CS 188: Artificial Intelligence
Fall 2008

Lecture 19: HMMs
11/4/2008

Dan Klein – UC Berkeley
Announcements

- Midterm solutions up, submit regrade requests within a week
- Midterm course evaluation up on web, please fill out!
- Final contest is posted!
VPI Example

MEU with no evidence

\[ \text{MEU}(\varnothing) = \max_a \text{EU}(a) = 70 \]

MEU if forecast is bad

\[ \text{MEU}(F = \text{bad}) = \max_a \text{EU}(a|\text{bad}) = 53 \]

MEU if forecast is good

\[ \text{MEU}(F = \text{good}) = \max_a \text{EU}(a|\text{good}) = 95 \]

Forecast distribution

\[
\begin{array}{c|ccc}
F & P(F) \\
\hline
\text{good} & 0.59 \\
\text{bad} & 0.41 \\
\end{array}
\]

\[ 0.59 \cdot (95 - 70) + 0.41 \cdot (53 - 70) \]

\[ 0.59 \cdot (+25) + 0.41 \cdot (-17) = +22 \]

\[
\text{VPI}_e(E') = \sum_{e'} P(e'|e) \left( \text{MEU}(e, e') - \text{MEU}(e) \right)
\]
VPI Properties

- Nonnegative in expectation

\[ \forall E', e : \text{VPI}_e(E') \geq 0 \]

- Nonadditive --- consider, e.g., obtaining \( E_j \) twice

\[ \text{VPI}_e(F_j, F_k) \neq \text{VPI}_e(F_j) + \text{VPI}_e(F_k) \]

- Order-independent

\[ \text{VPI}_e(E_j, E_k) = \text{VPI}_e(E_j) + \text{VPI}_{e,E_j}(E_k) \]

\[ = \text{VPI}_e(E_k) + \text{VPI}_{e,E_k}(E_j) \]
Reasoning over Time

- Often, we want to reason about a sequence of observations
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring

- Need to introduce time into our models
- Basic approach: hidden Markov models (HMMs)
- More general: dynamic Bayes’ nets
A Markov model is a chain-structured BN
- Each node is identically distributed (stationarity)
- Value of X at a given time is called the state
- As a BN:

\[ P(X_1) \quad P(X|X_{-1}) \]

Parameters: called transition probabilities or dynamics, specify how the state evolves over time (also, initial probs)

[DEMO: Battleship]
Conditional Independence

- **Basic conditional independence:**
  - Past and future independent of the present
  - Each time step only depends on the previous
  - This is called the (first order) Markov property

- **Note that the chain is just a (growing) BN**
  - We can always use generic BN reasoning on it (if we truncate the chain)
Example: Markov Chain

- **Weather:**
  - States: $X = \{\text{rain, sun}\}$
  - Transitions:
    - Initial distribution: 1.0 sun
    - What's the probability distribution after one step?

$$P(X_2 = \text{sun}) = P(X_2 = \text{sun}|X_1 = \text{sun})P(X_1 = \text{sun}) + P(X_2 = \text{sun}|X_1 = \text{rain})P(X_1 = \text{rain})$$

$$0.9 \cdot 1.0 + 0.1 \cdot 0.0 = 0.9$$

This is a CPT, not a BN!
Mini-Forward Algorithm

- Question: probability of being in state x at time t?
- Slow answer:
  - Enumerate all sequences of length t which end in s
  - Add up their probabilities

\[
P(X_t = \text{sun}) = \sum_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, \text{sun})
\]

\[
P(X_1 = \text{sun})P(X_2 = \text{sun}|X_1 = \text{sun})P(X_3 = \text{sun}|X_2 = \text{sun})P(X_4 = \text{sun}|X_3 = \text{sun})
\]

\[
P(X_1 = \text{sun})P(X_2 = \text{rain}|X_1 = \text{sun})P(X_3 = \text{sun}|X_2 = \text{rain})P(X_4 = \text{sun}|X_3 = \text{sun})
\]

\[
\vdots
\]

\[
\vdots
\]
Mini-Forward Algorithm

- Better way: cached incremental belief updates
  - An instance of variable elimination!

\[
P(x_1) = \text{known}
\]

\[
P(x_t) = \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1})
\]

Forward simulation
Example

- From initial observation of sun

\[
\begin{pmatrix}
1.0 \\
0.0
\end{pmatrix}
\begin{pmatrix}
0.9 \\
0.1
\end{pmatrix}
\begin{pmatrix}
0.82 \\
0.18
\end{pmatrix}
\begin{pmatrix}
0.5 \\
0.5
\end{pmatrix}
\]

\[P(X_1) \quad P(X_2) \quad P(X_3) \quad P(X_\infty)\]

- From initial observation of rain

\[
\begin{pmatrix}
0.0 \\
1.0
\end{pmatrix}
\begin{pmatrix}
0.1 \\
0.9
\end{pmatrix}
\begin{pmatrix}
0.18 \\
0.82
\end{pmatrix}
\begin{pmatrix}
0.5 \\
0.5
\end{pmatrix}
\]

\[P(X_1) \quad P(X_2) \quad P(X_3) \quad P(X_\infty)\]
Stationary Distributions

- If we simulate the chain long enough:
  - What happens?
  - Uncertainty accumulates
  - Eventually, we have no idea what the state is!

- Stationary distributions:
  - For most chains, the distribution we end up in is independent of the initial distribution (but not always uniform!)
  - Called the **stationary distribution** of the chain
  - Usually, can only predict a short time out

[DEMO: Battleship]
Web Link Analysis

- **PageRank over a web graph**
  - Each web page is a state
  - Initial distribution: uniform over pages
  - Transitions:
    - With prob. $c$, uniform jump to a random page (dotted lines)
    - With prob. $1-c$, follow a random outlink (solid lines)

- **Stationary distribution**
  - Will spend more time on highly reachable pages
  - E.g. many ways to get to the Acrobat Reader download page
  - Somewhat robust to link spam (but not immune)
  - Google 1.0 returned the set of pages containing all your keywords in decreasing rank, now all search engines use link analysis along with many other factors
Hidden Markov Models

- Markov chains not so useful for most agents
  - Eventually you don’t know anything anymore
  - Need observations to update your beliefs

- Hidden Markov models (HMMs)
  - Underlying Markov chain over states S
  - You observe outputs (effects) at each time step
  - As a Bayes’ net:
An HMM is defined by:

- Initial distribution: \( P(X_1) \)
- Transitions: \( P(X|X_{-1}) \)
- Emissions: \( P(E|X) \)
Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process, future depends on past via the present
  - Current observation independent of all else given current state

Quiz: does this mean that observations are independent given no evidence?
  - [No, correlated by the hidden state]
Real HMM Examples

- **Speech recognition HMMs:**
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)

- **Machine translation HMMs:**
  - Observations are words (tens of thousands)
  - States are translation options

- **Robot tracking:**
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)
Filtering / Monitoring

- Filtering, or monitoring, is the task of tracking the distribution $B(X)$ (the belief state).
- We start with $B(X)$ in an initial setting, usually uniform.
- As time passes, or we get observations, we update $B(X)$. 
Example: Robot Localization

Sensor model: never more than 1 mistake
Motion model: may not execute action with small prob.
Example: Robot Localization

\[ t=1 \]

Prob

0 1
Example: Robot Localization
Example: Robot Localization

t=3

Prob
0 1
Example: Robot Localization

\[ t=4 \]
Example: Robot Localization

Prob
0 1

$t=5$
Passage of Time

- Assume we have current belief $P(X | \text{evidence to date})$

$$B(X_t) = P(X_t|e_{1:t})$$

- Then, after one time step passes:

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

- Or, compactly:

$$B'(X_{t+1}) = \sum_{x_t} P(X'|x) B(x_t)$$

- Basic idea: beliefs get “pushed” through the transitions
  - With the “B” notation, we have to be careful about what time step the belief is about, and what evidence it includes
Example: Passage of Time

- As time passes, uncertainty “accumulates”

\[
B'(X) = \sum_x P(X'|x)B(x)
\]

Transition model: ships usually go clockwise
Observation

- Assume we have current belief $P(X \mid \text{previous evidence})$:

  $$B'(X_{t+1}) = P(X_{t+1} \mid e_{1:t})$$

- Then:

  $$P(X_{t+1} \mid e_{1:t+1}) \propto P(e_{t+1} \mid X_{t+1}) P(X_{t+1} \mid e_{1:t})$$

- Or:

  $$B(X_{t+1}) \propto P(e \mid X) B'(X_{t+1})$$

- Basic idea: beliefs reweighted by likelihood of evidence

- Unlike passage of time, we have to renormalize
Example: Observation

- As we get observations, beliefs get reweighted, uncertainty “decreases”

\[ B(X) \propto P(e|X)B'(X) \]
Example HMM
Example HMM
Updates: Time Complexity

- Every time step, we start with current $P(X \mid \text{evidence})$
- We must update for time:
  \[
P(X_t \mid e_{1:t-1}) \propto \sum_{x_{t-1}} P(X_t \mid x_{t-1})P(x_{t-1} \mid e_{1:t-1})
  \]
- We must update for observation:
  \[
P(X_t \mid e_{1:t}) \propto P(e_t \mid X_t)P(X_t \mid e_{1:t-1})
  \]
- So, linear in time steps, quadratic in number of states $|X|$
- Of course, can do both at once, too
The Forward Algorithm

- Can do belief propagation exactly as in previous slides, renormalizing each time step.
- In the standard forward algorithm, we actually calculate $P(X,e)$, without normalizing (it’s a special case of VE).

\[
P(x_t|e_{1:t}) \propto P(x_t, e_{1:t})
\]
\[
= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t})
\]
\[
= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t)
\]
\[
= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})
\]
Particle Filtering

- Sometimes $|X|$ is too big to use exact inference
  - $|X|$ may be too big to even store $B(X)$
  - E.g. $X$ is continuous
  - $|X|^2$ may be too big to do updates

- Solution: approximate inference
  - Track samples of $X$, not all values
  - Time per step is linear in the number of samples
  - But: number needed may be large

- This is how robot localization works in practice
Particle Filtering: Time

- Each particle is moved by sampling its next position from the transition model

\[ x' = \text{sample}(P(X'|x)) \]

- This is like prior sampling – samples are their own weights
- Here, most samples move clockwise, but some move in another direction or stay in place

- This captures the passage of time
  - If we have enough samples, close to the exact values before and after (consistent)
Particle Filtering: Observation

- Slightly trickier:
  - We don’t sample the observation, we fix it
  - This is similar to likelihood weighting, so we downweight our samples based on the evidence

\[ w(x) = P(e | x) \]

\[ B(X) \propto P(e | X) B'(X) \]

- Note that, as before, the probabilities don’t sum to one, since most have been downweighted (they sum to an approximation of \( P(e) \))
Particle Filtering: Resampling

- Rather than tracking weighted samples, we resample

- N times, we choose from our weighted sample distribution (i.e. draw with replacement)

- This is equivalent to renormalizing the distribution

- Now the update is complete for this time step, continue with the next one
Robot Localization

- **In robot localization:**
  - We know the map, but not the robot’s position
  - Observations may be vectors of range finder readings
  - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store $B(X)$
  - Particle filtering is a main technique

- [DEMOS]
SLAM

- SLAM = Simultaneous Localization And Mapping
  - We do not know the map or our location
  - Our belief state is over maps and positions!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

- [DEMOS]

DP-SLAM, Ron Parr