

For Friday

- Read chapter 11, sections 1-2
- Homework:
 - Chapter 14, exercises 1(a-d) and 3(a,b,d)

Program 3

Independencies

- If removing a subset of nodes S from the network renders nodes X_i and X_j disconnected, then X_i and X_j are independent given S , i.e.

$$P(X_i | X_j, S) = P(X_i | S)$$

- However, this is too strict a criteria for conditional independence since two nodes will still be considered independent if there simply exists some variable that depends on both. (i.e. Burglary and Earthquake should be considered independent since the both cause Alarm)

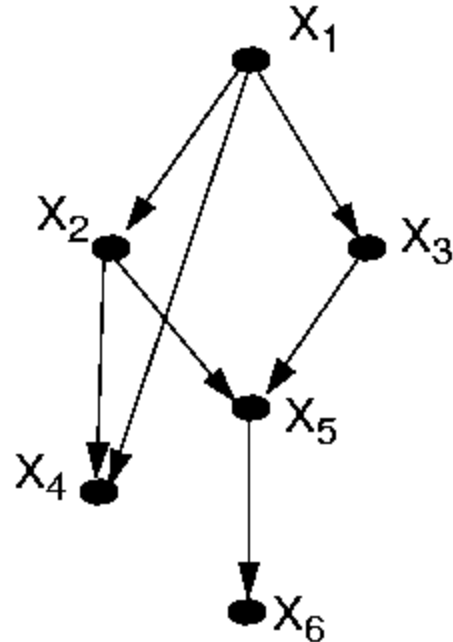
- Unless we know something about a **common effect** of two “independent causes” or a descendent of a common effect, then they can be considered independent.
- For example, **if** we know nothing else, Earthquake and Burglary are **independent**.
- However, if we have information about a common effect (or descendent thereof) then the two “independent” causes become probabilistically linked since evidence for one cause can “explain away” the other.
- If we know the alarm went off, then it makes earthquake and burglary dependent since evidence for earthquake decreases belief in burglary and vice versa.

Types of Connections

- Given a triplet of variables x, y, z where x is connected to z via y , there are 3 possible connection types:
 - tail-to-tail: $x \leftarrow y \rightarrow z$
 - head-to-tail: $x \leftarrow y \leftarrow z$, or $x \rightarrow y \rightarrow z$
 - head-to-head: $x \rightarrow y \leftarrow z$
- For tail-to-tail and head-to-tail connections, x and z are independent given y .
- For head-to-head connections, x and z are “marginally independent” but may become dependent given the value of y or one of its descendants (through “explaining away”).

Separation

- A subset of variables S is said to **separate** X from Y if all (undirected) paths between X and Y are separated by S .
- A path P is separated by a subset of variables S if at least one pair of successive links along P is **blocked** by S .
- Two links meeting head-to-tail or tail-to-tail at a node Z are blocked by S if Z is in S .
- Two links meeting head-to-head at a node Z are blocked by S if neither Z nor any of its descendants are in S .



X_2 and X_3 are separated by $\{X_1\}$ and $\{X_1, X_4\}$

X_2 and X_3 are not separated by $\{X_1, X_6\}$ since X_6 as a descendent of X_5 , unblocks the head-to-head connection at X_5 .

Does $\{X_1\}$ separate X_4 from X_3 ?

Does $\{X_1, X_6\}$ separate X_4 from X_3 ?

Does $\{X_5\}$ separate X_1 from X_6 ?

Does $\{X_2\}$ separate X_4 from X_6 ?

Probabilistic Inference

- Given known values for some evidence variables, we want to determine the **posterior probability** of some query variables.
- **Example:** Given that John calls, what is the probability that there is a Burglary?
- John calls **90%** of the time there is a burglary and the alarm detects **94%** of burglaries, so people generally think it should be fairly high (80-90%). But this ignores the **prior probability** of John calling. John also calls **5%** of the time when there is no alarm. So over the course of **1,000** days we expect one burglary and John will probably call. But John will also call with a false report **50** times during **1,000** days on average. So the call is about **50 times more likely** to be a false report
- **$P(\text{Burglary} \mid \text{JohnCalls}) \sim 0.02$.**
- Actual probability is **0.016** since the alarm is not perfect (an earthquake could have set it off or it could have just went off on its own). Of course even if there was no alarm and John called incorrectly, there could have been an undetected burglary anyway, but this is very unlikely.

Types of Inference

- **Diagnostic (evidential, abductive):** From effect to cause.

$$P(\text{Burglary} \mid \text{JohnCalls}) = 0.016$$

$$P(\text{Burglary} \mid \text{JohnCalls} \wedge \text{MaryCalls}) = 0.29$$

$$P(\text{Alarm} \mid \text{JohnCalls} \wedge \text{MaryCalls}) = 0.76$$

$$P(\text{Earthquake} \mid \text{JohnCalls} \wedge \text{MaryCalls}) = 0.18$$

- **Causal (predictive):** From cause to effect

$$P(\text{JohnCalls} \mid \text{Burglary}) = 0.86$$

$$P(\text{MaryCalls} \mid \text{Burglary}) = 0.67$$

More Types of Inference

- **Intercausal (explaining away):** Between causes of a common effect.

$$P(\text{Burglary} \mid \text{Alarm}) = 0.376$$

$$P(\text{Burglary} \mid \text{Alarm} \wedge \text{Earthquake}) = 0.003$$

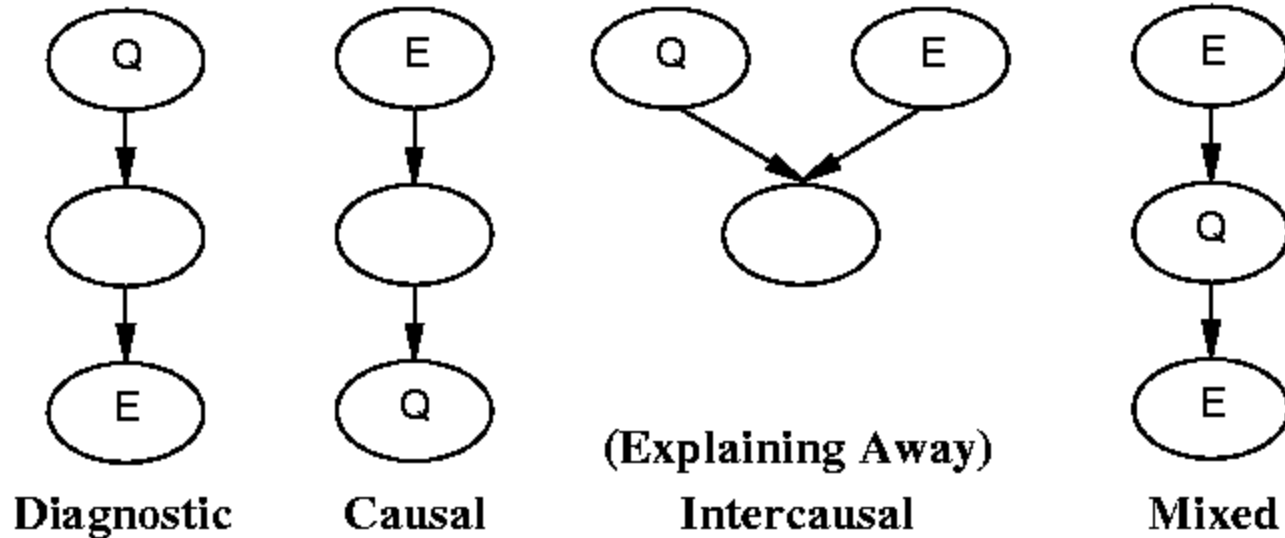
- **Mixed:** Two or more of the above combined
(diagnostic and causal)

$$P(\text{Alarm} \mid \text{JohnCalls} \wedge \neg \text{Earthquake}) = 0.03$$

(diagnostic and intercausal)

$$P(\text{Burglary} \mid \text{JohnCalls} \wedge \neg \text{Earthquake}) = 0.017$$

Types of Inference (figure)



Inference Algorithms

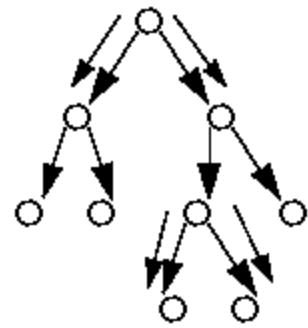
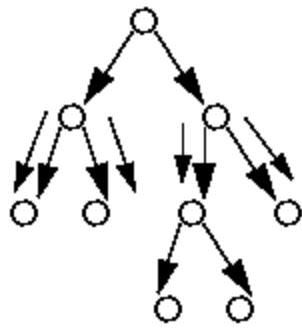
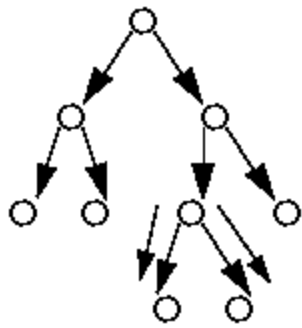
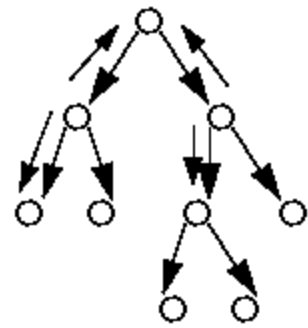
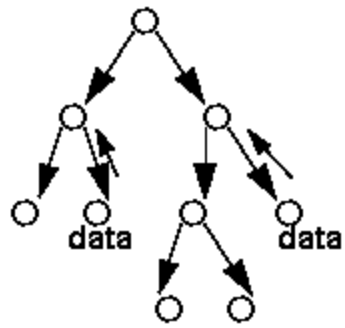
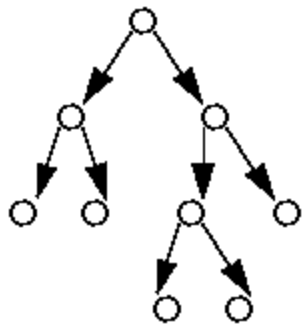
- Most inference algorithms for Bayes nets are not goal-directed and calculate posterior probabilities for all other variables.
- In general, the problem of Bayes net inference is NP-hard (exponential in the size of the graph).

Polytree Inference

- For singly-connected networks or polytrees, in which there are no undirected loops (there is at most one undirected path between any two nodes), polynomial (linear) time algorithms are known.
- Details of inference algorithms are somewhat mathematically complex, but algorithms for polytrees are structurally quite simple and employ simple propagation of values through the graph.

Belief Propagation

- Belief propagation and updating involves transmitting two types of messages between neighboring nodes:
 - λ messages are sent from children to parents and involve the strength of evidential support for a node.
 - π messages are sent from parents to children and involve the strength of causal support.



Propagation Details

- Each node B acts as a simple processor which maintains a vector $\lambda(B)$ for the total evidential support for each value of the corresponding variable and an analogous vector $\pi(B)$ for the total causal support.
- The belief vector $BEL(B)$ for a node, which maintains the probability for each value, is calculated as the normalized product:

$$BEL(B) = \alpha \lambda(B) \pi(B)$$

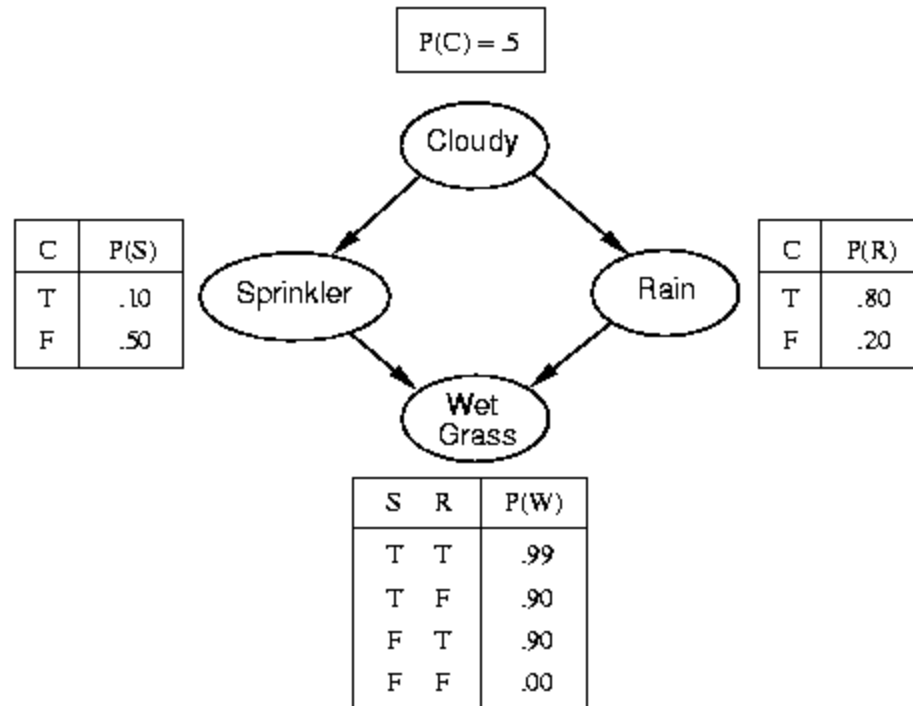
Propogation Details (cont.)

- Computation at each node involve λ and π message vectors sent between nodes and consists of simple matrix calculations using the CPT to update belief (the λ and π node vectors) for each node based on new evidence.
- Assumes CPT for each node is a matrix (M) with a column for each value of the variable and a row for each conditioning case (all rows must sum to 1).

		value of Alarm	
		T	F
Values of Burglary & Earthquake	T T	0.95	0.05
	T F	0.94	0.06
	F T	0.29	0.71
	F F	0.001	0.999

- Propagation algorithm is simplest for trees in which each node has only one parent (i.e. one cause).
- To initialize, $\lambda(B)$ for all leaf nodes is set to all 1's and $\pi(B)$ of all root nodes is set to the priors given in the CPT. Belief based on the root priors is then propagated down the tree to all leaves to establish priors for all nodes.
- Evidence is then added incrementally and the effects propagated to other nodes.

Multiply Connected Networks



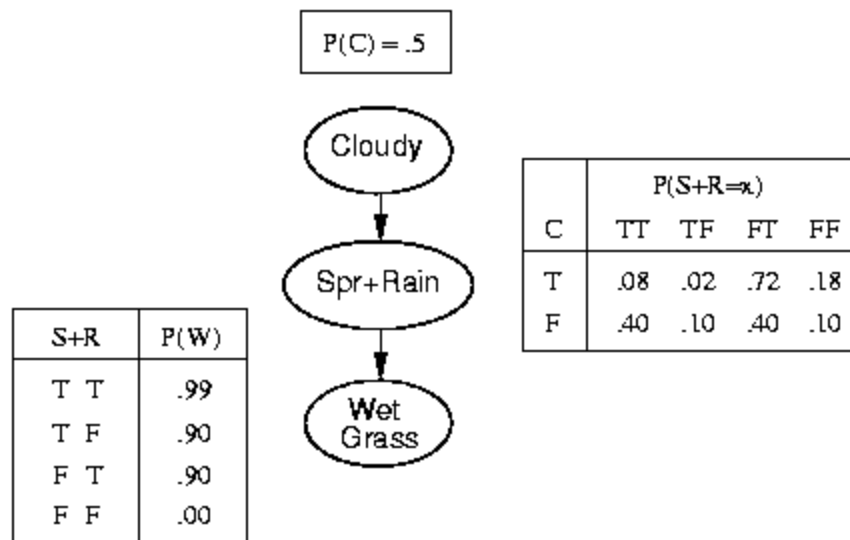
- Inference in **multiply connected networks** with undirected loops is exponential in general.

Basic Solution Approaches

- **Clustering**: Merge nodes to eliminate loops.
- **Cutset Conditioning**: Create several trees for each possible condition of a set of nodes that break all loops.
- **Stochastic simulation**: Approximate posterior probabilities by running repeated random trials testing various conditions.

Clustering

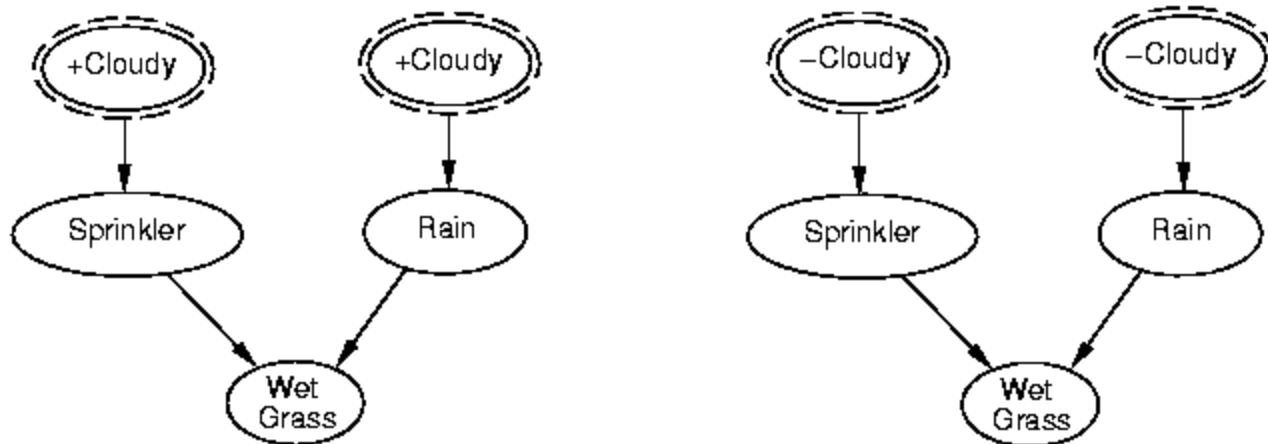
- Eliminate all loops by merging nodes to create **meganodes** that have the cross-product of values of the merged nodes.



- Number of values for merged node is exponential in the number of nodes merged.
- Still reasonably tractable for many network topologies requiring relatively little merging to eliminate loops.

Cutset Conditioning

- First find a **cutset** of nodes such that if all nodes in the set are removed, then all undirected loops are eliminated.
- Create a separate polytree for each possible combination of values for the nodes in the cutset, calculate results for each possible case using polytree methods, and combine total evidence.



- The number of polytrees needed is exponential in the size of the cutset.

Applications of Bayes Nets

- **Medical diagnosis** (Pathfinder, outperforms leading experts in diagnosis of lymph-node diseases)
- **Device diagnosis** (Diagnosis of printer problems in Microsoft Windows)
- **Information retrieval** (Prediction of relevant documents)
- **Computer vision** (Object recognition)

Planning

Search

- What are characteristics of good problems for search?
- What does the search know about the goal state?
- Consider the package problem on the exam:
 - How well would search REALLY work on that problem?

Search vs. Planning

- Planning systems:
 - Open up action and goal representation to allow selection
 - Divide and conquer by subgoaling
 - Relax the requirement for sequential construction of solutions

Planning in Situation Calculus

$PlanResult(p,s)$ is the situation resulting from executing p in s

$$PlanResult([],s) = s$$

$$PlanResult([a/p],s) = PlanResult(p,Result(a,s))$$

Initial state $At(Home,S_0) \wedge \neg Have(Milk,S_0) \wedge \dots$

Actions as Successor State axioms

$$Have(Milk,Result(a,s)) \Leftrightarrow [(a=Buy(Milk) \wedge At(Supermarket,s)) \vee Have(Milk,s) \wedge a \neq \dots]$$

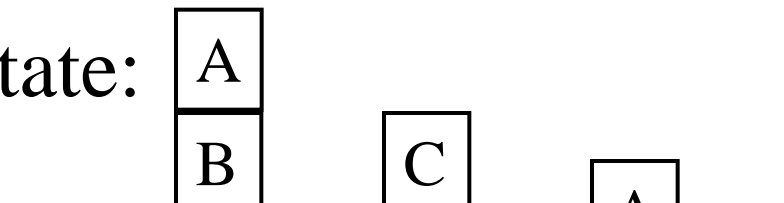

Query $s=PlanResult(p,S_0) \wedge At(Home,s) \wedge Have(Milk,s) \wedge \dots$

Solution $p = Go(Supermarket),Buy(Milk),Buy(Bananas),Go(HWS),\dots]$

- Principal difficulty: unconstrained branching, hard to apply heuristics

The Blocks World

- We have three blocks A, B, and C
- We can know things like whether a block is **clear** (nothing on top of it) and whether one block is **on** another (or on the table)

- Initial State: 
- Goal State: 

Situation Calculus in Prolog

```
holds(on(A,B),result(puton(A,B),S)) :-  
    holds(clear(A),S), holds(clear(B),S),  
    neq(A,B).
```

```
holds(clear(C),result(puton(A,B),S)) :-  
    holds(clear(A),S), holds(clear(B),S),  
    holds(on(A,C),S),  
    neq(A,B).
```

```
holds(on(X,Y),result(puton(A,B),S)) :-  
    holds(on(X,Y),S),  
    neq(X,A), neq(Y,A), neq(A,B).
```

```
holds(clear(X),result(puton(A,B),S)) :-  
    holds(clear(X),S), neq(X,B).
```

```
holds(clear(table),S).
```

neq(a,table).

neq(table,a).

neq(b,table).

neq(table,b).

neq(c,table).

neq(table,c).

neq(a,b).

neq(b,a).

neq(a,c).

neq(c,a).

neq(b,c).

neq(c,b).

Situation Calculus Planner

```
plan([],_,_).
```

```
plan([G1|Gs], S0, S) :-
```

```
    holds(G1,S),
```

```
    plan(Gs, S0, S),
```

```
    reachable(S,S0).
```

```
reachable(S,S).
```

```
reachable(result(_,S1),S) :-
```

```
    reachable(S1,S).
```

- However, what will happen if we try to make plans using normal Prolog depth-first search?

Stack of 3 Blocks

holds(on(a,b), s0).

holds(on(b,table), s0).

holds(on(c,table),s0).

holds(clear(a), s0).

holds(clear(c), s0).

| ?- cpu_time(db_prove(6,plan([on(a,b),on(b,c)],s0,S)), T).

S = result(puton(a,b),result(puton(b,c),result(puton(a,table),s0)))

T = 1.3433E+01

Invert stack

holds(on(a,table), s0).

holds(on(b,a), s0).

holds(on(c,b),s0).

holds(clear(c), s0).

?- cpu_time(db_prove(6,plan([on(b,c),on(a,b)],s0,S)),T).

S = result(puton(a,b),result(puton(b,c),result(puton(c,table),s0))),

T = 7.034E+00

Simple Four Block Stack

holds(on(a,table), s0).

holds(on(b,table), s0).

holds(on(c,table),s0).

holds(on(d,table),s0).

holds(clear(c), s0).

holds(clear(b), s0).

holds(clear(a), s0).

holds(clear(d), s0).

| ?- cpu_time(db_prove(7,plan([on(b,c),on(a,b),on(c,d)],s0,S)),T).

S = result(puton(a,b),result(puton(b,c),result(puton(c,d),s0))),

T = 2.765935E+04

7.5 hours!