Modeling Biological Sequence and Hidden Markov Models

(part II - The algorithms)
Challenges in Computational Biology

4 Genome Assembly

9 Regulatory motif discovery

6 Gene Finding

2 Sequence alignment

3 Database search

14 Comparative Genomics

10 Evolutionary Theory

5 Gene expression analysis

4 Cluster discovery

9 Gibbs sampling

11 Protein network analysis

12 Regulatory network inference

13 Emerging network properties
Modeling biological sequences

- **Ability to generate** DNA sequences of a certain **type**
  - Not exact alignment to previously known gene
  - Preserving ‘properties’ of **type**, not identical sequence
- **Ability to recognize** DNA sequences of a certain type
  - What (hidden) state is most likely to have generated observations
  - Find set of states and transitions that generated a long sequence
- **Ability to learn** distinguishing characteristics of each type
  - Training our generative models on large datasets
  - Learn to classify unlabelled data
Markov Chains & Hidden Markov Models

- **Markov Chain**
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities

- **HMM**
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities
HMM nomenclature

\[ \pi \text{ is the (hidden) path} \]
\[ x \text{ is the (observed) sequence} \]

- Find path \( \pi^* \) that maximizes total joint probability \( P[x, \pi] \)

\[ P(x, \pi) = a_{0\pi_1}^* \prod_i e_{\pi_i}(x_i) \times a_{\pi_i\pi_{i+1}} \]

[Diagram of HMM nomenclature with nodes and transitions labeled for start, emission, and transition]
HMM for the dishonest casino model

transitions

\[ P(\pi_i=L|\pi_{i-1}=F) = 0.05 \]

\[ a_{FL} \]

\[ a_{FF} = 0.95 \]

\[ a_{LF} \]

\[ a_{LL} = 0.95 \]

emissions

\[ e_F(1) = P(x_i=1|\pi_i=F) = 1/6 \]
\[ e_F(2) = 1/6 \]
\[ e_F(3) = 1/6 \]
\[ e_F(4) = 1/6 \]
\[ e_F(5) = 1/6 \]
\[ e_F(6) = 1/6 \]

\[ e_L(1) = P(x_i=1|\pi_i=L) = 1/10 \]
\[ e_L(2) = 1/10 \]
\[ e_L(3) = 1/10 \]
\[ e_L(4) = 1/10 \]
\[ e_L(5) = 1/10 \]
\[ e_L(6) = 1/2 \]
HMM for CpG islands

• Build a single model that combines both Markov chains:
  – ‘+’ states: A+, C+, G+, T+
    • Emit symbols: A, C, G, T in CpG islands
  – ‘-’ states: A-, C-, G-, T-
    • Emit symbols: A, C, G, T in non-islands

• Emission probabilities distinct for the ‘+’ and the ‘-’ states
  – Infer most likely set of states, giving rise to observed emissions
    ➜ ‘Paint’ the sequence with + and - states

Question: Why do we need so many states?

In the Dishonest Casino we only had 2 states: Fair / Loaded
Why do we need 8 states here: 4 CpG+ / 4 CpG- ?
 ➜ Encode ‘memory’ of previous state: count nucleotide transitions!
The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - GIVEN a HMM $M$, a path $\pi$, and a sequence $x$,
   - FIND $\text{Prob}[x, \pi | M]$
   - “Running the model”, simply multiply emission and transition probabilities
   - Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the sequence $\pi^*$ of states that maximizes $P[x, \pi | M]$
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the total probability $P[x | M]$ summed across all paths
   - Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total prob that emission $x_i$ came from state $k$, across all paths
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the total probability $P[\pi_i = k | x, M]$
   - Posterior decoding: run forward & backward algorithms to & from state $\pi_i = k$

5. **Supervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., labeled sequence $x$,
   - FIND parameters $\theta = (e_i, a_{ij})$ that maximize $P[x | \theta]$
   - Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., unlabeled sequence $x$,
   - FIND parameters $\theta = (e_i, a_{ij})$ that maximize $P[x | \theta]$
   - Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
1. Scoring

Multiply emissions, transitions
1. Scoring

\[ P(p, x) = (a_{0,C+} * 1) \times (a_{C+,G-} * 1) \times (a_{G-,C-} * 1) \times (a_{C-,G+} * 1) \times (a_{G+,0}) \]

Probability of given path \( p \) & observations \( x \)
The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - GIVEN a HMM $M$, a path $\pi$, and a sequence $x$,
   - FIND $\text{Prob}[x, \pi | M]$
   - "Running the model", simply multiply emission and transition probabilities
   - Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the sequence $\pi^*$ of states that maximizes $P[x, \pi | M]$
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - GIVEN a HMM $M$, a sequence $x$
   - FIND the total probability $P[x | M]$ summed across all paths
   - Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission $x_i$ came from state $k$, across all paths
   - GIVEN a HMM $M$, a sequence $x$
   - FIND the total probability $P[\pi_i = k | x, M]$
   - Posterior decoding: run forward & backward algorithms to & from state $\pi_i = k$

5. **Supervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., labeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta]$
   - Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., unlabeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta]$
   - Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
2. Decoding:
How can we find the most likely path?

Viterbi algorithm
Finding most likely state path

- Given the observed emissions, what was the path?
Finding the most likely path

- Find path $\pi^*$ that maximizes total joint probability $P[ x, \pi ]$

$$P(x, \pi) = a_{0\pi_1} \times \prod_i e_{\pi_i}(x_i) \times a_{\pi_i\pi_i+1}$$

- start
- emission
- transition
Calculate maximum $P(x, \pi)$ recursively

- Assume we know $V_j$ for the previous time step (i-1)
- Calculate $V_k(i) = \text{current max } V_j(i-1) \times \text{max ending in state } j \text{ at step } i \times a_{jk}$
  - $V_k(i)$: Hidden states
  - $V_j(i-1)$: Previous state
  - $a_{jk}$: Transition from state $j$
  - $e_k(x_i)$: This emission
  - $x_{i-1}$: Previous observation
  - $x_i$: Current observation
  - $k$: Current state
The Viterbi Algorithm

Input: \( x = x_1 \ldots x_N \)

Initialization:
\[
V_0(0) = 1, \quad V_k(0) = 0, \quad \text{for all } k > 0
\]

Iteration:
\[
V_k(i) = e_k(x_i) \times \max_j a_{jk} V_j(i-1)
\]

Termination:
\[
P(x, \pi^*) = \max_k V_k(N)
\]

Traceback:
Follow max pointers back

In practice:
Use log scores for computation

Running time and space:
Time: \( O(K^2N) \)
Space: \( O(KN) \)
# The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - GIVEN a HMM M, a path \( \pi \), and a sequence \( x \),
   - FIND \( \text{Prob}[ x, \pi | M ] \)
   - **Running the model**, simply multiply emission and transition probabilities
   - Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - GIVEN a HMM M, and a sequence \( x \),
   - FIND the sequence \( \pi^* \) of states that maximizes \( P[ x, \pi | M ] \)
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - GIVEN a HMM M, a sequence \( x \)
   - FIND the total probability \( P[ x | M ] \) summed across all paths
   - **Forward algorithm**, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission \( x_i \) came from state \( k \), across all paths
   - GIVEN a HMM M, a sequence \( x \)
   - FIND the total probability \( P[ \pi_i = k | x, M ] \)
   - **Posterior decoding**: run forward & backward algorithms to & from state \( \pi_i = k \)

5. **Supervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM M, with unspecified transition/emission probs., labeled sequence \( x \),
   - FIND parameters \( \theta = (E_i, A_{ij}) \) that maximize \( P[ x | \theta ] \)
   - Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM M, with unspecified transition/emission probs., unlabeled sequence \( x \),
   - FIND parameters \( \theta = (E_i, A_{ij}) \) that maximize \( P[ x | \theta ] \)
   - Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
3. Model evaluation:
Total $P(x|M)$, summed over all paths

Forward algorithm
Simple: Given the model, generate some sequence $x$

Given a HMM, we can generate a sequence of length $n$ as follows:

1. Start at state $\pi_1$ according to prob $a_{0\pi_1}$
2. Emit letter $x_1$ according to prob $e_{\pi_1}(x_1)$
3. Go to state $\pi_2$ according to prob $a_{\pi_1\pi_2}$
4. … until emitting $x_n$
Complex: Given x, was it generated by the model?

Given a sequence x,
What is the probability that x was generated by the model (using any path)?

\[ P(x) = \sum_{\pi} P(x, \pi) \]

- **Challenge:** exponential number of paths

- **(cheap) alternative:**
  - Calculate probability over maximum (Viterbi) path \( \pi^* \)

- **(real) solution**
  - Calculate sum iteratively using dynamic programming
The Forward Algorithm – derivation

Define the forward probability:

\[ f_l(i) = P(x_1 \ldots x_i, \pi_i = l) \]

\[ = \sum_{\pi_1 \ldots \pi_{i-1}} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1}, \pi_i = l) e_l(x_i) \]

\[ = \sum_k \sum_{\pi_1 \ldots \pi_{i-2}} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1} = k) a_{kl} e_l(x_i) \]

\[ = \sum_k f_k(i-1) a_{kl} e_l(x_i) \]

\[ = e_l(x_i) \sum_k f_k(i-1) a_{kl} \]
Calculate total probability $\sum_{\pi} P(x, \pi)$ recursively

• Assume we know $f_j$ for the previous time step (i-1)

• Calculate $f_k(i) = \text{current max} \times \left( \sum_j (f_j(i-1) \times a_{jk}) \right)$

  - $e_k(x_i)$: this emission
  - $f_j(i-1)$: sum ending in state j at step i
  - $a_{jk}$: transition from state j
  - $\sum_j$: sum of every possible previous state j
The Forward Algorithm

Input: \( x = x_1 \ldots x_N \)

**Initialization:**
\[
f_0(0) = 1, \quad f_k(0) = 0, \quad \text{for all } k > 0
\]

**Iteration:**
\[
f_k(i) = e_k(x_i) \times \sum_j a_{jk} f_j(i-1)
\]

**Termination:**
\[
P(x, \pi^*) = \sum_k f_k(N)
\]

**In practice:**
- Sum of log scores is difficult
  \( \Rightarrow \) approximate \( \exp(1+p+q) \)
  \( \Rightarrow \) scaling of probabilities

**Running time and space:**
- Time: \( O(K^2N) \)
- Space: \( O(KN) \)
Summary

• Generative model
  – Hidden states
  – Observed sequence

• ‘Running’ the model
  – Generate a random sequence

• Observing a sequence
  – What is the most likely path generating it?
    • Viterbi algorithm
  – What is the total probability generating it?
    • Sum probabilities over all paths
    • Forward algorithm

• Next: Classification
  – What is the probability that “CGGTACG” came from CpG+ ?
The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - GIVEN a HMM $M$, a path $\pi$, and a sequence $x$,
   - FIND $\text{Prob}(x, \pi | M)$
   - “Running the model”, simply multiply emission and transition probabilities
   - Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the sequence $\pi^*$ of states that maximizes $P(x, \pi | M)$
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - GIVEN a HMM $M$, a sequence $x$,
   - FIND the total probability $P(x | M)$ summed across all paths
   - Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission $x_i$ came from state $k$, across all paths
   - GIVEN a HMM $M$, a sequence $x$,
   - FIND the total probability $P(\pi_i = k | x, M)$
   - Posterior decoding: run forward & backward algorithms to & from state $\pi_i = k$

5. **Supervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., labeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P(x | \theta)$
   - Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., unlabeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P(x | \theta)$
   - Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
4. State likelihood

Find the likelihood an emission $x_i$ is generated by a state
Calculate $P(\pi_7 = \text{CPG}+ \mid x_7 = \text{G})$

- **With no knowledge (no characters)**
  - $P(\pi_i = k) = \text{most likely state (prior)}$
  - Time spent in markov chain states

- **With very little knowledge (just that character)**
  - $P(\pi_i = k \mid x_i = \text{G}) = (\text{prior}) \times (\text{most likely emission})$
  - Emission probabilities adjusted for time spent

- **With knowledge of entire sequence (all characters)**
  - $P(\pi_i = k \mid x=\text{AGCGCG…GATTATCGTCGTA})$
  - Sum over all paths that emit ‘G’ at position 7
  - **Posterior** decoding
Motivation for the Backward Algorithm

We want to compute

\[ P(\pi_i = k \mid x), \] the probability distribution on the \( i^{th} \) position, given \( x \)

We start by computing

\[
\begin{align*}
P(\pi_i = k, x) &= P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_N) \\
&= P(x_1 \ldots x_i, \pi_i = k) \ P(x_{i+1} \ldots x_N \mid x_1 \ldots x_i, \pi_i = k) \\
&= P(x_1 \ldots x_i, \pi_i = k) P(x_{i+1} \ldots x_N \mid \pi_i = k)
\end{align*}
\]

Forward, \( f_k(i) \)  
Backward, \( b_k(i) \)
The Backward Algorithm – derivation

Define the backward probability:

\[ b_k(i) = P(x_{i+1} \ldots x_N \mid \pi_i = k) \]
\[ = \sum_{\pi_{i+1} \ldots \pi_N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1}, \ldots, \pi_N \mid \pi_i = k) \]
\[ = \sum_l \sum_{\pi_{i+1} \ldots \pi_N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1} = l, \pi_{i+2}, \ldots, \pi_N \mid \pi_i = k) \]
\[ = \sum_l e_l(x_{i+1}) a_{kl} \underbrace{\sum_{\pi_{i+1} \ldots \pi_N} P(x_{i+2}, \ldots, x_N, \pi_{i+2}, \ldots, \pi_N \mid \pi_{i+1} = l)}_{b_l(i+1)} \]
\[ = \sum_l e_l(x_{i+1}) a_{kl} b_l(i+1) \]
Calculate total end probability recursively

- Assume we know $b_i$ for the next time step (i+1)
- Calculate $b_k(i) = \underset{\text{current max}}{\text{current max}} \sum_l \left( e_l(x_{i+1}) \times a_{kl} \times b_l(i+1) \right)$

Where:
- $e_l(x_{i+1})$: next emission
- $a_{kl}$: transition to next state
- $b_l(i+1)$: prob sum from state l to end
- $\sum_l$: sum over all possible next states
**The Backward Algorithm**

Input: $x = x_1 \ldots x_N$

**Initialization:**

$b_k(N) = a_{k0}$, for all $k$

**Iteration:**

$b_k(i) = \sum_l e_l(x_{i+1}) a_{kl} b_l(i+1)$

**Termination:**

$P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$

**In practice:**

Sum of log scores is difficult

$\rightarrow$ approximate $\exp(1+p+q)$

$\rightarrow$ scaling of probabilities

**Running time and space:**

Time: $O(K^2 N)$

Space: $O(KN)$
Putting it all together: Posterior decoding

- $P(k) = P(\pi_i = k | x) = f_k(i)^* b_k(i) / P(x)$
  - Probability that $i^{th}$ state is $k$, given all emissions $x$
- Posterior decoding
  - Define most likely state for every of sequence $x$
  - $\pi^i = \operatorname{argmax}_k P(\pi_i = k | x)$
- Posterior decoding ‘path’ $\pi^i$
  - For classification, more informative than Viterbi path $\pi^*$
    - More refined measure of “which hidden states” generated $x$
  - However, it may give an invalid sequence of states
    - Not all $j \to k$ transitions may be possible
Summary

• Generative model
  – Hidden states
  – Observed sequence
• ‘Running’ the model
  – Generate a random sequence
• Observing a sequence
  – What is the most likely path generating it?
    • Viterbi algorithm
  – What is the total probability generating it?
    • Sum probabilities over all paths
    • Forward algorithm
• Classification
  – What is the probability that “CGGTACG” came from CpG+?
    • Forward + backward algorithm
  – What is the most probable state for every position
    • Posterior decoding
The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - **GIVEN** a HMM $M$, a path $\pi$, and a sequence $x$,
   - **FIND** $\text{Prob}[x, \pi | M]$
   - “Running the model”, simply multiply emission and transition probabilities
   - Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - **GIVEN** a HMM $M$, and a sequence $x$,
   - **FIND** the sequence $\pi^*$ of states that maximizes $P[x, \pi | M]$
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - **GIVEN** a HMM $M$, a sequence $x$
   - **FIND** the total probability $P[x | M]$ summed across all paths
   - Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission $x_i$ came from state $k$, across all paths
   - **GIVEN** a HMM $M$, a sequence $x$
   - **FIND** the total probability $P[\pi_i = k | x, M]$
   - Posterior decoding: run forward & backward algorithms to & from state $\pi_i = k$

5. **Supervised learning** = optimize parameters of a model given training data
   - **GIVEN** a HMM $M$, with unspecified transition/emission probs., labeled sequence $x$,
   - **FIND** parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta]$
   - Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - **GIVEN** a HMM $M$, with unspecified transition/emission probs., unlabeled sequence $x$,
   - **FIND** parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta]$
   - Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
5: Supervised learning

Estimate model parameters based on **labeled** training data
Two learning scenarios

Case 1. Estimation when the “right answer” is known

**Examples:**

**GIVEN:** a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

**GIVEN:** the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

Case 2. Estimation when the “right answer” is unknown

**Examples:**

**GIVEN:** the porcupine genome; we don’t know how frequent are the CpG islands there, neither do we know their composition

**GIVEN:** 10,000 rolls of the casino player, but we don’t see when he changes dice

**QUESTION:** Update the parameters $\theta$ of the model to maximize $P(x|\theta)$
Case 1. When the right answer is known

Given \( x = x_1 \ldots x_N \)

for which the true \( \pi = \pi_1 \ldots \pi_N \) is known,

**Define:**

\[
A_{kl} = \text{# times } k \rightarrow l \text{ transition occurs in } \pi
\]

\[
E_k(b) = \text{# times state } k \text{ in } \pi \text{ emits } b \text{ in } x
\]

We can show that the maximum likelihood parameters \( \theta \) are:

\[
a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}}
\]

\[
e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)}
\]
Case 1. When the right answer is known

**Intuition:** When we know the underlying states, 
Best estimate is the average frequency of transitions & emissions that occur in the training data

**Drawback:**
Given little data, there may be **overfitting:** 
P(x|\(\theta\)) is maximized, but \(\theta\) is unreasonable 
0 probabilities – VERY BAD

**Example:**
Given 10 casino rolls, we observe 
\(x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3\) 

Then:
\(a_{FF} = 1; \quad a_{FL} = 0\)
\(e_F(1) = e_F(3) = .2; \quad e_F(2) = .3; \quad e_F(4) = 0; \quad e_F(5) = e_F(6) = .1\)
Pseudocounts

Solution for small training sets:

Add pseudocounts

\[
A_{kl} = \# \text{ times } k \rightarrow l \text{ transition occurs in } \pi + r_{kl}
\]

\[
E_k(b) = \# \text{ times state } k \text{ in } \pi \text{ emits } b \text{ in } x + r_k(b)
\]

\(r_{kl}, r_k(b)\) are pseudocounts representing our prior belief

Larger pseudocounts ⇒ Strong prior belief

Small pseudocounts \((\varepsilon < 1)\): just to avoid 0 probabilities
Pseudocounts

**Example:** dishonest casino

We will observe player for one day, 500 rolls

Reasonable pseudocounts:

\[
\begin{align*}
  r_{OF} &= r_{OL} = r_{F0} = r_{L0} = 1; \\
  r_{FL} &= r_{LF} = r_{FF} = r_{LL} = 1; \\
  r_F(1) &= r_F(2) = \ldots = r_F(6) = 20 \quad \text{(strong belief fair is fair)} \\
  r_F(1) &= r_F(2) = \ldots = r_F(6) = 5 \quad \text{(wait and see for loaded)}
\end{align*}
\]

Above #s pretty arbitrary – assigning priors is an art
The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   
   - **GIVEN** a HMM M, a path \( \pi \), and a sequence x,
   
   - **FIND** \( \text{Prob}(x, \pi | M) \)

   ➔ “Running the model”, simply multiply emission and transition probabilities

   ➔ Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states

   - **GIVEN** a HMM M, and a sequence x,
   
   - **FIND** the sequence \( \pi^* \) of states that maximizes \( \text{P}(x, \pi | M) \)

   ➔ Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths

   - **GIVEN** a HMM M, a sequence x
   
   - **FIND** the total probability \( \text{P}(x | M) \) summed across all paths

   ➔ Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission \( x_i \) came from state \( k \), across all paths

   - **GIVEN** a HMM M, a sequence x
   
   - **FIND** the total probability \( \text{P}(\pi_i = k | x, M) \)

   ➔ Posterior decoding: run forward & backward algorithms to & from state \( \pi_i = k \)

5. **Supervised learning** = optimize parameters of a model given training data

   - **GIVEN** a HMM M, with unspecified transition/emission probs., labeled sequence x,
   
   - **FIND** parameters \( \theta = (E_i, A_{ij}) \) that maximize \( \text{P}(x | \theta) \)

   ➔ Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data

   - **GIVEN** a HMM M, with unspecified transition/emission probs., unlabeled sequence x,
   
   - **FIND** parameters \( \theta = (E_i, A_{ij}) \) that maximize \( \text{P}(x | \theta) \)

   ➔ Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate

   ➔ Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
6: Unsupervised learning

Estimate model parameters based on **unlabeled** training data
Learning case 2. When the right answer is unknown

We don’t know the true $A_{kl}$, $E_k(b)$

Idea:

- We estimate our “best guess” on what $A_{kl}$, $E_k(b)$ are
- We update the parameters of the model, based on our guess
- We repeat
Case 2. When the right answer is unknown

Starting with our best guess of a model $M$, parameters $\theta$:

Given $x = x_1 \ldots x_N$

for which the true $\pi = \pi_1 \ldots \pi_N$ is unknown,

We can get to a provably more likely parameter set $\theta$

Principle: EXPECTATION MAXIMIZATION

1. Estimate $A_{kl}, E_k(b)$ in the training data
2. Update $\theta$ according to $A_{kl}, E_k(b)$
3. Repeat 1 & 2, until convergence
Estimating new parameters

To estimate $A_{kl}$:

At each position $i$ of sequence $x$,

Find probability transition $k \rightarrow l$ is used:

$$P(\pi_i = k, \pi_{i+1} = l \mid x) = \frac{1}{P(x)} \times P(\pi_i = k, \pi_{i+1} = l, x_1...x_N) = \frac{Q}{P(x)}$$

where $Q = P(x_1...x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1}...x_N) = P(\pi_{i+1} = l, x_{i+1}...x_N \mid \pi_i = k) P(x_1...x_i, \pi_i = k) = P(\pi_{i+1} = l, x_{i+1}x_{i+2}...x_N \mid \pi_i = k) f_k(i) = P(x_{i+2}...x_N \mid \pi_{i+1} = l) P(x_{i+1} \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) f_k(i) = b_l(i+1) e_l(x_{i+1}) a_{kl} f_k(i)$

So:

$$P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x \mid \theta)}$$

(For one such transition, at time step $i \rightarrow i+1$)
Estimating new parameters

(Sum over all $k \to l$ transitions, at any time step $i$)

So,

$$A_{kl} = \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_i \frac{f_k(i) a_{kl} e_l(\xi_{i+1}) b_l(i+1)}{P(x \mid \theta)}$$

Similarly,

$$E_k(b) = \frac{1}{P(x)} \sum \{i \mid x_i = b\} f_k(i) b_k(i)$$
Estimating new parameters

(Sum over all training seqs, all k→l transitions, all time steps i)

If we have several training sequences, \(x^1, \ldots, x^M\), each of length \(N\),

\[
A_{kl} = \sum_x \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_x \sum_i \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_{l(i+1)}}{P(x \mid \theta)}
\]

Similarly,

\[
E_k(b) = \sum_x (1/P(x)) \sum_{\{i \mid x_i = b\}} f_k(i) b_k(i)
\]
The Baum-Welch Algorithm

**Initialization:**
Pick the best-guess for model parameters
(or arbitrary)

**Iteration:**
1. Forward
2. Backward
3. Calculate $A_{kl, E_k(b)}$
4. Calculate new model parameters $a_{kl, e_k(b)}$
5. Calculate new log-likelihood $P(x | \theta)$

GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION

Until $P(x | \theta)$ does not change much
The Baum-Welch Algorithm – comments

Time Complexity:

\[ \# \text{ iterations} \times O(K^2N) \]

- Guaranteed to increase the log likelihood of the model

\[ P(\theta \mid x) = \frac{P(x, \theta)}{P(x)} = \frac{P(x \mid \theta)}{P(x) P(\theta)} \]

- Not guaranteed to find globally best parameters

Converges to local optimum, depending on initial conditions

- Too many parameters / too large model: Overtraining
**Initialization:** Same

**Iteration:**
1. Perform Viterbi, to find $\pi^*$
2. Calculate $A_{kl}$, $E_k(b)$ according to $\pi^*$ + pseudocounts
3. Calculate the new parameters $a_{kl}$, $e_k(b)$

**Until convergence**

**Notes:**
- Convergence is guaranteed – Why?
- Does not maximize $P(x \mid \theta)$
- In general, worse performance than Baum-Welch
### The main questions on HMMs

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - GIVEN a HMM $M$, a path $\pi$, and a sequence $x$,
   - FIND $\text{Prob}[x, \pi | M]$
   ➜ “Running the model”, simply multiply emission and transition probabilities
   ➜ Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the sequence $\pi^*$ of states that maximizes $P[x, \pi | M]$
   ➜ Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - GIVEN a HMM $M$, a sequence $x$
   - FIND the total probability $P[x | M]$ summed across all paths
   ➜ Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission $x_i$ came from state $k$, across all paths
   - GIVEN a HMM $M$, a sequence $x$
   - FIND the total probability $P[\pi_i = k | x, M)$
   ➜ Posterior decoding: run forward & backward algorithms to & from state $\pi_i = k$

5. **Supervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., labeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta ]$
   ➜ Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., unlabeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta ]$
   ➜ Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   ➜ Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
The main questions on HMMs: Pop quiz

1. **Scoring** = Joint probability of a sequence and a path, given the model
   - GIVEN a HMM $M$, a path $\pi$, and a sequence $x$,
   - FIND $\text{Prob}[x, \pi | M]$
   - “Running the model”, simply multiply emission and transition probabilities
   - Application: “all fair” vs. “all loaded” comparisons

2. **Decoding** = parsing a sequence into the optimal series of hidden states
   - GIVEN a HMM $M$, and a sequence $x$,
   - FIND the sequence $\pi^*$ of states that maximizes $P[x, \pi | M]$
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. **Model evaluation** = total probability of a sequence, summed across all paths
   - GIVEN a HMM $M$, a sequence $x$
   - FIND the total probability $P[x | M]$ summed across all paths
   - Forward algorithm, sum score over all paths (same result as backward)

4. **State likelihood** = total probability that emission $x_i$ came from state $k$, across all paths
   - GIVEN a HMM $M$, a sequence $x$
   - FIND the total probability $P[\pi_i = k | x, M]$
   - Posterior decoding: run forward & backward algorithms to & from state $\pi_i = k$

5. **Supervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., labeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta]$
   - Simply count frequency of each emission and transition observed in the training data

6. **Unsupervised learning** = optimize parameters of a model given training data
   - GIVEN a HMM $M$, with unspecified transition/emission probs., unlabeled sequence $x$,
   - FIND parameters $\theta = (E_i, A_{ij})$ that maximize $P[x | \theta]$
   - Viterbi training: guess parameters, find optimal Viterbi path (#2), update parameters (#5), iterate
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
What have we learned?

- Generative model
  - Hidden states / Observed sequence
- ‘Running’ the model
  - Generate a random sequence
- Observing a sequence
  - What is the most likely path generating it?
    • Viterbi algorithm
  - What is the total probability generating it?
    • Sum probabilities over all paths
    • Forward algorithm
- Classification
  - What is the probability that “CGGTACG” came from CpG+?
    • Forward + backward algorithm
  - What is the most probable state for every position
    • Posterior decoding
- Training
  - Estimating parameters of the HMM
    - When state sequence is known
      • Simply compute maximum likelihood A and E
    - When state sequence is not known
      • Baum-Welch: Iterative estimation of all paths / frequencies
      • Viterbi training: Iterative estimation of best path / frequencies