CS626-460: Speech, NLP and the Web

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Lecture 5: HMM; Viterbi
10th Jan, 2012
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

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

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

\[ p(O) = \sum_{s} p(O, S) = \sum_{s} p(S) p(O / S) \]

- 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>$U_1$</th>
<th>$U_2$</th>
<th>$U_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>$U_2$</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$U_3$</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>$U_1$</td>
<td>0.3</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>$U_2$</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>$U_3$</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Given:

Observation: RRGGBRGR

and

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>OBS: R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>B</td>
<td>R</td>
<td>G</td>
<td>R</td>
</tr>
<tr>
<td>State: S₁</td>
<td>S₂</td>
<td>S₃</td>
<td>S₄</td>
<td>S₅</td>
<td>S₆</td>
<td>S₇</td>
<td>S₈</td>
</tr>
</tbody>
</table>

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

\[ P(A \mid B) = \frac{P(A).P(B \mid A)}{P(B)} \]

\begin{align*}
P(A) & \text{ -: Prior} \\
P(B \mid A) & \text{ -: Likelihood}
\end{align*}

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

\[ P(S) = P(S_{1-8}) \]
\[ P(S) = P(S_1) \cdot P(S_2 \mid S_1) \cdot P(S_3 \mid S_{1-2}) \cdot P(S_4 \mid S_{1-3}) \cdots P(S_8 \mid S_{1-7}) \]

By Markov Assumption (k=1)

\[ P(S) = P(S_1) \cdot P(S_2 \mid S_1) \cdot P(S_3 \mid S_2) \cdot P(S_4 \mid S_3) \cdots P(S_8 \mid S_7) \]
Observation Sequence probability

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

Assumption that ball drawn depends only on the Urn chosen

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

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

\[ P(S|O) = P(S_1).P(S_2|S_1).P(S_3|S_2).P(S_4|S_3)...P(S_8|S_7).P(O_1|S_1).P(O_2|S_2).P(O_3|S_3)...P(O_8|S_8) \]
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: $\varepsilon$ R R G G B R G R
State: $S_0$ $S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $S_8$ $S_9$

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

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

Same theory works
\[ P(S).P(O|S) = P(O_0|S_0).P(S_1|S_0). P(O_1|S_1).P(S_2|S_1S_0). P(O_2|S_2).P(S_3|S_2S_1). P(O_3|S_3).P(S_4|S_3S_2). P(O_4|S_4).P(S_5|S_4S_3). \]

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
  - \(U_1U_1, U_1U_2, U_1U_3, U_2U_1, U_2U_2, U_2U_3, U_3U_1, U_3U_2, U_3U_3\)
Probabilistic FSM

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.

0.1 \times 0.1 = 0.01
0.1 \times 0.2 = 0.02
0.1 \times 0.4 = 0.04
0.3 \times 0.3 = 0.09
0.3 \times 0.2 = 0.06
The problem being addressed by this tree is $S^* = \arg\max_{S} P(S | a_1 - a_2 - a_1 - a_2, \mu)$

$a_1-a_2-a_1-a_2$ is the output sequence and $\mu$ the model or the machine
Path found: (working backward)

$S_1 \xrightarrow{a_1} S_2 \xrightarrow{a_2} S_1 \xrightarrow{a_1} S_2 \xrightarrow{a_2} S_1$

Problem statement: Find the best possible sequence

$$S^* = \arg \max_S P(S \mid O, \mu)$$

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

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

Start symbol \quad State collection \quad Alphabet set \quad Transitions

$T$ 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>$\varepsilon$</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 2\textsuperscript{nd} 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 $P(S_i \xrightarrow{a_k} S_j) \quad \forall i, j, k$

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 S₁]

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(Sj → a_k → Si)]
        BACKPTR(I,t) = index j that gives the MAX above
3. Seq. Identification

$C(T) = i$ that maximizes $SEQSCORE(i,T)$

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

$C(i) = BACKPTR[C(i+1),(i+1)]$

Optimizations possible:

1. $BACKPTR$ can be $1*T$
2. $SEQSCORE$ can be $T*2$

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

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