Web Search and Text Mining

Lecture 23: Hidden Markov Models
Markov Models

- Observable states: $1, 2, \ldots, N$ (discrete states)

- Observed sequence: $X_1, X_2, \ldots, X_T$ (discrete time)

- Markov assumption,
  \[ P(X_t = i|X_{t-1} = j, X_{t-2} = k, \ldots) = P(X_t = i|X_{t-1} = j) \]
  dependence on preceding time not any before that.

- Stationarity, probability law is time-invariant (homogeneous).
  \[ P(X_t = i|X_{t-1} = j) = P(X_{t+l} = i|X_{t+l-1} = j) \]
Probability law can be characterized by a state transition matrix, 

$$ A = [a_{i,j}]_{i,j=1}^N $$

with 

$$ a_{i,j} = P(X_t = j | X_{t-1} = i) $$

Constraints on $a_{i,j}$,

$$ a_{i,j} \geq 0 $$

$$ \sum_{j=1}^{N} a_{i,j} = 1 $$
Example

Modeling weather with three states:

1. Rainy (R) 2. Cloudy (C) 3. Sunny (S)

State transition matrix,

$$
\begin{bmatrix}
0.4 & 0.3 & 0.3 \\
0.2 & 0.6 & 0.2 \\
0.1 & 0.1 & 0.8 \\
\end{bmatrix}
$$
Weather predictor example of a Markov model

State 1: rain
State 2: cloud
State 3: sun

1 2
3
0.4 0.6
0.8
0.1 0.3 0.2

State-transition probabilities,
\[ A = \{a_{ij}\} = \begin{bmatrix}
0.4 & 0.3 & 0.3 \\
0.2 & 0.6 & 0.2 \\
0.1 & 0.1 & 0.8 \\
\end{bmatrix} \quad (12) \]

Sequence of observations, \( X_1, X_2, \ldots \)
Compute the probability of observing the sequence, 

$$SSRRSCS$$

given that today is S.

**Product rule**

$$P(AB) = P(A|B)P(A)$$

**The Markov chain rule**

$$P(X_1, X_2, \ldots, X_T) = P(X_T|X_1, X_2, \ldots, X_{T-1})P(X_1, X_2, \ldots, X_{T-1}) =$$

$$P(X_T|X_{T-1})P(X_1, X_2, \ldots, X_{T-1}) =$$

$$P(X_T|X_{T-1})P(X_{T-1}|X_{T-2}) \cdots P(X_2|X_1)P(X_1)$$
The observation sequence


Using the chain rule we have,

\[
P(O|\text{model}) = P(S)P(S|S)^2P(R|S)P(R|R)P(S|R)P(C|S)P(S|C) \\
= \pi_3a_{33}^2a_{31}a_{11}a_{13}a_{32}a_{23} = 1.53 \times 10^{-4}
\]

We used the prior probability (state distribution at time 1)

\[
\pi_i = P(X_1 = i)
\]
A Warm-Up Example

Suppose we want to compute $P(X_T = i)$, (brute-force)

$$P(X_T = i) = \sum_{O \text{ ends with } i} P(O)$$

sum over all sequences ending with state $i$

Complexity, $O(TN^T)$. But notice

$$P(X_{t+1} = i) = \sum_{i=j}^N P(X_{t+1} = i, X_t = j)$$

$$= \sum_{i=j}^N P(X_t = j)P(X_{t+1} = i|X_t = j) = \sum_{i=j}^N P(X_t = j)a_{ji}$$

Complexity, $O(TN^2)$. (same idea applied to many other calculations)
What is $P(q_t = s)$?

- For each state $s_i$, define $p_t(i) = \text{Prob. state is } s_i$ at time $t = P(q_t = s_i)$

- Computation is simple.
- Just fill in this table in this order:

<table>
<thead>
<tr>
<th>$t$</th>
<th>$p_t(1)$</th>
<th>$p_t(2)$</th>
<th>...</th>
<th>$p_t(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{\text{final}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$p_t(i) = P(X_t = i)$ (Andrew Moore)
Hidden Markov Models

- States are not observable
- Observations are probabilistic functions of state
- State transitions are still probabilistic
Occasionally Dishonest Casinos

Use a fair die most of the time but occasionally switch to a loaded die. Loaded die: 6/.5, 1-5/.1.

Transition matrix (between fair die.loaded die)

\[
\begin{bmatrix}
0.95 & 0.05 \\
0.1 & 0.9
\end{bmatrix}
\]

The observations is a sequence of rolls (1-6), and which die is used is hidden.
Hidden Markov Models

• Hidden states: 1, 2, ..., N (discrete states)

• Observable symbols: 1, 2, ..., M (discrete observations)

• Hidden state sequence: $X_1, X_2, \ldots, X_T$

• Observed sequence: $Y_1, Y_2, \ldots, Y_T$

• State transition matrix $A = [a_{ij}]_{i,j=1}^N$

  \[
  a_{ij} = P(X_t = j|X_{t-1} = i)
  \]
• Observation probability distribution (time-invariant)

\[ B_j(k) = P(Y_t = k | X_t = j), \quad k = 1, \ldots, M \]

• Initial state distribution

\[ \pi_i = P(X_1 = i), \quad i = 1, \ldots, N \]
Joint Probability

\[ P(X, Y) = P(X_1, \ldots, X_T, Y_1, \ldots, Y_T) = P(X)P(Y|X) = \]
\[ = P(X_T|X_{T-1}) \cdots P(X_2|X_1)P(X_1)P(Y_1|X_1) \cdots P(Y_T|X_T) \]
Three Fundamental Problems in HMMs

1. Evaluate $P(O)$ for a given observation sequence $O$

2. Find the most probable state sequence for a given observation sequence $O$

3. Estimate the HMM model parameters given observation sequence(s).
Problem I

Let the observation be \( O = y_1, y_2, \ldots, y_T \). Again using brute force,

\[
P(O) = \sum_{x_1, \ldots, x_T} P(x_1, \ldots, x_T, y_1, y_2, \ldots, y_T)
\]

Summing over \( N^T \) terms.

Forward-Backward Procedure

Consider forward variable,

\[
\alpha_t(i) = P(y_1, y_2, \ldots, y_t, X_t = i), \quad t = 1, \ldots, T, i = 1, \ldots, N
\]
1. Initialization

\[ \alpha_1(i) = \pi_i B_i(y_1), \quad i = 1, \ldots, N \]

2. Induction

\[ \alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] B_j(y_{t+1}) \]

3. Termination,

\[ P(O) = \sum_{i=1}^{N} \alpha_T(i) \]
Similarly, define the backward variables,

$$
\beta_t(i) = P(y_{t+1}, \ldots, y_T | X_t = i)
$$

1. Initialization $\beta_T(i) = 1$

2. Induction

$$
\beta_t(i) = \sum_{j=1}^{N} a_{ij} B_j(y_{t+1}) \beta_{t+1}(j)
$$
**Most Probable State Path: Viterbi Algorithm**

Given an observation sequence, choose the most likely state sequence.

\[
\{x_1, \ldots, x_T\} = \arg\max _{\{x_1, \ldots, x_T\}} P(\{x_1, \ldots, x_T\}, \{y_1, \ldots, y_T\})
\]

**One solution**, optimal individual state,

\[
\gamma_t(i) = P(X_t = i|O) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}
\]

Then choose

\[
x_t^* = \arg\max_{1 \leq i \leq N} \gamma_t(i)
\]

Why not a good idea?
Solution by DP. Define

\[ v_t(i) = \max_{x_1, \ldots, x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t = i, y_1, \ldots, y_t) \]

Highest probability path ending at state \( i \) at time \( t \).

Notice that

\[ P(x_1, \ldots, x_{t+1}, y_1, \ldots, y_{t+1}) = P(x_1, \ldots, x_t, y_1, \ldots, y_t)P(x_{t+1}|x_t)P(y_{t+1}|x_{t+1}) \]

Then

\[ v_{t+1}(j) = \max_{x_1, \ldots, x_t} P(x_1, \ldots, x_t, x_{t+1} = j, y_1, \ldots, y_{t+1}) \]

\[ = \max_i (v_t(i)a_{ij})B_j(y_{t+1}) \]
Viterbi Algorithm

Initialization

\[ v_1(i) = \pi_i B_i(y_1), \quad i = 1, \ldots, N \]
\[ \psi_1(i) = 0, \quad i = 1, \ldots, N \]

Recursion

\[ v_{t+1}(j) = \max_j (v_t(i)a_{ij}) B_j(y_{t+1}) \]
\[ \psi_t(j) = \arg\max_i (v_t(i)a_{ij}) \]

Termination

\[ P^* = \max_i v_T(i), \]
\[ x_T^* = \arg\max_i v_T(i) \quad x_t^* = \psi_{t+1}(x_{t+1}^*) \]
Word Segmentation

I can read words without spaces

For position $i$ in the string,

1) $best[i]$ the probability of the most probable segmentation from start to $i$.

2) $words[i]$, the word ending at $i$ with the best probability
Viterbi Word Segmentation

For a string \( \text{text} \) with length \( n + 1 \)

\[
\text{for } i = 0 \text{ to } n \text{ do } \\
\quad \text{for } j = 0 \text{ to } i-1 \text{ do } \\
\qquad \text{word} = \text{text}[j:i] \\
\qquad w = \text{length(word)} \\
\qquad \text{if } \text{P(word)} \cdot \text{best}[i-w] > \text{best}[i] \text{ then } \\
\qquad \quad \text{best}[i] = \text{P(word)} \cdot \text{best}[i-w] \\
\qquad \quad \text{words}[i] = \text{word}
\]

Last word is \( \text{words}[n] \), and let \( k = \text{length(\text{words}[n])} \), and next to the last is \( \text{words}[n-k] \), etc.
Bigram PoS tagger

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

\[
\sim \prod_{i=1}^{N} P(w_i|t_i)P(t_i|t_{i-1})
\]

(S. Renals)
### Transition and observation probabilities

#### Transition probabilities: $P(t_i|t_{i-1})$

<table>
<thead>
<tr>
<th></th>
<th>VB</th>
<th>TO</th>
<th>NN</th>
<th>PPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>0.019</td>
<td>0.0043</td>
<td>0.041</td>
<td>0.067</td>
</tr>
<tr>
<td>VB</td>
<td>0.0038</td>
<td>0.0345</td>
<td>0.047</td>
<td>0.070</td>
</tr>
<tr>
<td>TO</td>
<td>0.83</td>
<td>0</td>
<td>0.00047</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>0.0040</td>
<td>0.016</td>
<td>0.087</td>
<td>0.0045</td>
</tr>
<tr>
<td>PPSS</td>
<td>0.23</td>
<td>0.00079</td>
<td>0.0012</td>
<td>0.00014</td>
</tr>
</tbody>
</table>

#### Observation likelihoods: $P(w_i|t_i)$

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>race</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>0</td>
<td>0.0093</td>
<td>0</td>
<td>0.00012</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>0</td>
<td>0.99</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>0</td>
<td>0.000054</td>
<td>0</td>
<td>0.00057</td>
</tr>
<tr>
<td>PPSS</td>
<td>0.37</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Named Entity Extraction

Mr. Jones eats.

Mr. <ENAMEX TYPE=PERSON>Jones</ENAMEX> eats.

Probability computation,

\[
\Pr(\text{NOT-A-NAME} \mid \text{START-OF-SENTENCE}, +\text{end}+) * \\
\quad \Pr(\text{Mr.} \mid \text{NOT-A-NAME}, \text{START-OF-SENTENCE}) * \\
\quad \Pr(+\text{end}+ \mid \text{Mr.}, \text{NOT-A-NAME}) * \\
\quad \Pr(\text{PERSON} \mid \text{NOT-A-NAME}, \text{Mr.}) * \\
\quad \Pr(\text{Jones} \mid \text{PERSON}, \text{NOT-A-NAME}) * \\
\quad \Pr(+\text{end}+ \mid \text{Jones}, \text{PERSON}) * \\
\quad \Pr(\text{NOT-A-NAME} \mid \text{PERSON}, \text{Jones}) * \\
\quad \Pr(\text{eats} \mid \text{NOT-A-NAME}, \text{PERSON}) * \\
\quad \Pr(\text{.} \mid \text{eats}, \text{NOT-A-NAME}) * \\
\quad \Pr(+\text{end}+ \mid ., \text{NOT-A-NAME}) * \\
\quad \Pr(\text{END-OF-SENTENCE} \mid \text{NOT-A-NAME}, \text{.})
\]
Algorithms for NER

Decision trees

Hidden Markov models

Maximum entropy models

SVM

Boosting

Conditional random fields
**Generative Model**

Generation of words and name-classes:

1. Select a name-class NC, conditioning on the previous name-class and the previous word.

2. Generate the first word inside that name-class, conditioning on the current and previous name-classes.

3. Generate all subsequent words inside the current name-class, where each subsequent word is conditioned on its immediate predecessor.
Top-level Model

1. The probability for generating the first word of a name-class,
   \[ P(NC|NC_{-1}, w_{-1})P([w, f]_{first}|NC, NC_{-1}) \]

2. Generating all but the first word in a name-class,
   \[ P([w, f]|[w, f]_{-1}, NC) \]

3. Distinguished +end+ word, the final word of its name-class,
   \[ P([ + end+, other]|[w, f]_{fina}, NC) \]
Dealing with low-frequency words

- Split words into two sets: 1) frequent words and rare words;

- Map rare words to a small, finite set
While the number of word-states within each name-class is equal to $V$, this \textit{"interior"} bigram language model is ergodic, \textit{\ i.e.\ }, there is a probability associated with every one of the $V^2$ transitions. As a parameterized, trained model, if such a transition were never observed, the model "backs off" to a less powerful model, as described below, in §3.2.3.

\subsection{Words and Word-Features}

The word feature is the one part of this model that is language-dependent. Fortunately, the word feature computation is an extremely small part of the implementation, at roughly twenty lines of code. The rationale for having such features is clear:

\begin{itemize}
  \item \textit{\textcopyright} in Roman languages, capitalization gives good evidence of names.
  \item Numeric symbols can automatically be grouped into categories, as in the initial features in Table 3.1.
  \item Semantic classes can be defined by lists of words having a semantic feature.
  \item Special character sets such as the ones used for transliterating names in Chinese or in Japanese can be identified.
\end{itemize}

Throughout most of the model, we consider words to be ordered pairs (or two-element vectors), composed of word and word-feature, denoted $w_f$. The word feature is a simple, deterministic computation performed on each word as it is added to or looked up in the vocabulary. It produces one of the fourteen values in Table 3.1.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
Word Feature & Example Text & Intuition \\
\hline
twoDigitNum & 90 & Two-digit year \\
fourDigitNum & 1990 & Four digit year \\
containsDigitAndAlpha & A8956-67 & Product code \\
containsDigitAndDash & 09-96 & Date \\
containsDigitAndSlash & 11/9/89 & Date \\
containsDigitAndComma & 23,000.00 & Monetary amount \\
containsDigitAndPeriod & 1.00 & Monetary amount, percentage \\
otherNum & 456789 & Other number \\
allCaps & BBN & Organization \\
capPeriod & M. & Person name initial \\
firstWord & \textit{first word of sentence} & No useful capitalization information \\
initCap & Sally & Capitalized word \\
lowerCase & can & Uncapitalized word \\
other & \text{,} & Punctuation marks, all other words \\
\hline
\end{tabular}
\caption{Word features, examples and intuition behind them.}
\end{table}
Dealing with Low-Frequency Words: An Example

Profits soared at Boeing Co. easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

⇓

firstword soared at initCap Co., easily lowercase forecasts on initCap Street, as their CEO Alan initCap announced first quarter results.

| NA   | = No entity |
| SC   | = Start Company |
| CC   | = Continue Company |
| SL   | = Start Location |
| CL   | = Continue Location |

(Regina Barzilay)