Lecture 6
Hidden Markov Models

CS 6320
Outline

- Markov Chains
- Hidden Markov Model
- Likelihood: Forward Alg.
- Decoding: Viterbi Alg.
Definitions

- A **weighted finite-state automaton** adds probabilities to the arcs
  - The sum of the probabilities leaving any arc must sum to one
- A **Markov chain** is a special case of a WFSA in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can’t represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences
Markov Chain for Weather
Markov Chain for Words
Markov Chain Model

- A set of states
  - $Q = q_1, q_2 \ldots q_N$; the state at time $t$ is $q_t$
- Transition probabilities:
  - A set of probabilities $A = a_{01}a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each $a_{ij}$ represents the probability of transitioning from state $i$ to state $j$
  - The set of these is the transition probability matrix $A$

- Markov Assumption: Current state only depends on previous state

\[ P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1}) \]
Markov Chain Model

\[
\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i
\]

\( \pi = \pi_1, \pi_2, \ldots, \pi_N \) an initial probability distribution over states. \( \pi_i \) is the probability that the Markov chain will start in state \( i \). Some states \( j \) may have \( \pi_j = 0 \), meaning that they cannot be initial states. Also, \( \sum_{i=1}^{n} \pi_i = 1 \)

\( QA = \{ q_x, q_y \ldots \} \) a set \( QA \subset Q \) of legal accepting states
Weather example

Markov chains are useful when we need to compute the probabilities for a sequence of events that are observable.
Markov Chain for Weather

- What is the probability of 4 consecutive warm days?
- Sequence is warm-warm-warm-warm
- I.e., state sequence is 3-3-3-3
- \( P(3,3,3,3) = \pi_3 a_{33} a_{33} a_{33} = 0.2 \times (0.6)^3 = 0.0432 \)

- But what about if states are not observable?
HMM for Ice Cream

- You are a climatologist in the year 2799
- Studying global warming
- You can’t find any records of the weather in Baltimore, MA for summer of 2007
- But you find Jason Eisner’s diary
- Which lists how many ice-creams Jason ate every date that summer
- Our job: figure out how hot it was
Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
  - See *hot* weather: we’re in state *hot*
- But in part-of-speech tagging (and other things)
  - The output symbols are *words*
  - But the hidden states are *part-of-speech tags*
- So we need an extension!
- A **Hidden Markov Model** is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means *we don’t know which state we are in.*
Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_N$;
- Observations $O = o_1, o_2 \ldots o_N$;
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$
    \[
    a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \leq i, j \leq N
    \]
- Observation likelihoods
  - Output probability matrix $B = \{b_i(k)\}$
    \[
    b_i(k) = P(X_t = o_k \mid q_t = i)
    \]
- Special initial probability vector $\pi$
  \[
  \pi_i = P(q_1 = i) \quad 1 \leq i \leq N
  \]
Eisner Task

- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3...
- Produce:
HMM for Ice Cream

- There are two hidden states hot cold
- Observations are the number of ice cream events $O = \{1, 2, 3\}$
Transition Probabilities
Observation Likelihoods
# HMM for Three Basic Problems

| Problem 1 (Likelihood): | Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O|\lambda)$. |
|-------------------------|--------------------------------------------------------------------------------------------------|
| Problem 2 (Decoding):   | Given an observation sequence $O$ and an HMM $\lambda = (A, B)$, discover the best hidden state sequence $Q$. |
| Problem 3 (Learning):   | Given an observation sequence $O$ and the set of states in the HMM, learn the HMM parameters $A$ and $B$. |
Likelihood Computation

Given an HMM $\lambda = (A, B)$ and an observation sequence $O$. Determine the likelihood $P(O | \lambda)$.

Problem 1: Compute the probability of eating 3 1 3 ice creams.

Problem 2: Compute the probability of eating 3 1 3 ice creams when the hidden sequence is hot hot cold.
Likelihood Computation

- For a particular hidden state sequence \( Q \)
- And an observation sequence \( O \)
- The likelihood of the observation sequence is

\[
P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)
\]

\[
P(3\ 1\ 3|\text{hot hot cold}) = P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})
\]
Likelihood Computation

Joint probability of being in a weather state sequence $Q$ and a particular sequence of observations $O$ of ice cream events is:

$$P(O, Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

$$P(3 \ 1 \ 3, \ hot \ hot \ cold) = P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot})$$

$$\times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$
We can compute now the probability of a sequence of observations $O$ using the joint probabilities

$$P(O) = \sum_{Q} P(O, Q) = \sum_{Q} P(O|Q)P(Q)$$

$P(3\ 1\ 3) = P(3\ 1\ 3, \text{cold cold cold}) + P(3\ 1\ 3, \text{cold cold hot}) + \ldots$

$$\ldots + P(3\ 1\ 3, \text{hot hot hot})$$
Forward Algorithm

For $N$ hidden states and a sequence of $T$ observations Forward Algorithm uses $O(N^2T)$ operations instead of $N^T$

$a_t(j)$ is the probability of being in state $j$
after seeing the first $t$ observations

$$a_t(j) = P(o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

$$a_t(j) = \sum_{i=1}^{N} a_{t-1}(i) a_{ij} b_j(o_t)$$
Forward trellis for ice cream example
# Forward Algorithm

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{t-1}(i)$</td>
<td>the previous forward path probability from the previous time step</td>
</tr>
<tr>
<td>$a_{ij}$</td>
<td>the transition probability from previous state $q_i$ to current state $q_j$</td>
</tr>
<tr>
<td>$b_j(o_t)$</td>
<td>the state observation likelihood of the observation symbol $o_t$ given the current state $j$</td>
</tr>
</tbody>
</table>

1. Initialization

$$a_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N$$

2. Recursion

$$a_t(j) = \sum_{i=1}^{N} a_{t-1}(i) a_{ij} b_j(o_t) ; \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination

$$P(O|\lambda) = a_T(q_F) = \sum_{i=1}^{N} a_T(i) a_{iF}$$
Forward Algorithm

\[ \alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} b_j(o_t) \]
**Forward Algorithm**

```plaintext

function FORWARD(observations of len T, state-graph of len N) returns forward-prob

create a probability matrix forward[N+2,T]

for each state s from 1 to N do ; initialization step
    forward[s,1] <- a_{0,s} * b_s(o_1)

for each time step t from 2 to T do ; recursion step
    for each state s from 1 to N do
        forward[s,t] <- \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)

forward[q_F,T] <- \sum_{s=1}^{N} forward[s,T] * a_{s,q_F} ; termination step

return forward[q_F,T]

```

© 2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
Decoding

Decoding: given as input HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, \ldots, o_T$, find the most sequence of states $Q = q_1 q_2 q_3 \ldots q_T$

- POS tagging is such a problem, and so is the weather problem

- Recall that in the case of POS tagging we need to compute

$$\hat{t}_1^n = \arg \max_{t_1^n} P(t_1^n | w_1^n)$$

- We could just enumerate all paths given the input and use the model to assign probabilities to each.
  - Not a good idea.
  - Luckily dynamic programming helps us here
Viterbi Algorithm

Viterbi algorithm computes a trellis using dynamic programming. Observation is processed from left to right filling out a trellis of states \( v_t(j) \) is the probability that HMM is in state \( j \) after seeing the first \( t \) observations

\[
\begin{align*}
\nu_{t-1}(i) & \quad \text{the previous Viterbi path probability from the previous time step} \\
a_{ij} & \quad \text{the transition probability from previous state } q_i \text{ to current state } q_j \\
b_j(o_t) & \quad \text{the state observation likelihood of the observation symbol } o_t \text{ given the current state } j
\end{align*}
\]

\[
\nu_t(j) = \max_{q_0, q_1 \ldots q_{i-1}} P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t q_t = j | \lambda)
\]

\[
\nu_t(j) = \max_{i=1}^{N} \nu_{t-1}(i) a_{ij} b_j(o_t)
\]
Viterbi ttralis for ice cream example
Viterbi Algorithm

1. Initialization

\[ v_1(j) = a_{0j}b_j(o_1) \quad 1 \leq j \leq N \]

\[ bt_1(j) = 0 \]

2. Recursion

\[ v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij}b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T \]

\[ bt_t(j) = \arg\max_{i=1}^{N} v_{t-1}(i) a_{ij}b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T \]

3. Termination

The best score:

\[ P^* = v_t(q_F) = \max_{i=1}^{N} v_T(i) * a_{i,F} \]

The start of backtrace:

\[ q_T^* = bt_T(q_F) = \arg\max_{i=1}^{N} v_T(i) * a_{i,F} \]
Viterbi Traceback
The Viterbi Algorithm

function VITERBI(observations of len T, state-graph of len N) returns best-path

create a path probability matrix viterbi[N+2,T]

for each state s from 1 to N do ; initialization step
    viterbi[s,1] ← a_{0,s} * b_s(o_1)
    backpointer[s,1] ← 0

for each time step t from 2 to T do ; recursion step
    for each state s from 1 to N do
        viterbi[s,t] ← \max_{s' = 1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
        backpointer[s,t] ← \arg\max_{s' = 1}^{N} viterbi[s',t-1] * a_{s',s}

viterbi[q_F,T] ← \max_{s = 1}^{N} viterbi[s,T] * a_{s,q_F} ; termination step

backpointer[q_F,T] ← \arg\max_{s = 1}^{N} viterbi[s,T] * a_{s,q_F} ; termination step

return the backtrace path by following backpointers to states back in time from backpointer[q_F,T]
Viterbi Example

© 2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob path to each cell, (not all paths).
Evaluation

So once you have your POS tagger running how do you evaluate it?

- Overall error rate with respect to a gold-standard test set.
- Error rates on particular tags
- Error rates on particular words
- Tag confusions...
Error Analysis

- Look at a confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>RB</th>
<th>VBD</th>
<th>VBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>—</td>
<td>.2</td>
<td></td>
<td></td>
<td>.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJ</td>
<td>.2</td>
<td>—</td>
<td>3.3</td>
<td>2.1</td>
<td>1.7</td>
<td>.2</td>
<td>2.7</td>
</tr>
<tr>
<td>NN</td>
<td>8.7</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>.2</td>
<td></td>
</tr>
<tr>
<td>NNP</td>
<td>.2</td>
<td>3.3</td>
<td>4.1</td>
<td>—</td>
<td>.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB</td>
<td>2.2</td>
<td>2.0</td>
<td>.5</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBD</td>
<td>.3</td>
<td>.5</td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>4.4</td>
</tr>
<tr>
<td>VBN</td>
<td>2.8</td>
<td></td>
<td></td>
<td></td>
<td>2.6</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)
Evaluation

- The result is compared with a manually coded “Gold Standard”
  - Typically accuracy reaches 96-97%
  - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.