Probabilistic Reasoning

Unit # 7

BIC Score

- The Bayesian Information Criterion (BIC) score is defined as follows

$$BIC\,(\mathbb{G} : \mathsf{D}) = \ln\left(P(\mathsf{D}|\hat{\mathsf{P}}, \mathbb{G})\right) - \frac{d}{2}\ln m$$

- where p is the number of data items, g is the dimension of the DAG model, and ˆP is the set of maximum likelihood values of the parameters. The dimension is the number of parameters in the model.

BIC Score (Cont'd)

- The BIC score is intuitively appealing because it contains
  - a term that shows how well the model predicts the data when the parameter set is equal to its ML value, and
  - a term that punishes for model complexity.
- Another nice feature of the BIC is that it does not depend on the prior distribution of the parameters, which means there is no need to assess one.

Consistent Scoring Criterion

- A consistent scoring criterion for DAG models has the following two properties:
  - As the size of the data set approaches infinity, the probability approaches one that a DAG that includes P will score higher than a DAG that does not include P.
  - As the size of the data set approaches infinity, the probability approaches one that a smaller DAG that includes P will score higher than a larger DAG that includes P.

K2 Algorithm

Problem: Find a DAG that approximates maximizing $score(\mathbb{G} : \mathsf{D})$.

Inputs: A set V of n random variables; an upper bound u on the number of parents a node may have; data D.

Outputs: n sets of parent nodes PA_i, where $1 \leq i \leq n$, in a DAG that approximates maximizing $score(\mathbb{G} : \mathsf{D})$.

```
void K2  (set_of_variables V, int u,
          data D, for 1 ≤ i ≤ n parent_set& PA_i)
{
  for (i = 1; i <= n; i++) {      // n is the number of variables.
    PA_i^G = ∅;
    P_old = score(X_i, PA_i^G : D);
    findmore = true;
```


K2 Algorithm (Cont'd)

```
    while (findmore && |PA_i^G| < u) {
      Z = node in Pred(X_i) − PA_i that maximizes
            score(X_i, PA_i^G ∪ {Z} : D);
      P_new = score(X_i, PA_i^G ∪ {Z} : D) ;
      if (P_new > P_old) {
        P_old = P_new;
        PA_i^G = PA_i^G ∪ {Z};
      }
      else
        findmore = false;
    }
  }
}
```

Finding Node Ordering

- You might wonder where we could obtain the ordering required by K2.
- Such an ordering could possibly be obtained from domain knowledge such as a time ordering of the variables.
- For example, we might know that in patients, smoking precedes bronchitis and lung cancer and that each of these conditions precedes fatigue and a positive chest X-ray.

Algorithm without a Prior Ordering

- The following greedy search algorithm that does not require a time ordering.
- The search space is again the set of all DAGs containing the n variables.
- Following operations are allowed.
  1. If two nodes are not adjacent, add an edge between them in either direction.
  2. If two nodes are adjacent, remove the edge between them.
  3. If two nodes are adjacent, reverse the edge between them.

Algorithm without a Prior Ordering (Cont'd)

Problem: Find a DAG that approximates maximizing $score(\mathbb{G} : \mathsf{D})$.

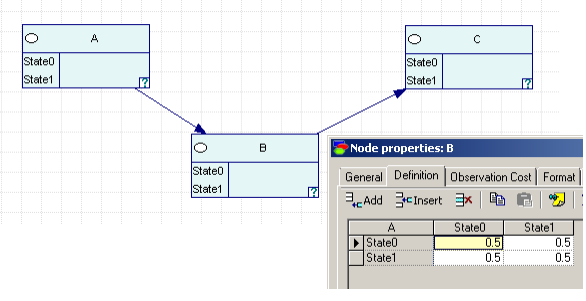
Inputs: A set V of n random variables; data D.

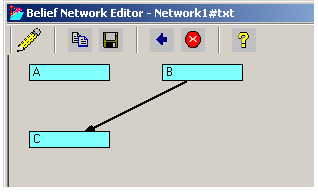
Outputs: A set of edges E in a DAG that approximates maximizing $score(\mathbb{G} : \mathsf{D})$.

```
void DAG_search (set_of_variables V, data D,
                 set_of_edges& E)
{
  E = ∅; G = (V, E);
  do
    if (any DAG in the neighborhood of our current DAG
    increases score(G : D))
      modify E according to the one that increases score(G : D) the most;
  while (some operation increases score(G : D));
}
```

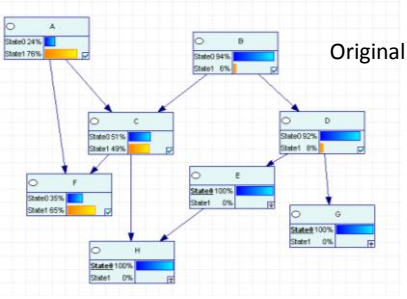

Structure Learning Example I

- Not a faithful DAG as due to the given conditional distribution A and B are independent.
- We generated 1000 records from the given network and then learn it using BN PowerConstructor.
- The network in the diagram below is learned by BN PowerConstructor.

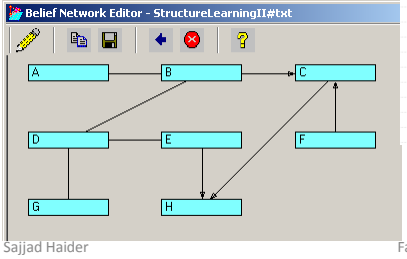




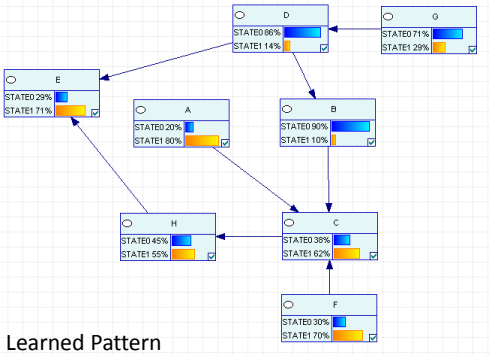
Structure Learning Example II



Original Model



Learned Pattern by BN PowerConstructor



Learned Model

Bayesian Information Criterion (BIC)

- The Bayesian information criterion (BIC) score is as follows:

$$BIC\left(\mathbb{G}:\mathrm{D}\right)=\ln\left(P(\mathrm{D}|\hat{\mathrm{P}},\mathbb{G})\right)-\frac{d}{2}\ln m,$$

- where m is the number of data items and d is the dimension of the DAG model.
- The dimension is the number of parameters in the model.

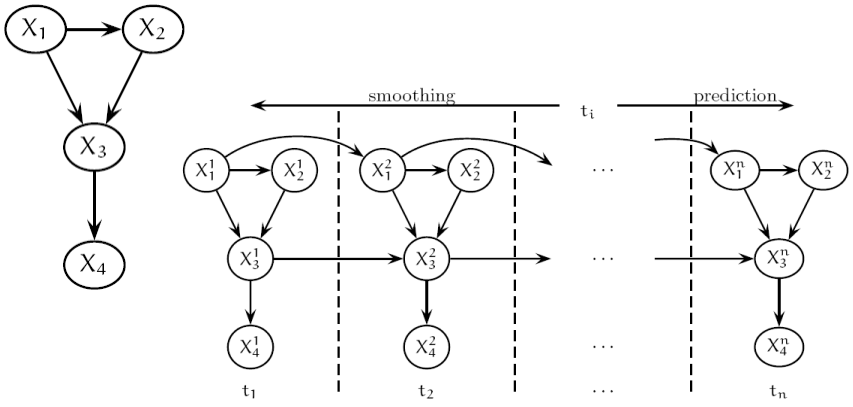
Advantages of BIC

- The BIC score is intuitively appealing because it contains
  1. a term that shows how well the model predicts the data when the parameter set is equal to its ML value, and
  2. a term that punishes for model complexity.
- Another nice feature of the BIC is that it does not depend on the prior distribution of the parameters, which means there is no need to assess one.

Dynamic Models

- The graph of a probabilistic network is restricted to be a finite acyclic directed graph.
- This seems to imply that probabilistic networks as such do not support models with feedback loops or models of dynamic systems changing over time. This is not the case.
- A common approach to representing and solving dynamic models or models with feedback loops is to unroll the dynamic model for the desired number of time steps and treat the resulting network as a static network.
- The unrolled static network is then solved using a standard algorithm applying evidence at the appropriate time steps.

Static vs. Dynamic Model



Temporal Links

- The *temporal links of a time-slice t_i is the set of links from variables of* time-slice t_{i-1} into variables of time-slice t_i .
- The temporal links of time slice t_i define the conditional distribution of the variables of time slice t_i given the variables of time slice t_{i-1}.
- The temporal links connect variables of adjacent time slices. For instance, the temporal links of time-slice t2 on previous slide is the set {(X_1^1 , X_1^2), (X_3^1 , X_3^2)}.

Important Concepts

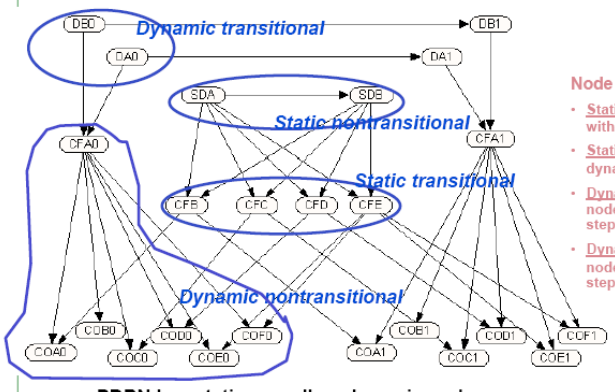
- Let i be the current time step, then
  - *smoothing* is the process of querying about the state of the system at a previous time step j < i given evidence about the system at time i,
  - *filtering* is the process of querying about the state of the system at the current time step, and
  - *prediction* is the process of querying about the state of the system at a future time step j > i.

Important Concepts (Cont'd)

- A dynamic Bayesian network is *stationary when the transition probability* distributions are invariant between time steps.
- A dynamic Bayesian network is first-order Markovian when the variables at time step i+1 are d-separated from the variables at time step i −1 given the variables at time step i.
- When a system is stationary and Markovian, the state of the system at time i + 1 only depends on its state at time i, and the probabilistic dependence relations are the same for all i.
- The Markovian property implies that arcs between time slices only go from one time slice to the subsequent time slice.

George Mason University

Partially Dynamic Bayesian Networks



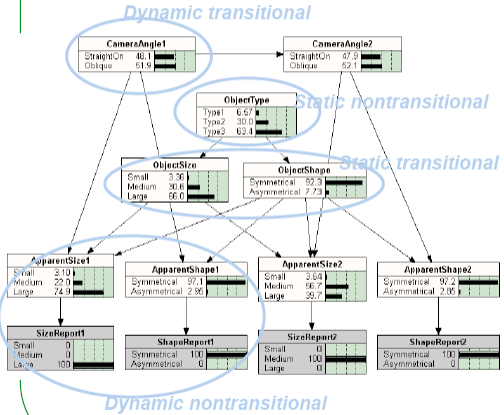
Node types:

- Static nontransitional - static nodes with no dynamic children
- Static transitional - static nodes with dynamic children
- Dynamic nontransitional - dynamic nodes with no children in next time step
- Dynamic transitional - dynamic nodes with children in next time step

- PDBN has static as well as dynamic nodes
- Most DBN theory and algorithms assume all variables are dynamic
- Static nodes can be exploited for efficiency but may degrade accuracy in algorithm not specialized to handle static nodes

George Mason University

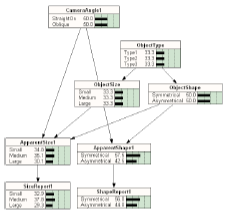
Example PDBN



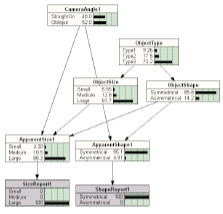
- There are 3 types of objects
  - Type 1 objects are usually small and asymmetrical
  - Type 2 objects are usually medium-sized and may be symmetrical or asymmetrical
  - Type 3 objects are usually large and symmetrical
- Objects can be viewed straight on or obliquely
  - Camera angle is unknown and changes with time
- Asymmetrical objects viewed obliquely may look smaller than actual size and may appear symmetrical

George Mason University

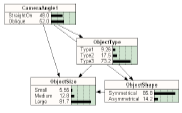
Example: Exact Rollup
(First Evidence)

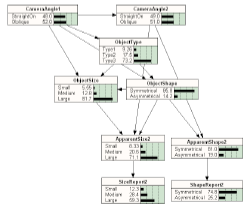


Initialize PDBN at Time Step 1



1. Apply evidence at current time step
2. Update beliefs on all unobserved RVs

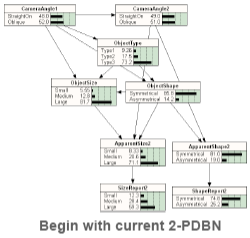




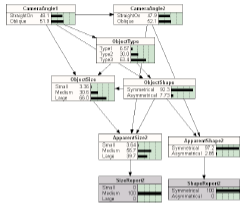
3. Absorb all RVs not part of past expression
4. Extend to next time step

George Mason University

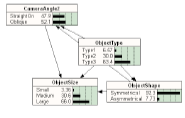
Example: Exact Rollup
(Next Evidence)

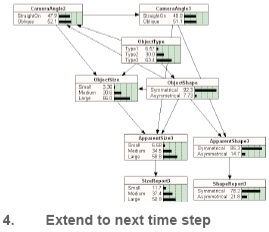


Begin with current 2-PDBN



1. Apply evidence at current time step
2. Update beliefs on all unobserved RVs

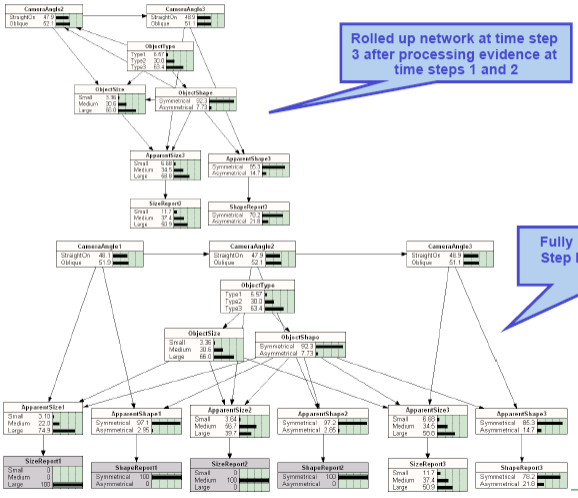




3. Absorb all RVs not part of past expression
4. Extend to next time step

George Mason University

Comparison

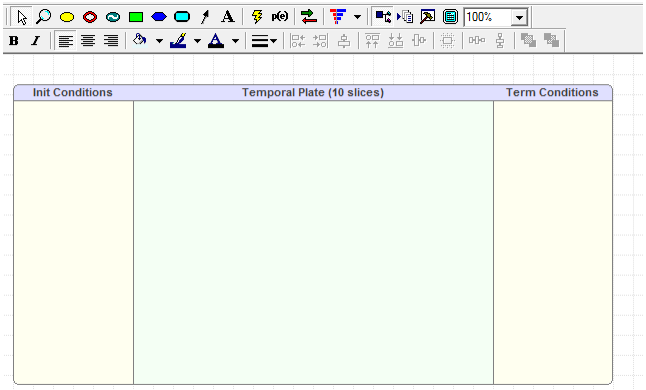


©Kathryn Blackmond Laskey Spring 2008 Unit 5 - 14 -

GeNIe Demo for Temporal Models

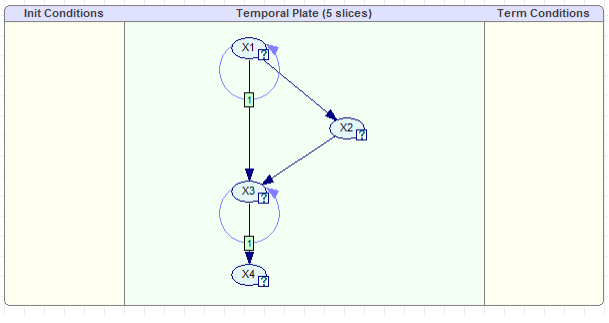
GeNIe Demo (Cont'd)

- You can bring the "Temporal Plate" by clicking Network -> Enable Temporal Plate



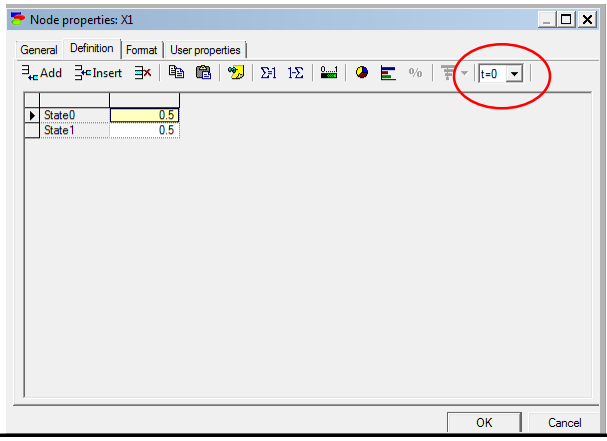
GeNIe Demo (Cont'd)

- Dependence of a node on its previous state is modeled by drawing a self-loop.



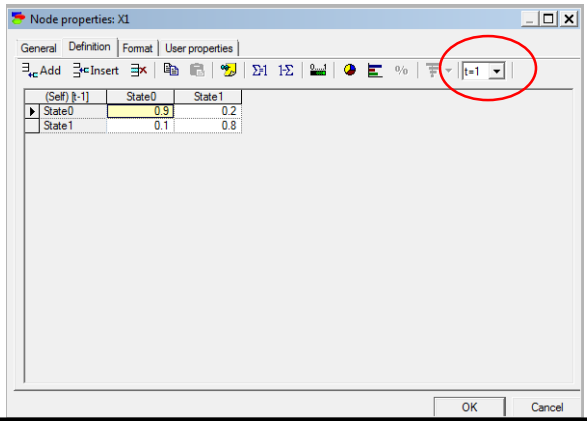
GeNIe Demo (Cont'd)

- Prior probability for Node X1 is defined in the usual manner. The only difference is the presence of t=0.



GeNIe Demo (Cont'd)

- Temporal dependence is captured by specifying the transitional probabilities after setting t=1



GeNIe Demo (Cont'd)

- DBN can be unrolled by selecting Network → Unroll option.