Non-Standard-Datenbanken und Data Mining

Bayesian Networks, Inference, and Learning

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

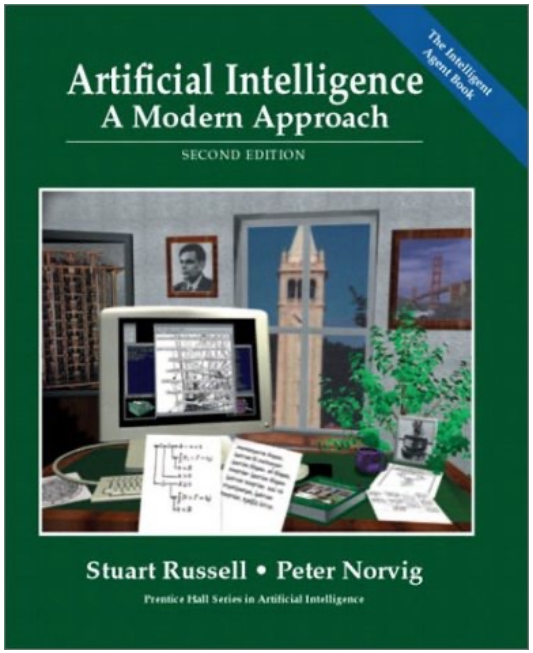
Übersicht

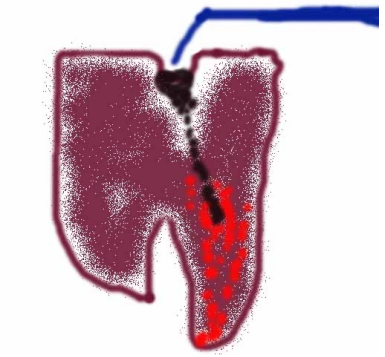
- Semistrukturierte Datenbanken (JSON, XML) und Volltextsuche
- Information Retrieval
- Mehrdimensionale Indexstrukturen
- Cluster-Bildung
- Einbettungstechniken
- First-n-, Top-k-, und Skyline-Anfragen
- Probabilistische Datenbanken, Anfragebeantwortung, Top-k-Anfragen und Open-World-Annahme
- Probabilistische Modellierung, Bayes-Netze, Anfragebeantwortungsalgorithmen, Lernverfahren,
- Temporale Datenbanken und das relationale Modell,
- Probabilistische Temporale Datenbanken
- SQL: neue Entwicklungen (z.B. JSON-Strukturen und Arrays), Zeitreihen (z.B. TimeScaleDB)
- Stromdatenbanken, Prinzipien der Fenster-orientierten inkrementellen Verarbeitung
- Approximationstechniken für Stromdatenverarbeitung, Stream-Mining
- Probabilistische raum-zeitliche Datenbanken und Stromdatenverarbeitungsssysteme: Anfragen und Indexstrukturen, Raum-zeitliches Data Mining, Probabilistische Skylines
- Von NoSQL- zu NewSQL-Datenbanken, CAP-Theorem, Blockchain-Datenbanken

Beispiel

Zahnarzt-Problem mit vier Variablen:

- Toothache (Sind besagte Schmerzen wirklich Zahnschmerzen?)
- Cavity (Es könnte ein Loch sein?)
- Catch (Stahlinstrument erzeugt Testschmerz?)
- Weather (Wetter: sunny,rainy,cloudy,snow)





Nachfolgende Präsentationen enthalten Material aus Kapitel 14 (Sektion 1 and 2)

Bayesian Approach: Prior probability

- Prior or unconditional probabilities of propositions

  e.g., P(*Cavity* = true) = 0.1 and P(*Weather* = sunny) = 0.72 correspond to belief prior to arrival of any (new) evidence

- Probability distribution

  gives values for all possible assignments:

  P(*Weather*) = $\langle 0.72, 0.1, 0.08, 0.1 \rangle^T$

  (normalized, i.e., sums to 1 because one condition must be the case)

Full joint probability distribution

- Joint probability distribution for a set of random variables gives the probability of every atomic event on those random variables
  P(Weather,Cavity) is a 4 × 2 matrix of values:

Weather =	sunny	rainy	cloudy	snow
Cavity = true	0.144	0.02	0.016	0.02
Cavity = false	0.576	0.08	0.064	0.08

- Full joint probability distribution: all random variables involved
  - **P**(Toothache, Catch, Cavity, Weather)
- Every query about a domain can be answered by the full joint distribution

Discrete random variables: Notation

- Dom(Weather) = {sunny, rainy, cloudy, snow} and Dom(Weather) disjoint from domain of other random variables:
  - Atomic event Weather=rainy often written as rainy
  - Example: P(rainy), the random variable Weather is implicitly defined by the value rainy
- Boolean variable Cavity
  - Atomic event Cavity=true written as cavity
  - Atomic event Cavity=false written as ¬cavity
  - Examples: P(cavity) or P(¬cavity)

Inference by enumeration

- Start with the joint probability distribution:

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

- For any proposition φ, sum the probability where it is true: $P(\varphi) = \Sigma_{\omega:\omega \models \varphi} P(\omega)$

- $P(toothache) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$

- Unconditional or **marginal probability** of toothache
- Process is called marginalization or summing out

Ax_3: $P(A \cup B) = P(A) + P(B)$
for disjoint events $A, B \subseteq \Omega$

Conditional probability

- Conditional or posterior probabilities
  e.g., P(*cavity* | *toothache*) = 0.8
  or: <0.8>
  i.e., given that *Toothache=true* is all I know

- (Notation for conditional distributions:
  P(*Cavity* | *Toothache*) is a 2-element vector of 2-element vectors

- If we know more, e.g., *cavity* is also given, then we have
  P(*cavity* | *toothache, cavity*) = 1

- New evidence may be irrelevant, allowing simplification, e.g.,
  P(*cavity* | *toothache, sunny*) = P(*cavity* | *toothache*) = 0.8

- This kind of inference, sanctioned by domain knowledge, is crucial

Conditional probability

- Definition of conditional probability (in terms of uncond. prob.):
  $P(a \mid b) = P(a \wedge b) / P(b)$ if $P(b) > 0$

- Product rule gives an alternative formulation ($\wedge$ is commutative):
  $P(a \wedge b) = P(a \mid b)\, P(b) = P(b \mid a)\, P(a)$

- A general version holds for whole distributions, e.g.,
  $\mathbf{P}(Weather, Cavity) = \mathbf{P}(Weather \mid Cavity)\, \mathbf{P}(Cavity)$

View as a set of 4 × 2 equations, not matrix mult.

(1,1) $P(Weather=sunny \mid Cavity=true)\, P(Cavity=true)$

(1,2) $P(Weather=sunny \mid Cavity=false)\, P(Cavity=false)$, ….

- Chain rule is derived by successive application of product rule:

$$\begin{aligned}\mathbf{P}(X_1, \ldots, X_n) &= \mathbf{P}(X_1,\ldots,X_{n-1})\, \mathbf{P}(X_n \mid X_1,\ldots,X_{n-1}) \\ &= \mathbf{P}(X_1,\ldots,X_{n-2})\, \mathbf{P}(X_{n-1} \mid X_1,\ldots,X_{n-2})\, \mathbf{P}(X_n \mid X_1,\ldots,X_{n-1}) \\ &= \ldots \\ &= \prod_{i=1}^{n} \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1})\end{aligned}$$

Marginalization and conditioning

- Let Y, Z be sequences of random variables s.th. $Y \cup Z$ denotes all random variables describing the world

- Marginalization
  - $\mathbf{P}(\mathbf{Y}) = \Sigma_{\mathbf{z} \in \mathbf{Z}} \mathbf{P}(\mathbf{Y},\mathbf{z})$

- Conditioning
  - $\mathbf{P}(\mathbf{Y}) = \Sigma_{\mathbf{z} \in \mathbf{Z}} \mathbf{P}(\mathbf{Y}|\mathbf{z})P(\mathbf{z})$

Inference by enumeration

- Start with the joint probability distribution:

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

For any proposition φ, sum the atomic events where it is true:
$P(\varphi) = \Sigma_{\omega:\omega \models \varphi} P(\omega)$

- $P(\text{cavity} \vee \textit{toothache}) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$

$(P(\textit{cavity} \vee \textit{toothache}) = P(\textit{cavity}) + P(\textit{toothache}) - P(\textit{cavity} \wedge \textit{toothache}))$

Inference by enumeration

- Start with the joint probability distribution:

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

- Can also compute conditional probabilities:

$$P(\neg cavity \mid toothache) = \frac{P(\neg cavity \wedge toothache)}{P(toothache)}$$

Product rule

$$= \frac{0.016+0.064}{0.108 + 0.012 + 0.016 + 0.064}$$

$$= 0.08/0.2 = 0.4$$

$$P(cavity \mid toothache) = (0.108+0.012)/0.2 = 0.6$$

Normalization

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

- Denominator **P**(**z**) (or P(toothache) in the example before) can be viewed as a normalization constant α

P(Cavity | toothache) = α **P**(Cavity,toothache)

= α [**P**(Cavity,toothache,catch) + **P**(Cavity,toothache,¬ catch)]

= α [<0.108,0.016> + <0.012,0.064>]

= α <0.12,0.08> = <0.6,0.4>

General idea: compute distribution on query variable by fixing evidence variables (Toothache) and summing over hidden variables (Catch)

Inference by enumeration, contd.

Typically, we are interested in
the posterior joint distribution of the query variables **Y**
given specific values **e** for the evidence variables **E**
(**X** are all variables of the modeled world)

Let the hidden variables be **H** = **X** - **Y** – **E** then the required summation of joint entries is done by summing out the hidden variables:

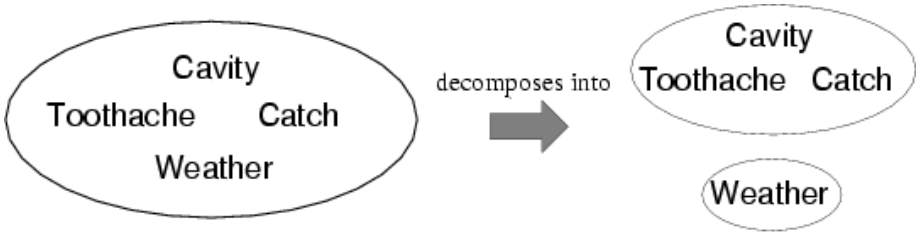
$$\mathbf{P}(\mathbf{Y} \mid \mathbf{E} = \mathbf{e}) = \alpha\mathbf{P}(\mathbf{Y},\mathbf{E} = \mathbf{e}) = \alpha\Sigma_{\mathbf{h}}\mathbf{P}(\mathbf{Y},\mathbf{E}= \mathbf{e}, \mathbf{H} = \mathbf{h})$$

- The terms in the summation are joint entries because **Y**, **E** and **H** together exhaust the set of random variables (**X**)

- Obvious problems:
  1. Worst-case time complexity $O(d^n)$ where d is the largest arity and n denotes the number of random variables
  2. Space complexity $O(d^n)$ to store the joint distribution
  3. How to find the numbers for $O(d^n)$ entries?

Independence

- A and B are independent iff
  $\mathbf{P}(A|B) = \mathbf{P}(A)$ or $\mathbf{P}(B|A) = \mathbf{P}(B)$ or $\mathbf{P}(A, B) = \mathbf{P}(A)\,\mathbf{P}(B)$



$\mathbf{P}$(Toothache, Catch, Cavity, Weather)
$= \mathbf{P}$(Toothache, Catch, Cavity) $\mathbf{P}$(Weather)

- 32 entries reduced to 12;
- *Absolute* independence powerful but rare
- Dentistry is a large field with hundreds of variables, none of which are independent. What to do?

Conditional independence

- **P**(Toothache, Cavity, Catch) has $2^3 - 1 = 7$ independent entries
- If I have a cavity, the probability that the probe catches in it doesn't depend on whether I have a toothache:
  (1) **P**(catch | toothache, cavity) = **P**(catch | cavity)

- The same independence holds if I haven't got a cavity:
  (2) **P**(catch | toothache,¬cavity) = **P**(catch | ¬cavity)

- Catch is conditionally independent of Toothache given Cavity:
  P(Catch | Toothache,Cavity) = **P**(Catch | Cavity)

- Equivalent statements:
  P(Toothache | Catch, Cavity) = **P**(Toothache | Cavity)

  P(Toothache, Catch | Cavity) = **P**(Toothache | Cavity) **P**(Catch | Cavity)

Representation of Domain Knowledge

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

Catch is conditionally independent of Toothache given Cavity:
$\mathbf{P}(\text{Catch} \mid \text{Toothache}, \text{Cavity}) = \mathbf{P}(\text{Catch} \mid \text{Cavity})$

Equivalent statements:
$\mathbf{P}(\text{Toothache} \mid \text{Catch}, \text{Cavity}) = \mathbf{P}(\text{Toothache} \mid \text{Cavity})$

$\mathbf{P}(\text{Toothache} \mid \text{Catch}, \text{Cavity})$
$= \mathbf{P}(\text{Toothache}, \text{Catch}, \text{Cavity}) \;/\; \mathbf{P}(\text{Catch}, \text{Cavity})$
$= \mathbf{P}(\text{Catch}, \text{Toothache}, \text{Cavity}) / \mathbf{P}(\text{Catch} \mid \text{Cavity})\, \mathbf{P}(\text{Cavity})$
$= \mathbf{P}(\text{Catch} \mid \text{Toothache}, \text{Cavity})\, \mathbf{P}(\text{Toothache}, \text{Cavity}) / \mathbf{P}(\text{Catch} \mid \text{Cavity})\, \mathbf{P}(\text{Cavity})$
$= \mathbf{P}(\text{Catch} \mid \text{Cavity})\, \mathbf{P}(\text{Toothache} \mid \text{Cavity})\, \mathbf{P}(\text{Cavity}) / \mathbf{P}(\text{Catch} \mid \text{Cavity})\, \mathbf{P}(\text{Cavity})$
$= \mathbf{P}(\text{Toothache} \mid \text{Cavity})$

Conditional independence contd.

Very many factorizations

- Write out full joint distribution using chain rule:
  P(Toothache, Catch, Cavity)
  = **P**(Toothache | Catch, Cavity) **P**(Catch, Cavity)

  = **P**(Toothache | Catch, Cavity) **P**(Catch | Cavity) **P**(Cavity)

  conditional independence

  = **P**(Toothache | Cavity) **P**(Catch | Cavity) **P**(Cavity)

  i.e., 2 + 2 + 1 = 5 independent numbers
- In most cases, the use of conditional independence reduces the size of the representation of the joint distribution from exponential in n to linear in n
- Conditional independence is our most basic and robust form of knowledge about uncertain environments

Naïve Bayes Model

$$\begin{aligned}
&\mathbf{P}(Cavity|toothache \wedge catch) \\
&= \alpha\,\mathbf{P}(toothache \wedge catch|Cavity)\mathbf{P}(Cavity) \\
&= \alpha\,\mathbf{P}(toothache|Cavity)\mathbf{P}(catch|Cavity)\mathbf{P}(Cavity)
\end{aligned}$$

Usually, the assumption that effects are independent is wrong,
but works well in practice

Agents

- Example: Agent D (doctor) treating patients
- Joint distribution: his understanding of the world (belief state)
  - reason about inputs and
  - act such that outputs are produced
- Input (so-called evidence) is based on the structure of the joint
- Evidence about the world can be uncertain, i.e. we need to update belief state
- For a computational agent we need a computationally feasible representation of the joint for implementing reasoning and acting

If the structure of joint distribution were adapted by the agent itself, we would enter the realm of AI

Uncertain Evidence: Updating the Belief State

	Toothache		¬Toothache	
	PCatch	¬PCatch	PCatch	¬PCatch
Cavity	0.108	0.012	0.072	0.008
¬Cavity	0.016	0.064	0.144	0.576

- Let D now observe Toothache=true with probability 0.8 (e.g., "the patient says so")
- How should D update its belief state?

Uncertain Evidence: Updating the Belief State

	Toothache		¬Toothache	
	PCatch	¬PCatch	PCatch	¬PCatch
Cavity	0.108	0.012	0.072	0.008
¬Cavity	0.016	0.064	0.144	0.576

- Let E be the evidence such that P(Toothache|E) = 0.8
- We want to compute P(c∧t∧pc|E) = P(c∧pc|t,E) P(t|E)
- Since E is not directly related to the cavity or the probe catch, we consider that c and pc are independent of E given t, hence: P(c∧pc|t,E) = P(c∧pc|t)

Uncertain Evidence: Updating the Belief State

	Toothache		¬Toothache	
	PCatch	¬PCatch	PCatch	¬PCatch
Cavity	~~0.108~~ 0.432	~~0.012~~ 0.048	~~0.072~~ 0.018	~~0.008~~ 0.002
¬Cavity	~~0.016~~ 0.064	~~0.064~~ 0.256	~~0.144~~ 0.036	~~0.576~~ 0.144

- Let E be the evidence such that P(Toothache|E) = 0.8
- t∧pc|E) = P(c∧pc|t,E) P(t|E)
- Since E is not directly related to the cavity or the probe catch, we consider that of E given t, hence: P(c∧pc|t,E) = P(c∧

To get these 4 probabilities we normalize their sum to 0.8

To get these 4 probabilities we normalize their sum to 0.2

Issues

- If a state is described by n propositions, then a belief state contains 2^n states (possibly, some have probability 0)
- $\rightarrow$ Modeling difficulty: many numbers must be entered in the first place
- $\rightarrow$ Computational issue: memory size and time

	Toothache		¬Toothache	
	PCatch	¬PCatch	PCatch	¬PCatch
Cavity	0.108	0.012	0.072	0.008
¬Cavity	0.016	0.064	0.144	0.576

- Toothache and PCatch are independent given Cavity (or ¬Cavity), but this relation is hidden in the numbers ! [Verify this]
- Bayesian networks explicitly represent independence among propositions to reduce the number of probabilities defining a belief state

Bayesian networks

- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions

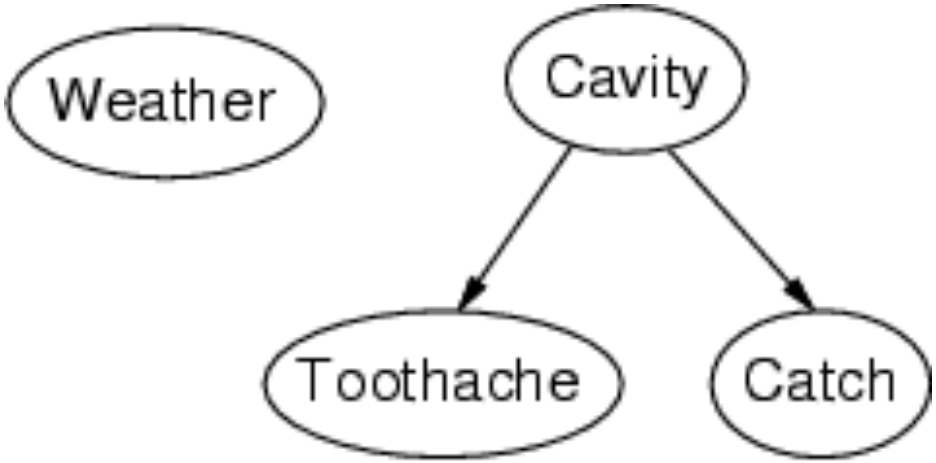
- Syntax:
  - a set of nodes, one per variable
  - a directed, acyclic graph (link ≈ "directly influences")
  - a conditional distribution for each node given its parents:

$$\mathbf{P}(X_i \mid \text{Parents}(X_i))$$

- In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over X_i for each combination of parent values

Example

- Topology of network encodes conditional independence assertions:



- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

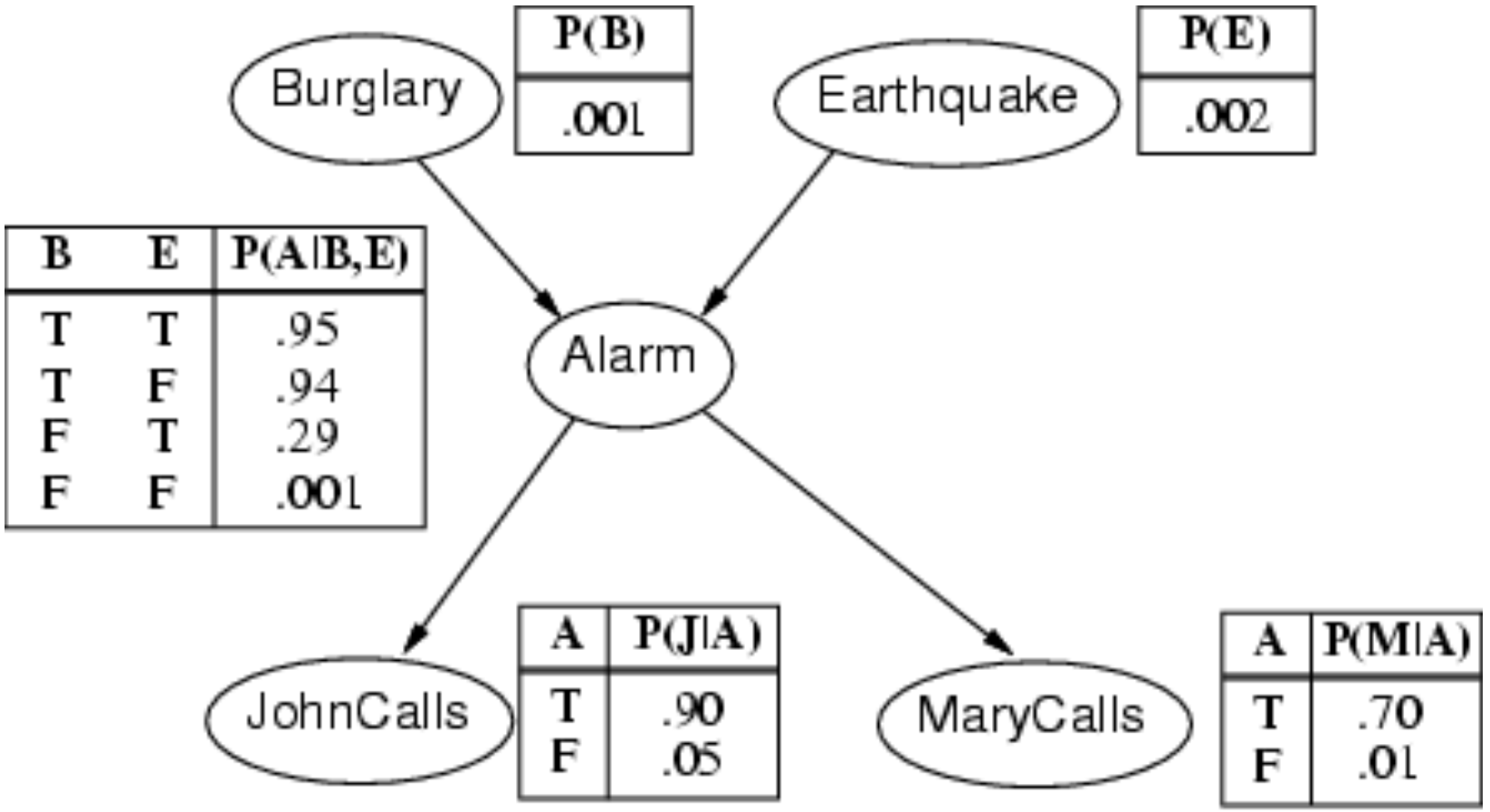
Example

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglary?

- Variables: *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

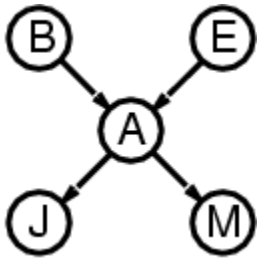
- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

Example contd.



Compactness

- A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values



- Each row requires one number p for $X_i = true$ (the number for $X_i = false$ is just $1\text{-}p$)

- If each variable has no more than k parents, the complete network requires $n \cdot 2^k$ numbers

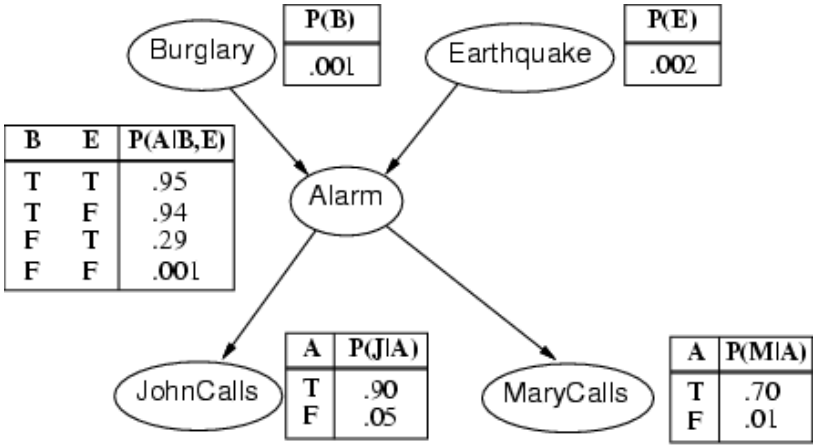
- i.e., grows linearly with n, vs. 2^n for the full joint distribution

- For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5\text{-}1 = 31$)

Semantics

The full joint distribution is defined as the product of the local conditional distributions:

$$\boldsymbol{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \boldsymbol{P}(X_i \mid Parents(X_i))$$



e.g., $\boldsymbol{P}(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$= \boldsymbol{P}(j \mid a)\, \boldsymbol{P}(m \mid a)\, \boldsymbol{P}(a \mid \neg b, \neg e)\, \boldsymbol{P}(\neg b)\, \boldsymbol{P}(\neg e)$

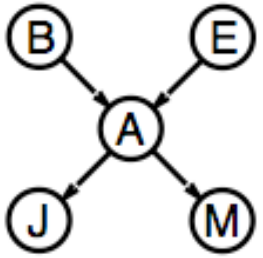
$= 0.90 \times 0.7 \times 0.001 \times 0.999 \times 0.998$

≈ 0.00063

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:



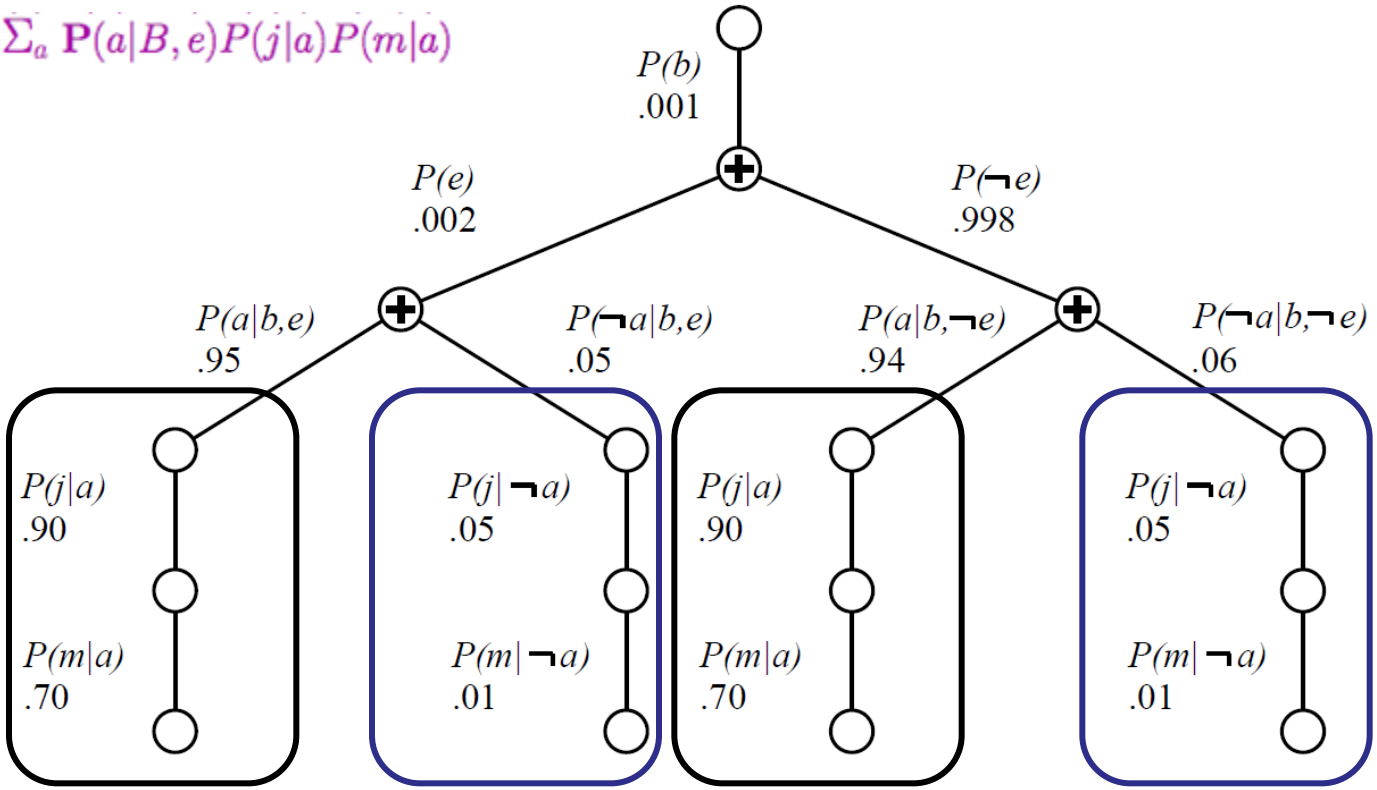
$$\begin{aligned}
&\mathbf{P}(B|j,m) \\
&= \mathbf{P}(B,j,m)/P(j,m) \\
&= \alpha\mathbf{P}(B,j,m) \\
&= \alpha \sum_e \sum_a \mathbf{P}(B,e,a,j,m)
\end{aligned}$$

Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}
&\mathbf{P}(B|j,m) \\
&= \alpha \sum_e \sum_a \mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a) \\
&= \alpha\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e)P(j|a)P(m|a)
\end{aligned}$$

Evaluation Tree

$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e)P(j|a)P(m|a)$

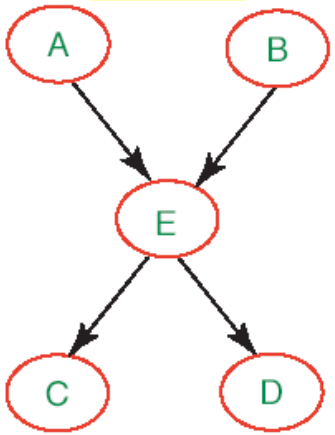


Enumeration is inefficient: repeated computation
e.g., computes $P(j|a)P(m|a)$ for each value of e

Learning BNs: Data Science w/ Complete Data

- We will start by applying ML to the simplest type of BNets learning:
  - known structure
  - Data containing observations for all variables
    - ✓ All variables are observable, no missing data
- The only thing that we need to learn are the network's parameters

Model → Data → Probabilities



A	B	C	D	E
t	f	t	t	f
f	t	t	t	t
t	t	f	t	f
		...		

$P(A)$
$P(B)$
$P(E|A, B)$
$P(C|E)$
$P(D|E)$

Maximum-Likelihood-Parameterschätzung

- Nehme an, die Struktur eines BNs sei bekannt
- Ziel: Schätze BN-Parameter θ
  - Einträge in CPTs, P(X | Parents(X))
- Eine Parametrierung θ ist gut, falls hierdurch die beobachteten Daten wahrscheinlich generiert werden:

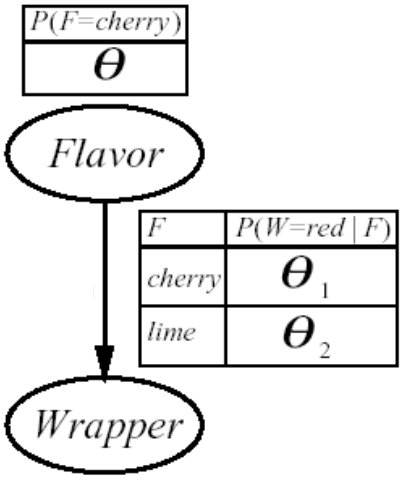
$$P(D \mid \theta) = \prod_m P(x[m] \mid \theta)$$

- Maximum Likelihood Estimation (MLE) Prinzip: Wähle θ^* so, dass $P(D \mid \theta^*)$ maximiert wird

Gleichverteilte, unabhängige Stichprobem (i.i.d. samples)

Anwendungsbeispiel Bonbonfabrik

- Ein Hersteller wählt die Farbe des Bonbonpapiers mit einer bestimmten Wahrscheinlichkeit je nach Geschmack, wobei die entsprechende Verteilung nicht bekannt sei
  - Wenn Geschmack=cherry, wähle rotes Papier mit W'keit θ_1
  - Wenn Geschmack=lime, wähle rotes Papier mit W'keit θ_2
- Das Bayessche Netzwerk enthält drei zu lernende Parameter
  - $\theta\, \theta_1\, \theta_2$

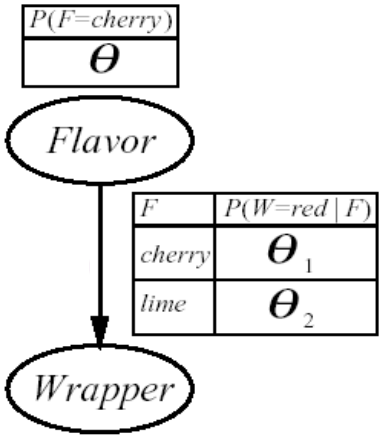


Anwendungsbeispiel Bonbonfabrik

- $P(W=green, F=cherry \mid h_{\theta\theta_1\theta_2}) = \mathbf{(*)}$

  $= P(W=green \mid F=cherry, h_{\theta\theta_1\theta_2})\, P(F=cherry \mid h_{\theta\theta_1\theta_2})$

  $= \theta\,(1-\theta_1)$



- Wir packen N Bonbons aus
  - *c* sind cherry und *ℓ* sind lime
  - r^c cherry mit rotem Papier, g^c cherry mit grünem Papier
  - r^l lime mit rotem Papier, g^ℓ lime mit grünem Papier
  - Jeder Versuch liefert eine Kombination aus Papier und Geschmack wie bei (*)
- $P(\mathbf{d} \mid h_{\theta\theta_1\theta_2})$

  $= \prod_j P(d_j \mid h_{\theta\theta_1\theta_2}) = \theta^c\,(1-\theta)^\ell\,(\theta_1)^{r^c}\,(1-\theta_1)^{g^c}\,(\theta_2)^{r^\ell}\,(1-\theta_2)^{g^\ell}$

Anwendungsbeispiel Bonbonfabrik

- Maximierung des Logarithmus der Zielfunktion
  - $L = c\log\theta + \ell \log(1-\theta) + r^c \log\theta_1 + g^c \log(1-\theta_1) + r^l \log\theta_2 + g^\ell \log(1-\theta_2)$
- Bestimmung der Ableitungen bzgl. $\theta, \theta_1, \theta_2$
  - Ausdrücke ohne Term, nach dem abgeleitet wird, verschwinden

$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c+\ell}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \quad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c+g_c}$$

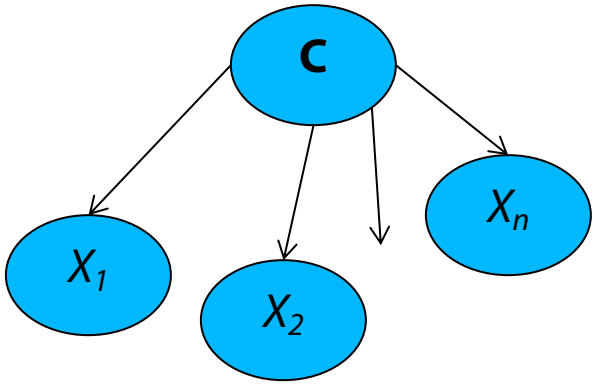
$$\frac{\partial L}{\partial \theta_2} = \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \quad \Rightarrow \quad \theta_2 = \frac{r_\ell}{r_\ell+g_\ell}$$

Maximum-Likelihood-Parameterschätzung

- Schätzung durch Bildung relativer Häufigkeiten
- Dieser Prozess ist auf jedes voll beobachtbare BN anwendbar
- Mit vollständigen Daten und Maximum-Likelihood-Parameterschätzung:
  - Parameterlernen zerfällt in separate Lernprobleme für jeden Parameter (CPT) durch Logarithmierung
  - Jeder Parameter wird durch die relative Häufigkeit eines Knotenwertes bei gegebenen Werten der Elternknoten bestimmt

Beliebte Anwendung: Naives Bayes-Modell

- Naïve Bayes-Modell: Sehr einfaches Bayessches Netzwerk zur Klassifikation
  - *Klassenvariable C* (vorherzusagen) bildet Wurzel
  - Attributvariablen X_i (Beobachtungen) sind Blätter



- Naiv, weil angenommen wird, dass die Attributwerte bedingt unabhängig sind, wenn die Klasse gegeben ist

$$\boldsymbol{P}(C|x_1,x_2,..,x_n) = \frac{\boldsymbol{P}(C,x_1,x_2,..,x_n)}{\boldsymbol{P}(x_1,x_2,..,x_n)} = \alpha \boldsymbol{P}(C)\prod_i \boldsymbol{P}(x_n \mid C)$$

- Deterministische Vorhersagen können durch Wahl der wahrscheinlichsten Klasse erreicht werden
- Skalierung auf realen Daten sehr gut:
  - 2n + 1 Parameter benötigt

Anwendung: Diagnose

Useful for assessing diagnostic probability from causal probability:

$$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$

E.g., let M be meningitis, S be stiff neck:

$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.8 \times 0.0001}{0.1} = 0.0008$$

Note: posterior probability of meningitis still very small!

Constructing Bayesian networks

- 1. Choose an ordering of variables $X_1, \ldots, X_n$
- 2. For $i = 1$ to n
  - add X_i to the network
  - select parents from $X_1, \ldots, X_{i-1}$ such that

$$\mathbf{P}(X_i \mid \text{Parents}(X_i)) = \mathbf{P}(X_i \mid X_1, \ldots X_{i-1})$$

This choice of parents guarantees:

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1}) \quad \text{(chain rule)}$$

$$= \prod_{i=1}^{n} \mathbf{P}(X_i \mid \text{Parents}(X_i)) \quad \text{(by construction)}$$

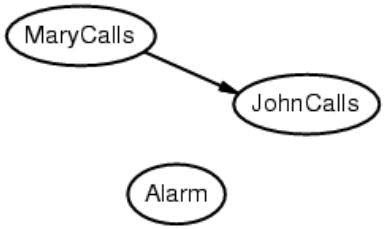
Example

- Suppose we choose the ordering M, J, A, B, E

$\mathbf{P}(J \mid M) = \mathbf{P}(J)$?

Example

- Suppose we choose the ordering M, J, A, B, E

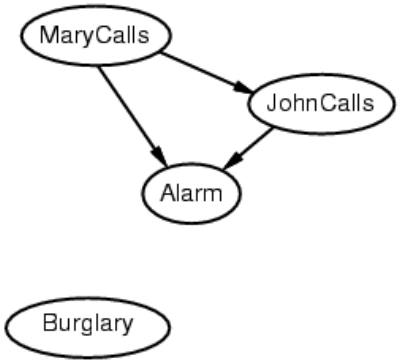


P(J | M) = **P**(J)? **No**

P(A | J, M) = **P**(A)? **P**(A | J, M) = **P**(A | J)?

Example

- Suppose we choose the ordering M, J, A, B, E



P(J | M) = **P**(J)? **No**

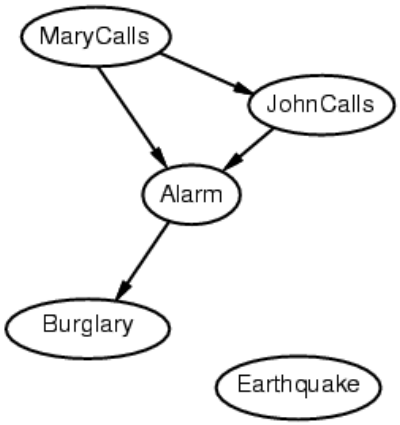
P(A | J, M) = **P**(A)? **P**(A | J, M) = **P**(A | J)? **No**

P(B | A, J, M) = **P**(B | A)?

P(B | A, J, M) = **P**(B)?

Example

- Suppose we choose the ordering M, J, A, B, E



P(J | M) = **P**(J)? **No**

P(A | J, M) = **P**(A | J)? **P**(A | J, M) = **P**(A)? **No**

P(B | A, J, M) = **P**(B | A)? **Yes**

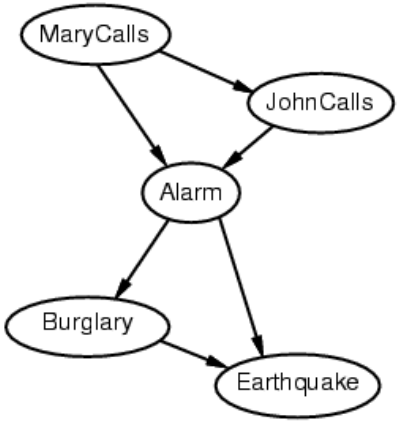
P(B | A, J, M) = **P**(B)? **No**

P(E | B, A ,J, M) = **P**(E | A)?

P(E | B, A, J, M) = **P**(E | A, B)?

Example

- Suppose we choose the ordering M, J, A, B, E



P(J | M) = **P**(J)? **No**

P(A | J, M) = **P**(A | J)? **P**(A | J, M) = **P**(A)? **No**

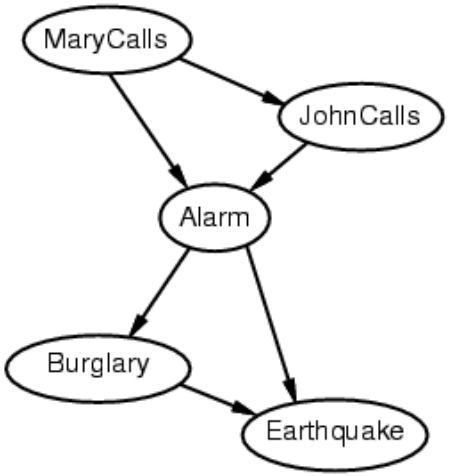
P(B | A, J, M) = **P**(B | A)? **Yes**

P(B | A, J, M) = **P**(B)? **No**

P(E | B, A, J, M) = **P**(E | A)? **No**

P(E | B, A, J, M) = **P**(E | A, B)? **Yes**

Example contd.

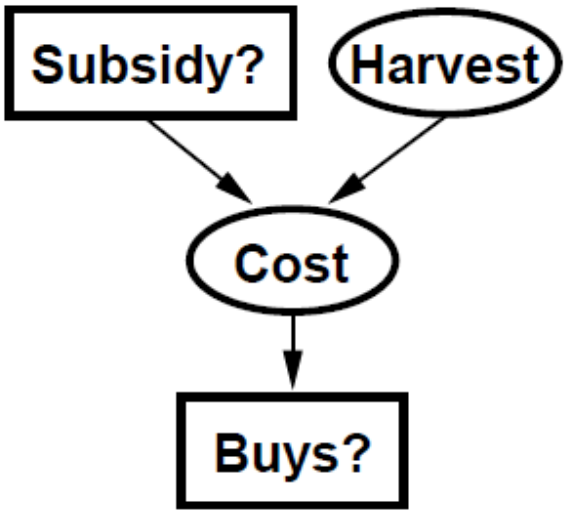


- Deciding conditional independence is hard in non-causal directions
- (Causal models and conditional independence seem hardwired for humans!)
- Network is less compact: 1 + 2 + 4 + 2 + 4 = 13 numbers needed instead of 10.

Hybrid (discrete+continuous) networks

BNs can also deal with continuous RVs or even hybrid ones

Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)



Option 1: discretization—possibly large errors, large CPTs
Option 2: finitely parameterized canonical families (e.g., Gaussian Distribution)

1) Continuous variable, discrete+continuous parents (e.g., *Cost*)
2) Discrete variable, continuous parents (e.g., *Buys?*)

Continuous child variables

Need one conditional density function for child variable given continuous parents, for each possible assignment to discrete parents

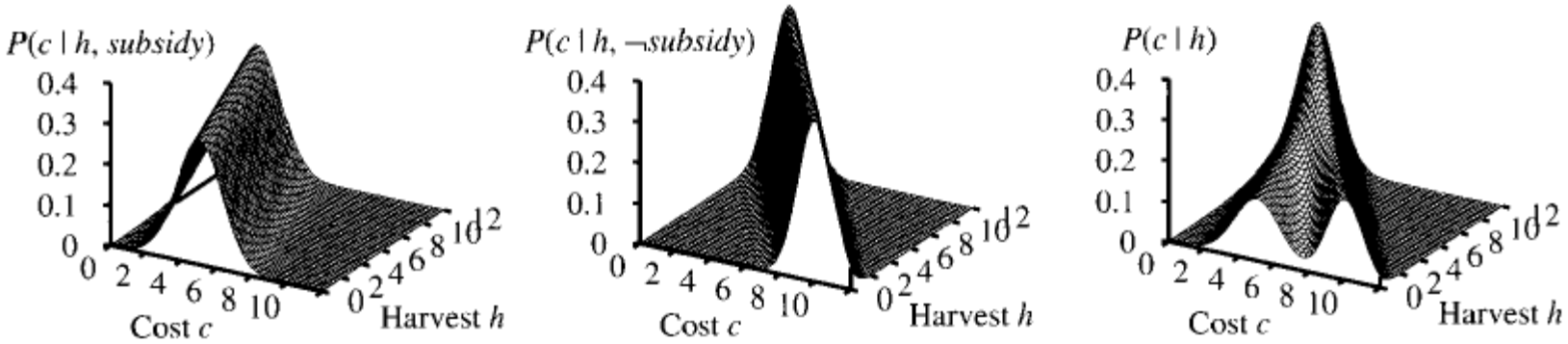
Most common is the linear Gaussian model, e.g.,:

$$\begin{aligned} & P(Cost = c | Harvest = h, Subsidy? = true) \\ &= N(a_t h + b_t, \sigma_t)(c) \\ &= \frac{1}{\sigma_t \sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right) \end{aligned}$$

Mean $Cost$ varies linearly with $Harvest$, variance is fixed

Linear variation is unreasonable over the full range
but works OK if the **likely** range of $Harvest$ is narrow

Continuous child variables



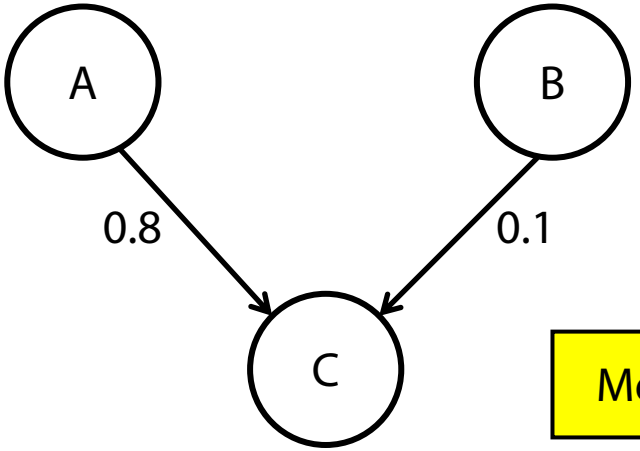
All-continuous network with LG distributions

$\Rightarrow$ full joint distribution is a multivariate Gaussian

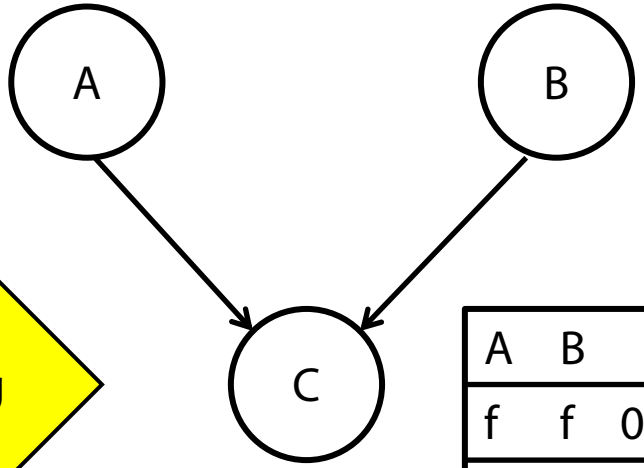
Discrete+continuous LG network is a conditional Gaussian network i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

LG = linear Gaussian

Noisy-OR-Representation



Meaning



A	B	c
f	f	0.0
f	t	0.1
t	f	0.8
t	t	0.1+0.8 - 0.1*0.8

- Substantial reduction of the probabilities to be stored

Inference tasks

Simple queries: compute posterior marginal $\mathbf{P}(X_i|\mathbf{E}=\mathbf{e})$
e.g., $P(NoGas|Gauge=empty, Lights=on, Starts=false)$

Conjunctive queries: $\mathbf{P}(X_i, X_j|\mathbf{E}=\mathbf{e}) = \mathbf{P}(X_i|\mathbf{E}=\mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E}=\mathbf{e})$

Optimal decisions: decision networks include utility information;
probabilistic inference required for $P(outcome|action, evidence)$

Value of information: which evidence to seek next?

Sensitivity analysis: which probability values are most critical?

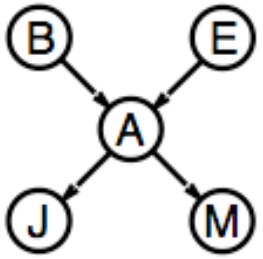
Explanation: what must be true to allow for a certain (desired) conclusion?

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\mathbf{P}(B,j,m)/P(j,m)\\&=\alpha\mathbf{P}(B,j,m)\\&=\alpha\,\Sigma_e\,\Sigma_a\,\mathbf{P}(B,e,a,j,m)\end{aligned}$$



Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&=\alpha\,\Sigma_e\,\Sigma_a\,\mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)\\&=\alpha\mathbf{P}(B)\,\Sigma_e\,P(e)\,\Sigma_a\,\mathbf{P}(a|B,e)P(j|a)P(m|a)\end{aligned}$$

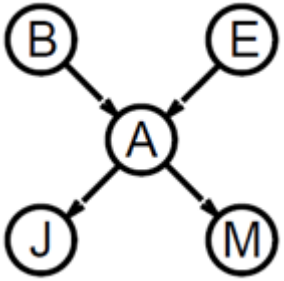
Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

(Calculation over sum-product evaluation tree
with path length n (=number of RVs) and degree d
(= maximal number domain elements))

Basic Objects of VE

$$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

- Track objects called factors
  (right-to-left, in tree: bottom up)
- Initial factors are local CPTs



$$\underbrace{P(B)}_{f_B(B)} \quad \underbrace{P(J|A)}_{f_J(A,J)} \quad \underbrace{P(A|B,E)}_{f_A(A,B,E)} \quad \underbrace{P(M|A)}_{f_M(A)}$$

- During elimination create new factors
- Form of Dynamic Programming

Basic Operations: Pointwise Product

- Pointwise product of factors $\mathbf{f}_1$ and $\mathbf{f}_2$
  - for example: $\mathbf{f}_1(A,B) * \mathbf{f}_2(B,C) = \mathbf{f}(A,B,C)$
  - in general:
    $\mathbf{f}_1(X_1,\ldots,X_j,Y_1,\ldots,Y_k) * \mathbf{f}_2(Y_1,\ldots,Y_k,Z_1,\ldots,Z_l) =$
    $\mathbf{f}(X_1,\ldots,X_j,Y_1,\ldots,Y_k,Z_1,\ldots,Z_l)$
  - has 2^{j+k+l} entries (if all variables are binary)

Join by pointwise product

A	$f_{JM}(A)$
T	.9 * .7
F	.05 * .01

=

A	$f_J(A)$
T	.9
F	.05

A	$f_M(A)$
T	.7
F	.01

$$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

=

A	B	E	$f_A \cdots(A,B,E)$
T	T	T	.95
T	T	F	.94
T	F	T	.29
T	F	F	.001
F	T	T	.05
F	T	F	.06
F	F	T	.71
F	F	F	.999

A	$f_{JM}(A)$
T	.63
F	.0005

Basic Operations: Summing out

- Summing out a variable from a product of factors
  - Move any constant factors outside the summation
  - Add up submatrices in pointwise product of remaining factors

$$\Sigma_x f_1 * \ldots * f_k = f_1 * \ldots * f_i * \Sigma_x f_{i+1} * \ldots * f_k$$
$$= f_1 * \ldots * f_i * f_X$$

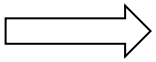
assuming $\mathbf{f}_1, \ldots, \mathbf{f}_i$ do not depend on X

Summing out

$$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

Summing out a



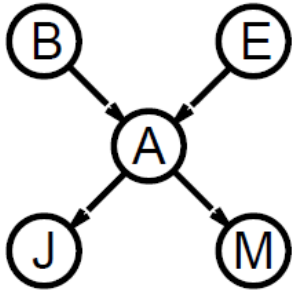
B	E	$f_{\bar{A}JM}(B,E)$
T	T	.95 * .63 + .05 * .0005 = .5985
T	F	.94 * .63 + .06 * .0005 = .5922
F	T	.29 * .63 + .71 * .0005 = .1830
F	F	.001 * .63 + .999 * .0005 = .001129

VE result for Burglary Example

$$
\begin{aligned}
&\mathbf{P}(B|j,m) \\
&\quad = \alpha \underbrace{\mathbf{P}(B)}_{B} \Sigma_e \underbrace{P(e)}_{E} \Sigma_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M} \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) P(j|a) f_M(a) \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) f_J(a) f_M(a) \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a f_A(a,b,e) f_J(a) f_M(a) \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) f_{\bar{A}JM}(b,e) \text{ (sum out } A\text{)} \\
&\quad = \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E\text{)} \\
&\quad = \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

Optimization by Finding Irrelevant Variables

Consider the query $P(JohnCalls|Burglary = true)$



$$P(J|b) = \alpha P(b) \sum_e P(e) \sum_a P(a|b,e) P(J|a) \sum_m P(m|a)$$

Sum over m is identically 1; M is **irrelevant** to the query

Thm 1: Y is irrelevant unless $Y \in Ancestors(\{X\} \cup \mathbf{E})$

Here, $X = JohnCalls$, $\mathbf{E} = \{Burglary\}$, and
$Ancestors(\{X\} \cup \mathbf{E}) =$ {Alarm, Earthquake, Burglary}
so $MaryCalls$ is irrelevant

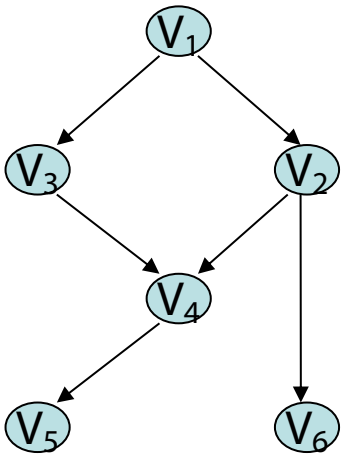
Irrelevant variables can be identified by a graph theoretical criterion on the associated moral graph by m-separation

What else can we do?

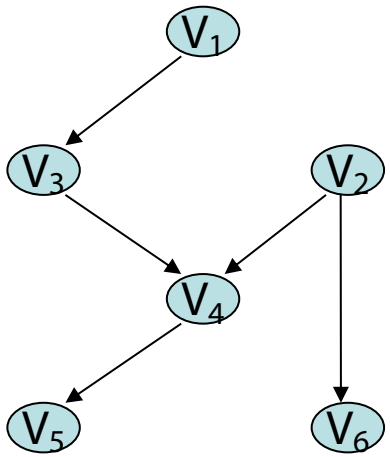
- General methodology: Consider special cases (special data structures)
- Here: Consider polytrees (singly connected networks)
- The following slides on Pearl's belief propagation algorithm are based on slides by Tomas Singliar (adapted by Daniel Lowd)

Polytree Bayesian Networks

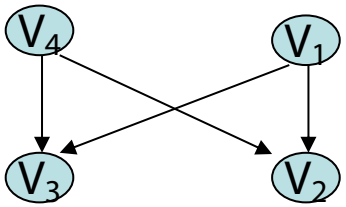
- Bayesian network graph is a *polytree (alias: singly connected network)* iff there is at most one path between any two nodes, V_i and V_k (iff the underlying non-directed graph is a tree)
- Exact BN inference is NP-hard but O(n) on polytree
- (Non-)Examples:



-



+



-

Our inference task

- We know the values of evidence variables given in set E:

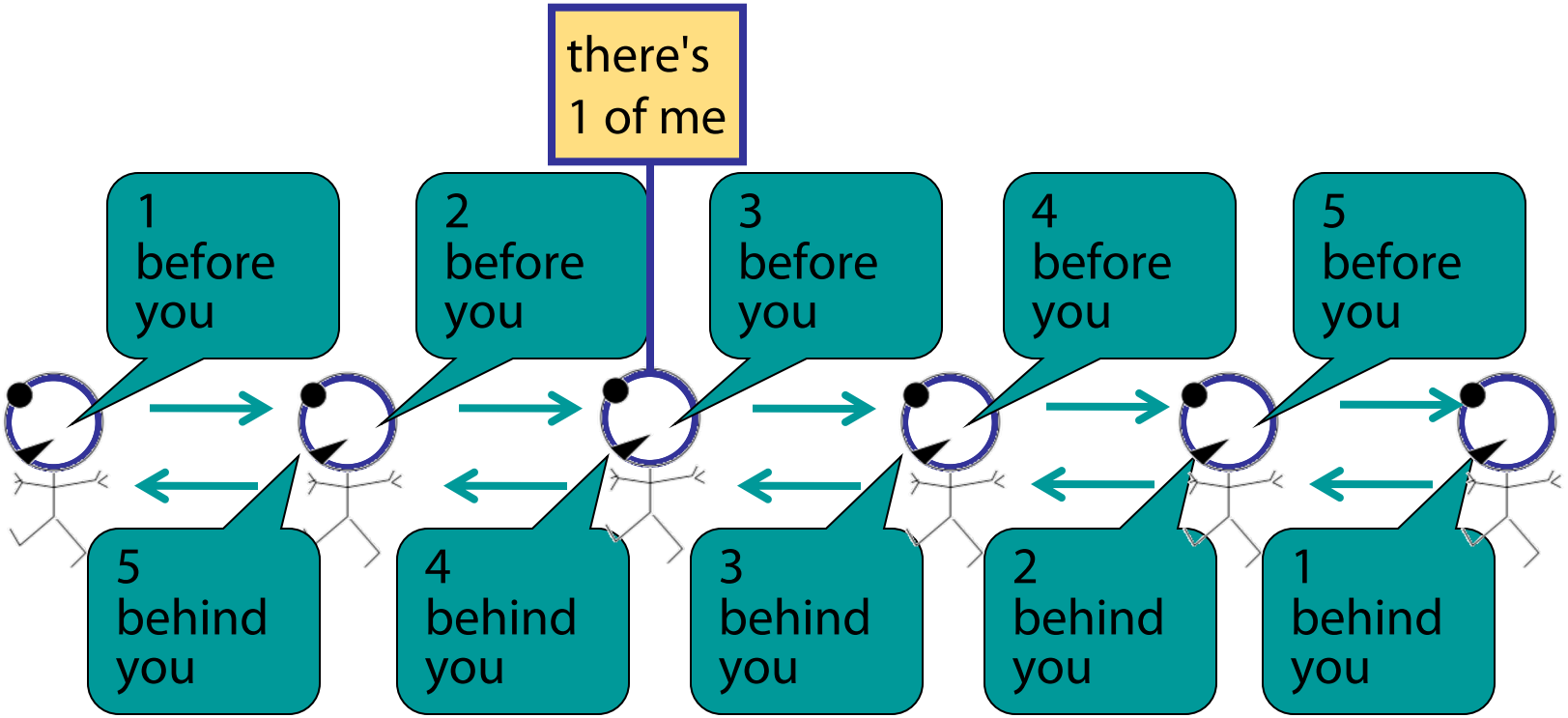
$$V_{e_1}, \ldots, V_{e_{|E|}}$$

- We wish to compute the conditional probability $P(X_i \mid E)$ for all non-evidence variables X_i.

- IDEA:
  - Do variable elimination in parallel (locally on each node/RV)
  - Send/receive required data to/from other nodes

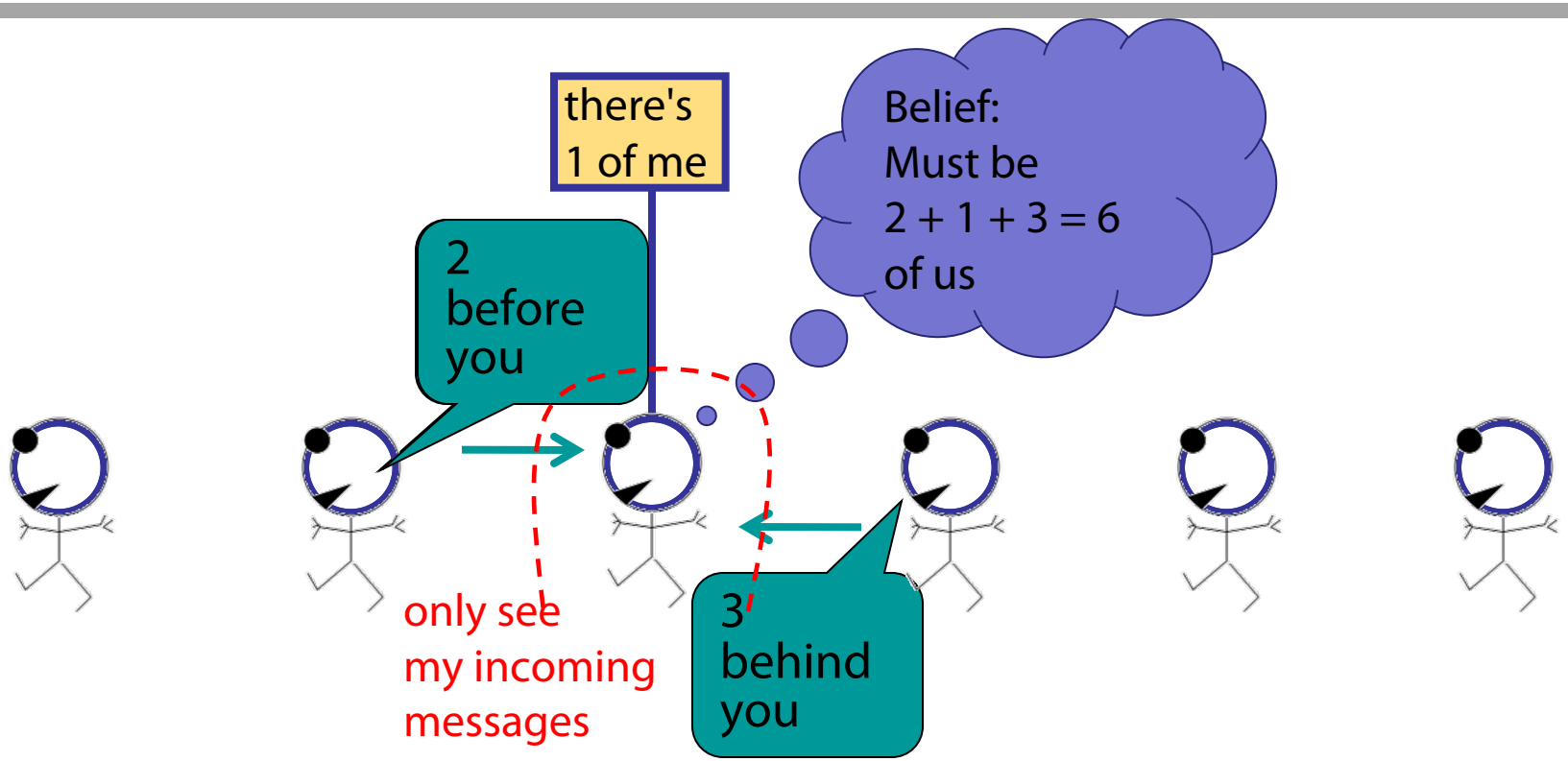
- The following soldier counting example on message passing is adapted from slides of M. Gormley

Counting soldiers example (linear)

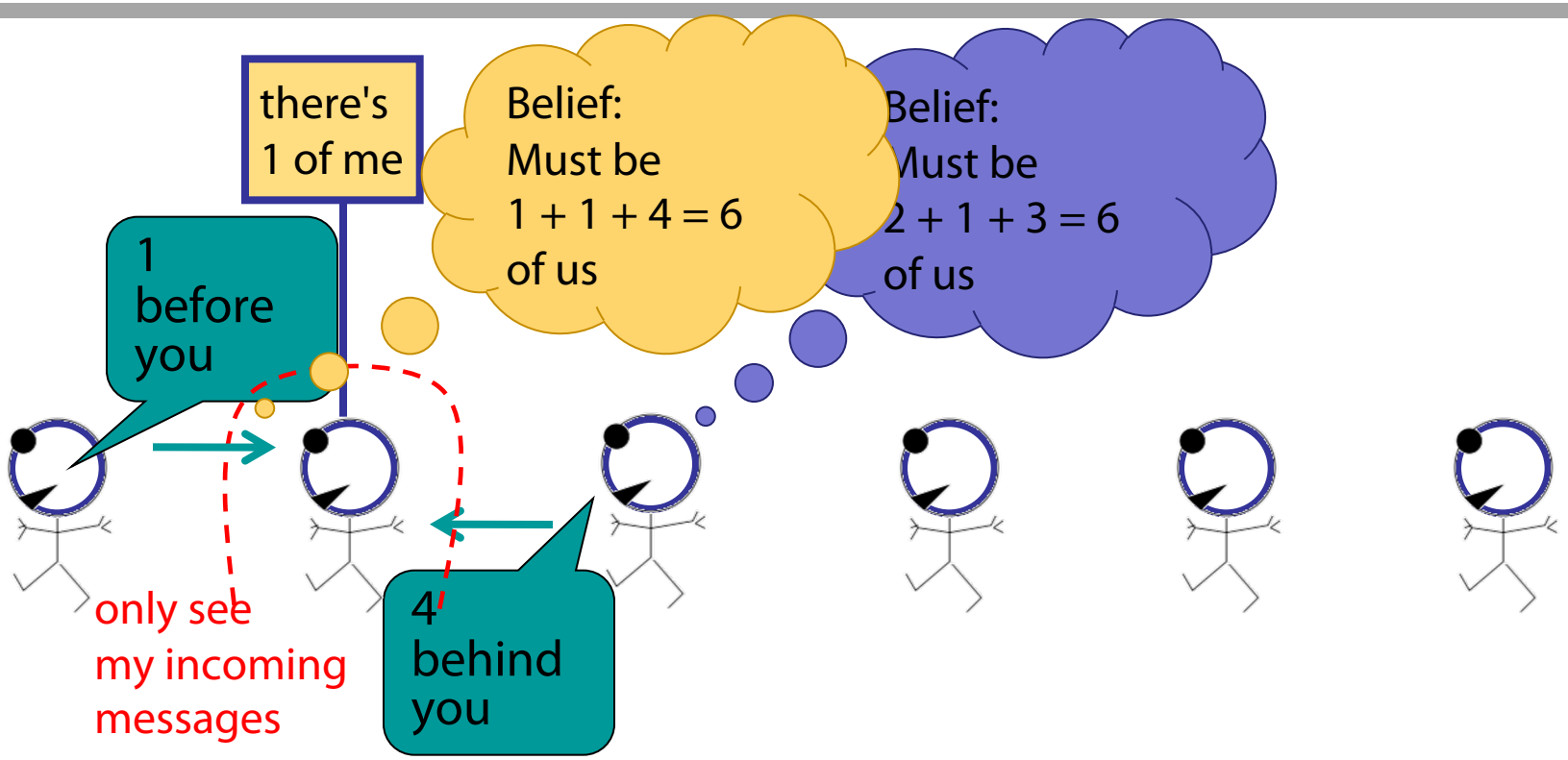


adapted from
MacKay (2003): Information Theory, Inference and Learning Algorithms.

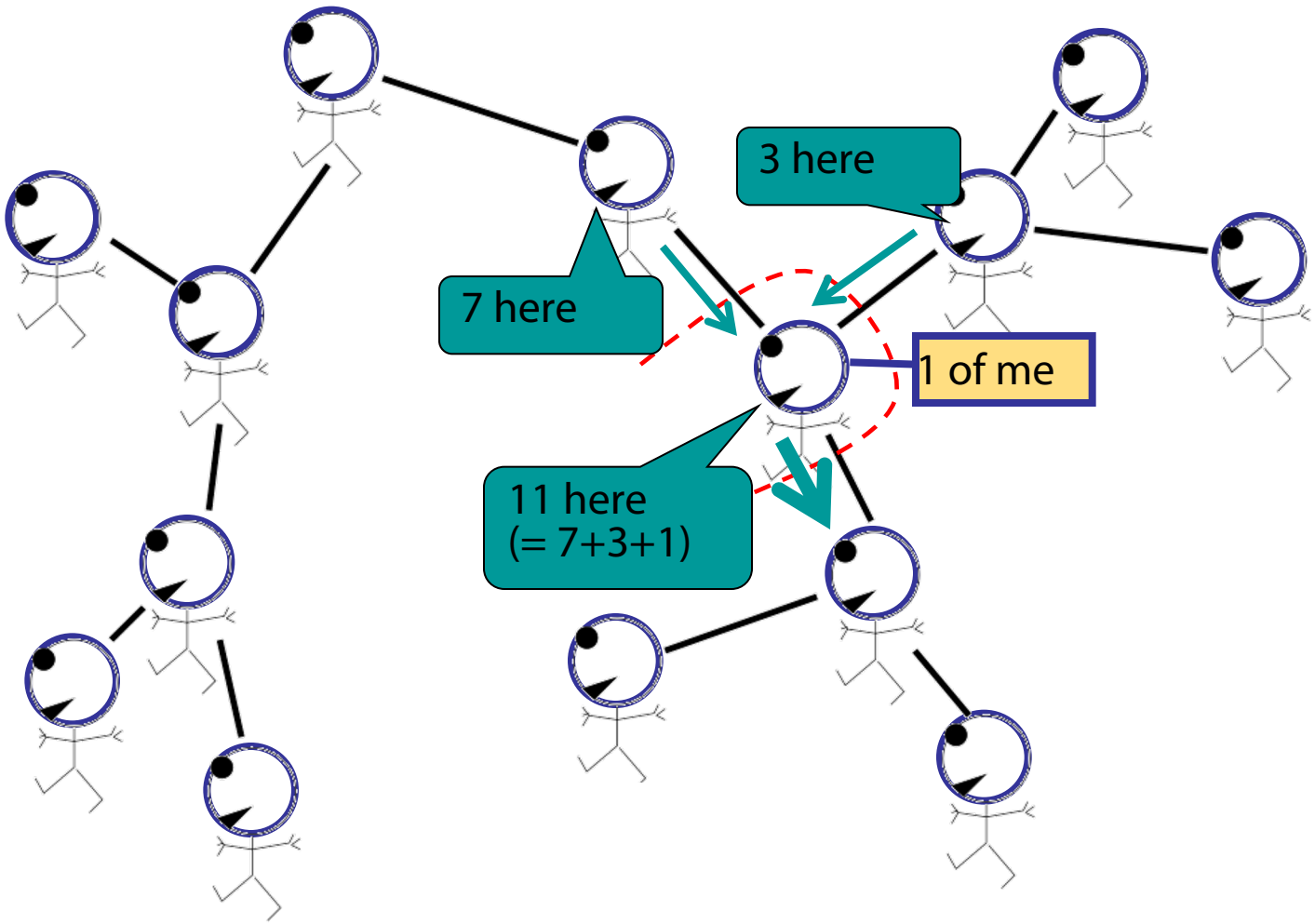
Counting soldiers example (beliefs)



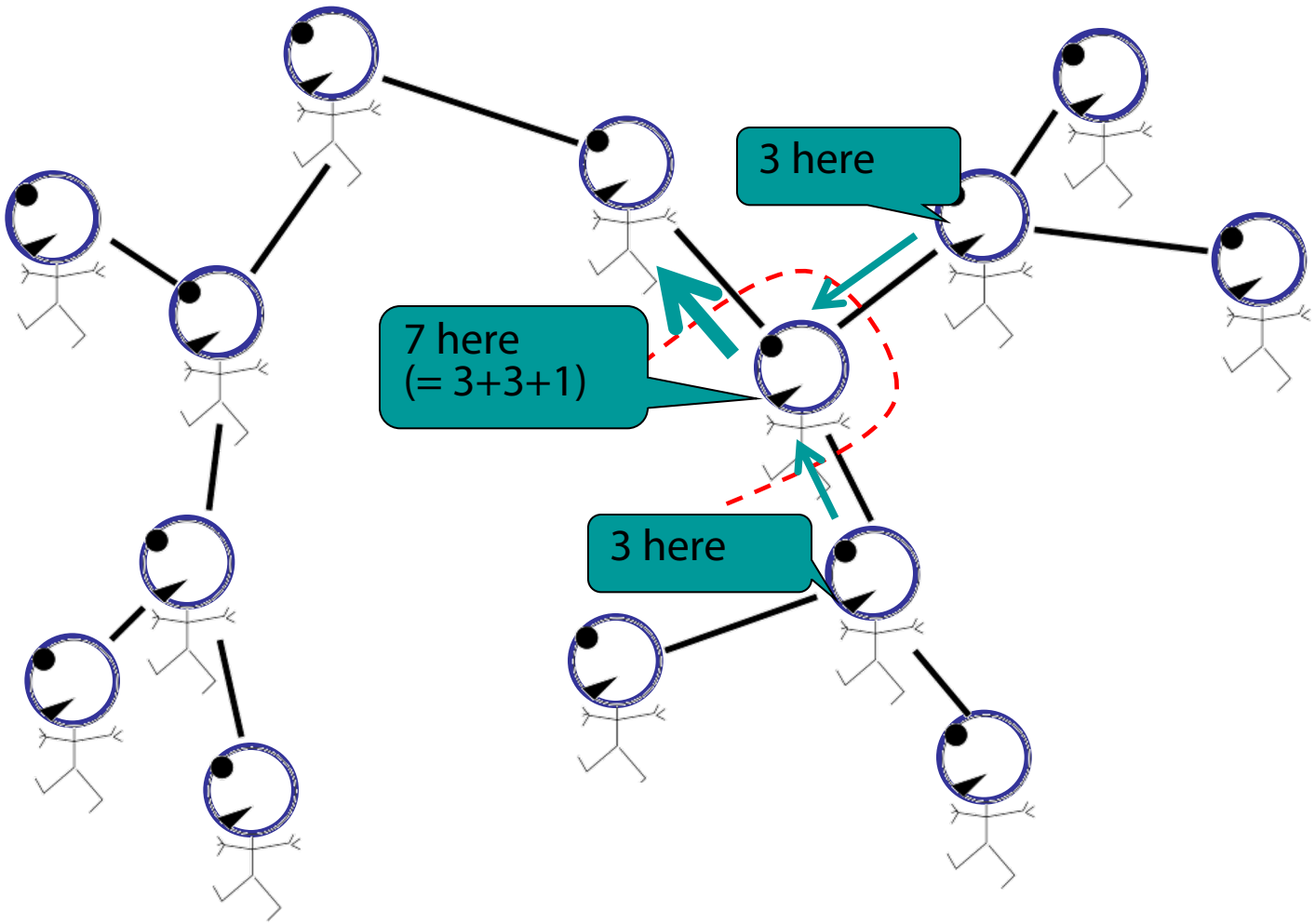
Counting soldiers example (locality)



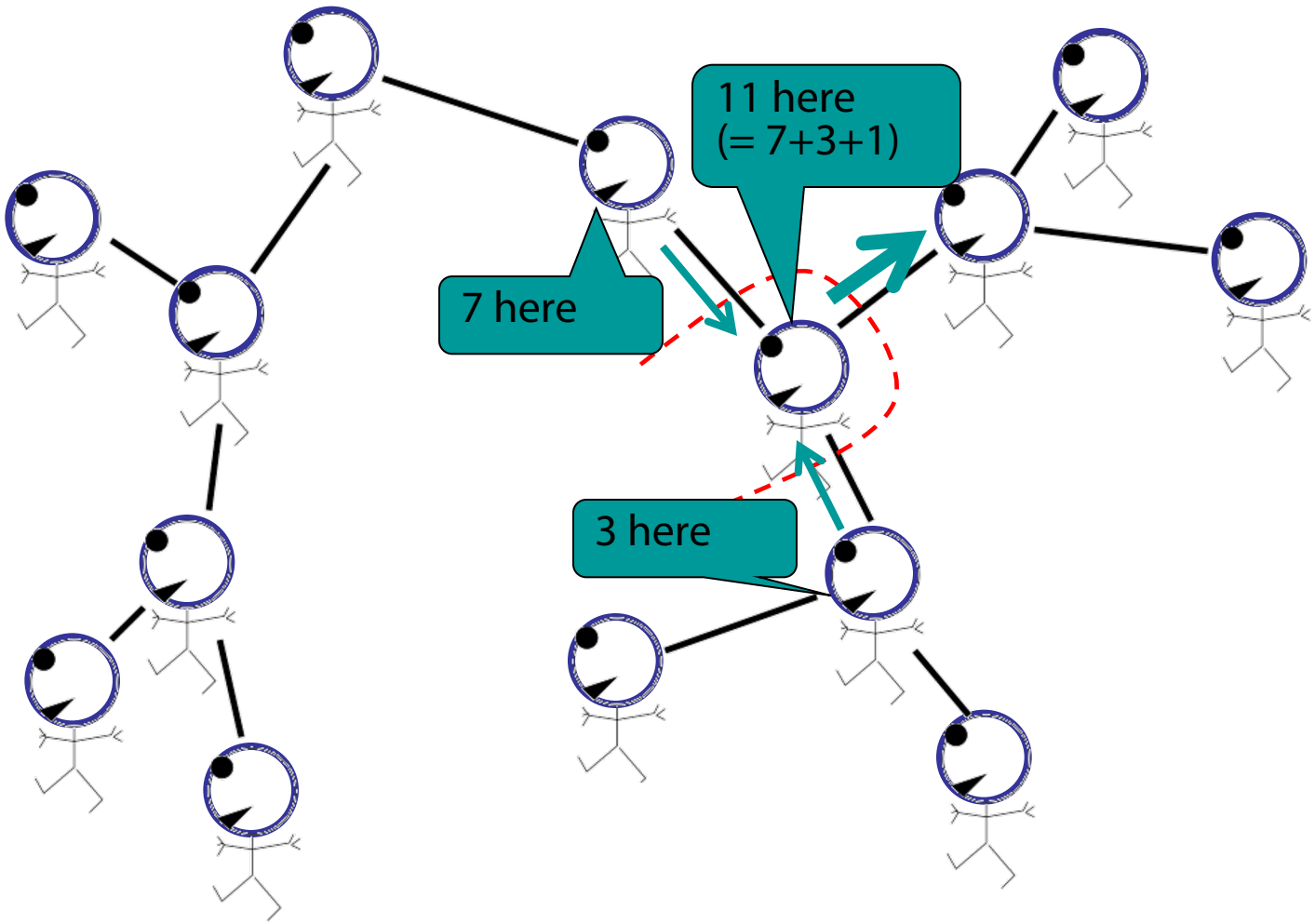
Counting soldiers example (in polytree)



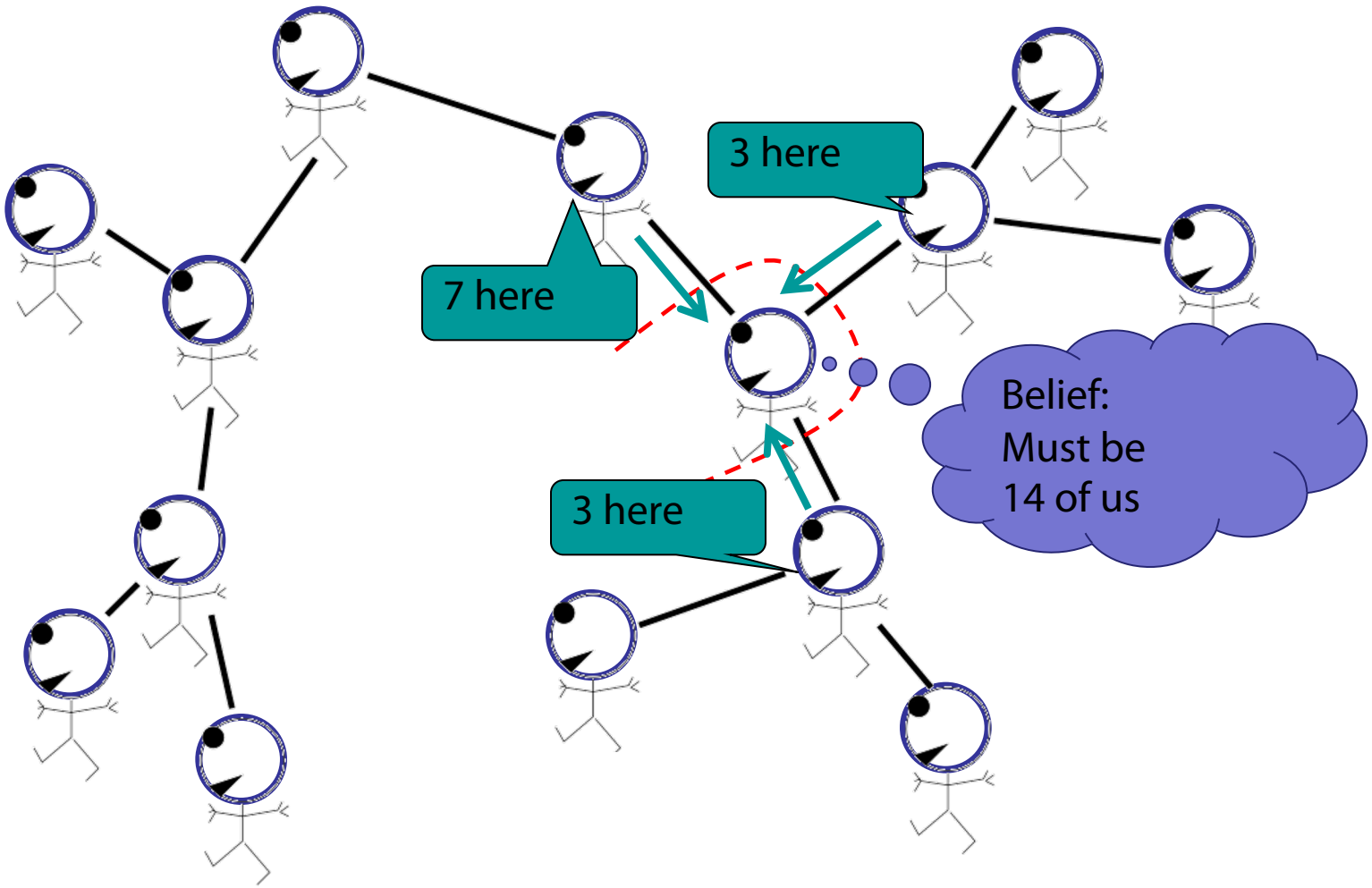
Counting soldiers example (in polytree)



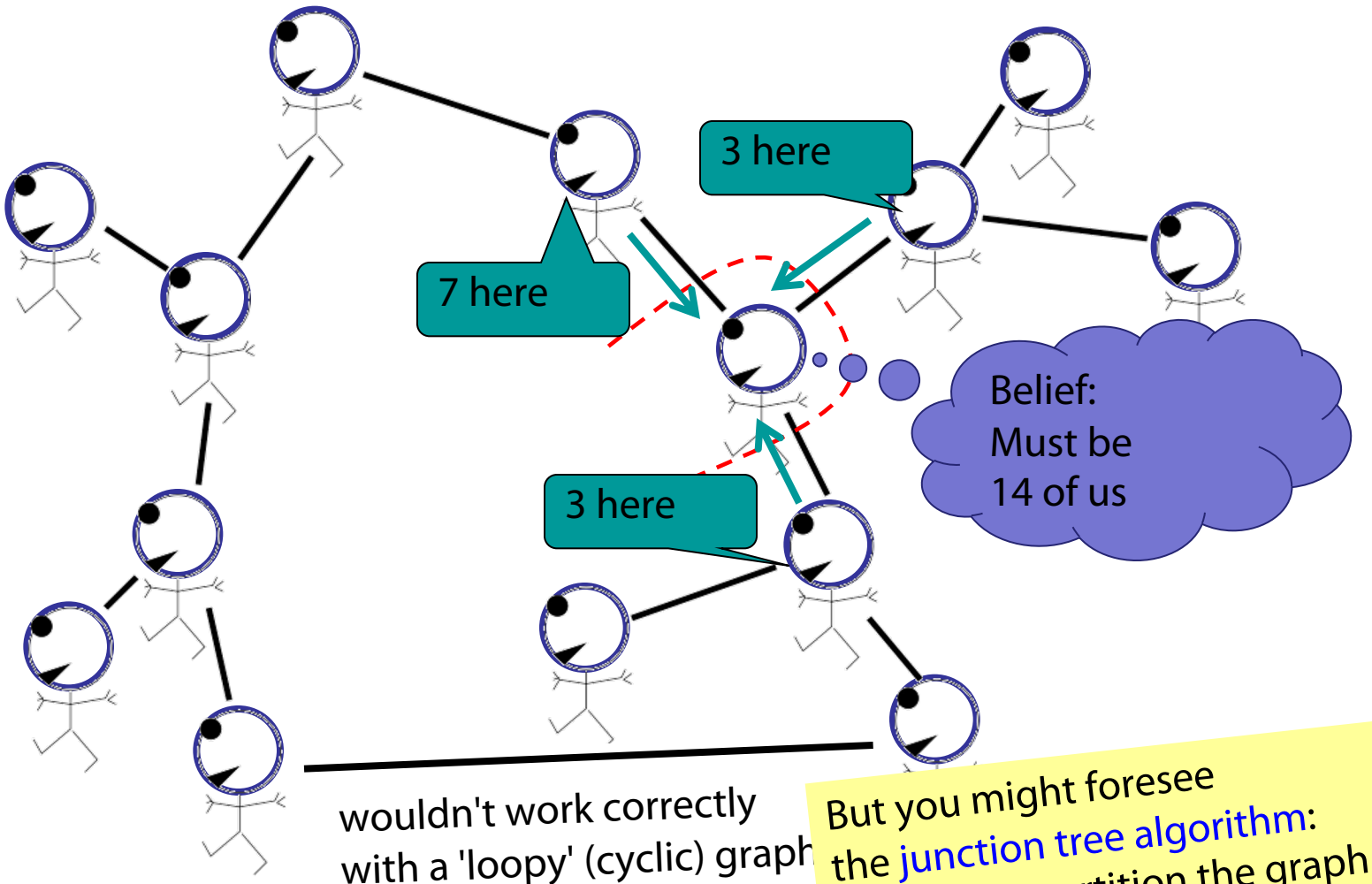
Counting soldiers example (in polytree)



Counting soldiers example (in polytree)



Counting soldiers example (non-circularity)



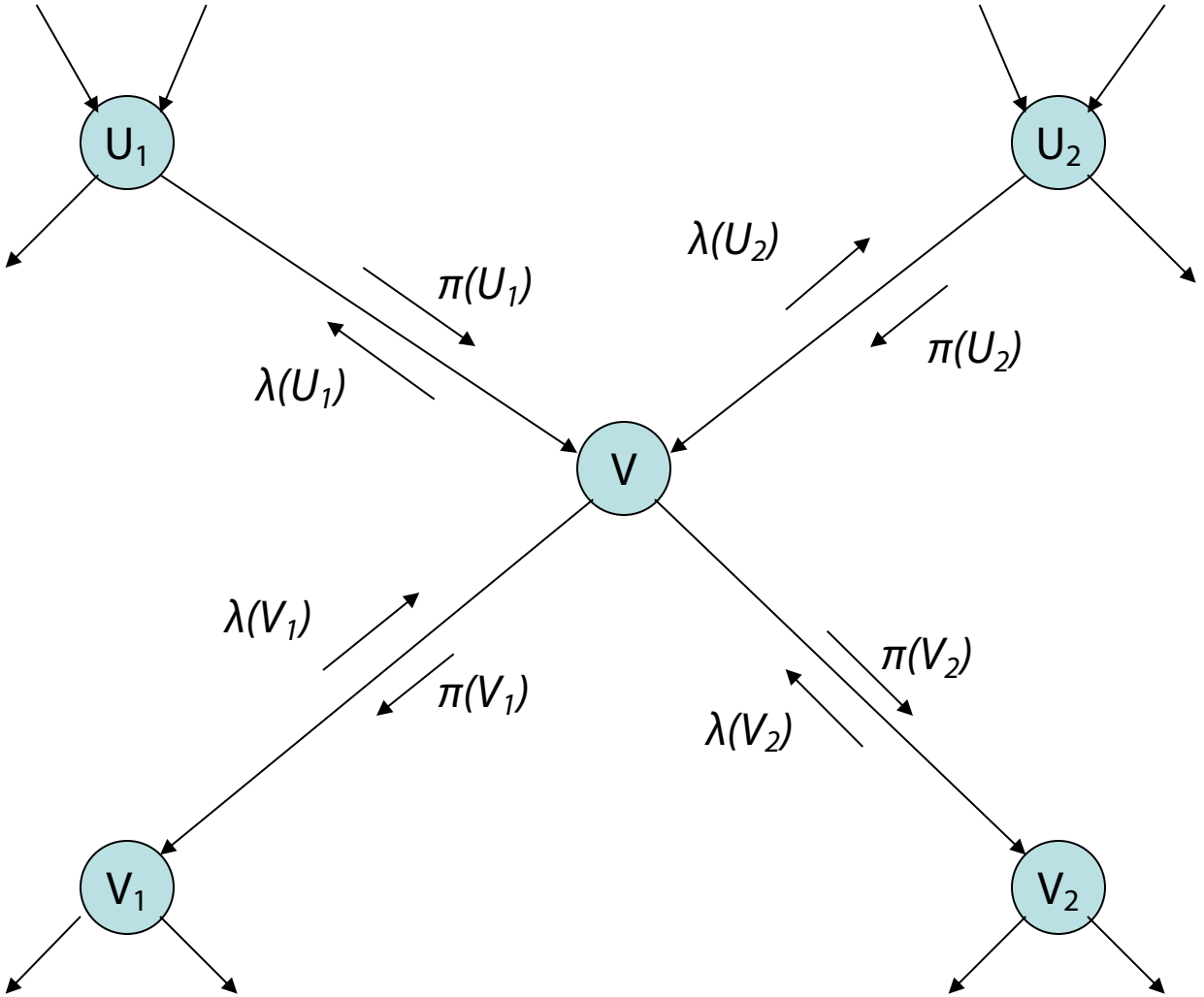
But you might foresee
the junction tree algorithm:
One has to partition the graph
into a tree of clusters

Pearl's belief propagation algorithm

- Local computation for one node V desired
- Information flows through the links of G
  - flows as messages of two types – λ and π
- V splits network into two disjoint parts
  - Strong independence assumptions induced – crucial!

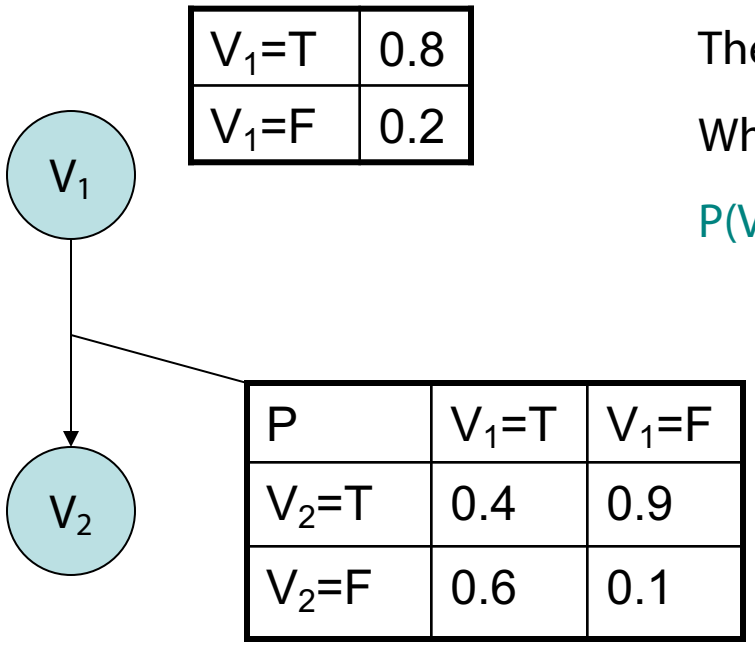
- Denote E_V^+ the part of evidence accessible through the parents of V (causal, **p**redictive, **p**rior)
  - passed downward in π messages
- Analogously, let E_V^- be the diagnostic (or **l**ikelihood) evidence
  - passed upwards in λ messages

Pearl‘s Belief Propagation



The π Messages

- What are the messages?
- For simplicity, let the nodes be Boolean



V_1=T	0.8
V_1=F	0.2

P	V_1=T	V_1=F
V_2=T	0.4	0.9
V_2=F	0.6	0.1

The message passes on information.

What information? Observe:

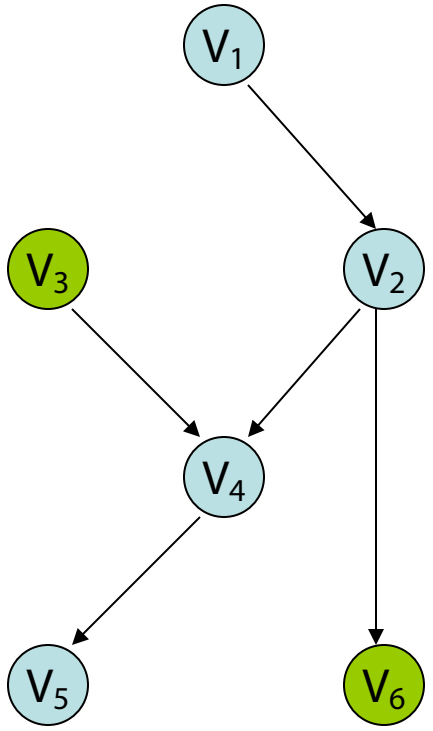
$$P(V_2) = P(V_2|\, V_1{=}T)P(V_1{=}T) + P(V_2|\, V_1{=}F)P(V_1{=}F)$$

The information needed is the CPT of $V_1 = \pi_V(V_1)$

π messages capture information passed from parent to child

The Evidence

- Evidence – values of observed nodes
  - $V_3 = T, V_6 = 3$
- Our belief in what the value of V_i ‘should’ be changed
- This belief is propagated
- As if the CPTs became

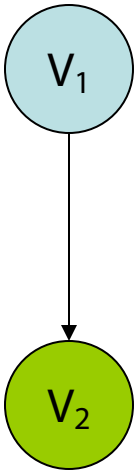


V_3=T	1.0
V_3=F	0.0

P	V_2=T	V_2=F
V_6=1	0.0	0.0
V_6=2	0.0	0.0
V_6=3	1.0	1.0

The Messages

- We know what the π messages are
- What about λ?



Assume $E = \{ V_2 \}$ and compute by Bayes rule:

$$P(V_1 \mid V_2) = \frac{P(V_1)P(V_2 \mid V_1)}{P(V_2)} = \alpha P(V_1)P(V_2 \mid V_1)$$

The information not available at V_1 is $P(V_2|V_1)$. To be passed upwards by a λ-message. Again, this is not in general exactly the CPT, but the belief based on evidence down the tree.

- The messages are $\pi(V)=P(V|E^+)$ and $\lambda(V)=P(E^-|V)$

Combination of evidence in node V

- Recall that $E_V = E_V^+ \cup E_V^-$ and let us compute

$$P(V \mid E) = P(V \mid E_V^+, E_V^-) = \alpha' P(E_V^+, E_V^- \mid V) P(V) =$$
$$\alpha' P(E_V^- \mid V) P(E_V^+ \mid V) P(V) = \alpha P(E_V^- \mid V) P(V \mid E_V^+) =$$
$$\alpha \lambda(V) \pi(V) = BEL(V)$$

- α is the normalization constant ($1/P(E^- \mid E^+)$)
  - normalization is not necessary (can do it at the end)
  - but may prevent numerical underflow problems

λ message combination

- Assume X received λ-messages from neighbors
- How to compute $\lambda(x) = P(e^-|x)$?
- Let $Y_1, \ldots, Y_c$ be the children of X
- $\lambda_{YX}(x)$ denotes the λ-message sent from Y to X

$$\lambda(x) = \prod_{j=1}^{c} \lambda_{Y_j X_i}(x)$$

Derivation:
- Assume two children Y, Z of X; then:
- $P(e^-|x) = P(e^-_Y, e^-_Z|x) = P(e^-_Y|x) * P(e^-_Z|x)$
  (siblings independent given parent)

π message combination

- Assume X received π-messages from neighbors
- How to compute $\pi(x) = P(x|e^+)$?
- Let $U_1, \ldots, U_p$ be the parents of X
- $\pi_{XY}(x)$ denotes the π-message sent from X to Y
- summation over the CPT

$$\pi(x) = \sum_{u_1,\ldots,u_p} P(x \mid u_1,\ldots,u_p) \prod_{j=1}^{p} \pi_{U_j X_i}(u_j)$$

Derivation: Condition on all U_i and note that each pair (parent U_i,evidence e^+_i) is independent of the other pairs (U_j,e^+_j)

Messages to pass

- We need to compute $\pi_{XY}(x)$ for each child Y

$$\pi_{XY_J}(x) = \alpha \pi_X(x) \prod_{k \neq j} \lambda_{Y_k X}(x)$$

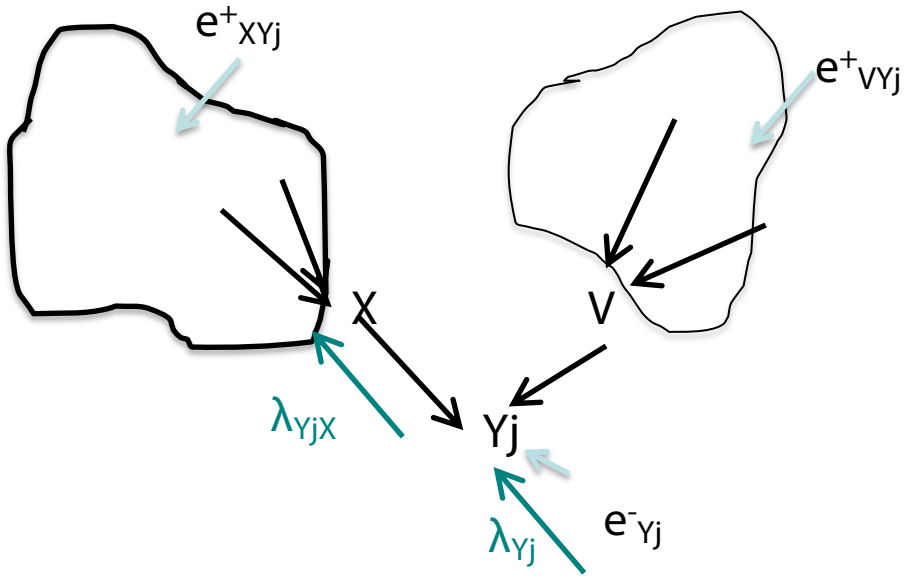
Derivation

π_{XYj}
$= P(x \mid e^{+}_{XYj})$
$= P(x \mid e - e^{-}_{XYj}) =$
= [Bel(x) when evidence e^{-}_{XYj} is surpressed]
= Bel(x) setting $\lambda_{XYj}(x) = 1$
= formula above

e^{+}_{XY} = evidences on tail side (X side) of link X->Y

e^{-}_{XY} = evidences on head side (Y-side) of link X -> Y

Messages to pass



Derivation hint

- $\lambda_{YjX} = P(e^-_{XYj}|x)$
  $= P(e^+_{VYj}, e^-_{Yj}|X)$
- then condition on YJ and V,
- And use independences
  - Y_j separates e^+_{VYj} from e^-_{Yj}
  - V separates e^+_{VYj} from X

- Symbolically, group other ($\neq X$) parents of Y_j into single complex $V = V_1, \ldots, V_q$ and treat link $X \rightarrow Y_j$ vs. $V \rightarrow Y_j$

$$\lambda_{Y_jX}(x) = \sum_{y_j} \lambda_{Y_j}(y_j) \sum_{v_1,\ldots,v_q} p(y \mid v_1,\ldots,v_q) \prod_{k=1}^{q} \pi_{V_kY_j}(v_k)$$

Pearl's Belief Propagation Algorithm

- Initialization step
  - For all $V_i=e_i$ in E:
    - $\lambda(x_i) = 1$ whenever $x_i = e_i$; 0 otherwise
    - $\pi(x_i) = 1$ whenever $x_i = e_i$; 0 otherwise
  - For nodes (not in E) without parents
    - $\pi(x_i) = p(x_i)$ - prior probabilities
  - For nodes without children
    - $\lambda(x_i) = 1$ uniformly (normalize at end)

Pearl's Belief Propagation Algorithm

- Iterate until no change occurs
  - (For each node X) if X has received all the π messages from its parents, calculate π(x)
  - (For each node X) if X has received all the λ messages from its children, calculate λ(x)
  - (For each node X) if π(x) has been calculated and X received all the λ-messages from all its children (except Y), calculate $\pi_{XY}(x)$ and send it to Y.
  - (For each node X) if λ(x) has been calculated and X received all the π-messages from all parents (except U), calculate $\lambda_{XU}(u)$ and send it to U.
- Compute BEL(X) = λ(x)π(x) and normalize

Complexity

- On a polytree, the BP algorithm converges in time proportional to diameter of network – at most linear
- Work done in a node is proportional to the size of CPT
- Hence BP is linear in number of network parameters
- For general BNs
  - Exact inference is NP-hard
  - Approximate inference is NP-hard

- Most BNs are not polytrees. What to do?
  - Clustering (e.g., junction tree) algorithm (somewhat later) to make graph tree like and then use BP
  - Approximate inference (next slides)

Approximate Inference over BNs

Approximation

- We are going to approximate answers to queries such as the posterior $P(X|E)$.

- Here we presume an intuitive notion of approximation:
  - The answer may be erroneous to some extent

- Formally treated in PAC theory (Probably Approximately Correct) by parameters (δ,ε)
  - Confidence (quantified by δ) in that found solution maximally deviates from true solution up to ε

Approximate Inference in Bayesian Networks

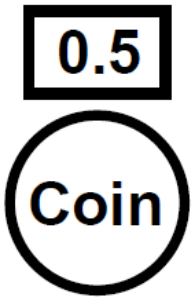
Monte Carlo algorithm

- Widely used to estimate quantities that are difficult to calculate exactly (Remember: for BNs NP-hardness)
- Randomized sampling algorithm
- Accuracy depends on the number of samples
- Two families
  - Direct sampling
  - Markov chain sampling

Inference by stochastic simulation

Basic idea:

1) Draw N samples from a sampling distribution S
2) Compute an approximate posterior probability $\hat{P}$
3) Show this converges to the true probability P



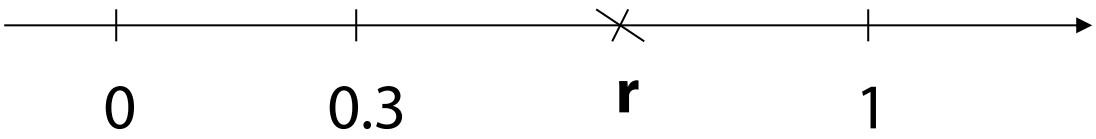
Outline:

- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Likelihood weighting: use evidence to weight samples
- Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Sampling A Value

What does it mean to sample x_i^t from $P(X_i \mid pa_i)$?

- Assume $D(X_i)=\{0,1\}$
- Assume $P(X_i \mid pa_i) = (0.3, 0.7)$

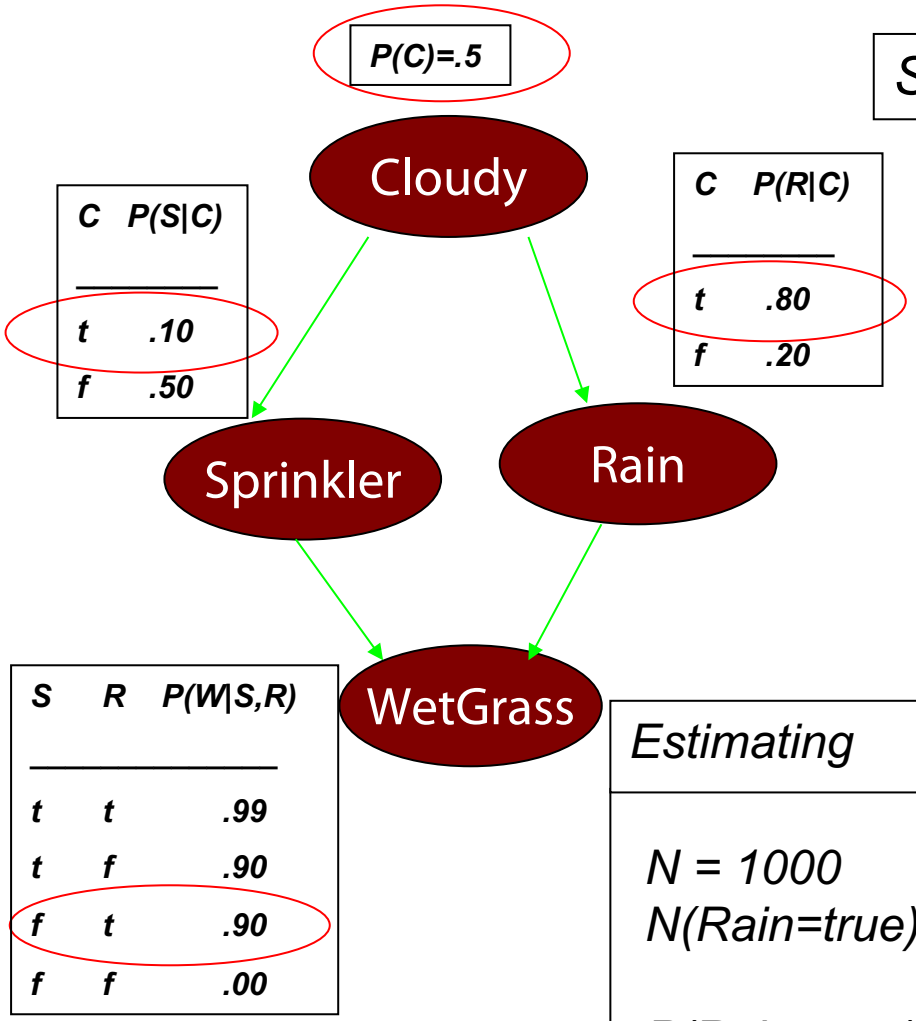


- Draw a random number **r** from [0,1]

  If **r** falls in [0,0.3], set $X_i = 0$

  If **r** falls in [0.3,1], set $X_i=1$

Example in simple case



Sampling

[Cloudy, Sprinkler, Rain, WetGrass]

[true, , ,]

[true, false, ,]

[true, false, true,]

[true, false, true, true]

Estimating

N = 1000 *(# generated samples)*

N(Rain=true) = N([_ , _ , true, _]) = 511

(# of samples for this event)

P(Rain=true) = 0.511 *(approximated probability)*

Sampling from empty network

- Generating samples from a network that has no evidence associated with it (*empty* network)
- Basic idea
  - sample a value for each variable in topological order
  - using the specified conditional probabilities

function PRIOR-SAMPLE(bn) **returns** an event sampled from bn
  inputs: bn, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with n elements
  for $i = 1$ **to** n **do**
    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
      given the values of $Parents(X_i)$ in $\mathbf{x}$
  return $\mathbf{x}$

Properties

Probability that PRIORSAMPLE generates a particular event

$$S_{PS}(x_1 \ldots x_n) = \Pi_{i=1}^{n} P(x_i | parents(X_i)) = P(x_1 \ldots x_n)$$

i.e., the true prior probability

E.g., $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

Let $N_{PS}(x_1 \ldots x_n)$ be the number of samples generated for event $x_1, \ldots, x_n$

Then we have

$$\begin{aligned} \lim_{N \to \infty} \hat{P}(x_1, \ldots, x_n) &= \lim_{N \to \infty} N_{PS}(x_1, \ldots, x_n)/N \\ &= S_{PS}(x_1, \ldots, x_n) \\ &= P(x_1 \ldots x_n) \end{aligned}$$

That is, estimates derived from PRIORSAMPLE are consistent

Shorthand: $\hat{P}(x_1, \ldots, x_n) \approx P(x_1 \ldots x_n)$

What if evidence is given?

- Sampling as defined above would generate cases that cannot be used

Rejection Sampling

- Used to compute conditional probabilities
- Procedure
  - Generating sample from prior distribution specified by the Bayesian Network
  - Rejecting all attempts that do not match the evidence
  - Estimating probability

Rejection Sampling

$\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

function REJECTION-SAMPLING($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{N}$, a vector of counts over X, initially zero

  for $j = 1$ to N **do**
    $\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)
    if $\mathbf{x}$ is consistent with $\mathbf{e}$ **then**
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where x is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}[X]$)

Rejection Sampling Example

- Let us assume we want to estimate P(Rain|Sprinkler = true) with 100 samples

- 100 samples
  - 73 samples => Sprinkler = false
  - 27 samples => Sprinkler = true
    - 8 samples => Rain = true
    - 19 samples => Rain = false

- P(Rain|Sprinkler = true) = NORMALIZE((8,19)) = (0.296,0.704)

- Problem
  - It rejects too many samples

Analysis of rejection sampling

$$\begin{aligned}\hat{\mathbf{P}}(X|\mathbf{e}) &= \alpha \mathbf{N}_{PS}(X,\mathbf{e}) && \text{(algorithm defn.)}\\ &= \mathbf{N}_{PS}(X,\mathbf{e})/N_{PS}(\mathbf{e}) && \text{(normalized by } N_{PS}(\mathbf{e})\text{)}\\ &\approx \mathbf{P}(X,\mathbf{e})/P(\mathbf{e}) && \text{(property of PRIORSAMPLE)}\\ &= \mathbf{P}(X|\mathbf{e}) && \text{(defn. of conditional probability)}\end{aligned}$$

Hence rejection sampling returns consistent posterior estimates

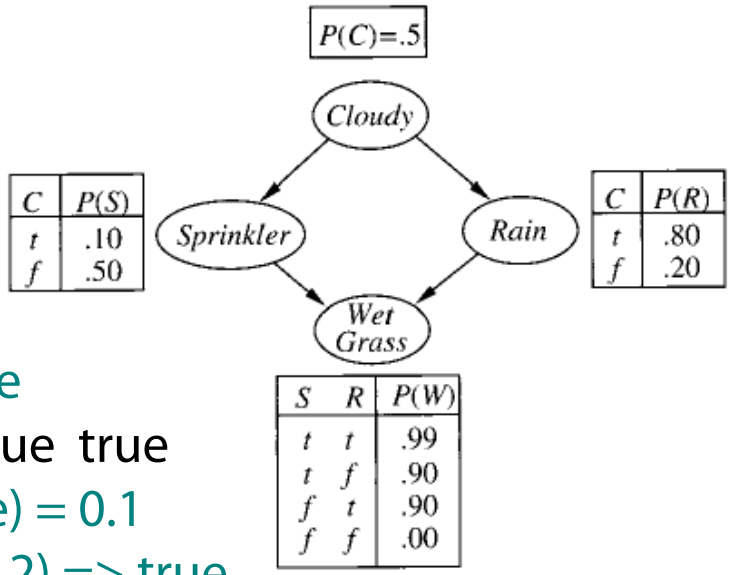
Problem: hopelessly expensive if $P(\mathbf{e})$ is small

$P(\mathbf{e})$ drops off exponentially with number of evidence variables!

Likelihood Weighting

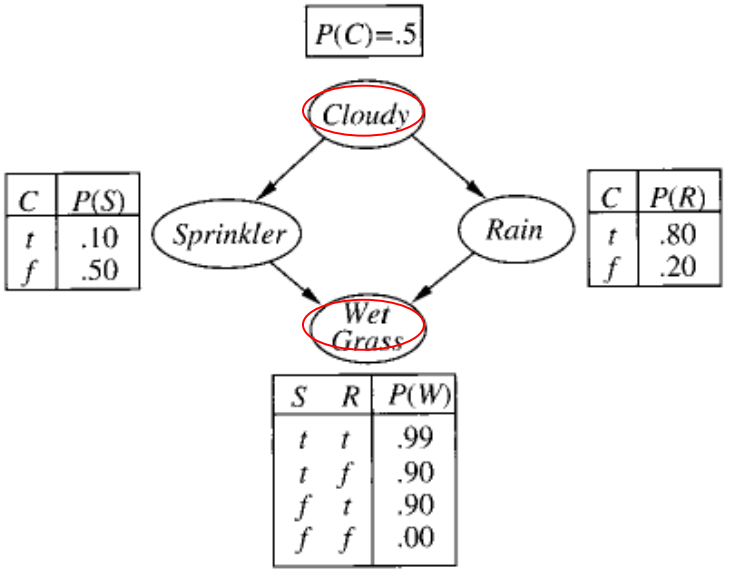
- Goal
  - Avoiding inefficiency of rejection sampling
- Idea
  - Generating only events consistent with evidence
  - Each event is weighted by likelihood that the event accords to the evidence

Likelyhood Weighting: Example



- **P**(Rain|Sprinkler=true, WetGrass = true)?
- Sampling
  - The weight is set to 1.0
  - Sample from **P**(Cloudy) = (0.5,0.5) => true
  - Sprinkler is an evidence variable with value true
    w ← w * P(Sprinkler=true | Cloudy = true) = 0.1
  - Sample from P(Rain|Cloudy=true)=(0.8,0.2) => true
  - WetGrass is an evidence variable with value true
    w ←w * P(WetGrass=true |Sprinkler=true, Rain = true) = 0.099
  - [true, true, true, true] with weight 0.099
- Estimating
  - Accumulating weights to either Rain=true or Rain=false
  - Normalize (= divide by sum of weights)

Likelyhood Weighting: Example



- **P**(Rain|Cloudy=true, WetGrass = true)?
- Sampling
  - Cloudy is an evidence
    w ← w * P(Cloudy = true) = 0.5
  - Sprinkler no evidence
    Sample from P(Sprinkler| Cloudy=true)=(0.1, 0.9) => false
  - Sample from P(Rain|Cloudy=true)=(0.8,0.2) => true
  - WetGrass is an evidence variable with value true
    w ←w * P(WetGrass=true |Sprinkler=false, Rain = true) = 0.45
  - [true, false, true, true] with weight 0.45

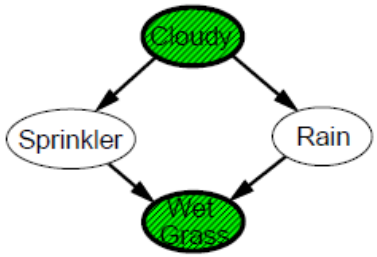
Likelihood analysis

Sampling probability for WeightedSample is

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | parents(Z_i))$$

Note: pays attention to evidence in **ancestors** only

$\Rightarrow$ somewhere "in between" prior and posterior distribution



Weight for a given sample $\mathbf{z}, \mathbf{e}$ is

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | parents(E_i))$$

Weighted sampling probability is

$$S_{WS}(\mathbf{z}, \mathbf{e}) w(\mathbf{z}, \mathbf{e})$$
$$= \prod_{i=1}^{l} P(z_i | parents(Z_i)) \prod_{i=1}^{m} P(e_i | parents(E_i))$$
$$= P(\mathbf{z}, \mathbf{e}) \text{ (by standard global semantics of network)}$$

Hence likelihood weighting returns consistent estimates but performance still degrades with many evidence variables because a few samples have nearly all the total weight

Likelihood weighting

function LIKELIHOOD-WEIGHTING($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$

local variables: $\mathbf{W}$, a vector of weighted counts over X, initially zero

for $j = 1$ to N **do**

$\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE(bn)

$\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in $\mathbf{x}$

return NORMALIZE($\mathbf{W}[X]$)

---

function WEIGHTED-SAMPLE($bn, \mathbf{e}$) **returns** an event and a weight

$\mathbf{x} \leftarrow$ an event with n elements; $w \leftarrow 1$

for $i = 1$ **to** n **do**

if X_i has a value x_i in $\mathbf{e}$

then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$

else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$

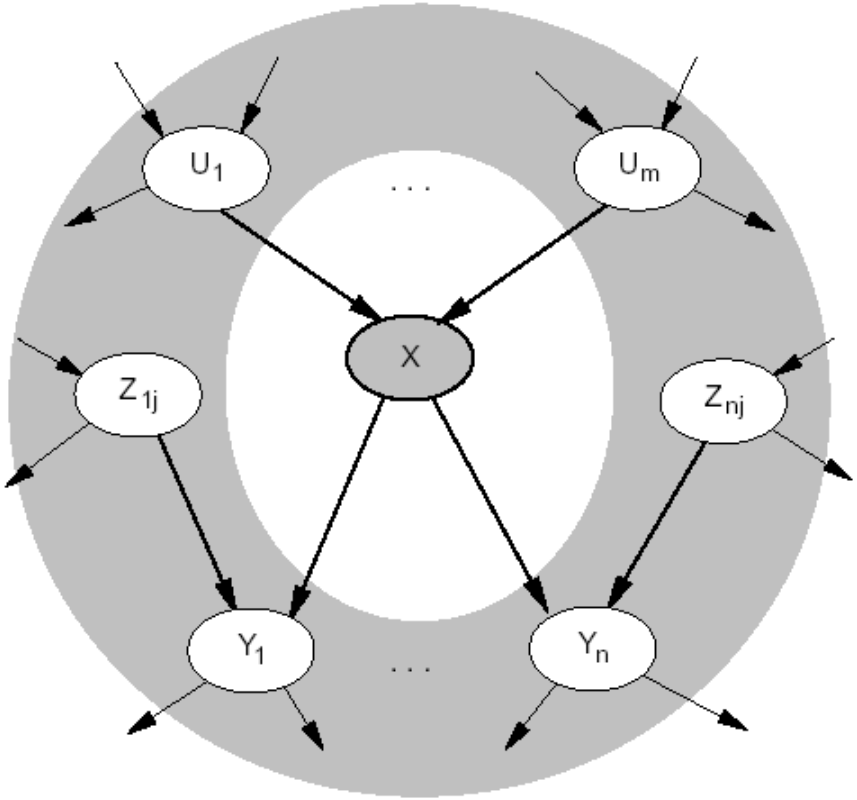
return $\mathbf{x}$, w

Markov Chain Monte Carlo

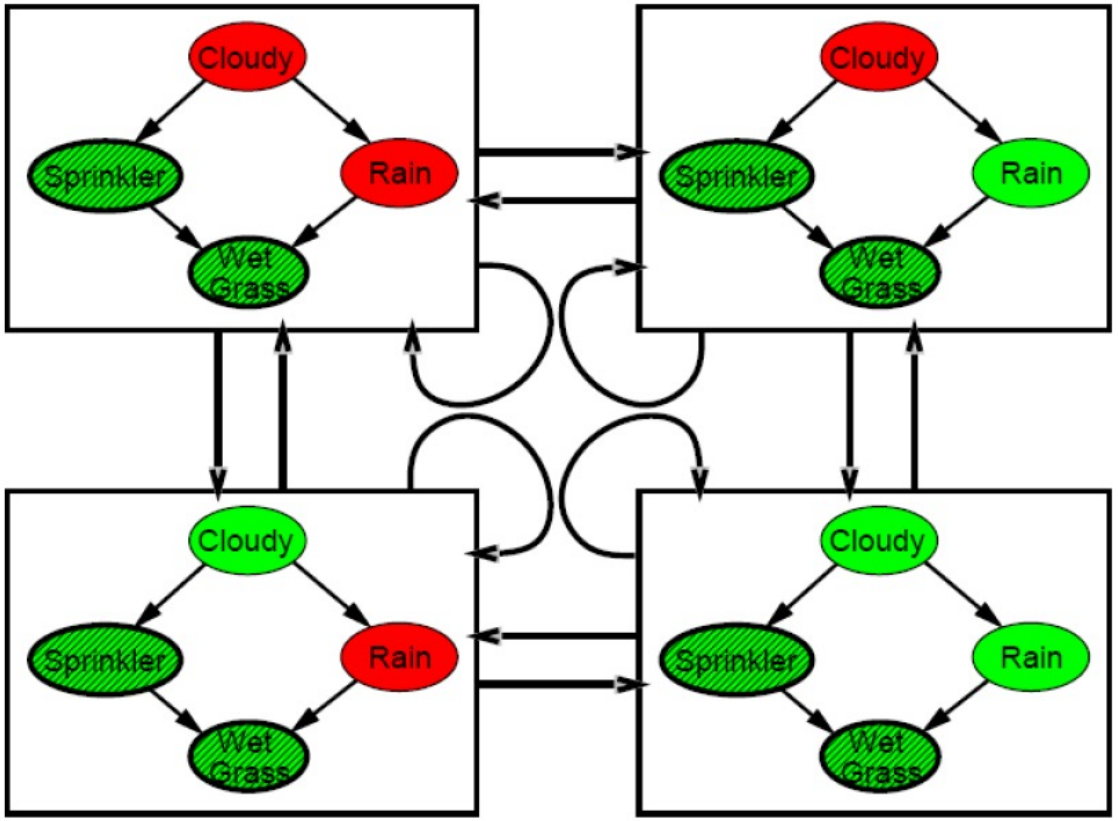
- Let's think of the network as being in a particular current state specifying a value for every variable
- MCMC generates each event by making a random change to the preceding event
- The next state is generated by randomly sampling a value for one of the non-evidence variables X_i, conditioned on the current values of the variables in the Markov blanket of X_i
- Likelihood Weighting only takes into account the evidences of the parents. (Problematic if evidence on leaves).

Markov Blanket

- Markov blanket: Parents + children + children's parents
- Node is conditionally independent of all other nodes in network, given its Markov Blanket



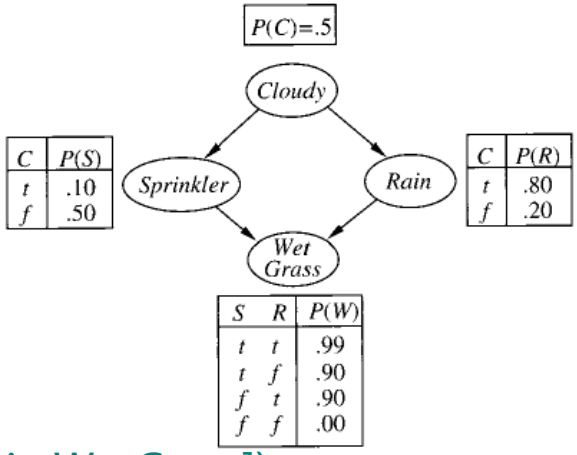
With $Sprinkler = true, WetGrass = true$, there are four states:



Wander about for a while, average what you see

Arrows describe transition probabilities; leads to a (the Markov) chain of states

Markov Chain Monte Carlo: Example



- P(Rain|Sprinkler = true, WetGrass = true) = ?
- Initial state is [true, true, false, true] (for [Cloudy,Sprinkler,Rain,WetGrass])

- The following steps are executed repeatedly:
  - Cloudy is sampled, given the current values of its Markov blanket variables
    So, we sample from P(Cloudy|Sprinkler= true, Rain=false)
    Suppose the result is Cloudy = false.
  - Now current state is [false, true, false, true] and counts are updated
  - Rain is sampled, given the current values of its Markov blanket variables
    So, we sample from P(Rain|Cloudy=false,Sprinkler=true, WetGrass=true)
    Suppose the result is Rain = true.
  - Then current state is [false, true, true, true]
- After all the iterations, let's say the process visited 20 states where Rain is true and 60 states where Rain is false then the answer of the query is NORMALIZE((20,60))=(0.25,0.75)

MCMC

"State" of network = current assignment to all variables.

Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

function MCMC-ASK($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{N}[X]$, a vector of counts over X, initially zero
    $\mathbf{Z}$, the nonevidence variables in bn
    $\mathbf{s}$, the current state of the network, initially copied from $\mathbf{e}$
  initialize $\mathbf{s}$ with random values for the variables in $\mathbf{Z}$
  for $j = 1$ to N **do**
    for each Z_i in $\mathbf{Z}$ **do**
      sample the value of Z_i in $\mathbf{s}$ from $\mathbf{P}(Z_i|mb(Z_i))$
        given the values of $MB(Z_i)$ in $\mathbf{s}$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{s}$
  return NORMALIZE($\mathbf{N}[X]$)

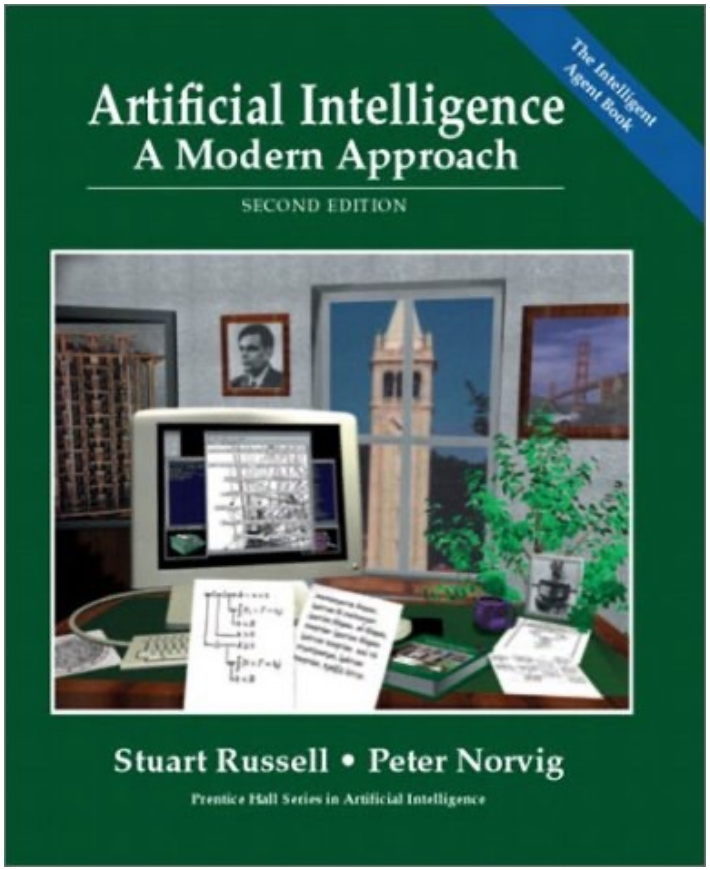
Can also choose a variable to sample at random each time

Summary

- Bayesian networks provide a natural representation for (causally induced) conditional independence
- Topology + CPTs = compact representation of joint distribution
- Generally easy for domain experts to construct
- Exact inference by variable elimination
  - polytime on polytrees, NP-hard on general graphs
  - space can be exponential as well
- Exact inference by belief propagation on polytrees
- Approximate inference based on sampling and counting help to overcome complexity of exact inference

Acknowledgements

- Slides from AIMA book provided by Cristina Conati, UBC



Data Mining Bayesian Networks

- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
  - Fully observable
  - With hidden (unobservable) variables

Full Bayesian Learning

- In the learning methods we have seen so far, the idea was always to find the best model that could explain some observations
- In contrast, full Bayesian learning sees learning as Bayesian updating of a probability distribution over the hypothesis space, given data
  - H is the hypothesis variable
  - Possible hypotheses (values of H) $h_1 \ldots, h_n$
  - $P(H)$ = prior probability distribution over hypothesis space
- j_{th} observation d_j gives the outcome of random variable D_j
  - training data $\mathbf{d} = d_1,..,d_k$

Example

- Suppose we have 5 types of candy bags
  - 10% are 100% cherry candies (h_{100})
  - 20% are 75% cherry + 25% lime candies (h_{75})
  - 40% are 50% cherry + 50% lime candies (h_{50})
  - 20% are 25% cherry + 75% lime candies (h_{25})
  - 10% are 100% lime candies (h_0)

- Then we observe candies drawn from some bag 

- Let's call θ the parameter that defines the fraction of cherry candy in a bag, and h_θ the corresponding hypothesis

- Which of the five kinds of bag has generated my 10 observations? $P(h_\theta | \mathbf{d})$.

- What flavour will the next candy be? Prediction $\mathbf{P}(X|\mathbf{d})$

Full Bayesian Learning

- Given the data so far, each hypothesis h_i has a posterior probability:
  - $P(h_i \mid \mathbf{d}) = \alpha P(\mathbf{d} \mid h_i)\, P(h_i)$ (**Bayes theorem**)
  - where $P(\mathbf{d} \mid h_i)$ is called the likelihood of the data under each hypothesis
- Predictions over a new entity X are a weighted average over the prediction of each hypothesis:
  - $\mathbf{P}(X \mid \mathbf{d}) =$
    $= \sum_i \mathbf{P}(X, h_i \mid \mathbf{d})$
    $= \sum_i \mathbf{P}(X \mid h_i, \mathbf{d})\, P(h_i \mid \mathbf{d})$
    $= \sum_i \mathbf{P}(X \mid h_i)\, P(h_i \mid \mathbf{d})$
    $\sim \sum_i \mathbf{P}(X \mid h_i)\, P(\mathbf{d} \mid h_i)\, P(h_i)$
  - The weights are given by the data likelihood and prior of each h

The data does not add anything to a prediction given a hypothesis

+ No need to pick one best-guess hypothesis!

– Need to iterate over all hypotheses

Man findet $\propto$ oder $\sim$ oder $= \alpha \cdot \ldots$

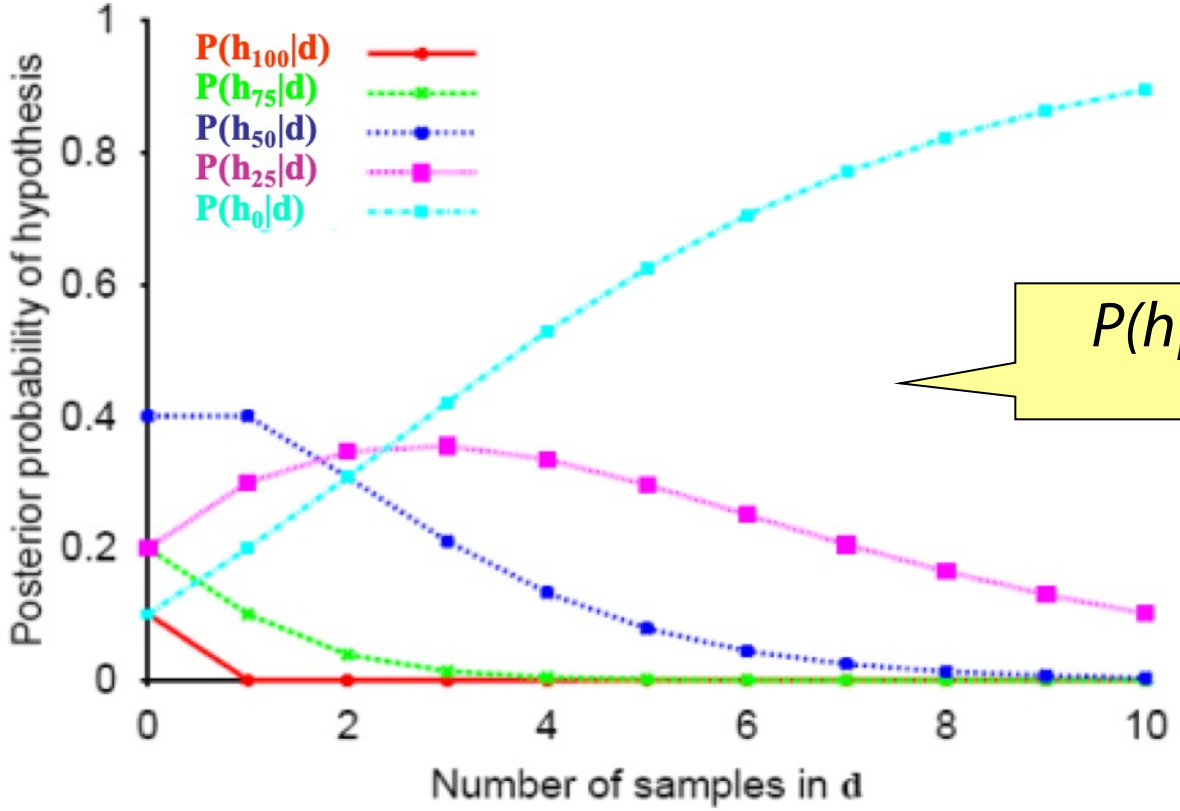
Example

- If we re-wrap each candy and return it to the bag, our 10 observations are independent and identically distributed, i.i.d, so
  - $P(\mathbf{d}|\, h_\theta) = \prod_j P(d_j|\, h_\theta)$ for $j=1,..,10$
- For a given h_θ , the value of $P(d_j|\, h_\theta)$ is
  - $P(d_j = \text{cherry}|\, h_\theta) = \theta$; $P(d_j = \text{lime}|h_\theta) = (1-\theta)$
- And given N observations, of which c are cherry and l = N-c lime

$$P(\boldsymbol{d} \mid h_\theta) = \prod_{j=1}^{c} \theta \prod_{j=1}^{\ell} (1-\theta) = \theta^c (1-\theta)^\ell$$

  - **Binomial distribution**: probability of # of successes in a sequence of N independent trials with binary outcome, each of which yields success with probability θ.
- For instance, after observing 3 lime candies in a row:
  - $P([\text{lime, lime, lime}] \mid h_{50}) = 0.5^3$ because the probability of seeing lime for each observation is 0.5 under this hypotheses

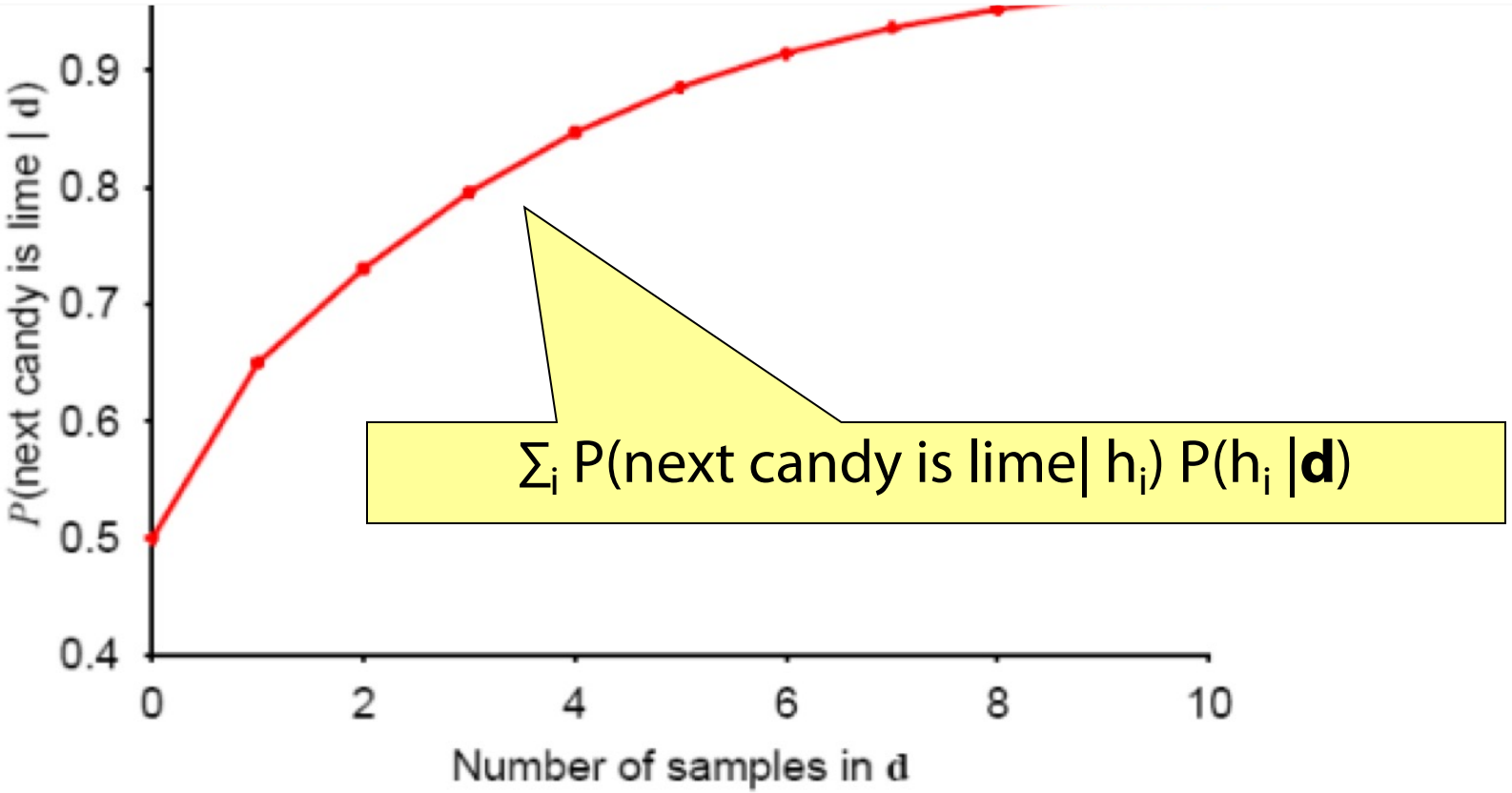
All-limes: Posterior Probability of H



$$P(h_i \mid \mathbf{d}) = \alpha P(\mathbf{d} \mid h_i)\, P(h_i)$$

- Initially, the h_p with higher priors dominate (h_{50} with prior = 0.4)
- As data comes in, the finally best hypothesis (h_0) starts dominating, as the probability of seeing this data given the other hypotheses gets increasingly smaller
  - After seeing three lime candies in a row, the probability that the bag is the all-lime one starts taking off

Prediction Probability



- The probability that the next candy is lime increases with the probability that the bag is an all-lime one

Overview

- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
  - Fully observable
  - With hidden (unobservable) variables

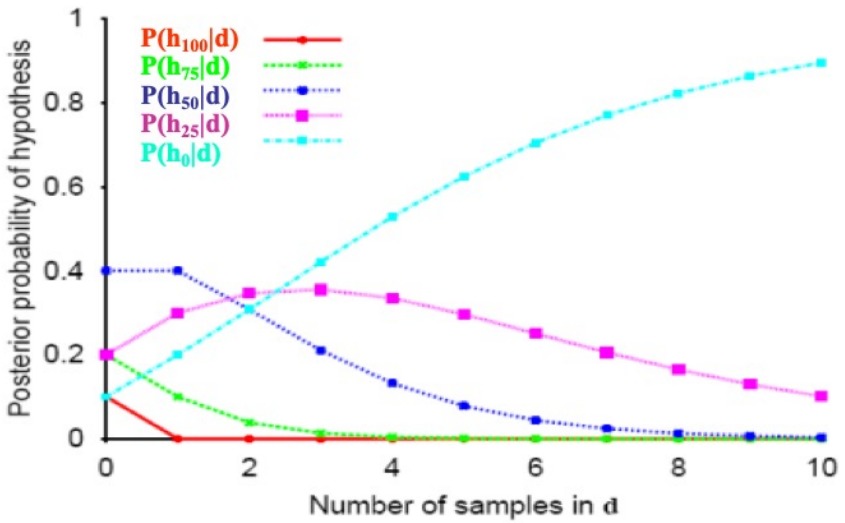
MAP approximation

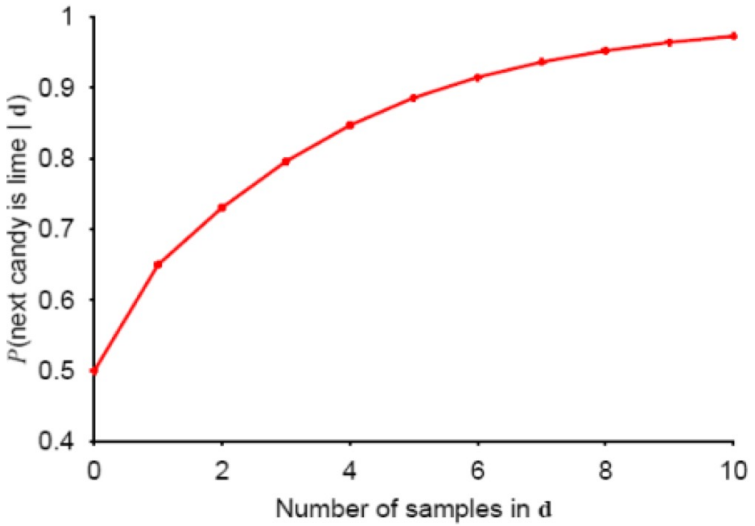
- Full Bayesian learning seems like a very safe bet, but unfortunately it does not work well in practice
  - Summing over the hypothesis space is often intractable (e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)
- Very common approximation: Maximum a posterior (MAP) learning:
  - Instead of doing prediction by considering all possible hypotheses , as in
    - $\mathbf{P}(X|\mathbf{d}) = \sum_i \mathbf{P}(X|h_i)\, P(h_i|\mathbf{d}) \propto \sum_i \mathbf{P}(X|h_i)\, P(\mathbf{d}|h_i)\, P(h_i)$
  - Make predictions based on h_{MAP} that maximises $P(h_i|\mathbf{d})$
    - I.e., maximize $P(\mathbf{d}|h_i)\, P(h_i)$
    - $\mathbf{P}(X|\mathbf{d}) \approx \mathbf{P}(X|h_{MAP})$

Man findet $\propto$ oder $\sim$ oder $= \alpha \cdot \ldots$

MAP approximation

- MAP is a good approximation when $P(X \mid \mathbf{d}) \approx P(X \mid h_{MAP})$
  - In our example, h_{MAP} is the all-lime bag after only 3 candies, predicting that the next candy will be lime with $p = 1$
  - The Bayesian learner gave a prediction of 0.8, safer after seeing only 3 candies





Bias

- As more data arrive, MAP and Bayesian prediction become closer, as MAP's competing hypotheses become less likely
- Often easier to find MAP (optimization problem) than deal with a large summation problem
- *P(H)* plays an important role in both MAP and Full Bayesian Learning (defines learning bias)
- Used to define a tradeoff between model complexity and its ability to fit the data
  - More complex models can explain the data better => higher $P(\mathbf{d}| h_i)$ **danger of overfitting**
  - But they are less likely a priory because there are more of them than simpler model => lower $P(h_i)$
  - I.e. common learning bias is to penalize complexity

Overview

- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
  - Fully observable
  - With hidden (unobservable) variables

Maximum Likelihood (ML) Learning

- Further simplification over full Bayesian and MAP learning
  - Assume uniform priors over the space of hypotheses
  - MAP learning (maximize $P(\mathbf{d}|h_i)\,P(h_i)$) reduces to maximize $P(\mathbf{d}|h_i)$
- When is ML appropriate?

Howard, R.: Decision analysis: Perspectives on inference, decision, and experimentation. Proceedings of the IEEE 58(5), 632-643, **1970**

Maximum Likelihood (ML) Learning

- Further simplification over Full Bayesian and MAP learning
  - Assume uniform prior over the space of hypotheses
  - MAP learning (maximize $P(\mathbf{d}|h_i)\,P(h_i)$) reduces to maximize $P(\mathbf{d}|h_i)$
- When is ML appropriate?
  - Used in statistics as the standard (non-bayesian) statistical learning method by those who distrust subjective nature of hypotheses priors
  - When the competing hypotheses are indeed equally likely (e.g. have same complexity)
  - With very large datasets, for which $P(\mathbf{d}|h_i)$ tends to overcome the influence of $P(h_i)$

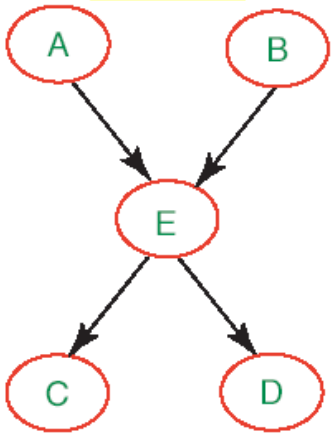
Overview

- Full Bayesian Learning
- MAP learning
- Maximum Likelihood Learning
- Learning Bayesian Networks
  - Fully observable (complete data)
  - With hidden (unobservable) variables

Learning BNs: Data Science w/ Complete Data

- We will start by applying ML to the simplest type of BNets learning:
  - known structure
  - Data containing observations for all variables
    - ✓ All variables are observable, no missing data
- The only thing that we need to learn are the network's parameters

Model → Data → Probabilities



A	B	C	D	E
t	f	t	t	f
f	t	t	t	t
t	t	f	t	f
		...		

$P(A)$
$P(B)$
$P(E|A, B)$
$P(C|E)$
$P(D|E)$

Maximum-Likelihood Estimation (MLE)

$$\underset{\theta}{\text{argmax}}\ P(D \mid \theta) = \prod_{m} P(x[m] \mid \theta)$$

- Maximize log P(D | θ)
- Compute the derivative of every summand
- Compute roots, resolve equation to each parameter
- Determine frequencies according to graph structure and CPT construction

Extension to MAP

If we replace the likelihood in the MLE formula above with the posterior, we get:

$$\theta_{MAP} = \arg\max_{\theta} P(X|\theta)P(\theta)$$

$$= \arg\max_{\theta} \log P(X|\theta) + \log P(\theta)$$

$$= \arg\max_{\theta} \log \prod_{i} P(x_i|\theta) + \log P(\theta)$$

$$= \arg\max_{\theta} \sum_{i} \log P(x_i|\theta) + \log P(\theta)$$

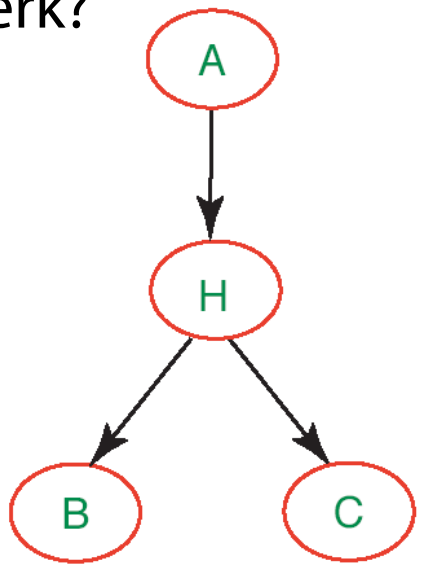
Comparing both MLE and MAP equation, the only thing differs is the inclusion of prior $P(\theta)$ in MAP, otherwise they are identical. What it means is that, the likelihood is now weighted with some weight coming from the prior.

Überblick

- Volles Bayessches Lernen (BMA)
- MAP-Lernen
- Maximum-Likelihood-Lernen
  - Vollständig beobachtbar (vollständige Daten)
  - Mit versteckten (unbeobachtbaren) Variablen

Parameterlernen mit versteckten Variablen

- Bisher haben wir angenommen, Daten für jede Variable des BNs stehen zur Verfügung
- Was machen wir, wenn das nicht der Fall ist, d.h. es gibt versteckte (hidden) Variablen im Netzwerk?

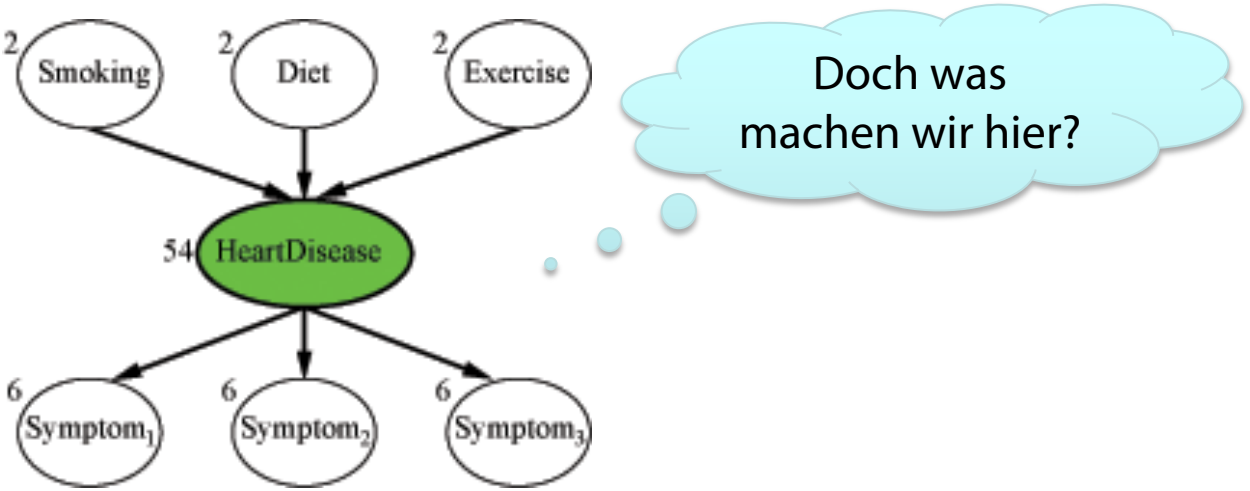


A	B	C
t	f	t
f	t	t
t	t	f
	...	

- Den Ansatz mit relativen Häufigkeiten können wir nicht so einfach übernehmen (keine Zähler für Variable H bekannt)

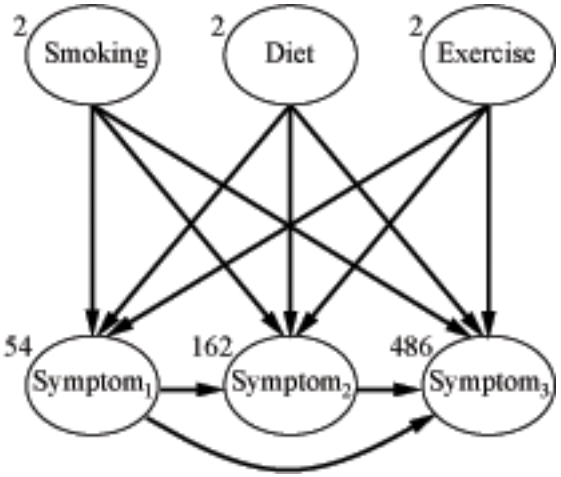
Vermeintliche Lösung

- Vermeide versteckte Variablen
- Könnte klappen bei kleinen Netzwerken



- Jede Variable habe 3 Werte (low, moderate, high)
- Die Zahlen an den Knoten repräsentieren, wie viele Parameter für die CPT des betreffenden Knotens bestimmt werden müssen
- 78 Wahrscheinlichkeitswerte insgesamt

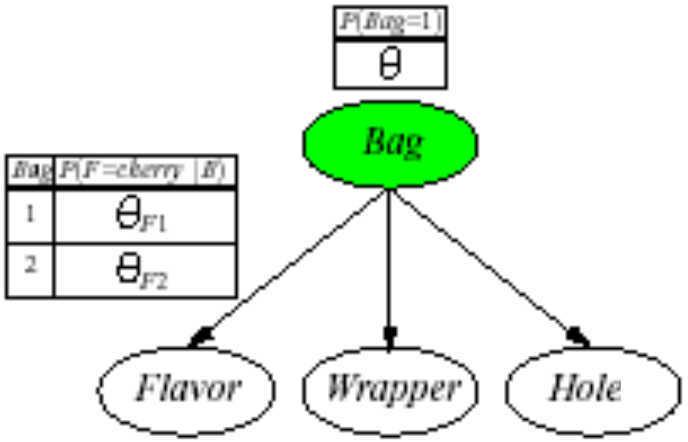
HeartDisease wegzulassen ist gar keine gute Lösung!



- Die Symptome sind nicht länger bedingt unabhängig gegeben die Elternknotenwerte
  - Sehr viel mehr Kanten, sehr viel mehr Wahrscheinlichkeiten zu spezifizieren: 708 insgesamt
  - Wir brauchen sehr viel mehr Daten, um diese ganzen Werte vernünftig zu lernen

Beispiel: Bonbonfabrik wieder einmal

- Zwei Bonbontüten (1 and 2) wurden vermischt
- Bonbons werden durch drei Eigenschaften beschrieben: Flavor und Wrapper wie vorher, plus Hole (ob ein Loch in der Mitte ist)
- Die Merkmale von Bonbons hängen mit bestimmten Wahrscheinlichkeiten von der Tüte ab, aus der sie kommen
- Wir wollen für jedes Bonbon vorhersagen, aus welcher Tüte es kam, je nach vorgefundenen Eigenschaften: Naïve-Bayes-Modell



$\theta = P(Bag = 1)$

$\theta_{Fj} = P(Flavor = cherry | Bag = j)$

$\theta_{Wj} = P(Wrapper = red | Bag = j)$

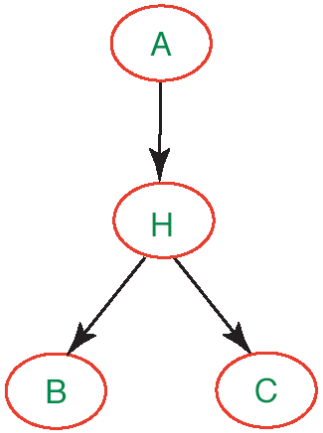
$\theta_{Hj} = P(Hole = yes | Bag = j)$

$j = 1,2$

Expectation-Maximization (EM)

- Wenn wir versteckte Variablen beibehalten und die Netzwerkparameter aus Daten lernen wollen, haben wir eine Form des unüberwachten Lernens (*unsupervised learning*)
  - Die Daten geben nicht über alle Aspekte von Datenpunkten Auskunft
- Expectation-Maximization
  - Genereller Algorithmus zum Lernen von Modellparametern bei unvollständigen Daten
  - Hier für diskrete Datenwerte im BN-Kontext gezeigt

EM: Generelle Idee



A	B	C
t	f	t
f	t	t
t	t	f
	...	

- Falls wir Daten für alle Variablen im BN hätten, könnten wir die Parameter mit ML- (oder MAP-) Ansätzen lernen
  - Relative Häufigkeiten bestimmen, wie vorher besprochen
- Falls wir die Parameter hätten, könnten wir die Posterior-Wahrscheinlichkeiten jedes Ereignisses schätzen:

$P(H|A,B,C)$

Dempster, A.P., Laird. N.M., Rubin, D.B.: *Maximum-Likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society, **1977**

EM: Generelle Idee

- Der Algorithmus startet mit "erfundenen" (zufällig generierten) Informationen, um das Lernproblem zu lösen:
  - Erfundene Netzwerkparameterwerte (virtuell notiert in den CPTs)
- Dann werden die Initialwerte in zwei Schritten verfeinert:
  - Expectation (E): Aktualisiere die Daten (virtuell) mit aus dem aktuellen Modell hergeleiteten Erwartungen
  - Maximization (M): Gegeben die aktualisierten Daten, aktualisieren die Parameter des BNs mitteln Maximum Likelihood (ML) Ansatz
    - ✓ Das ist der gleiche Schritt, wie im voll beobachtbaren Fall

CPT: Conditional Probability Table

EM: Wie funktioniert das mit Naive Bayes-Modelen

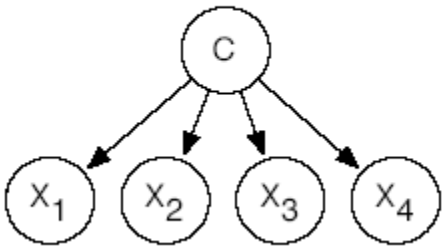
- Betrachtet die folgenden Daten, …
  - N Beispiele mit Booleschen Attributen X_1, X_2, X_3, X_4

Data

X_1	X_2	X_3	X_4
t	f	t	t
f	t	t	f
f	f	t	t
		…	

- … die wir kategorisieren wollen mit möglichen Werten einer Klasse C = {1,2,3}
- Wir verwenden einen naiven Bayes-Klassifikator mit versteckter Variable C

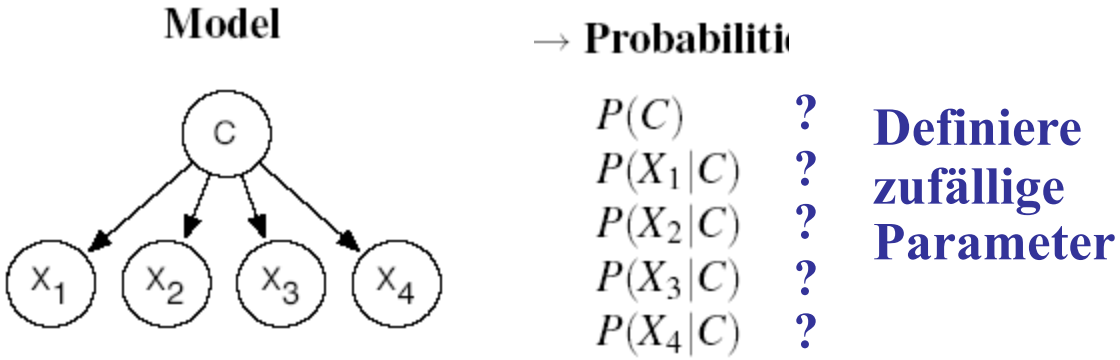
Model → **Probabiliti**



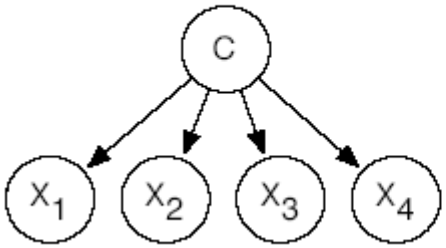
$$P(C) \quad ?$$
$$P(X_1|C) \quad ?$$
$$P(X_2|C) \quad ?$$
$$P(X_3|C) \quad ?$$
$$P(X_4|C) \quad ?$$

EM: Initialisierung

- Der Algorithmus startet mit "erfundenen" (zufällig generierten) Informationen, um das Lernproblem zu lösen:
  - Erfundene Netzwerkparameterwerte (virtuell notiert in den CPTs)



EM: Expectation-Schritt (Bestimmung der erwarteter Zahlen)



Data

X_1	X_2	X_3	X_4
t	f	t	t
f	t	t	f
f	f	t	t
		$\ldots$	

→ Probabilities

$P(C)$?
$P(X_1|C)$?
$P(X_2|C)$?
$P(X_3|C)$?
$P(X_4|C)$?

- Was benötigen wir, um die Netzwerkparameter mit dem Maximum-Likelihood-Ansatz zu lernen?
  - Für P(C) = Anzahl(Datenpunkte mit C=i) / Anzahl (alle Datenpunkte) i=1,2,3
  - Für $P(X_h|C)$ = Anzahl(Daten mit $X_h = val_k$ und C=i) / Anzahl(Daten mit C=i)

    für alle Werte val_k von X_h und i=1,2,3

EM: Expectation-Schritt (Bestimmung der erwarteter Zahlen)

- Bislang haben wir nur: N = Anzahl(Datenpunkte)
- Wir approximieren alle weiteren Zähler mit erwarteten Zählerwerten, die aus dem Modell mit den "erfundenen" Parametern abgeleitet werden
- Der erwartete Zähler $\hat{N}(C = i)$ ist die Summe, über alle N Beispieldaten, der Wahrscheinlichkeiten, dass jedes Beispiel in Kategorie i fällt

$$\hat{N}(C = i) = \sum_{j=1}^{N} P(C = i \mid \text{Attribute von Beispiel } e_j)$$

$$= \sum_{j=1}^{N} P(C = i \mid x1_j, x2_j, x3_j, x4_j)$$

EM: Expectation-Schritt (Bestimmung der erwarteter Zahlen)

- Wie erhalten wir die notwendigen Werte aus dem Modell?

$$\hat{N}(C=i) = \sum_{j=1}^{N} P(C=i \mid \text{ Attribute von Beispiel } e_j)$$

$$= \sum_{j=1}^{N} P(C=i \mid x1_j, x2_j, x3_j, x4_j)$$

- Leicht mit naivem Bayesschen Netzwerk

$$P(C=i \mid x1_j, x2_j, x3_j, x4_j) = \frac{P(C=i, x1_j, x2_j, x3_j, x4_j)}{P(x1_j, x2_j, x3_j, x4_j)}$$

$$= \frac{P(x1_j \mid C=i)..,P(x4_j \mid C=i)P(C=i)}{P(x1_j, x2_j, x3_j, x4_j)}$$

Auch direkt aus Netzwerk bestimmbar. Übungsaufgabe

Erfundene Parameter für das Netzwerk

EM: Expectation-Schritt (Bestimmung der erwarteter Zahlen)

- Durch einen ähnlichen Schritt erhalten wir die erwarteten Zähler für Beispiele mit Attribut $X_h = val_k$ und Kategorie i
- Diese werden später benötigt zur Schätzung von $\mathbf{P}(X_h \mid C)$:

$$P(X_h \mid C) = \frac{\text{Exp. Counts(examples with } X_h = val_k \text{ and } C = i)}{\text{Exp. Counts(examples with } C = i)} = \frac{\hat{N}(X_h = val_k, C = i)}{\hat{N}(C = i)}$$

  - für alle Werte val_k von X_h und i=1,2,3

- Zum Beispiel

$$\hat{N}(X_1 = t, C = 1) = \sum_{e_j \text{ with } X_1 = t} P(C = i \mid x1_j = t, x2_j, x3_j, x4_j)$$

Wiederum erhalten wir diese W'keiten aus dem aktuellen Modell

EM: Generelle Idee

- Der Algorithmus startet mit "erfundenen" (zufällig generierten) Informationen, um das Lernproblem zu lösen:
  - Erfundene Netzwerkparameterwerte (virtuell notiert in den CPTs)
- Dann werden die Initialwerte in zwei Schritten verfeinert:
  - Expectation (E): Aktualisiere die Daten (virtuell) mit aus dem aktuellen Modell hergeleiteten Erwartungen
  - Maximization (M): Gegeben die aktualisierten Daten, aktualisieren die Parameter des BNs mitteln Maximum Likelihood (ML) Ansatz
    - ✓ Das ist der gleiche Schritt, wie im voll beobachtbaren Fall

CPT: Conditional Probability Table

Maximization-Schritt: (Verfeinerung der Parameter)

- Nun verfeinern wir die Netzwerkparameter mittels Maximum-Likelihood-Lernen auf den erwarteten Zählern

$$P(C = i) = \frac{\hat{N}(C = i)}{N}$$

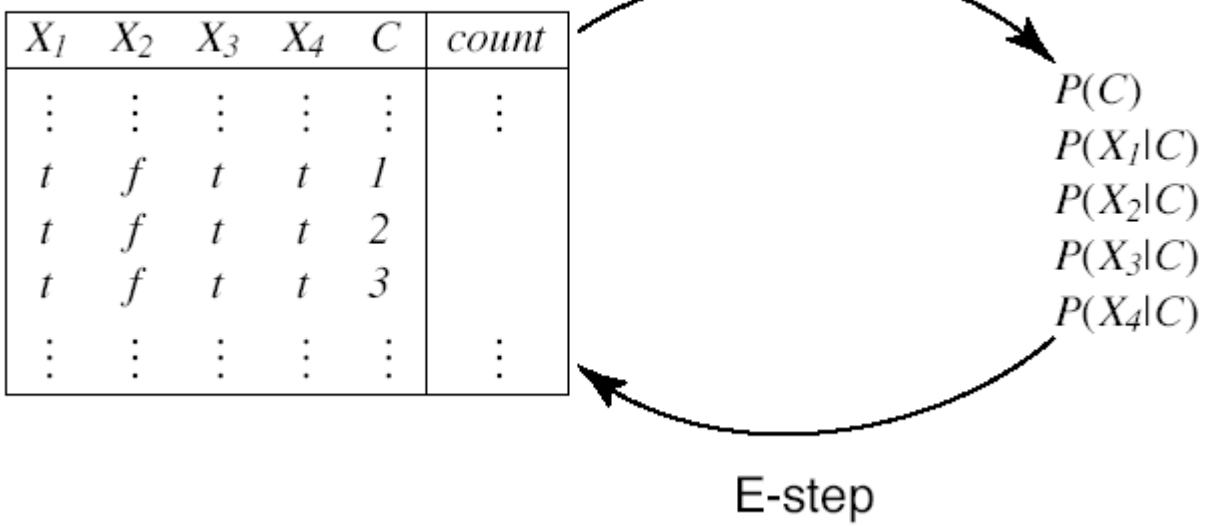
$$P(X_j = val_k \mid C = i) = \frac{\hat{N}(X_j = val_k C = i)}{\hat{N}(C = i)}$$

  - für alle Werte val_k von X_j und $i=1,2,3$

EM-Zyklus

- Nun kann der E-Schritt wiederholt werden

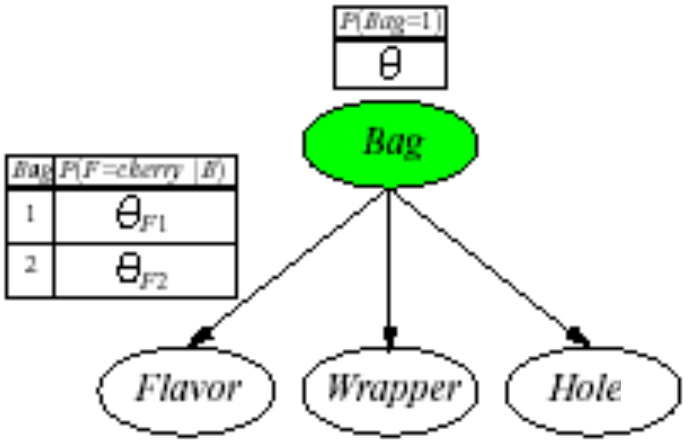
Erwartete Zähler
("Augmented data")



Wahrscheinlichkeiten

Beispiel: Bonbonfabrik wieder einmal

- Zwei Bonbontüten (1 and 2) wurden vermischt
- Bonbons werden durch drei Eigenschaften beschrieben: Flavor und Wrapper wie vorher, plus Hole (ob ein Loch in der Mitte ist)
- Die Merkmale von Bonbon hängen mit bestimmten Wahrscheinlichkeiten von der Tüte ab, aus der sie kommen
- Wir wollen für jedes Bonbon vorhersagen, aus welcher Tüte es kam, je nach vorgefundenen Eigenschaften: Naïve-Bayes-Modell



$\theta = P(Bag = 1)$

$\theta_{Fj} = P(Flavor = cherry | Bag = j)$

$\theta_{Wj} = P(Wrapper = red | Bag = j)$

$\theta_{Hj} = P(Hole = yes | Bag = j)$

$j = 1,2$

Daten

- Nehmen wir an, die wahren Parameter seien:
  - $\theta = 0.5$;
  - $\theta_{F1} = \theta_{W1} = \theta_{H1} = 0.8$;
  - $\theta_{F2} = \theta_{W2} = \theta_{H2} = 0.3$;
- Annahme: Die folgenden Zähler werden "gesampelt" aus **P**(C, F, W, H) (N = 1000): Wir haben einen Datensatz

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=lime	79	100	94	167

- Wir wollen nun die wahren Parameter aus diesen Daten mittels EM rekonstruieren

EM: Initialisierung

- Weise Parametern zufällige Initialwerte zu
  - Zu Vereinfachung der Darstellung verwenden wir hier:

$$\theta^{(0)} = 0.6;$$
$$\theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6;$$
$$\theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$$

- Nun durchlaufen wir einen EM-Zyklus zur Berechnung von $\theta^{(1)}$.

E-Schritt

- Zuerst brauchen wir die erwarteten Zähler für Bonbon von Tüte 1:
  - Addiere die Wahrscheinlichkeiten, dass jedes der N Datenpunkte aus Tüte 1 kommt
  - Seien $flavor_j$, $wrapper_j$, $hole_j$ die Werte der entsprechenden Attribute für den j-ten Datenpunkt

$$\hat{N}(\text{Bag} = 1) = \sum_{j=1}^{N} P(Bag = 1 \mid flavor_j, wrapper_j, hole_j) =$$

$$= \sum_{j=1}^{N} \frac{P(flavor_j, wrapper_j, hole_j | Bag = 1) P(Bag = 1)}{P(flavor_j, wrapper_j, hole_j)}$$

$$= \sum_{j=1}^{N} \frac{P(flavor_j | Bag = 1) P(wrapper_j | Bag = 1) P(hole_j | Bag = 1) P(Bag = 1)}{\sum_i P(flavor_j | Bag = i) P(wrapper_j | Bag = i) P(hole_j | Bag = i) P(Bag = i)}$$

E-step

$$\sum_{j=1}^{N} \frac{P(flavor_j|Bag=1)P(wrapper_j|Bag=1)P(hole_j|Bag=1)P(Bag=1)}{\sum_i P(flavor_j|Bag=i)P(wrapper_j|Bag=i)P(hole_j|Bag=i)P(Bag=i)}$$

- Die Summation kann in die 8 Bonbongruppen aus der Tabelle aufgebrochen werden.
  - Zum Beispiel ergibt die Summe über 273 cherry-Bonbons mit rotem Papier und einem Loch (erster Eintrag in der Tabelle)

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=lime	79	100	94	167

$$= 273 \frac{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)}}{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)} + \theta_{F2}^{(0)}\theta_{W2}^{(0)}\theta_{H2}^{(0)}(1-\theta^{(0)})} =$$

$$273 \frac{0.6^4}{0.6^4 + 0.4^4} = 273 \frac{0.1296}{0.1552} = 227.97$$

$$\theta^{(0)} = 0.6;$$

$$\theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6;$$

$$\theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$$

M-Schritt

- Mit der Berechnung der anderen 7 Bonbongruppe erhalten wir

$$\hat{N}(Bag = 1) = 612.4$$

- Nun führen wir den M-Schritt aus, um θ zu verfeinern. Hierzu nehmen wir den erwarteten Zählerwert der Datenpunkte aus Tüte 1

$$\theta^{(1)} = \frac{\hat{N}(Bag = 1)}{N} = 0.6124$$

Noch ein Parameter…

- Wir machen das gleiche für den Parameter θ_{F1}
- E-Schritt: Bestimme erwarteten Zählerwert von cherry-Bonbons aus Tüte 1

$$\hat{N}(Bag = 1 \wedge Flavor = cherry) = \sum_{j:Flavor_j=cherry} P(Bag = 1 \mid Flavor_j = cherry, wrapper_j, hole_j)$$

  - Bestimmbar aus naivem Bayesschen Modell wie vorher besprochen
  - Als Übung…
- M-Schritt: Verfeinere θ_{F1} durch Bestimmung der relativen Häufigkeiten

$$\theta_{F1}^{(1)} = \frac{\hat{N}(Bag = 1 \wedge Flavor = cherry)}{\hat{N}(Bag = 1)}$$

Lernergebnis

- Nach einem vollen Zyklus über alle Parameter haben wir

$$\theta^{(1)} = 0.6124;$$

$$\theta_{F1}^{(1)} = 0.6684; \quad \theta_{W1}^{(1)} = 0.6483; \quad \theta_{H1}^{(1)} = 0.658;$$

$$\theta_{F2}^{(1)} = 0.3887; \quad \theta_{W2}^{(1)} = 0.3817; \quad \theta_{H2}^{(1)} = 0.3827;$$

- Für jeden Satz von Parametern können wir die Log-Likelihood bestimmen wie auch schon früher

- Man kann zeigen, dass dieser Wert mit jeder EM-Iteration steigt (Konvergenz)

- EM mit Maximum-Likelihood bleibt aber bei lokalen Maxima hängen bleiben (ggf. mehrere Startwerte nehmen, Priors berücksichtigen oder MAP nehmen)

Lernergebnis

- Nach einem vollen Zyklus über alle Parameter haben wir

$$\theta^{(1)} = 0.6124;$$

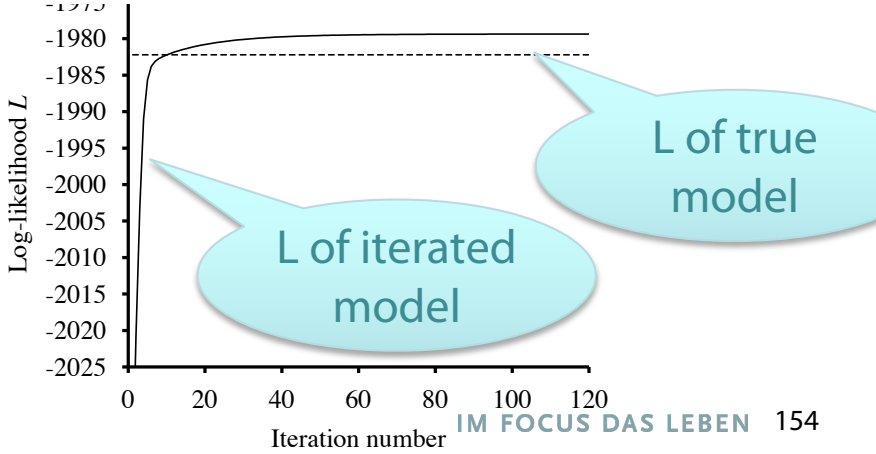
$$\theta_{F1}^{(1)} = 0.6684; \quad \theta_{W1}^{(1)} = 0.6483; \quad \theta_{H1}^{(1)} = 0.658;$$

$$\theta_{F2}^{(1)} = 0.3887; \quad \theta_{W2}^{(1)} = 0.3817; \quad \theta_{H2}^{(1)} = 0.3827;$$

- Für jeden Satz von Parametern können wir die Log-Likelihood bestimmen wie auch schon früher

$$P(\mathbf{d} \mid h_{\theta^{(i)}\theta_{F1}^{(i)}\theta_{W1}^{(i)}\theta_{H1}^{(i)}\ \theta_{F2}^{(i)}\ \theta_{W2}^{(i)}\theta_{H2}^{(i)}}) = \prod_{j=1}^{1000} P(d_j \mid h_{\theta^{(i)}\theta_{F1}^{(i)}\theta_{W1}^{(i)}\theta_{H1}^{(i)}\ \theta_{F2}^{(i)}\ \theta_{W2}^{(i)}\theta_{H2}^{(i)}})$$

- Wir können zeigen, das die Log-Likelihood $L = \log P(d, h_\theta)$ in jeder Iteration ansteigt, so dass die initiale Likelihood schon nach drei Iterationen überflügelt wird



EM: Diskussion

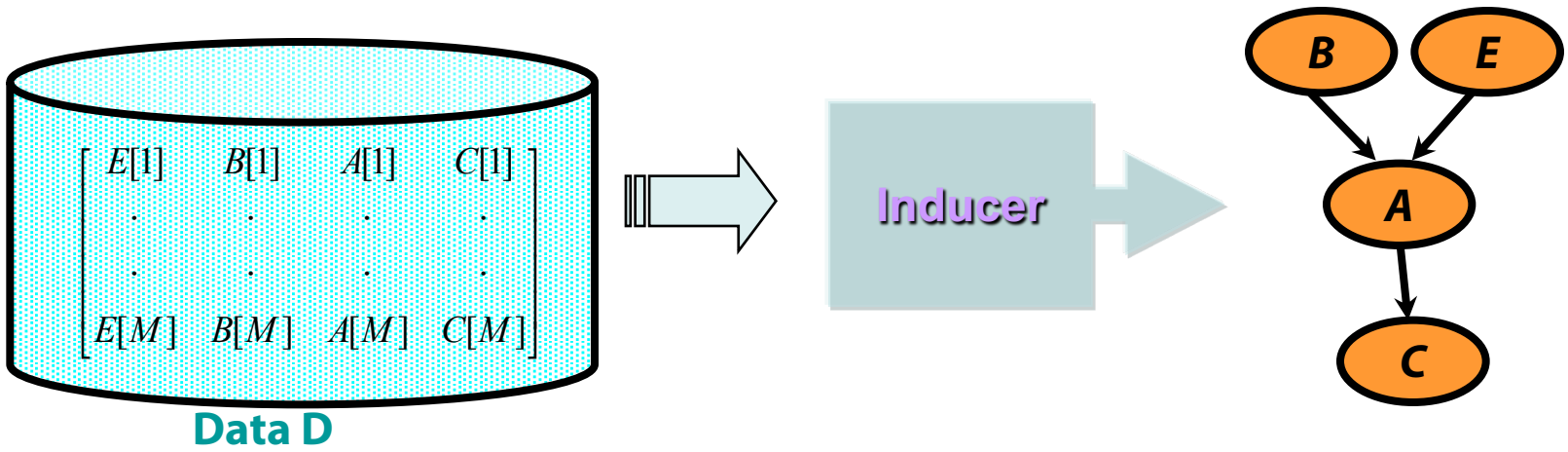
- Für komplexere BNs ist der Algorithmus der gleiche
  - Im allgemeinen müssen wir die CPT-Einträge für jede Variable X_i gegeben die Elternknotenwerte Pa_i berechnen
  - $\theta_{ijk} = P(X_i = x_{ij} | Pa_{i=} pa_{ik})$

$$\theta_{ijk} = \frac{\hat{N}(X_i = x_{ij}; Pa_i = pa_{ik})}{\hat{N}(Pa_i = pa_{ik})}$$

- Die erwarteten Zählerwerte werden durch Summierung über die Beispiele berechnet, nachdem all notwendigen $P(X_i = x_{ij}, Pa_{i=} pa_{ik})$ mittels BN-Algorithmen bestimmt sind
- Das Verfahren kann in eine kombinatorische Explosion laufen (ggf. Approximationstechniken angewendet)

Learning Bayesian network structures

- Given training set $\boldsymbol{D} = \{\boldsymbol{x}[1],\ldots,\boldsymbol{x}[\boldsymbol{M}]\}$
- Find model that best matches ***D***
  - model selection
  - parameter estimation



Some of the following slides from an AI course „Graphical models“ by Burgard/De Raedt/Kersting/Nebel

Model selection

Goal: Select the best network structure, given the data

Input:

- Training data
- Scoring function

Output:

- A network that maximizes the score

Structure selection: Scoring

- Bayesian: prior over parameters and structure
  - get balance between model complexity and fit to data as a byproduct

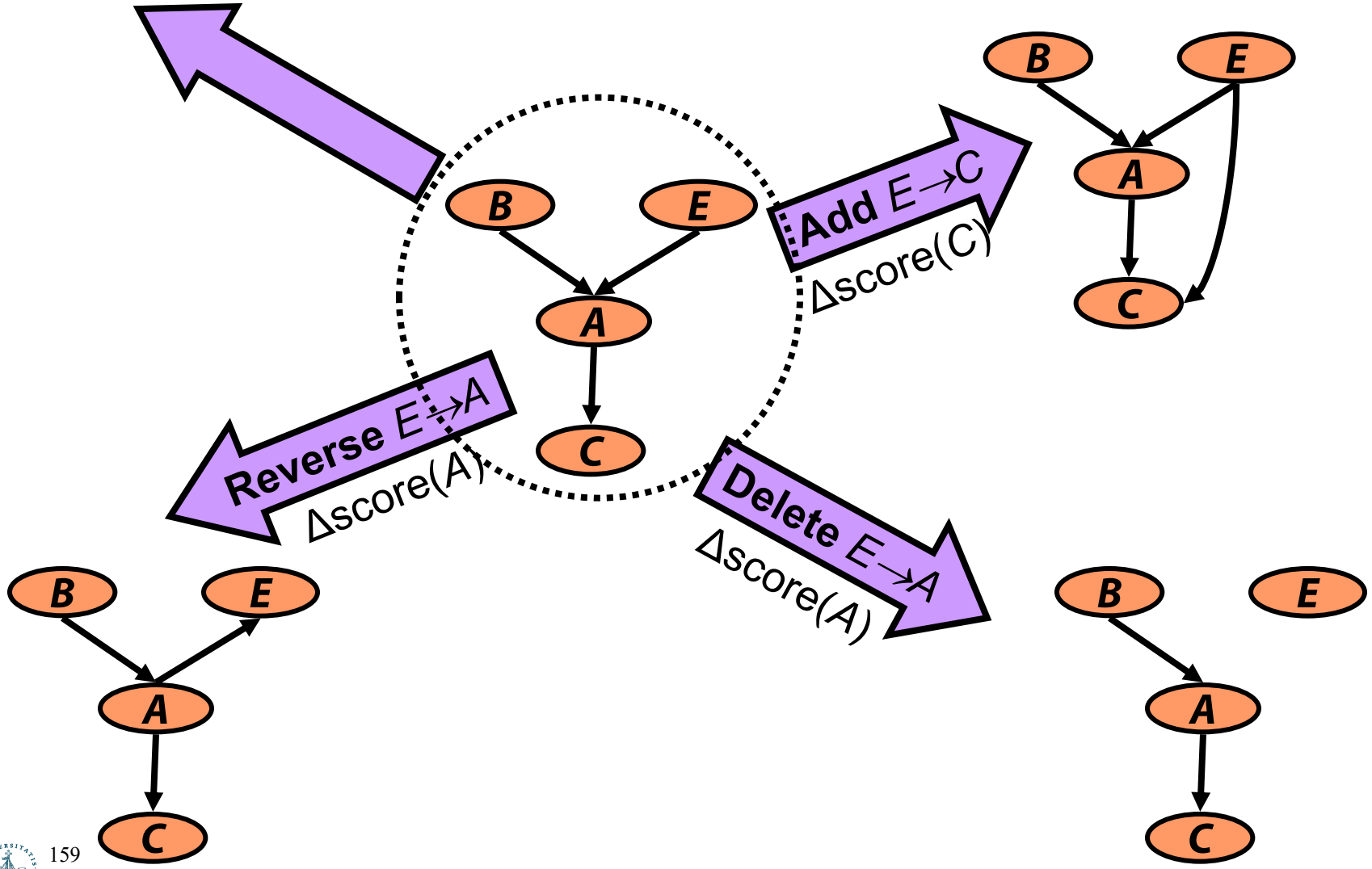
Can we learn G's params from D? | Does G explain D with ML? | Prior w.r.t. MDL

- $\text{Score}_D(G) = \log P(G|D) = \alpha \log [P(D|G)\, P(G)]$
- Marginal likelihood just comes from our parameter estimates
- Prior on structure can be any measure we want; typically a function of the network complexity (MDL principle)

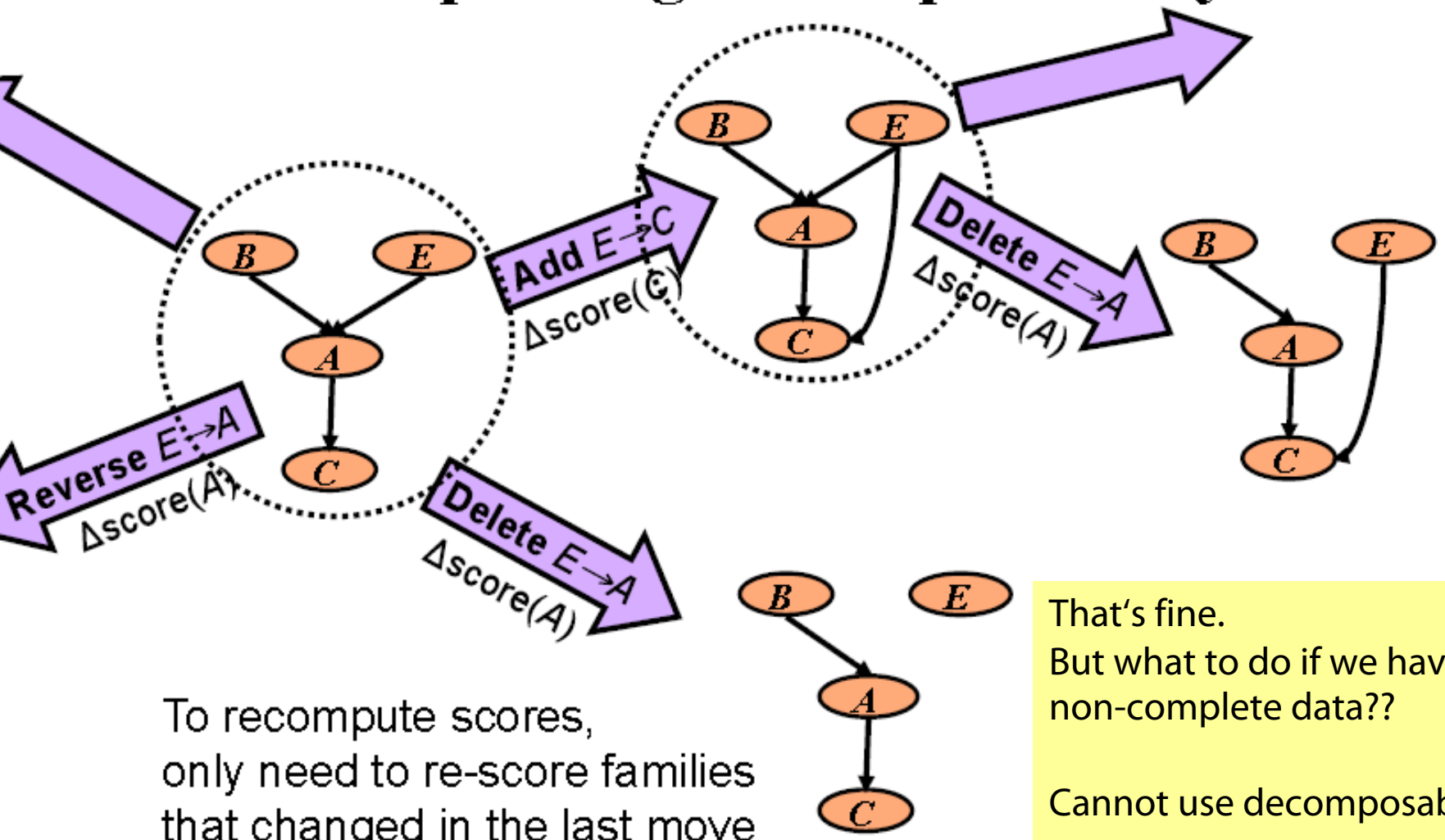
Same key property: Decomposability

$$\text{Score(structure)} = \Sigma_i \text{ Score(substructure of } X_i)$$

Heuristic search



Exploiting decomposability



To recompute scores,
only need to re-score families
that changed in the last move

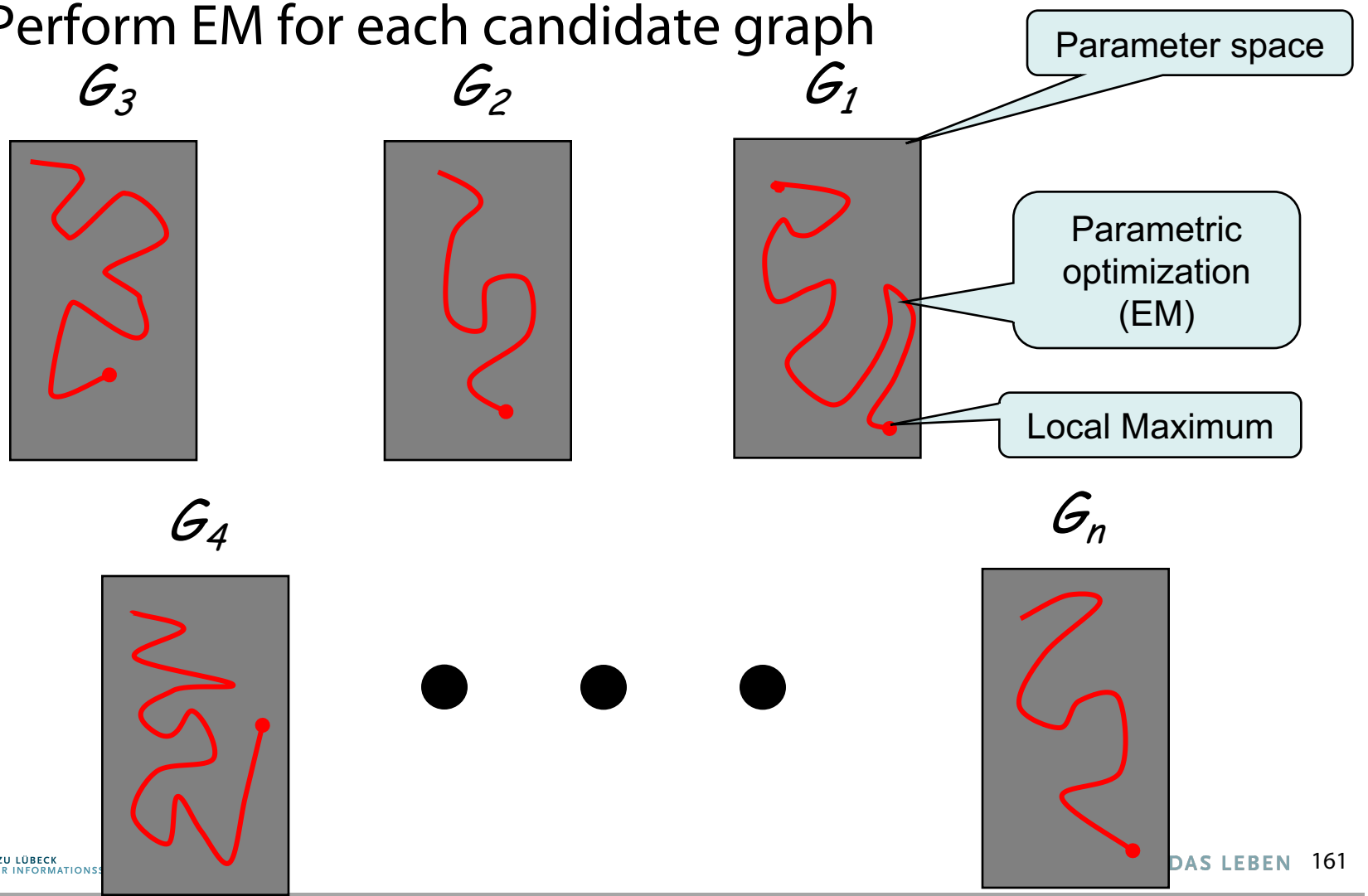
That's fine.
But what to do if we have non-complete data??

Cannot use decomposability

=> Rerun parameter estimation

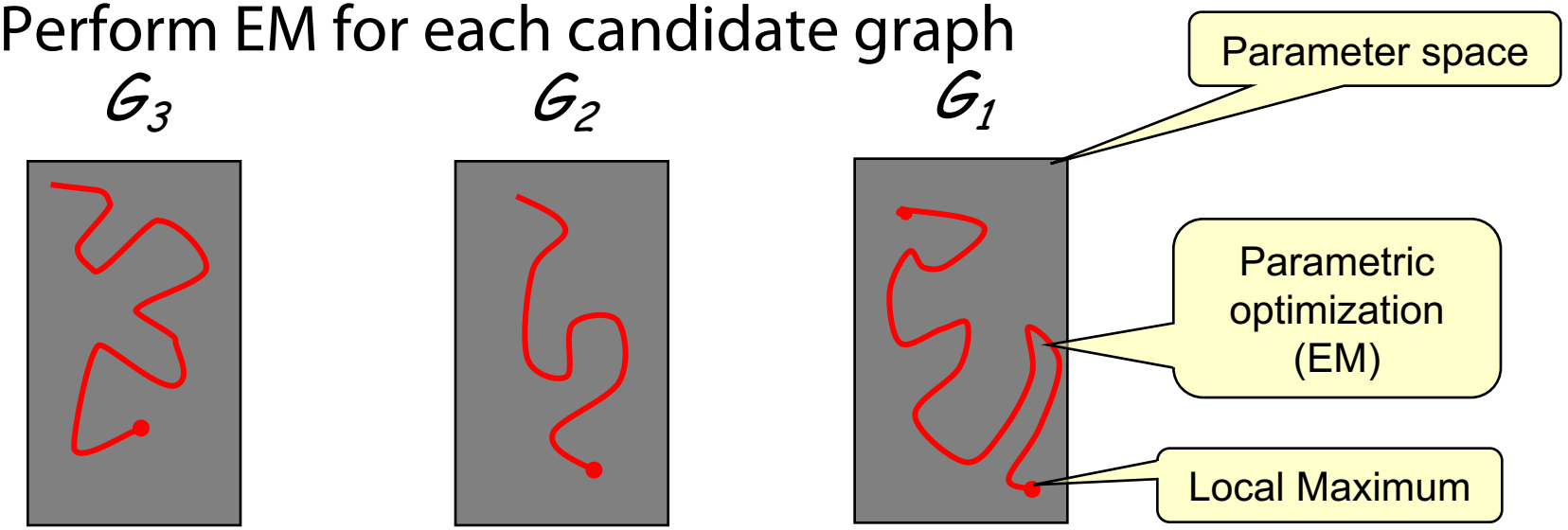
Local Search in Practice

- Perform EM for each candidate graph



Local Search in Practice

- Perform EM for each candidate graph



- Computationally expensive:
  - Parameter optimization via EM — non-trivial
  - Need to perform EM for all candidate structures
  - Spend time even on poor candidates

$\Rightarrow$ In practice, considers only a few candidates

Structural EM [Friedman et al. 98]

Recall, in complete data we had

– Decomposition $\Rightarrow$ efficient search

Idea:

- Instead of optimizing the real score…
- Find decomposable alternative score
- Such that maximizing new score

$\Rightarrow$ improvement in real score

Structural EM

Idea:

- Use current model to help evaluate new structures

Outline:

- Perform search in (Structure, Parameters) space
- At each iteration, use current model for finding either:
  - Better scoring parameters: "parametric" EM step

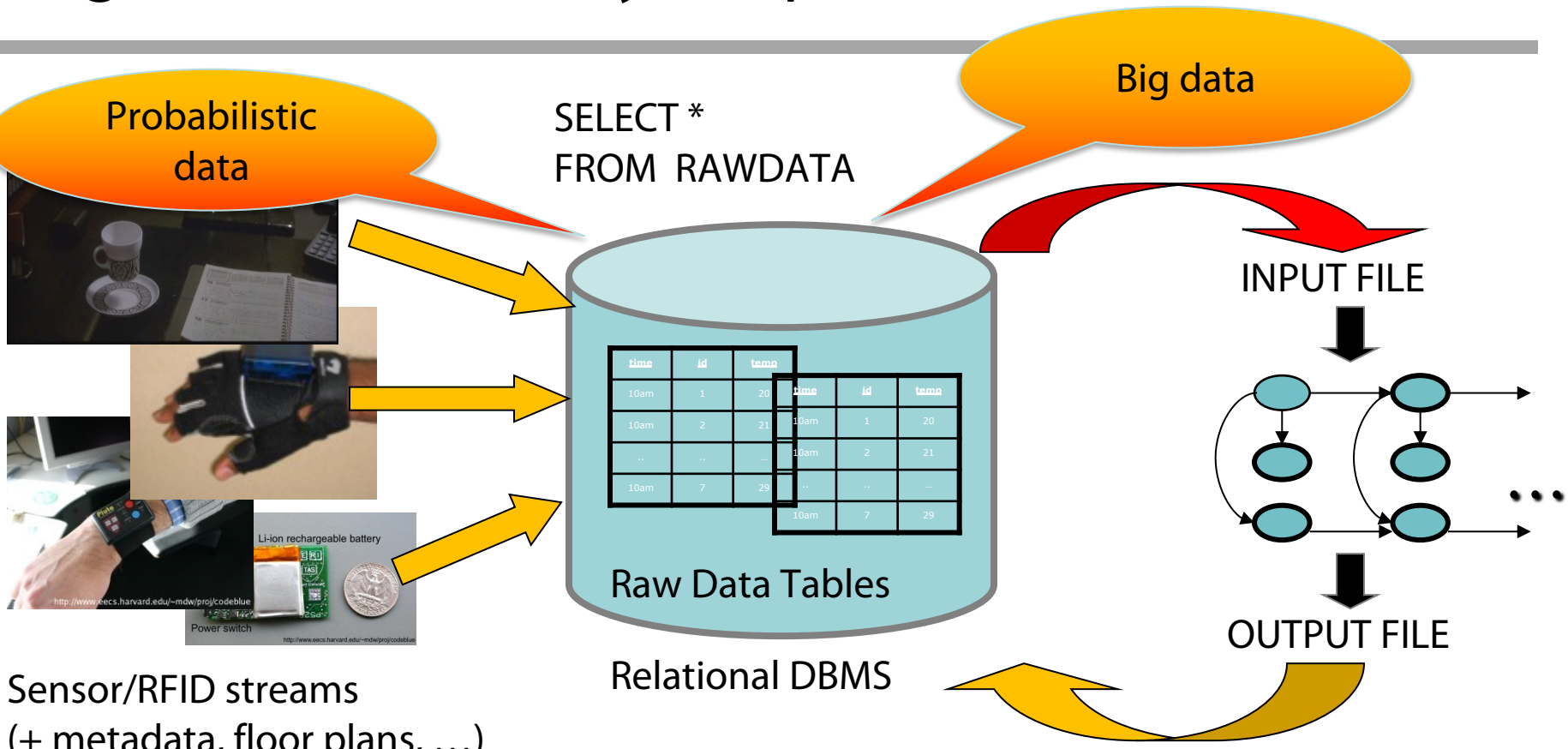
  or
  - Better scoring structure: "structural" EM step

Nir Friedman. The Bayesian structural EM algorithm. In Proceedings of the 14th conference on Uncertainty in artificial intelligence (UAI'98), 129-138. **1998**

Variations on a theme

- **Known structure, fully observable**: only need to do parameter estimation
- **Known structure, hidden variables:** use expectation maximization (EM) to estimate parameters
- **Unknown structure, fully observable:** do heuristic search through structure space, then parameter estimation
- **Unknown structure, hidden variables:** structural EM

Big Datasets Destroy Simple Abstractions



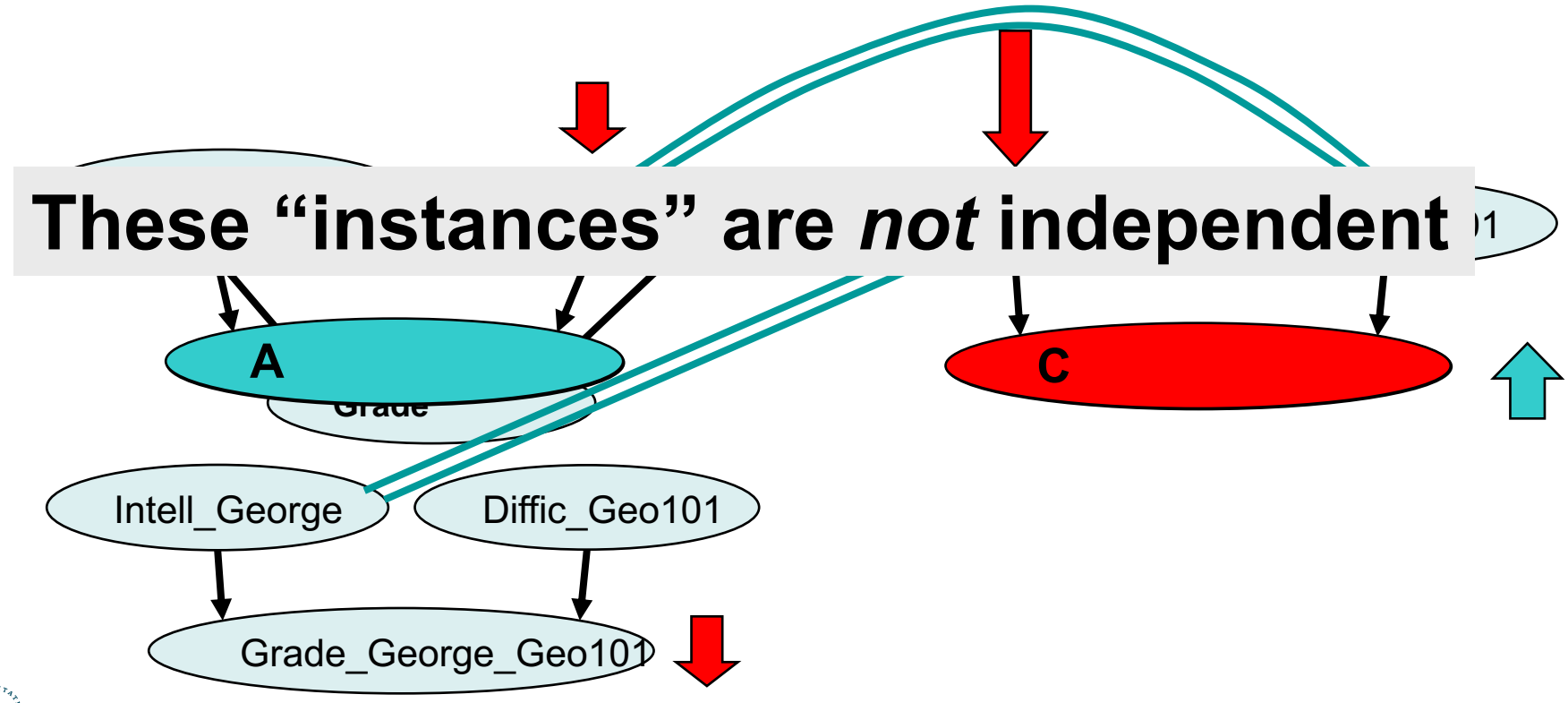
Managing Large, Uncertain Data
Repositories with Probabilistic Graphical Models
Daisy Zhe Wang+, Eirinaios Michelakis+, Minos Garofalakis*+, Joseph M. Hellerstein+ University of California Berkeley+, Yahoo! Research*
25th August **2008**, VLDB

HeisenData -- Towards a Next-Generation Uncertain Data Management System
Project Funding: FP7 Marie-Curie International Reintegration Grants (FP7-PEOPLE-2009-RG, Reference No. 249217)
Project Duration: **1/3/2010 - 28/2/2014**
Fellow & Scientist-in-Charge: Prof. Minos Garofalakis

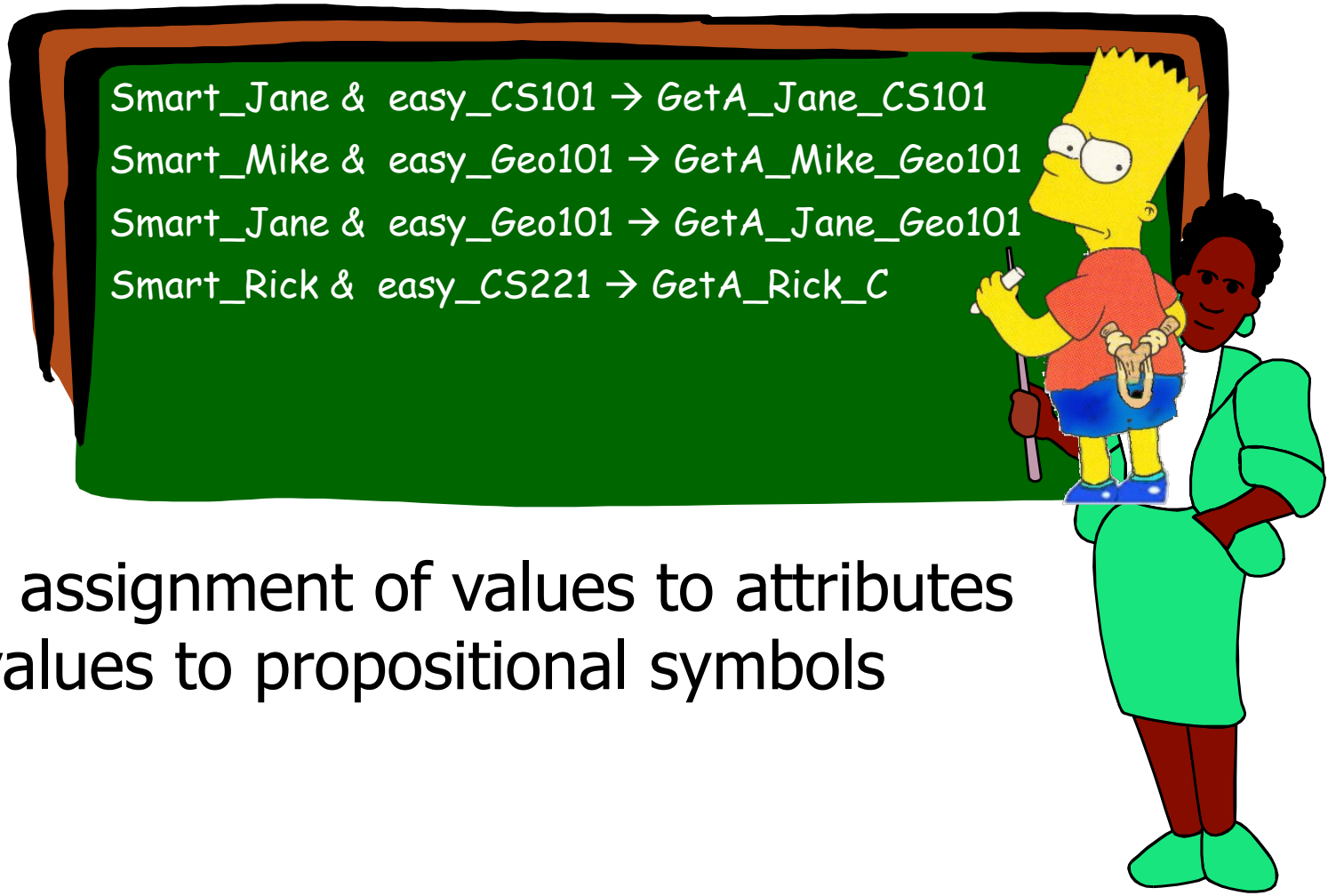
Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakis, and Joseph M. Hellerstein. "**BAYESSTORE**: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models", Proceedings of VLDB'2008 (PVLDB, Vol. 1), **2008**.

Bayesian Networks: Isomorphic Structures

- Bayesian nets use propositional representation
- Real world has objects, related to each other

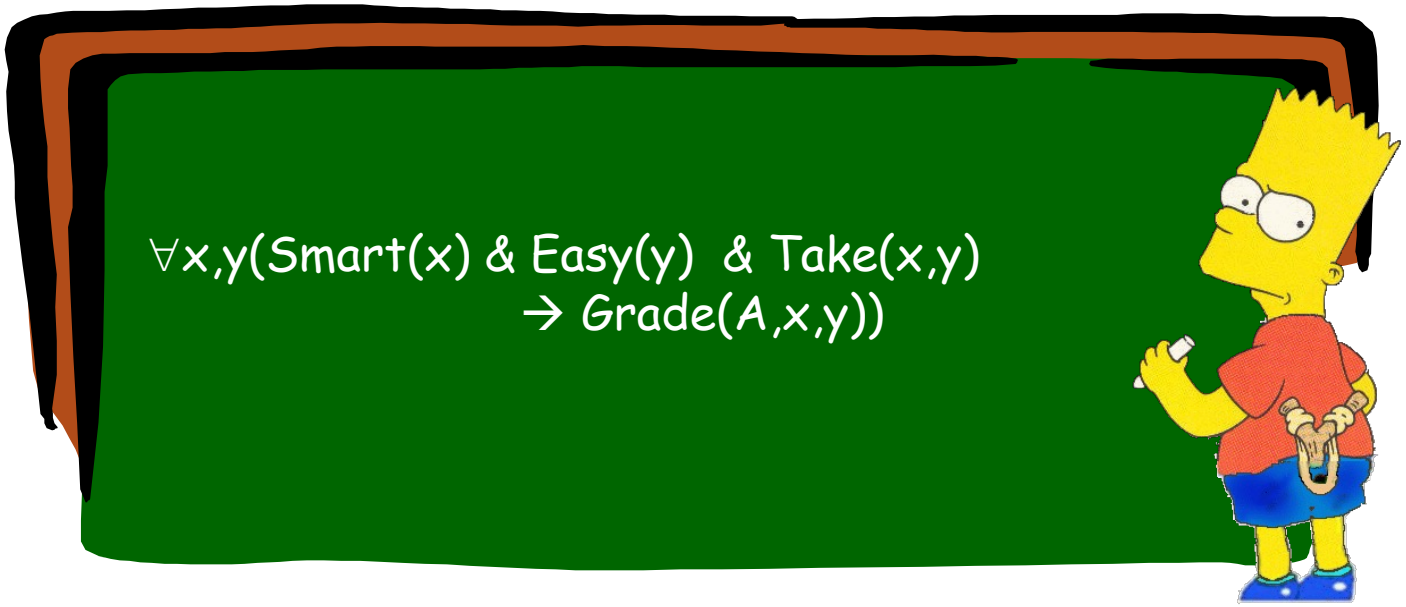


Attribute-Based Worlds



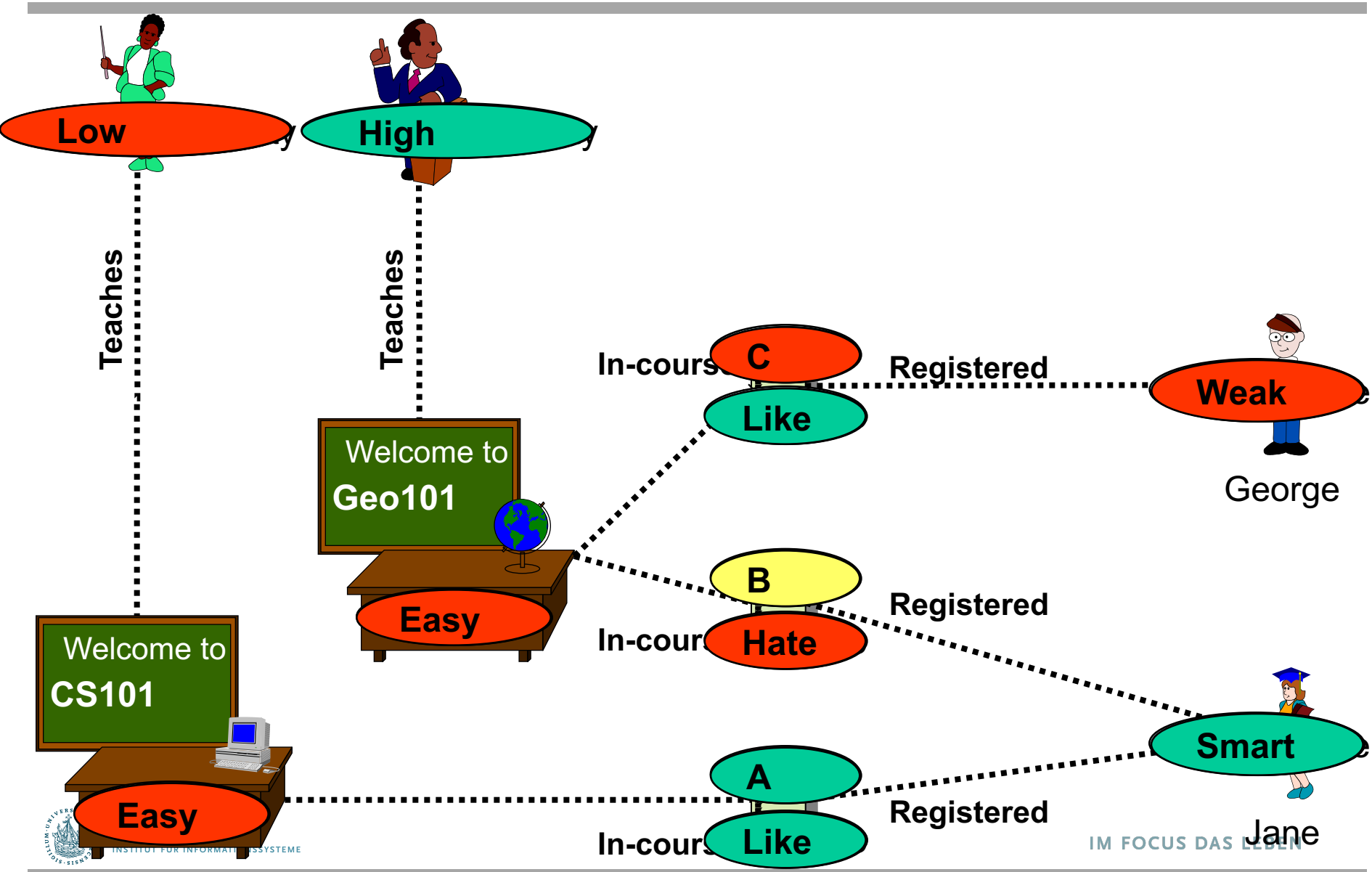
- World = assignment of values to attributes / truth values to propositional symbols

Object-Relational Worlds



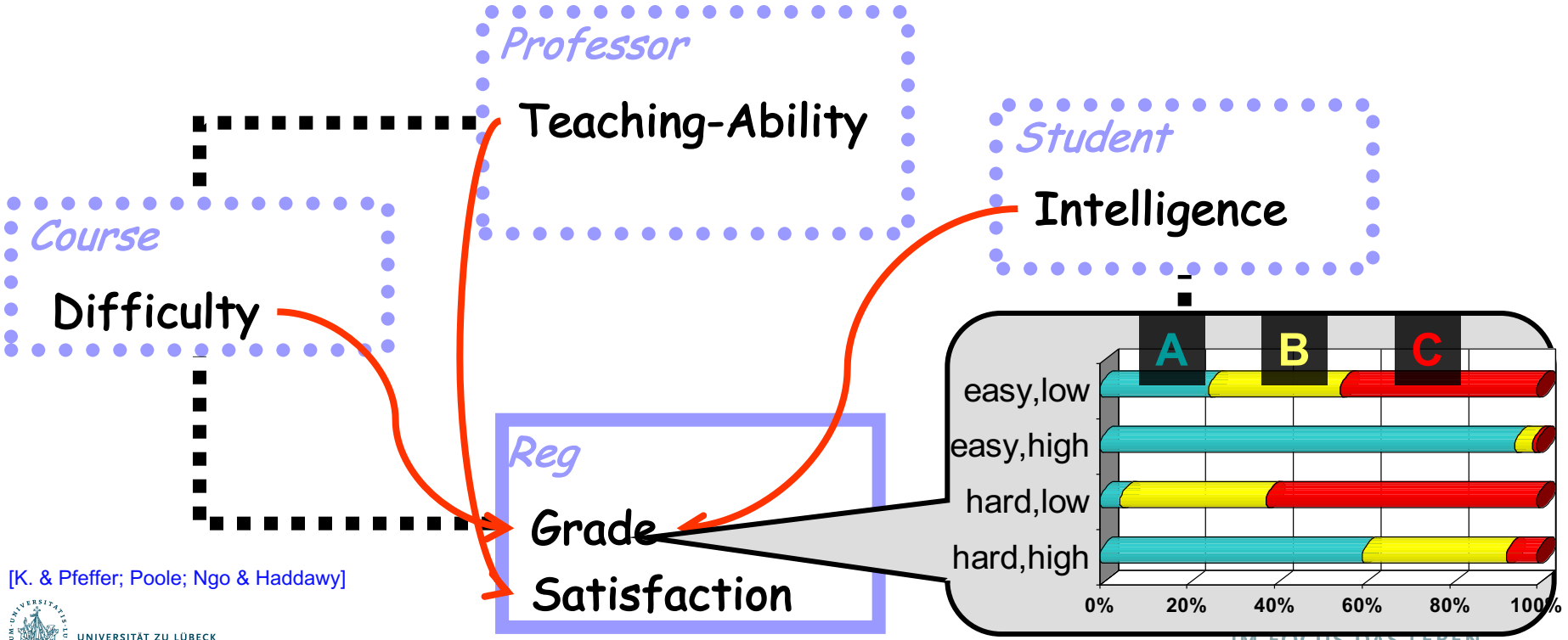
- World = relational interpretation:
  - Objects in the domain
  - Properties of these objects
  - Relations (links) between objects

St. Nordaf University

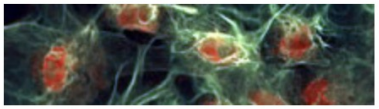


Relational Bayesian Network

- Universals: Probabilistic patterns hold for all objects in class
- Locality: Represent direct probabilistic dependencies
  - Links define potential interactions



[K. & Pfeffer; Poole; Ngo & Haddawy]



Forschung
- DFG-Fachkollegienwahl 2023
- Forschungsprofil
- Verbundforschung
- Forschungsförderung
- Interne Fördermöglichkeiten
- Universitäre Preise
- Forschungsdatenmanagement
- Forschungsinformationssystem FILU
- Strategische Internationalisierung
- Tierexperimentelle Forschung
- Kommissionen
- Termine
- Service & Kontakt

Aktuelles zur Forschung

AnoMed: Aus Daten lernen – aber sich

Donnerstag, 15.12.2022

Prof. Dr. Esfandiar Mohammadi, wissenschaftlicher Leiter des Kompetenzclusters. Foto: Tim Jelonnek

Develop probabilistic relational model inside a database

Bundesministerium für Bildung und Forschung

Neues Anonymisierungsnetzwerk adressiert drängende Herausforderungen in der Medizin und Informatik

Die Menge an verfügbaren sensiblen medizinisch-relevanten Daten wächst rasant. Mithilfe Maschineller Lernverfahren (sog. KI-Techniken) können große Datenmengen immer effektiver verarbeitet werden. Aber wie kann dabei die Anonymität der hinter diesen Datenmengen stehenden Patientinnen und Patienten gewahrt werden? Obgleich die Forschung in den letzten Jahren für dieses Anonymisierungsproblem vielversprechende Fortschritte gemacht hat, sind aktuell bekannte Anonymisierungslösungen für die meisten medizinischen Anwendungen nicht nutzbar. Genau hier setzt AnoMed (https://anomed.de)als eines von bundesweit fünf Kompetenzclustern Anonymisierung an. Das AnoMed-Cluster soll als Katalysator für Anonymisierungsforschung auf medizinischen Anwendungen dienen und medizinischen Anwendern Gefahren von Deanonymisierung und Möglichkeiten von neuesten Anonymisierungstechniken aufzeigen.