorithm(M,S);
8.      updateCounts(A,E,F,B,Q,S);
9.    updateModel(M,A,E);
```

compute Fwd & Bkwd DP matrices

accumulate expected counts for E & A

$$\text{updateModel}(M,A,E)\begin{cases} a_{i,j} = \dfrac{A_{i,j}}{\sum_{h=0}^{|Q|-1} A_{i,h}} \\ e_{i,k} = \dfrac{E_{i,k}}{\sum_{h=0}^{|\alpha|-1} E_{i,h}} \end{cases}$$

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

$$\frac{F(i,k+1)B(i,k+1)}{P(S)}$$

$$\frac{F(m,k)P_t(q_n \mid q_m)P_e(x_k \mid q_n)B(n,k+1)}{P(S)}$$

Duke UNIVERSITY

Using Logarithms in Forward & Backward

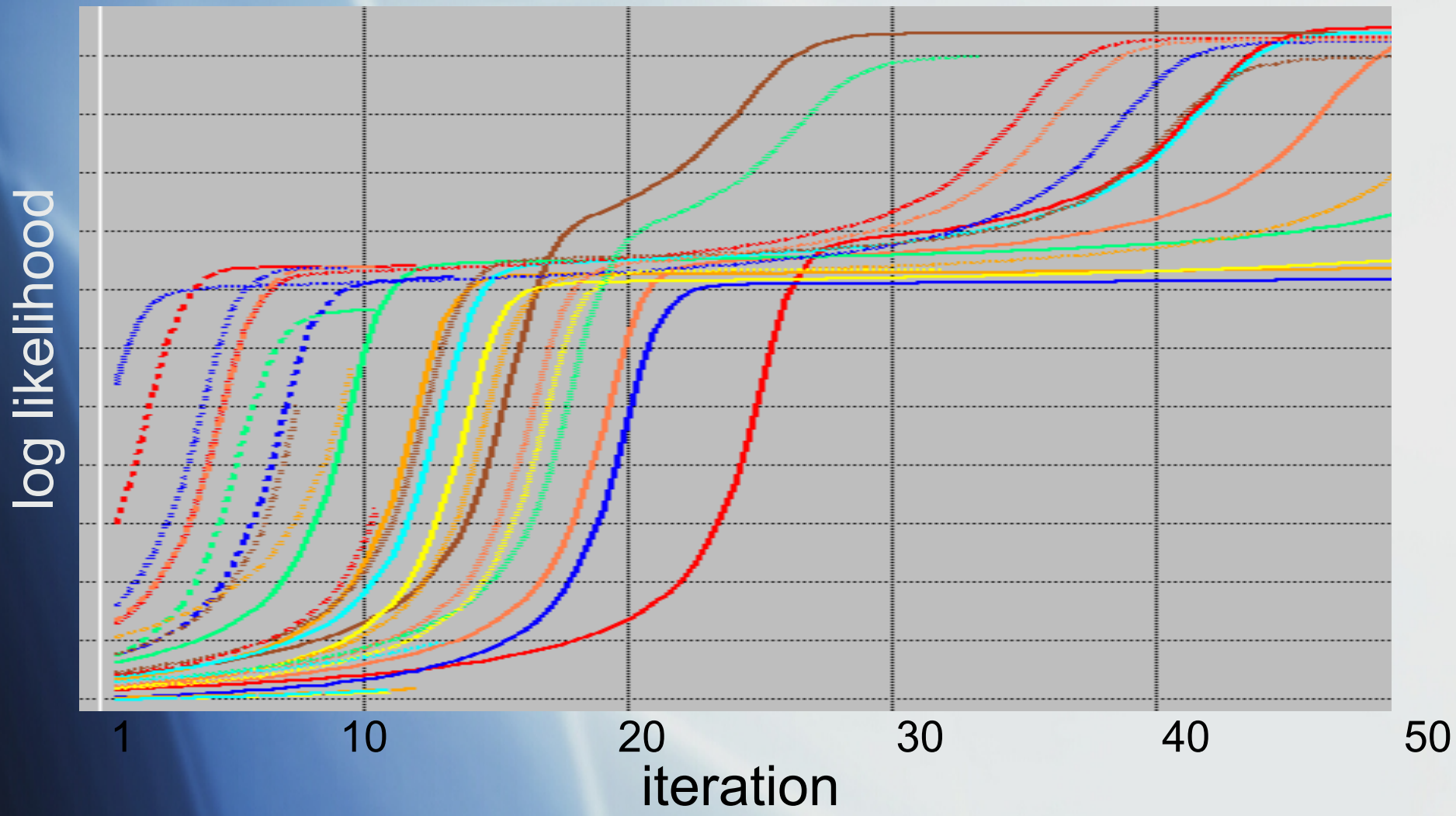
$$\log\left(\sum_{i=0}^{n-1} p_i\right) = \log p_0 + \log\left(1 + \sum_{i=1}^{n-1} e^{\log p_i - \log p_0}\right)$$

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.

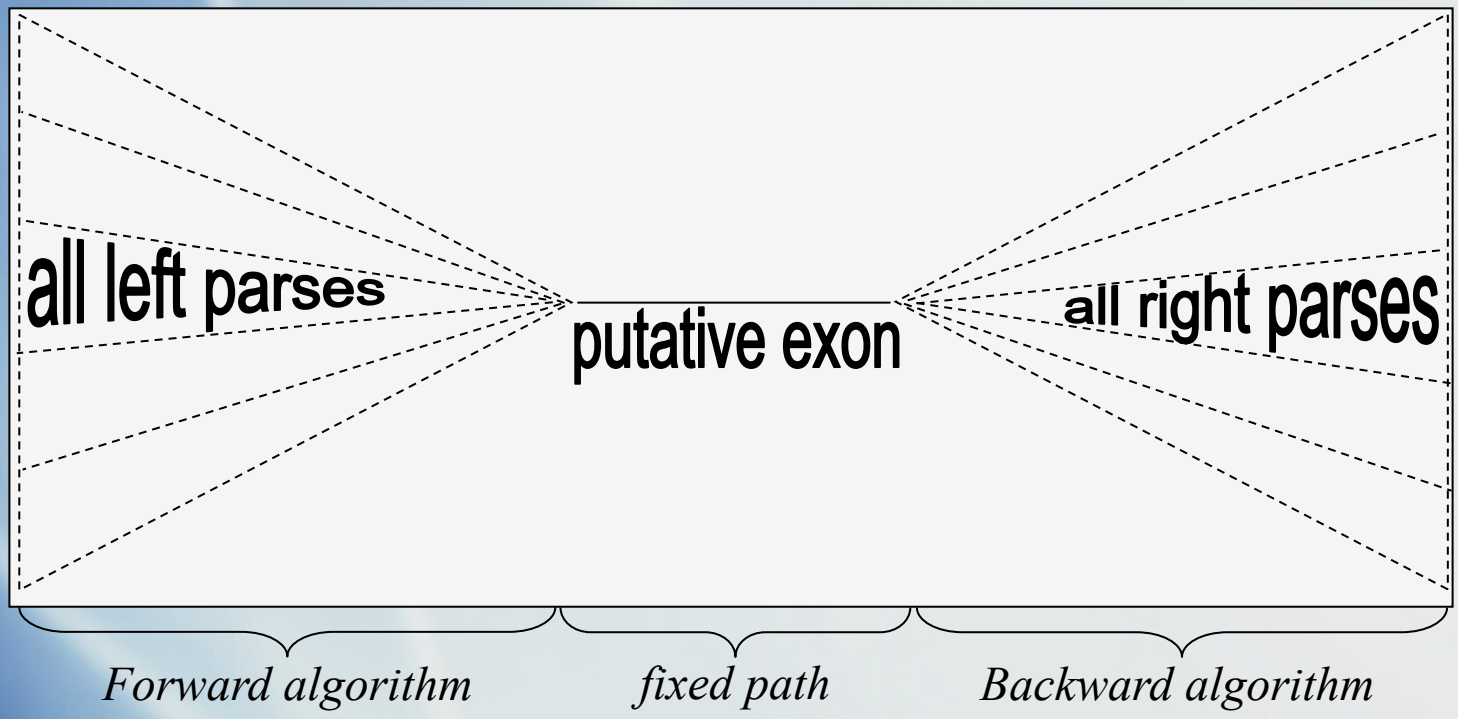
(due to Kingsbury & Rayner, 1971)

Tip: watch out for zero probabilities!

Monotonic Convergence - To a Local Maximum



Another use of Forward-Backward: *Posterior Decoding*



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

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

$$\times P_t(q_{exon} \mid q_{exon})^{j-i} P_t(z \mid q_{exon}) P_e(x_{j+1} \mid z) B(z, j+2) / P(S)$$

Why Baum-Welch Works: EM

Define $L(\theta) = \log \sum_{\mathbf{Y}} P(\mathbf{X}, \mathbf{Y} | \theta)$ for observables $\mathbf{X}$, unobservables $\mathbf{Y}$, and model θ.

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

$$Q(\theta_{k+1} | \theta_k) = \sum_{\mathbf{Y}} P(\mathbf{Y} | \mathbf{X}, \theta_k) \log P(\mathbf{X}, \mathbf{Y} | \theta_{k+1})$$

"expected value of the complete-data log likelihood"

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

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

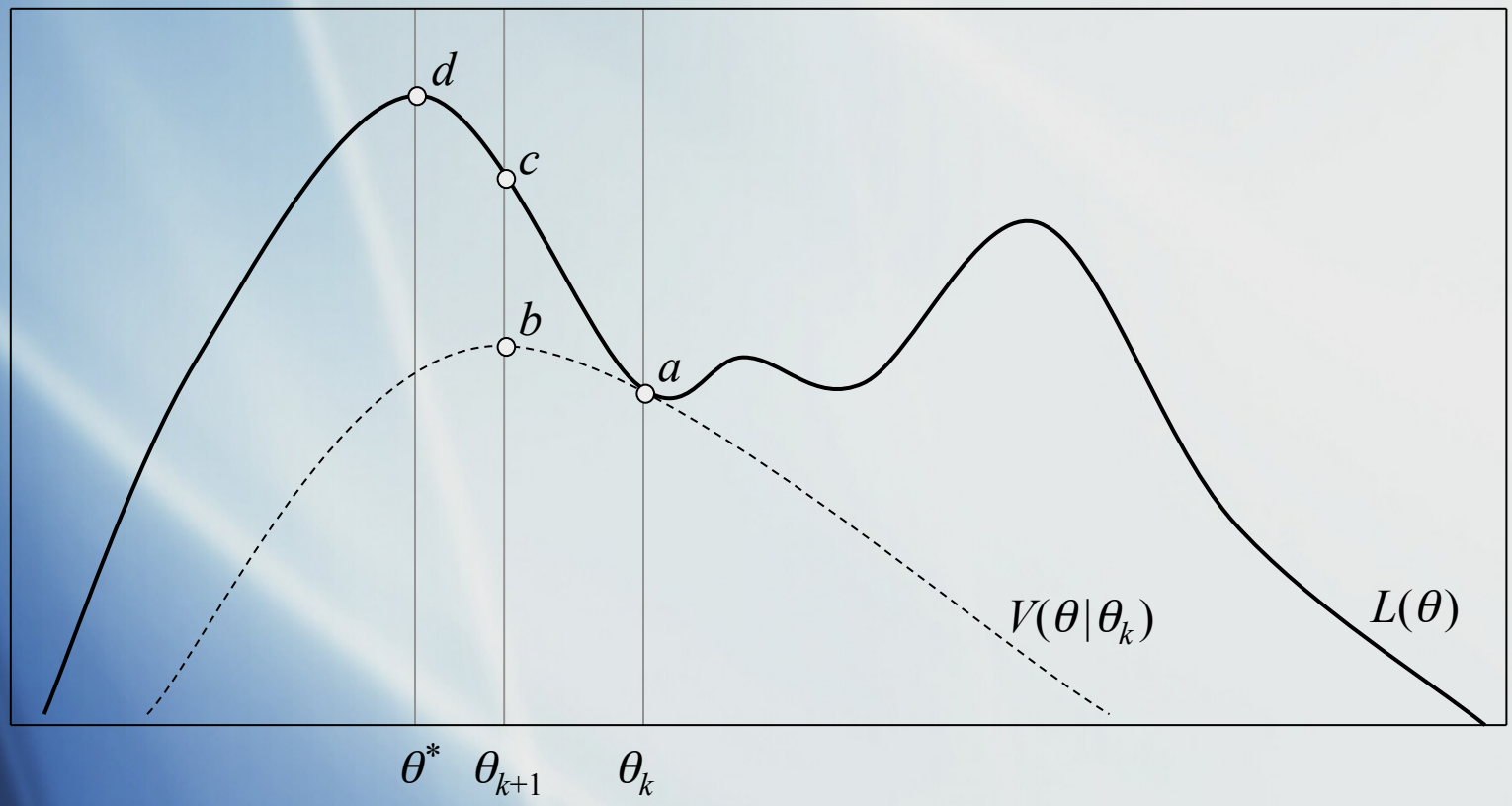
(Borman, 2006). The two useful properties of V are:

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

The first property follows from *Jensen's inequality* and the concavity of the log function; the second follows directly from the definition of V.

Visualizing EM

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



Observe that $c \geq b \geq a$. Thus, by stepping from θ_k to θ_{k+1} we will increase L:

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

$$\therefore L(\theta_{k+1}) \geq L(\theta_k)$$

EM

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 θ drops out of the optimization:

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

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:

$$a_{i,j}^{k+1} = \frac{A_{i,j}}{\sum_{h=0}^{|Q|-1} A_{i,h}} \qquad e_{i,j}^{k+1} = \frac{E_{i,j}}{\sum_{h=0}^{|\alpha|-1} E_{i,h}} \qquad \theta_{k+1} = \left\{e_{\bullet,\bullet}^{k+1}, a_{\bullet,\bullet}^{k+1}\right\}$$

"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 θ_{k+1} and an alternative θ:

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

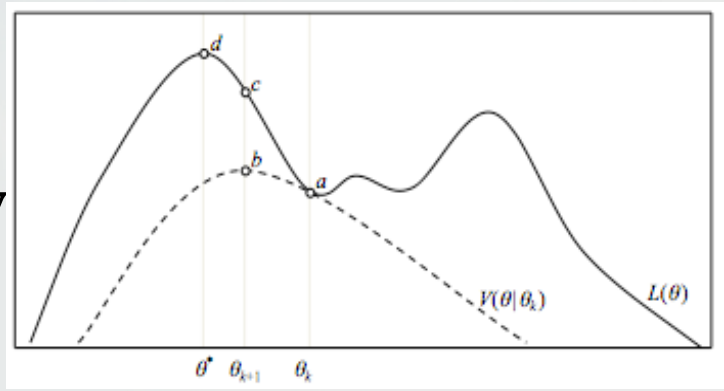
$$\sum_{i=1}^{N-1}\left(\sum_{y\in\alpha} E_{i,y}\right)\underbrace{\sum_{x\in\alpha} e_{i,x}^{k+1}\log\frac{e_{i,x}^{k+1}}{e_{i,x}}}_{\textit{relative entropy}} + \sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1} A_{i,j}\right)\underbrace{\sum_{h=0}^{N-1} a_{i,h}^{k+1}\log\frac{a_{i,h}^{k+1}}{a_{i,h}}}_{\textit{relative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

(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 θ_k. Thus, $Q(\theta_{k+1}|\theta_k) \geq Q(\theta|\theta_k)$ for any θ other than θ_{k+1}, so $Q(\theta_{k+1}|\theta_k)$ is maximal.

Summary for Baum-Welch

We want to maximize the likelihood, L.

We can do this by iteratively maximizing V



Maximizing V is equivalent to maximizing Q:

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

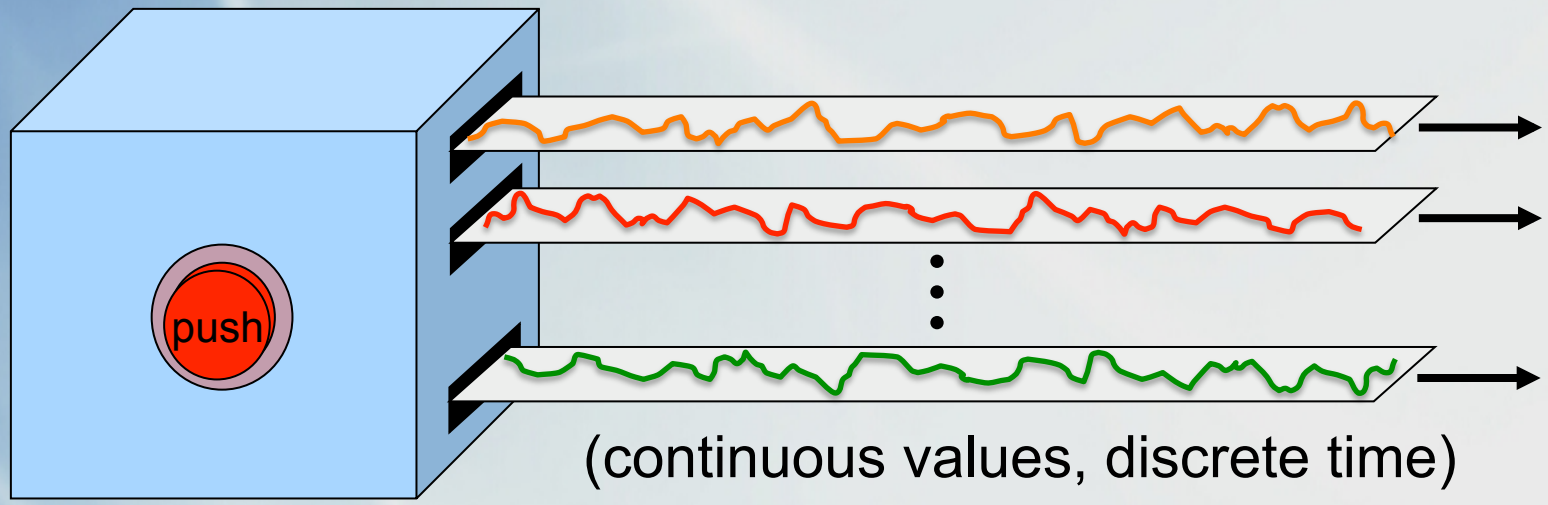
Using the normalized, expected counts maximizes Q:

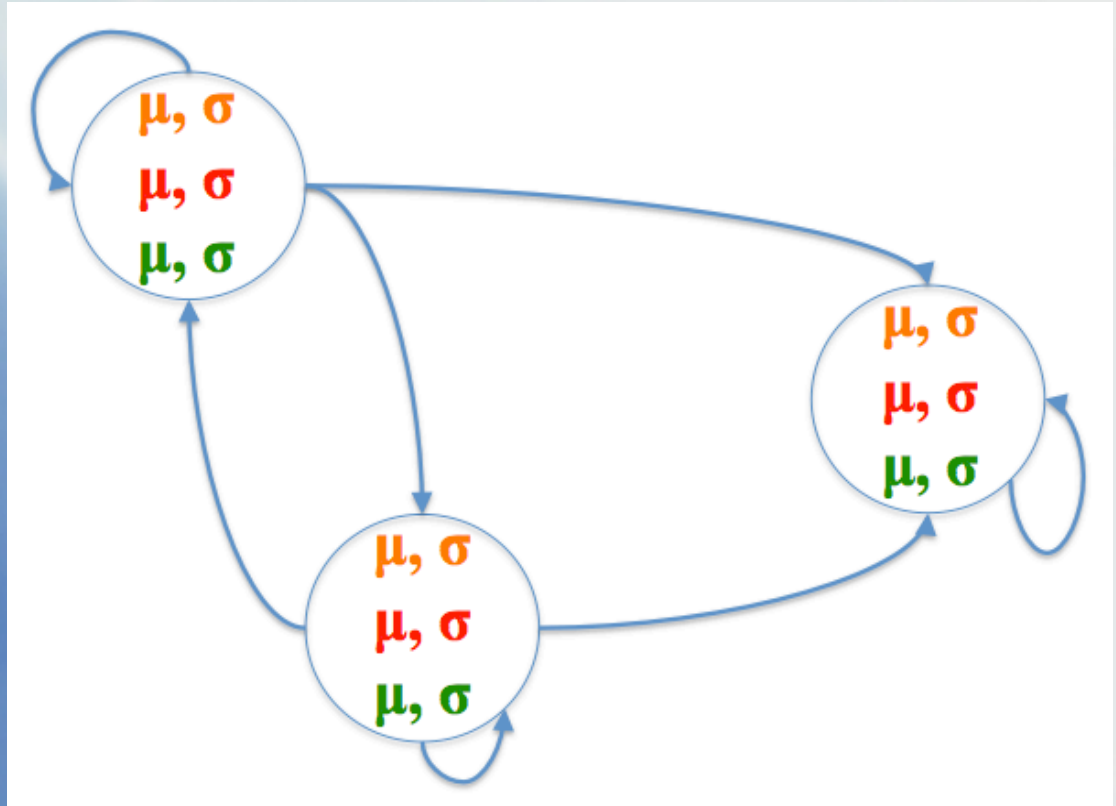
$$a_{i,j}^{k+1} = \frac{A_{i,j}}{\sum_{h=0}^{|Q|-1} A_{i,h}} \qquad e_{i,j}^{k+1} = \frac{E_{i,j}}{\sum_{h=0}^{|\alpha|-1} E_{i,h}}$$

$$Q(\theta_{k+1}|\theta_k) - Q(\theta|\theta_k) = \sum_{i=1}^{N-1}\left(\sum_{y\in\alpha} E_{i,y}\right)\underbrace{\sum_{x\in\alpha} e_{i,x}^{k+1}\log\frac{e_{i,x}^{k+1}}{e_{i,x}}}_{\text{relative entropy}} + \sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1} A_{i,j}\right)\underbrace{\sum_{h=0}^{N-1} a_{i,h}^{k+1}\log\frac{a_{i,h}^{k+1}}{a_{i,h}}}_{\text{relative entropy}}$$

Therefore, these formulas maximize the likelihood.

Continuous, Multivariate HMMs





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

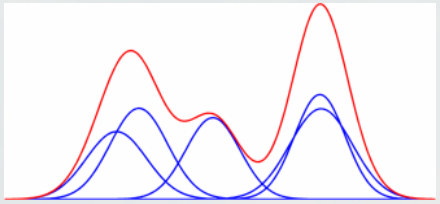
Examples of numeric sequential data:

Read counts from deep sequencing
DNAse I hypersensitivity
Chromatin marks
DNA methylation assays
Raw sequencer output

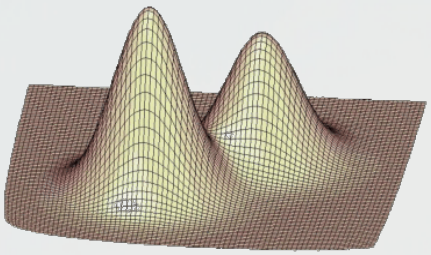
Modeling Continuous Emissions

Gaussian mixture model:

D variates ("signals", "dimensions")
m mixture components



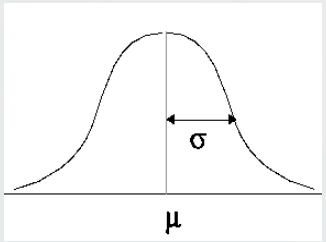
D=1, m=5



D=2, m=2

For each component i we have:

λ_i = a single mixture weight
$\boldsymbol{\mu}_i$ = vector of D means (one mean per variate)
$\mathbf{C}_i$ = $D \times D$ covariance matrix (one)



$$e_q(\mathbf{s}) = \sum_{j=0}^{m-1} \lambda_{qj} \mathcal{N}(\mathbf{s}; \mu_j, \mathbf{C}_j)$$

"mixture of Gaussians"

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

"Gaussian density"

Training a Gaussian Mixture with EM

Posterior probability of mixture component *j* for observation *i*:

$$\rho_{ij} = \frac{\lambda_j \mathcal{N}(\mathbf{S}_i; \mu_j, \mathbf{C}_j)}{\sum_{j=0}^{m-1} \lambda_j \mathcal{N}(\mathbf{S}_i; \mu_j, \mathbf{C}_j)}$$

EM update formulas:

$$\mu_j \leftarrow \frac{\sum_{i=0}^{L-1} \mathbf{S}_i \rho_{ij}}{\sum_{i=0}^{L-1} \rho_{ij}} \qquad \mathbf{C}_j \leftarrow \frac{\sum_{i=0}^{L-1} (\mathbf{S}_i - \mu_j)(\mathbf{S}_i - \mu_j)^T \rho_{ij}}{\sum_{i=0}^{L-1} \rho_{ij}} \qquad \lambda_j \leftarrow \frac{\sum_{i=0}^{L-1} \rho_{ij}}{\sum_{j=0}^{m-1} \sum_{i=0}^{L-1} \rho_{ij}}$$

Training a Gaussian HMM with EM

State emission probability:

$$e_q(\mathbf{s}) = \sum_{j=0}^{m-1} \lambda_{qj} \mathcal{N}(\mathbf{s}; \mu_j, \mathbf{C}_j)$$

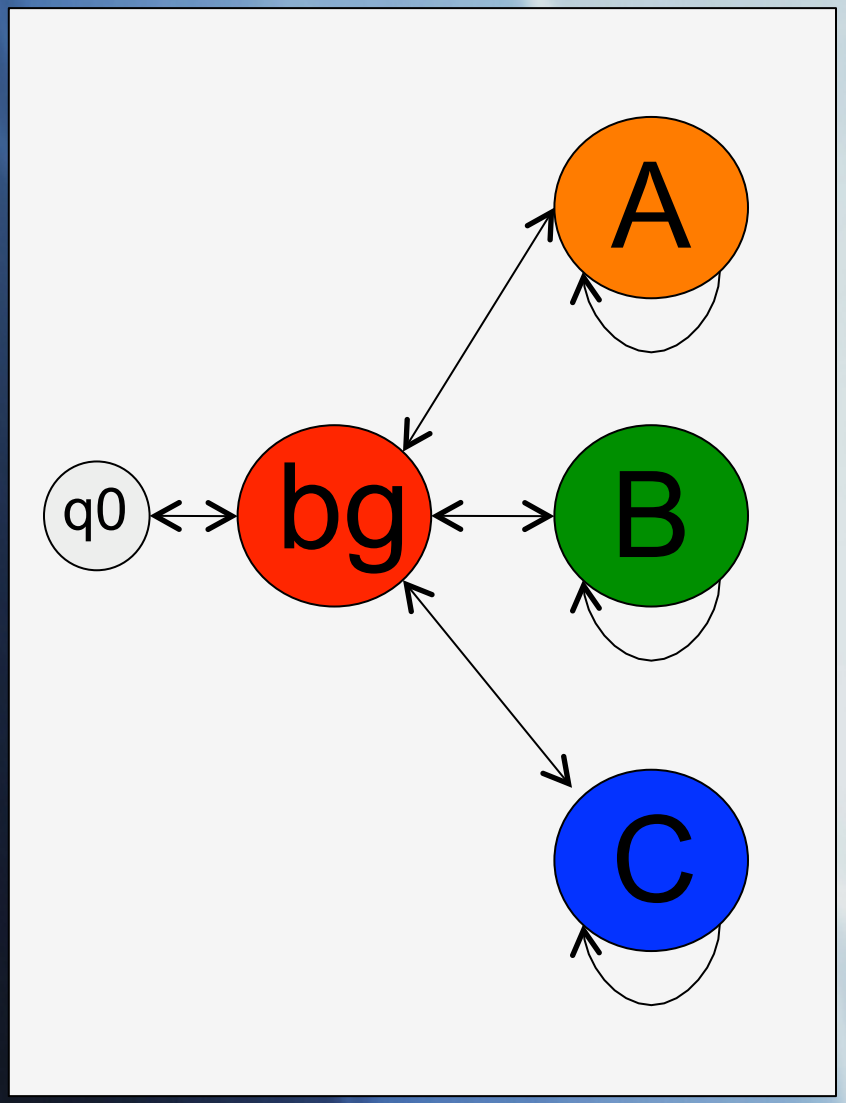
Posterior probability of state *q* & component *j* for observation *i*:

$$\rho_{qij} = \left(\frac{F_{qi} B_{qi}}{\sum_{q \in Q} F_{qi} B_{qi}} \right) \left(\frac{\lambda_{qj} \mathcal{N}(\mathbf{S}_i; \mu_j, \mathbf{C}_j)}{\sum_{j=0}^{m-1} \lambda_{qj} \mathcal{N}(\mathbf{S}_i; \mu_j, \mathbf{C}_j)} \right)$$

EM update formulas:

$$\mu_j \leftarrow \frac{\sum_{q \in Q} \sum_{i=0}^{L-1} \mathbf{S}_i \rho_{qij}}{\sum_{q \in Q} \sum_{i=0}^{L-1} \rho_{qij}} \qquad \mathbf{C}_j \leftarrow \frac{\sum_{q \in Q} \sum_{i=0}^{L-1} (\mathbf{S}_i - \mu_j)(\mathbf{S}_i - \mu_j)^T \rho_{qij}}{\sum_{q \in Q} \sum_{i=0}^{L-1} \rho_{qij}} \qquad \lambda_{qj} \leftarrow \frac{\sum_{i=0}^{L-1} \rho_{qij}}{\sum_{j=0}^{m-1} \sum_{i=0}^{L-1} \rho_{qij}}$$

Simulation: Chromatin Marks



9-dimensional continuous outputs:

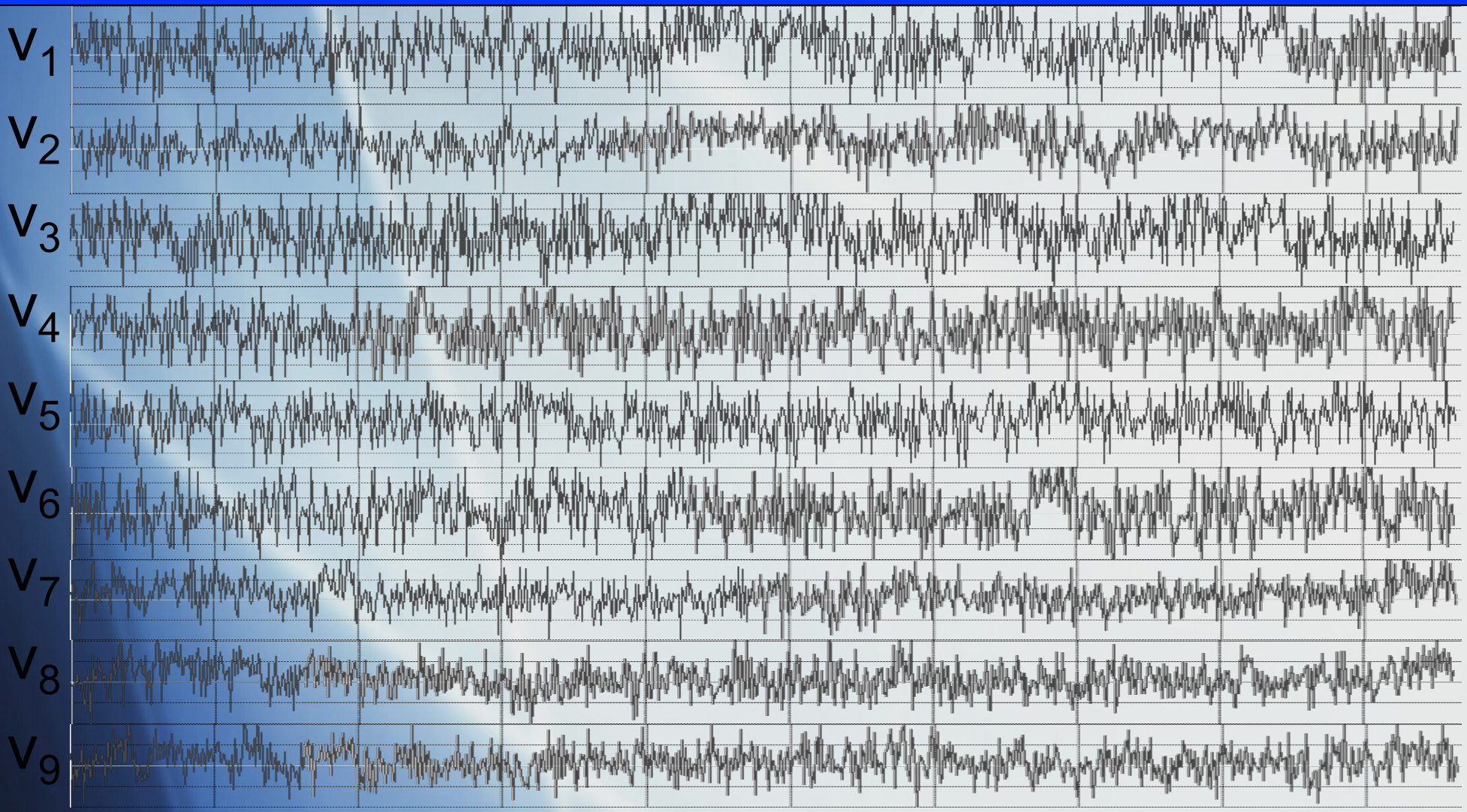
A tends to emit high values for marks v_1-v_3,

B tends to emit high values for marks v_4-v_6,

C tends to emit high values for marks v_7-v_9

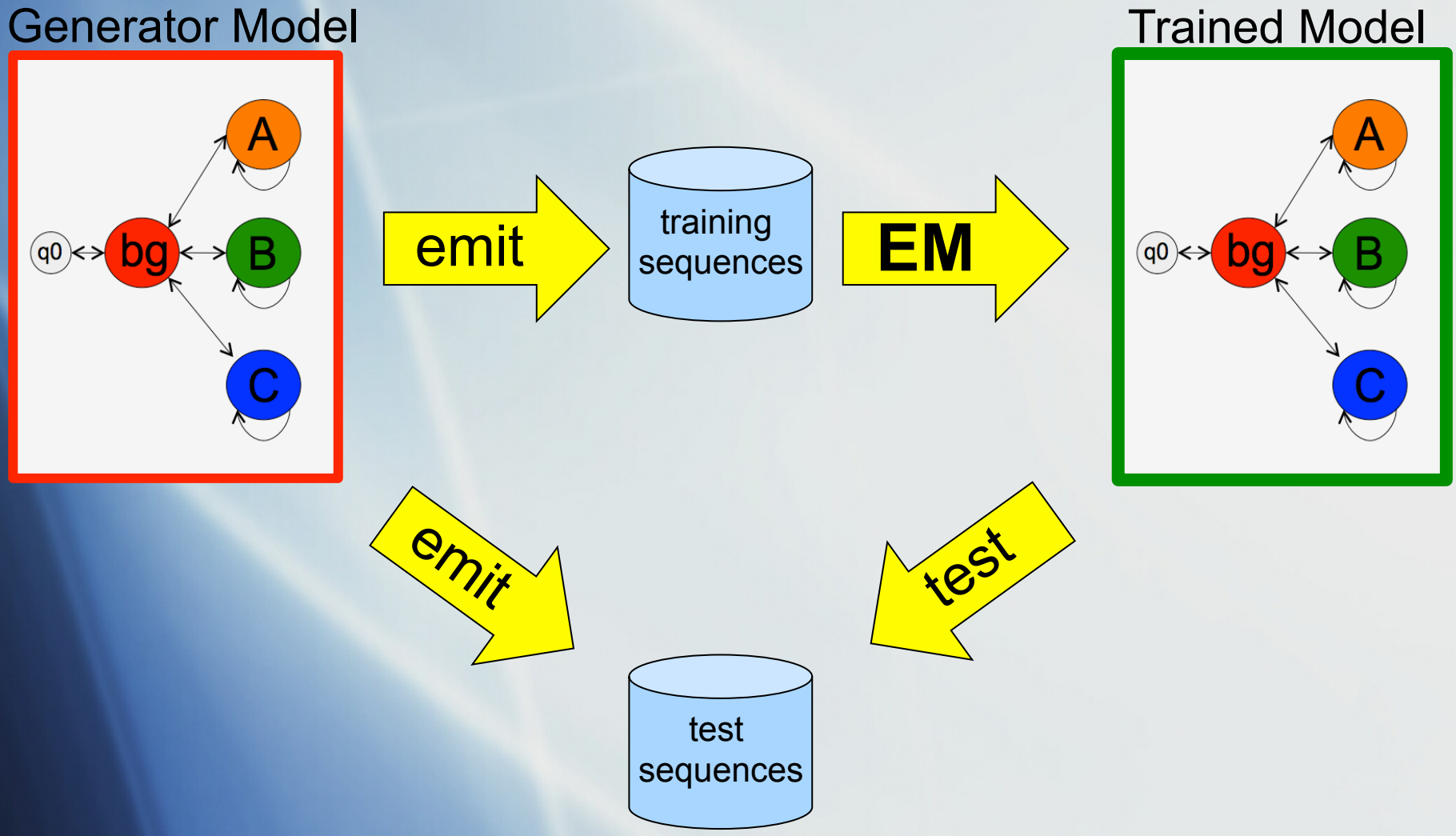
bg (background) is unbiased

Simulation Output

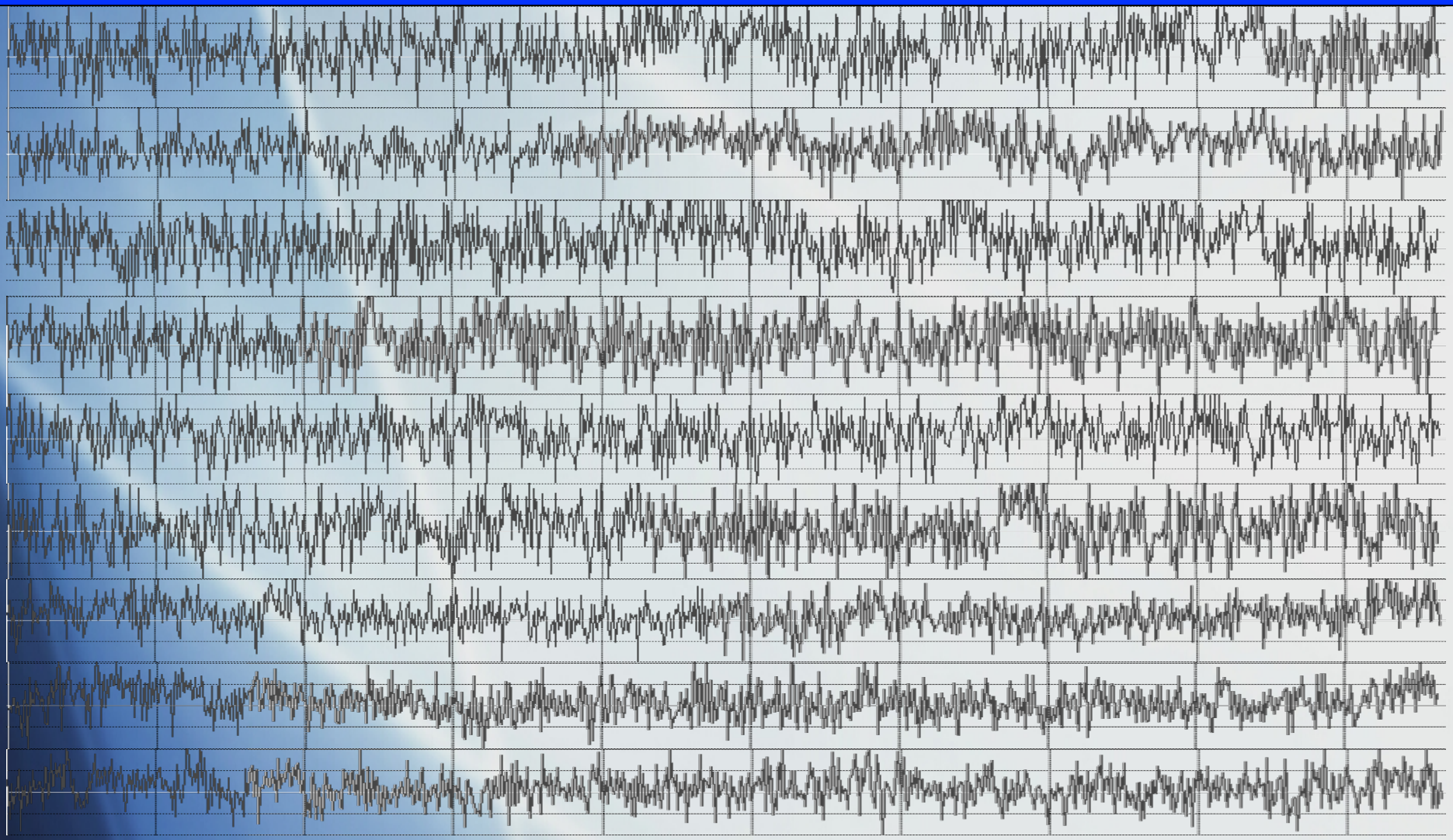


CORRECT PARSE:

Training a Model from the Simulated Data



Sample Output



CORRECT PARSE:

PREDICTED PARSE:

Summary

- 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 eipigentic sequence data, and they can be trained using EM

References

Expectation–maximization algorithm. (2011). In *Wikipedia, The Free Encyclopedia.*
http://en.wikipedia.org/wiki/Expectation–maximization_algorithm

Bilmes JA (1998) A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models.
http://crow.ee.washington.edu/people/bulyko/papers/em.pdf

Borman S. (2009) The Expectation Maximization Algorithm : A short tutorial.
http://www.seanborman.com/publications/EM_algorithm.pdf

Dempster AP, Laird NM, Rubin DB (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society B 39(1):1–38.
http://www.jstor.org/stable/2984875