**CS 2001**

# Bayesian belief networks

**Milos Hauskrecht**
milos@cs.pitt.edu
MIB 318
X4-8845



## Milos' research interests

**Artificial Intelligence**

- Planning, reasoning and optimization in the presence of uncertainty
- Machine learning
- Applications:
  - medicine
  - Finance and investments

**Main research focus:**

- Models of high dimensional stochastic problems and their efficient solutions

## KB for medical diagnosis.

We want to build a KB system for the **diagnosis of pneumonia**.

**Problem description:**

- **Disease:** pneumonia
- **Patient symptoms (findings, lab tests)**:
  - Fever, Cough, Paleness, WBC (white blood cells) count, Chest pain, etc.

**Representation of a patient case:**

- Statements that hold (are true) for that patient.

  E.g: Fever =*True*
  Cough =*False*
  WBCcount=*High*

**Diagnostic task:** we want to infer whether the patient suffers from the pneumonia or not given the symptoms



## Uncertainty

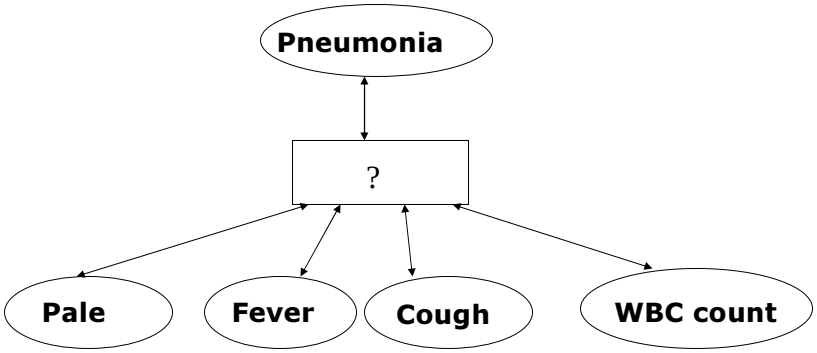
**To make diagnostic inference possible we need to represent rules or axioms that relate symptoms and diagnosis**

**Problem:** disease/symptoms relation is not deterministic (things may vary from patient to patient) – it is **uncertain**

- **Disease ⟶ Symptoms uncertainty**
  - A patient suffering from pneumonia may not have fever all the times, may or may not have a cough, white blood cell test can be in a normal range.
- **Symptoms ⟶ Disease uncertainty**
  - High fever is typical for many diseases (e.g. bacterial diseases) and does not point specifically to pneumonia
  - Fever, cough, paleness, high WBC count combined do not always point to pneumonia

## Modeling the uncertainty.

- How to describe, represent the relations in the presence of uncertainty?
- How to manipulate such knowledge to make inferences?
  - Humans can reason with uncertainty.






## Methods for representing uncertainty

**KB systems** based on propositional and first-order logic often represent uncertain statements, axioms of the domain in terms of

- rules with various **certainty factors**

Very popular in 70-80s (MYCIN)

| | |
|---|---|
| **If** | 1. The stain of the organism is gram-positive, and<br>2. The morphology of the organism is coccus, and<br>3. The growth conformation of the organism is chains |
| **Then** | **with certainty 0.7**<br>the identity of the organism is streptococcus |

**Problems:**

- Chaining of multiple inference rules (propagation of uncertainty)
- Combinations of rules with the same conclusions
- After some number of combinations results not intuitive.

## Representing certainty factors

- **Facts** (propositional statements about the world) are assigned some certainty number reflecting the belief in that the statement is satisfied:

  $CF(Pneumonia = True) = 0.7$

- **Rules** incorporate tests on the certainty values

  $(A \text{ in } [0.5,1]) \wedge (B \text{ in } [0.7,1]) \rightarrow C \text{ with } CF = 0.8$

- **Methods for combination of conclusions**

  $(A \text{ in } [0.5,1]) \wedge (B \text{ in } [0.7,1]) \rightarrow C \text{ with } CF = 0.8$

  $(E \text{ in } [0.8,1]) \wedge (D \text{ in } [0.9,1]) \rightarrow C \text{ with } CF = 0.9$

  $CF(C) = \max[0.9;0.8] = 0.9$

  $CF(C) = 0.9 * 0.8 = 0.72$ **?**

  $CF(C) = 0.9 + 0.8 - 0.9 * 0.8 = 0.98$



## Probability theory

a well-defined coherent theory for representing uncertainty and for reasoning with it

**Representation:**

**Proposition statements** – assignment of values to random variables

$Pneumonia = True \qquad WBCcount = high$

**Probabilities** over statements model the degree of belief in these statements

$P(Pneumonia = True) = 0.001$

$P(WBCcount = high) = 0.005$

$P(Pneumonia = True, Fever = True) = 0.0009$

$P(Pneumonia = False, WBCcount = normal, Cough = False) = 0.97$

# Joint probability distribution

**Joint probability distribution (for a set variables)**

- Defines probabilities for all possible assignments to values of variables in the set

**P**( *pneumonia* , *WBCcount* ) $2 \times 3$ table

**P**(*Pneumonia*)

*WBCcount*

| *Pneumonia* | *high* | *normal* | *low* | |
|---|---|---|---|---|
| *True* | 0.0008 | 0.0001 | 0.0001 | 0.001 |
| *False* | 0.0042 | 0.9929 | 0.0019 | 0.999 |
| | 0.005 | 0.993 | 0.002 | |

**P**(*WBCcount*)

**Marginalization** (summing of rows, or columns)
- summing out variables



# Conditional probability distribution

**Conditional probability distribution:**

- Probability distribution of A given (fixed B)

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

- Conditional probability is defined in terms of joint probabilities
- Joint probabilities can be expressed in terms of conditional probabilities

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

- Conditional probability – is useful for **diagnostic reasoning**
  - the effect of a symptoms (findings) on the disease

$P(Pneumonia = True \mid Fever = True, WBCcount = high, Cough = True)$

## Modeling uncertainty with probabilities

- **Full joint distribution:** joint distribution over all random variables defining the domain
  - it is sufficient to represent the complete domain and to do any type of probabilistic reasoning

**Problems:**

- **Space complexity.** To store full joint distribution requires to remember $O(d^n)$ numbers.
  $n$ – number of random variables, $d$ – number of values
- **Inference complexity.** To compute some queries requires $O(d^n)$ steps.
- **Acquisition problem.** Who is going to define all of the probability entries?



## Pneumonia example. Complexities.

- **Space complexity**.
  - Pneumonia (2 values: T,F), Fever (2: T,F), Cough (2: T,F), WBCcount (3: high, normal, low), paleness (2: T,F)
  - Number of assignments: 2*2*2*3*2=48
  - We need to define at least 47 probabilities.
- **Time complexity**.
  - Assume we need to compute the probability of Pneumonia=T from the full joint

$$P(Pneumonia = T) =$$
$$= \sum_{i \in T,F} \sum_{j \in T,F} \sum_{k=h,n,l} \sum_{u \in T,F} P(Fever = i, Cough = j, WBCcount = k, Pale = u)$$

  - Sum over 2*2*3*2=24 combinations

## Modeling uncertainty with probabilities

- Knowledge based system era (70s – early 80's)
  - Extensional non-probabilistic models
  - Probability techniques avoided because of space, time and acquisition bottlenecks in defining full joint distributions
  - Negative effect on the advancement of KB systems and AI in 80s in general

- Breakthrough (late 80s, beginning of 90s)
  - **Bayesian belief networks**
    - Give solutions to the space, acquisition bottlenecks
    - Significant mprovements in time cost of inferences



## Bayesian belief networks (BBNs)

**Bayesian belief networks.**

- Represent the full joint distribution more compactly with smaller number of parameters.
- Take advantage of conditional and marginal independences among components in the distribution

- **A and B are independent**

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

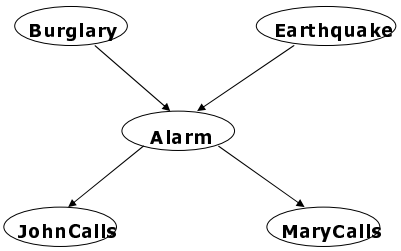
- **A and B are conditionally independent given C**

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

## Alarm system example.

- Assume your house has an **alarm system** against **burglary**. You live in the seismically active area and the alarm system can get occasionally set off by an **earthquake**. You have two neighbors, **Mary** and **John**, who do not know each other. If they hear the alarm they call you, but this is not guaranteed.
- We want to represent the probability distribution of events:
  - Burglary, Earthquake, Alarm, Mary calls and John calls

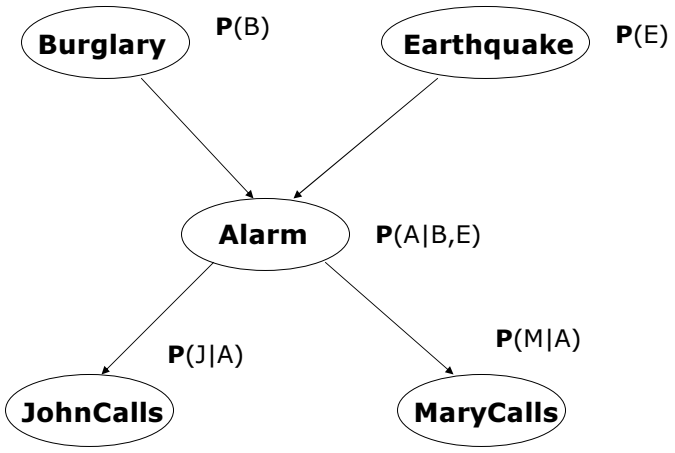
**Causal relations**



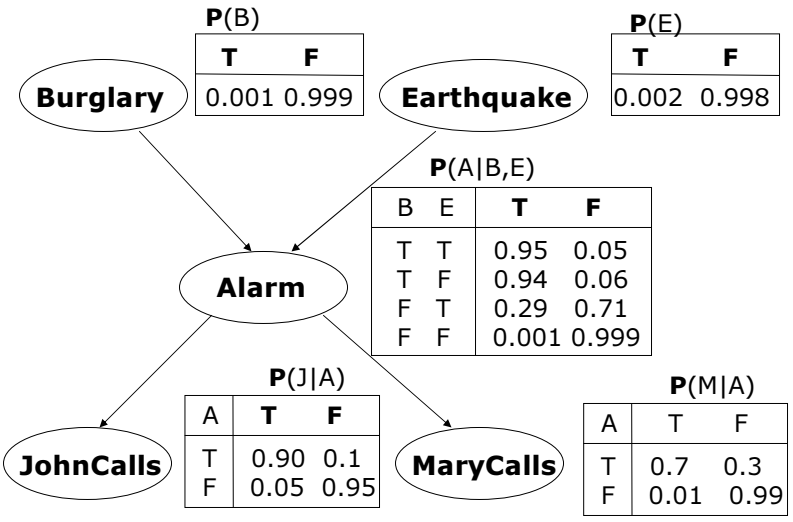



## Bayesian belief network.

1. Graph reflecting direct (causal) dependencies between variables
2. Local conditional distributions relating variables and their parents



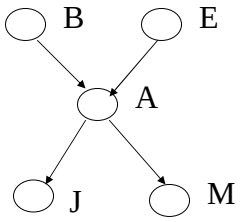
# Bayesian belief network.






# 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 $X_i$

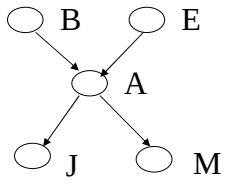
**P**(A|B,E)

| 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 |

# Full joint distribution in BBNs

**Full joint distribution** is defined in terms of local conditional distributions (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:

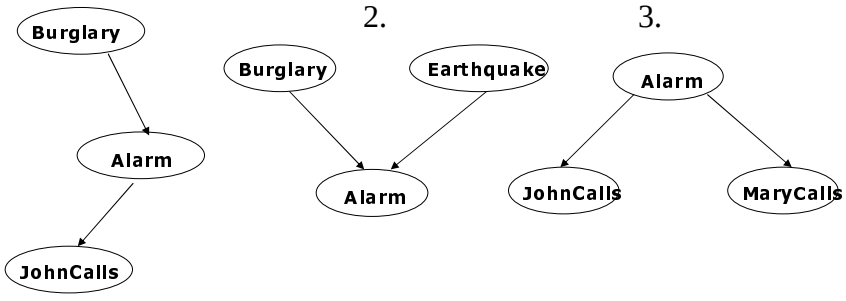
$$P(B=T, E=T, A=T, J=T, M=F) =$$
$$P(B=T)P(E=T)P(A=T \mid B=T, E=T)P(J=T \mid A=T)P(M=F \mid A=T)$$



# Independences in BBNs

- **3 basic independence structures**

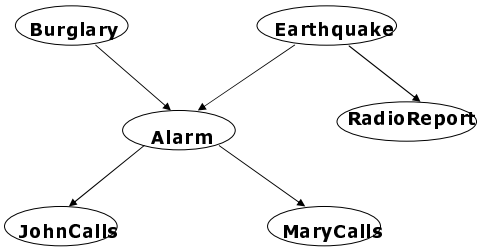
1. 2. 3.




1. JohnCalls **is independent** of Burglary given Alarm
2. Burglary **is independent** of Earthquake (not knowing Alarm)
   Burglary and Earthquake **become dependent** given Alarm !!
3. MaryCalls **is independent** of JohnCalls given Alarm

## Independences in BBNs

- Other dependences/independences in the network



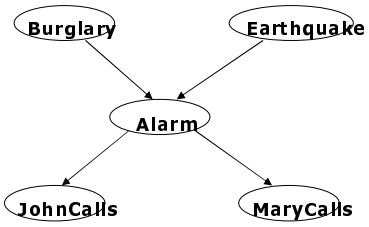

- Earthquake and Burglary are **dependent** given MaryCalls
- Burglary and MaryCalls **are dependent** (not knowing Alarm)
- Burglary and RadioReport **are independent** given Earthquake
- Burglary and RadioReport **are dependent** given MaryCalls



## Parameter complexity problem

- In the BBN the full joint distribution is expressed as a product of conditionals (of smaller) complexity

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




**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 very intuitive in many applications domain and it is relatively easy to obtain from the domain expert

**Probability parameters of BBN** correspond to conditional distributions relating random variables and their parents

- The complexity of local distributions is much smaller than the full joint
- Easier to estimate the probability parameters by consulting an expert or by learning them from data

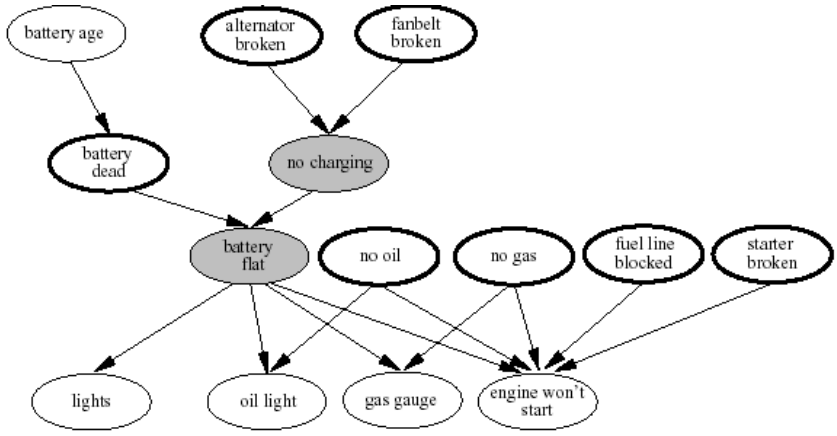


## BBNs built in practice

- **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
  - Insurance, credit applications

# Diagnosis of car engine

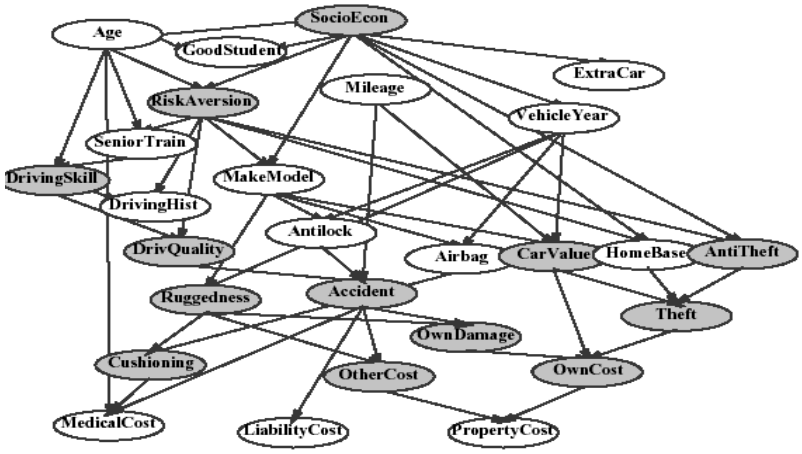
- Diagnose the engine start problem



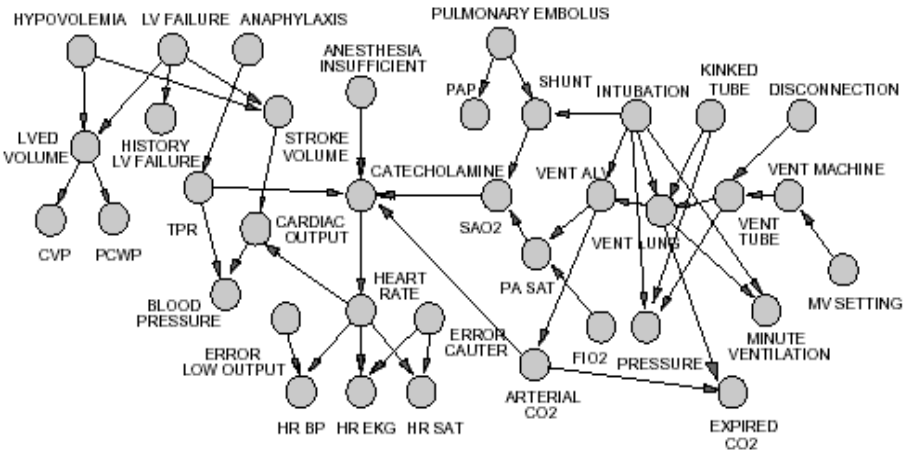



# Car insurance example

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



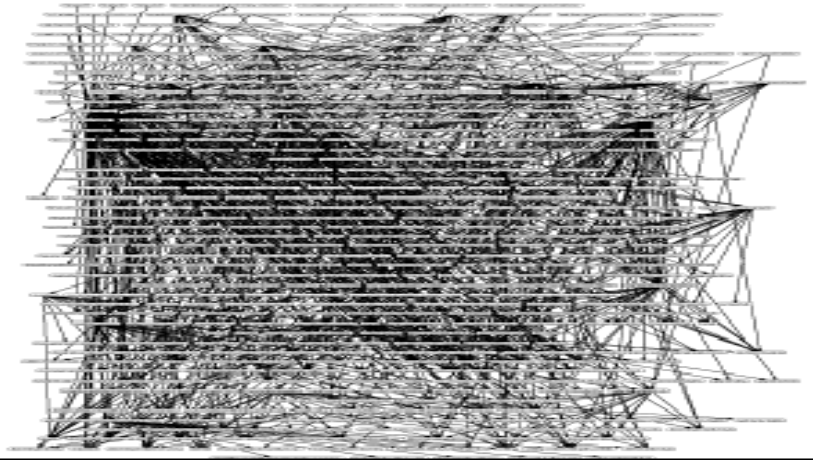
# (ICU) Alarm network






# CPCS

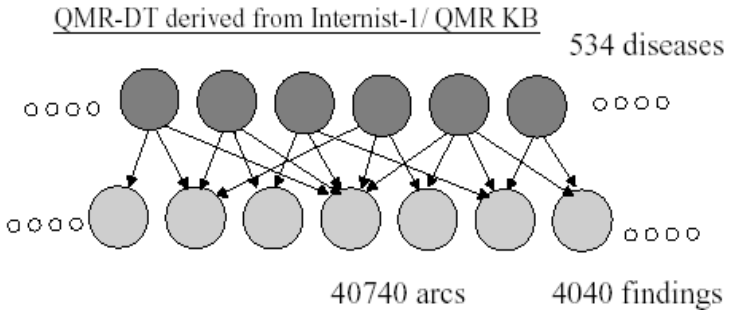
- **C**omputer-based **P**atient **C**ase **S**imulation system (CPCS-PM) developed by Parker and Miller (at University of Pittsburgh)
- 422 nodes and 867 arcs



# 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)**

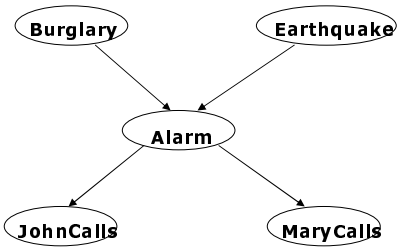
    $\mathbf{P}(\mathit{Burglary} \mid \mathit{JohnCalls} = T)$
  - **Prediction task. (from cause to effect)**

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

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

## 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 uninstantiated 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)+2*(1+2*(1))=**9**

Number of products: 2*(2+2*(1)+2*(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: $P(B=T, J=T)$

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

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

$$P(J=T) =$$

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

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

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

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

- 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

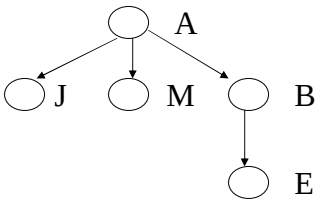
## Inference in Bayesian networks

- 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



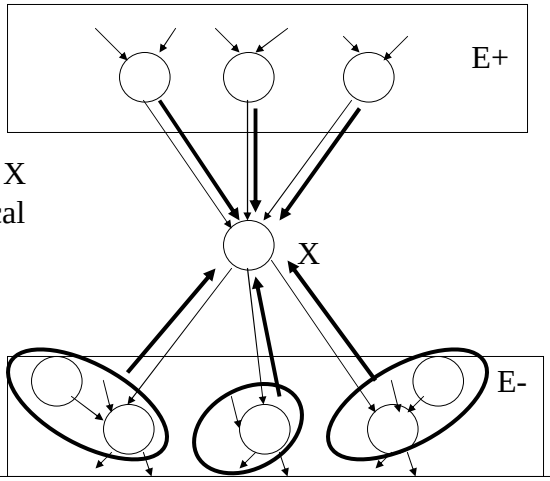
## Inference in Bayesian network

- **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

## Message passing algorithm (Pearl)

- Suitable when we want to compute the probability distribution of X given an evidence **E**, $\mathbf{P}(X \mid E)$

Effect of evidence on X passed to X via its local neighborhood






## Learning Bayesian belief networks

- **Why learning?**
  - "subjective" estimates of conditional probability parameters by a human
    - need to adapt parameters in the light of observed data
  - large databases available
    - uncover important probabilistic dependencies from data and use them in inference tasks

## Learning of BBN

**Learning**. Two steps:

- Learning of the network structure
- Learning of parameters of conditional probabilities

- **Variables**:
  - Observable – values present in every data sample
  - Hidden – values are never in the sample
  - Missing values – values sometimes present, sometimes not
- **<u>Here</u>:**
  - learning parameters for the fixed structure
  - All variables are observable



## Learning via parameter estimation

- We have **a dataset** $D = \{d_1, d_2, .., d_n\}$
  of examples $d_i =< \mathbf{x}_i >$
  Where $\mathbf{x}_i$ is a vector of assignments of values to random variables $\boldsymbol{X}$
- We have **a model of the distribution** over variables in $\boldsymbol{X}$ with parameters $\Theta$
- **Objective:** find parameters $\Theta$ that fit the data the best
- There are various criteria for defining the best set of parameters

## Parameter estimation. Criteria.

- **Maximum likelihood (ML)**

  maximize $P(D \mid \Theta, \xi)$

  $\xi$ - represents prior (background) knowledge

- **Maximum a posteriori probability (MAP)**

  maximize $P(\Theta \mid D, \xi)$

$$P(\Theta \mid D, \xi) = \frac{P(D \mid \Theta, \xi) P(\Theta \mid \xi)}{P(D \mid \xi)}$$



## Parameter estimation. Criteria.

- Using a single set of parameters (either ML or MAP) may not be the best solution
  - two very different parameter settings can be close in terms of probability, using only one of them in inference may introduce a strong bias
- **Solution to this: Full Bayesian approach**
  - Consider all parameter settings and average the result in inference tasks

$$P(\Delta \mid D, \xi) = \int_{\Theta} P(\Delta \mid \Theta, \xi)\, p(\Theta \mid D, \xi) d\Theta$$

## Parameter estimation. Coin example.

- Assume we have a coin, that is biased
  - Outcomes: two possible values -- head or tail
  - We would like to estimate the probability of a head/tail

**Data:** $D$ -- a sequence of $N$ outcomes (tails and heads)

$N_1$ - number of heads seen $N_2$ - number of tails seen

**Model:** probability of a head $\theta$

**Maximum likelihood** estimate of $\theta$

$$\theta_{ML} = \arg\max_{\theta} P(D \mid \theta, \xi)$$

**Likelihood of data:** $P(D \mid \theta, \xi) = \theta^{N_1}(1-\theta)^{N_2}$

**Solution:** $\theta_{ML} = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$



## Parameter estimation. Coin example.

**Maximum a posteriori estimate** $\theta_{MAP} = \arg\max_{\theta} P(\theta \mid D, \xi)$

$$P(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi) P(\theta \mid \xi)}{P(D \mid \xi)}$$ **(via Bayes rule)**

$P(D \mid \theta, \xi)$ - is the likelihood of data

$P(\theta \mid \xi)$ - is the prior probability on $\theta$

**Choice of prior:** Beta distribution

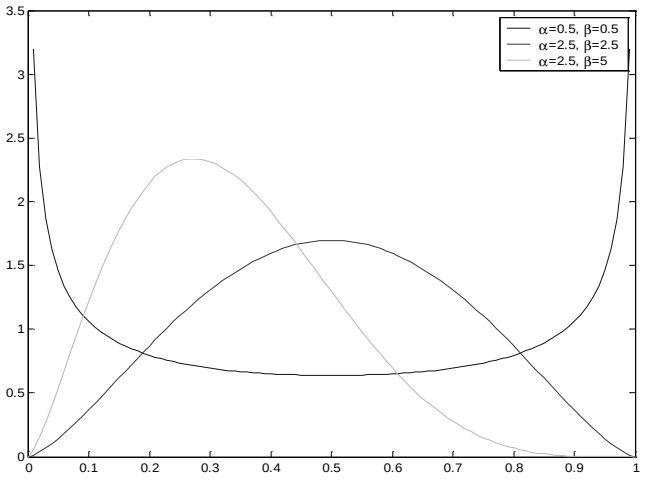
$$P(\theta \mid \xi) = Beta(\theta \mid \alpha_1, \alpha_2) = \frac{\Gamma(\alpha + \alpha_2)_1}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \theta^{\alpha_1 - 1}(1-\theta)^{\alpha_2 - 1}$$

Beta distribution "fits" binomial sampling - **conjugate choices**

$$P(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi) Beta(\theta \mid \alpha_1, \alpha_2)}{P(D \mid \xi)} = Beta(\theta \mid \alpha_1 + N_1, \alpha_2 + N_2)$$

**Solution:** $\theta_{MAP} = \frac{\alpha_1 + N_1}{\alpha_1 + \alpha_2 + N_1 + N_2}$

# Beta distribution






# Estimates of parameters.

- Solutions for the coin toss with two outcomes can be extended to problems with multiple outcomes **(e.g. rolling a dice)**.

**Data:** a sequence of *N* outcomes

$N_i$ - a number of times an outcome i has been seen

**Model parameters:** $\theta = (\theta_1, \theta_2, \ldots \theta_k)$ **s.t.** $\sum_{i=1}^{k} \theta_i = 1$

**ML estimate:** $\theta_{i,ML} = \frac{N_i}{N}$

**MAP estimate (using the Dirichlet prior):**

$$P(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi) Dir(\theta \mid \alpha_1, \alpha_2, .. \alpha_k)}{P(D \mid \xi)} = Dir(\theta \mid \alpha_1 + N_1, .., \alpha_k + N_k)$$

$$\theta_{i,MAP} = \frac{\alpha_i + N_i}{\sum_{i=1,..k} (\alpha_i + N_i)}$$

## Learning of parameters of BBNs

- **Notation**:
  - $i$ ranges over all possible variables $i=1,..,n$
  - $j=1,..,q$ ranges over all possible parent combinations
  - $k=1,..,r$ ranges over all possible variable values

$\theta_{ij}$ is a vector of $\theta_{ijk}$ representing parameters of conditional probability distribution; such that $\sum_{k=1}^{r} \theta_{ijk} = 1$

$N_{ijk}$ - a number of instances in the dataset where parents have value $j$ and the child value $k$

$N_{ij} = \sum_{k=1}^{r} N_{ijk}$

$\alpha_{ikj}$ - prior counts



## Estimates of parameters of BBN

- Two assumptions:
  - **Sample independence**

$$P(D \mid \Theta, \xi) = \prod_{u=1}^{N} p(D_u \mid \Theta, \xi)$$

  - **Parameter independence**

$$P(\Theta \mid D, \xi) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\theta_{ij} \mid D, \xi)$$

Parameters of each node-parents conditional can be optimized independently

## Estimates of parameters of BBN

- Much like a (multiway) **coin toss**. We observe outcomes of random variable values for every combination of values of its parent nodes.

**Estimate of the vector** $\theta_{ij}$ **s.t.** $\sum_{k=1}^{r} \theta_{ijk} = 1$

**ML estimate:** $\theta_{ijk} = \frac{N_{ijk}}{N_{ij}}$

**MAP estimate:** **Use Dirichlet distribution as a prior**

$$P(\theta \mid D,\xi) = \frac{P(D \mid \theta,\xi) Dir(\theta \mid \alpha_1,\alpha_2,..\alpha_k)}{P(D \mid \xi)} = Dir(\theta \mid \alpha_1 + N_1,..,\alpha_k + N_k)$$

$$\theta_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\sum_k \alpha_{ijk} + N_{ijk}}$$



## ML Course

### CS2750 Machine Learning, Spring 2002

Instructor: **Milos Hauskrecht**

Monday, Wednesday – 4:00-5:20pm, MIB 113

**web page**: http://www.cs.pitt.edu/~milos/courses/cs2750/

- Covers modern machine learning techniques, including learning of BBNs, their structures and parameters in different settings, as well as, many other learning frameworks, such as neural networks, support vector machines etc.