Informatics 2D: Reasoning and Agents

Alex Lascarides

School of Informatics
The University of Edinburgh

Lecture 28: Dynamic Bayesian Networks

Where are we?

Last time ...

- Inference in temporal models
- Discussed general model (forward-backward, Viterbi etc.)
- Specific instances: HMMs
- But what is the connection to Bayesian networks?

Today ...

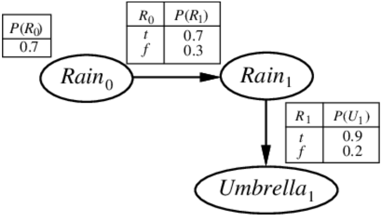
- **Dynamic Bayesian Networks**

Dynamic Bayesian Networks

- We've already seen an example of a DBN—Umbrella World
- A DBN is a BN describing a temporal probability model that can have *any number* of state variables $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$
- 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

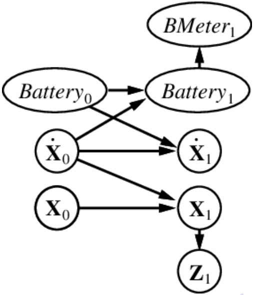
Constructing DBNs

- We have to specify prior distribution of state variables $\mathbf{P}(\mathbf{X}_0)$, transition model $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)$, and sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$
- Also, we have to fix topology of nodes
- Stationarity assumption
  most convenient to specify topology for first slice
- Umbrella world example:



An example

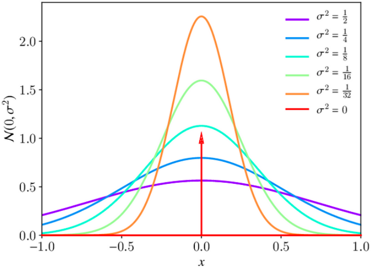
- Consider a battery-driven robot moving in the $X \times Y$ plane
- Let $\mathrm{X}_t = (X_t, Y_t)$ and $\dot{\mathrm{X}}_t = (\dot{X}_t, \dot{Y}_t)$ state variables for position and velocity, and Z_t measurements of position (e.g. GPS)
- Add $Battery_t$ for battery charge level and $BMeter_t$ for the measurement of it
- We obtain the following basic model:



Modelling failure

- Assume $Battery_t$ and $BMeter_t$ take on discrete values (e.g. integer between 0 and 5)
- These variables should be identically distributed (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

Gaussian Error model



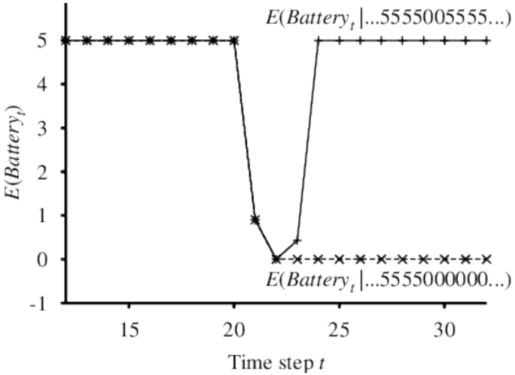
- The bigger the σ, the more expected errors there are.
- We can approximate this also for the discrete case through an appropriate distribution.
- Error 'on one side' when meter reading is 5 or 0.
- But problem can be much worse:
  sensor failure rather than inaccurate measurements.

Transient failure

- **Transient failure**: sensor occasionally sends inaccurate data
- Robot example: after 20 consecutive readings of 5 suddenly $BMeter_{21} = 0$
- In Gaussian error model belief about $Battery_{21}$ depends on:
  - Sensor model: $P(BMeter_{21} = 0|Battery_{21})$ and
  - Prediction model: $P(Battery_{21}|BMeter_{1:20})$
- If probability of large sensor error is smaller than sudden transition to 0, then with high probability battery is considered empty
- A measurement of zero 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
- But robot made completely wrong judgement ...

Transient failure

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

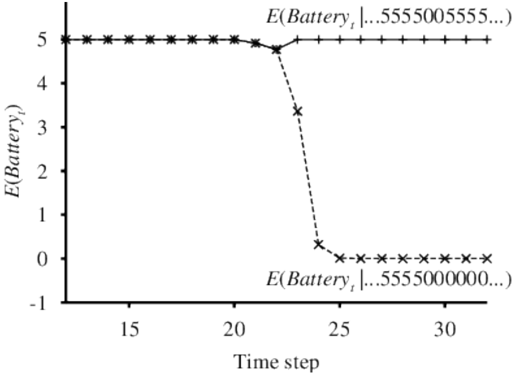


How to fix inferences about errors: first attempt

- To handle failure properly, sensor model must include possibility of (radical) meter failure
- Simplest failure model: assign larger probability to 0 reading than a Gaussian error model would; e.g. $P(BMeter_t = 0 | Battery_t = 5) = 0.03$
- When faced with 0 reading, provided that predicted probability of empty battery is much less than 0.03, best explanation is meter failure
- This model is much less susceptible to poor inference, because an explanation is available
- However, it cannot cope with **persistent** failure either

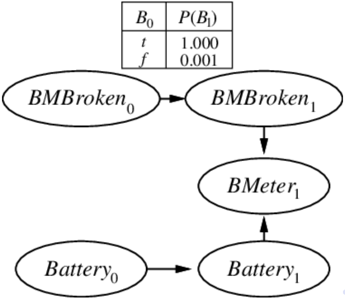
Transient failure model

- Handling transient failure with explicit error models
- In case of permanent failure the robot will (wrongly) believe the battery is empty



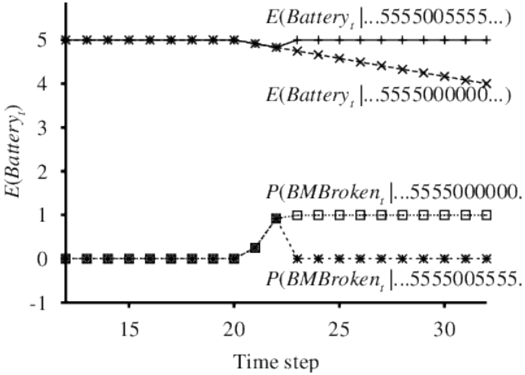
How to fix inferences about errors: second attempt

- **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 (**persistence arc**)
- When *BMBroken* is true, *BMeter* will be 0 regardless of *Battery*:



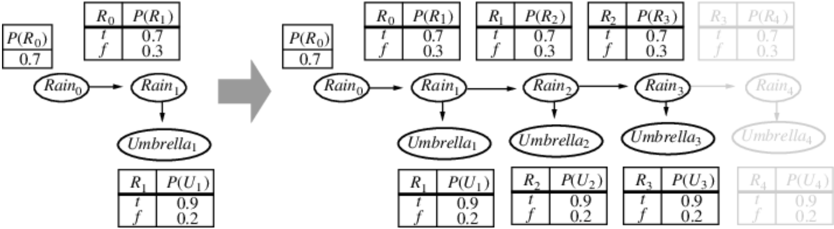
Persistent failure

- In case of temporary blip probability of broken sensor rises quickly but goes back if 5 is observed
- 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.
- This issue is currently a hot topic in AI. . .

Summary

- Account of time and uncertainty complete
- Looked at general Markovian models
- HMMs
- DBNs as general case
- Quite intractable, but powerful
- Next time: **Decision Making under Uncertainty**