Knowledge Engineering with Bayesian Networks

Prof. Paulo André Castro
www.comp.ita.br/~pauloac
pauloac@ita.br
Sala 110, IEC-ITA
Knowledge Engineering with Bayesian Networks

- When constructing a Bayesian network, the major modeling issues that arise are:
  - What are the variables? What are their values/states?
  - What is the graph structure?
  - What are the parameters (probabilities)?

- When building decision nets, the additional questions are:
  - What are the available actions/decisions, and what impact do they have?
KEBN life cycle model

- A simple view of the software engineering process construes it as having a lifecycle: the software is born (design), matures (coding), has a lengthy middle age (maintenance) and dies of old age (obsolescence).

- One effort at construing KEBN in such a lifecycle model (also called a “waterfall” model) is shown next.
KEBN “waterfall” life cycle model

1) Building the BN
   i) Structure
   ii) Parameters
   iii) Preferences

2) Validation
   Sensitivity Analysis
   Accuracy Testing

3) Field Testing
   Alpha/Beta Testing
   Acceptance Testing

4) Industrial Use
   Collection of Statistics

5) Refinement
   Updating Procedures
   Regression Testing
Phases

- In the **building phase**, the major network components of structure, parameters and, if a decision network, utilities (preferences) must be determined through elicitation from experts, or learned with data mining methods, or some combination of the two.
Phases - Validation

- **Validation** aims to establish that the network is right for the job, answering such questions as:
  - Is the predictive accuracy for a query node satisfactory?
  - Does it respect any known temporal order of the variables?
  - Does it incorporate known causal structure?

- **Sensitivity analysis** looks at how sensitive the network is to changes in input and parameter values, which can be useful both for validating that the network is correct and for understanding how best to use the network in the field.
Phases – Field Testing

- **Field testing** first puts the BN into actual use, allowing its usability and performance to be gauged.
  - **Alpha testing** refers to an intermediate test of the system by inhouse people who were not directly involved in developing it; for example, by other inhouse BN experts.
  - **Beta testing** is testing in an actual application by a “friendly” end-user, who is prepared to accept hitting bugs in early release software.
  - **Acceptance testing** is surely required: it means getting the end users to accept that the BN software meets their criteria for use.
Phases – Industrial Use and Refinement

- **Industrial use** sees the BN in regular use in the field and requires that procedures be put in place for this continued use.
  - It is a good idea to collect statistics on the performance of the BN and statistics monitoring the application domain, in order to further validate and refine the network.

- **Refinement** requires some kind of change management regime to deal with requests for updating or fixing bugs. **Regression testing** verifies that any changes do not cause a degradation in prior performance.
Iterative approach for KEBN

- An iterative and incremental approach for KEBN seems to be a better approach for us.

- The software should grow by stages (prototypes) from childhood to adulthood, but at any given stage it is a self-sufficient, if limited, organism.

- Prototypes are functional implementations of software: they accept real input, such as the final system can be expected to deal with, and produce output of the type end-users will expect to find in the final system.
A spiral model for KEBN (Korb, Nicholson, 2011)
Iterative lifecycle model for KEBN (Boneh, 2010)
# Common mistakes

<table>
<thead>
<tr>
<th>KEBN aspect</th>
<th>Mistake</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The Process</strong></td>
<td>Parameterizing before evaluating structure</td>
</tr>
<tr>
<td></td>
<td>Trying to build the full model all at once</td>
</tr>
<tr>
<td><strong>The Problem</strong></td>
<td>Not understanding the problem context</td>
</tr>
<tr>
<td></td>
<td>Complexity without value</td>
</tr>
<tr>
<td><strong>Structure - Nodes</strong></td>
<td>Getting the node values wrong</td>
</tr>
<tr>
<td></td>
<td>Node values aren’t exhaustive</td>
</tr>
<tr>
<td></td>
<td>Node values aren’t mutually exclusive</td>
</tr>
<tr>
<td></td>
<td>Incorrect modeling of mutually exclusive outcomes</td>
</tr>
<tr>
<td></td>
<td>Trying to model fuzzy categories</td>
</tr>
<tr>
<td></td>
<td>Confusing state and probability</td>
</tr>
<tr>
<td></td>
<td>Confusion about what the node represents</td>
</tr>
<tr>
<td><strong>Structure - Arcs</strong></td>
<td>Getting the arc directions wrong</td>
</tr>
<tr>
<td></td>
<td>(a) Modeling reasoning rather than causation</td>
</tr>
<tr>
<td></td>
<td>(b) Inverting cause and effect</td>
</tr>
<tr>
<td></td>
<td>(c) Missing variables</td>
</tr>
<tr>
<td></td>
<td>Too many parents</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>Experts’ estimates of probabilities are biased</td>
</tr>
<tr>
<td></td>
<td>(a) Overconfidence</td>
</tr>
<tr>
<td></td>
<td>(b) Anchoring</td>
</tr>
<tr>
<td></td>
<td>(c) Availability</td>
</tr>
<tr>
<td></td>
<td>Inconsistent “filling in” of large CPTs</td>
</tr>
<tr>
<td></td>
<td>Incoherent probabilities (not summing to 1)</td>
</tr>
<tr>
<td></td>
<td>Being dead certain</td>
</tr>
</tbody>
</table>
Stage 1: Bayesian Network Structure

- Common Modeling Mistake: Not understanding the problem context

- It is crucial for the knowledge engineer to gain a clear understanding of the problem context. Ideally, this should be available in some form of project description. The knowledge engineer should ask questions like:

  Q: “What do you want to reason about?”
  Q: “What don’t you know?”
  Q: “What information do you have?”
  Q: “What do you know?”
Stage 1: Bayesian Network Structure

- **Complexity without value**
  - A very common impulse, when something is known about the problem, is to want to put it in the model.

  - But it may add complexity to the model without adding any value (and in fact often reduces value)

- Instead, the knowledge engineer must focus on the question:

  **Q:** “Which of the known variables are most relevant to the problem?”
Another commons mistakes in structures

- Getting the node values wrong
- Node values aren’t exhaustive
- Node values aren’t mutually exclusive
- Incorrect modeling of mutually exclusive outcomes
  - Creation of separate nodes for different states of the same variable. For example, create both a FineWeather variable and a WetWeather variable (both Boolean). They are mutually exclusive!
- Trying to model fuzzy categories
- Confusing state and probability
- Confusion about what the node represents
Stage 1: Bayesian Network Structure

- Other common mistake: Getting the arc directions wrong
  - (a) Modeling reasoning rather than causation
  - (b) Inverting cause and effect
  - (c) Missing variables

- Too many parents
  - It is usually worse to have many parents than more parents. Modeling new nodes may help…
Reducing parents by intermediate nodes
While it is possible to build BNs with continuous variables without discretization, the simplest approach is to discretize them.

Indeed, many of the current BN software tools available require this. They provide a choice between doing the discretization for you crudely, into even-sized chunks, or allowing the knowledge engineer more control over the process.
Stage 2: Probability Parameters

- Experts’ estimates of probabilities are biased, including
  - Overconfidence: the tendency to attribute higher than justifiable probabilities to events that have a probability sufficiently greater than 0.5.
  - Anchoring: the tendency for subsequent estimates to be biased by an initial estimate (Kahneman and Tversky, 1973)
  - Availability: Assessing an event as more probable than is justifiable, because it is easily remembered or more salient (Kahneman and Tversky, 1973)
Stage 2: Probability Parameters

- Inconsistent “filling in” of large CPTs
  - For example, the expert uses 0.99 for “almost certain” in one part of the CPT, and 0.999 in another. Or it may be inconsistency across the CPT; for example, using different distributions for combinations of parents that in fact are very similar.

- Incoherent probabilities (not summing to 1)

- Being dead certain
Stage 3: Decision Structure

- First, we must model what decisions can be made, through the addition of one or more decision nodes.
- If the decision task is to choose only a single decision at any one time from a set of possible actions, only one decision node is required.

- Combinations of actions can be modeled within the one node, for example, by explicitly adding a sequence of actions (ex. “surgery-medication”).
  - This modeling solution avoids the complexity of multiple decision nodes, but has the disadvantage that the overlap between different actions.

- Alternatively, you can use precedence links or Dynamic BN as you have seen, but you will have to deal with additional complexity!
Stage 4: Utilities (Preferences)

- The next KE task for decision making is to model the utility of outcomes.

- The first stage is to decide what the unit of measure ("utile") will mean. Remember that money is not equal to utility (but it is related!)

- Remember the process to evaluate utilities of situations through lotteries as we have seen!