Probabilistic Graphical Models

Lecture 6 –Variable Elimination

CS/CNS/EE 155
Andreas Krause

Announcements

- Recitations: Every Tuesday 4-5:30 in 243 Annenberg
- Homework 1 due in class Wed Oct 21
- Project proposals due tonight (Monday Oct 19)

Structure learning

- Two main classes of approaches:

- Constraint based
  - Search for P-map (if one exists):
  - Identify PDAG
  - Turn PDAG into BN (using algorithm in reading)
  - **Key problem**: Perform independence tests
- Optimization based ← coming up!
  - Define scoring function (e.g., likelihood of data)
  - Think about structure as parameters
  - More common; can solve simple cases exactly

Finding the optimal MLE structure

- Optimal solution for MLE is always the fully connected graph!!! ☹
  - ➔Non-compact representation; Overfitting!!

- Solutions:
  - Priors over parameters / structures (later)
  - Constraint optimization (e.g., bound #parents)

Bayesian learning

- Make prior assumptions about parameters P(θ)
- Compute posterior

$$P(\theta \mid D) = \frac{P(\theta)P(D \mid \theta)}{P(D)} \propto P(\theta)\, P(D \mid \theta)$$

Given data D want to predict

$$P(X \mid D) = \int P(\theta \mid D)\, \underbrace{P(X \mid \theta)} \, d\theta$$

In MLE

$$P(X \mid D) \approx P(X \mid \hat{\theta}) \qquad \hat{\theta} = \arg\max P(D \mid \theta)$$

Conjugate priors

- Consider parametric families of prior distributions:
  - $P(\theta) = f(\theta; \alpha)$
  - α is called "hyperparameters" of prior
- A prior $P(\theta) = f(\theta; \alpha)$ is called **conjugate** for a likelihood function $P(D \mid \theta)$ if $P(\theta \mid D) = f(\theta; \alpha')$
  - Posterior has same parametric form
  - Hyperparameters are updated based on data D

- Obvious questions (answered later):
  - How to choose hyperparameters??
  - Why limit ourselves to conjugate priors??

Posterior for Beta prior

- Beta distribution

$$P(\theta) = \mathrm{Beta}(\theta; \alpha_H, \alpha_T) = \frac{\theta^{\alpha_H - 1}(1-\theta)^{\alpha_T - 1}}{B(\alpha_H, \alpha_T)}$$

- Likelihood:

$$P(\mathcal{D} \mid \theta) = \theta^{m_H}(1-\theta)^{m_T}$$

- Posterior:

$$P(\theta|D) \propto P(\theta) P(D|\theta) \propto \theta^{\alpha_H + m_H - 1} (1-\theta)^{\alpha_T + m_T - 1}$$

$$P(\theta|D) = \mathrm{Beta}(\theta; \alpha_H + m_H, \alpha_T + m_T)$$

Why do priors help avoid overfitting?

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) dP(\theta_{\mathcal{G}} \mid \mathcal{G})$$

- This Bayesian Score is tricky to analyze. Instead use:

$$\log P(\mathcal{D} \mid \mathcal{G}) \approx \log P(\mathcal{D} \mid \mathcal{G}, \widehat{\theta_{\mathcal{G}}}) - \frac{\log m}{2} \mathrm{Dim}(\mathcal{G})$$

- Why??
- **Theorem**: For Dirichlet priors, and for m→∞:

$$\log P(\mathcal{D} \mid \mathcal{G}) \to \log P(\mathcal{D} \mid \mathcal{G}, \widehat{\theta_{\mathcal{G}}}) - \frac{\log m}{2} \mathrm{Dim}(\mathcal{G}) + \mathcal{O}(1)$$

BIC score

$$\log P(\mathcal{D} \mid \mathcal{G}) \approx \underline{\log P(\mathcal{D} \mid \mathcal{G}}, \widehat{\theta_{\mathcal{G}}}) - \frac{\log m}{2} \operatorname{Dim}(\mathcal{G})$$

- This approximation is known as **Bayesian Information Criterion** (related to Minimum Description Length)

$$\log P(\mathcal{D} \mid \mathcal{G}) \approx m \sum_i \Big(\underline{\widehat{I}(X_i; \mathbf{Pa}_i) - \widehat{H}(X_i)} \Big) - \underline{\frac{\log m}{2}} \operatorname{Dim}(\mathcal{G})$$

- Trades goodness-of-fit and structure complexity!
- Decomposes along families (computational efficiency!)
- Independent of hyperparameters! (Why??)

Consistency of BIC

- Suppose true distribution has P-map G*
- A scoring function Score(G ; D) is called **consistent**, if, as m $\rightarrow \infty$ and probability $\rightarrow$ 1 over D:
  - G* maximizes the score
  - All non-I-equivalent structures have strictly lower score

- **Theorem**: BIC Score is consistent!

- Consistency requires m $\rightarrow \infty$. For finite samples, priors matter!

Parameter priors

- How should we choose priors for discrete CPDs?
- Dirichlet (computational reasons). But how do we specify hyperparameters??
- K2 prior:
  - Fix α
  - $P(\theta_{X \mid Pa_X}) = Dir(\alpha, \ldots, \alpha)$
- Is this a good choice?

X Y

$P(\theta_Y) = Dir(\alpha, \alpha)$

$\Rightarrow$ Equiv. sample size 2α

$X \rightarrow Y$

$P(\theta_{Y \mid X=H}) = Dir(\alpha, \alpha)$

$P(\theta_{Y \mid X=T}) = Dir(\alpha, \alpha)$

$\Rightarrow$ Equiv sample size 4α

BDe prior

- Want to ensure "equivalent sample size" m' is constant
- Idea:
  - Define $P'(X_1,\ldots,X_n)$

    For example: $P'(X_1,\ldots,X_n) = \prod_i \text{Uniform}(Val(X_i))$
  - Choose equivalent sample size m'
  - Set $\alpha_{x_i \mid pa_i} = m' P'(x_i, pa_i)$

$X \qquad Y \qquad\qquad X \rightarrow Y$

$$\alpha_y = m' P'(y) = m' \sum_x P'(x,y) = \sum_x \alpha_{y|x}$$

Score consistency

- A scoring function is called score-consistent, if all I-equivalent structures have same score

$X \rightarrow Y \rightarrow Z$ (G) $\qquad X \leftarrow Y \rightarrow Z$ (G')

$I(G) = I(G')$

Score consistency: $I(G) = I(G') \Rightarrow Score(G : D) = Score(G' : D)$

- K2 prior is inconsistent!
- BDe prior is consistent
- In fact, Bayesian score is consistent ⇔ BDe prior on CPTs!!

Score decomposability

- Proposition: Suppose we have
  - **Parameter independence**
  - **Parameter modularity**: if X has same parents in G, G', then same prior.
  - **Structure modularity**: P(G) is product over factors defined over families (e.g.: $P(G) = \exp(-c|G|)$)
- Then Score(D : G) **decomposes** over the graph:

$$\text{Score}(G ; D) = \sum_i \text{FamScore}(X_i \mid Pa_i; D)$$

- If G' results from G by modifying a single edge, only need to recompute the score of the affected families!!

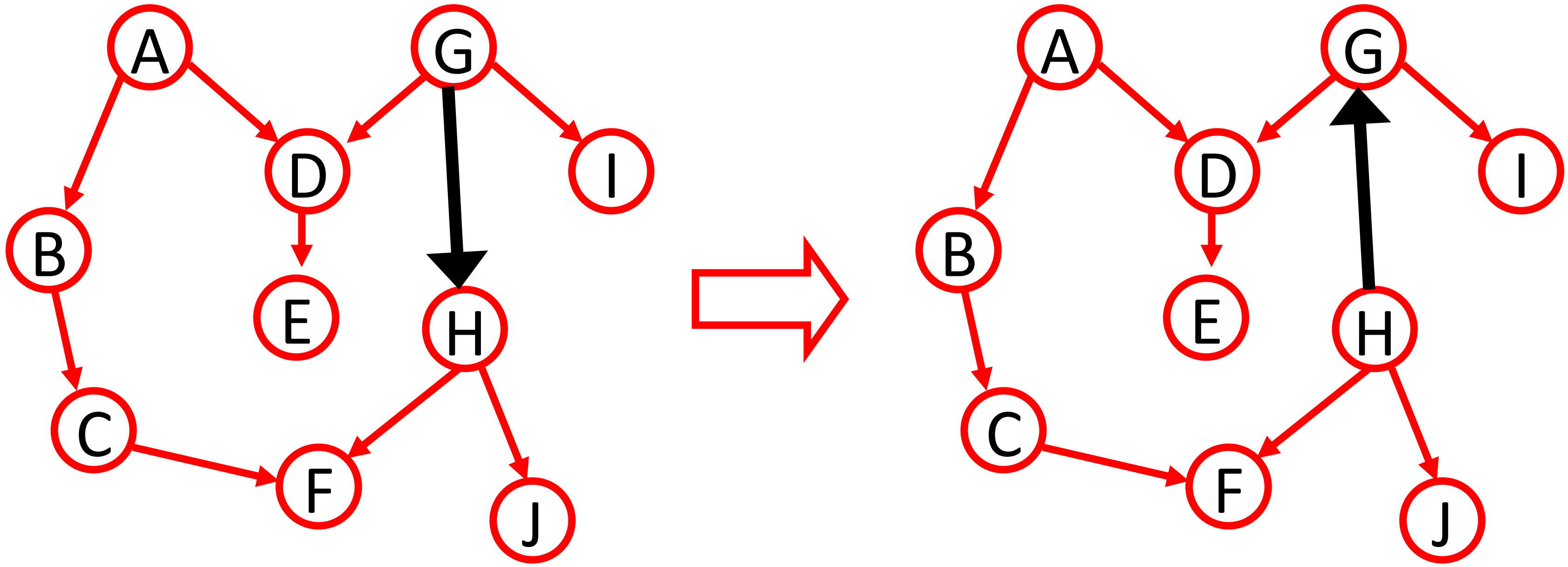
Bayesian structure search

- Given consistent scoring function Score(G : D), want to find to find graph G* that maximizes the score
- Finding the optimal structure is **NP-hard** in most interesting cases (details in reading). ☹
- Can find optimal tree/forest efficiently (Chow-Liu) ☺
- Want practical algorithm for learning structure of more general graphs..

Local search algorithms

- Start with empty graph (better: Chow-Liu tree)
- Iteratively modify graph by
  - Edge addition
  - Edge removal
  - Edge reversal
- Need to guarantee acyclicity (can be checked efficiently)
- Be careful with I-equivalence (can search over equivalence classes directly!)
- May want to use simulated annealing to avoid local maxima

Efficient local search



G

Score(G:D)

G'

Score(G':D)

- If Score is decomposable, only need to recompute affected families!

Alternative: Fixed order search

- Suppose we fix order $X_1,\ldots,X_n$ of variables
- Want to find optimal structure s.t. for all X_i: $\mathbf{Pa_i} \subseteq \{X_1,\ldots,X_{i-1}\}$

For $i = 1:n$
  For each $A \subset \{X_1 \ldots X_{i-1}\}$
    Compute FamScore$(X_i \mid A)$
    $A^* = \arg\max_A$ FamScore$(X_i \mid A)$

    $Pa_i = A^*$

$$\text{Score}(G : D) = \underbrace{\sum_i \text{FamScore}(X_i \mid Pa_i)}_{n \text{ independent opt. problems}}$$

$\Rightarrow$ Find optimal structure!

Fixed order for d parents

- Fix ordering
- For each variable X_i
  - For each subset $\mathbf{A} \subseteq \{X_1,\ldots,X_{i-1}\}$, $|A| \leq d$ compute FamScore(X_i | **A**)
  - Set $Pa_i = \text{argmax}_{\mathbf{A}}$ FamScore(X_i | **A**)
- If score is decomposable ➔ optimal solution!!
- Can find best structure by searching over all orderings!

Searching structures vs orderings?

- Ordering search
  - Find optimal BN for fixed order
  - Space of orderings "much smaller" than space of graphs..
    - **n!** orderings vs $\mathbf{2^{n^2}}$ directed graphs (counting DAGs more complicated)

- Structure search
  - Can have arbitrary number of parents
  - Cheaper per iteration
  - More control over possible graph modifications

What you need to know

- Conjugate priors
  - Beta / Dirichlet
  - Predictions, updating of hyperparameters
- Meta-BN encoding parameters as variables
- Choice of hyperparameters
  - BDe prior ⇒ score consistency
- Decomposability of scores and implications
- Local search
  - On graphs
  - On orderings (optimal for fixed order)

Key questions

- How do we specify distributions that satisfy particular independence properties?

  ➔ **Representation**

- How can we identify independence properties present in data?

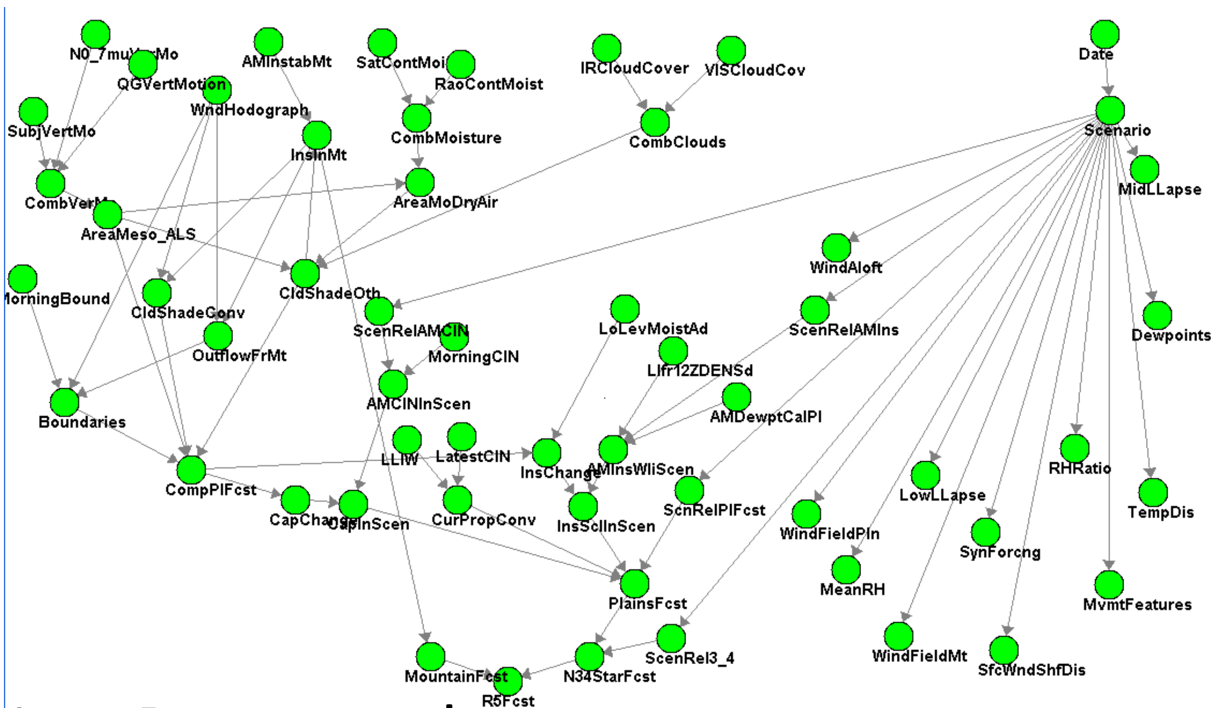
  ➔ **Learning**

- How can we exploit independence properties for efficient computation?

  ➔ **Inference**

Bayesian network inference

- Compact representation of distributions over large number of variables
- (Often) allows efficient **exact inference** (computing marginals, etc.)



JavaBayes applet

HailFinder

56 vars

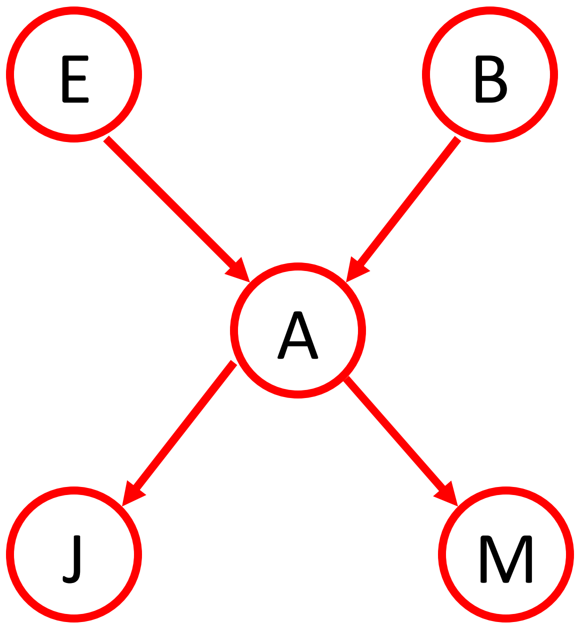
~ 3 states each

➔ $\sim 10^{26}$ terms

> **10.000 years**

on Top supercomputers

Typical queries: Conditional distribution



- Compute distribution of some variables given values for others

Observe $M=T$

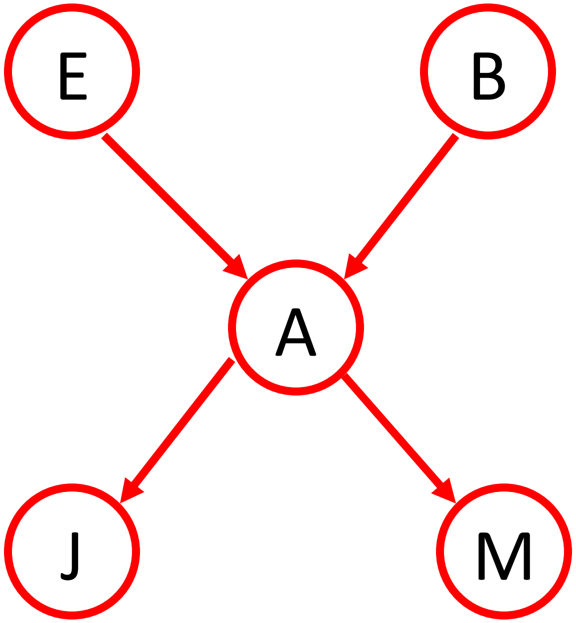
Compute $P(E=T \mid M=T)$

$$P(E=T \mid M=T) = \frac{P(E=T, M=T)}{P(M=T)}$$

$$P(E=T, M=T) = \underbrace{\sum_b \sum_a \sum_j}_{2^3 \text{ terms}} \underbrace{P(E=T, M=T, B=b, A=a, J=j)}_{P(E)P(B)P(A \mid E,B)P(J \mid A)P(M \mid A)}$$

Naive approach exponential in # vars...

Typical queries: Maxizimization



MPE and MAP don't necessarily give same answers...

- MPE (Most probable explanation):

  Given values for some vars, compute most likely assignment to all remaining vars

  Given $J=F$, $M=T$, find

  $$(e^*, b^*, a^*) = \underset{e,b,a}{\mathrm{argmax}}\ P(J=F, M=T, e, b, a)$$

- MAP (Maximum a posteriori):

  Compute most likely assignment to some variables

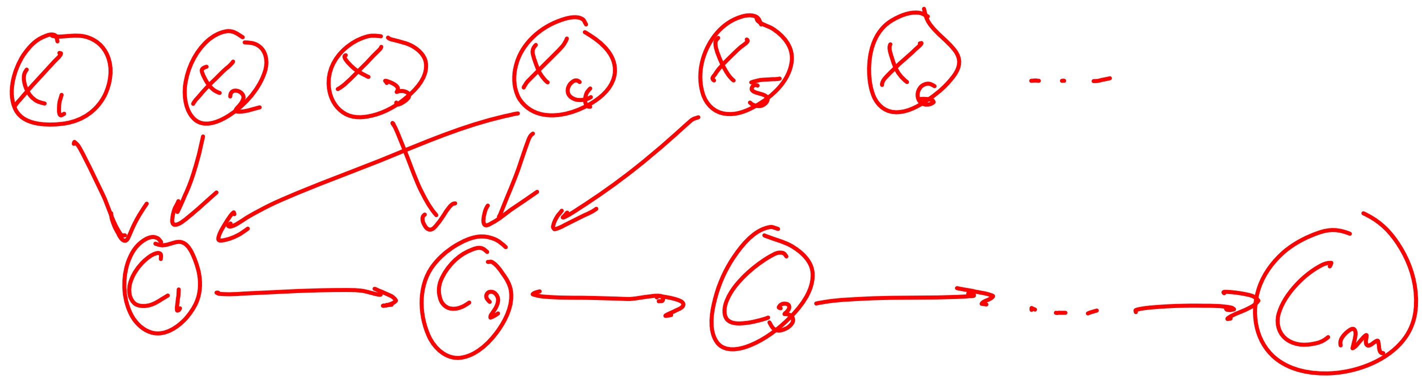
  $$e^* = \underset{e}{\mathrm{argmax}}\ P(J=F, M=T, E=e) =$$
  $$= \underset{e}{\mathrm{argmax}} \sum_{a,b} P(J=F, M=T, e, a, b)$$

Hardness of computing conditional prob.

- Computing P(X=x | E=e) is NP-hard
- **Proof**: by reduction from 3SAT

Given boolean formula $\varphi = (X_1 \vee X_2 \vee X_4) \wedge (\neg X_3 \vee X_4 \vee X_5) \wedge \ldots$

does there exist a sat. assignment to $X_1 \ldots X_n$



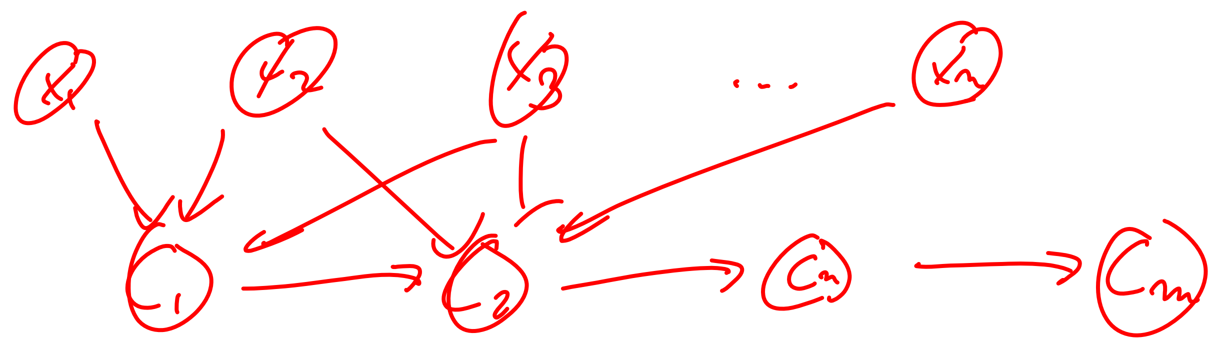
$P(X_i) = \mathrm{Bern}(0.5)$

$C_i = C_{i-1} \wedge$ Truth val for Clause$_i$

φ sat. $\Leftrightarrow P(C_m = T) > 0$

Hardness of computing cond. prob.

- In fact, it's even worse: P(X=x | E=e) is #P complete



$$P(C_m = T) = \sum_{x_1 \dots x_n} \underbrace{P(x_1 \dots x_n)}_{\frac{1}{2^n}} \cdot \underbrace{P(C_m = T \mid x_1 \dots x_n)}_{\substack{1 \text{ if } x_1 \dots x_n \text{ sat.} \\ 0 \text{ otherwise}}}$$

$$P(C_m = T) = \frac{\#\text{ sat assignments}}{2^n}$$

$\Rightarrow$ #P hardness

Hardness of inference for general BNs

- Computing conditional distributions:
  - Exact solution: #P-complete
  - Approximate solution: NP-hard:

Absolute approx: Finding $|P(x) - \hat{P}(x)| < \varepsilon$ NP-hard even for $\varepsilon = \frac{1}{2}$

Relative approx: $1-\varepsilon < \frac{\hat{P}(x)}{P(x)} < 1+\varepsilon$ NP hard for $\varepsilon > 0$

- Maximization:
  - MPE: NP-complete
  - MAP: NP^{PP}-complete

$\max_{x_1 \ldots x_m} \sum_{x_{m+1} \ldots x_{2m}}$

- Inference in general BNs is really hard ☹
- Is all hope lost?

Inference

- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations

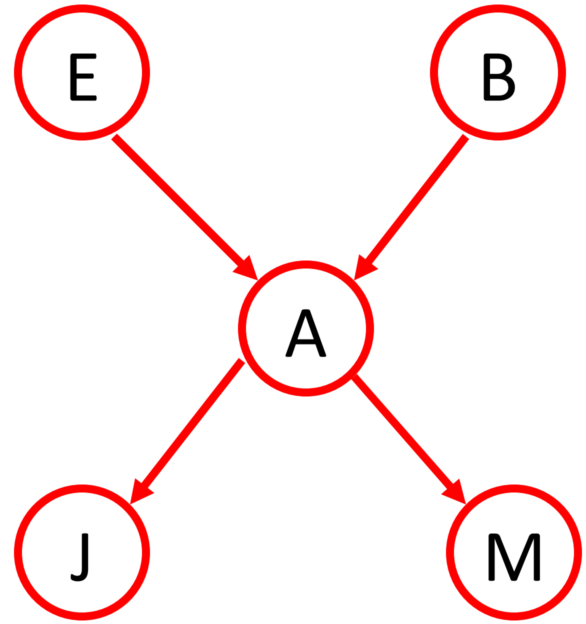
- For BNs where exact inference is not possible, can use algorithms for **approximate inference** (later this term)

Computing conditional distributions

- Query: P(X | E=e)

$$P(X \mid e) = \frac{P(X, e)}{P(e)} \propto P(X, e)$$

$\Rightarrow$ Renormalize (over x) to get $P(X \mid e)$

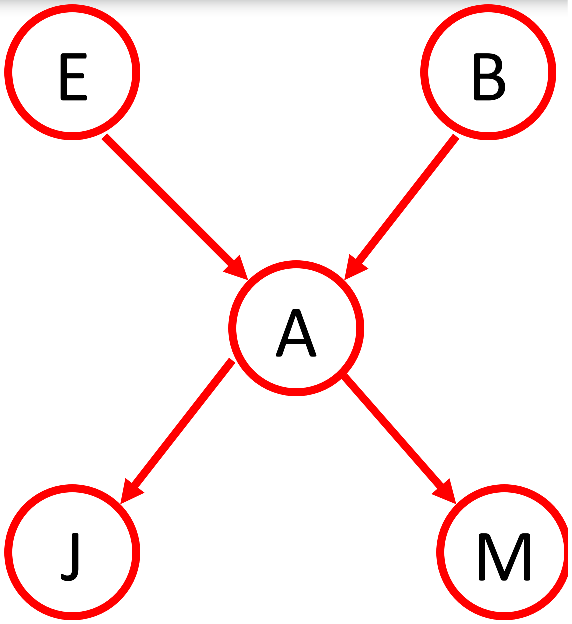


Inference example

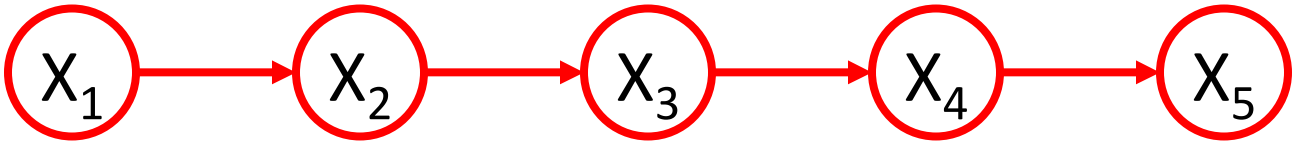
$$P(E \mid m) \propto P(E, m)$$

$$= \sum_{a,j,b} P(E, m, a, j, b)$$

$$= \underbrace{\sum_{a,j,b} P(E)\, P(b)\, P(a \mid E, b)\, P(j \mid a)\, P(m \mid a)}_{\text{in general, exponentially many terms}}$$



Potential for savings: Variable elimination!



$$P(X_5, X_1) = \sum_{X_2}\sum_{X_3}\sum_{X_4} P(X_1)\,P(X_2|X_1)\,P(X_3|X_2)\,P(X_4|X_3)\,P(X_5|X_4)$$

8

distributivity $P(X_1)\sum_{X_2} P(X_2|X_1)\sum_{X_3} P(X_3|X_2)\underbrace{\sum_{X_4} P(X_4|X_3)\,P(X_5|X_4)}_{g_4(X_3,X_5)\ 1}$

How many additions: 3

$X_1 \to X_2 \to X_3 \to X_5$ $\quad\quad = P(X_5|X_3)$

$g_3(X_2, X_5)$ 1

$X_1 \to X_2 \to X_5$

$\underbrace{g_2(X_1, X_5)}_{= P(X_5|X_1)}$ 1

$X_1 \to X_5$

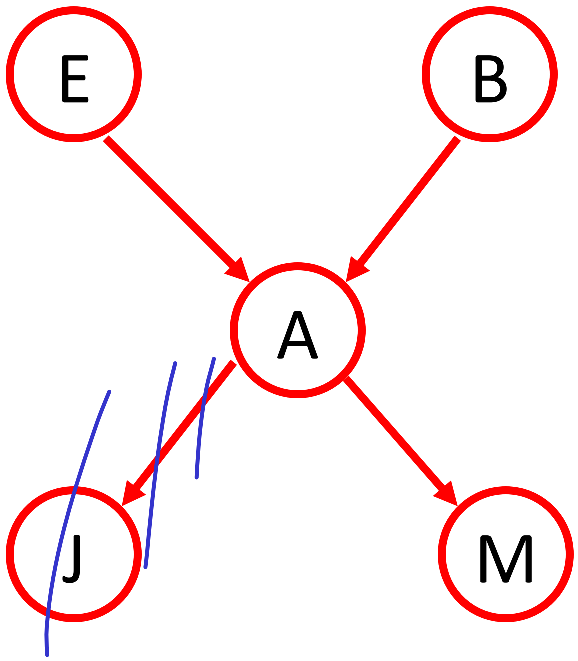
Intermediate solutions are distributions on fewer variables!

Variable elimination in general graphs

- Push sums through product as far as possible
- Create new factor by summing out variables

$$P(E,m) = \sum_{b,a,j} P(E)\,P(b)\,P(a|E,b)\,P(j|a)\,P(m|a)$$

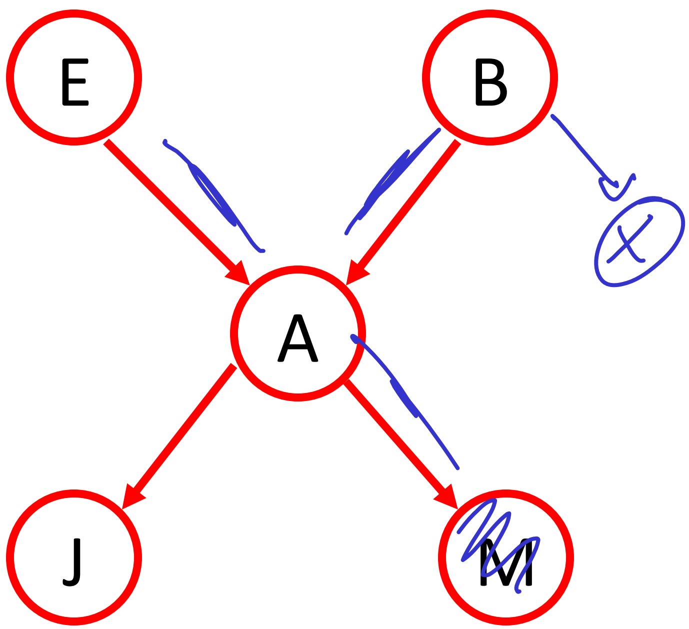
$$= P(E)\,\underbrace{\sum_b P(b)\,\underbrace{\sum_a P(a|E,b)\,P(m|a)\,\underbrace{\sum_j P(j|a)}_{1}}_{g_A(E,b,m)}}_{g_B(E,m)}$$



Removing irrelevant variables

$P(E, m) = \sum \dots$

$\underbrace{\sum_x P(x|b)}_{=1} \quad \underbrace{\sum_j P(j|A)}_{=1}$



Delete nodes not on active trail between query vars.

Variable elimination algorithm

- Given BN and Query P(X | **E**=**e**)
- Remove irrelevant variables for {X,**e**}
- Choose an ordering of $X_1,\ldots,X_n$
- Set up initial factors: $f_i = P(X_i \mid \mathbf{Pa_i})$
- For i =1:n, $X_i \notin \{X,\mathbf{E}\}$
  - Collect all factors f that include X_i
  - Generate new factor by marginalizing out X_i

$$g = \sum_{x_i} \prod_j f_j$$

  - Add g to set of factors
- Renormalize P(x,**e**) to get P(x | **e**)

Multiplying factors

$$g = \sum_{x_i} \prod_j f_j$$

B \ A	T	F
T	.	.
F	.	.

C \ B	T	F
T		
F		

$f_1(A,B)$, $f_2(B,C)$

$f' = f_1 \cdot f_2$

$f'(A,B,C)$

A \ BC	TT	TF	FT	FF
T		.	.	.
F				

$f'(A=T, B=F, C=F) = f_1(A=T, B=F) \cdot f_2(B=F, C=F)$

Marginalizing factors

$$g = \sum_{x_i} \underbrace{\prod_j f_j}_{f'}$$

$f'(A, B)$

A \ B	T	F
T	.	.
F	.	.

$\Rightarrow g = \sum_A f'(A, B)$

B	g(B)
T	
F	

$g(B) = f'(A=T, B) + f'(A=F, B)$

Tasks

- Read Koller & Friedman Chapter 17.4, 18.3-5, 19.1-3
- Homework 1 due in class Wednesday Oct 21