

Course Overview

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

- ✓ 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

2

Performance of approximation algorithms

Exercise
Uncertainty over Time
Speech Recognition
Learning

- 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 δ
- 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)

3

Summary

Exercise
Uncertainty over Time
Speech Recognition
Learning

- 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

4

Outline

Wumpus World

1. Exercise

2. Uncertainty over Time

3. Speech Recognition

4. Learning

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 B OK	2,2	3,2	4,2
1,1 OK	2,1 B OK	3,1	4,1

$P_{ij} = true$ iff $[i, j]$ contains a pit

$B_{ij} = true$ iff $[i, j]$ is breezy

Include only $B_{1,1}, B_{1,2}, B_{2,1}$ in the probability model

5

6

Specifying the probability model

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

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

(Do it this way to get $P(Effect|Cause)$.)

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

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

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

for n pits.

7

Observations and query

We know the following facts:

$$b = \neg b_{1,1} \wedge b_{1,2} \wedge b_{2,1}$$

$$known = \neg p_{1,1} \wedge \neg p_{1,2} \wedge \neg p_{2,1}$$

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

Define $Unknown = P_{ij}$ s other than $P_{1,3}$ and $Known$

For inference by enumeration, we have

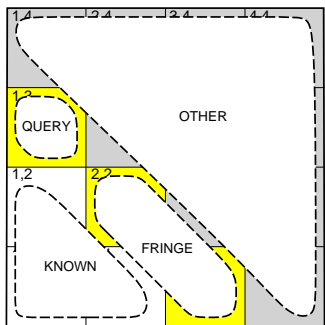
$$P(P_{1,3} | known, b) = \alpha \sum_{unknown} P(P_{1,3}, unknown, known, b)$$

Grows exponentially with number of squares!

8

Using conditional independence

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



Define $Unknown = Fringe \cup Other$

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

Manipulate query into a form where we can use this!

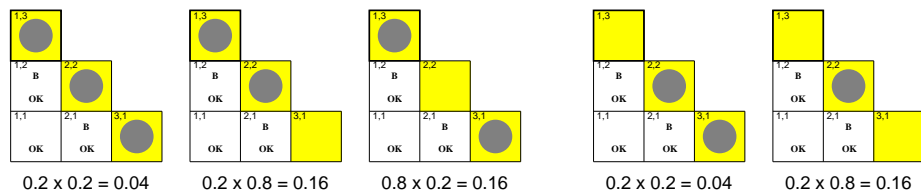
Using conditional independence contd.

$$\begin{aligned} P(P_{1,3}|known, b) &= \alpha \sum_{unknown} P(P_{1,3}, unknown, known, b) \\ &= \alpha \sum_{unknown} P(b|P_{1,3}, known, unknown)P(P_{1,3}, known, unknown) \\ &= \alpha \sum_{fringe} \sum_{other} P(b|known, P_{1,3}, fringe, other)P(P_{1,3}, known, fringe, other) \\ &= \alpha \sum_{fringe} \sum_{other} P(b|known, P_{1,3}, fringe)P(P_{1,3}, known, fringe, other) \\ &= \alpha \sum_{fringe} P(b|known, P_{1,3}, fringe) \sum_{other} P(P_{1,3}, known, fringe, other) \\ &= \alpha \sum_{fringe} P(b|known, P_{1,3}, fringe) \sum_{other} P(P_{1,3})P(known)P(fringe)P(other) \\ &= \alpha P(known)P(P_{1,3}) \sum_{fringe} P(b|known, P_{1,3}, fringe)P(fringe) \sum_{other} P(other) \\ &= \alpha' P(P_{1,3}) \sum_{fringe} P(b|known, P_{1,3}, fringe)P(fringe) \end{aligned}$$

9

10

Using conditional independence contd.



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

$$P(P_{2,2}|known, b) \approx \langle 0.86, 0.14 \rangle$$

Outline

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

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

- 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
 \mathbf{X}_t = set of unobservable state variables at time t
 e.g., *BloodSugar_t*, *StomachContents_t*, etc.
 \mathbf{E}_t = set of observable evidence variables at time t
 e.g., *MeasuredBloodSugar_t*, *PulseRate_t*, *FoodEaten_t*
- This assumes **discrete time**; step size depends on problem
- Notation: $\mathbf{X}_{a:b} = \mathbf{X}_a, \mathbf{X}_{a+1}, \dots, \mathbf{X}_{b-1}, \mathbf{X}_b$

13

14

Markov processes (Markov chains)

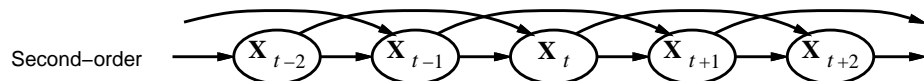
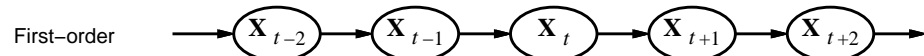
Construct a Bayes net from these variables:

- unbounded number of conditional probability table
- unbounded number of parents

Markov assumption: \mathbf{X}_t depends on **bounded** subset of $\mathbf{X}_{0:t-1}$

First-order Markov process: $P(\mathbf{X}_t | \mathbf{X}_{0:t-1}) = P(\mathbf{X}_t | \mathbf{X}_{t-1})$

Second-order Markov process: $P(\mathbf{X}_t | \mathbf{X}_{0:t-1}) = P(\mathbf{X}_t | \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$



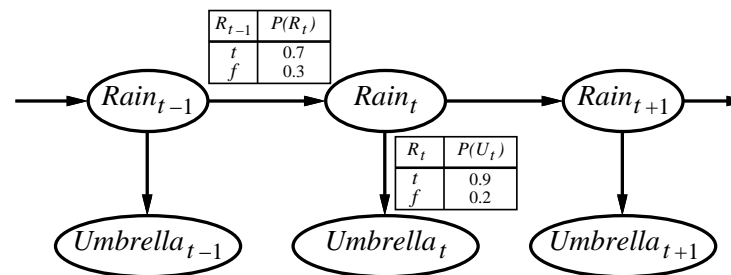
Sensor Markov assumption: $P(\mathbf{E}_t | \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = P(\mathbf{E}_t | \mathbf{X}_t)$

⇒ Stationary process:

- transition model $P(\mathbf{X}_t | \mathbf{X}_{t-1})$ and
- sensor model $P(\mathbf{E}_t | \mathbf{X}_t)$ fixed for all t

15

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 *Temp_t*, *Pressure_t*

Example: robot motion.

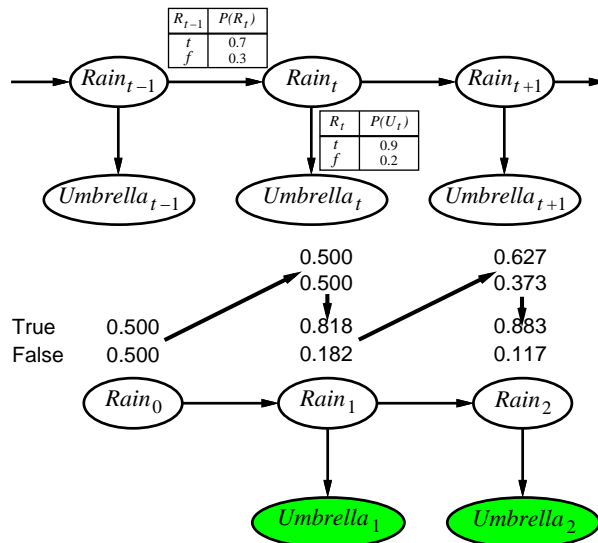
Augment position and velocity with *Battery_t*

16

Inference tasks

- Filtering:** $P(\mathbf{X}_t | \mathbf{e}_{1:t})$
belief state—input to the decision process of a rational agent
- Prediction:** $P(\mathbf{X}_{t+k} | \mathbf{e}_{1:t})$ for $k > 0$
evaluation of possible action sequences;
like filtering without the evidence
- Smoothing:** $P(\mathbf{X}_k | \mathbf{e}_{1:t})$ for $0 \leq k < t$
better estimate of past states, essential for learning
- Most likely explanation:** $\arg \max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} | \mathbf{e}_{1:t})$
speech recognition, decoding with a noisy channel

Filtering example



Filtering

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

$$P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, P(\mathbf{X}_t | \mathbf{e}_{1:t}))$$

$$\begin{aligned} P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) &= P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\ &= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}) \\ &= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}) \end{aligned}$$

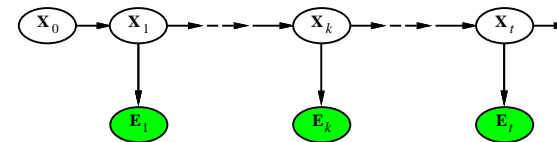
I.e., **prediction** + **estimation**. Prediction by summing out \mathbf{X}_t :

$$\begin{aligned} P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) &= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{X}_{t+1} | \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t | \mathbf{e}_{1:t}) \\ &= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{X}_{t+1} | \mathbf{x}_t) P(\mathbf{x}_t | \mathbf{e}_{1:t}) \end{aligned}$$

$\mathbf{f}_{1:t+1} = \text{Forward}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$ where $\mathbf{f}_{1:t} = P(\mathbf{X}_t | \mathbf{e}_{1:t})$

Time and space **constant** (independent of t) by keeping track of \mathbf{f}

Smoothing



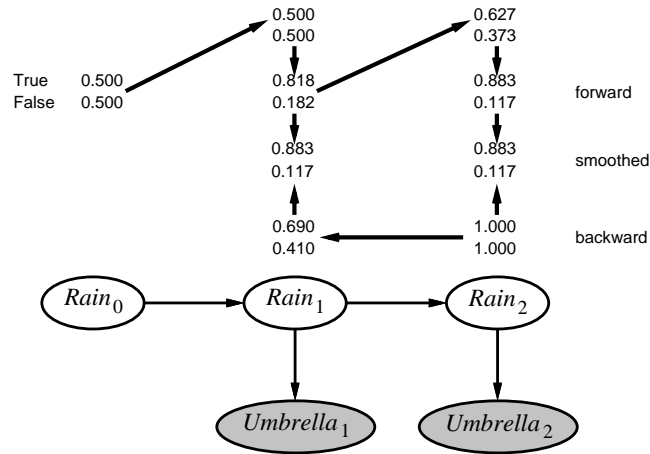
Divide evidence $\mathbf{e}_{1:t}$ into $\mathbf{e}_{1:k}$, $\mathbf{e}_{k+1:t}$:

$$\begin{aligned} P(\mathbf{X}_k | \mathbf{e}_{1:t}) &= P(\mathbf{X}_k | \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\ &= \alpha P(\mathbf{X}_k | \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} | \mathbf{X}_k, \mathbf{e}_{1:k}) \\ &= \alpha P(\mathbf{X}_k | \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} | \mathbf{X}_k) \\ &= \alpha \mathbf{f}_{1:k} \mathbf{b}_{k+1:t} \end{aligned}$$

Backward message computed by a backwards recursion:

$$\begin{aligned} P(\mathbf{e}_{k+1:t} | \mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} | \mathbf{X}_k, \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} | \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} | \mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t} | \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \end{aligned}$$

Smoothing example



If we want to smooth the whole sequence:

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

21

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

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

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

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

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

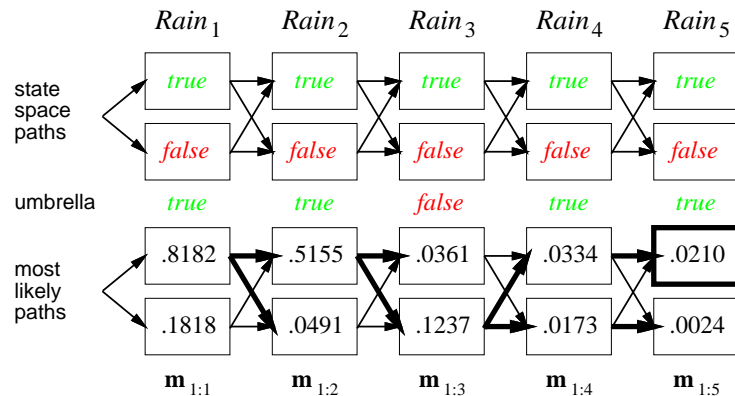
i.e., $m_{1:t}(i)$ gives the probability of the most likely path to state i .

Update has sum replaced by max, giving the Viterbi algorithm:

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

22

Viterbi example



23

Hidden Markov models

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

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

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:

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

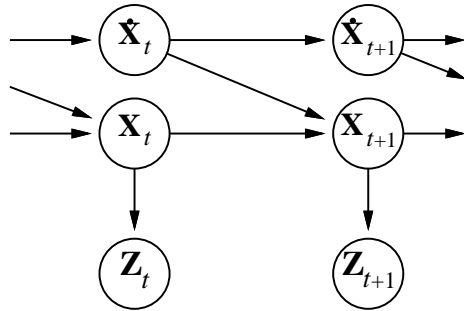
$$b_{k+1:t} = T O_{k+1} b_{k+2:t}$$

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

24

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, ...



Gaussian prior, linear Gaussian transition model and sensor model

Updating Gaussian distributions

Prediction step: if $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ is Gaussian, then prediction

$$P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}) = \int_{\mathbf{x}_t} P(\mathbf{X}_{t+1} | \mathbf{x}_t) P(\mathbf{x}_t | \mathbf{e}_{1:t}) d\mathbf{x}_t$$

is Gaussian. If $P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t})$ is Gaussian, then the updated distribution

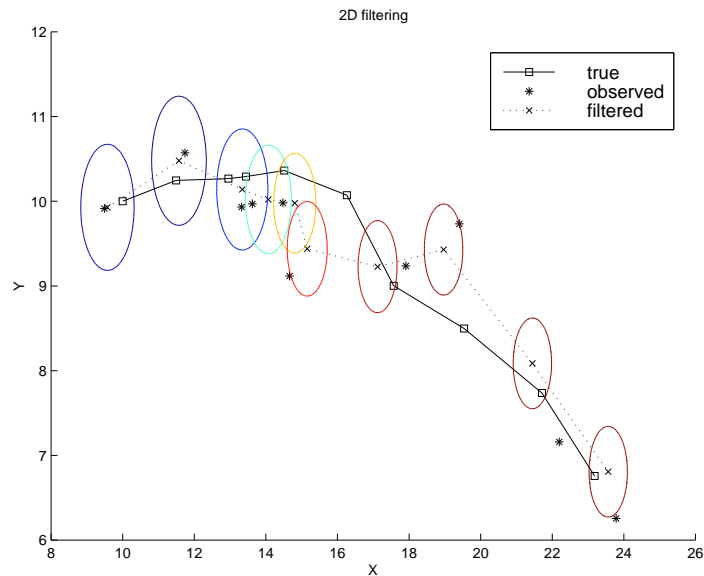
$$P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t})$$

is Gaussian

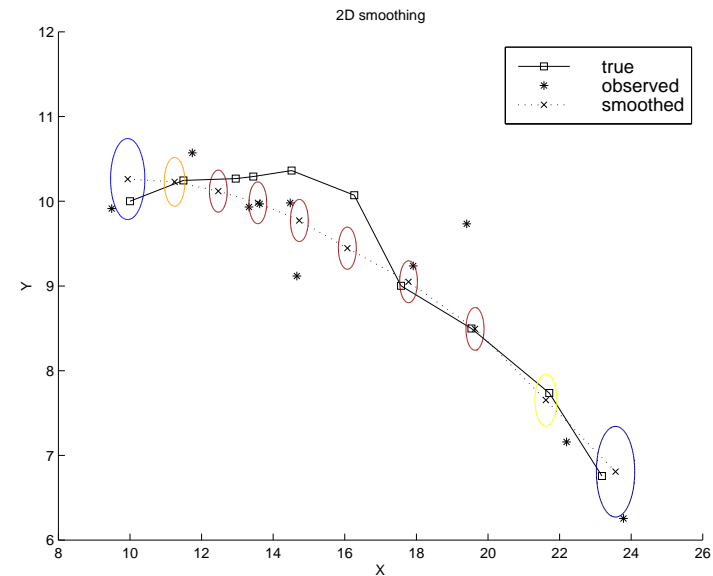
Hence $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ is multivariate Gaussian $N(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ for all t

General (nonlinear, non-Gaussian) process: description of posterior grows **unboundedly** as $t \rightarrow \infty$

2-D tracking example: filtering



2-D tracking example: smoothing

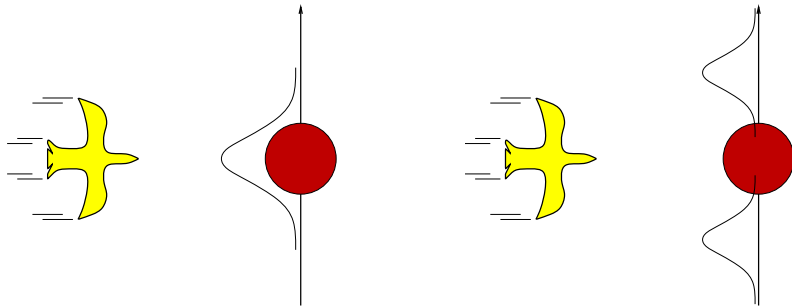


Where it breaks

Cannot be applied if the transition model is nonlinear

Extended Kalman Filter models transition as **locally linear** around $\mathbf{x}_t = \mu_t$

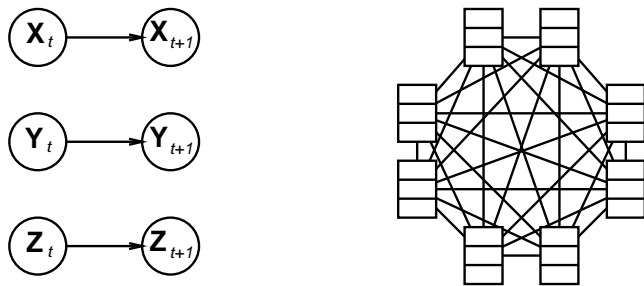
Fails if systems is locally unsmooth



29

DBNs vs. HMMs

Every HMM is a single-variable DBN; every discrete DBN is an HMM



Sparse dependencies \Rightarrow exponentially fewer parameters;

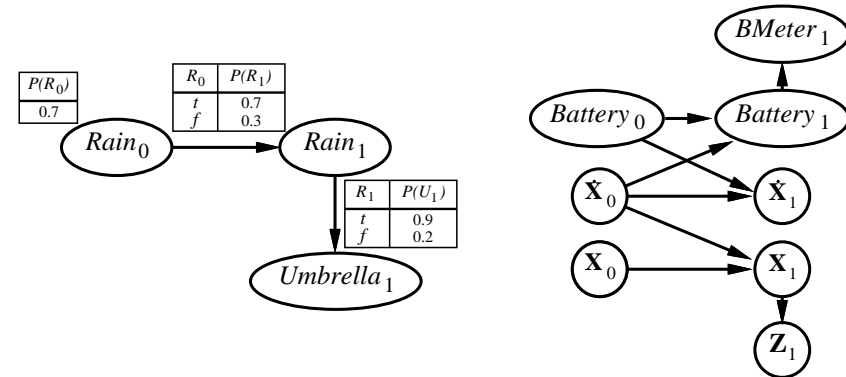
e.g., 20 state variables, three parents each

DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

31

Dynamic Bayesian networks

$\mathbf{X}_t, \mathbf{E}_t$ contain arbitrarily many variables in a replicated Bayes net



30

DBNs vs Kalman filters

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

32

Summary

- Temporal models use state and sensor variables replicated over time
- Markov assumptions and stationarity assumption, so we need
 - transition model $P(\mathbf{X}_t|\mathbf{X}_{t-1})$
 - sensor model $P(\mathbf{E}_t|\mathbf{X}_t)$
- 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

33

Outline

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

34

Outline

- ◇ Speech as probabilistic inference
- ◇ Speech sounds
- ◇ Word pronunciation
- ◇ Word sequences

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

35

36

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

[iy]	<u>beat</u>	[b]	<u>bet</u>	[p]	<u>pet</u>
[ih]	<u>bit</u>	[ch]	<u>Chet</u>	[r]	<u>rat</u>
[ey]	<u>bet</u>	[d]	<u>debt</u>	[s]	<u>set</u>
[ao]	<u>bought</u>	[hh]	<u>hat</u>	[th]	<u>thick</u>
[ow]	<u>boat</u>	[hv]	<u>high</u>	[dh]	<u>that</u>
[er]	<u>Bert</u>	[l]	<u>let</u>	[w]	<u>wet</u>
[ix]	<u>roses</u>	[ng]	<u>sing</u>	[en]	<u>button</u>
⋮	⋮	⋮	⋮	⋮	⋮

E.g., "ceiling" is [s iy | ih ng] / [s iy | ix ng] / [s iy | en]

37

Isolated words

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

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

- Prior probability $P(\text{word})$ obtained simply by counting word frequencies
 $P(e_{1:t} | \text{word})$ can be computed recursively: define

$$\ell_{1:t} = \mathbf{P}(\mathbf{X}_t, \mathbf{e}_{1:t})$$

and use the recursive update

$$\ell_{1:t+1} = \text{Forward}(\ell_{1:t}, \mathbf{e}_{t+1})$$

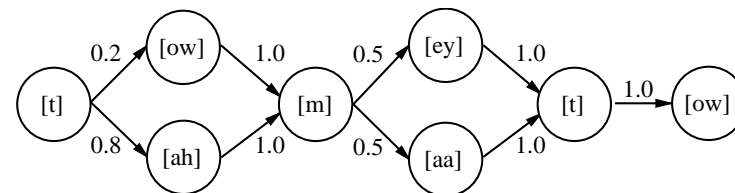
and then $P(e_{1:t} | \text{word}) = \sum_{\mathbf{x}_t} \ell_{1:t}(\mathbf{x}_t)$

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

42

Word pronunciation models

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



$$P([\text{towmeytow}] | \text{"tomato"}) = P([\text{towmaatow}] | \text{"tomato"}) = 0.1$$

$$P([\text{tahmeytow}] | \text{"tomato"}) = P([\text{tahmaatow}] | \text{"tomato"}) = 0.4$$

Structure is created manually, transition probabilities learned from data

41

Continuous speech

Not just a sequence of isolated-word recognition problems!

- Adjacent words highly correlated
- Sequence of most likely words \neq 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

43

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

44

Outline

1. Exercise

2. Uncertainty over Time

3. Speech Recognition

4. Learning

46

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 $-\log P(w_i | w_{i-1})$

45

Outline

- ◇ Learning agents
- ◇ Inductive learning
- ◇ Decision tree learning
- ◇ Measuring learning performance

47

Learning

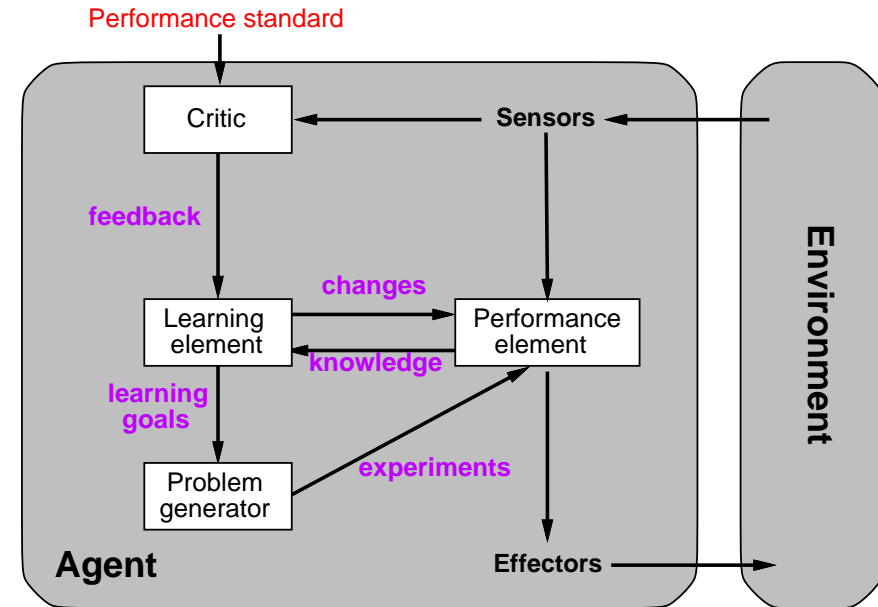
Back to Turing's article:

- 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



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:

Performance element	Component	Representation	Feedback
Alpha-beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor-state axioms	Outcome
Utility-based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept-action fn	Neural net	Correct action

Supervised learning: correct answers for each instance

Reinforcement learning: occasional rewards