Lecture 16: Hidden Markov Models

Sanjeev Arora          Elad Hazan
Course progress

• Learning from examples
  • Definition + fundamental theorem of statistical learning, motivated efficient algorithms/optimization
  • Convexity, greedy optimization – gradient descent
  • Neural networks

• Knowledge Representation
  • NLP
  • Logic
  • Bayes nets
  • Optimization: MCMC
  • HMM (today) (a special case of Bayes nets)

• Next: reinforcement learning
Admin

• (written) ex4 – announced today
• Due after Thanksgiving (Thu)
Markov Chain

Markov chain with three states ($s = 3$)

Directed graph, and a transition matrix giving, for each $i, j$ the probability of stepping to $j$ when at $i$.

Transition matrix

\[
T = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0.1 & 0.9 \\
0.6 & 0.4 & 0
\end{bmatrix}
\]
Ergodic theorem

Every irreducible and a-periodic Markov chain has a unique stationary distribution, and every random walk starting from any node converges to it!
Non-stationary Markov chains

Notice: self-loop $\rightarrow$ not periodic anymore
Non-stationary Markov chains

Irreducible → for any pair of vertices, $T_{ij} > 0$ after finitely many iterations.

“reducible”
This lecture: temporal models
Hidden Markov Models

\[ X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_3 \rightarrow X_5 \]

Hidden variables

Evidence variables (observed)
Applications

• Time-dependent variables / problems (e.g. treating patients with changing biometrics over time)

• Natural sequential data (speech, text, etc.).

• Example - text tagging:

  the dog saw a cat

  D   N   V   D   N
Hidden Markov Models: definitions

• $X_t$ = state at time $t$
• $E_t$ = evidence at time $t$
• $P(X_0)$ = initial state
• $P(X_t|X_{t-1})$ – transition model = Markov chain
• $P(E_t|X_t)$ – sensor/observation model, random
• Assumptions:
  • Future is independent of past given present (1st order)
    $P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})$
  • Current evidence only depends on current state
    $P(E_t|X_{0:t}, E_{1:t-1}) = P(E_t|X_t)$
Hidden Markov Models – 2\textsuperscript{nd} order dependencies

natural extension

\[ P(X_t | X_{0:t-1}) = P(X_t | X_{t-1}, X_{t-2}) \]
Hidden Markov Models – translation

English

Hebrew
HMMs – questions we want to solve

1. **Filtering**: what’s the current state?
   \[ P(X_t|E_t) = ? \]

2. **Prediction**: where will I be in k steps?
   \[ P(X_{t+k}|E_{1:t}) = ? \]

3. **Smoothing**: where was I in the past?
   \[ P(X_k|E_{1:t}) = ? \]

4. **Most likely sequence to the data**
   \[ \arg \max_{X_{0:t}} P(X_{0:t}|E_{1:t}) = ? \]
Example – word tagging by trigram HMM

*the dog saw a cat*

D  N  V  D  N

- Let \( K = \{V,N,D,Adv,\ldots,*\text{-STOP}\} \) be a set of labels. These are going to be our states.
- \( V = \) dictionary words, these are the observations.
- Model = HMM with 3 arcs back. Trigram assumption:

\[
P(X_t | X_{0:t-1}) = P(X_t | X_{t-1}, X_{t-2}), \quad P(E_t | X_{0:t}, E_{1:t-1}) = P(E_t | X_t)
\]
Example – word tagging by trigram HMM

*the dog saw a cat*

D N V D N

• We will see,

\[
P(\text{the dog laughs}, D N V \text{STOP}) =
\]

\[
P(D|*,*) \times P(N|*,D) \times P(V|D N) \times P(\text{STOP}|N,V) \times
\]

\[
\times P(\text{the }|D) \times P(\text{dog }|N) \times P(\text{laughs }|V)
\]
Example – word tagging by trigram HMM

• Assume we know transition probabilities $P(X_t | X_{t-1}, X_{t-2})$
• Assume we know observation frequencies $P(E_t | X_t)$
• (how can we estimate these from labelled data?)
Decoding HMMs

Input: sentence $E_{1, E_2, ..., E_t}$
Output: tagging according to labels in $K$ (N,V,...), i.e. the states $X_{1, ..., X_t}$

i.e. $\arg\max_{X_{0:t}} P(X_{0:t} = x_{0:t} | E_{1:t} = e_{1:t})$

By the trigram Markov assumption, we have:

$$P(x_{0:t}, e_{1:t}) = \prod_{i=1 \text{ to } t} P(x_i | x_{i-1}, x_{i-2}) \prod_{i=1 \text{ to } t} P(e_i | x_i)$$

Why?
Decoding HMMs

\[ P(x_{0:t}, e_{1:t}) = \]

\[ = P(x_{1:t}) \times P(e_{0:t} | x_{1:t}) \quad \text{(complete probability)} \]

\[ = \prod_{i=1}^{t} P(x_i | x_{1:i-1}) \times \prod_{i=1}^{t} P(e_i | x_{1:i-1}, e_{1:t}) \quad \text{(chain rule)} \]

\[ = \prod_{i=1}^{t} P(x_i | x_{i-1}, x_{i-2}) \times \prod_{i=1}^{t} P(e_i | x_{1:i-1}, e_{1:t}) \quad \text{(2nd order MC)} \]

\[ = \prod_{i=1}^{t} P(x_i | x_{i-1}, x_{i-2}) \times \prod_{i=1}^{t} P(e_i | x_i) \quad \text{(cond. independence)} \]
Decoding HMMs – Viterbi algorithm

Let

\[ f(X_{0:k}) = \prod_{i=1}^{k} P(X_i|X_{i-1},X_{i-2}) \prod_{i=1}^{k} P(e_i|X_i) \]

And define

\[ \pi_k(u,v) = \max_{X_{0:k-2}} f(X_{0:k-2},u,v) \]

Recall: we want to compute:

\[ \arg \max_{x_{0:t}} P(x_{0:t},e_{1:t}) = \arg \max_{x_{0:t}} f(x_{0:t}) \]
Decoding HMMs – Viterbi algorithm

Let

\[ f(X_{0:k}) = \prod_{i=1}^{k} P(X_i|X_{i-1},X_{i-2}) \prod_{i=1}^{k} P(e_i|X_i) \]

And define

\[ \pi_k(u,v) = \max_{X_{0:k-2}} f(X_{0:k-2},u,v) \]

Main lemma:

\[ \pi_k(u,v) = \max_w \{ \pi_{k-1}(w,u) \times P(v|w,u) \times P(e_k|v) \} \]

Now the algorithm is straightforward: compute this recursively! (a.k.a. dynamic programming)
Viterbi: explicit pseudo code

Input: observations $e_1,...,e_t$

Output: most likely variable assignments $x_0,...,x_t$

Initialize: set $x_0,x_1$ to be “*”

For $k=1,2,...,t$ do:

• For $u,v$ in $K$ do:
  1. $\pi_k(u,v) = \max_w \{ \pi_{k-1}(w,u) \times P(v|w,u) \times P(e_k|v) \}$
  2. Save the $\pi_k(u,v)$ value and the assignments which meets it

• end

Return $\max_{u,v} \{ \pi_t(u,v) \times P(STOP|u,v) \}$ and assignments which meets it

Computational complexity?
Hidden Markov Models – another view

Hidden variables: 2 states

Evidence variables (observed, 2 options)
Hidden Markov Models – another view

Markov chain with:
1. Transition probabilities that govern state change
2. Distribution over signals/observations from each state

Transition matrix:

<table>
<thead>
<tr>
<th></th>
<th>E = “a”</th>
<th>E = “b”</th>
<th>E = “c”</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(“a”</td>
<td>x_t</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>P(“b”</td>
<td>x_t</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Observation matrices:
“forward algorithm”

To compute $P(X_{t+1}|e_{1:t+1})$, recursive formula (similar to what we did)

$$P(X_{t+1}|e_{1:t+1}) = \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) P(x_t|e_{1:t})$$
"forward algorithm"

To compute $P(X_{t+1}|e_{1:t+1})$, recursive formula
(similar to what we did)

$$P(X_{t+1}|e_{1:t+1}) = \alpha \ P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) \ P(x_t|e_{1:t})$$

Derivation

$$P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{1:t}, e_{t+1})$$

$$= \frac{1}{P(e_{t+1}|e_{1:t})} \ P(e_{t+1}|X_{t+1}, e_{1:t})P(X_{t+1}|e_{1:t}) \quad \text{(Bayes)}$$

$$= \alpha \ P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t}) \quad \text{(Markov assumption)}$$

$$= \alpha \ P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) \ P(x_t|e_{1:t})$$
“forward algorithm”

To compute $P(X_{t+1}|e_{1:t+1})$, recursive formula
(similar to what we did)

$$P(X_{t+1}|e_{1:t+1}) = \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) P(x_t|e_{1:t})$$

Or in matrix form, if $f_t$ is the vector of $f_t(x) = P(X_t = x, e_{1:t})$:

$$f_{t+1} = \alpha O_{t+1} T^T f_t$$

$O_t$ - observation matrix corresponding to $E_t$.
$\alpha$ - normalizing constant to 1 (equal to $\frac{1}{P(e_{1:t})}$).
“backward algorithm”

Let $b_t$ be the vector of $b_{k:t}(x) = P(e_{k:t}, X_{k-1})$:

$$b_{k+1:t} = T \cdot O_{k+1} b_{k+2:t}$$

$O_t$ - observation matrix corresponding to $E_t$. 

Summary

• HMMs - useful to model time-dependent variables / problems (e.g. treating patients with changing biometrics over time)

• Example - text tagging

• Viterbi algorithm (dynamic programming) to find the most likely assignment to the hidden variables. (assuming the transition probabilities are known)

• Independence assumptions allow “forward” + “backward” computations of conditional probabilities