CS 1571 Introduction to AI
Lecture 19

Inference in Bayesian belief networks

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Bayesian belief networks (BBNs)

Bayesian belief networks.

- Represent the full joint distribution over the variables more compactly with a **smaller number of parameters**.
- Take advantage of **conditional and marginal independences** among random variables

- **A and B are independent**

$$P(A,B) = P(A)P(B)$$

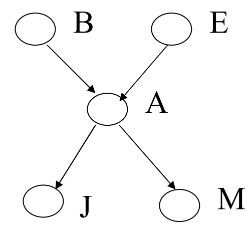
- **A and B are conditionally independent given C**

$$P(A,B \mid C) = P(A \mid C)P(B \mid C)$$
$$P(A \mid C,B) = P(A \mid C)$$

Bayesian belief networks (general)

Two components: $B = (S, \Theta_S)$

- **Directed acyclic graph**
  – Nodes correspond to random variables
  – (Missing) links encode independences



- **Parameters**
  – Local conditional probability distributions for every variable-parent configuration

$$\mathbf{P}(X_i \mid pa(X_i))$$

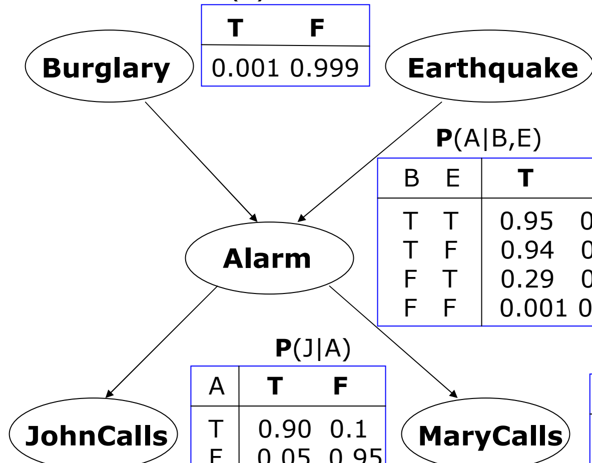
Where:

$pa(X_i)$ - stand for parents of X_i

P(A|B,E)

B	E	**T**	**F**
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

Bayesian belief network.



P(B)

T	**F**
0.001	0.999

P(E)

T	**F**
0.002	0.998

P(A|B,E)

B	E	**T**	**F**
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

P(J|A)

A	**T**	**F**
T	0.90	0.1
F	0.05	0.95

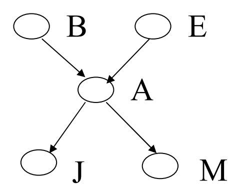
P(M|A)

A	T	F
T	0.7	0.3
F	0.01	0.99

Full joint distribution in BBNs

Full joint distribution is defined in terms of local conditional distributions (obtained via the chain rule):

$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$



Example:

Assume the following assignment of values to random variables

$B=T, E=T, A=T, J=T, M=F$

Then its probability is:

$P(B=T, E=T, A=T, J=T, M=F) =$

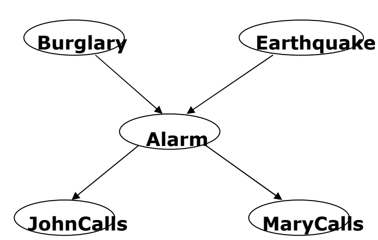
$P(B=T)P(E=T)P(A=T \mid B=T, E=T)P(J=T \mid A=T)P(M=F \mid A=T)$

Parameter complexity problem

- In the BBN the **full joint distribution** is

$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$

- **What did we save?**



Parameters:

full joint: $2^5 = 32$

BBN: $2^3 + 2(2^2) + 2(2) = 20$

Parameters to be defined:

full joint: $2^5 - 1 = 31$

BBN: $2^2 + 2(2) + 2(1) = 10$

Model acquisition problem

The structure of the BBN

- typically reflects causal relations
  (BBNs are also sometime referred to as **causal networks)**
- Causal structure is intuitive in many applications domain and it is relatively easy to define to the domain expert

Probability parameters of BBN

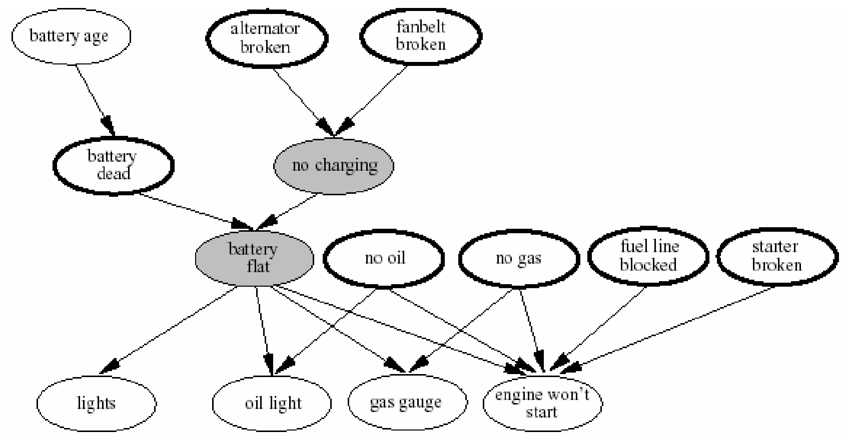
- are conditional distributions relating random variables and their parents
- Complexity is much smaller than the full joint
- It is much easier to obtain such probabilities from the expert or learn them automatically from data

BBNs built in practice

- **In various areas:**
  – Intelligent user interfaces (Microsoft)
  – Troubleshooting, diagnosis of a technical device
  – Medical diagnosis:
    • Pathfinder (Intellipath)
    • CPSC
    • Munin
    • QMR-DT
  – Collaborative filtering
  – Military applications
  – Business and finance
    • Insurance, credit applications

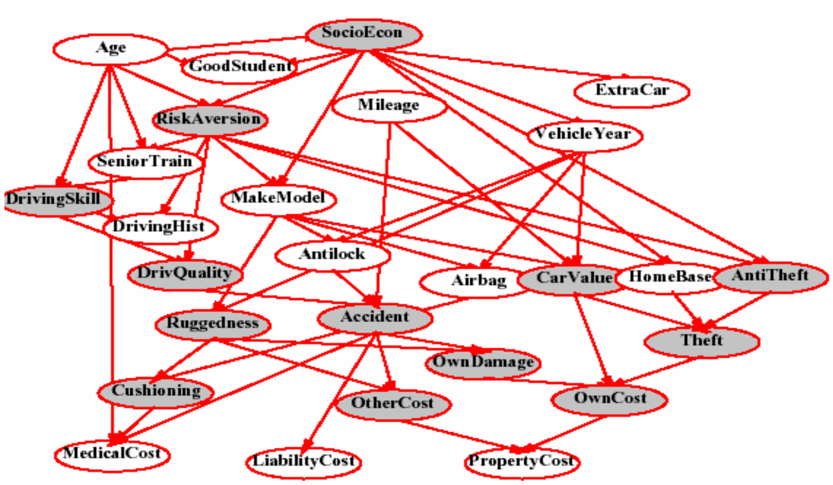
Diagnosis of car engine

- Diagnose the engine start problem

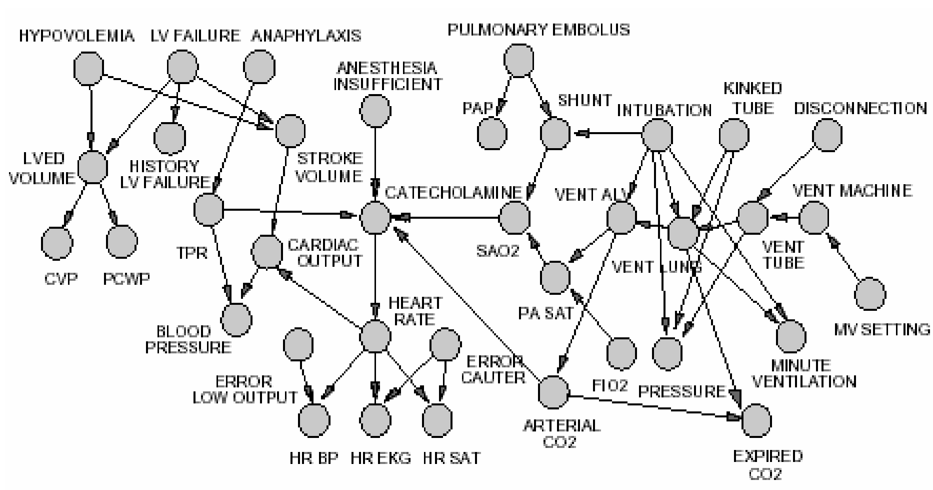


Car insurance example

- Predict claim costs (medical, liability) based on application data

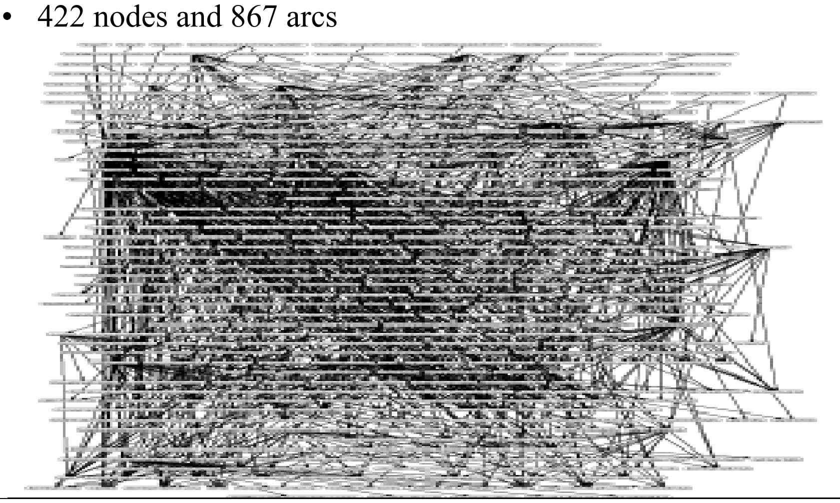


(ICU) Alarm network



CPCS

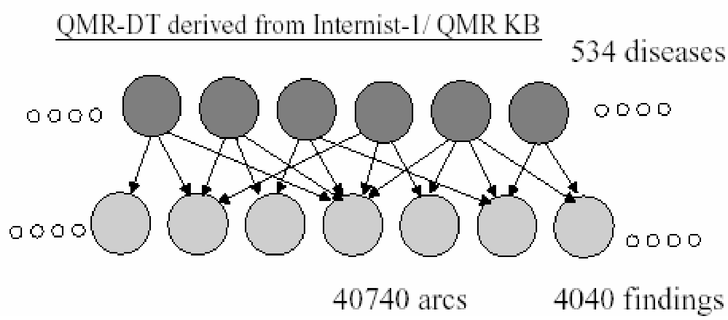
- **C**omputer-based **P**atient **C**ase **S**imulation system (CPCS-PM) developed by Parker and Miller (University of Pittsburgh)
- 422 nodes and 867 arcs



QMR-DT

- **Medical diagnosis in internal medicine**

Bipartite network of disease/findings relations



Inference in Bayesian networks

- BBN models compactly the full joint distribution by taking advantage of existing independences between variables
- Simplifies the acquisition of a probabilistic model
- But we are interested in solving various **inference tasks**:
  - **Diagnostic task. (from effect to cause)**

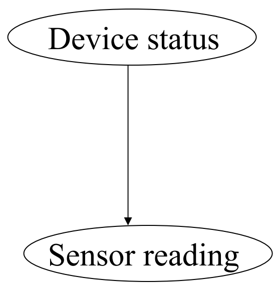
    $\mathbf{P}(Burglary \mid JohnCalls = T)$
  - **Prediction task. (from cause to effect)**

    $\mathbf{P}(JohnCalls \mid Burglary = T)$
  - **Other probabilistic queries** (queries on joint distributions).

    $\mathbf{P}(Alarm)$
- **Main issue:** Can we take advantage of independences to construct special algorithms and speeding up the inference?

Simple BBN. Inference example.

- **Device** (equipment):
  - operating *normally* or *malfunctioning*.
- **A sensor** indirectly monitors the operation of the device
  - Sensor reading is either *high* or *low*



P(Device status)

normal	**malfunctioning**
0.9	0.1

P(Sensor reading| Device status)

Device\Sensor	**high**	**low**
normal	0.1	0.9
malfunctioning	0.6	0.4

Diagnostic inference. Example.

- **Diagnostic inference:** compute the probability of device operating normally given the sensor reading is high

$$P(\mathrm{D}=normal \mid \mathrm{S}=high)=\frac{P(\mathrm{D}=normal,\mathrm{S}=high)}{P(\mathrm{S}=high)}$$

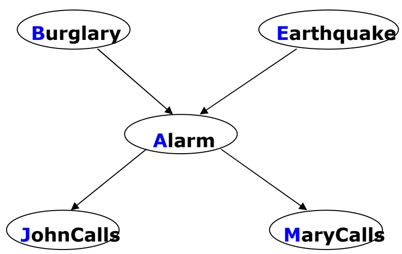
$$P(\mathrm{D}=normal,\mathrm{S}=high)=P(\mathrm{S}=high \mid \mathrm{D}=normal).P(\mathrm{D}=normal)$$

$$P(\mathrm{S}=high)=\sum_{j=\{normal,malfunc\}} P(\mathrm{S}=high,\mathrm{D}=j).$$

$$P(\mathrm{S}=high)=P(\mathrm{S}=high \mid \mathrm{D}=normal)P(\mathrm{D}=normal)+$$
$$+P(\mathrm{S}=high \mid \mathrm{D}=malfunc)P(\mathrm{D}=malfunc)$$

Inference in Bayesian network

- **Bad news:**
  - Exact inference problem in BBNs is NP-hard (Cooper)
  - Approximate inference is NP-hard (Dagum, Luby)
- **But** very often we can achieve significant improvements
- Assume our Alarm network



- Assume we want to compute: $P(J = T)$

Inference in Bayesian networks

Computing: $P(J = T)$

Approach 1. Blind approach.

- Sum out all un-instantiated variables from the full joint,
- express the joint distribution as a product of conditionals

$P(J = T) =$

$$= \sum_{b \in T,F} \sum_{e \in T,F} \sum_{a \in T,F} \sum_{m \in T,F} P(B = b, E = e, A = a, J = T, M = m)$$

$$= \sum_{b \in T,F} \sum_{e \in T,F} \sum_{a \in T,F} \sum_{m \in T,F} P(J = T \mid A = a) P(M = m \mid A = a) P(A = a \mid B = b, E = e) P(B = b) P(E = e)$$

Computational cost:

Number of additions: ?

Number of products: ?

Inference in Bayesian networks

Computing: $P(J = T)$

Approach 1. Blind approach.

- Sum out all un-instantiated variables from the full joint,
- express the joint distribution as a product of conditionals

$$P(J = T) =$$

$$= \sum_{b \in T,F} \sum_{e \in T,F} \sum_{a \in T,F} \sum_{m \in T,F} P(B = b, E = e, A = a, J = T, M = m)$$

$$= \sum_{b \in T,F} \sum_{e \in T,F} \sum_{a \in T,F} \sum_{m \in T,F} P(J = T \mid A = a) P(M = m \mid A = a) P(A = a \mid B = b, E = e) P(B = b) P(E = e)$$

Computational cost:

Number of additions: 15

Number of products: ?

Inference in Bayesian networks

Computing: $P(J = T)$

Approach 1. Blind approach.

- Sum out all un-instantiated variables from the full joint,
- express the joint distribution as a product of conditionals

$$P(J = T) =$$

$$= \sum_{b \in T,F} \sum_{e \in T,F} \sum_{a \in T,F} \sum_{m \in T,F} P(B = b, E = e, A = a, J = T, M = m)$$

$$= \sum_{b \in T,F} \sum_{e \in T,F} \sum_{a \in T,F} \sum_{m \in T,F} P(J = T \mid A = a) P(M = m \mid A = a) P(A = a \mid B = b, E = e) P(B = b) P(E = e)$$

Computational cost:

Number of additions: 15

Number of products: 16*4=64

Inference in Bayesian networks

Approach 2. Interleave sums and products

- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$P(J=T)=$$

$$=\sum_{b\in T,F}\sum_{e\in T,F}\sum_{a\in T,F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(A=a\mid B=b,E=e)P(B=b)P(E=e)$$

$$=\sum_{b\in T,F}\sum_{a\in T.F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]$$

$$=\sum_{a\in T,F}P(J=T\mid A=a)[\sum_{m\in T,F}P(M=m\mid A=a)][\sum_{b\in T,F}P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]]$$

Computational cost:

Number of additions: 1+2*[1+1+2*1]=**?**

Number of products: 2*[2+2*(1+2*1)]=?

Inference in Bayesian networks

Approach 2. Interleave sums and products

- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$P(J=T)=$$

$$=\sum_{b\in T,F}\sum_{e\in T,F}\sum_{a\in T,F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(A=a\mid B=b,E=e)P(B=b)P(E=e)$$

$$=\sum_{b\in T,F}\sum_{a\in T.F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]$$

$$=\sum_{a\in T,F}P(J=T\mid A=a)[\sum_{m\in T,F}P(M=m\mid A=a)][\sum_{b\in T,F}P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]]$$

Computational cost:

Number of additions: 1+2*[1+1+2*1]=**9**

Number of products: 2*[2+2*(1+2*1)]=?

Inference in Bayesian networks

Approach 2. Interleave sums and products

- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$P(J=T)=$$

$$=\sum_{b\in T,F}\sum_{e\in T,F}\sum_{a\in T,F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(A=a\mid B=b,E=e)P(B=b)P(E=e)$$

$$=\sum_{b\in T,F}\sum_{a\in T.F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]$$

$$=\sum_{a\in T,F}P(J=T\mid A=a)[\sum_{m\in T,F}P(M=m\mid A=a)][\sum_{b\in T,F}P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]]$$

Computational cost:

Number of additions: 1+2*[1+1+2*1]=**9**

Number of products: 2*[2+2*(1+2*1)]=16

Inference in Bayesian networks

- The smart interleaving of sums and products can help us to speed up the computation of joint probability queries
- What if we want to compute: $P(B=T,J=T)$

$$P(B=T,J=T)=$$

$$=\sum_{a\in T,F}P(J=T\mid A=a)[\sum_{m\in T,F}P(M=m\mid A=a)]\left[P(B=T)\left[\sum_{e\in T,F}P(A=a\mid B=T,E=e)P(E=e)\right]\right]$$

$$P(J=T)=$$

$$=\sum_{a\in T,F}P(J=T\mid A=a)[\sum_{m\in T,F}P(M=m\mid A=a)]\left[\sum_{b\in T,F}P(B=b)\left[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)\right]\right]$$

- A lot of shared computation
  - Smart cashing of results can save the time for more queries

Inference in Bayesian networks

- The smart interleaving of sums and products can help us to speed up the computation of joint probability queries
- What if we want to compute: $P(B=T, J=T)$

$$P(B=T, J=T) =$$
$$= \sum_{a \in T,F} P(J=T \mid A=a)\Big[\sum_{m \in T,F} P(M=m \mid A=a)\Big]\Big[P(B=T)\Big[\sum_{e \in T,F} P(A=a \mid B=T, E=e)P(E=e)\Big]\Big]$$

$$P(J=T) =$$
$$= \sum_{a \in T,F} P(J=T \mid A=a)\Big[\sum_{m \in T,F} P(M=m \mid A=a)\Big]\Big[\sum_{b \in T,F} P(B=b)\Big[\sum_{e \in T,F} P(A=a \mid B=b, E=e)P(E=e)\Big]\Big]$$

- A lot of shared computation
  - Smart cashing of results can save the time if more queries

Inference in Bayesian networks

- When cashing of results becomes handy?
- What if we want to compute a diagnostic query:

$$P(B=T \mid J=T) = \frac{P(B=T, J=T)}{P(J=T)}$$

- Exactly probabilities we have just compared !!
- There are other queries when cashing and ordering of sums and products can be shared and saves computation

$$\mathbf{P}(B \mid J=T) = \frac{\mathbf{P}(B, J=T)}{P(J=T)} = \alpha \mathbf{P}(B, J=T)$$

- General technique: **Variable elimination**

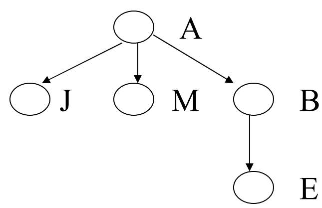
Inference in Bayesian networks

- General idea of variable elimination

$$P(True) = 1 =$$

$$= \sum_{a \in T,F} \underbrace{[\sum_{j \in T,F} P(J=j \mid A=a)]}_{f_J(a)} \underbrace{[\sum_{m \in T,F} P(M=m \mid A=a)]}_{f_M(a)} \underbrace{[\sum_{b \in T,F} P(B=b) \underbrace{[\sum_{e \in T,F} P(A=a \mid B=b, E=e) P(E=e)]}_{f_E(a,b)}}_{f_B(a)}$$

Variable order:



Results cashed in the tree structure

Inference in Bayesian network

- **Exact inference algorithms:**
  - Symbolic inference (D'Ambrosio)
  - Recursive decomposition (Cooper)
  - Message passing algorithm (Pearl)
  - Clustering and joint tree approach (Lauritzen, Spiegelhalter)
  - Arc reversal (Olmsted, Schachter)

- **Approximate inference algorithms:**
  - Monte Carlo methods:
    - Forward sampling, Likelihood sampling
  - Variational methods