CSCI1950-Z
Computational Methods for Biology
Lecture 20

Ben Raphael
April 15, 2009
http://cs.brown.edu/courses/csci1950-z/

Outline

• Finish aCGH + HMM.
• Introduction to networks.
CGH Analysis (1)

Divide genome into segments of equal copy number

A model for CGH data

K states copy numbers

Emissions: Gaussians

Genome coordinate

- Homozygous Deletion (copy =0)
- Heterozygous Deletion (copy =1)
- Normal (copy =2)
- Duplication (copy >2)
Hidden Markov Models

Definition of a hidden Markov model

**Definition:** A hidden Markov model (HMM)

- **Alphabet** \( \Sigma = \{ b_1, b_2, \ldots, b_M \} \)
- **Set of states** \( Q = \{ 1, \ldots, K \} \)
- **Transition probabilities** between any two states
  
  \[
  a_{ij} = \text{transition prob from state } i \text{ to state } j \\
  a_{i1} + \ldots + a_{iK} = 1, \text{ for all states } i = 1\ldots K
  \]

- **Start probabilities** \( a_{0i} \)
  
  \[
  a_{01} + \ldots + a_{0K} = 1
  \]

- **Emission probabilities** within each state
  
  \[
  e_i(b) = P( x_i = b | \pi_i = k ) \\
  e_i(b_1) + \ldots + e_i(b_M) = 1, \text{ for all states } i = 1\ldots K
  \]
A HMM is memory-less

At each time step $t$, the only thing that affects future states is the current state $\pi_t$

$$P(\pi_{t+1} = k \mid "\text{whatever happened so far"}) = \frac{P(\pi_{t+1} = k \mid \pi_1, \pi_2, ..., \pi_t, x_1, x_2, ..., x_t)}{P(\pi_{t+1} = k \mid \pi_t)}$$

A parse of a sequence

Given a sequence $x = x_1, ..., x_N$, a parse of $x$ is a sequence of states $\pi = \pi_1, ..., \pi_N$
Likelihood of a Parse

Simply, **multiply all the orange arrows**!
(transition probs and emission probs)

The dishonest casino model

- FAIR
  - $P(1|F) = 1/6$
  - $P(2|F) = 1/6$
  - $P(3|F) = 1/6$
  - $P(4|F) = 1/6$
  - $P(5|F) = 1/6$
  - $P(6|F) = 1/6$

- LOADED
  - $P(1|L) = 1/10$
  - $P(2|L) = 1/10$
  - $P(3|L) = 1/10$
  - $P(4|L) = 1/10$
  - $P(5|L) = 1/10$
  - $P(6|L) = 1/2$
Question #1 – Evaluation

**GIVEN**

A sequence of rolls by the casino player:

1245264621461361366166466163661636163616515615115146123562344

**QUESTION**

How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs.

Prob = $1.3 \times 10^{-35}$

Question #2 – Decoding

**GIVEN**

A sequence of rolls by the casino player:

1245264621461361366166466163661636163616515615115146123562344

**QUESTION**

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs.  
This is what we want to solve for CGH analysis.
Question # 3 – Learning

**GIVEN**

A sequence of rolls by the casino player

```
1245526121461366166466163661636165511516123562344
```

**QUESTION**

How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs

---

**The three main questions on HMMs**

1. **Decoding**
   
   **GIVEN:** HMM $M$, and a sequence $x$,
   
   **FIND:** sequence $\pi$ of states that maximizes $P[ x, \pi | M ]$

2. **Evaluation**

   **GIVEN:** HMM $M$, and a sequence $x$,
   
   **FIND:** $\text{Prob}[ x | M ]$

3. **Learning**

   **GIVEN:** HMM $M$, with unspecified transition/emission probs. and a sequence $x$,
   
   **FIND** parameters $\theta = (e_i, a_{ij})$ that maximize $P[ x | \theta ]$
Problem 1: Decoding

Find the most likely parse of a sequence

Decoding

GIVEN $x = x_1 x_2 \ldots x_N$

Find $\pi = \pi_1, \ldots, \pi_N$

to maximize $P[x, \pi]$

$\pi^* = \arg \max_{\pi} P[x, \pi]$

Maximizes $a_{0\pi_1} e_{\pi_1}(x_1) a_{\pi_1\pi_2} \ldots a_{\pi_{N-1}\pi_N} e_{\pi_N}(x_N)^{x_N}$

*$Dynamic Programming!$

$V_k(i) = \max_{\pi_1, \ldots, \pi_{i-1}} P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i = k]$

= Prob. of most likely sequence of states ending at state $\pi_i = k$
The Viterbi Algorithm

Input: $x = x_1 \ldots x_n$

**Initialization:**
- $V_0(0) = 1$ (0 is the imaginary first position)
- $V_k(0) = 0$, for all $k > 0$

**Iteration:**
- $V_j(i) = e_j(x_i) \times \max_k a_{kj} V_k(i – 1)$
- $\text{Ptr}_j(i) = \text{argmax}_k a_{kj} V_k(i – 1)$

**Termination:**
- $P(x, \pi^*) = \max_k V_k(N)$

**Traceback:**
- $\pi_n^* = \text{argmax}_i V_i(N)$
- $\pi_{i-1}^* = \text{Ptr}_{i}(i)$

Problem 2: Evaluation

Find the likelihood a sequence is generated by the model
Generating a sequence by the model

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

1. Start at state $\pi_1$ with probability $a_{0\pi_1}$
2. Emit letter $x_1$ with probability $e_{\pi_1}(x_1)$
3. Go to state $\pi_2$ with probability $a_{\pi_1\pi_2}$
4. ... until emitting $x_n$

A couple of questions

Given a sequence $x$,

- What is the probability that $x$ was generated by the model?
- Given a position $i$, what is the most likely state that emitted $x_i$?

Example: the dishonest casino

Say $x = 12341...23162616364616234112...21341$

Most likely path: $\pi = FF......F$

However: marked letters more likely to be L than unmarked letters
Evaluation

We will develop algorithms that allow us to compute:

- $P(x)$: Probability of $x$ given the model
- $P(x_i...x_j)$: Probability of a substring of $x$ given the model
- $P(\pi_i = k \mid x)$: Probability that the $i^{th}$ state is $k$, given $x$

A more refined measure of which states $x$ may be in

The Forward Algorithm

We want to calculate

$P(x) = \text{probability of } x, \text{ given the HMM}$

Sum over all possible ways of generating $x$:

$$P(x) = \sum_{\pi} P(x, \pi) = \sum_{\pi} P(x \mid \pi) P(\pi)$$

To avoid summing over an exponential number of paths $\pi$, define

$$f_k(i) = P(x_1...x_i, \pi_i = k) \quad \text{(the forward probability)}$$
The Forward Algorithm – derivation

Define the forward probability:

\[ f_k(i) = P(x_1...x_i, \pi_i = k) \]

\[ = \sum_{i=1}^{i-1} P(x_1...x_{i-1}, \pi_1,..., \pi_{i-1}, \pi_i = k) e_k(x_i) \]

\[ = \sum_{i} \sum_{i=1}^{i-2} P(x_1...x_{i-1}, \pi_1,..., \pi_{i-2}, \pi_{i-1} = l) a_{lk} e_k(x_i) \]

\[ = \sum_{i} P(x_1...x_{i-1}, \pi_{i-1} = l) a_{lk} e_k(x_i) \]

\[ = e_k(x_i) \sum_{j} f_j(i-1) a_{lk} \]

The Forward Algorithm

We can compute \( f_k(i) \) for all \( k, i \), using dynamic programming!

**Initialization:**

\[ f_0(0) = 1 \]

\[ f_k(0) = 0, \text{ for all } k > 0 \]

**Iteration:**

\[ f_k(i) = e_k(x_i) \sum f_j(i-1) a_{lk} \]

**Termination:**

\[ P(x) = \sum_k f_k(N) a_{k0} \]

Where, \( a_{k0} \) is the probability that the terminating state is \( k \) (usually = \( a_{0k} \))
### 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

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

**Forward, \( f_k(i) \)**  **Backward, \( b_k(i) \)**

Then, \( P(\pi_i = k \mid x) = P(\pi_i = k, x) / P(x) \)
The Backward Algorithm – derivation

Define the backward probability:

\[ b_k(i) = P(x_{i+1}...x_N \mid \pi_i = k) \]

\[ = \sum_{\pi_{i+1}...\pi_N} P(x_{i+1}, x_{i+2}, ..., x_N, \pi_{i+1}, ..., \pi_N \mid \pi_i = k) \]

\[ = \sum_{\pi_{i+1}...\pi_N} P(x_{i+1}, x_{i+2}, ..., x_N, \pi_{i+1} = l, \pi_{i+2}, ..., \pi_N \mid \pi_i = k) \]

\[ = \sum_{l} e_l(x_{i+1}) a_{kl} \sum_{\pi_{i+1}...\pi_N} P(x_{i+2}, ..., x_N, \pi_{i+2}, ..., \pi_N \mid \pi_{i+1} = l) \]

\[ = \sum_{l} e_l(x_{i+1}) a_{kl} b_l(i+1) \]

The Backward Algorithm

We can compute \( b_k(i) \) for all \( k, i \), using dynamic programming

**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:**

\( P(x) = \sum_{i} a_{0i} e_i(x_1) b_i(1) \)
Computational Complexity

What is the running time, and space required, for Forward, and Backward?

Time: $O(K^2N)$
Space: $O(KN)$

K = number of states
N = length of sequence

Useful implementation technique to avoid underflows

Viterbi: sum of logs
Forward/Backward: rescaling at each position by multiplying by a constant

Posterior Decoding

We can now calculate

$$P(\pi_i = k \mid x) = \frac{f_k(i) b_k(i)}{P(x)}$$

Then, we can ask

What is the most likely state at position $i$ of sequence $x$:

Define $\pi^*$ by Posterior Decoding:

$$\pi^*_i = \arg\max_k P(\pi_i = k \mid x)$$
Posterior Decoding

- For each state,
  - Posterior Decoding gives us a curve of likelihood of state for each position
  - That is sometimes more informative than Viterbi path $\pi^*$
- Posterior Decoding may give an invalid sequence of states
  - Why?

\[
P(\pi_i = k \mid x) = \sum_\pi P(\pi \mid x) \mathbb{1}(\pi_i = k) = \sum_{\{\pi: \pi[i] = k\}} P(\pi \mid x)
\]
**Viterbi, Forward, Backward**

<table>
<thead>
<tr>
<th>VITERBI</th>
<th>FORWARD</th>
<th>BACKWARD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization:</strong></td>
<td><strong>Initialization:</strong></td>
<td><strong>Initialization:</strong></td>
</tr>
<tr>
<td>$V_i(0) = 1$</td>
<td>$f_i(0) = 1$</td>
<td>$b_k(N) = a_{k0}$, for all $k$</td>
</tr>
<tr>
<td>$V_i(0) = 0$, for all $k &gt; 0$</td>
<td>$f_i(0) = 0$, for all $k &gt; 0$</td>
<td></td>
</tr>
<tr>
<td><strong>Iteration:</strong></td>
<td><strong>Iteration:</strong></td>
<td><strong>Iteration:</strong></td>
</tr>
<tr>
<td>$V_i(i) = e_i(x_i) \max_k V_k(i-1) a_{i0}$</td>
<td>$f_i(i) = e_i(x_i) \sum_k f_k(i-1) a_{i0}$</td>
<td>$b_i(i) = e_i(x_{i+1}) a_{0i} b_{i}(i+1)$</td>
</tr>
<tr>
<td><strong>Termination:</strong></td>
<td><strong>Termination:</strong></td>
<td><strong>Termination:</strong></td>
</tr>
<tr>
<td>$P(x, \pi^*) = \max_k V_k(N)$</td>
<td>$P(x) = \sum_k f_k(N) a_{k0}$</td>
<td>$P(x) = \sum_k a_{0k} e_k(x_1) b_k(1)$</td>
</tr>
</tbody>
</table>

**Problem 3: Learning**

Re-estimate the parameters of the model based on training data
Two learning scenarios

1. Estimation when the “right answer” is known

Examples:
- **GIVEN:** a genomic region \( x = x_1 \cdots x_{1,000,000} \) where we have good (experimental) annotations of copy numbers
- **GIVEN:** the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

2. Estimation when the “right answer” is unknown

Examples:
- **GIVEN:** tumor sample; we don’t know how the frequency and length of segments
- **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 \( P(x|\theta) \)

---

1. When the right answer is known

Given \( x = x_1 \cdots x_N \)
for which the true \( \pi = \pi_1 \cdots \pi_N \) is known,

**Define:**
- \( A_{kl} = \) # times \( k \rightarrow l \) transition occurs in \( \pi \)
- \( E_k(b) = \) # times state \( k \) in \( \pi \) emits \( b \) in \( x \)

We can show that the maximum likelihood parameters \( \theta \) (maximize \( P(x|\theta) \)) are:

\[
\begin{align*}
a_{kl} &= \frac{A_{kl}}{\sum_k A_{kl}} \\
e_k(b) &= \frac{E_k(b)}{\sum_c E_k(c)}
\end{align*}
\]
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|θ) is maximized, but θ 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:
\[a_{FL} = 1; \quad a_{FF} = 0\]
\[e_{f}(1) = e_{f}(3) = .2;\]
\[e_{f}(2) = .3; \quad e_{f}(4) = 0; \quad e_{f}(5) = e_{f}(6) = .1\]

---

**Pseudocounts**

Solution for small training sets:

Add pseudocounts

\[a_{kl} = \text{# times } k \rightarrow l \text{ transition occurs in } \pi + r_{kl}\]
\[E_{k}(b) = \text{# times state } k \text{ in } \pi \text{ emits } b \text{ in } x + r_{k}(b)\]

\[r_{kl}, r_{k}(b) \text{ are pseudocounts representing our prior belief}\]

Larger pseudocounts ⇒ Strong prior belief

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

**Example:** dishonest casino

We will observe player for one day, 600 rolls

Reasonable pseudocounts:

- \( r_{OF} = r_{OL} = r_{FO} = r_{LO} = 1 \);
- \( r_{FL} = r_{LF} = r_{FF} = r_{LL} = 1 \);
- \( r_f(1) = r_f(2) = \ldots = r_f(6) = 20 \) (strong belief fair is fair)
- \( r_l(1) = r_l(2) = \ldots = r_l(6) = 5 \) (wait and see for loaded)

Above #s pretty arbitrary – assigning priors is an art

---

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
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 \( \theta \)

i.e., \( \theta \) that increases the probability \( P(x \mid \theta) \)

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 \( A_{ii} \): (assume \( | \theta_{\text{CURRENT}} \) in all formulas below)

At each position \( i \) of sequence \( x \), find probability transition \( k \rightarrow l \) is used:

\[
P(\pi_i = k, \pi_{i+1} = l \mid x) = \frac{1}{P(x)} \times P(\pi_i = k, \pi_{i+1} = l, x_1 \ldots x_n) = \frac{Q}{P(x)}
\]

where \( Q = P(x_1 \ldots x_N, \pi_i = k, \pi_{i+1} = l, x_{i+2} \ldots x_N) = \)

\[
= P(\pi_{i+1} = l, x_{i+2} \ldots x_N \mid \pi_i = k) P(x_1 \ldots x_n, \pi_i = k) = \]

\[
= P(\pi_{i+1} = l, x_{i+2} \ldots x_N \mid \pi_i = k) f_i(i) = \]

\[
= P(x_{i+1} \mid \pi_{i+1} = l) P(x_{i+2} \ldots x_N \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) f_i(i) = \]

\[
= b_{i}(i+1) e_i(x_{i+1}) a_{kl} f_i(i)
\]

So:

\[
P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_i(i) a_{kl} e_i(x_{i+1}) b_{i}(i+1)}{P(x \mid \theta_{\text{CURRENT}})}
\]
Estimating new parameters

- So, $A_{uk}$ is the $E[\text{# times transition } k \rightarrow l, \text{ given current } \theta]$

$$A_{uk} = \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_i \frac{f_i(i) a_{kl} e_i(x_{i+1}) b_{l(i+1)}}{p(x \mid \theta)}$$

- Similarly,

$$E_k(b) = \frac{1}{P(x \mid \theta)} \sum \{i \mid x_i = b\} f_i(i) b_{l(i)}$$

The Baum-Welch Algorithm

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

**Iteration:**
1. Forward
2. Backward
3. Calculate $A_{ul} E_u(b)$, given $\theta_{\text{CURRENT}}$
4. Calculate new model parameters $\theta_{\text{NEW}} : a_{ul} e_u(b)$
5. Calculate new log-likelihood $P(x \mid \theta_{\text{NEW}})$

GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION

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

Time Complexity:

# iterations \times O(K^2N)

- Guaranteed to increase the log likelihood \( P(x \mid \theta) \)
- Not guaranteed to find globally best parameters

Converges to local optimum, depending on initial conditions
- Too many parameters / too large model: Overtraining

Alternative: Viterbi Training

Initialization: Same

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

Until convergence

Notes:
- Not guaranteed to increase \( P(x \mid \theta) \)
- Guaranteed to increase \( P(x, \mid \theta, \pi^*) \)
- In general, worse performance than Baum-Welch
A model for CGH data

K states
copy numbers

\( S_1 \)

Homozygous
Deletion
(copy =0)

\( \mu_1, \sigma_1 \)

Emissions:
Gaussians

\( S_2 \)

Heterozygous
Deletion
(copy =1)

\( \mu_2, \sigma_2 \)

\( S_3 \)

Normal
(copy =2)

\( \mu_3, \sigma_3 \)

\( S_4 \)

Duplication
(copy >2)

\( \mu_4, \sigma_4 \)

Use Viterbi algorithm to derive segmentation.
Forward/backward to compute state of a single probe.

CGH Segmentation: Model Selection

How many states copy number states \( K \)?
Larger \( K \):
1. Better fit to observed data
2. More parameters to estimate

Avoid overfitting by model selection.

Let \( \theta = (A, B, \pi) \) be parameters for HMM.

Try different \( k = 1, ..., K_{\text{max}} \)

Compute \( L(\theta | O) \) by dynamic programming
(forward-backward algorithm)

Calculate:

\[ \psi(k) = -\log (L(\theta | O)) + q_k D(N)/N \]

\( N = \) number of probes (data points)
\( q_k = \) number of parameters
\( D(N) = 2 \) (AIC) or \( D(N) = \log(N) \) (BIC)

Choose \( K = \arg\min_k \psi(k) \)
Problems with HMM model

Length of sequence emitted from fixed state is geometrically distributed.

\[ P(j j j j j j j j) = P(\pi_{t+1} = j \mid \pi_t = j)^n \]

For CGH this means,
1) Length of aberrant intervals
2) Separation between two intervals of same copy number
Will be geometrically distributed

CGH Segmentation: Transitions

Let \( l_x \) = length of sequence in state X.

- \( P[l_x = 1] = 1-p \)
- \( P[l_x = 2] = p(1-p) \)
- ...
- \( P[l_x = k] = p^k(1-p) \)
- \( E[l_x] = 1/(1-p) \)
- Geometric distribution, with mean \( 1/(1-p) \)
Biological Interaction Networks

Many types:

- Protein-DNA (regulatory)
- Protein-metabolite (metabolic)
- Protein-protein (signaling)
- RNA-RNA (regulatory)
- Genetic interactions (gene knockouts)
Remaining Lectures

• Biology of cellular interaction networks
• Network Alignment
• Network Motifs
• Network Integration

Metabolic Networks

Nodes = reactants
Edges = reactions labeled by enzyme (protein) that catalyzes reaction
Regulatory Networks

Nodes = genes
Edges = regulatory interaction

Protein-DNA interaction network

A “activates” B
A “represses” C
Signaling Networks

Sources

• http://ai.stanford.edu/~serafim/CS262_2006/ (HMM slides)