Lecture 11
Dynamic Bayesian Networks and Hidden Markov Models
Decision Trees

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Slides by Stuart Russell and Peter Norvig
Course Overview

- Introduction
  - Artificial Intelligence
  - Intelligent Agents

- Search
  - Uninformed Search
  - Heuristic Search

- Adversarial Search
  - Minimax search
  - Alpha-beta pruning

- Knowledge representation and Reasoning
  - Propositional logic
  - First order logic
  - Inference

- Uncertain knowledge and Reasoning
  - Probability and Bayesian approach
  - Bayesian Networks
    - Hidden Markov Chains
    - Kalman Filters

- Learning
  - Decision Trees
  - Maximum Likelihood
  - EM Algorithm
  - Learning Bayesian Networks
  - Neural Networks
  - Support vector machines
Exercise
Uncertainty over Time
Speech Recognition
Learning

Performance of approximation algorithms

- **Absolute approximation:** \(|P(X|e) - \hat{P}(X|e)| \leq \epsilon\)
Performance of approximation algorithms

• Absolute approximation: \(|P(X|e) - \hat{P}(X|e)| \leq \epsilon\)

• Relative approximation: \(\frac{|P(X|e) - \hat{P}(X|e)|}{P(X|e)} \leq \epsilon\)
Performance of approximation algorithms

- Absolute approximation: \(|P(X|e) - \hat{P}(X|e)| \leq \epsilon\)

- Relative approximation: \(\frac{|P(X|e) - \hat{P}(X|e)|}{P(X|e)} \leq \epsilon\)

- Relative \(\Rightarrow\) absolute since \(0 \leq P \leq 1\) (may be \(O(2^{-n})\))
Performance of approximation algorithms

- Absolute approximation: \(|P(X|e) - \hat{P}(X|e)| \leq \epsilon\)

- Relative approximation: \(\frac{|P(X|e) - \hat{P}(X|e)|}{P(X|e)} \leq \epsilon\)

- Relative \(\implies\) absolute since \(0 \leq P \leq 1\) (may be \(O(2^{-n})\))

- Randomized algorithms may fail with probability at most \(\delta\)
Performance of approximation algorithms

- **Absolute approximation**: \(|P(X|e) - \hat{P}(X|e)| \leq \epsilon\)

- **Relative approximation**: \(\frac{|P(X|e) - \hat{P}(X|e)|}{P(X|e)} \leq \epsilon\)

- Relative \(\implies\) absolute since \(0 \leq P \leq 1\) (may be \(O(2^{-n})\))

- Randomized algorithms may fail with probability at most \(\delta\)

- Polytime approximation: \(\text{poly}(n, \epsilon^{-1}, \log \delta^{-1})\)
Performance of approximation algorithms

- **Absolute approximation:** $|P(X|e) - \hat{P}(X|e)| \leq \epsilon$

- **Relative approximation:** $\frac{|P(X|e) - \hat{P}(X|e)|}{P(X|e)} \leq \epsilon$

- Relative $\implies$ absolute since $0 \leq P \leq 1$ (may be $O(2^{-n})$)

- Randomized algorithms may fail with probability at most $\delta$

- Polytime approximation: $\text{poly}(n, \epsilon^{-1}, \log \delta^{-1})$

- Theorem (Dagum and Luby, 1993): both absolute and relative approximation for either deterministic or randomized algorithms are NP-hard for any $\epsilon, \delta < 0.5$
  (Absolute approximation polytime with no evidence—Chernoff bounds)
Summary

Exact inference by variable elimination:
- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by Likelihood Weighting (LW), Markov Chain Monte Carlo Method (MCMC):

- PriorSampling and RejectionSampling unusable as evidence grow
  - LW does poorly when there is lots of (late-in-the-order) evidence
  - LW, MCMC generally insensitive to topology
  - Convergence can be very slow with probabilities close to 1 or 0
  - Can handle arbitrary combinations of discrete and continuous variables
Outline

1. Exercise

2. Uncertainty over Time

3. Speech Recognition

4. Learning
Exercise  
Uncertainty over Time  
Speech Recognition  
Learning  

Wumpus World

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1,4</td>
<td>2,4</td>
<td>3,4</td>
<td>4,4</td>
</tr>
<tr>
<td>1,3</td>
<td>2,3</td>
<td>3,3</td>
<td>4,3</td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td>3,2</td>
<td>4,2</td>
</tr>
<tr>
<td>1,1</td>
<td>2,1</td>
<td>3,1</td>
<td>4,1</td>
</tr>
</tbody>
</table>

\[ P_{ij} = \text{true} \] iff \([i, j]\) contains a pit \n\[ B_{ij} = \text{true} \] iff \([i, j]\) is breezy

Include only \( B_{1,1}, B_{1,2}, B_{2,1} \) in the probability model
Specifying the probability model

The full joint distribution is \( P(P_{1,1}, \ldots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1}) \)

Apply product rule: \( P(B_{1,1}, B_{1,2}, B_{2,1} | P_{1,1}, \ldots, P_{4,4}) P(P_{1,1}, \ldots, P_{4,4}) \)

(Do it this way to get \( P(\text{Effect}|\text{Cause}) \).)
Specifying the probability model

The full joint distribution is \( P(P_{1,1}, \ldots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1}) \)

Apply product rule: \( P(B_{1,1}, B_{1,2}, B_{2,1} \mid P_{1,1}, \ldots, P_{4,4})P(P_{1,1}, \ldots, P_{4,4}) \)

(Do it this way to get \( P(Effect \mid Cause) \).)

First term: 1 if pits are adjacent to breezes, 0 otherwise

Second term: pits are placed randomly, probability 0.2 per square:

\[
P(P_{1,1}, \ldots, P_{4,4}) = \prod_{i,j=1,1}^{4,4} P(P_{i,j}) = 0.2^n \times 0.8^{16-n}
\]

for \( n \) pits.
Observations and query

We know the following facts:

\[ b = \neg b_{1,1} \land b_{1,2} \land b_{2,1} \]
\[ known = \neg p_{1,1} \land \neg p_{1,2} \land \neg p_{2,1} \]

Query is \( P(P_{1,3}|known, b) \)

Define \textit{Unknown} = \( P_{ij} \)s other than \( P_{1,3} \) and \textit{Known}
Observations and query

We know the following facts:

\[ b = \neg b_{1,1} \land b_{1,2} \land b_{2,1} \]
\[ known = \neg p_{1,1} \land \neg p_{1,2} \land \neg p_{2,1} \]

Query is \( P(P_{1,3}|known, b) \)

Define \textit{Unknown} = \( P_{ij} \)'s other than \( P_{1,3} \) and \textit{Known}

For inference by enumeration, we have

\[ P(P_{1,3}|known, b) = \alpha \sum_{unknown} P(P_{1,3}, unknown, known, b) \]

Grows exponentially with number of squares!
Using conditional independence

Basic insight: observations are conditionally independent of other hidden squares given neighbouring hidden squares
Using conditional independence

Basic insight: observations are conditionally independent of other hidden squares given neighbouring hidden squares

Define $\text{Unknown} = \text{Fringe} \cup \text{Other}$

$$P(b|P_{1,3}, \text{Known, Unknown}) = P(b|P_{1,3}, \text{Known, Fringe})$$

Manipulate query into a form where we can use this!
Using conditional independence contd.

\[ P(P_{1,3} | \text{known}, b) = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b) \]
Using conditional independence contd.

\[
P(P_{1,3}|\text{known}, b) = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b)
\]

\[
= \alpha \sum_{\text{unknown}} P(b|P_{1,3}, \text{known}, \text{unknown})P(P_{1,3}, \text{known}, \text{unknown})
\]
Using conditional independence contd.

\[
P(P_{1,3}| \text{known}, b) = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b)
\]

\[
= \alpha \sum_{\text{unknown}} P(b|P_{1,3}, \text{known}, \text{unknown})P(P_{1,3}, \text{known}, \text{unknown})
\]

\[
= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe}, \text{other})P(P_{1,3}, \text{known}, \text{fringe}, \text{other})
\]
Using conditional independence contd.

\[
P(P_{1,3}|\text{known}, b) = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b) \\
= \alpha \sum_{\text{unknown}} P(b|P_{1,3}, \text{known}, \text{unknown})P(P_{1,3}, \text{known}, \text{unknown}) \\
= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe}, \text{other})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \\
= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe})P(P_{1,3}, \text{known}, \text{fringe}, \text{other})
\]
Using conditional independence contd.

\[ P(P_{1,3}|\text{known}, b) = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b) \]

\[ = \alpha \sum_{\text{unknown}} P(b|P_{1,3}, \text{known}, \text{unknown})P(P_{1,3}, \text{known}, \text{unknown}) \]

\[ = \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe}, \text{other})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \]

\[ = \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \]

\[ = \alpha \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe}) \sum_{\text{other}} P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \]
Using conditional independence contd.

\[ P(P_{1,3}|\text{known}, b) = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b) \]

\[ = \alpha \sum_{\text{unknown}} P(b|P_{1,3}, \text{known}, \text{unknown})P(P_{1,3}, \text{known}, \text{unknown}) \]

\[ = \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe}, \text{other})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \]

\[ = \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \]

\[ = \alpha \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe}) \sum_{\text{other}} P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \]

\[ = \alpha \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe}) \sum_{\text{other}} P(P_{1,3})P(\text{known})P(\text{fringe})P(\text{other}) \]
Using conditional independence contd.

\[ P(P_{1,3}|known, b) = \alpha \sum_{unknown} P(P_{1,3}, unknown, known, b) \]

\[ = \alpha \sum_{unknown} P(b|P_{1,3}, known, unknown)P(P_{1,3}, known, unknown) \]

\[ = \alpha \sum_{fringe} \sum_{other} P(b|known, P_{1,3}, fringe, other)P(P_{1,3}, known, fringe, other) \]

\[ = \alpha \sum_{fringe} \sum_{other} P(b|known, P_{1,3}, fringe)P(P_{1,3}, known, fringe, other) \]

\[ = \alpha \sum_{fringe} P(b|known, P_{1,3}, fringe) \sum_{other} P(P_{1,3}, known, fringe, other) \]

\[ = \alpha \sum_{fringe} P(b|known, P_{1,3}, fringe) \sum_{other} P(P_{1,3})P(known)P(fringe)P(other) \]

\[ = \alpha P(known)P(P_{1,3}) \sum_{fringe} P(b|known, P_{1,3}, fringe)P(fringe) \sum_{other} P(other) \]
Using conditional independence contd.

\[
\begin{align*}
P(P_{1,3}|\text{known}, b) &= \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, b) \\
&= \alpha \sum_{\text{unknown}} P(b|P_{1,3}, \text{known}, \text{unknown})P(P_{1,3}, \text{known}, \text{unknown}) \\
&= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe}, \text{other})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \\
&= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b|\text{known}, P_{1,3}, \text{fringe})P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \\
&= \alpha \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe}) \sum_{\text{other}} P(P_{1,3}, \text{known}, \text{fringe}, \text{other}) \\
&= \alpha \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe}) \sum_{\text{other}} P(P_{1,3})P(\text{known})P(\text{fringe})P(\text{other}) \\
&= \alpha P(\text{known})P(P_{1,3}) \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe})P(\text{fringe}) \sum_{\text{other}} P(\text{other}) \\
&= \alpha' P(P_{1,3}) \sum_{\text{fringe}} P(b|\text{known}, P_{1,3}, \text{fringe})P(\text{fringe})
\end{align*}
\]
Using conditional independence contd.

\[
P(P_{1,3} | \text{known}, b) = \alpha' \langle 0.2(0.04 + 0.16 + 0.16), 0.8(0.04 + 0.16) \rangle \\
\approx \langle 0.31, 0.69 \rangle
\]

\[
P(P_{2,2} | \text{known}, b) \approx \langle 0.86, 0.14 \rangle
\]
Outline

1. Exercise

2. Uncertainty over Time

3. Speech Recognition

4. Learning
Outline

♦ Time and uncertainty
♦ Inference: filtering, prediction, smoothing
♦ Hidden Markov models
♦ Kalman filters (a brief mention)
♦ Dynamic Bayesian networks (an even briefer mention)
Time and uncertainty

- The world changes; we need to track and predict it
Time and uncertainty

- The world changes; we need to track and predict it
- Diabetes management vs vehicle diagnosis
Time and uncertainty

- The world changes; we need to track and predict it
- Diabetes management vs vehicle diagnosis
- Basic idea: copy state and evidence variables for each time step
  \[ X_t = \text{set of unobservable state variables at time } t \]
  e.g., \( \text{BloodSugar}_t, \text{StomachContents}_t \), etc.
  \[ E_t = \text{set of observable evidence variables at time } t \]
  e.g., \( \text{MeasuredBloodSugar}_t, \text{PulseRate}_t, \text{FoodEaten}_t \)
The world changes; we need to track and predict it

Diabetes management vs vehicle diagnosis

Basic idea: copy state and evidence variables for each time step

\( X_t = \) set of unobservable state variables at time \( t \)
  
  e.g., \( \text{BloodSugar}_t, \text{StomachContents}_t \), etc.

\( E_t = \) set of observable evidence variables at time \( t \)
  
  e.g., \( \text{MeasuredBloodSugar}_t, \text{PulseRate}_t, \text{FoodEaten}_t \)

This assumes discrete time; step size depends on problem
Time and uncertainty

- The world changes; we need to track and predict it
- Diabetes management vs vehicle diagnosis
- Basic idea: copy state and evidence variables for each time step
  \[ X_t = \text{set of unobservable state variables at time } t \]
  e.g., \( \text{BloodSugar}_t \), \( \text{StomachContents}_t \), etc.
  \[ E_t = \text{set of observable evidence variables at time } t \]
  e.g., \( \text{MeasuredBloodSugar}_t \), \( \text{PulseRate}_t \), \( \text{FoodEaten}_t \)
- This assumes **discrete time**; step size depends on problem
- Notation: \( X_{a:b} = X_a, X_{a+1}, \ldots, X_{b-1}, X_b \)
Markov processes (Markov chains)

Construct a Bayes net from these variables:
Markov processes (Markov chains)

Construct a Bayes net from these variables:
- unbounded number of conditional probability table
- unbounded number of parents
Markov processes (Markov chains)

Construct a Bayes net from these variables:
- unbounded number of conditional probability table
- unbounded number of parents

Markov assumption: $X_t$ depends on bounded subset of $X_{0:t-1}$
First-order Markov process: $P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})$
Second-order Markov process: $P(X_t|X_{0:t-1}) = P(X_t|X_{t-2}, X_{t-1})$
Markov processes (Markov chains)

Construct a Bayes net from these variables:
- unbounded number of conditional probability table
- unbounded number of parents

Markov assumption: \( X_t \) depends on bounded subset of \( X_{0:t-1} \)
First-order Markov process: \( P(X_t | X_{0:t-1}) = P(X_t | X_{t-1}) \)
Second-order Markov process: \( P(X_t | X_{0:t-1}) = P(X_t | X_{t-2}, X_{t-1}) \)

Sensor Markov assumption: \( P(E_t | X_{0:t}, E_{0:t-1}) = P(E_t | X_t) \)
Markov processes (Markov chains)

Construct a Bayes net from these variables:
- unbounded number of conditional probability table
- unbounded number of parents

Markov assumption: $X_t$ depends on bounded subset of $X_{0:t-1}$
First-order Markov process: $P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})$
Second-order Markov process: $P(X_t|X_{0:t-1}) = P(X_t|X_{t-2}, X_{t-1})$

Sensor Markov assumption: $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t)$
Stationary process:
- transition model $P(X_t|X_{t-1})$ and
- sensor model $P(E_t|X_t)$ fixed for all $t$
Example

Exercise
Uncertainty over Time
Speech Recognition
Learning

First-order Markov assumption not exactly true in real world!

Possible fixes:
1. Increase order of Markov process
2. Augment state, e.g., add Temp, Pressure

Example: robot motion. Augment position and velocity with Battery

<table>
<thead>
<tr>
<th>$R_{t-1}$</th>
<th>$P(R_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>0.7</td>
</tr>
<tr>
<td>$f$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R_t$</th>
<th>$P(U_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>0.9</td>
</tr>
<tr>
<td>$f$</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Example

First-order Markov assumption not exactly true in real world!
Possible fixes:

1. **Increase order** of Markov process
2. **Augment state**, e.g., add $\text{Temp}_t$, $\text{Pressure}_t$

Example: robot motion.

Augment position and velocity with $\text{Battery}_t$
Inference tasks

1. Filtering: $P(X_t|e_{1:t})$
   belief state—input to the decision process of a rational agent

2. Prediction: $P(X_{t+k}|e_{1:t})$ for $k > 0$
   evaluation of possible action sequences;
   like filtering without the evidence

3. Smoothing: $P(X_k|e_{1:t})$ for $0 \leq k < t$
   better estimate of past states, essential for learning

4. Most likely explanation: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$
   speech recognition, decoding with a noisy channel
Filtering

Aim: devise a recursive state estimation algorithm:

\[ P(X_{t+1} | e_{1:t+1}) = f(e_{t+1}, P(X_t | e_{1:t})) \]
Filtering

Aim: devise a **recursive** state estimation algorithm:

\[
P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t}))
\]

\[
P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{1:t}, e_{t+1})
\]

\[
= \alpha P(e_{t+1}|X_{t+1}, e_{1:t}) P(X_{t+1}|e_{1:t})
\]

\[
= \alpha P(e_{t+1}|X_{t+1}) P(X_{t+1}|e_{1:t})
\]

I.e., **prediction + estimation.**
Filtering

Aim: devise a **recursive** state estimation algorithm:

\[ P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t})) \]

\[ P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{1:t}, e_{t+1}) \]
\[ = \alpha P(e_{t+1}|X_{t+1}, e_{1:t}) P(X_{t+1}|e_{1:t}) \]
\[ = \alpha P(e_{t+1}|X_{t+1}) P(X_{t+1}|e_{1:t}) \]

I.e., **prediction + estimation**. Prediction by summing out \( X_t \):

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

\( f_{1:t+1} = \text{Forward}(f_{1:t}, e_{t+1}) \) where \( f_{1:t} = P(X_t|e_{1:t}) \)

Time and space **constant** (independent of \( t \)) by keeping track of \( f \)
Filtering example

\begin{align*}
\begin{array}{|c|c|}
\hline
R_{t-1} & P(R_t) \\
\hline
\text{t} & 0.7 \\
\text{f} & 0.3 \\
\hline
\end{array}
\end{align*}

\begin{align*}
\begin{array}{|c|c|}
\hline
R_t & P(U_t) \\
\hline
\text{t} & 0.9 \\
\text{f} & 0.2 \\
\hline
\end{array}
\end{align*}
Filtering example

<table>
<thead>
<tr>
<th>$R_{t-1}$</th>
<th>$P(R_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>0.7</td>
</tr>
<tr>
<td>$f$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R_t$</th>
<th>$P(U_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>0.9</td>
</tr>
<tr>
<td>$f$</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Smoothing

\[ \begin{align*}
X_0 & \rightarrow X_1 & \cdots & \rightarrow X_k & \rightarrow X_t \\
E_1 & \rightarrow & E_k & \rightarrow & E_t
\end{align*} \]

\[
\text{Divide evidence } e_1: t \text{ into } e_1:k, e_k+1:t:
\]

\[
P(X_k | e_1:k) = \alpha P(X_k | e_1:k, e_k+1:t) = \alpha P(X_k | e_1:k) P(e_k+1:t | X_k, e_1:k) \]

\[
P(e_k+1:t | X_k, e_1:k) = \sum_{x_k+1} P(e_k+1:t | x_{k+1}) P(x_{k+1} | X_k, e_1:k)
\]

Backward message computed by a backwards recursion:

\[
P(e_k+1:t | X_k) = \sum_{x_{k+1}} P(e_k+1:t | x_{k+1}) P(x_{k+1} | X_k)
\]
Divide evidence $e_{1:t}$ into $e_{1:k}$, $e_{k+1:t}$:

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

$$= \alpha P(X_k|e_{1:k})P(e_{k+1:t}|X_k, e_{1:k})$$

$$= \alpha P(X_k|e_{1:k})P(e_{k+1:t}|X_k)$$

$$= \alpha f_{1:k}b_{k+1:t}$$
Smoothing

Divide evidence $e_{1:t}$ into $e_{1:k}$, $e_{k+1:t}$:

\[
P(X_k | e_{1:t}) = P(X_k | e_{1:k}, e_{k+1:t})
= \alpha P(X_k | e_{1:k}) P(e_{k+1:t} | X_k, e_{1:k})
= \alpha P(X_k | e_{1:k}) P(e_{k+1:t} | X_k)
= \alpha f_{1:k} b_{k+1:t}
\]

Backward message computed by a backwards recursion:

\[
P(e_{k+1:t} | X_k) = \sum_{x_{k+1}} P(e_{k+1:t} | X_k, x_{k+1}) P(x_{k+1} | X_k)
= \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1}) P(x_{k+1} | X_k)
= \sum_{x_{k+1}} P(e_{k+1} | x_{k+1}) P(e_{k+2:t} | x_{k+1}) P(x_{k+1} | X_k)
\]
If we want to smooth the whole sequence:

Forward–backward algorithm: cache forward messages along the way
Time linear in $t$ (polytree inference), space $O(t|f|)$
Most likely explanation

Most likely sequence $\neq$ sequence of most likely states (joint distr.)!
Most likely explanation

Most likely sequence $\neq$ sequence of most likely states (joint distr.)!

Most likely path to each $x_{t+1}$

$= \text{most likely path to some } x_t \text{ plus one more step}$

$$
\max_{x_1 \ldots x_t} P(x_1, \ldots, x_t, X_{t+1}|e_{1:t+1})
$$

$$
= P(e_{t+1}|X_{t+1}) \max_{x_t} \left( P(X_{t+1}|x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t|e_{1:t}) \right)
$$
Most likely explanation

Most likely sequence $\neq$ sequence of most likely states (joint distr.)!
Most likely path to each $x_{t+1}$

$\Rightarrow$ most likely path to some $x_t$ plus one more step

$$
\max_{x_1 \ldots x_t} P(x_1, \ldots, x_t, X_{t+1} | e_1:t+1) \\
= P(e_{t+1} | X_{t+1}) \max_{x_t} \left( P(X_{t+1} | x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t | e_1:t) \right)
$$

Identical to filtering, except $f_{1:t}$ replaced by

$$
m_{1:t} = \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, X_t | e_1:t),
$$

I.e., $m_{1:t}(i)$ gives the probability of the most likely path to state $i$. 
Most likely explanation

Most likely sequence \( \neq \) sequence of most likely states (joint distr.)!
Most likely path to each \( x_{t+1} \)
= most likely path to some \( x_t \) plus one more step

\[
\max_{x_1 \ldots x_t} P(x_1, \ldots, x_t, X_{t+1}|e_{1:t+1})
= P(e_{t+1}|X_{t+1}) \max_{x_t} \left( P(X_{t+1}|x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t|e_{1:t}) \right)
\]

Identical to filtering, except \( f_{1:t} \) replaced by

\[
m_{1:t} = \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, X_t|e_{1:t}),
\]

I.e., \( m_{1:t}(i) \) gives the probability of the most likely path to state \( i \).
Update has sum replaced by max, giving the Viterbi algorithm:

\[
m_{1:t+1} = P(e_{t+1}|X_{t+1}) \max_{x_t} (P(X_{t+1}|x_t)m_{1:t})
\]
Viterbi example

Exercise
Uncertainty over Time
Speech Recognition Learning

Rain 1
true
false
true
false
true
false
true
false
true
false
.8182
.1818
m 1:1

Rain 2
true
false
true
false
true
false
true
false
true
false
.5155
.0491
m 1:2

Rain 3
true
false
true
false
true
false
true
false
true
false
.0361
.1237
m 1:3

Rain 4
true
false
true
false
true
false
true
false
true
false
.0334
.0173
m 1:4

Rain 5
true
false
true
false
true
false
true
false
true
false
.0210
.0024
m 1:5

state space paths
umbrella
most likely paths
Hidden Markov models

$X_t$ is a single, discrete variable (usually $E_t$ is too)

Domain of $X_t$ is $\{1, \ldots, S\}$

Transition matrix $T_{ij} = P(X_t = j | X_{t-1} = i)$, e.g.,

$$
\begin{pmatrix}
0.7 & 0.3 \\
0.3 & 0.7
\end{pmatrix}
$$

Sensor matrix $O_t$ for each time step, diagonal elements $P(e_t | X_t = i)$

e.g., with $U_1 = true$, $O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

Forward and backward messages as column vectors:

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

$$
b_{k+1:t} = TO_{k+1} b_{k+2:t}
$$

Forward-backward algorithm needs time $O(S^2 t)$ and space $O(St)$
Kalman filters

Modelling systems described by a set of continuous variables, e.g., tracking a bird flying—\( \mathbf{X}_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z} \).

Airplanes, robots, ecosystems, economies, chemical plants, planets, \ldots

---

Gaussian prior, linear Gaussian transition model and sensor model
Updating Gaussian distributions

Prediction step: if $P(X_t|e_{1:t})$ is Gaussian, then prediction

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

is Gaussian. If $P(X_{t+1}|e_{1:t})$ is Gaussian, then the updated distribution

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

is Gaussian

Hence $P(X_t|e_{1:t})$ is multivariate Gaussian $N(\mu_t, \Sigma_t)$ for all $t$
Updating Gaussian distributions

Prediction step: if $P(X_t|e_{1:t})$ is Gaussian, then prediction

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

is Gaussian. If $P(X_{t+1}|e_{1:t})$ is Gaussian, then the updated distribution

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

is Gaussian

Hence $P(X_t|e_{1:t})$ is multivariate Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$ for all $t$

General (nonlinear, non-Gaussian) process: description of posterior grows unboundedly as $t \to \infty$
2-D tracking example: filtering
Exercise
Uncertainty over Time
Speech Recognition
Learning

2-D tracking example: smoothing

2D smoothing

- true
- * observed
- - - - - - smoothed

X
Y

true
observed
smoothed

Exercise
Uncertainty over Time
Speech Recognition
Learning
Where it breaks

Cannot be applied if the transition model is nonlinear

Extended Kalman Filter models transition as **locally linear** around $x_t = \mu_t$

Fails if systems is locally unsmooth
Dynamic Bayesian networks

$X_t$, $E_t$ contain arbitrarily many variables in a replicated Bayes net

$$
\begin{array}{c|c|c}
R_0 & P(R_1) \\
\hline
0.7 & 0.3 \\
\end{array}
$$

$$
\begin{array}{c|c|c}
R_1 & P(U_1) \\
\hline
0.7 & 0.9 \\
0.2 & \\
\end{array}
$$
Every HMM is a single-variable DBN; every discrete DBN is an HMM

Sparse dependencies $\Rightarrow$ exponentially fewer parameters;
e.g., 20 state variables, three parents each
DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$
DBNs vs Kalman filters

Every Kalman filter model is a DBN, but few DBNs are KFs; real world requires non-Gaussian posteriors
Summary

- Temporal models use state and sensor variables replicated over time

- Markov assumptions and stationarity assumption, so we need
  - transition model \( P(X_t \mid X_{t-1}) \)
  - sensor model \( P(E_t \mid X_t) \)

- Tasks are filtering, prediction, smoothing, most likely sequence; all done recursively with constant cost per time step

- Hidden Markov models have a single discrete state variable; used for speech recognition

- Kalman filters allow \( n \) state variables, linear Gaussian, \( O(n^3) \) update

- Dynamic Bayes nets subsume HMMs, Kalman filters; exact update intractable
Outline

1. Exercise

2. Uncertainty over Time

3. Speech Recognition

4. Learning
Outline

♦ Speech as probabilistic inference
♦ Speech sounds
♦ Word pronunciation
♦ Word sequences
Speech as probabilistic inference

- Speech signals are noisy, variable, ambiguous
Speech as probabilistic inference

- Speech signals are noisy, variable, ambiguous

- What is the **most likely** word sequence, given the speech signal?
  I.e., choose *Words* to maximize \( P(\text{Words}|\text{signal}) \)
Speech as probabilistic inference

- Speech signals are noisy, variable, ambiguous
- What is the **most likely** word sequence, given the speech signal? 
  I.e., choose *Words* to maximize \( P(\text{Words}|\text{signal}) \)
- Use Bayes’ rule:
  \[
P(\text{Words}|\text{signal}) = \alpha P(\text{signal}|\text{Words})P(\text{Words})
\]
  I.e., decomposes into *acoustic model* + *language model*
Speech as probabilistic inference

- Speech signals are noisy, variable, ambiguous

- What is the **most likely** word sequence, given the speech signal? I.e., choose *Words* to maximize $P(\text{Words}|\text{signal})$

- Use Bayes’ rule:

  $$P(\text{Words}|\text{signal}) = \alpha P(\text{signal}|\text{Words})P(\text{Words})$$

  I.e., decomposes into **acoustic model** + **language model**

- *Words* are the hidden state sequence, *signal* is the observation sequence
All human speech is composed from 40-50 phones, determined by the configuration of articulators (lips, teeth, tongue, vocal cords, air flow). Form an intermediate level of hidden states between words and signal:

⇒ acoustic model = pronunciation model + phone model

ARPAbet designed for American English

| [ey] | bet  | [d] | debt | [s] | set |
| [ow] | boat | [hv] | high | [dh] | that |
| [er] | Bert | [l] | let | [w] | wet |
| [ix] | roses | [ng] | sing | [en] | button |
|       |       |     |      |     |     |

E.g., “ceiling” is [s iy l ih ng] / [s iy l ix ng] / [s iy l en]
Word pronunciation models

Each word is described as a distribution over phone sequences
Distribution represented as an HMM transition model

\[ P([towmeytow]|"tomato") = P([towmaatow]|"tomato") = 0.1 \]
\[ P([tahmeytow]|"tomato") = P([tahmaatow]|"tomato") = 0.4 \]

Structure is created manually, transition probabilities learned from data
Isolated words

- Phone models + word models fix likelihood $P(e_{1:t}|word)$ for isolated word

$$P(word|e_{1:t}) = \alpha P(e_{1:t}|word)P(word)$$

- Prior probability $P(word)$ obtained simply by counting word frequencies

$P(e_{1:t}|word)$ can be computed recursively: define

$$\ell_{1:t} = P(X_t, e_{1:t})$$

and use the recursive update

$$\ell_{1:t+1} = \text{Forward}(\ell_{1:t}, e_{t+1})$$

and then $P(e_{1:t}|word) = \sum_{x_t} \ell_{1:t}(x_t)$

- Isolated-word dictation systems with training reach 95–99% accuracy
Continuous speech

Not just a sequence of isolated-word recognition problems!
– Adjacent words highly correlated
– Sequence of most likely words ≠ most likely sequence of words
– Segmentation: there are few gaps in speech
– Cross-word coarticulation—e.g., “next thing”

Continuous speech systems manage 60–80% accuracy on a good day
Language model

Prior probability of a word sequence is given by chain rule:

\[ P(w_1 \cdots w_n) = \prod_{i=1}^{n} P(w_i|w_1 \cdots w_{i-1}) \]

**Bigram model:**

\[ P(w_i|w_1 \cdots w_{i-1}) \approx P(w_i|w_{i-1}) \]

Train by counting all word pairs in a large text corpus
More sophisticated models (trigrams, grammars, etc.) help a little bit
Combined HMM

- States of the combined language+word+phone model are labelled by the word we’re in + the phone in that word + the phone state in that phone

- Viterbi algorithm finds the most likely phone state sequence

- Does segmentation by considering all possible word sequences and boundaries

- Doesn’t always give the most likely word sequence because each word sequence is the sum over many state sequences

- Jelinek invented A* in 1969 a way to find most likely word sequence where “step cost” is $-\log P(w_i|w_{i-1})$
Outline

1. Exercise
2. Uncertainty over Time
3. Speech Recognition
4. Learning
Outline

♦ Learning agents
♦ Inductive learning
♦ Decision tree learning
♦ Measuring learning performance
Learning

Back to Turing’s article:
- child mind program
- education

Reward & Punishment

- Learning is essential for unknown environments, i.e., when designer lacks omniscience
- Learning is useful as a system construction method, i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent’s decision mechanisms to improve performance
Learning agents

Performance standard

- Critic
- Sensors
- Learning element
- Performance element
- Problem generator
- Effectors

feedback

changes

learning goals

knowledge

experiments

Environment

Agent
Learning element

Design of learning element is dictated by
- what type of performance element is used
- which functional component is to be learned
- how that functional component is represented
- what kind of feedback is available
Learning element

Design of learning element is dictated by
♦ what type of performance element is used
♦ which functional component is to be learned
♦ how that functional component is represented
♦ what kind of feedback is available

Example scenarios:

<table>
<thead>
<tr>
<th>Performance element</th>
<th>Component</th>
<th>Representation</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha–beta search</td>
<td>Eval. fn.</td>
<td>Weighted linear function</td>
<td>Win/loss</td>
</tr>
<tr>
<td>Logical agent</td>
<td>Transition model</td>
<td>Successor–state axioms</td>
<td>Outcome</td>
</tr>
<tr>
<td>Utility–based agent</td>
<td>Transition model</td>
<td>Dynamic Bayes net</td>
<td>Outcome</td>
</tr>
<tr>
<td>Simple reflex agent</td>
<td>Percept–action fn</td>
<td>Neural net</td>
<td>Correct action</td>
</tr>
</tbody>
</table>
Learning element

Design of learning element is dictated by
♦ what type of performance element is used
♦ which functional component is to be learned
♦ how that functional component is represented
♦ what kind of feedback is available

Example scenarios:

<table>
<thead>
<tr>
<th>Performance element</th>
<th>Component</th>
<th>Representation</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha–beta search</td>
<td>Eval. fn.</td>
<td>Weighted linear function</td>
<td>Win/loss</td>
</tr>
<tr>
<td>Logical agent</td>
<td>Transition model</td>
<td>Successor–state axioms</td>
<td>Outcome</td>
</tr>
<tr>
<td>Utility–based agent</td>
<td>Transition model</td>
<td>Dynamic Bayes net</td>
<td>Outcome</td>
</tr>
<tr>
<td>Simple reflex agent</td>
<td>Percept–action fn</td>
<td>Neural net</td>
<td>Correct action</td>
</tr>
</tbody>
</table>

Supervised learning: correct answers for each instance
Reinforcement learning: occasional rewards