Hidden Markov Models (cont’d)

Outline

- Recap HMM
- Computational Issues: Original vs. Log Form
- Matrix Form
- Continuous Values
Hidden Markov Models

The underlying source: genomic entities, dice.

The sequence: Ploy NT, sequence of rolls,

Definition (of HMM)

- Observation space
  - Alphabetic set: \( C = \{c_1, c_2, \ldots, c_K\} \)
  - Euclidean space: \( \mathbb{R}^d \)
- Index set of hidden states \( I = \{1, 2, \ldots, M\} \)
- Transition probabilities between any two states \( p(y'_i = 1 | y'_{i-1} = 1) = a_{i,j} \), or \( p(y'_i = 1 | y'_{i-1} = 1) \sim \text{Multinomial}(a_{1,i}, a_{2,i}, \ldots, a_{M,i}) \forall i \in I \).
- Start probabilities \( p(y'_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M) \)
- Emission probabilities associated with each state \( p(x_i | y'_i = 1) \sim \text{Multinomial}(b_{1,i}, b_{2,i}, \ldots, b_{K,i}) \forall i \in I \).
  - or in general: \( p(x_i | y'_i = 1) \sim f(i | \theta_i) \forall i \in I \).
Three Main Questions on HMMs

1. Evaluation
   GIVEN an HMM $M$, and a sequence $x$,
   FIND $\text{Prob}(x \mid M)$
   ALGO. Forward

2. Decoding
   GIVEN an HMM $M$, and a sequence $x$,
   FIND the sequence $y$ of states that maximizes, e.g., $P(y \mid x, M)$, or the most probable subsequence of states
   ALGO. Viterbi, Forward-backward

3. Learning
   GIVEN an HMM $M$, with unspecified transition/emission probs., and a sequence $x$;
   FIND parameters $\theta = (\pi, a_{ij}, \eta_k)$ that maximize $P(x \mid \theta)$
   ALGO. Baum-Welch (EM)

The HMM Algorithms

Questions:

- **Evaluation**: What is the probability of the observed sequence? **Forward**
- **Decoding**: What is the probability that the state of the 3rd roll is loaded, given the observed sequence? **Forward-Backward**
- **Decoding**: What is the most likely die sequence? **Viterbi**
- **Learning**: Under what parameterization are the observed sequences most probable? **Baum-Welch (EM)**
The Forward Algorithm

- We want to calculate \( P(x) \), the likelihood of \( x \), given the HMM
- Sum over all possible ways of generating \( x \):
  \[
p(x) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_{t-1}} \pi_{y_1} \prod_{r=2}^{t-2} a_{y_{r-1} y_r} \prod_{r=t-1}^{t} p(x_t | y_r)
  \]
- To avoid summing over an exponential number of paths \( y \), define
  \[
  \alpha_t(y_t^k) = \alpha_t^k \overset{\text{def}}{=} P(x_1, \ldots, x_t, y_t^k) = 1 \quad \text{(the forward probability)}
  \]
- The recursion:
  \[
  \alpha_t^k = p(x_t | y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i, k}
  \]
  \[
  P(x) = \sum_k \alpha_t^k
  \]

The Forward Algorithm – derivation

- Compute the forward probability:
  \[
  \alpha_t^k = p(x_1, \ldots, x_{t-1}, x_t, y_t^k = 1)
  \]
  \[
  = \sum_{y_{t-1}} p(x_1, \ldots, x_{t-2}, y_{t-1}) p(y_t^k = 1 | y_{t-1}, x_1, \ldots, x_{t-1}) p(x_t | y_t^k = 1, x_1, \ldots, x_{t-1}, y_{t-1})
  \]
  \[
  = \sum_{y_{t-1}} p(x_1, \ldots, x_{t-1}, y_{t-1}) p(y_t^k = 1 | y_{t-1}) p(x_t | y_t^k = 1)
  \]
  \[
  = p(x_t | y_t^k = 1) \sum_{y_{t-1}} p(x_1, \ldots, x_{t-1}, y_{t-1}) p(y_t^k = 1 | y_{t-1})
  \]
  \[
  = p(x_t | y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i, k}
  \]

Chain rule: \( P(A, B, C) = P(A)P(B | A)P(C | A, B) \)
The Forward Algorithm

- We can compute $\alpha_t^k$ for all $k, t$, using dynamic programming!

**Initialization:**

$\alpha_1^k = P(x_1 | y_1^k = 1) \pi_k$

$\alpha_t^k = P(x_t | y_t^k = 1) P(y_t^k = 1 | y_{t-1}^k = 1) \alpha_{t-1}^k$

**Termination:**

$P(x) = \sum_k \alpha_T^k$

The Backward Algorithm – derivation

- Define the backward probability:

$\beta_t^k = P(x_{t+1}, ..., x_T | y_T^k = 1)$

$= \sum_{y_{t-1}} P(x_{t+1}, ..., x_T, y_{t-1} | y_T^k = 1)$

$= \sum_{y_{t-1}} P(y_{t-1}^k = 1 | y_T^k = 1) P(x_{t+1}, ..., x_T | y_{t-1}^k = 1, y_T^k = 1)$

$= \sum_{y_{t-1}} P(y_{t-1}^k = 1 | y_T^k = 1) P(x_{t+1}, ..., x_T, y_{t-1} | y_T^k = 1)$

$= \sum_{y_{t-1}} a_{y_{t-1}} p(x_{t+1} | y_{t-1}^k = 1) \beta_{t-1}^k$

Chain rule: $P(A, B, C | \alpha) = P(A | \alpha) P(B | A, \alpha) P(C | A, B, \alpha)$
Computational Issues

- Suppose we have a program for the Forward Algorithm
- Calculate the probability of an observed sequence in the Dishonest Casino

"Observed" sequence $X = 111...1$ (10 ones), $P = 6.2447e-9$
$X = 111...1$ (100 ones), $P = 8.2396e-81$
$X = 111...1$ (500 ones), $P = 0!!$

What happened?
Computational Issues

- Why P=0?
  - Underflow

- A simpler example in Matlab
  
  ```matlab
  >> 10^100
  ans = 1.0000e+100
  >> 10^200
  ans = 1.0000e+200
  >> 10^300
  ans = 1.0000e+300
  >> 10^400
  ans = 0
  ```

- Range of Values: 1.7E+/-308 with 15 digits
  - When a number is smaller than 10^-324, it becomes 0 in Matlab.
  - The same to double data type in C

- Solution: save values in logarithm

Arithmetic in Log Form

<table>
<thead>
<tr>
<th>Original Form</th>
<th>Logarithm Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B</td>
<td>a = log A, b = log B</td>
</tr>
<tr>
<td>Multiply: C=A*B</td>
<td>c=a+b</td>
</tr>
<tr>
<td>Add: C=A+B</td>
<td>c=a+log(1+exp(b-a)), if a&gt;b</td>
</tr>
<tr>
<td></td>
<td>c=b+log(1+exp(a-b)), otherwise</td>
</tr>
</tbody>
</table>
Log Form Summary

- **Advantage**
  - Overcome underflow and overflow issues

- **Disadvantage**
  - Less straightforward
  - Longer computation time for Add operation
  - Lose a bit precision
    - Adding one million $10^6$ in log form, we get $1-5.5 \times 10^{-14}$

- **Suggestion:** Use it when necessary

Outline

- Recap HMM
- Computational Issues: Original vs. Log Form
- **Matrix Form**
- Continuous Values
Matrix Form in Forward Algorithm

Recall:
\[
\alpha_t^k = p(x_t | y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}
\]

Let \( \alpha_t^k \) = \( \alpha_t^k \)

\( a(j,k) = a_{j,k} \)

\( b(k, x_t) = p(x_t | y_t^k = 1) \)

for \( t = 2:T \)
for \( k = 1:N \)
    \[
    \text{sum} = 0;
    \text{for } j = 1:N \\
    \text{sum} = \text{sum} + \ldots \\
    \alpha(t-1,j) * a(j,k); \\
    \text{end}
    \]
\[
\alpha(t,k) = b(k, x(t)) \star \text{sum}; \\
\text{end}
\]
end
end

for \( t = 2:T \)
for \( k = 1:N \)
    \[
    \alpha(t,k) = \ldots \\
    b(k, x(t)) \star (\alpha(t-1,:) \star a(:,k)); \\
    \text{end}
    \]
end
end
Matrix Form in Forward Algorithm

\[ \alpha_t^k = p(x_t \mid y_t^k) = 1 \sum_{i} \alpha_{t-1}^i a_{i,k} \]

for t=2:T
  for k=1:N
    alpha(t,k) = …
    b(k,x(t))*(alpha(t-1,:)*a(:,k));
  end
end

for t=2:T
  alpha(t,:) = …
  b(:,x(t))' .* (alpha(t-1,:)*a);
end

Comparison of running time

- version 1 (Scalar operations): 2.6 sec
- version 3 (Matrix operations): 0.3 sec

Problem size
- # of Hidden States = 100
- # of Observed States = 6
- T=100

About 9 times faster!
Outline

- Recap HMM
- Computational Issues: Original vs. Log Form
- Matrix Form
- Continuous Values

Continuous Values in X

- $f(X \mid Y)$
  - No longer a Multinomial Dist.
  - But some continuous dist.
    - Gaussian
    - Mix of two Gaussian

- Replace $P(X \mid Y_k = 1)$ by $f(X \mid Y)$
Continuous Values in Y

- Refer to State Space Models (Kalman Filter algorithm)

Summary

- Logarithm form is helpful when facing underflow/overflow problem
- Matrix operation can greatly speed-up program in Matlab
- Continuous HMMs have similar Forward, Forward-Backward, Viterbi, and Baum-Welch algorithm as Discrete HMM