CS626-460: Speech, NLP and the Web

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Lecture 5,7: HMM and Viterbi
10th and 16th Jan, 2012

(Lecture 6 was on Computational Biomedicine research at Houston University by Prof. Ioannis)
HMM Definition

- Set of states : $S$ where $|S|=N$
- Output Alphabet : $O$ where $|O|=K$
- Transition Probabilities : $A = \{a_{ij}\}$
  - prob. of going from state $S_i$ to state $S_j$
- Emission Probabilities : $B = \{b_{pq}\}$
  - prob. of outputting symbol $O_q$ from state $S_p$
- Initial State Probabilities : $\pi$

$$\lambda = (A, B, \pi)$$
Markov Processes

- Properties
  - Limited Horizon: Given previous \( t \) states, a state \( i \), is independent of preceding \( 0 \) to \( t-k+1 \) states.
    - \( P(X_t=i|X_{t-1}, X_{t-2}, \ldots X_0) = P(X_t=i|X_{t-1}, X_{t-2}, \ldots X_{t-k}) \)
  - Order \( k \) Markov process
  - Time invariance: (shown for \( k=1 \))
    - \( P(X_t=i|X_{t-1}=j) = P(X_1=i|X_0=j) \ldots = P(X_n=i|X_{n-1}=j) \)
Three basic problems (contd.)

- Problem 1: Likelihood of a sequence
  - Forward Procedure
  - Backward Procedure
- Problem 2: Best state sequence
  - Viterbi Algorithm
- Problem 3: Re-estimation
  - Baum-Welch (Forward-Backward Algorithm)
Probabilistic Inference

- O: Observation Sequence
- S: State Sequence

Given O find S* where \( S^* = \arg \max_S p(S \mid O) \) called Probabilistic Inference

- Infer “Hidden” from “Observed”
- How is this inference different from logical inference based on propositional or predicate calculus?
Essentials of Hidden Markov Model

1. Markov + Naive Bayes

2. Uses both transition and observation probability

\[ p(S_k \rightarrow ^{O_k} S_{k+1}) = p(O_k / S_k) p(S_{k+1} / S_k) \]

3. Effectively makes Hidden Markov Model a Finite State Machine (FSM) with probability
Probability of Observation Sequence

\[ p(O) = \sum_s p(O, S) = \sum_s p(S) p(O \mid S) \]

- Without any restriction, 
  - Search space size = \(|S|^{|O|}\)
Continuing with the Urn example

Colored Ball choosing

Urn 1
# of Red = 30
# of Green = 50
# of Blue = 20

Urn 2
# of Red = 10
# of Green = 40
# of Blue = 50

Urn 3
# of Red = 60
# of Green = 10
# of Blue = 30
Example (contd.)

Transition Probability

<table>
<thead>
<tr>
<th></th>
<th>U₁</th>
<th>U₂</th>
<th>U₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>U₂</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>U₃</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Observation/output Probability

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>0.3</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>U₂</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>U₃</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Given:

Observation: RRGGBRGR

What is the corresponding state sequence?
Diagrammatic representation (1/2)
Diagrammatic representation (2/2)
Observations and states

<table>
<thead>
<tr>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
<th>O5</th>
<th>O6</th>
<th>O7</th>
<th>O8</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>B</td>
<td>R</td>
<td>G</td>
<td>R</td>
</tr>
</tbody>
</table>

State: S₁ S₂ S₃ S₄ S₅ S₆ S₇ S₈

Sᵢ = U₁/U₂/U₃; A particular state
S: State sequence
O: Observation sequence
S* = “best” possible state (urn) sequence
Goal: Maximize P(S|O) by choosing “best” S
Goal

- Maximize $P(S|O)$ where $S$ is the State Sequence and $O$ is the Observation Sequence

$$S^* = \arg \max_S (P(S | O))$$
Baye’s Theorem

\[ P(A | B) = P(A).P(B | A) / P(B) \]

- \( P(A) \) - : Prior
- \( P(B | A) \) - : Likelihood

\[ \text{argmax}_S P(S | O) = \text{argmax}_S P(S).P(O | S) \]
State Transitions Probability

\[ P(S) = P(S_{1-8}) \]
\[ P(S) = P(S_i) P(S_2 | S_i) P(S_3 | S_{i-2}) P(S_4 | S_{i-3}) \ldots P(S_8 | S_{i-7}) \]

By Markov Assumption (k=1)

\[ P(S) = P(S_i) P(S_2 | S_i) P(S_3 | S_2) P(S_4 | S_3) \ldots P(S_8 | S_7) \]
Observation Sequence

\[ P(O|S) = P(O_1|S_{1-8}).P(O_2|O_1,S_{1-8}).P(O_3|O_{1-2},S_{1-8})...P(O_8|O_{1-7},S_{1-8}) \]

Assumption that ball drawn depends only on the Urn chosen

\[ P(O|S) = P(O_1|S_1).P(O_2|S_2).P(O_3|S_3)...P(O_8|S_8) \]

\[ P(S|O) = P(S).P(O|S) \]

\[ P(S|O) = P(S_1).P(S_2|S_1).P(S_3|S_2).P(S_4|S_3)...P(S_8|S_7).P(O_1|S_1).P(O_2|S_2).P(O_3|S_3)...P(O_8|S_8) \]
Grouping terms

<table>
<thead>
<tr>
<th>O_0</th>
<th>O_1</th>
<th>O_2</th>
<th>O_3</th>
<th>O_4</th>
<th>O_5</th>
<th>O_6</th>
<th>O_7</th>
<th>O_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\varepsilon)</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>B</td>
<td>R</td>
<td>G</td>
<td>R</td>
</tr>
</tbody>
</table>

State: \(S_0\) \(S_1\) \(S_2\) \(S_3\) \(S_4\) \(S_5\) \(S_6\) \(S_7\) \(S_8\) \(S_9\)

\[
P(S).P(O|S) = [P(O_0|S_0).P(S_1|S_0)].
[P(O_1|S_1).P(S_2|S_1)].
[P(O_2|S_2).P(S_3|S_2)].
[P(O_3|S_3).P(S_4|S_3)].
[P(O_4|S_4).P(S_5|S_4)].
[P(O_5|S_5).P(S_6|S_5)].
[P(O_6|S_6).P(S_7|S_6)].
[P(O_7|S_7).P(S_8|S_7)].
[P(O_8|S_8).P(S_9|S_8)].
\]

We introduce the states \(S_0\) and \(S_9\) as initial and final states respectively.

After \(S_8\) the next state is \(S_9\) with probability 1, i.e., \(P(S_9|S_8)=1\).

\(O_0\) is \(\varepsilon\)-transition
Introducing useful notation

Obs: ε R R G G B R G R
State: S₀ S₁ S₂ S₃ S₄ S₅ S₆ S₇ S₈ S₉

\[ P(O_k | S_k) \cdot P(S_{k+1} | S_k) = P(S_k \xrightarrow{O_k} S_{k+1}) \]
Viterbi Algorithm for the Urn problem (first two symbols)

\[ \begin{align*}
S_0 & \rightarrow 0.5 \\
& \rightarrow 0.3 \\
& \rightarrow 0.2 \\
\end{align*} \]

\[ \begin{align*}
U_1 & \rightarrow 0.03 \\
U_2 & \rightarrow 0.15 \\
U_3 & \rightarrow 0.24 \\
\end{align*} \]

\[ \begin{align*}
& \rightarrow 0.08 \\
& \rightarrow 0.06 \\
& \rightarrow 0.18 \\
& \rightarrow 0.18 \\
\end{align*} \]

\[ \begin{align*}
& \rightarrow 0.015 \\
& \rightarrow 0.04 \\
& \rightarrow 0.075^* \\
& \rightarrow 0.018 \\
& \rightarrow 0.006 \\
& \rightarrow 0.006 \\
& \rightarrow 0.036^* \\
& \rightarrow 0.048^* \\
& \rightarrow 0.036 \\
\end{align*} \]

*: winner sequences
Markov process of order $>1$ (say 2)

<table>
<thead>
<tr>
<th>O_0</th>
<th>O_1</th>
<th>O_2</th>
<th>O_3</th>
<th>O_4</th>
<th>O_5</th>
<th>O_6</th>
<th>O_7</th>
<th>O_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>B</td>
<td>R</td>
<td>G</td>
<td>R</td>
</tr>
</tbody>
</table>

State: S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9

Same theory works

\[
P(S).P(O|S) = P(O_0|S_0).P(S_1|S_0)\cdot [P(O_1|S_1).P(S_2|S_1S_0)]. [P(O_2|S_2).P(S_3|S_2S_1)]. [P(O_3|S_3).P(S_4|S_3S_2)]. [P(O_4|S_4).P(S_5|S_4S_3)]. [P(O_5|S_5).P(S_6|S_5S_4)]. [P(O_6|S_6).P(S_7|S_6S_5)]. [P(O_7|S_7).P(S_8|S_7S_6)]. [P(O_8|S_8).P(S_9|S_8S_7)].
\]

We introduce the states $S_0$ and $S_9$ as initial and final states respectively.

After $S_8$ the next state is $S_9$ with probability 1, i.e., $P(S_9|S_8S_7)=1$

$O_0$ is $\varepsilon$-transition
Adjustments

- Transition probability table will have tuples on rows and states on columns
- Output probability table will remain the same
- In the Viterbi tree, the Markov process will take effect from the 3rd input symbol (εRR)
- There will be 27 leaves, out of which only 9 will remain
- Sequences ending in same tuples will be compared
  - Instead of U1, U2 and U3
  - U1U1, U1U2, U1U3, U2U1, U2U2, U2U3, U3U1, U3U2, U3U3
The question here is:
“what is the most likely state sequence given the output sequence seen”
Developing the tree

Choose the winning sequence per state per iteration
The problem being addressed by this tree is $S^* = \arg \max_{S} P(S \mid a_1 - a_2 - a_1 - a_2, \mu)$

$a_1-a_2-a_1-a_2$ is the output sequence and $\mu$ the model or the machine
Problem statement: Find the best possible sequence

\[ S^* = \arg \max_{S} P(S | O, \mu) \]

where, \( S \rightarrow \) State Seq, \( O \rightarrow \) Output Seq, \( \mu \rightarrow \) Model or Machine

Model or Machine = \{\( S_0, S, A, T \)\}

\( T \) is defined as \( P(S_i \xrightarrow{a_k} S_j) \) \( \forall i, j, k \)
Tabular representation of the tree

<table>
<thead>
<tr>
<th>Latest symbol observed</th>
<th>Ending state</th>
<th>$\epsilon$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S_1$</td>
<td>1.0</td>
<td>(1.0<em>0.1,0.0</em>0.2)</td>
<td>(0.02, 0.09)</td>
<td>(0.009, <strong>0.012</strong>)</td>
<td>(0.0024, <strong>0.0081</strong>)</td>
</tr>
<tr>
<td></td>
<td>$S_2$</td>
<td>0.0</td>
<td>(1.0<em>0.3,0.0</em>0.3)</td>
<td>(0.04,<strong>0.06</strong>)</td>
<td>(<strong>0.027</strong>,0.018)</td>
<td>(0.0048,0.0054)</td>
</tr>
</tbody>
</table>

**Note:** Every cell records the winning probability ending in that state.

The bold faced values in each cell shows the sequence probability ending in that state. Going backward from final winner sequence which ends in state $S_2$ (indicated by the 2nd tuple), we recover the sequence.
Algorithm
(following James Alan, Natural Language Understanding (2nd edition), Benjamin Cummins (pub.), 1995

Given:

1. The HMM, which means:
   a. Start State: $S_1$
   b. Alphabet: $A = \{a_1, a_2, \ldots, a_p\}$
   c. Set of States: $S = \{S_1, S_2, \ldots, S_n\}$
   d. Transition probability $P(S_i \xrightarrow{a_k} S_j) \quad \forall i, j, k$
      which is equal to $P(S_j, a_k \mid S_i)$

2. The output string $a_1a_2\ldots a_T$

To find:

The most likely sequence of states $C_1C_2\ldots C_T$ which produces the given output sequence, i.e., $C_1C_2\ldots C_T = \arg \max_C [P(C \mid a_1, a_2, \ldots, a_T, \mu)]$
Algorithm contd...

Data Structure:
1. A N*T array called SEQSCORE to maintain the winner sequence always (N=#states, T=length of o/p sequence)
2. Another N*T array called BACKPTR to recover the path.

Three distinct steps in the Viterbi implementation
1. Initialization
2. Iteration
3. Sequence Identification
1. Initialization

SEQSCORE(1,1)=1.0
BACKPTR(1,1)=0
For(i=2 to N) do

SEQSCORE(i,1)=0.0

[expressing the fact that first state is $S_1$]

2. Iteration

For(t=2 to T) do

For(i=1 to N) do

SEQSCORE(i,t) = Max$\{j=1,N\}$

\[
[ SEQSCORE( j , ( t - 1 ) ) * P( S_j \xrightarrow{a_k} S_i ) ]
\]

BACKPTR(I,t) = index $j$ that gives the MAX above
3. Seq. Identification

\[ C(T) = i \text{ that maximizes } \text{SEQSCORE}(i,T) \]

For \( i \) from \( (T-1) \) to 1 do

\[ C(i) = \text{BACKPTR}[C(i+1),(i+1)] \]

Optimizations possible:

1. BACKPTR can be \( 1 \times T \)
2. SEQSCORE can be \( T \times 2 \)

Homework: Compare this with A*, Beam Search [Homework]

Reason for this comparison:
Both of them work for finding and recovering sequence