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**

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

**HMM nomenclature**

- Find path \( \pi^* \) that maximizes total joint probability \( P( x, \pi ) \)
  \[
  P( x, \pi ) = \alpha_{\pi L} \prod_i \phi_i( x_i ) \times \alpha_{\pi_{i+1} L}
  \]

**HMM for the dishonest casino model**

- Transitions
  - \( a_{FL} = 0.05 \)
  - \( a_{LF} = 0.95 \)
  - \( a_{FF} = 0.95 \)
  - \( a_{LL} = 0.05 \)
- Emissions
  - \( e_F(1) = P( x=1 | \pi=F ) = 1/6 \)
  - \( e_L(1) = P( x=1 | \pi=L ) = 1/10 \)
  - \( e_L(2) = 1/10 \)
  - \( e_L(3) = 1/10 \)
  - \( e_L(4) = 1/10 \)
  - \( e_L(5) = 1/10 \)
  - \( e_L(6) = 1/2 \)
The main questions on HMMs

1. **Scoring**
   - Joint probability of a sequence and a path, given the model
     - Given an HMM \( M \), a path \( \pi \), and a sequence \( x \)
     - \( \text{Find } P(\pi, x | M) \)
     - \( \text{Application: } \text{all fair} \) vs. \( \text{all loaded} \) comparisons

2. **Decoding**
   - Parsing a sequence into the optimal series of hidden states
     - Given an HMM \( M \), a sequence \( x \)
     - \( \text{Find } \pi^* \) 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 an HMM \( M \), a sequence \( x \)
     - \( \text{Find } P(x | M) \)
     - \( \text{Application: } \text{all fair} \) vs. \( \text{all loaded} \) comparisons

4. **State likelihood**
   - Total probability that emission \( x \) came from state \( k \), across all paths
     - Given an HMM \( M \), a sequence \( x \)
     - \( \text{Find } P(x = k | M) \)
     - \( \text{Application: } \text{all fair} \) vs. \( \text{all loaded} \) comparisons

5. **Supervised learning**
   - Optimize parameters of a model given training data
     - Given an HMM \( M \), emission/transitions probs., labeled sequence \( x \)
     - \( \text{Find } \theta = (e_{ij}, a_{ij}) \) that maximize \( P(x | M) \)
     - \( \text{Viterbi training: } \text{guess parameters, find optimal Viterbi path } \pi^* \), update \( \theta \), iterate

6. **Unsupervised learning**
   - Optimize parameters of a model given training data
     - Given an HMM \( M \), emission/transitions probs., unlabeled sequence \( x \)
     - \( \text{Find } \theta = (e_{ij}, a_{ij}) \) that maximize \( P(x | M) \)
     - \( \text{Baum-Welch training: } \text{guess parameters, find optimal Viterbi path } \pi^* \), update \( \theta \), iterate

---

**HMM for CpG islands**

- Build a single model that combines both Markov chains:
  - \( '+' \) states: A, C, G, T in CpG islands
  - \( '-' \) states: A, C, G, T in non-islands
- Emission probabilities distinct for the \( '+' \) and \( '-' \) 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!
Finding most likely state path

- Given the observed emissions, what was the path?

\[
\begin{align*}
A & \rightarrow \\
T & \rightarrow \\
G & \rightarrow \\
C & \rightarrow \\
A & \rightarrow \\
T & \rightarrow \\
G & \rightarrow \\
C & \rightarrow \\
A & \rightarrow \\
T & \rightarrow \\
G & \rightarrow \\
C & \rightarrow \\
A & \rightarrow \\
T & \rightarrow \\
G & \rightarrow \\
C & \rightarrow \\
\end{align*}
\]

Finding the most likely path

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

\[
P(x, \pi) = a_0 \prod_{i=1}^{n} (a_i(x_i) \times a_{\pi(i+1)})
\]

Calculate maximum \( P(x, \pi) \) recursively

- Assume we know \( V_j \) for the previous time step \((i-1)\)
- Calculate \( V_k(i) = e_k(x_i) \times \max_j (a_{kj} V_j(i-1)) \)

The Viterbi Algorithm

- Input: \( x = x_1 \ldots x_N \)
- Initialization:
  \( V_0(0) = 1, V_k(0) = 0, \) for all \( k > 0 \)
- Iteration:
  \( V_k(i) = e_k(x_i) \times \max_j (a_{kj} V_j(i-1)) \)
- 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: \( O(K^2 N) \)
  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 \( x \)
   - 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 \( x \)
   - Viterbi algorithm, dynamic programming, max score over all paths, trace pointers find path

3. Model evaluation: Total \( P(x|M) \), summed over 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, and a sequence \( x \)
   - Posterior decoding: run forward & backward algorithms to & from state \( k \)

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 parameters, find optimal Viterbi path (#2), update parameters (#5), iterate

3. Model evaluation:
   Total \( P(x|M) \), summed over all paths

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

1. Start at state $\pi_1$ according to prob $a_{01}$
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?

- 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:

$$f(i) = P(x_1...x_i, \pi_i = l)$$

- Assume we know $f_j$ for the previous time step ($i-1$)
- Calculate $f_k(i) = e_k(x_i) \times \sum_j (a_{jk} \times f_j(i-1))$

Calculate total probability $\sum_\pi P(x, \pi)$ recursively

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 π, and a sequence x,
   - FIND Prob[x, π | M]
   - 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 π of states that maximizes Prob[x, π | 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 xi came from state k, across all paths
   - GIVEN a HMM M, a sequence x
   - FIND the total probability P[πi = k | x, M)
   - Posterior decoding: run forward & backward algorithms to & from state πi

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 θ = (Ei, Aij) that maximize P[x | θ]
   - 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 θ = (Ei, Aij) that maximize P[x | θ]
   - 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 xi is generated by a state

Calculate P(π7= CpG+ | x7=G)

- With no knowledge (no characters)
  - P(πi=k) = most likely state (prior)
  - Time spent in markov chain states
- With very little knowledge (just that character)
  - P(πi=k | xi=G) = (prior) * (most likely emission)
  - Emission probabilities adjusted for time spent
- With knowledge of entire sequence (all characters)
  - P(πi=k | x=AGCGCG…GATTATCGTCGTA)
  - Sum over all paths that emit ‘G’ at position 7
  \( \Rightarrow \) Posterior decoding

Motivation for the Backward Algorithm
We want to compute P(πi=k | x), the probability distribution on the i\(^{th}\) position, given x
We start by computing P(πi=k, x) = P(x1…xi, xi+1…xN | πi=k)
= P(xi, πi=k) P(xi+1…xN | x, πi=k)
\( \Rightarrow \) Calculate total end probability recursively
\( b_k(i) = \sum_l P(\pi_i = l | x, π_i = k) \)
### The Backward Algorithm

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

**Initialization:**
- \( b_k(N) = a_{k0}, \text{ for all } k \)

**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: \( O(K^2N) \)
- 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 the state \( k \) is given all emissions \( x \)
- Posterior decoding
- Define most likely state for every of sequence \( x \)
- \( \Rightarrow \) approximate \( \exp(1+p+q) \)
- Posterior decoding ‘path’ \( \pi \)
- 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 \( \pi \to k \) 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 ‘CGGTAG’ came from CpG+?
  - Forward + backward algorithm
  - What is the most probable state for every position
  - Posterior algorithm

### 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 \( \max P(\pi, x | M) \)
   - ‘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 \( P(\pi, x | M) \)
   - 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 \( P(\pi, x | M) \) 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
   - Estimate model parameters based on labeled training data

#### Two learning scenarios

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

**Examples:**
- a genomic region \( x = x_1 \ldots x_{1,000,000} \) where we have good (experimental) annotations of the CpG islands
- 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:**
- the porcupine genome; we don’t know how frequent are the CpG islands there, neither do we know their composition
- 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:

\[
a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}} \\
e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)}
\]

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.

Example:

Given 10 casino rolls, observe:

\[ x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3 \]


Then:

\[ a_{FF} = 1; \quad a_{FL} = 0 \]

\[ e_F(1) = e_F(3) = 0.2; \quad e_F(2) = 0.3; \quad e_F(4) = 0; \quad e_F(5) = e_F(6) = 0.1 \]

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 \( \Rightarrow \) strong prior belief.

Small pseudocounts \( (\epsilon < 1) \): just to avoid 0 probabilities.

Example: Dishonest casino

We will observe player for one day, 500 rolls.

Reasonable pseudocounts:

\[ r_{FF} = r_{FL} = r_{FF} = r_{F} = 1; \]

\[ r_{F} = 2; \quad r_F(2) = 0; \quad r_F(3) = 0; \quad r_F(4) = 5 \]

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 \( P(x|\pi,M) \).
   - "Running the model": simply multiply emission & 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 probability that emission \( x_i \) came from state \( k \), across all paths.
   - Given a HMM M, a sequence \( x \).
   - Find the total probability \( P(x_i = k|M) \) summed across all paths.
   - Posterior decoding: run forward & backward algorithms to & from state \( k \).

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 \( P(x|\theta,M) \).
   - Simply count frequency of each emission and transition observed in the training data.

6. Unsupervised learning: Estimate model parameters based on unlabeled training data.
   - Given a HMM M, with unspecified transition/emission probs., unlabeled sequence \( x \).
   - Find parameters \( \theta = (E_i, A_{ij}) \) that maximize \( P(x|\theta,M) \).
   - Viterbi training: guess parameters, find optimal Viterbi path \( \pi \), update parameters \( \theta \), iterate.
   - Baum-Welch training: guess, sum over all emissions/transitions \( \pi \), update \( \theta \), 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 $A_k, E_k(b)$

Idea:

• We estimate our "best guess" on what $A_k, E_k(b)$ 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 $x = x_1...x_n$

for which the true $\pi = \pi_1...\pi_n$ is unknown,

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

Principle: EXPECTATION MAXIMIZATION

1. Estimate $A_k, E_k(b)$ in the training data
2. Update $\theta$ according to $A_k, E_k(b)$
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:

$$P(\pi_i = k, \pi_{i+1} = l | x) = \frac{Q}{P(x)}$$

where

$$Q = P(x_1...x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1}...x_N) = P(\pi_{i+1} = l, x_{i+1}...x_N | \pi_i = k) P(x_1...x_i, \pi_i = k)$$

So:

$$P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \frac{f_k(i) a_k e(x_{i+1}) b_l(i+1)}{P(x | \theta)}$$

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

Estimating new parameters

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

So,

$$A_{kl} = \sum_{x} \sum_{i} P(\pi_i = k, \pi_{i+1} = l | x, 0) = \frac{\sum_{x} \sum_{i} f_k(i) a_k e(x_{i+1}) b_l(i+1)}{P(x | 0)}$$

Similarly,

$$E_k(b) = \frac{1}{P(x)} \sum_{x} \sum_{i} f_k(i) b_k(i)$$

(Sum over all training seqs, all $k \rightarrow l$ transitions, all time steps $i$)

If we have several training sequences, $x^1, ..., x^M$, each of length $N$,

$$A_{kl} = \sum_{x} \sum_{i} P(\pi_i = k, \pi_{i+1} = l | x, 0) = \frac{\sum_{x} \sum_{i} f_k(i) a_k e(x_{i+1}) b_l(i+1)}{P(x | 0)}$$

Similarly,

$$E_k(b) = \sum_{x} \left( \frac{1}{P(x)} \sum_{i} f_k(i) b_k(i) \right)$$

The Baum-Welch Algorithm

Initialization:

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

Iteration:

1. Forward
2. Backward
3. Calculate $A_k, E_k(b)$
4. Calculate new model parameters $a_k, e_k(b)$
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:

# iterations \times O(K^2 N)

• Guaranteed to increase the log likelihood of the model

P(\theta | x) = \text{total probability of a sequence, summed across all paths}

• Not guaranteed to find globally best parameters

Converges to local optimum, depending on initial conditions

• Too many parameters / too large model: Overtraining

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 P(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 \xi came from state \ik, across all paths
   - GIVEN a HMM M, a sequence x
   - FIND the total probability \pi = \{x, \xi | M\}
   - Posterior decoding: run forward & backward algorithms to & from state \ik
5. Transition likelihood = total probability of all emissions/transitions (#4), update (#5), iterate
   - GIVEN a HMM M, with unspecified transition/emission probs., unlabeled sequence x
   - FIND parameters \theta = (E, A) that maximize P(x | \theta)
   - Baum-Welch training: \text{guess, sum over all emissions/transitions (#4), update (#5), iterate}
6. 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, A) that maximize \Pi \pi
   - Viterbi training: guess parameters, find optimal Viterbi path (\hat{\pi}), update parameters (\hat{E}, \hat{A})
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate
   - Posterior decoding
   - 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

Alternative: Viterbi Training

Initialization: Same

Iteration:
1. Perform Viterbi, to find \hat{\pi}:
2. Calculate \hat{A}_{\hat{\pi}}(\hat{\xi}) according to \hat{\pi} + pseudocounts
3. Calculate the new parameters \hat{\theta}_{\hat{\pi}}(\hat{\xi})

Until convergence

Notes:
- Convergence is guaranteed – Why?
- Does not maximize P(x | \theta)
- In general, worse performance than Baum-Welch

What have we learned ?

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 P(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 \xi came from state \ik, across all paths
   - GIVEN a HMM M, a sequence x
   - FIND the total probability \pi = \{x, \xi | M\}
   - Posterior decoding: run forward & backward algorithms to & from state \ik
5. Transition likelihood = total probability of all emissions/transitions (#4), update (#5), iterate
   - GIVEN a HMM M, with unspecified transition/emission probs., unlabeled sequence x
   - FIND parameters \theta = (E, A) that maximize P(x | \theta)
   - Baum-Welch training: \text{guess, sum over all emissions/transitions (#4), update (#5), iterate}
6. 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, A) that maximize P(x | \theta)
   - Simple count frequency of each emission and transition observed in the training data
   - Viterbi training: guess parameters, find optimal Viterbi path (\hat{\pi}), update parameters (\hat{E}, \hat{A})
   - Baum-Welch training: guess, sum over all emissions/transitions (#4), update (#5), iterate

The main questions on HMMs