HMMs

CompSci 570

Ronald Parr
Department of Computer Science
Duke University

Overview

• Bayes nets are (mostly) atemporal
• Need a way to talk about a world that changes over time
• Necessary for planning
• Many important applications
  – Target tracking
  – Patient/factory monitoring
  – Speech recognition
Back to Atomic Events

• We began talking about probabilities from the perspective of atomic events

• An atomic event is an assignment to every random variable in the domain

• For $n$ binary random variables, there are $2^n$ possible atomic events

States

• When reasoning about time, we often call atomic events states

• States, like atomic events, form a mutually exclusive and jointly exhaustive partition of the space of possible events

• We can describe how a system behaves with a state-transition diagram
Example: Speech Recognition

- Speech is broken down into atoms called phonemes, e.g., see arpanet: [http://en.wikipedia.org/wiki/Arpabet](http://en.wikipedia.org/wiki/Arpabet)
- Phonemes are pulled from the audio stream using a variety of techniques
- Words are stochastic finite automata (HMMs) with outputs that are phonemes
You say tomato, I say...

Real variations in speech between speakers can be much more subtle and complicated than this: How do we learn these?

Using HMMs for Speech Recognition

• Create one HMM for every word
• Upon hearing a word:
  – Break down word into string of phonemes
  – Compute probability that string came from each HMM
  – Go with word (HMM) that assigns highest probability to string
State Transition Diagrams

• Make a lot of assumptions
  – Transition probabilities don’t change over time (stationarity)
  – The event space does not change over time
  – Probability distribution over next states depends only on the current state (Markov assumption)
  – Time moves in uniform, discrete increments

The Markov Assumption

• Let $S_t$ be a random variable for the state at time $t$

• $P(S_t | S_{t-1}, ..., S_0) = P(S_t | S_{t-1})$

• (Use subscripts for time; $S_0$ is different from $S_0$)

• Markov is special kind of conditional independence

• Future is independent of past given current state
Markov Models

• A system with states that obey the Markov assumption is called a Markov Model

• A sequence of states resulting from such a model is called a Markov Chain

• The mathematical properties of Markov chains are studied heavily in mathematics, statistics, computer science, electrical engineering, etc.

What’s The Big Deal?

• A system that obeys the Markov property can be described succinctly with a transition matrix, where the $i,j$th entry of the matrix is $P(S_j|S_i)$

• The Markov property ensures that we can maintain this succinct description over a potentially infinite time sequence

• Properties of the system can be analyzed in terms of properties of the transition matrix
  – Steady-state probabilities
  – Convergence rate, etc.
Observations

- Introduce $E_t$ for the observation at time $t$
- Observations are like evidence
- Define the probability distribution over observations as function of current state: $P(E|S)$
- Assume observations are conditionally independent of other variables given current state
- Assume observation probabilities are stationary

A Bayes Net View of HMMs

Note: These are random variables, not states!
Applications

• Monitoring/Filtering: $P(S_t:E_0...E_t)$
  – $S$ is the current status of the patient/factory
  – $E$ is the current measurement

• Prediction: $P(S_t:E_0...E_k), t>k$
  – $S$ is the current/future position of an object
  – $E$ are our past observations
  – Project $S$ into the future

Applications

• Smoothing/hindsight: $P(S_k:E_0...E_t), t>k$
  – Update view of the past based upon future
  – Diagnosis: Factory exploded at time $t=20$, what happened at $t=5$ to cause this?

• Most likely explanation
  – What is the most likely sequence of events (from start to finish) to explain observations?
  – NB: Answer is a single path, not a distribution
Example: Robot Self Tracking

- Consider Roomba-like robot with:
  - Known map of the room
  - 4-way proximity sensors
  - Unknown initial position (kidnapped robot problem)
- We consider a discretized version of this problem
  - Map discretized into grid
  - Discrete, one-square movements

(Images from iRobot’s web page)

Simple Map, Kidnapped Robot
Robot Senses

Obstacles up and down, none left and right

Robot Updates Distribution
Robot Moves Right, Updates

Obstacles up and down, none left and right

Robot Updates Probabilities
What Just Happened

• This was an example of robot tracking

• We can also do:
  – Prediction (where would the robot be?)
  – Smoothing (where was the robot?)
  – Most likely path (what path did robot take?)

Prediction

Suppose the Robot Moves Right Twice
New Robot Position Distribution

Are these probabilities uniform?

What Isn’t Realistic Here?

- Where does the map come from?
- Does the robot really have these sensors?
- Are right/left/up/down the correct sort of actions? (Even if the robot has a map, it may not know its orientation.)
- Are robot actions deterministic?
- Are sensing actions deterministic?
- Would a probabilistic sensor model conflate sensor noise and incorrect modeling?
- Can the world be modeled as a grid?

- Good news: Despite these problems, robotic mapping and localization (tracking) can actually be made to work!
Most Likely (Viterbi) Path

From definition of Bayes net (or HMM):

\[ P(S_0...S_T | e_0...e_T) \propto P(S_0)P(e_0 | S_0) \prod_{i=1}^{T} P(S_i | S_{i-1})P(e_i | S_i) \]

Suppose we want max probability sequence of states:

\[
\max_{S_0...S_T} P(S_0...S_T | e_0...e_T) = \max_{S_0...S_T} P(S_0)P(e_0 | S_0) \prod_{i=1}^{T} P(S_i | S_{i-1})P(e_i | S_i)
\]

\[
= \max_{S_0...S_T} P(e_0 | S_0) \prod_{i=1}^{T} P(S_i | S_{i-1})P(e_i | S_i) \max_{S_0...S_T} P(S_0)P(e_0 | S_0)
\]

\[
= \max_{S_0...S_T} P(e_0 | S_0) \prod_{i=1}^{T} P(S_i | S_{i-1})P(e_i | S_i) \max_{S_0...S_T} P(S_0)P(e_0 | S_0)
\]

Keep distributing max over product! Compare with Dijkstra’s algorithm, dynamic programming.

Implementing the Viterbi Algorithm (forward part)

- \( P_0 = \) initial distribution
- For \( t = 1 \) to \( T \)
  - \( P_t = [0...0] \)
  - For \( \text{Next}S = 1 \) to \( n \)
    - For \( \text{Prev}S = 1 \) to \( n \)
      - \( P_t[\text{Next}S] = \max[P_t[\text{Next}S], P_t[\text{Prev}S]*P(\text{Next}S | \text{Prev}S)] \)
      - \( P_t[\text{Next}S] = P_t[\text{Next}S]P(e_t | \text{Next}S) \)

What is is needed: Store \( \arg\max \), reconstruct path in backward pass (compare with reconstructing the path in search)
Algebraic View: Our Main Tool

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

Conditional Probability with Extra Evidence

- Recall: \( P(AB) = P(A \mid B)P(B) \)
- Add extra evidence \( C \)
  (can be a set of variables)
- \( P(AB \mid C) = P(A \mid BC)P(B \mid C) \)
Extending Bayes Rule

\[ P(A \mid BC) = \frac{P(B \mid AC)P(A \mid C)}{P(B \mid C)} \]

How to think about this: The C is like “extra” evidence. This forces us into one corner of the event space. Given that we are in this corner, everything behaves the same.

Using Conditional Independence And the Markov Property

- Conditional probability w/extra evidence:
  - \( P(AB \mid C) = P(A \mid BC)P(B \mid C) \)

- \( P(S_tS_{t-1} \mid e_{t-1}e_0) = P(S_t \mid S_{t-1}e_{t-1}e_0) \cdot P(S_{t-1} \mid e_{t-1}e_0) = P(S_t \mid S_{t-1}) \cdot P(S_{t-1} \mid e_{t-1}e_0) \)
Monitoring

We want: \( P(S_t | e_t \ldots e_0) \)

\[
P(S_t | e_t \ldots e_0) = \frac{P(e_t | S_t, e_{t-1} \ldots e_0)P(S_t | e_{t-1} \ldots e_0)}{P(e_t | e_{t-1} \ldots e_0)}
\]

\[
= \alpha P(e_t | S_t) P(S_t | e_{t-1} \ldots e_0)
\]

\[
= \alpha P(e_t | S_t) \sum_{S_{t-1}} P(S_t | S_{t-1}) P(S_{t-1} | e_{t-1} \ldots e_0)
\]

Recursive

Implementation

NB: These are conditioned on \( e_0 \ldots e_{t-1} \), but condition is omitted to fit in box.

NB: These are conditioned on \( e_0 \ldots e_t \), but condition is omitted to fit in box.

Maintain a vector of probabilities at each time step

Arcs correspond \( P(s_t | s_{t-1}) \) in summation of previous slide:

- Each color is a different iteration through the loop
- Add up probability of all paths that lead to each state
Implementation

Maintain a vector of probabilities at each time step

- Each color is a different iteration through the loop
- Add up probability of all paths that lead to each state

NB: These are conditioned on \( e_0 \ldots e_{t-1} \), but condition is omitted to fit in box.

Stay tuned for the rest of the slides...
Example

- $W = \text{grad student is working}$
- $R = \text{student has produced results}$
- Advisor observes whether student has produced results
- Infer whether student is working given observations

\[
P(w_{t+1} | w_t) = 0.8 \\
P(w_{t+1} | \bar{w}_t) = 0.3 \\
P(r | w) = 0.6 \\
P(r | \bar{w}) = 0.2
\]

Problem

- Assume student starts semester in a productive (working) state
- Prof. has observed two consecutive meetings without results
- What is probability the student was working in the second week?
Let's Do The Math

\[ P(\omega_{t+1} | \omega_t) = 0.8 \]
\[ P(\omega_{t+1} | \bar{\omega}_t) = 0.3 \]
\[ P(\tilde{r} | \omega) = 0.6 \]
\[ P(\tilde{r} | \bar{\omega}) = 0.2 \]

\[ P(W_2 | \bar{r}_1 \bar{r}_1) = \alpha_1 P(r_2 | W_2) \sum_{W_1} P(W_2 | W_1) P(W_1 | \bar{r}_1) \]
\[ P(W_1 | \bar{r}_1) = \alpha_2 P(r_1 | \bar{r}_1) \sum_{W_0} P(W_1 | W_0) P(W_0) \]
\[ P(\omega_1 | \bar{r}_1) = \alpha_2 0.4(0.8 \times 1 + 0.3 \times 0) = \alpha_2 0.32 \]
\[ P(\bar{\omega}_1 | \bar{r}_1) = \alpha_2 0.8(0.2 \times 1 + 0.7 \times 0) = \alpha_2 0.16 \]
\[ P(\omega_1 | \bar{r}_1) = 0.67, P(\bar{\omega}_1 | \bar{r}_1) = 0.33 \]

More Math

\[ P(\omega_{t+1} | \omega_t) = 0.8 \]
\[ P(\omega_{t+1} | \bar{\omega}_t) = 0.3 \]
\[ P(\tilde{r} | \omega) = 0.6 \]
\[ P(\tilde{r} | \bar{\omega}) = 0.2 \]
\[ P(\omega_1 | \bar{r}) = 0.67 \]
\[ P(\bar{\omega}_1 | \bar{r}) = 0.33 \]

\[ P(W_2 | \bar{r}_2 \bar{r}_1) = \alpha_1 P(r_2 | W_2) \sum_{W_1} P(W_2 | W_1) P(W_1 | \bar{r}_1) \]
\[ P(W_2 | \bar{r}_2 \bar{r}_1) = \alpha_1 0.4(0.8 \times 0.67 + 0.3 \times 0.33) = \alpha_1 0.25 \]
\[ P(\bar{\omega}_2 | \bar{r}_2 \bar{r}_1) = \alpha_1 0.8(0.2 \times 0.67 + 0.7 \times 0.33) = \alpha_1 0.292 \]
\[ P(\omega_2 | \bar{r}_2 \bar{r}_1) = 0.46, P(\bar{\omega}_2 | \bar{r}_2 \bar{r}_1) = 0.54 \]
Hindsight

\[ P(S_k | e_t \ldots e_0) = \alpha P(e_t \ldots e_{k+1} | S_k, e_k \ldots e_0)P(S_k | e_k \ldots e_0) \]

\[ = \alpha P(e_t \ldots e_{k+1} | S_k)P(S_k | e_k \ldots e_0) \]

Monitoring!

\[ P(e_t \ldots e_{k+1} | S_k) = \sum_{S_{k+1}} P(e_t \ldots e_{k+1} | S_{k+1})P(S_{k+1} | S_k) \]

\[ = \sum_{S_{k+1}} P(e_t \ldots e_{k+1} | S_{k+1})P(S_{k+1} | S_k) \]

Recursive

Implementation

There is no \( e_{t+1} \)!
What does this mean?
Can assume all ones, or just ignore \( P(e_{t+1} | s) \)

Black, blue, green are different iterations through loop implied by summation on previous slide

\[ P(s_i | s_{i-1}) \]

Weight by \( P(e_{i-1} | s_i) \)
Weight by \( P(e_i | s_i) \)
Weight by \( P(e_{i+1} | s_i) \)

\( \leftarrow \ldots \text{backwards} \)
Hindsight (smoothing) Summary

- **Forward**: Compute time $k$ state distribution given
  - Forward distribution up to $k$
  - Observations up to $k$
  - Equivalent to monitoring up to $k$

- **Backward**: Compute conditional evidence distribution after $k$
  - Work backward from $t$ to $k$

- Smoothed state distribution is *proportional* to product of forward and backward components

Implementation Sanity Checks

- Make sure you never double count observations:
  Any *path* through the HMM should multiply by each $P(e_i|s_i)$ exactly once
  (think of forward/backward as summing probabilities of paths, weighted by observations)

- Make sure you handle base cases
  - Forward message starts with initial distribution at time 0
  - Observations beyond the horizon can be ignored
    (or assume first backwards message is all ones)
Problem II

Can we revise our estimate of the probability that the student worked at step 1?

We initially thought:

\[ P(w_1 \mid \bar{r}_1) = 0.67, P(w_1 \mid \bar{r}_1) = 0.33 \]

Since the employee didn’t have results at time 2, is it now less likely that he was working at time 1?

Let’s Do More Math

\[ P(W_1 \mid r_2 \bar{r}_1) = \alpha P(W_1 \mid \bar{r}_1) P(r_2 \mid W_1) \]

\[ P(r_2 \mid w_1) = \sum_{w_2} P(r_2 \mid w_2) P(w_2 \mid w_1) \]

\[ P(r_2 \mid w_1) = (0.4 \times 0.8 + 0.8 \times 0.2) = 0.48 \]

\[ P(\bar{r}_2 \mid w_1) = (0.4 \times 0.3 + 0.8 \times 0.7) = 0.68 \]

\[ P(w_1 \mid r_2 \bar{r}_1) = \alpha 0.67 \times 0.48 = \alpha 0.3216 \]

\[ P(\bar{w}_1 \mid r_2 \bar{r}_1) = \alpha 0.33 \times 0.68 = \alpha 0.2244 \]

\[ P(w_1 \mid r_2 \bar{r}_1) = 0.59, P(\bar{w}_1 \mid r_2 \bar{r}_1) = 0.41 \]
Checkpoint

• Done: Forward Monitoring and Backward Smoothing

• Monitoring is recursive from the past to the present
• Backward smoothing requires two recursive passes (forward then backward)
• Implemented as two loops (not recursively)

• Called the forward-backward algorithm
  – Independently discovered many times throughout history
  – Was classified for many years by US Govt.

What’s Left?

• We have seen that filtering and smoothing can be done efficiently, so what’s the catch?

• We’re still working at the level of atomic events

• There are too many atomic events!

• We need a generalization of Bayes nets to let us think about the world at the level of state variables and not states
Dynamic Bayes Nets

Time → \( t \) \( \rightarrow \) \( t+1 \)

\begin{align*}
X & \rightarrow X' \\
Y & \rightarrow Y' \\
Z & \rightarrow Z'
\end{align*}

State Variables

CPT

<table>
<thead>
<tr>
<th>Event</th>
<th>( P(z') )</th>
<th>( P(\neg z') )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_t z_t )</td>
<td>0.25</td>
<td>0.75</td>
</tr>
<tr>
<td>( y_t \neg z_t )</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>( \neg y_t z_t )</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>( \neg y_t \neg z_t )</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Working With DBNs

Can we do variable elimination for DBNs?
Harsh Reality

- While BN inference in the static case was a very nice story, there are essentially no tractable, exact algorithms for DBNs

- Dealing with intractability
  - Approximate inference algorithms
    - Variational methods
    - Assumed density filtering (ADF)
  - Sampling methods
    - Sequential Importance sampling
    - Sequential Importance Sampling with Resampling (SISR, particle filter, condensation, etc.)

Continuous Variables
(outside of scope of class)

- How do we represent a probability distribution over a continuous variable?
  - Probability density function
  - Summations become integrals

- Very messy except for some special cases:
  - Distribution over variable X at time t+1 is a multivariate normal with a mean that is a linear function of the variables at the previous time step
  - This is a linear-Gaussian model
Inference in Linear Gaussian Models

- Filtering and smoothing integrals have closed form solution

- Elegant solution known as the Kalman filter
  - Used for tracking projectiles (radar)
  - State is modeled as a set of linear equations
    - $S = vt$
    - $V = at$
  - What about pilot controls?

HMM Conclusion

- Elegant algorithms for temporal reasoning over discrete atomic events, Gaussian continuous variables
  (many practical systems are approximately such)

- Exact Bayes net methods don’t generalize well to state variable representation in the temporal case: little hope for exponential savings

- Approximations required for large/complex/continuous systems