CpG Islands - (Durbin Ch.3)

- In human genomes the C nucleotide of a dinucleotide CG is typically methylated
- Methyl-C has a high chance of mutating into a T
- Thus the dinucleotide CG (CpG) is under-represented
- Methylation is suppressed in some short stretches such as promoters and start regions of genes
- These areas are called CpG islands (higher frequency of CG)
- Questions:
  - Given a short stretch of genomic data, does it come from a CpG island?
  - Given a long piece of genomic data, does it contain CpG islands in it, where, what length?

General decoding problem

- Common theme: given a sequence from a certain alphabet suggest what is it?
  - gene, coding sequence, promoter area, CpG island . . .
- How can we determine if a given sequence x is a CpG island?
- Construct two data generating models H_0 ("ocean") and H_1 ("island")
  - which one is more likely to have generated the given data (classification problem)

LLR statistic and the Neyman-Pearson lemma

- Problem: decide whether a given data was generated under H_0 or H_1
- Solution: compute the LLR statistic

$$S(\boldsymbol{x}) = \log \frac{P_{H_1}(\boldsymbol{x})}{P_{H_0}(\boldsymbol{x})}$$

- Classify according to a predetermined threshold $(S(\boldsymbol{x}) > s_\alpha)$
- Neyman-Pearson: this test is optimal if H_0 and H_1 are *simple* hypotheses (as opposed to composite)
  - H_i is a simple hypothesis if $P_{H_0}(\boldsymbol{x})$ is well defined
  - For composite hypotheses the likelihood is replaced by a sup
- The optimality of the test:
  - for a given type I error $=$ probability of falsely rejecting H_0
  - the type II error $=$ probability of falsely accepting H_0 is minimized

Modeling CpG Islands - I

- For example, we can set both H_0 and H_1 to be Markov chains with different parametrization (transition probabilities)
- Learn the parameters from an annotated sample
  - if the sample is big enough use ML estimators:

$$a^+_{st} := \frac{c^+_{st}}{\sum_{t'} c^+_{st'}}$$

  - otherwise, smooth using a prior (add dummy counts)
- Based on 60,000 nucleotides:

+	A	C	G	T
A	.18	.27	.43	.12
C	.17	.37	.27	.19
G	.16	.34	.38	.12
T	. . .			

0	A	C	G	T
A	.30	.20	.29	.21
C	.32	.30	.08	.30
G	.25	.25	.30	.20
T	. . .			

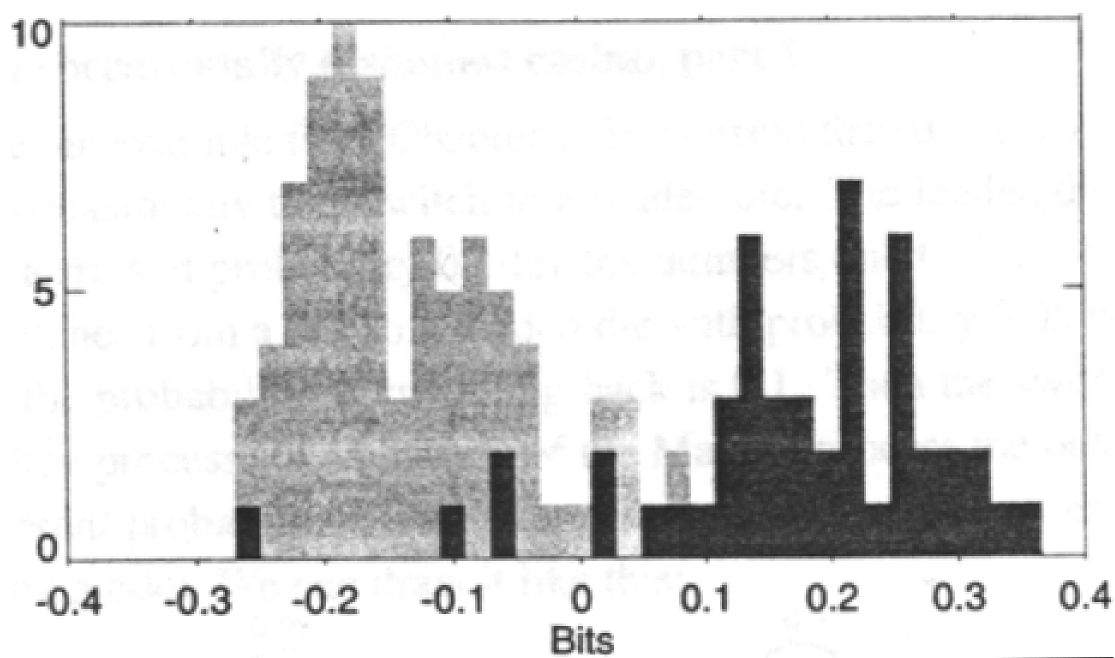
Modeling CpG Islands - I (cont.)

- Using the LLR statistic we have

$$S(\boldsymbol{x}) = \log \frac{P_{H_1}(\boldsymbol{x})}{P_{H_0}(\boldsymbol{x})} = \sum_i \log \frac{a^+_{x_{i-1}x_i}}{a^-_{x_{i-1}x_i}} = \sum_i \beta_{x_{i-1}x_i}$$

  where x_0 is an artificial start point: $a_{x_0 x_1} = P(x_1)$

- If $S(\boldsymbol{x}) > 1$ CpG island is more likely, otherwise no CpG island



Hidden Markov Models

- The occasionally dishonest casino
  - A casino uses a fair die most of the time
  - occasionally switches to a loaded one: $p_l(i) = \begin{cases} .5 & i = 6 \\ .1 & i \neq 6 \end{cases}$
  - Assume $P(\text{switch to loaded}) = 0.05$ and $P(\text{switch from loaded}) = 0.1$
- Let S_n denote the die used at the nth roll then SS is a Markov chain
  - which is hidden from us
  - we see only the outcomes which could have been "emitted" from either one of the states of the chain
- An example of a Hidden Markov Model (HMM)

Hidden Markov Models (cont.)

- More formally: $(\boldsymbol{S}, \boldsymbol{X})$ is an HMM if $\boldsymbol{S}$ is a Markov chain and

$$P(X_n = x | \boldsymbol{S}, X_1, \ldots, X_{n-1}, X_{n+1}, \ldots, X_L) = P(X_n = x | S_n) =: e_{S_n}(x)$$

  - $e_s(x) = P(X_i = x | S_i = s)$ are called the emission probabilities

- Application in communication:
  - message sent is $(s_1, \ldots, s_m)$
  - received $(x_1, \ldots, x_m)$
  - What is the most likely message sent?

- Speech recognition (HMM's origins)

- Claim. The joint probability is given by

$$P(\boldsymbol{S} = \boldsymbol{s}, \boldsymbol{X} = \boldsymbol{x}) = p(\boldsymbol{s}, \boldsymbol{x}) = \prod_{i=1}^{L} a_{s_{i-1} s_i} e_{s_i}(x_i),$$

HMM for CpG island

- States: $\{+,-\} \times \mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}$
- Emissions: $e_{+x}(y) = e_{-x}(y) = 1_{x=y}$
- All states are communicating with transition probabilities estimated from annotated sequences
- We are interested in *decoding* a given sequence $\boldsymbol{x}$: what is the most likely path that generated this sequence
- A path automatically yields annotation of CpG islands

The Viterbi algorithm

- Problem: given the parameters $\theta = (a_{st}, e_s)$ of an HMM and an emitted sequence $\boldsymbol{x}$, find

$$\boldsymbol{s}^* := \operatorname{argmax}_{\boldsymbol{s}} P(\boldsymbol{S} = \boldsymbol{s} | \boldsymbol{X} = \boldsymbol{x})$$

- Note that

$$\boldsymbol{s}^* = \operatorname{argmax}_{\boldsymbol{s}} P(\boldsymbol{S} = \boldsymbol{s} | \boldsymbol{X} = \boldsymbol{x}) P(\boldsymbol{X} = \boldsymbol{x}) = \operatorname{argmax}_{\boldsymbol{s}} p(\boldsymbol{s}, \boldsymbol{x})$$

- Let $E_{ik}(\boldsymbol{s}, \boldsymbol{x}) := \{\boldsymbol{S}_{1:i} = (\boldsymbol{s}_{1:i-1}, k), \boldsymbol{X}_{1:i} = \boldsymbol{x}_{1:i}\}$ and let $v_k(i) := \max_{\boldsymbol{s}} P[E_{ik}(\boldsymbol{s}, \boldsymbol{x})]$
- Claim. $v_l(i+1) = e_l(x_{i+1}) \max_k (v_k(i) a_{kl})$
- Note that this is a constructive recursive claim

The Viterbi algorithm (cont.)

- We add the special initial state 0
- Initialization: $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$
- For $i = 1 \dots L$ do, for each state l:
    - $v_l(i) = e_l(x_i) \max_k v_k(i-1)a_{kl}$
    - $\mathrm{ptr}_i(l) = \mathrm{argmax}_k\, v_k(i-1)a_{kl}$
- Termination:
    - $p(\boldsymbol{s}^*, \boldsymbol{x}) = \max_k v_k(L)$
- Traceback:
    - $\boldsymbol{s}^*_L = \mathrm{argmax}_k\, v_k(L)$
    - for $i = L, \dots, 2$: $\boldsymbol{s}^*_{i-1} = \mathrm{ptr}_i(\boldsymbol{s}^*_i)$

The Viterbi algorithm (cont.)

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

300 rolls of our casino $a_{FL} = 0.05$, $a_{LF} = 0.1$, $e_L(i) = \begin{cases} .5 & i = 6 \\ .1 & i \neq 6 \end{cases}$.

The forward algorithm for computing $p(x)$

- We want to compute $p(\boldsymbol{x}) = \sum_{\boldsymbol{s}} p(\boldsymbol{x}, \boldsymbol{s})$
- The number of summands grow exponentially with L
- Fortunately we have the *forward* algorithm based on:
  - Let $E_i(\boldsymbol{x}, k) := \{S_i = k, \boldsymbol{X}_{1:i} = \boldsymbol{x}_{1:i}\}$
  - Claim. $f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}$
- As in the Viterbi case, this is a constructive recursion:
  - Initialization: $f_0(0) := 1, f_k(0) := 0$ for $k > 0$
  - For $i = 1, \ldots, L$: $f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$
  - Termination: $p(\boldsymbol{x}) = \sum_k f_L(k)$
- By itself the forward algorithm is not that important
  - However it is an important for decoding: computing $P(S_i = k|\boldsymbol{x})$
  - e.g.: did you loose your house on a loaded die?

Posterior distribution of S_i

- What is $p_i(k|\boldsymbol{x}) := P(S_i = k|\boldsymbol{X} = \boldsymbol{x})$?
- Since we know $p(\boldsymbol{x})$, its suffices to find $P(S_i = k, \boldsymbol{X} = \boldsymbol{x})$:

$$
\begin{aligned}
P(S_i = k, \boldsymbol{X} = \boldsymbol{x}) &= P(S_i = k, \boldsymbol{X}_{1:i} = \boldsymbol{x}_{1:i}, \boldsymbol{X}_{i+1:L} = \boldsymbol{x}_{i+1:L}) \\
&= P(S_i = k, \boldsymbol{X}_{1:i} = \boldsymbol{x}_{1:i}) \times \\
&\qquad P(\boldsymbol{X}_{i+1:L} = \boldsymbol{x}_{i+1:L}|S_i = k, \boldsymbol{X}_{1:i} = \boldsymbol{x}_{1:i}) \\
&= f_k(i)\, \underbrace{P(\boldsymbol{X}_{i+1:L} = \boldsymbol{x}_{i+1:L}|S_i = k)}_{b_k(i)}
\end{aligned}
$$

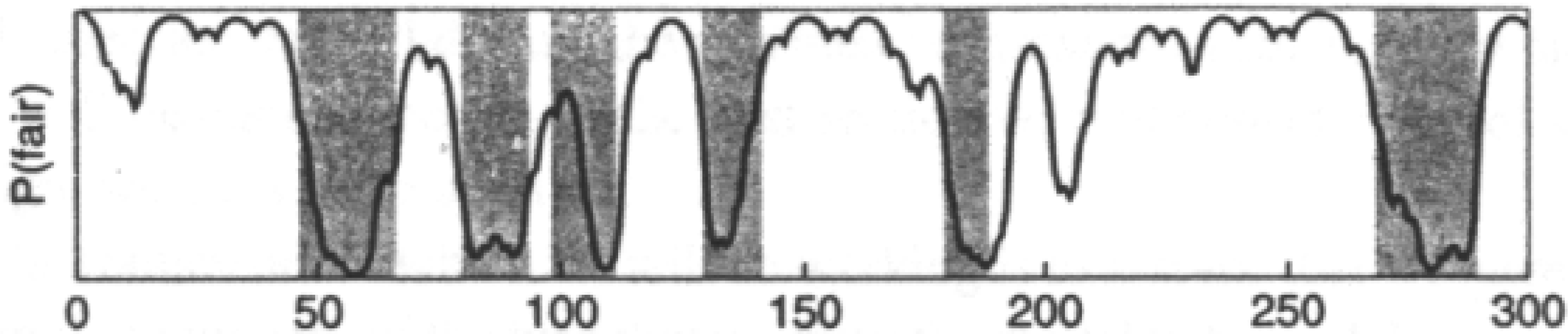
The backward algorithm

- The *backward* algorithm computes $b_k(i)$ based on
- Claim. $b_k(i) = \sum_l a_{kl} b_l(i+1) e_l(x_{i+1})$
- The algorithm:
  - Initialization: $b_k(L) = 1$ (more generally $b_k(L) = a_{k\triangle}$, where $\triangle$ is a terminating state)
  - For $j = L-1, \ldots, i$: $b_k(j) = \sum_l a_{kl} b_l(j+1) e_l(x_{j+1})$
- Finally,

$$p_i(k|\boldsymbol{x}) = \frac{f_k(i) b_k(i)}{p(\boldsymbol{x})}$$

- To compute $p_i(k|\boldsymbol{x})$ for all i, k, run both the backward and forward algorithms once storing $f_k(i)$ and $b_k(i)$ for all i, k.
- Complexity: let m be the number of states, space $O(mL)$, time $O(m^2 L)$

Decoding example



$p_i(F|\boldsymbol{x})$ for same $\boldsymbol{x}_{1:300}$ as in the previous graph. Shaded areas correspond to a loaded die. As before,

$$a_{FL} = 0.05,\ a_{LF} = 0.1,\ e_L(i) = \begin{cases} .5 & i = 6 \\ .1 & i \neq 6 \end{cases}.$$

More on posterior decoding

- More generally we might be interested in the expected value of some function of the path, $g(\boldsymbol{S})$ conditioned on the data $\boldsymbol{x}$.
- For example, if for the CpG HMM $g(\boldsymbol{s}) = 1_+(s_i)$, then

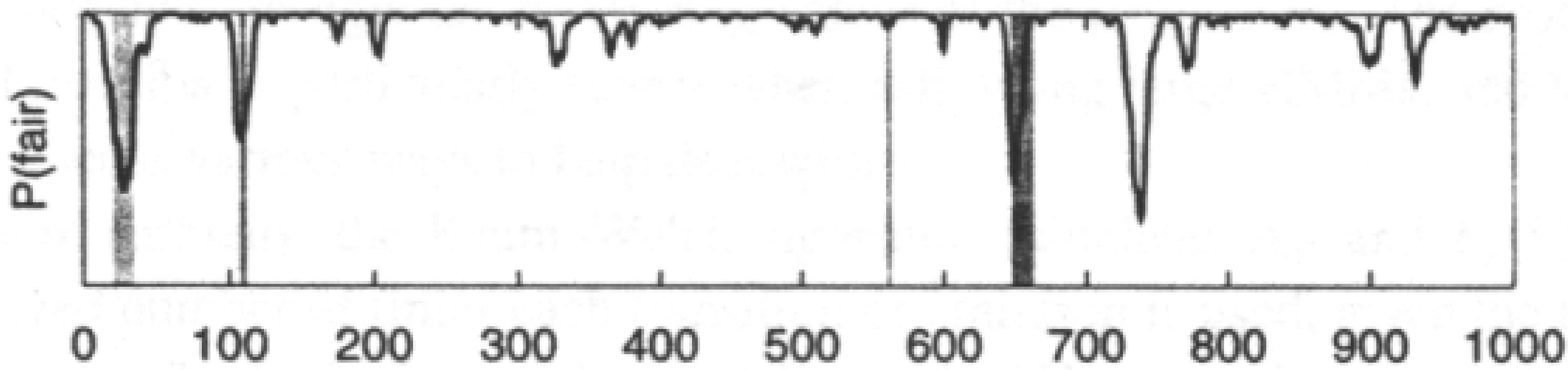
$$E[g(\boldsymbol{S})|\boldsymbol{x}] = \sum_k P(S_i = k^+|\boldsymbol{x}) = P(+|\boldsymbol{x})$$

- Comparing that with $P(-|\boldsymbol{x})$ we can find the most probable labeling for x_i
- We can do that for every i

More on posterior decoding/labeling

- This maximal posterior labeling procedure applies more generally when labeling defines a partition of the states
  - Warning: this is not the same as the most probable global labeling of a given sequence!
  - In *our example* the latter is given by the Viterbi algorithm
  - pp. 60-61 in Durbin compare the two approaches:
    - ▷ Same FN, posterior predicts more short FP

Decoding example



$p_i(F|\boldsymbol{x})$ for $\boldsymbol{x}_{1:300}$. Shaded areas correspond to a loaded die. Note that $\underline{a_{FL} = 0.01}$, $a_{LF} = 0.1$. Viterbi misses the loaded die altogether!

Parameter Estimation for HMMs

- An HMM model is defined by the parameters:

$$\Theta = \{a_{kl}, e_k(b) \; : \; \forall \text{ states } k, l \text{ and symbols } b\}$$

- We determine Θ using data, or a *training set* $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$, where $\boldsymbol{x}^j$ are *independent* samples generated by the model

- The *likelihood* of Θ given the data is

$$P(\cap_j \{\boldsymbol{X}^j = \boldsymbol{x}^j\} | \Theta) := P_\Theta(\cap_j \{\boldsymbol{X}^j = \boldsymbol{x}^j\}) = \prod_j P_\Theta(\boldsymbol{X}^j = \boldsymbol{x}^j)$$

- For better numerical stability we work with log-likelihood

$$l(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n | \Theta) = \sum_j \log P_\Theta(\boldsymbol{X}^j = \boldsymbol{x}^j)$$

- The maximum likelihood estimator of Θ is the value of Θ that maximizes the likelihood given the data.

Parameter Estimation for HMMs - special case

- Suppose our data is labeled in the sense that in addition to each $\boldsymbol{x}^j$ we are given the corresponding path $\boldsymbol{s}^j$
- In the CpG model this would correspond to having annotated sequences
- Can our framework handle the new data?
- Yes, the likelihood of Θ is, as before, the probability of the data assuming it was generated by the "model Θ":

$$l(\{\boldsymbol{x}^j, \boldsymbol{s}^j\}|\Theta) = \sum_j \log P_\Theta(\boldsymbol{X}^j = \boldsymbol{x}^j, \boldsymbol{S}^j = \boldsymbol{s}^j)$$

- The addition of the path information turns the ML estimation problem into a trivial one
  - Analogy: it is easier to compute $p(\boldsymbol{x}|\boldsymbol{s})$ than $p(\boldsymbol{x})$

MLE for HMMs when the path is given

- Let $A_{kl} = |\{(j,i) : s_{i-1}^j = k, s_i^j = l\}|$
- and $E_k(b) = |\{(j,i) : s_i^j = k, x_i^j = b\}|$, then

$$
\begin{aligned}
l(\{\boldsymbol{x}^j, \boldsymbol{s}^j\}|\Theta) &= \sum_j \log P_\Theta(\boldsymbol{X}^j = \boldsymbol{x}^j, \boldsymbol{S}^j = \boldsymbol{s}^j) \\
&= \sum_j \sum_i \log a_{s_{i-1}^j s_i^j} + \sum_j \sum_i \log e_{s_i^j}(x_i^j) \\
&= \sum_{k,l} A_{kl} \log a_{kl} + \sum_{k,b} E_k(b) \log e_k(b) \\
&= \sum_k \sum_l A_{kl} \log a_{kl} + \sum_k \sum_b E_k(b) \log e_k(b)
\end{aligned}
$$

- Thus,

$$
\sup_\Theta l(\{\boldsymbol{x}^j, \boldsymbol{s}^j\}|\Theta) = \sum_k \sup_{a_{kl}} \sum_l A_{kl} \log a_{kl} + \sum_k \sup_{e_k(b)} \sum_b E_k(b) \log e_k(b)
$$

MLE for HMMs when the path is given - cont.

- For each fixed k maximizing $\sum_l A_{kl} \log a_{kl}$ is subject to the constraint $\sum_l a_{kl} = 1$
- Can use Lagrange multipliers for the function $f(\boldsymbol{a}) = \sum_l A_l \log a_l$ and the constraint $g(\boldsymbol{a}) = \sum_l a_l = 1$ to get the ML estimates:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

- If the sample set is too small we add pseudocounts to the actual counts: we use $A'_{kl} = A_{kl} + r_{kl}$ and $E'_k(b) = E_k(b) + r_k(b)$
  - $r_{kl}, r_k(b) > 0$ and their magnitude reflects our prior biases
  - There is a natural Bayesian framework to justify this

MLE for HMMs when the path is *not* given

- No such elegant ML solution exists when the path is not given: simply computing $p(\boldsymbol{x})$ requires the forward algorithm
  - we resort to heuristics
- Had we known the path the problem would have been easy
- We can think of the path as "missing data"
- A general algorithm that works in this framework is the EM algorithm by Dempster Laird & Rubin (77)
- The Baum-Welch algorithm (72) is a particular example of EM applied to our case
- It is an iterative algorithm that *monotonically* converges to a *local* maximum of $l(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n | \Theta)$:
  - $l(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n | \Theta_m)$ increases with m

The Baum-Welch algorithm

- Suppose we had, Θ_0, an initial guess of Θ
- This Θ_0 would induce a conditional distribution
  - on the space of paths given the data, e.g.,

$$P(S_i^j = k | \Theta_0, \boldsymbol{X}^j = \boldsymbol{x}^j) = \frac{f_k(i) b_k(i)}{p(\boldsymbol{x}^j)}$$

    - ▷ where is Θ_0 on the rhs?
  - and on the joint space of state and emission given the data:

$$P(S_i^j = k, X_i^j = b | \Theta_0, \boldsymbol{X}^j = \boldsymbol{x}^j) = \frac{f_k(i) b_k(i)}{p(\boldsymbol{x}^j)} \cdot 1_{X_i^j = b}$$

    - ▷ from which we can deduce a conditional emission distribution per each state

The Baum-Welch algorithm - cont.

- We then replace our currently random counts A_{kl} and $E_k(b)$ by their *expected* value with respect to the above distribution
  - The E step in Expectation Maximization
- Next we update our guess of Θ by thinking about the expected counts as real counts
- More precisely, we maximize

$$l_{\Theta_0}(\{\boldsymbol{x}^j, \boldsymbol{s}^j\}|\Theta) := \sum_k \sum_l A_{kl} \log a_{kl} + \sum_k \sum_b E_k(b) \log e_k(b),$$

  where A_{kl} and $E_k(b)$ are the expected counts wrt Θ_0
  - The M step in Expectation Maximization
- Iterate E & M steps stopping according to a convergence criterion
- Claim. $l(\boldsymbol{x}^1, \dots, \boldsymbol{x}^n|\Theta_m)$ increases with m

The M-step

- Maximize

$$l_{\Theta_0}(\{\boldsymbol{x}^j, \boldsymbol{s}^j\}|\Theta) = \sum_k \sum_l A_{kl} \log a_{kl} + \sum_k \sum_b E_k(b) \log e_k(b)$$

- We already know how to solve this problem:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

The E-step

$$
\begin{aligned}
A_{kl} &= \sum_{j=1}^{n} \sum_{i=1}^{L} P(S_{i-1}^j = k, S_i^j = l | \Theta_0, \boldsymbol{X}^j = \boldsymbol{x}^j) \\
&= \sum_j \frac{1}{p(\boldsymbol{x}^j)} \sum_i P_{\Theta_0}(S_{i-1}^j = k, S_i^j = l, \boldsymbol{X}^j = \boldsymbol{x}^j) \\
&= \sum_j \frac{1}{p(\boldsymbol{x}^j)} \sum_i P_{\Theta_0}(S_{i-1}^j = k, \boldsymbol{X}_{1:i-1}^j = \boldsymbol{x}_{1:i-1}^j) \\
&\quad \times P_{\Theta_0}(S_i^j = l | S_{i-1}^j = k, \boldsymbol{X}_{1:i-1}^j = \boldsymbol{x}_{1:i-1}^j) \\
&\quad \times P_{\Theta_0}(X_i^j = x_i^j | S_i^j = l, S_{i-1}^j = k, \boldsymbol{X}_{1:i-1}^j = \boldsymbol{x}_{1:i-1}^j) \\
&\quad \times P_{\Theta_0}(X_{i+1:L}^j = x_{i+1:L}^j | S_i^j = l, S_{i-1}^j = k, \boldsymbol{X}_{1:i}^j = \boldsymbol{x}_{1:i}^j) \\
&= \sum_j \frac{1}{p(\boldsymbol{x}^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1)
\end{aligned}
$$

The E-step - cont.

- Finally,

$$E_k(b) = \sum_j \frac{1}{p(\boldsymbol{x}^j)} \sum_{i:x_i^j=b} f_k^j(i) b_k^j(i)$$

- Proof. HW exercise

The EM algorithm

- $\boldsymbol{x}$ is the observed data, $\boldsymbol{y}$ is the missing data
- Looking for Θ that will maximize $P_\Theta(\boldsymbol{X} = \boldsymbol{x})$
  - same as maximizing $\log p_\Theta(\boldsymbol{x})$
- "Readily" solvable if $\boldsymbol{y}$ was known, but it is not
- Solution: guess $\boldsymbol{y}$ then maximize Θ and use the new model to update your guess of $\boldsymbol{y}$
- More precisely your guess is distributional: many guesses weighted according to your belief in them
- It is technically beneficial to assume that $\boldsymbol{Y}$ is discrete

The EM algorithm - cont.

- Start with some Θ_0
- Θ_0 and the given data $\boldsymbol{x}$ induce a conditional distribution on $\boldsymbol{y}$:

$$p_{\Theta_0}(\boldsymbol{y}|\boldsymbol{x}) := P_{\Theta_0}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \boldsymbol{x})$$

- At the E-step we compute

$$\begin{aligned} E_{\Theta_0}[\log p_\Theta(\boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{X} = \boldsymbol{x}] &= E_{\Theta_0}[\log p_\Theta(\boldsymbol{x}, \boldsymbol{Y})|\boldsymbol{X} = \boldsymbol{x}] \\ &= \sum_{\boldsymbol{y}} \log p_\Theta(\boldsymbol{x}, \boldsymbol{y}) p_{\Theta_0}(\boldsymbol{y}|\boldsymbol{x}) \end{aligned}$$

- At the M-step we choose Θ that maximizes the conditional expectation we computed at the E-step:

$$\Theta_1 := \mathrm{argmax}_\Theta \sum_{\boldsymbol{y}} \log p_\Theta(\boldsymbol{x}, \boldsymbol{y}) p_{\Theta_0}(\boldsymbol{y}|\boldsymbol{x})$$

- Iterate the EM steps till desired numerical convergence

Properties of the EM algorithm

- Theorem. The likelihood increases monotonically

$$\log p_{\theta_{t+1}}(\boldsymbol{x}) \geq \log p_{\theta_t}(\boldsymbol{x})$$

- Proof. See notes
- Theorem. If

$$(\Theta, \Theta_0) \mapsto E_{\Theta_0}[\log p_\Theta(\boldsymbol{X}, \boldsymbol{Y}) | \boldsymbol{X} = \boldsymbol{x}]$$

is a continuous function of Θ and Θ_0, then for any sequence of Θ_k:

  - all its limit points are stationary points of $\Theta \mapsto \log p_\Theta(\boldsymbol{x})$
  - $\log p_{\Theta_k}(\boldsymbol{x})$ converges to a stationary value $L^* = \log p_{\Theta^*}(\boldsymbol{x})$ for some stationary point Θ^*
  - if L^* is uniquely attained then $\Theta_k \to \Theta^*$

- Proof. Wu (83)

Comments on the EM algorithm

- EM is a determinstic algorithm
- EM relies on the fact that maximizing

$$E_{\Theta_0}[\log p_\Theta(\boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{X} = \boldsymbol{x}] = \sum_{\boldsymbol{y}} \log p_\Theta(\boldsymbol{x}, \boldsymbol{y}) p_{\Theta_0}(\boldsymbol{y}|\boldsymbol{x})$$

  can be done either in closed form or in a relatively simple numerical calculation

- For convergence it suffices to choose Θ_{t+1} so that

$$E_{\Theta_t}[\log p_{\Theta_{t+1}}(\boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{X} = \boldsymbol{x}] > E_{\Theta_t}[\log p_{\Theta_t}(\boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{X} = \boldsymbol{x}]$$

  - Generalized EM

Potential EM pitfalls

- EM can converge to $L^* := \lim_k \log p_{\Theta_k}(\boldsymbol{x})$ which is not the value at a stationary point
- Even if $L^* = \log p_{\Theta^*}(\boldsymbol{x})$ where Θ^* is a stationary point, Θ_k might not converge to Θ^*
- Moreover, Θ^* can be a saddle point or. . . a local minimum!
- While the latter two are somewhat pathological cases, convergence to a local maximum is typically the reality for complex systems

Baum-Welch as EM

- The missing data is the path: $\boldsymbol{Y} = \boldsymbol{S}$
- The full log-likelihood function is

$$\log_{\Theta}(\boldsymbol{x}, \boldsymbol{s}) = \sum_{k} \sum_{l} A_{kl}(\boldsymbol{s}) \log a_{kl} + \sum_{k} \sum_{b} E_k(b, \boldsymbol{s}) \log e_k(b)$$

- Taking conditional expectation $E_{\Theta_t}(\cdot|\boldsymbol{X} = \boldsymbol{x})$ we get

$$E_{\Theta_t}[\log_{\Theta}(\boldsymbol{X}, \boldsymbol{S})|\boldsymbol{X} = \boldsymbol{x}] = \sum_{k} \sum_{l} E_{\Theta_t}[A_{kl}(\boldsymbol{S})|\boldsymbol{X} = \boldsymbol{x}] \log a_{kl} + \sum_{k} \sum_{b} E_{\Theta_t}[E_k(b, \boldsymbol{S})|\boldsymbol{X} = \boldsymbol{x}] \log e_k(b)$$

- Which is exactly the expression we maximized wrt $\Theta = \{a_{kl}, e_k(b)\}$