Hidden Markov Models II

Matthew Stone
CS 520, Spring 2000
Lecture 9

HMM – Recap

- Models based on key independence assumptions for time-series data

Events:
\[ \delta_i^{(t)} \quad v_k^{(t)} \]

Arcs determine matrix A
\[ A_{ij} = P(\delta_j^{(t)} \mid \delta_i^{(t-1)}) \]

Obs governed by matrix B
\[ B_{jk} = P(v_k^{(t)} \mid \delta_j^{(t)}) \]
HMM – Recap

- Basic event: seeing (given) observations when system follows (hypothesized) path
  \[ P(v, s) = L^{[\delta]} \prod_{u=2}^{m} A^{[\delta,u]} \prod_{u=1}^{m} B^{[v,\delta,u]} \]

- We started with the evaluation problem
  Compute \( P(v | \omega_i, \text{len} = m) \)

Example

- Track robot motion in a 3x3 grid of rooms:
  - Robot moves randomly to adjacent rooms
  - Rooms have either red, green or blue walls
    - color is observed
    - start at ♠
Example (CONTINUED)

- **Observe sequence:**
  - blue (b), green (g), red (r)
- **Question: are you in this environment** \((\omega_0)\)
  - (or some other?)
- **Answer using evaluation:**
  \[
P(b^{(1)}g^{(2)}r^{(3)} | \omega_0)
  \]

Step Through Evaluation

- **Build a table**
  - of the likelihood of being in room \(r\) at time \(t\)
given the observations so far
Step Through Evaluation 1

• Start with the first step

\[ P(r_i^{(1)},b^{(1)}) = L_iB_b \]

\[ = \begin{cases} 0.9 & \text{for start state} \\ 0 & \text{otherwise} \end{cases} \]

Assumes color confusion:

\[
\begin{array}{ccc}
    b & g & r \\
0.87 & 0.1 & 0.03 \\
0.1 & 0.87 & 0.03 \\
0.03 & 0.03 & 0.94 \\
\end{array}
\]

Step Through Evaluation 2

• Sum up transitions to nearby states

\[ P(r_j^{(2)},b^{(1)}) = \sum_i A_{ij}P(r_i^{(1)},b^{(1)}) \]

Assumes transition matrix:

\[
\begin{array}{cccc}
    \text{move} & 00 & 01 & 10 & 11 \\
0 & 0.05 & 0.3 & 0.5 & 0.15 \\
\end{array}
\]
**Step Through Evaluation**

**3**

- **Factor in second observation**

\[
P(r_j^{(2)}, b^{(1)}, g^{(2)}) = B_{jg} P(r_j^{(2)}, b^{(1)})
\]

Apply confusion matrix:

\[
\begin{array}{ccc}
  b & g & r \\
  b & 0.87 & 0.1 & 0.03 \\
  g & 0.1 & 0.87 & 0.03 \\
  r & 0.03 & 0.03 & 0.94 \\
\end{array}
\]

**Step Through Evaluation**

**4**

- **Sum up transitions to nearby states**

\[
P(r_k^{(3)}, b^{(1)}, g^{(2)}) = \sum_j A_{jk} P(r_j^{(2)}, b^{(1)}, g^{(2)})
\]

Assumes transition matrix:

\[
\begin{array}{cccc}
  move & 00 & 01 & 10 & 11 \\
  p & 0.05 & 0.3 & 0.5 & 0.15 \\
\end{array}
\]
Step Through Evaluation

5

- Factor in third observation

\[ P(r_k^{(3)}, b^{(1)}, g^{(2)}, r^{(3)}) = B_k P(r_k^{(3)}, b^{(1)}, g^{(2)}) \]

Apply confusion matrix:

\[
\begin{pmatrix}
  b & g & r \\
  b & 0.87 & 0.1 & 0.03 \\
  g & 0.1 & 0.87 & 0.03 \\
  r & 0.03 & 0.03 & 0.94 \\
\end{pmatrix}
\]

Step Through Evaluation

6

- Sum up to account for observations

\[ P(b^{(1)}, g^{(2)}, r^{(3)}) = \sum_k P(r_k^{(3)}, b^{(1)}, g^{(2)}, r^{(3)}) \]

= \[\bullet\]
Forward Algorithm

• Key points:
  – For each new step, only the state at the last step (and the probability we ended there) is needed
  – Sum over all state sequences using dynamic programming
  – Finish by summing out over possible final states

HMM – Recap

• We have seen the evaluation problem
  Compute $P(v \mid \omega_i, \text{len} = m)$

• Now we turn to the decoding problem
  Find $\arg\max_{\delta} P(\delta \mid v, \omega_i)$
Decoding Example

For our robot from before –

\[
\begin{align*}
\arg\max_{\delta} P(\delta | b^{(1)}, g^{(2)}, r^{(3)}) &= \arg\max_{\delta} \frac{P(\delta, b^{(1)}, g^{(2)}, r^{(3)})}{P(b^{(1)}, g^{(2)}, r^{(3)})} \\
&= \arg\max_{\delta} P(\delta, b^{(1)}, g^{(2)}, r^{(3)}) \\
&= \bigcirc
\end{align*}
\]

Decoding vs. Evaluation

The sequence of most likely states from evaluation is different:

Most likely state at time 2

Impossible transition, so “very unlikely” sequence
Decoding vs. Evaluation

- Part of the difference is that the forward algorithm only uses past observations
  - Most likely state at time 2, given 2 observations
  - Not likely state for time 2, given 3 observations

Decoding vs. Evaluation

- Part of the difference is that the forward algorithm only uses past observations
- We could get around this
  - by assuming system is in state $i$ at time $t$, and reapplying forward algorithm onward
  - by computing full distribution on states at time $t$ given future info, via the backward algorithm (as we’ll see Wednesday)
Decoding vs. Evaluation

• Part of the difference is that the forward algorithm only uses past observations
• We could get around this

• But decoding is different from evaluation in another way…

Decoding vs. Evaluation

Let $\hat{\delta} = \operatorname{argmax}_\delta P(\delta \mid \mathbf{v})$

• **Overall** likelihood of being in state given observations:

$$P(\hat{\delta}_j^{(t)}, \mathbf{v})$$

• **Do not** have:

$$\hat{\delta}^{(t)} = \operatorname{argmax}_{\delta_j} P(\delta_j^{(t)}, \mathbf{v})$$
**Decoding vs. Evaluation**

**VISUALIZATION**

- **In words:**
  - State $t$ of the highest probability sequence need not have highest probability at $t$

  ![Diagram showing probabilities](image)

  - Many low probability paths can give high $P(\delta_1^{(t)}, v)$ and outweigh single high probability path at $\delta_3$

**Viterbi Decoding**

- **Tabling also works for decoding**
- **As with evaluation**
  - Unfold the HMM through time
  - Assign a value to each state at each step
- **For decoding, value is**
  - Most likely state sequence up to there
  - Probability of that sequence and past observations
Formal Justification

• Key probabilities to maximize:
  \[ P(\delta^{(\leq t)}, v^{(\leq t)}) \text{ subject to } \delta^{(t)} = \delta_j \]
  \[ = A^{[\delta,t]} B^{[v,\delta,t]} P(\delta^{(\leq t-1)}, v^{(\leq t-1)}) \]
  maximized at \( M_{j,t} \)

• Any pair \( \delta^{(t-1)}, \delta^{(t)} \) determines \( A^{[\delta,t]} B^{[v,\delta,t]} \)

Formal Justification (CONTINUED)

• So we conclude
  \( \hat{\delta} \) maximizes \( P(\delta^{(\leq t)}, v^{(\leq t)}) \) subject to \( \hat{\delta}^{(t)} = \delta_j \)

• Exactly when
  \( \delta \) maximizes \( P(\delta^{(\leq t-1)}, v^{(\leq t-1)}) \) subject to \( \delta^{(t-1)} = \delta_i \)
  \( \left( \text{at } P(\delta^{(\leq t-1)}, v^{(\leq t-1)}) = M_{i,t-1} \right) \)

• And
  \( A_j B_j^{[v,t]} M_{i,t-1} \) exceeds other \( A_j B_j^{[v,t]} M_{i,t-1} \)
Viterbi Algorithm

• Initialize
  – Set $M_{i,1} = L_i B_i^{[v,1]}$
  – Set best seq($i, 1$) = $\delta_i^{(1)}$

• Step
  – Set $h = \arg\max_i A_{ij} B_j^{[v,t]} M_{i,t-1}$
  – Set $M_{j,t} = A_{ij} B_j^{[v,t]} M_{h,t-1}$
  – Set best seq($j, t$) = best seq($h, t-1$), $\delta_j^{(t)}$

Viterbi Algorithm
(CONTINUED)

• Finish
  – After step $m$,
  – Set $h = \arg\max_i M_{i,m}$
  – Return best seq($h, m$)
Classic Illustration of Viterbi

- **Part-of-speech tagging**
  - Preprocessing step in natural language processing
- **Words are ambiguous**
  - They may fulfill different roles in a sentence
  - Each role may be used with different senses of the word

Lexical Ambiguity

- **Here’s an example of the contrast:**
  - Same word can provide object or action
    - The *plants* grow in Sandy’s yard.
    - Sandy *plants* tomatoes in the yard.
  - But may describe different objects too
    - The *plants* bear fruit in August.
    - The *plants* employ union workers.
Part-of-speech Tagging

- Research shows
  - you can identify the right word sense well – if you know the role it plays in sentence – its part of speech
  - you can do a good job predicting parts of speech using local (Markov) models of word sequences

POS Tagging

- Simplest case: bigram tagging
  - (hidden) states are just parts of speech
  - observations are words
  - HMM parameter A gives probabilities of seeing parts of speech in succession
  - HMM parameter B gives probabilities of seeing words with different parts of speech

- Trigram tagging typically used in practice
POS Tagging

- **Use decoding**
  - Given a string of words \( v \)
  - Find sequence of parts of speech to maximize \( P(v, \delta) \)

Illustration

- **Process the example:**
  - Q: Why is this ski run difficult?
  - A: The slopes fall fast.
Processing Sketch

• Initialize

Many initial states are possible
But *the* can only be used as a determiner
So only one nonzero entry for $M_{i,1}$

Processing Sketch (CONTINUED)

• Next Step

Determiners may be followed by a range of parts of speech
This translates to a number of nonzero $A$ entries
Processing Sketch (CONTINUED)

- **Step 2, continued**

  And there are a range of ways to use *slopes*, which translate to nonzero *B* entries

  ![Diagram](image1)

  ![Diagram](image2)

- **Step 2, continued**

  When we multiply *A* and *B* entries, the choice is clear

  We save sequence *DET, NPL* and its probability

  ![Diagram](image3)
Next Step

Processing Sketch (CONTINUED)

We consider what follows plural nouns...

Step 3, continued

...and the range of ways to use fall...
• Step 3, continued

...to understand the next word.

We save sequence DET, NPL, VPL and its probability