Hidden Markov Models

Lecture 6, Thursday April 17, 2003

Decoding

Given \( x = x_1, \ldots, x_n \)

We want to find \( \pi = \pi_1, \ldots, \pi_n \) such that \( P(x, \pi) \) is maximized

\( \pi^* = \arg \max \pi P(x, \pi) \)

We can use dynamic programming!

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

= Probability of most likely sequence of states ending at state \( \pi = k \)

The Viterbi Algorithm

Similar to “aligning” a set of states to a sequence

Time: \( O(K^2N) \)

Space: \( O(KN) \)

Evaluation

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 \( x \)

The Forward Algorithm

We can compute \( f_{ki} \) for all \( k, i \), using dynamic programming!

Initialization:

\( f_{k0} = 1 \)

\( f_{k0} = 0 \), for all \( k > 0 \)

Iteration:

\( f_{ki} = a(k) \sum_{j} f_{kj} a_{ji} \)

Termination:

\( P(x) = \sum_{k} f_{kn} a_{nk} \)

Where, \( a_{nk} \) is the probability that the terminating state is \( k \) (usually \( a_{nk} \))
The Backward Algorithm

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

**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:**
\[
P(x) = \sum_l a_{l0} e_l(x_1) b_l(1)
\]

Posterior Decoding

We can now calculate
\[
P(\pi = k \mid x) = \frac{f_k(i) \ b_k(i)}{P(x)}
\]

Then, we can ask: What is the most likely state at position \( i \) of sequence \( x \):

Define \( \pi^* \) by Posterior Decoding:
\[
\pi^*_i = \arg \max_k P(\pi_i = k \mid x)
\]

Today

- Example: CpG Islands
- Learning

Implementation Techniques

**Viterbi:** Sum-of-logs:
\[
V(l) = \log a_{l0} + \max_k \{ V(l-1) + \log a_{kl} \}
\]

**Forward/Backward:** Scaling by \( c(i) \)

One way to perform scaling:
\[
f_k(l) = c(i) \times \frac{b_k(l)}{\sum_k f_k(l-1) a_{kl}}
\]

Where \( c(i) \) = \( \frac{1}{\sum_k f_k(i)} \)

Details in Rabiner’s Tutorial on HMMs, 1989

Example: CpG Islands

CpG nucleotides in the genome are frequently methylated.

(Write CpG not to confuse with CG base pair)

\[
C \rightarrow \text{methyl-C} \rightarrow T
\]

Methylation often suppressed around genes, promoters
In CpG islands, CG is more frequent. Other pairs (AA, AG, AT...) have different frequencies.

**Question:** Detect CpG islands computationally.

**A model of CpG Islands – (1) Architecture**

A+ C+ G+ T+

A- C- G- T-

**A model of CpG Islands – (2) Transitions**

How do we estimate the parameters of the model?

- **Emission probabilities:** 1/0
  - A: 0.740, C: 0.419, G: 0.580, T: 0.803
  - G: 0.913, A: -0.302, C: 0.812, T: 0.685

1. **Transition probabilities within CpG islands**
   - Established from many known (experimentally verified) CpG islands.
   - Parenthesis - log likelihoods:
     - A: -0.679 + 0.393 + 0.573 - 1.169
     - C: -0.730 + 0.331 + 0.461 - 0.624
     - G: -0.685 + 1.812 + 0.302 - 0.913
     - T: -0.803 + 0.580 + 0.419 - 0.740

2. **Transition probabilities within other regions**
   - Established from many known non-CpG islands.

A better way to see effects of transitions:

\[ L(u, v) = \log \frac{P(uv | +)}{P(uv | -)} \]

Given a region \( x = x_1 \ldots x_N \)

A quick & dirty way to decide whether entire \( x \) is CpG:

\[ \sum_i L(x_i, x_{i+1}) > 0 \]

**Parenthesis – log likelihoods:**

- A: 0.740, C: 0.419, G: 0.580, T: 0.803
- G: 0.913, A: -0.302, C: 0.812, T: 0.685

**A model of CpG Islands – (2) Transitions**

What about transitions between (+) and (-) states?

They affect:

- Avg. length of CpG island
- Avg. separation between two CpG islands

**Length distribution of region X:**

Length distribution of region X:

\[ P(l_i = 1) = 1 - p \]
\[ P(l_i = 2) = p(1 - p) \]
\[ \ldots \]
\[ P(l_i = k) = p(1 - p)^{k-1} \]

\[ E[l_i] = 1/(1 - p) \]

Exponential distribution, with mean 1/(1 - p)

**A problem with this model:** CpG islands don’t have exponential length distribution.

This is a defect of HMMs – compensated with ease of analysis & computation.
Applications of the model

Given a DNA region $x$,

- The Viterbi algorithm predicts locations of CpG islands.

Given a nucleotide $x_i$ (say $x_i = A$)

- The Viterbi parse tells whether $x_i$ is in a CpG island in the most likely general scenario.

- The Forward/Backward algorithms can calculate $P(x_i \text{ is in CpG island}) = P(x_i = A^+ | x)$.

Posterior Decoding can assign locally optimal predictions of CpG islands

$$x_i^* = \text{argmax}_k P(x_i = k | x)$$

What if a new genome comes?

- We just sequenced the porcupine genome.
- We know CpG islands play the same role in this genome.
- However, we have no known CpG islands for porcupines.
- We suspect the frequency and characteristics of CpG islands are quite different in porcupines.
- How do we adjust the parameters in our model?

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:**
     - A genomic region $x = x_1 \ldots x_N$ where we have good (experimental) annotations of all CpG islands.
     - 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:**
     - 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 $P(x | \theta)$.

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:**

$$A_{kl} = \# \text{ times } k \rightarrow l \text{ transition occurs in } x$$

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

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

$$a_{kl} = \frac{A_{kl}}{\Sigma_i A_{ki}}$$

$$\omega_k(b) = \frac{E_k(b)}{\Sigma_c E_k(c)}$$

1. When the right answer is known

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

**Drawback:** Given test data, there may be **overfitting**. $P(x)$ is maximized, but is unreasonable.

**0 probabilities - VERY BAD**

**Example:**

Given 10 casino rolls, we observe

$$x = 2, 1, 3, 1, 2, 3, 6, 2, 3$$


Then:

$$a_{FF} = 1; a_{FL} = 0$$

$$\omega_F(1) = \omega_F(3) = 0.2; \omega_F(2) = 0; \omega_F(4) = 0; \omega_F(5) = \omega_F(6) = 0.1$$
Pseudocounts

Solution for small training sets:

Add pseudocounts

\[ A_{kl} = \text{# times } k \rightarrow l \text{ transition occurs in } \pi \]
\[ E_k(b) = \text{# times state } k \text{ in } \pi \text{ emits } b \text{ in } x \]

\( r_{kl}, r_k(b) \) 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:

\[ r_{F0} = r_{L0} = r_{F1} = r_{L1} = 1; \]
\[ r_{F2} = r_{L2} = \ldots = r_{F6} = 20 \quad \text{strong belief fair is fair} \]
\[ r_{F2} = r_{L2} = \ldots = r_{F6} = 5 \quad \text{wait and see for loaded} \]

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

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:

\[ P(\pi_i = k, \pi_{i+1} = l | x) = \frac{1}{P(x)} \times \frac{P(\pi_i = k, \pi_{i+1} = l, x) = Q/P(x)}{P(x | \theta)} \]

where \( Q = P(x, \pi_i = k, \pi_{i+1} = l) = \sum_{i} P(x, \pi_i = k, \pi_{i+1} = l) \)

So:

\[ P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \frac{\frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x | \theta)}}{P(x | \theta)} \]
### Estimating new parameters

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

$$ A_{kl} = \sum_j \sum_i P(\pi_i = k, \pi_{i+1} = l | x, \theta) = \sum_j \sum_i \frac{P(x | \theta)}{P(x_i | \theta)} $$

Similarly, 

$$ E_k(b) = \sum_j (1/P(x_j)) \sum_i \frac{f_k(i) b_k(i)}{b_k(i)} $$

---

### The Baum-Welch Algorithm

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

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

GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION

Until $P(x | \theta)$ does not change much

---

### Alternative: Viterbi Training

**Initialization:**
- Same

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

Until convergence

**Notes:**
- Convergence is guaranteed - Why?
- Does not maximize $P(x | \theta)$
- In general, worse performance than Baum-Welch
- Convenient - when interested in Viterbi parsing, no need to implement additional procedures (Forward, Backward)!!

---

### Exercise – Submit any time – Groups up to 3

1. Implement a HMM for the dishonest casino (or any other simple process you feel like)
2. Generate training sequences with the model
3. Implement BaumWelch and Viterbi training
4. Show a few sets of initial parameters such that:
   a. BaumWelch and Viterbi differ significantly, and/or
   b. BaumWelch converges to parameters close to the model, and to unreasonable parameters, depending on initial parameters

   - Do not use 0-probability transitions
   - Do not use 0s in the initial parameters
   - Do use pseudocounts in Viterbi

   This exercise will replace the 1-3 lowest problems, depending on thoroughness