Inf2D-Reasoning and Agents
Spring 2017

Lecture (2)8:– Dynamic Bayesian Networks

Peggy Seriès, pseries@inf.ed.ac.uk

Based on previous slides by A. Lascarides
Where are we?

- Last time . . .
- Inference in temporal models: smoothing and most likely sequence
- Discussed general model (forward-backward algorithm for smoothing, Viterbi algorithm for most likely sequence)
- Specific instances: HMMs
- today … bit more about HMM, Kalman filters and mostly Dynamic Bayesian Networks
Smoothing

- Smoothing is computation of distribution of past states given current evidence, i.e. $P(X_k | e_{1:t})$, $1 \leq k < t$

- Easiest to view as 2-step process (up to $k$, then $k + 1$ to $t$)

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

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

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

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

(Bayes’ rule)

(conditional independence)

- Here “backward” message is $b_{k+1:t} = P(e_{k+1:t} | X_k)$ analogous to “forward” message
Forward message (Filtering)

- Formula for forward message:

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

- The forward message is computed from \( t=1 \) to \( k \) (forward!)
Backward Message

- **Formula for backward message:**

  \[
  P(e_{k+1:t} | X_k) = \sum_{x_{k+1}} P(e_{k+1} | x_{k+1}) P(e_{k+2:t} | x_{k+1}) P(x_{k+1} | X_k)
  \]

  - Define \( b_{k+1:t} = \text{Backward}(b_{k+2:t}, e_{k+1:t}) \)
  - The backward message is computed from \( t \) to \( k+1 \) (backwards!)
  - The backward phase has to be initialised with \( b_{t+1:t} = P(e_{t+1:t} | X_t) = 1 \) (a vector of 1s) because probability of observing empty sequence is 1
Hidden Markov Models

- Hidden Markov Models (HMMs): temporal probabilistic model in which state of the process is described by a single variable
- Like our umbrella example (single variable Rain_t)
- More than one variable can be accommodated, but only by combining them into a single “mega-variable”
- the restricted structure of HMMs allows for a very simple and elegant matrix implementation of basic algorithms
Hidden Markov Models

- $X_t$ is a single, discrete variable (usually $E_t$ is too)
- Domain of $X_t$ is \{1, \ldots, S\} (S possible states)
- The transition model $T = P(X_t|X_{t-1})$ becomes a SxS Transition matrix, $T_{ij} = P(X_t = j|X_{t-1} = i)$
- For example the Transition matrix for the umbrella world is
  
  \[
  T = P(X_t | X_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}
  \]

- The sensor model can also expressed as a matrix. In this case, the value of $E_t$ is known at time $t$ (call it $e_t$), so we need only specify, for each state $i$, $P(e_t|X_t=i)$.
- For mathematical convenience we place these values into an SxS diagonal matrix, $O_t$, whose $i_{th}$ diagonal entry is $P(e_t|X_t=i)$ and whose other entries are 0.
Hidden Markov Models

• For example, if on day 1 we have $U_1=true$ and on day 3 we have $U_3=false$

\[
O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}; \quad O_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}
\]

• **Smoothing:** Now if we use column vectors to represent the forward and the backward message, all the computations become simple matrix-vector operations.

• The forward message can be written:

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

• The backward message can be written:

\[
b_{k+1:t} = TO_{k+1}b_{k+2:t}
\]
- The product between matrices $\mathbf{MN}$ is defined only if the second matrix has the same number of rows as the first has columns:
if $M$ is of size $a \times b$, then $N$ must be of size $b \times c$, and resulting matrix is of size $a \times c$. If the matrices are of appropriate size, then:
$\mathbf{MN} = \sum_j m_{ij}n_{jk}$
In the example:
$M + N = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 3 & 4 \\ 5 & 2 \end{pmatrix} = \begin{pmatrix} 1 \times 3 + 2 \times 5 & 1 \times 4 + 2 \times 2 \\ 2 \times 3 + 3 \times 5 & 2 \times 4 + 3 \times 2 \end{pmatrix} = \begin{pmatrix} 13 & 8 \\ 21 & 14 \end{pmatrix}$

- The transpose of a matrix $\mathbf{M}$ is written $\mathbf{M}^T$ and is formed turned rows into columns. For example:
$\mathbf{N}^T = \begin{pmatrix} 3 & 4 \\ 5 & 2 \end{pmatrix}^T = \begin{pmatrix} 3 & 5 \\ 4 & 2 \end{pmatrix}$
Hidden Markov Models

- The time complexity of the forward–backward algorithm applied to a sequence of length t is $O(S^2t)$, because each step requires multiplying an $S$-element vector by an $S \times S$ matrix.

- The space requirement is $O(St)$, because the forward pass stores $t$ vectors of size $S$.

- Besides providing an elegant description of the filtering and smoothing algorithms for HMMs, the matrix formulation reveals opportunities for improved algorithms.
a note about Kalman Filters

Modelling systems described by a set of continuous variables, e.g., tracking a bird flying—\( X_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z} \).
Airplanes, robots, ecosystems, economies, chemical plants, planets, …

Gaussian prior, linear Gaussian transition model and sensor model
Constructing DBNs

• A DBN is a BN describing a **temporal probability model** that can have any number of state variables $X_t$ and evidence variables $E_t$

• HMMs are DBNs with a **single state and a single evidence variable**

• But recall that one can combine a set of discrete (evidence or state) variables into a single variable (whose values are tuples).

• So every discrete-variable DBN can be described as a HMM.

• So why bother with DBNs?

• Because decomposing a complex system into constituent variables, as a DBN does, ameliorates **sparseness** in the temporal probability model.
Constructing DBNs

• We have to specify prior distribution of state variables $P(X_0)$,
• transition model $P(X_{t+1}|X_t)$, and sensor model $P(E_t|X_t)$
• Also, we have to fix topology of nodes
• Stationarity assumption
• most convenient to specify topology for first slice
• Umbrella world example:
The art of representing complex problems: An example

• Consider a battery-driven robot moving in the $X \times Y$ plane

• Let $\mathbf{X}_t = (X_t, Y_t)$ and $\dot{\mathbf{X}}_t = (X_t', Y_t')$ state variables for position and velocity, and $Z_t$ measurements of position (e.g. GPS)

• Add $\text{Battery}_t$ for battery charge level and $\text{BMeter}_t$ for the measurement of it

• We obtain the following basic model:
Modeling Failure

• Assume $Battery_t$ and $BMeter_t$ take on discrete values (e.g. integer between 0 and 5)
  These variables should be identical (CPT=identity matrix) unless error creeps in

• One way to model error is through Gaussian error model, i.e. a small Gaussian error is added to the meter reading — We can approximate this also for the discrete case through an appropriate distribution

• Problem: This model will lead to very wrong estimation if sensor failure rather than inaccurate measurements …
Transient failure

• **Transient failure**: sensor occasionally sends inaccurate data,

• Robot example: after 20 consecutive readings of 5 suddenly $\text{B} \text{M} \text{e} \text{t} \text{e} \text{r}_{21} = 0$

• How should this be interpreted?

• In Gaussian error model, belief about $\text{B} \text{a} \text{t} \text{t} \text{e} \text{r}_{21}$ depends on:
  - Sensor model: $P(\text{B} \text{M} \text{e} \text{t} \text{e} \text{r}_{21} = 0|\text{B} \text{a} \text{t} \text{t} \text{e} \text{r}_{21})$ and
  - Prediction model: $P(\text{B} \text{a} \text{t} \text{t} \text{e} \text{r}_{21}|\text{B} \text{M} \text{e} \text{t} \text{e} \text{r}_{1:20})$

• If probability of large sensor error is smaller than sudden transition to 0, then with high probability battery is considered empty

• A measurement of 0 at $t = 22$ will make this (almost) certain

• After a reading of 5 at $t = 23$ the probability of full battery will go back to high level

• which of course is impossible … robot made completely wrong judgement . . .
Transient failure model

- Curves for prediction depending on whether $BMeter_t$ is only 0 for $t = 22/23$ or whether it stays 0 indefinitely
Transient failure model

• To handle failure properly, sensor model must include possibility of failure

• Simplest failure model: assign small probability to incorrect values, e.g. \[ P(B\text{Meter}_t = 0 | \text{Battery}_t = 5) = 0.03 \]

• When faced with 0 reading, provided that predicted probability of empty battery is much less than 0.03, best explanation is failure

• This model is much less susceptible to failure, because an explanation is available

• However, it cannot cope with persistent failure either
Transient failure model

- Handles transient failure
- **Problem:** In case of permanent failure the robot will (wrongly) believe the battery is empty
Persistent failure model

- **Persistent failure models** describe how sensor behaves under normal conditions and after failure.
- Add **additional variable** `BMBroken`, and CPT to next `BMBroken` state has a very small probability if not broken, but 1.0 if broken before.
- When `BMBroken` is true, `BMeter` will be 0 regardless of `Battery`:
Persistent failure model

- In case of persistent failure, robot assumes discharge of battery at “normal” rate
Exact Inference in DBNs

- Since DBNs are BNs, we already have inference algorithms like variable elimination.
- Essentially DBN equivalent to infinite “unfolded” BN, but slices beyond required inference period are irrelevant.
- Unrolling: reproducing basic time slice to accommodate observation sequence.
Exact Inference in DBNs

• Exact inference in DBNs is intractable, and this is a major problem.
• There are approximate inference methods that work well in practice, likelihood weighting, MCMC, particle filtering.
• This issue is currently a hot topic in AI. ..
Summary

• Account of time and uncertainty complete
• HMMs and matrix operations.
• Kalman Filters
• DBNs as general case
• the art of representing complex processes, robot example: choosing a transition model and sensor models.
• Quite intractable / exact inference, but powerful
• Next time: Decision Making under Uncertainty