

Probabilistic Graphical Models

Lecture 2 – Bayesian Networks
Representation

CS/CNS/EE 155
Andreas Krause

Announcements

- Will meet in Steele 102 for now
- Still looking for another 1-2 TAs..
- Homework 1 will be out soon. Start early!! 😊

Multivariate distributions

- Instead of random variable, have random vector

$$\mathbf{X}(\omega) = [X_1(\omega), \dots, X_n(\omega)]$$

- Specify $P(X_1=x_1, \dots, X_n=x_n)$
- Suppose all X_i are Bernoulli variables.
- How many parameters do we need to specify?

Marginal distributions

- Suppose we have joint distribution $P(X_1, \dots, X_n)$
- Then

$$P(\underline{X_i = x_i}) = \sum_{\substack{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n}} P(\underline{\underline{x_1, \dots, x_n}})$$

- If all X_i binary: How many terms?

Rules for random variables

- Chain rule

$$P(x_1 \dots x_n) = P(x_1) P(Y_2|X_1) \dots P(X_n|X_1 \dots X_{n-1})$$

- Bayes' rule

$$P(x|y) = \frac{P(y|x) P(x)}{P(y)}$$

How do we get $P(y)$?

Key concept: Conditional independence

- Events α, β conditionally independent given γ if

$$P(\alpha \cap \beta | \gamma) = P(\alpha | \gamma) P(\beta | \gamma)$$

- Random variables X and Y cond. indep. given Z if
for all $x \in \text{Val}(X)$, $y \in \text{Val}(Y)$, $z \in \text{Val}(Z)$

$$\underbrace{P(X = x, Y = y | Z = z)}_{\text{P(X=x | Z=z) P(Y=y | Z=z)}} = \underbrace{P(X = x | Z = z)}_{\text{P(X=x | Z=z)}} \underbrace{P(Y = y | Z = z)}_{\text{P(Y=y | Z=z)}}$$

- If $P(Y=y | Z=z) > 0$, that's equivalent to

$$P(X = x | Z = z, Y = y) = P(X = x | Z = z)$$

Similarly for sets of random variables X, Y, Z

We write: $P \models \cancel{X \perp Y | Z}$

Why is conditional independence useful?

- $P(X_1, \dots, X_n) = P(X_1) P(X_2 | X_1) \dots P(\underbrace{X_n | X_1, \dots, X_{n-1}}_{n-1})$

How many parameters?

$$2^0 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1$$

- Now suppose $X_1 \dots X_{i-1} \perp X_{i+1} \dots X_n | X_i$ for all i

Then

$$P(X_1, \dots, X_n) = \underbrace{P(X_1)}_1 \cdot \underbrace{P(X_2 | X_1)}_2 \cdot \underbrace{P(X_3 | X_2)}_2 \cdot \dots \cdot \underbrace{P(X_n | X_{n-1})}_2$$

$$2^{n-1} < 2^n$$

How many parameters?

Exponential reduction in # params

- Can we compute $P(X_n)$ more efficiently?

Yes (often)

Properties of Conditional Independence

- **Symmetry**

- $X \perp Y | Z \Rightarrow Y \perp X | Z$

- **Decomposition**

- $X \perp Y, W | Z \Rightarrow X \perp Y | Z$

- **Contraction**

"Converse Decomposition"

- $(X \perp Y | Z) \wedge (X \perp W | Y, Z) \Rightarrow X \perp Y, W | Z$

- **Weak union**

- $X \perp Y, W | Z \Rightarrow X \perp Y | Z, W$

- **Intersection**

- $(X \perp Y | Z, W) \wedge (X \perp W | Y, Z) \Rightarrow X \perp Y, W | Z$

- Holds only if distribution is positive, i.e., $P > 0$

Key questions

- How do we specify distributions that satisfy particular independence properties?
 → Representation
- How can we exploit independence properties for efficient computation?
 → Inference
- How can we identify independence properties present in data?
 → Learning

Will now see example: Bayesian Networks

Key idea

- Conditional parameterization
(instead of joint parameterization)
- For each RV, specify $P(X_i | \underline{X_A})$ for set X_A of RVs
- Then use chain rule to get joint parametrization

$$P(x_1 \dots x_m) = \prod P(X_i | X_{A_i})$$

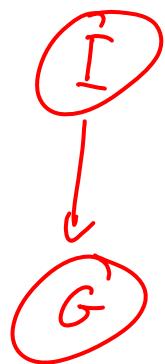
- Have to be careful to guarantee legal distribution...

$$P(X|Y), P(Y|X)$$

Does there exist $P(X_i, Y)$ with above ~~dist~~ cond. distributions

Not in general

Example: 2 variables

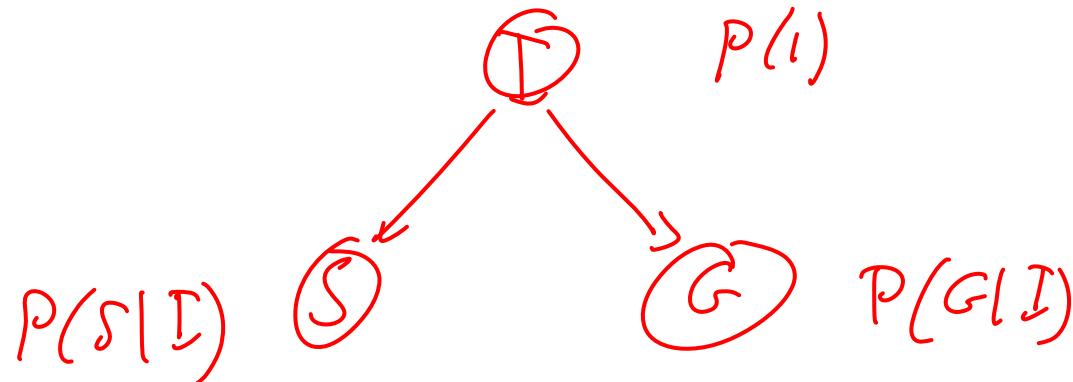


$$P(I = VH) = 0.8$$

$$P(G|I)$$

| | | |
|----|------------|------------|
| | A | B |
| VH | 0.8 | 0.2 |
| H | 0.6 | 0.4 |
| | $\sum_i=1$ | $\sum_j=1$ |

Example: 3 variables



$$P(I, S, G) = P(I) P(G|I) P(S|I)$$

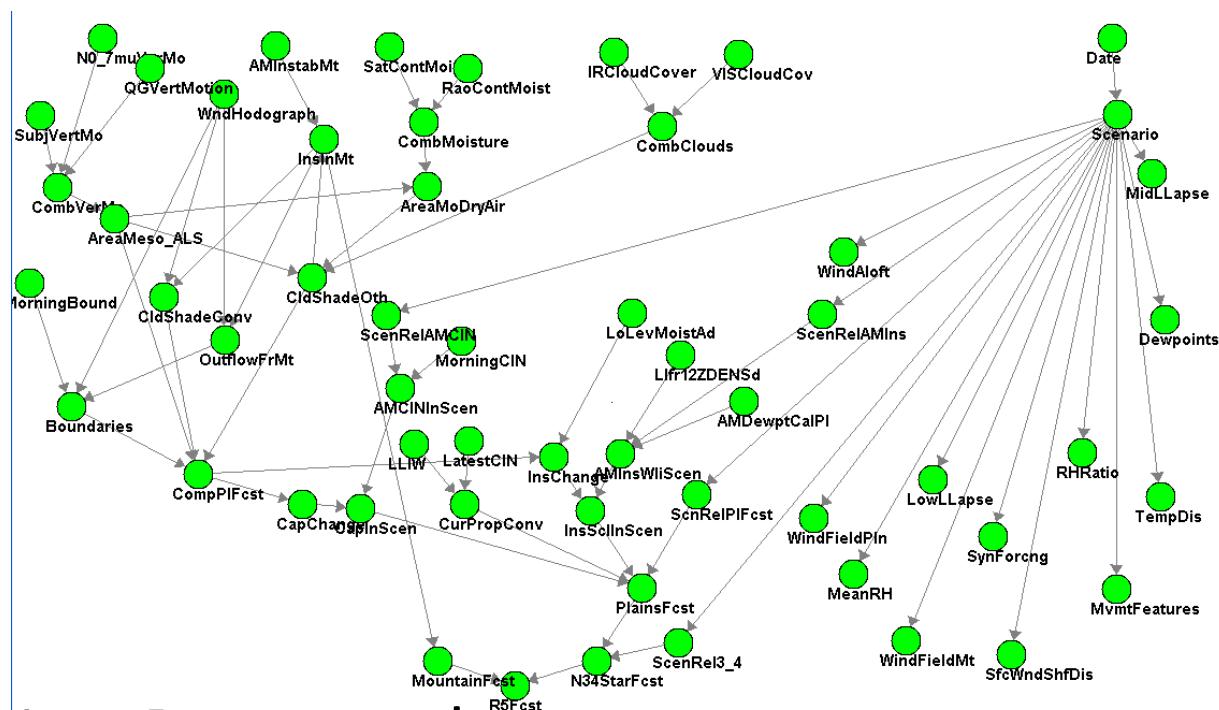
Example: Naïve Bayes models

- Class variable Y
- Evidence variables X_1, \dots, X_n
- Assume that $X_A \perp X_B \mid Y$
for all subsets X_A, X_B of $\{X_1, \dots, X_n\}$
- Conditional parametrization:
 - Specify $P(Y)$
 - Specify $P(X_i \mid Y)$
- Joint distribution

$$P(x_1, \dots, x_n, y) = P(y) \prod_i P(x_i \mid y)$$

Today: Bayesian networks

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



HailFinder

56 vars

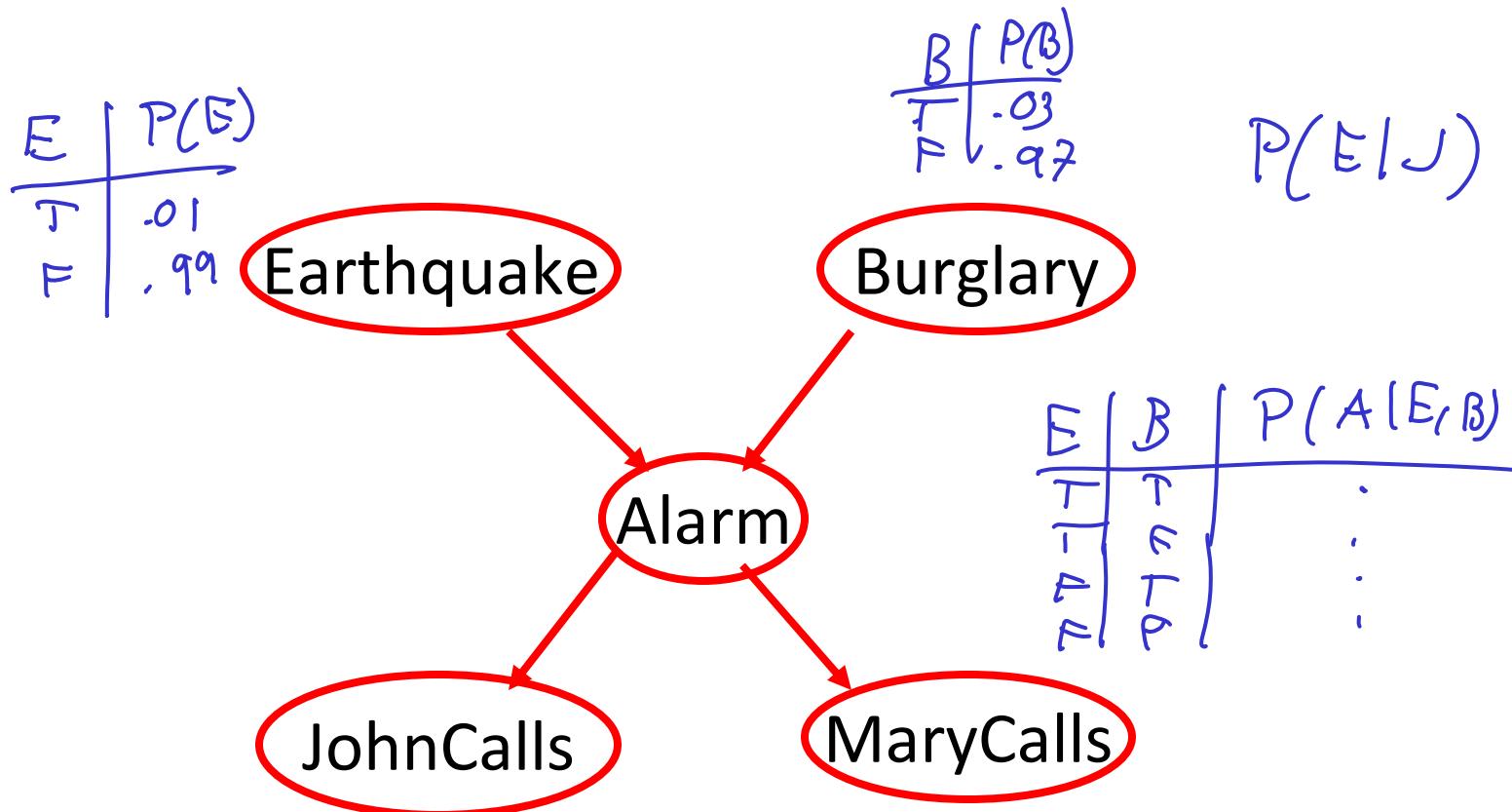
~ 3 states each

$\rightarrow \sim 10^{26}$ terms
> 10.000 years
on Top
supercomputers

JavaBayes applet

Causal parametrization

- Graph with directed edges from (immediate) causes to (immediate) effects



Bayesian networks

- A Bayesian network structure is a directed, acyclic graph G, where each vertex s of G is interpreted as a random variable X_s (with unspecified distribution)



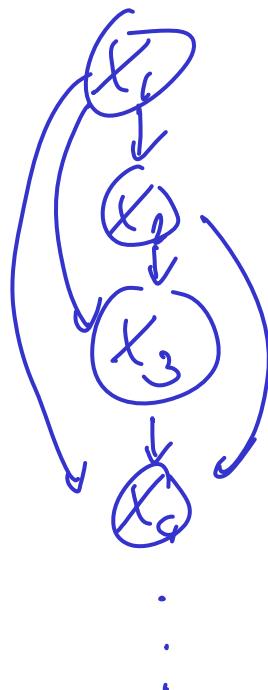
- A **Bayesian network** (G, P) consists of
 - A BN structure G and ..
 - ..a set of conditional probability distributions (CPDs)
 $P(X_s \mid \text{Pa}_{X_s})$, where Pa_{X_s} are the parents of node X_s such that
 - (G, P) defines joint distribution

$$P(X_1, \dots, X_n) = \prod_i P(X_i \mid \text{Pa}_{X_i})$$

Bayesian networks

- Can every probability distribution be described by a BN?

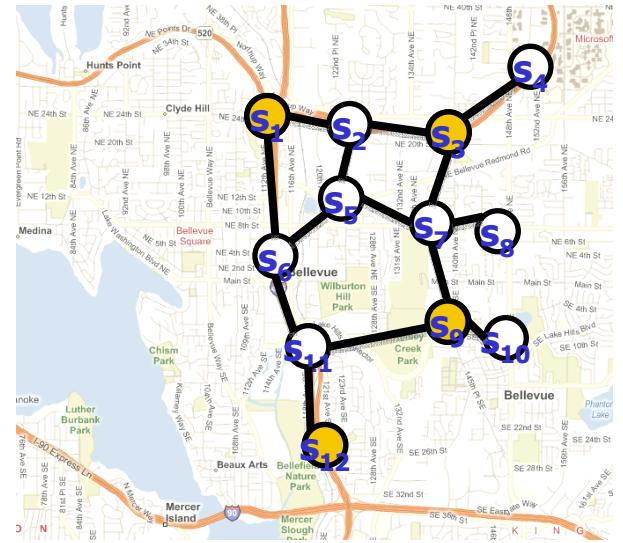
$$P(x_1, \dots, x_n) = P(x_1) P(x_2 | x_1) \dots P(x_n | x_1, \dots, x_{n-1})$$



Representing the world using BNs



represent

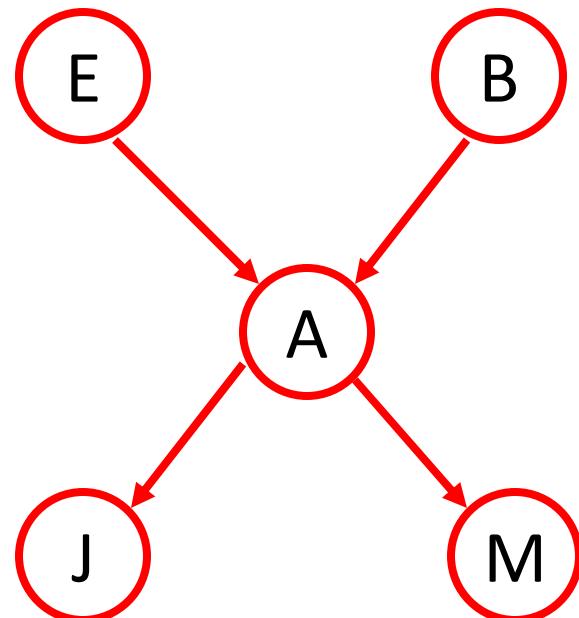


True distribution $\underline{P'}$
with cond. ind. $I(P')$

Bayes net (G, P)
with $I(P)$

- Want to make sure that $I(P) \subseteq I(P')$
- Need to understand CI properties of BN (G, P)

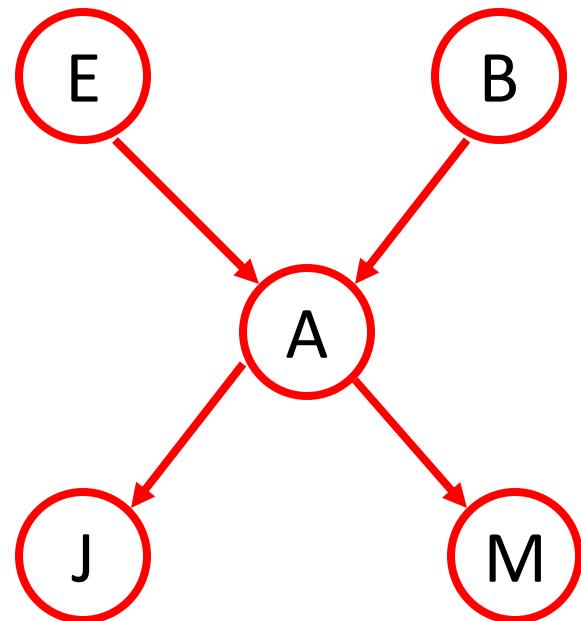
Which kind of CI does a BN imply?



$$E \perp B$$

$$\begin{aligned} P(E, B) &= \sum_{AJM} P(E, B, A, J, M) \\ &= \sum_{AJM} \frac{P(E) P(B) P(A|E, B) P(J|A) P(M|A)}{P(M|A)} \\ &= P(E) P(B) \sum_{AJM} P(A|E, B) P(J|A) P(M|A) \\ &= P(E) P(B) \end{aligned}$$

Which kind of CI does a BN imply?



$$\Rightarrow P(J|AM) = \frac{P(J, AM)}{P(AM)} = \frac{P(J|A) \cancel{P(A, M)}}{\cancel{P(A, M)}} = P(J|A)$$

$$J \perp M \mid A$$

$$P(J \mid AM) = \frac{P(J, A, M)}{\underline{P(A, M)}}$$

$$P(J, A, M) = \sum_{E, B} P(J, A, M, E, B)$$

$$\Rightarrow \sum_{E, B} P(E) P(B) P(A \mid E, B) P(J \mid A) P(M \mid A)$$

$$= P(J \mid A) P(M \mid A) \underbrace{\sum_{E, B} P(E) P(B) P(A \mid E, B)}_{= P(A)} \underbrace{\Rightarrow P(M, A)}$$

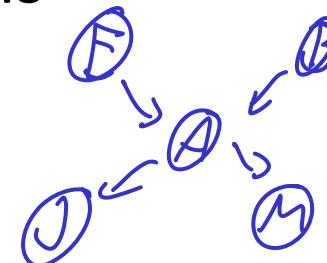
四

Local Markov Assumption

- Each BN Structure G is associated with the following conditional independence assumptions

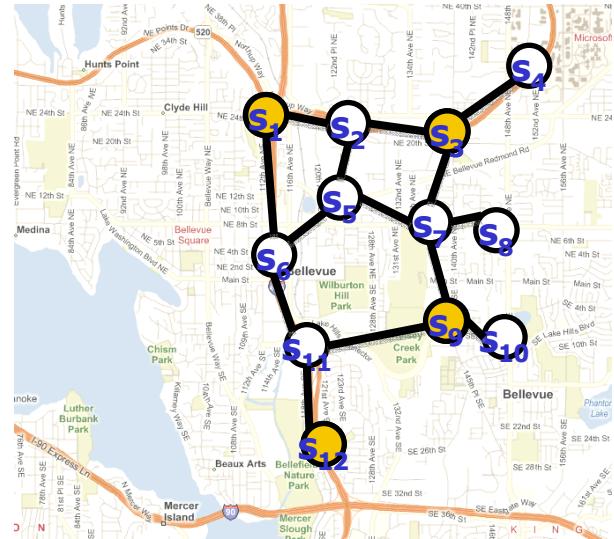
$$J \perp B \mid A$$

$$X \perp \text{NonDescendents}_X \mid \text{Pa}_X$$



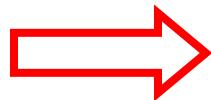
- We write $I_{loc}(G)$ for these conditional independences
- Suppose (G, P) is a Bayesian network representing P
Does it hold that $I_{loc}(G) \subseteq I(P)$?
If this holds, we say G is an I-map for P .

Factorization Theorem



$$I_{loc}(G) \subseteq I(P)$$

G is an **I-map** of P
(independence map)



True distribution P
can be represented exactly as

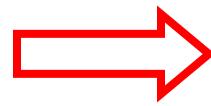
$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

i.e., P can be represented as
a Bayes net (G,P)

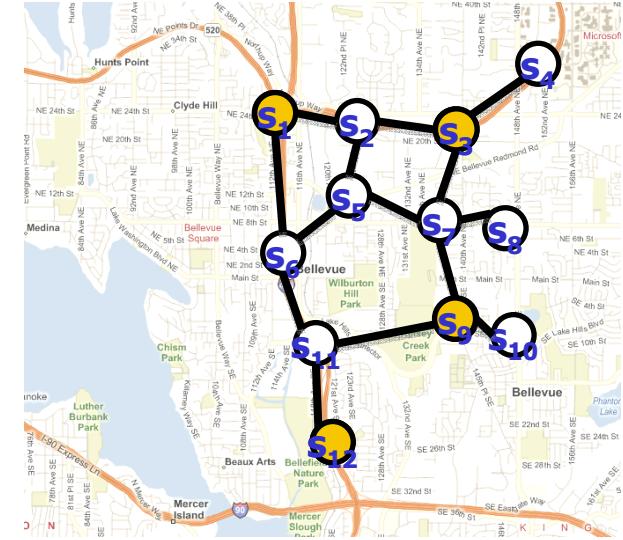
Factorization Theorem



True distribution P
can be represented exactly as
a Bayes net (G, P)

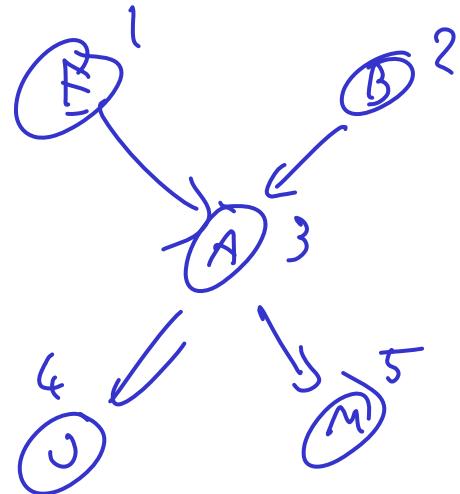


$$P(X_1, \dots, X_n) = \prod_i P(X_i \mid \text{Pa}_{X_i})$$



G is an **I-map** of P
(independence map)

Proof: I-Map to factorization



Ordering $\pi : \{1 \dots n\} \rightarrow \{1 \dots n\}$ topological

If: $\forall X_j$: descendant of X_i

$$\pi(j) > \pi(i)$$

Can find topological ordering in linear time

$$P(X_1 \dots X_n) = \prod_{i=1}^n P(X_{\pi(i)} | X_{\pi(1)} \dots X_{\pi(i-1)})$$

Chain rule

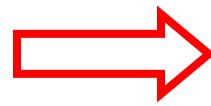
$$P(X_{\pi(i)} | \text{Pa}_{X_{\pi(i)}})$$

□

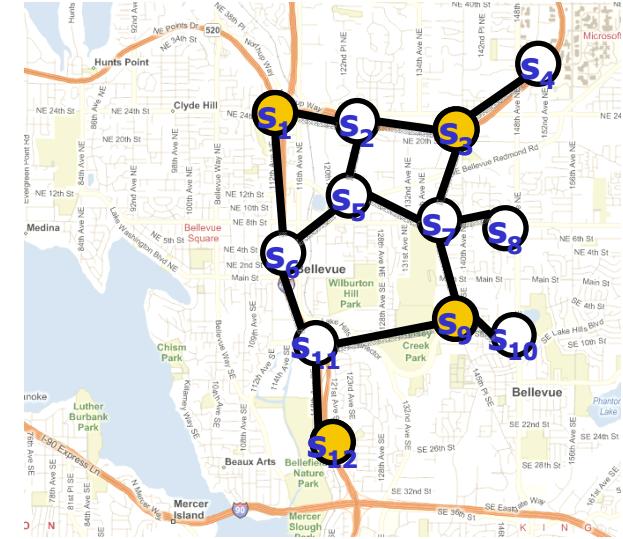
Factorization Theorem



True distribution P
can be represented exactly as
a Bayes net (G, P)



$$P(X_1, \dots, X_n) = \prod_i P(X_i \mid \text{Pa}_{X_i})$$



G is an **I-map** of P
(independence map)

The general case

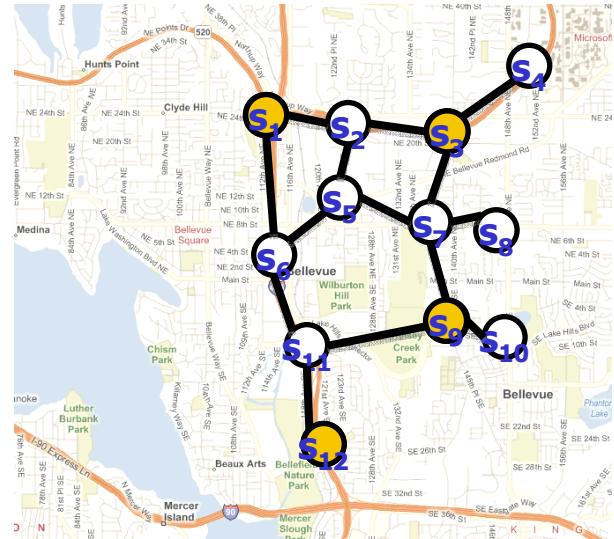
$$P(X_1, \dots, X_m) = \prod P(X_i | \text{Pa}_{X_i}) \stackrel{?}{=} \forall X_i: \forall N \subseteq \text{Nondesc}_{X_i} \\ X_i \perp N | \text{Pa}_{X_i}$$

$Y = \text{Pa}_X;$
 $Z = \text{Nondesc}_{X_i} \setminus (Y \cup N)$
 $\text{② } D = \text{Desc}(X_i)$

$P(X_i | Y, N) = \frac{P(X_i, Y, N)}{P(Y, N)}$
 $P(X_i, Y, N) = \sum_{Z, D} P(X_i, Y, N, Z, D)$
 $= \sum_{Z, D} P(X_i | Y) \prod_{x \in D} P(x | \text{Pa}_x) \prod_{x' \in (N \cup Z \cup Y)} P(x' | \text{Pa}_{x'})$
 $= P(X_i | Y) \underbrace{\sum_{Z} \prod_{x' \in (N \cup Z \cup Y)} P(x' | \text{Pa}_{x'})}_{(*)} \underbrace{\sum_{D} \prod_{x \in D} P(x | \text{Pa}_x)}_{=1} = 1$

$P(Y, N) = \sum_{X_i} P(X_i, Y, N) = \sum_{Z} \prod_{x' \in (N \cup Z \cup Y)} P(x' | \text{Pa}_{x'}) \sum_{X_i} P(X_i | Y) = 1$
 $\Rightarrow P(X_i | Y, N) = P(X_i | Y) \quad \square$

Factorization Theorem



$$I_{loc}(G) \subseteq I(P)$$

G is an I-map of P
(independence map)



True distribution P
can be represented exactly as
Bayesian network (G, P)

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

Defining a Bayes Net

- Given random variables and known conditional independences
- Pick ordering $\underline{X_1, \dots, X_n}$ of the variables
- For each X_i
 - Find minimal subset $A \subseteq \{X_1, \dots, X_{i-1}\}$ such that $X_i \perp X_{\neg A} \mid A$,
where $\neg A = \{X_1, \dots, X_n\} \setminus A$
Ensure local Markov property holds!
 - Specify / learn $\text{CPD}(X_i \mid A)$
A parents of X_i
- Ordering matters a lot for compactness of representation! More later this course.

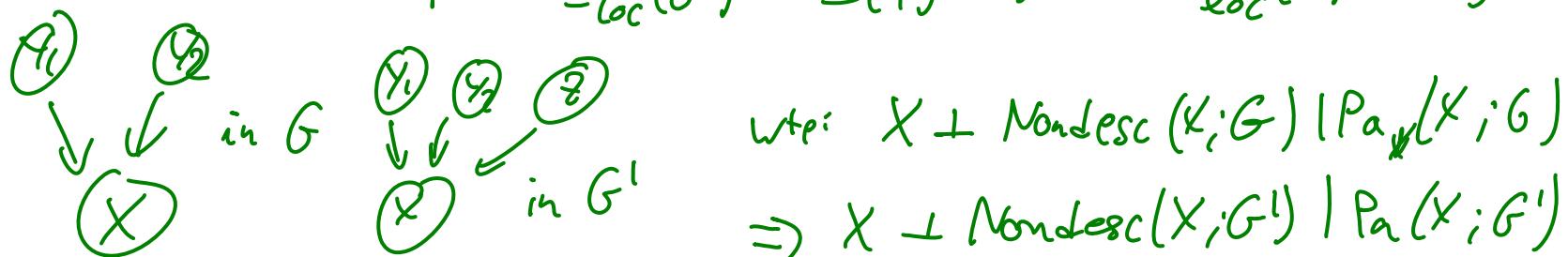
Adding edges doesn't hurt

Theorem:

Let G be an I-Map for P , and G' be derived from G by adding an edge. Then G' is an I-Map of P
(G' is strictly more expressive than G)

- **Proof :** wtp: $I_{loc}(G') \subseteq I_{loc}(G)$

Then $I_{loc}(G') \subseteq I(P)$ since $I_{loc}(G) \subseteq I(P)$



wtp: $X \perp \text{Nondesc}(Y; G) \mid \text{Pa}_Y(X; G)$

$\Rightarrow X \perp \text{Nondesc}(Y; G') \mid \text{Pa}_Y(X; G')$

$X \perp N, Y \mid Z \Rightarrow X \perp N \mid Y, Z$

holds b/c: Weak Union property of C.I. D₂₉

Additional conditional independencies

- BN specifies joint distribution through conditional parameterization that satisfies Local Markov Property
- But we also talked about additional properties of CI
 - Weak Union, Intersection, Contraction, ...
- Which additional CI does a particular BN specify?
 - All CI that can be derived through algebraic operations

Local Markov prop. $I_{loc}(G) \subseteq I(G)$

What you need to know

- Bayesian networks
- Local Markov property
- I-Maps
- Factorization Theorem

Tasks

- Subscribe to Mailing list
<https://utils.its.caltech.edu/mailman/listinfo/cs155>
- Read Koller & Friedman Chapter 3.1-3.3
- Form groups and think about class projects. If you have difficulty finding a group, email Pete Trautman

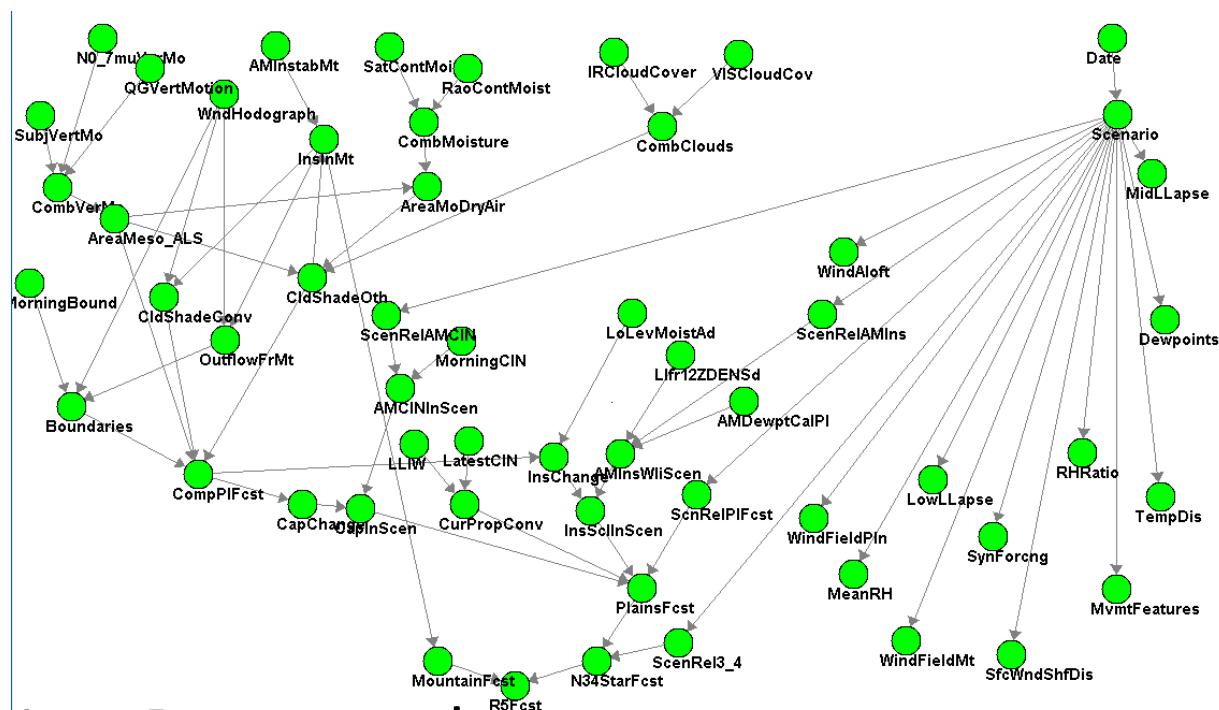
Probabilistic Graphical Models

Lecture 3 – Bayesian Networks
Semantics

CS/CNS/EE 155
Andreas Krause

Bayesian networks

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



HailFinder

56 vars

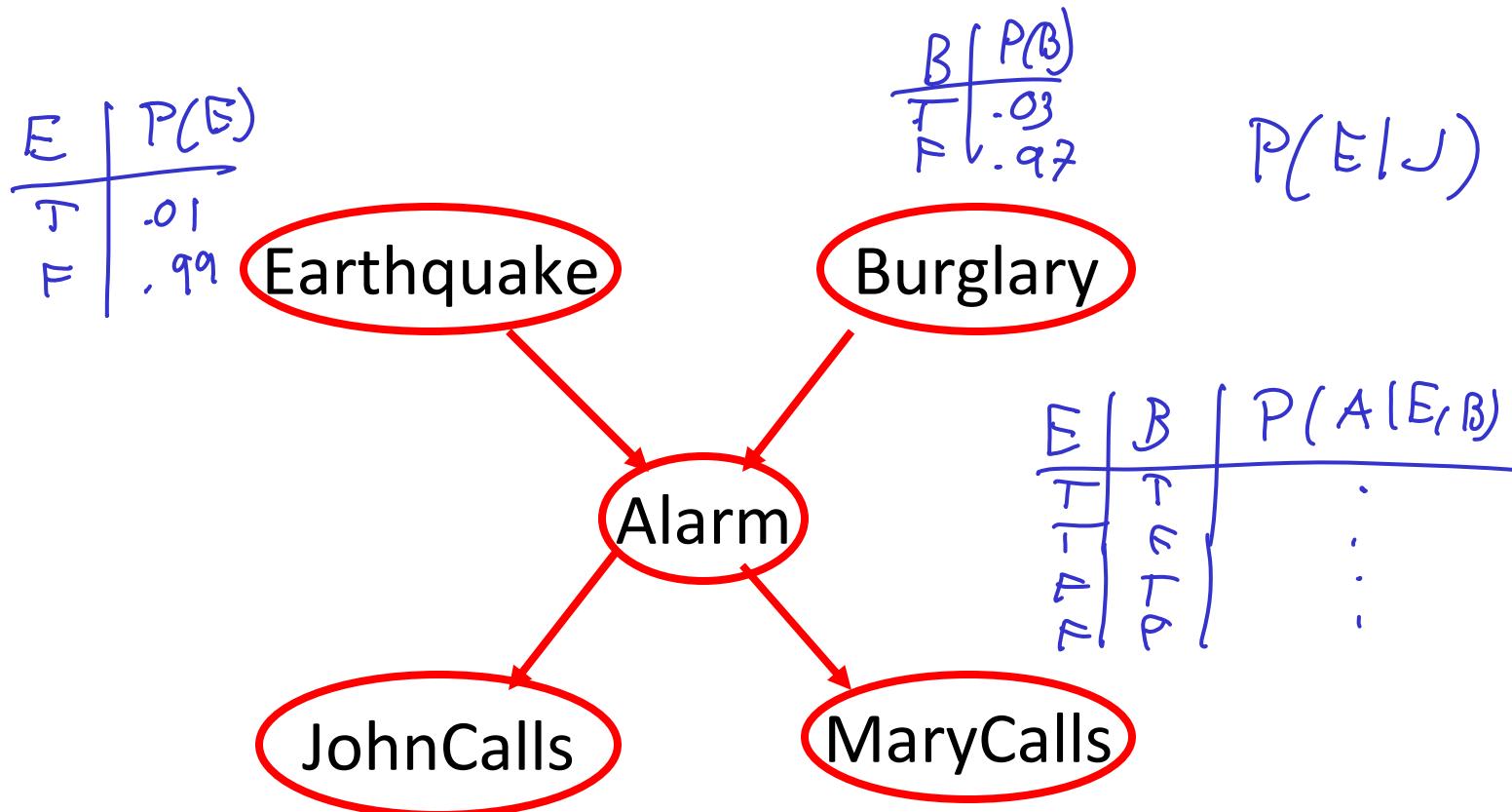
~ 3 states each

$\rightarrow \sim 10^{26}$ terms
> 10.000 years
on Top
supercomputers

JavaBayes applet

Causal parametrization

- Graph with directed edges from (immediate) causes to (immediate) effects



Bayesian networks

- A Bayesian network structure is a directed, acyclic graph G, where each vertex s of G is interpreted as a random variable X_s (with unspecified distribution)



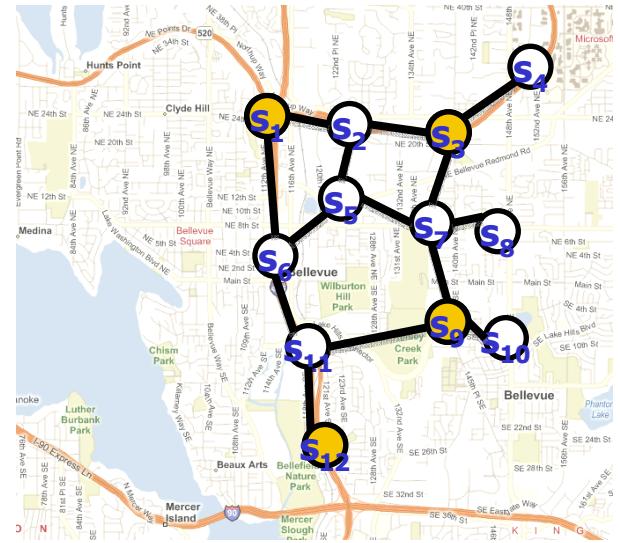
- A **Bayesian network** (G, P) consists of
 - A BN structure G and ..
 - ..a set of conditional probability distributions (CPDs)
 $P(X_s \mid \text{Pa}_{X_s})$, where Pa_{X_s} are the parents of node X_s such that
 - (G, P) defines joint distribution

$$P(X_1, \dots, X_n) = \prod_i P(X_i \mid \text{Pa}_{X_i})$$

Representing the world using BNs



represent



True distribution P'
with cond. ind. $I(P')$

Bayes net (G, P)
with $I(P)$

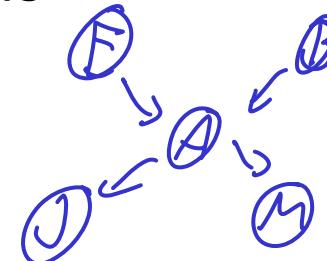
- Want to make sure that $I(P) \subseteq I(P')$
- Need to understand CI properties of BN (G, P)

Local Markov Assumption

- Each BN Structure G is associated with the following conditional independence assumptions

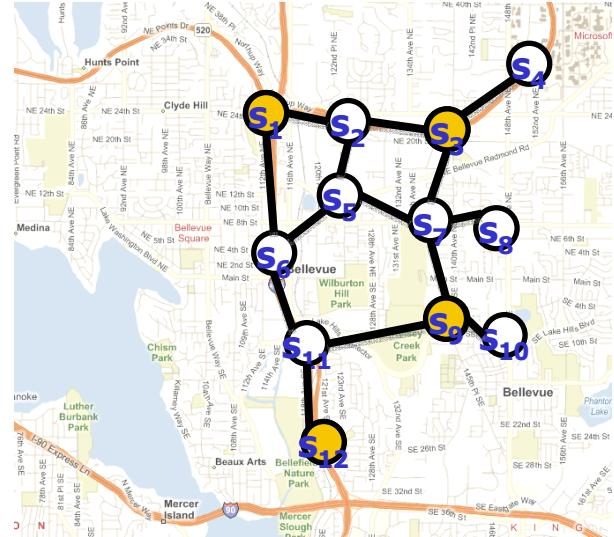
$$J \perp B \mid A$$

$$X \perp \text{NonDescendents}_X \mid \text{Pa}_X$$



- We write $I_{loc}(G)$ for these conditional independences
- Suppose (G, P) is a Bayesian network representing P
Does it hold that $I_{loc}(G) \subseteq I(P)$?
If this holds, we say G is an I-map for P .

Factorization Theorem



$$I_{loc}(G) \subseteq I(P)$$

G is an I-map of P
(independence map)



True distribution P
can be represented exactly as
Bayesian network (G, P)

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

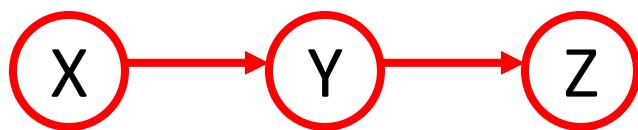
Additional conditional independencies

- BN specifies joint distribution through conditional parameterization that satisfies Local Markov Property
 $I_{loc}(G) = \{(X_i \perp \text{Nondescendants}_{X_i} \mid Pa_{X_i})\}$
- But we also talked about additional properties of CI
 - Weak Union, Intersection, Contraction, ...
- Which additional CI does a particular BN specify?
 - All CI that can be derived through algebraic operations

→ proving CI is very cumbersome!!

Is there an easy way to find all independences of a BN just by looking at its graph??

BNs with 3 nodes



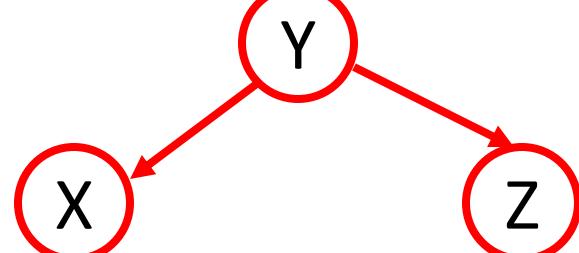
$$\begin{aligned} X \perp\!\!\!\perp Z | Y \\ \gamma(X \perp\!\!\!\perp Z) \end{aligned}$$

Local Markov Property:
 $X \perp\!\!\!\perp \text{NonDesc}(X) | \text{Pa}(X)$

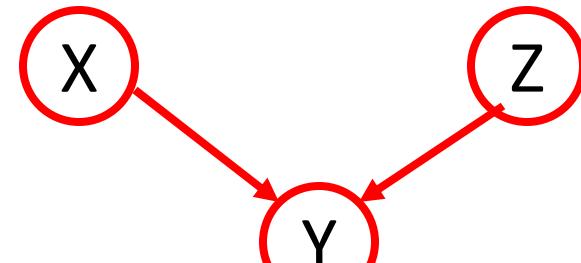


$$\begin{aligned} X \perp\!\!\!\perp Z | Y \\ \gamma(X \perp\!\!\!\perp Z) \end{aligned}$$

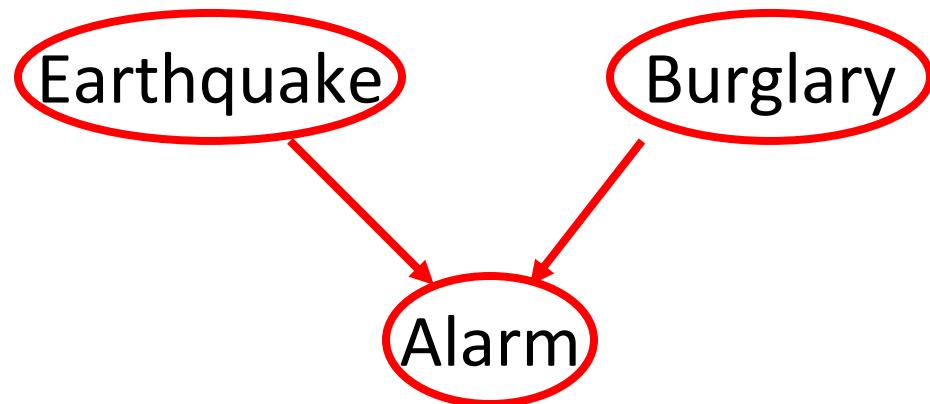
$$\begin{aligned} X \perp\!\!\!\perp Z \\ \gamma(X \perp\!\!\!\perp Z | Y) \end{aligned}$$



$$\begin{aligned} X \perp\!\!\!\perp Z | Y \\ \gamma(X \perp\!\!\!\perp Z) \end{aligned}$$



V-structures

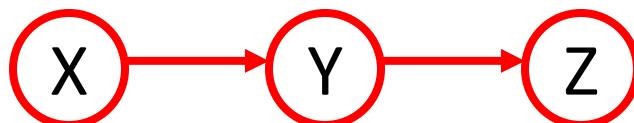


- Know $E \perp B$
- Suppose we know A. Does $E \perp B \mid A$ hold?

(can happen: $P(E = T \mid A = T, B = T) < P(E = T \mid A = T)$
Explaining away

BNs with 3 nodes

Indirect causal effect



Local Markov Property:
 $X \perp \text{NonDesc}(X) \mid \text{Pa}(X)$

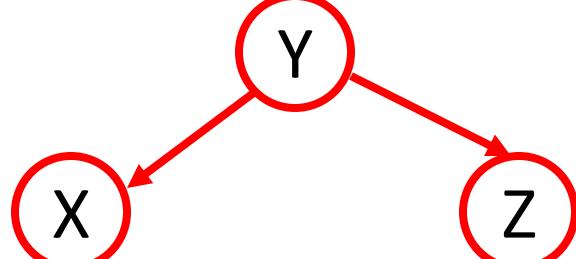
Indirect evidential effect



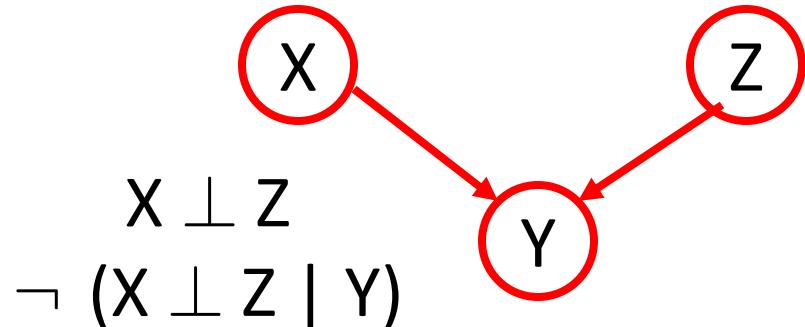
$$X \perp Z \mid Y$$

$$\neg(X \perp Z)$$

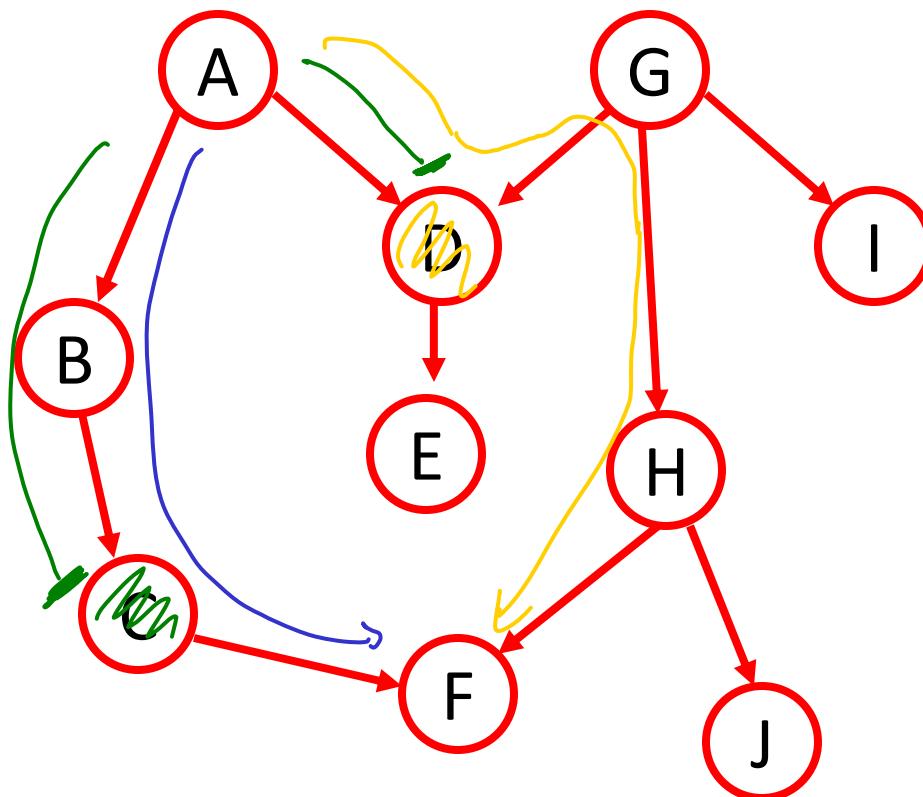
Common cause



Common effect

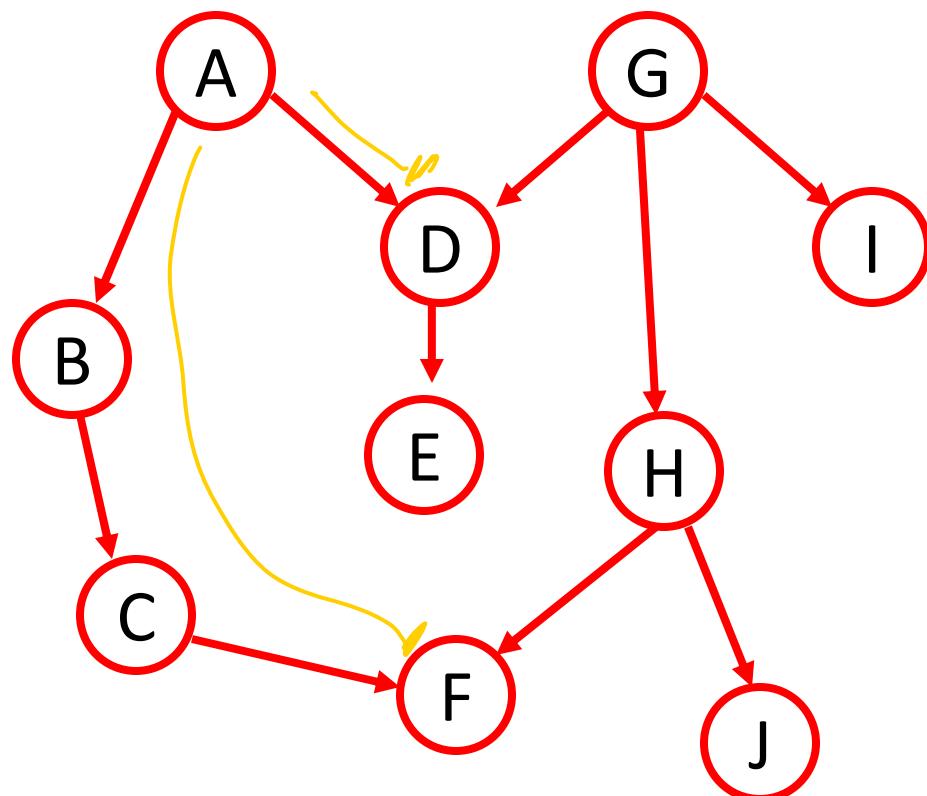


Examples



$A \perp F$ ✓
 $A \perp F | C$ ✓
 $A \perp F | C, D$ ✗

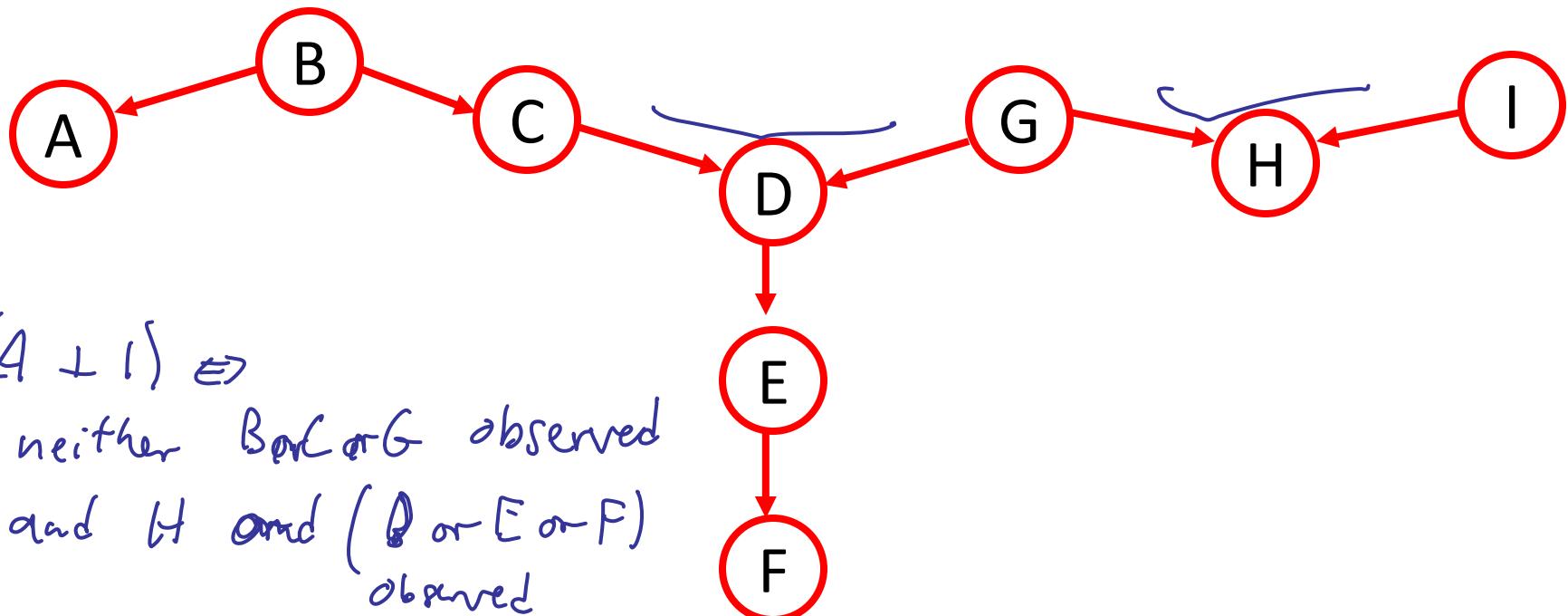
More examples



$A \perp G$
 $A \perp G | D \times$
 $A \perp G | E$

Active trails

- When are A and I independent?



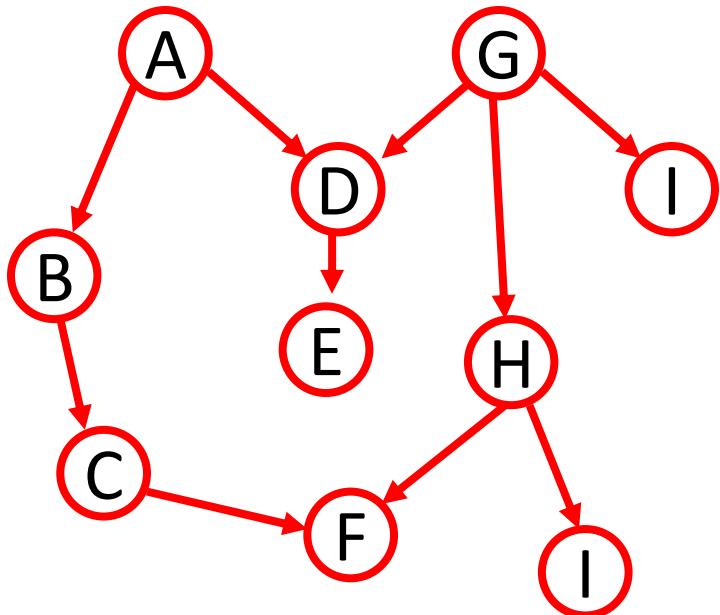
Active trails

- An undirected path in BN structure G is called **active trail** for observed variables $\mathbf{O} \subseteq \{X_1, \dots, X_n\}$, if for every consecutive triple of vars X, Y, Z on the path

- blocked
if
 $Y \in \mathbf{O}$*
- $X \rightarrow Y \rightarrow Z$ and Y is unobserved ($Y \notin \mathbf{O}$)
 - $X \leftarrow Y \leftarrow Z$ and Y is unobserved ($Y \notin \mathbf{O}$)
 - $X \leftarrow Y \rightarrow Z$ and Y is unobserved ($Y \notin \mathbf{O}$)
 - $X \rightarrow Y \leftarrow Z$ and Y or any of Y 's descendants is observed

- Any variables X_i and X_j for which \nexists active trail for observations \mathbf{O} are called d-separated by \mathbf{O}
We write $d\text{-sep}(X_i; X_j | \mathbf{O})$
- Sets \mathbf{A} and \mathbf{B} are d-separated given \mathbf{O} if $d\text{-sep}(X, Y | \mathbf{O})$ for all $X \in \mathbf{A}, Y \in \mathbf{B}$. Write $d\text{-sep}(\mathbf{A}; \mathbf{B} | \mathbf{O})$

d-separation and independence



Theorem:

$$\text{d-sep}(X;Y \mid Z) \rightarrow X \perp Y \mid Z$$

i.e., X cond. ind. Y given Z
if there does not exist
any active trail
between X and Y
for observations Z

- Proof uses algebraic properties of conditional independence

Soundness of d-separation

- Have seen: P factorizes according to $G \Leftrightarrow I_{loc}(G) \subseteq I(P)$
- Define $I(G) = \{(X \perp Y \mid Z) : d\text{-sep}_G(X; Y \mid Z)\}$
- **Theorem:** Soundness of d-separation
 P factorizes over $G \rightarrow I(G) \subseteq I(P)$
- Hence, d-separation captures only true independences
- How about $I(G) = I(P)$?

Does the converse hold?

Suppose P factorizes over G .

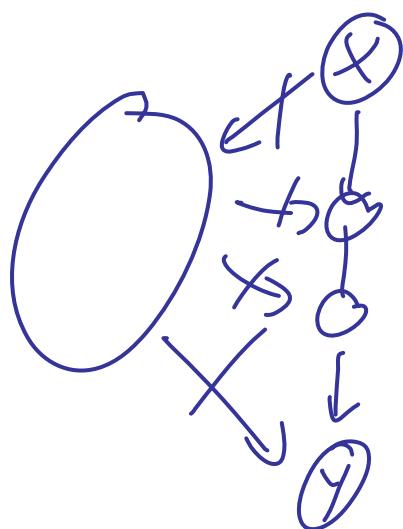
Does it hold that $I(P) \subseteq I(G)$?

$$P \vdash X \perp Y \quad I(P) = \{ (X \perp Y) \}$$

$$G: \bigotimes \rightarrow \bigotimes \quad I(G) = \{ \}$$

Existence of dependences for non-d-separated variables

- **Theorem:** If X and Y are not d-separated given Z , then there exists some distribution P factorizing over G in which X and Y are dependent given Z
- **Proof sketch:**



Pick active trail
Parameterize CPDs along trail
to create dependence
Everything else set to independent
to avoid cancelling dependencies

Completeness of d-separation

- Theorem: For “almost all” distributions P that factorize over G it holds that $I(G) = I(P)$
 - “almost all”: except for a set of distributions with measure 0, assuming only that no finite set of distributions has measure > 0



$$P(X=T) = p$$

$$P(Y=T | X=T) = r$$

$$P(Y=T | X=F) = q$$

$$P(Y|X) = P(Y)$$

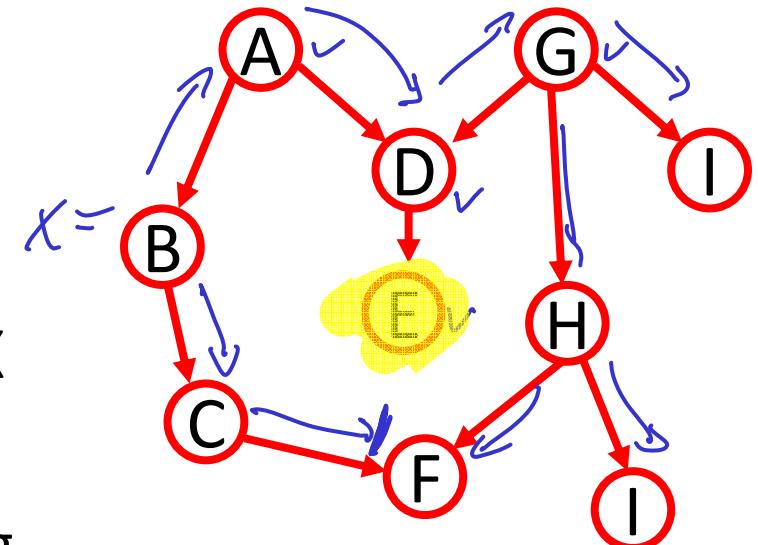
$$P(Y=T | X=T) = P(Y=T)$$

$$r = rp + q(1-p) \Rightarrow r(1-p) = q(1-p)$$

happens with prob. 0

Algorithm for d-separation

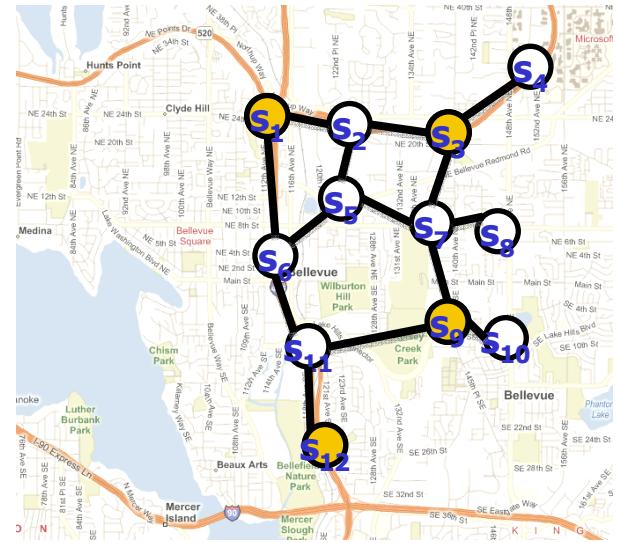
- How can we check if $X \perp Y | Z$?
 - Idea: Check every possible path connecting X and Y and verify conditions
 - Exponentially many paths!!! 😞
- Linear time algorithm:
Find all nodes reachable from X
 - 1. Mark **Z** and its ancestors
 - 2. Do breadth-first search starting from X; stop if path is blocked
 - Have to be careful with implementation details (see reading)



Representing the world using BNs



represent



True distribution P'
with cond. ind. $I(P')$

Bayes net (G, P)
with $I(P)$

- Want to make sure that $I(P) \subseteq I(P')$
- Ideally: $\underline{I(P) = I(P')}$
- Want BN that **exactly** captures independencies in P' !

Minimal I-maps

- Lemma: Suppose G' is derived from G by adding edges
- Then $I(G') \subseteq I(G)$
- Proof:

$$I_{loc}(G') \leq I_{loc}(G)$$

Completeness: $I(G) = \{ \text{all } Cl \text{s derivable from } I_{loc}(G) \text{ using } Cl \text{ properties} \}$

$$\Rightarrow I(G') \leq I(G) \quad \text{D}$$

- Thus, want to find graph G with $I(G) \subseteq I(P)$ such that when we remove any single edge, for the resulting graph G' it holds that $I(G') \not\subseteq I(P)$
- Such a graph G is called **minimal I-map**

Existence of Minimal I-Maps

- Does every distribution have a minimal I-Map?

Yes: Start with full graph G , $I(G) = \emptyset$
Keep removing edges as long as
 $I(G) \subseteq I(P)$

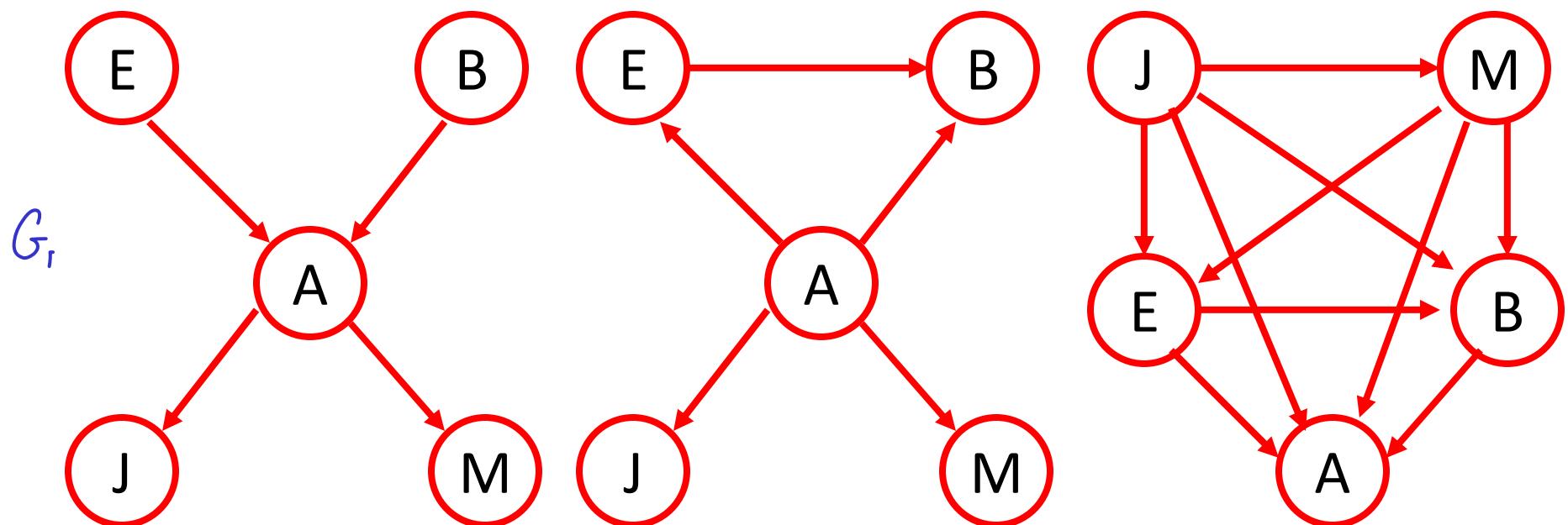
Algorithm for finding minimal I-map

- Given random variables and known conditional independences
- Pick ordering $\underline{X_1, \dots, X_n}$ of the variables
- For each X_i
 - Find minimal subset $A \subseteq \{X_1, \dots, X_{i-1}\}$ such that
 $P(X_i | \underline{X_1, \dots, X_{i-1}}) = P(X_i | A)$
 - Specify / learn CPD $P(X_i | A)$
- Will produce minimal I-map!

Uniqueness of Minimal I-maps

- Is the minimal I-Map unique?

$$I(P) = I(G_i)$$



Perfect maps

- Minimal I-maps are easy to find, but can contain many unnecessary dependencies.
- A BN structure G is called **P-map** (perfect map) for distribution P if $I(G) = I(P)$
- Does every distribution P have a P-map?

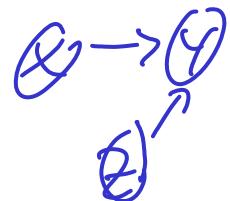
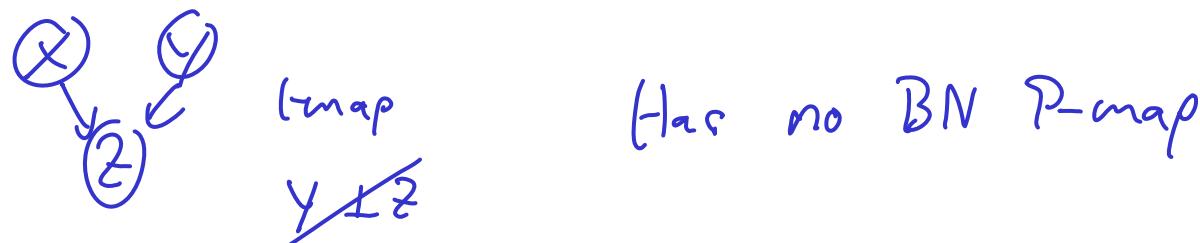
Existence of perfect maps

$$X, Y \sim \text{Ber}(0.5)$$

$$Z = X \text{ XOR } Y$$

$$X \perp Y, Y \perp Z, Z \perp X$$

$$\gamma(X \perp Y \perp Z)$$

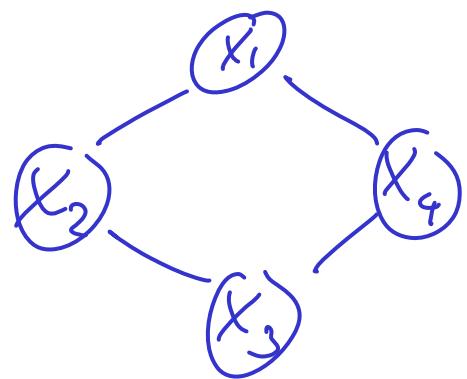


Existence of perfect maps

x_1, \dots, x_q

$x_1 \perp x_3 \mid x_2, x_q$

$x_2 \perp x_q \mid x_1, x_3$



Undirected GM
is a P-map

but NO BN is P-map

Uniqueness of perfect maps

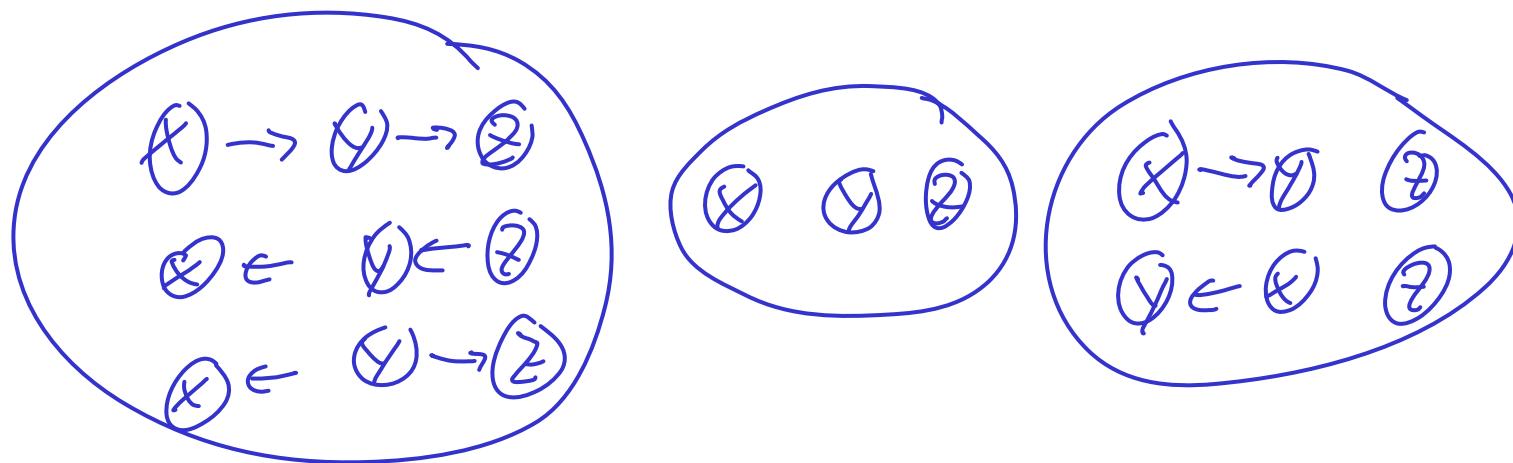
$$\emptyset \rightarrow \emptyset \quad G_1$$

$$I(G_1) = I(G_2)$$

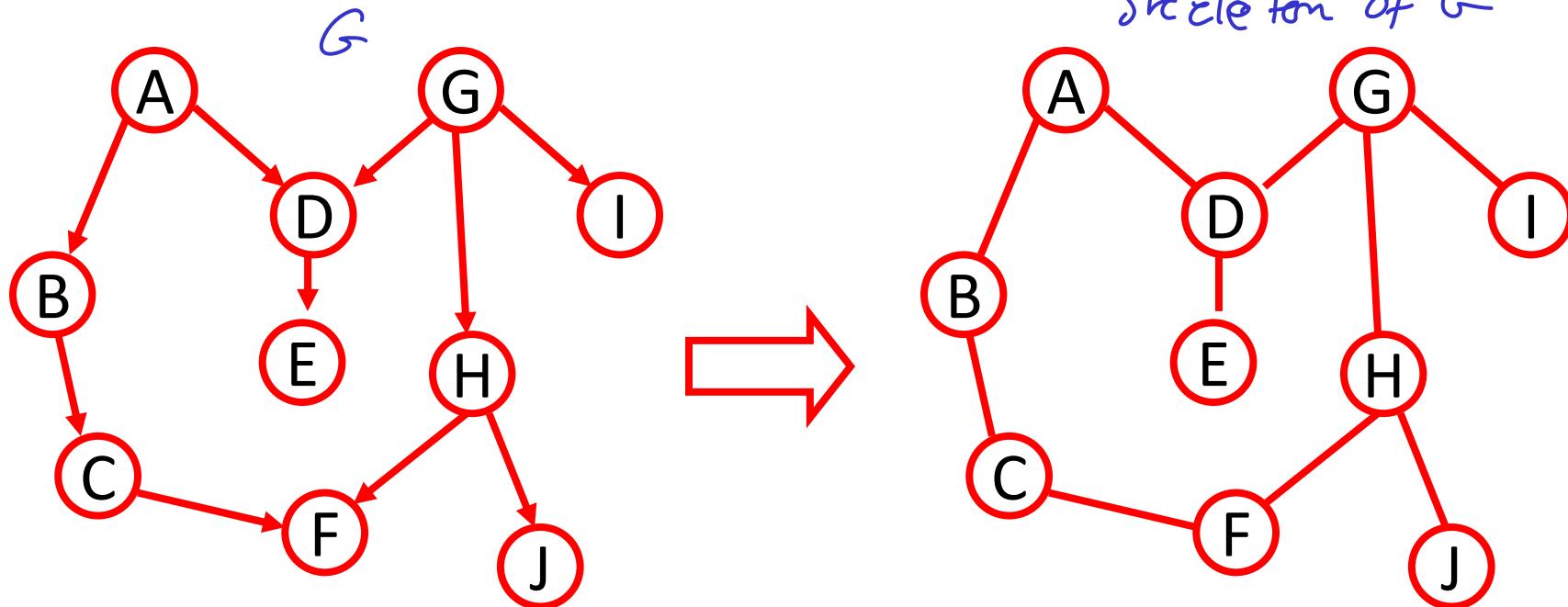
$$\emptyset \leftarrow \emptyset \quad G_2$$

I-Equivalence

- Two graphs G, G' are called I-equivalent if $I(G) = I(G')$
- I-equivalence partitions graphs into equivalence classes

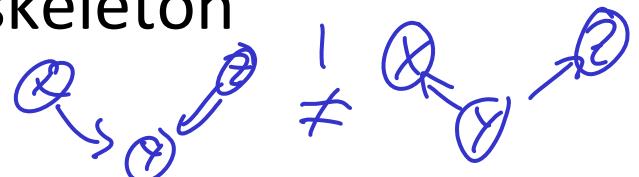


Skeletons of BNs



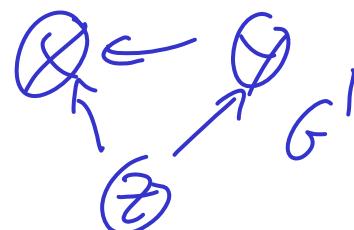
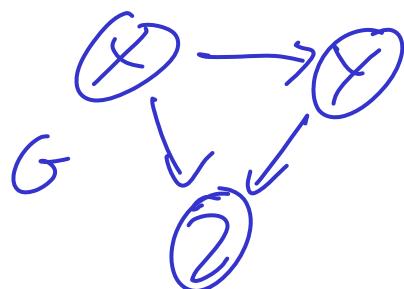
- I-equivalent BNs must have same skeleton

Some skeleton $\not\Rightarrow$ I-equiv.



Importance of V-structures

- **Theorem:** If G, G' have same skeleton and same V-structure, then $I(G) = I(G')$
- Does the converse hold?



$$I(G) = \{\} \quad = \quad I(G') = \{\}$$

Some skeleton, Not same V-structures

Immoralities and I-equivalence

- A V-structure $X \rightarrow Y \leftarrow Z$ is called **immoral** if there is no edge between X and Z (“unmarried parents”)



- **Theorem:** $I(G) = I(G') \Leftrightarrow G$ and G' have the same skeleton and the same immoralities.

Tasks

- Subscribe to Mailing list
<https://utils.its.caltech.edu/mailman/listinfo/cs155>
- Read Koller & Friedman Chapter 3.3-3.6
- Form groups and think about class projects. If you have difficulty finding a group, email Pete Trautman
- Homework 1 out tonight, due in 2 weeks. Start early!

Probabilistic Graphical Models

Lecture 4 – Learning
Bayesian Networks

CS/CNS/EE 155
Andreas Krause

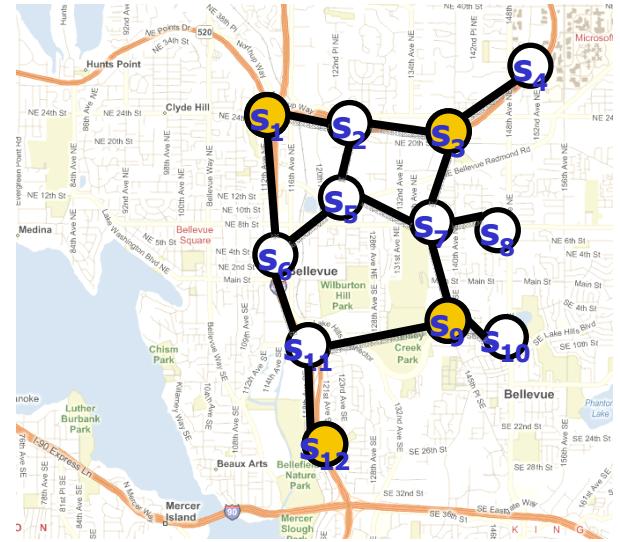
Announcements

- Another TA: Hongchao Zhou
- Please fill out the questionnaire about recitations
- Homework 1 out. Due in class Wed Oct 21
- Project proposals due Monday Oct 19

Representing the world using BNs



represent

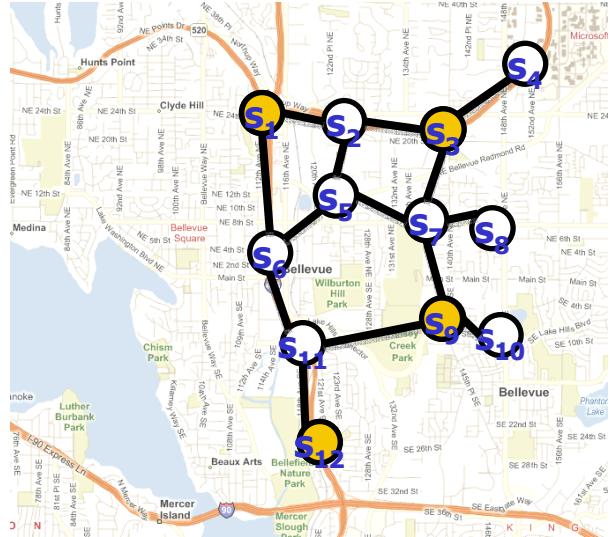


True distribution P'
with cond. ind. $I(P')$

Bayes net (G, P)
with $I(P)$

- Want to make sure that $I(P) \subseteq I(P')$
- Need to understand CI properties of BN (G, P)

Factorization Theorem



$$I_{loc}(G) \subseteq I(P)$$

G is an I-map of P
(independence map)



True distribution P
can be represented exactly as
Bayesian network (G, P)

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

Additional conditional independencies

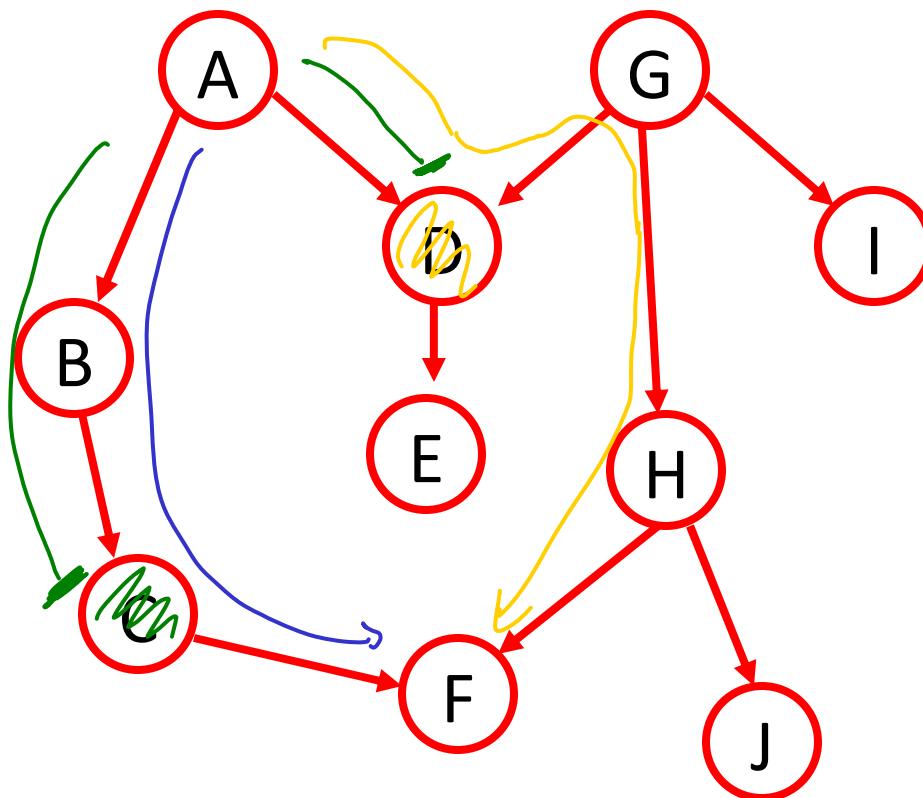
- BN specifies joint distribution through conditional parameterization that satisfies Local Markov Property
 $I_{loc}(G) = \{(X_i \perp \text{Nondescendants}_{X_i} \mid Pa_{X_i})\}$

- But we also talked about additional properties of CI
 - Weak Union, Intersection, Contraction, ...
- Which additional CI does a particular BN specify?
 - All CI that can be derived through algebraic operations

→ proving CI is very cumbersome!!

Is there an easy way to find all independences of a BN just by looking at its graph??

Examples



$A \perp F$ ✓
 $A \perp F | C$ ✓
 $A \perp F | C, D$ ✗

Active trails

- An undirected path in BN structure G is called **active trail** for observed variables $\mathbf{O} \subseteq \{X_1, \dots, X_n\}$, if for every consecutive triple of vars X, Y, Z on the path
 - $X \rightarrow Y \rightarrow Z$ and Y is unobserved ($Y \notin \mathbf{O}$)
 - $X \leftarrow Y \leftarrow Z$ and Y is unobserved ($Y \notin \mathbf{O}$)
 - $X \leftarrow Y \rightarrow Z$ and Y is unobserved ($Y \notin \mathbf{O}$)
 - $X \rightarrow Y \leftarrow Z$ and Y or any of Y 's descendants is observed
- Any variables X_i and X_j for which \nexists active trail for observations \mathbf{O} are called d-separated by \mathbf{O}
We write **d-sep($X_i; X_j | \mathbf{O}$)**
- Sets \mathbf{A} and \mathbf{B} are d-separated given \mathbf{O} if $d\text{-sep}(X, Y | \mathbf{O})$ for all $X \in \mathbf{A}, Y \in \mathbf{B}$. Write **d-sep($\mathbf{A}; \mathbf{B} | \mathbf{O}$)**

Soundness of d-separation

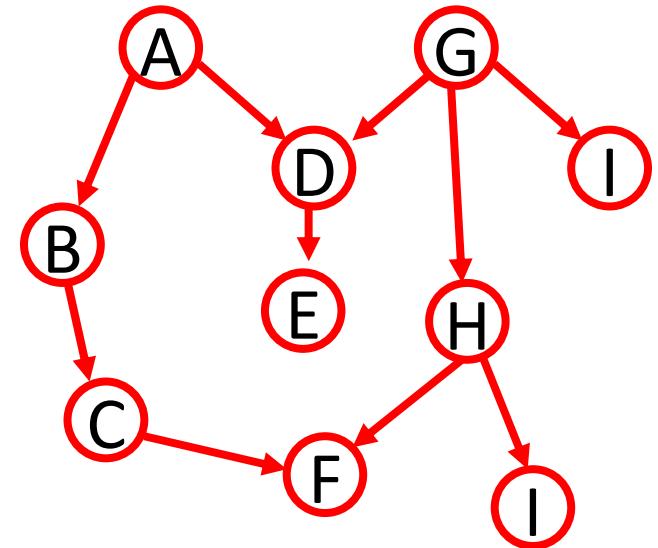
- Have seen: P factorizes according to $G \Leftrightarrow I_{loc}(G) \subseteq I(P)$
- Define $I(G) = \{(X \perp Y \mid Z) : d\text{-sep}_G(X; Y \mid Z)\}$
- **Theorem:** Soundness of d-separation
 P factorizes over $G \rightarrow I(G) \subseteq I(P)$
- Hence, d-separation captures only true independences
- How about $I(G) = I(P)$?

Completeness of d-separation

- Theorem: For “almost all” distributions P that factorize over G it holds that $I(G) = I(P)$
 - “almost all”: except for a set of distributions with measure 0, assuming only that no finite set of distributions has measure > 0

Algorithm for d-separation

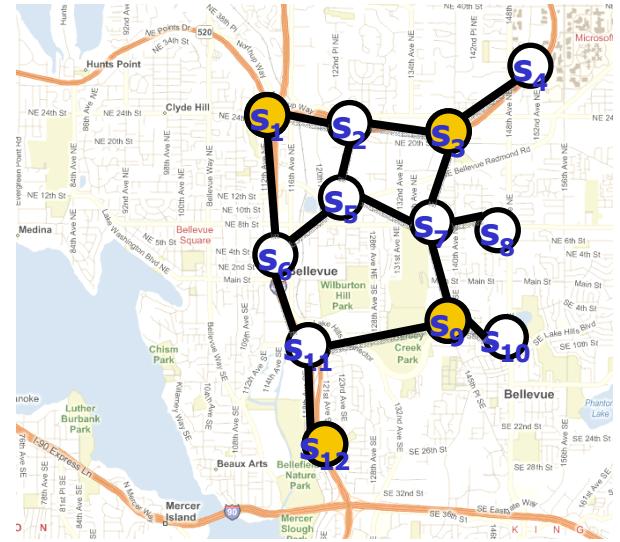
- How can we check if $X \perp\!\!\! \perp Y | Z$?
 - Idea: Check every possible path connecting X and Y and verify conditions
 - Exponentially many paths!!! 😞
- Linear time algorithm:
Find all nodes reachable from X
 - 1. Mark **Z** and its ancestors
 - 2. Do breadth-first search starting from X; stop if path is blocked
 - Have to be careful with implementation details (see reading)



Representing the world using BNs



represent



True distribution P'
with cond. ind. $I(P')$

Bayes net (G, P)
with $I(P)$

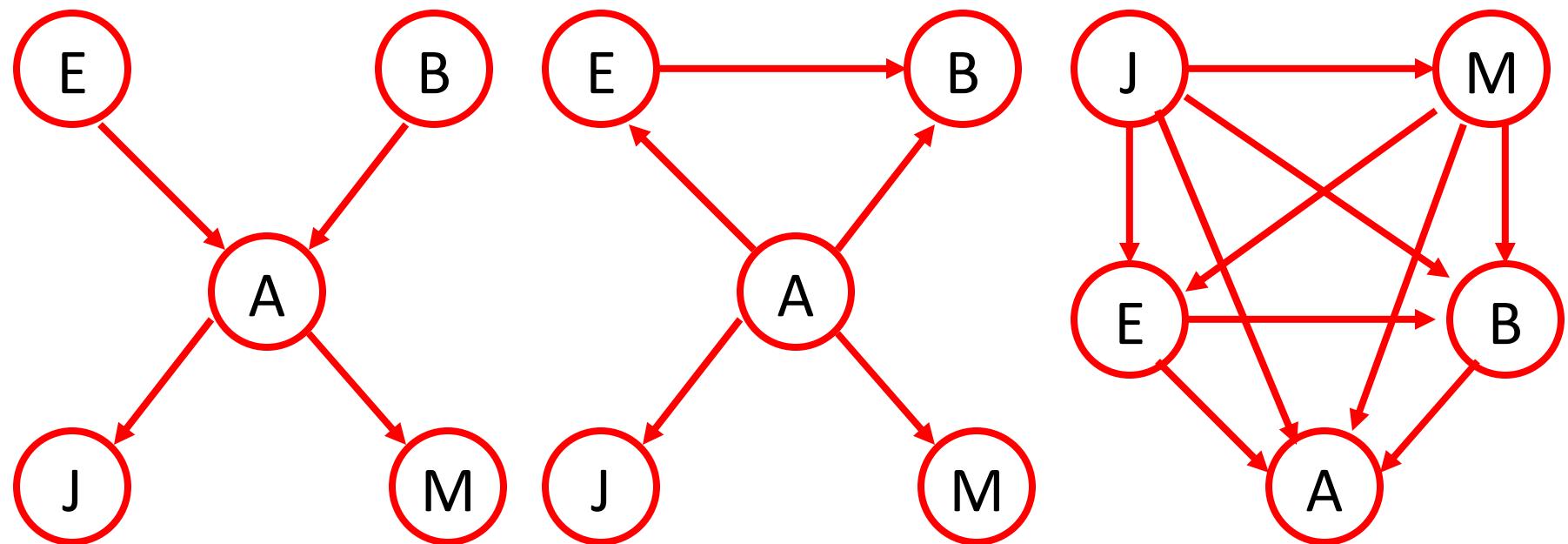
- Want to make sure that $I(P) \subseteq I(P')$
- Ideally: $I(P) = I(P')$
- Want BN that **exactly** captures independencies in P' !

Minimal I-map

- Graph G is called minimal I-map if it's an I-map, and if any edge is deleted → no longer I-map.

Uniqueness of Minimal I-maps

- Is the minimal I-Map unique?

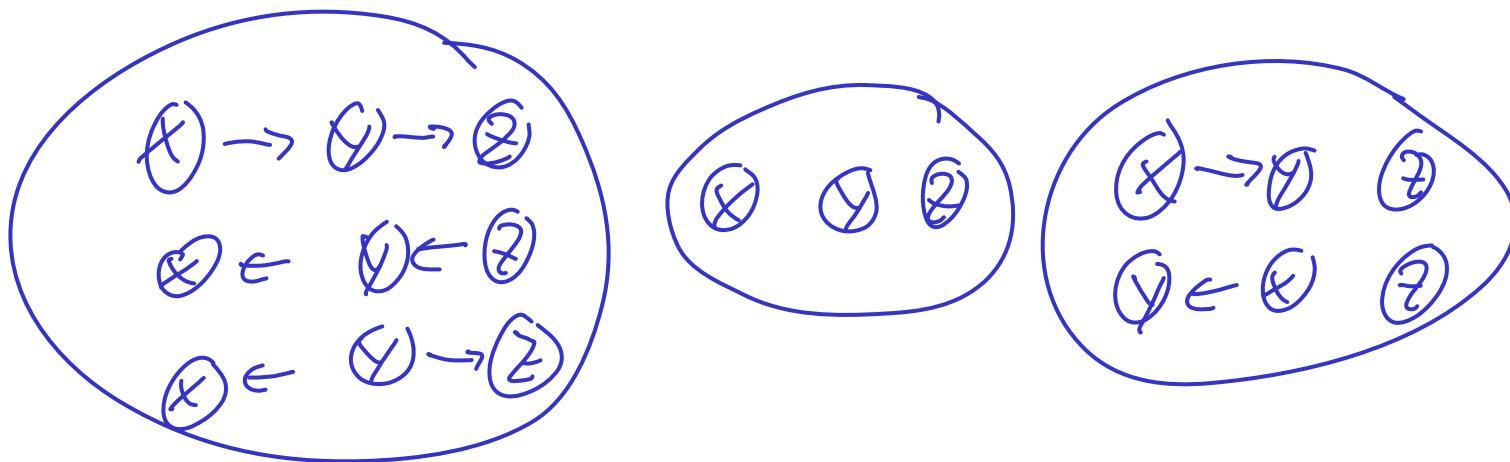


Perfect maps

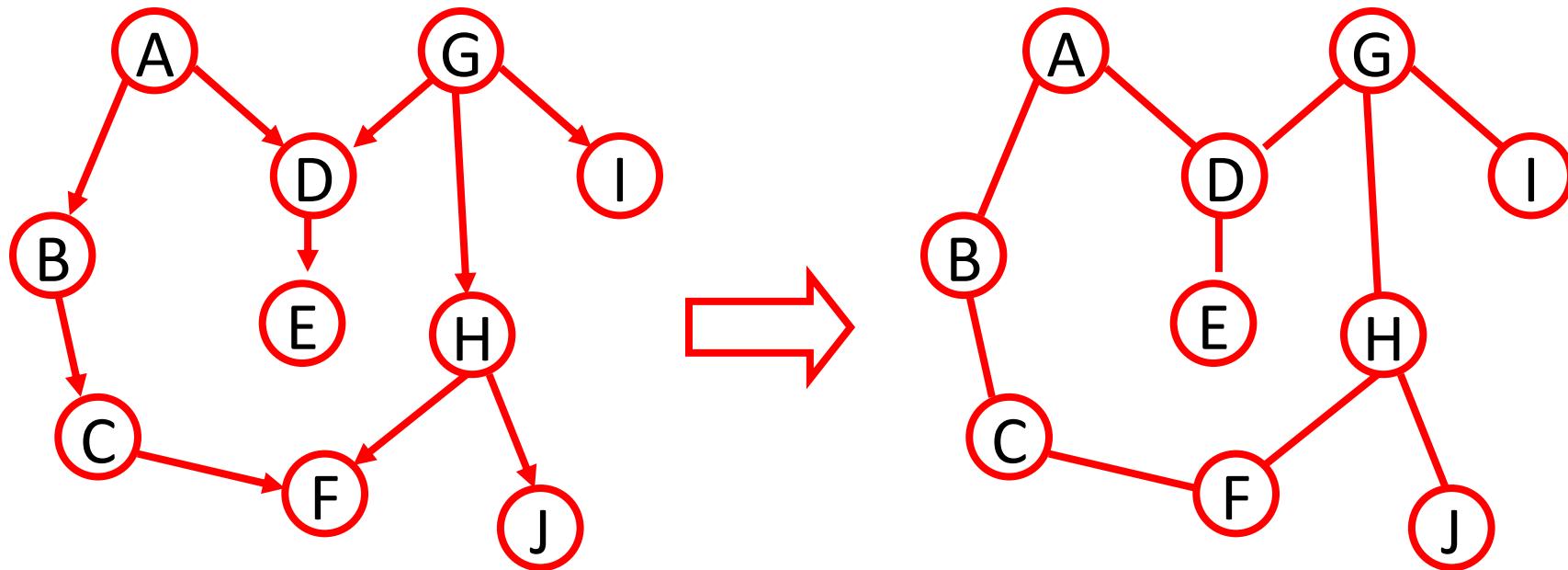
- Minimal I-maps are easy to find, but can contain many unnecessary dependencies.
- A BN structure G is called **P-map** (perfect map) for distribution P if $I(G) = I(P)$
- Does every distribution P have a P-map?

I-Equivalence

- Two graphs G, G' are called I-equivalent if $I(G) = I(G')$
- I-equivalence partitions graphs into equivalence classes



Skeletons of BNs



- I-equivalent BNs must have same skeleton

Immoralities and I-equivalence

- A V-structure $X \rightarrow Y \leftarrow Z$ is called **immoral** if there is no edge between X and Z (“unmarried parents”)



- **Theorem:** $I(G) = I(G') \Leftrightarrow G$ and G' have the same skeleton and the same immoralities.

Today: Learning BN from data

- Want P-map if one exists
- Need to find
 - Skeleton
 - Immoralities

Identifying the skeleton

- When is there an edge between X and Y?

If $\neg(X \perp Y)$? 
 $\emptyset \rightarrow \emptyset \rightarrow \emptyset$: $\neg(X \perp Y)$
but no edge

If $\neg(X \perp Y)$ (everything else)
 

- When is there no edge between X and Y?

$\Leftrightarrow \exists u \in \{x_1, \dots, x_m\} \setminus \{X, Y\} : X \perp Y | u$
(Local Markov assumption)

Algorithm for identifying the skeleton

Set G to be complete graph

For every pair $X, Y \in G$

For every $U \subset \{X_1, \dots, X_n\} \setminus \{X, Y\}$

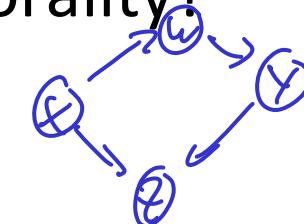
If $X \perp Y | U$ then remove edge between X, Y

Not necessarily practical ...

Identifying immoralities

- When is $X - Z - Y$ an immorality?

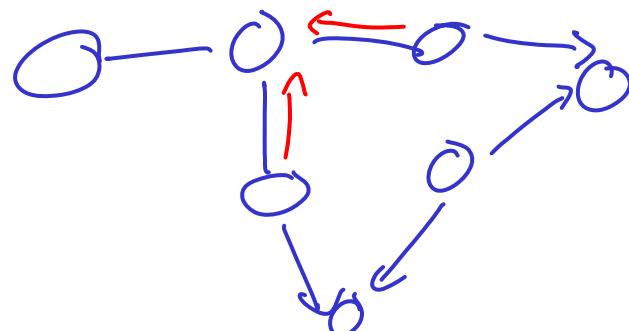
$$\neg(X \perp\!\!\!\perp Y | Z)$$



- Immoral \Leftrightarrow for all $\mathbf{U}, Z \in \mathbf{U}$: $\neg(X \perp\!\!\!\perp Y | \mathbf{U})$

From skeleton & immoralities to BN Structures

- Represent I-equivalence class as partially-directed acyclic graph (PDAG)



- How do I convert PDAG into BN?

Have to be careful when orienting edges
to avoid cycles

Poly time algo in reading

Testing independence

- So far, assumed that we know $I(P')$, i.e., all independencies associated with true dist. P'
- Often, access to P' only through sample data (e.g., sensor measurements, etc.)
- Given vars X, Y, Z , want to test whether $X \perp Y | Z$

$$X \perp Y | Z \Leftrightarrow I(X; Y | Z) = 0$$

$$\sum_z \sum_{xy} p(x,y,z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)}$$

Estimate $p(x,y,z)$ from data $\Rightarrow \hat{p}(x,y,z)$

$$\text{Compute } \hat{I}(X; Y | Z) = \sum_{x,y,z} \hat{p}(x,y,z) \log \frac{\hat{p}(x,y|z)}{\hat{p}(x|z)\hat{p}(y|z)}$$

Test whether $\hat{I}(X; Y | Z) < \epsilon$

Next topic: Learning BN from Data

- Two main parts:
 - Learning structure (conditional independencies)
 - Learning parameters (CPDs)

Parameter learning

- Suppose X is Bernoulli distribution (coin flip) with unknown parameter $P(X=H) = \theta$.
- Given training data $D = \{x^{(1)}, \dots, x^{(m)}\}$ (e.g., H H T H H H T T H T H H..) how do we estimate θ ?

$$P(D|\theta) = \theta^{n_H} \cdot (1-\theta)^{n_T}$$

n_H : # of H in D

n_T : — — T —

$$\begin{aligned}\hat{\theta} &= \underset{\theta}{\operatorname{argmax}} P(D|\theta) = \underset{\theta}{\operatorname{argmax}} \log P(D|\theta) \\ &= \underset{\theta}{\operatorname{argmax}} n_H \log \theta + n_T \log(1-\theta)\end{aligned}$$

want $\frac{d \log P(D|\theta)}{d\theta} = 0$.

Maximum Likelihood Estimation

- **Given:** data set D
- **Hypothesis:** data generated i.i.d. from binomial distribution with $P(X = H) = \theta$
- Optimize for θ which makes D most likely:

Solving the optimization problem

$$\log P(D|\theta) = n_H \log \theta + n_T \log (1-\theta)$$

$$\frac{d}{d\theta} \log P(D|\theta) = \frac{n_H}{\theta} - \frac{n_T}{1-\theta} \stackrel{!}{=} 0$$

$$n_H(1-\theta) = n_T\theta$$

$$n_H = \theta(n_T + n_H)$$

$$\Rightarrow \hat{\theta} = \frac{n_H}{n_T + n_H} = \frac{n_H}{m}$$

$$= \frac{\text{Count}(X=H)}{m}$$

Learning general BNs

| | Known structure | Unknown structure |
|------------------|-----------------|-------------------|
| Fully observable | <u>Easy</u> | hard 2. |
| Missing data | hard 3. (EM) | very hard (ast) |

Estimating CPDs

- Given data $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ of samples from X, Y , want to estimate $P(X | Y)$

$$P(X=x | Y=y) = \theta_{x|y}$$

$$\hat{\theta}_{x|y} \stackrel{MLE}{=} \frac{\text{Count}(x,y)}{\text{Count}(y)}$$

$$\hat{P}(x,y) = \frac{\text{Count}(x,y)}{m}$$

$$\hat{p}(y) = \frac{\text{Count}(y)}{m}$$

$$\hat{\theta}_{x|y} = \frac{\hat{P}(x,y)}{\hat{p}(y)}$$

MLE for Bayes nets

$$\begin{aligned}\log P(D | \theta) &= \log \prod_l \prod_i P(X_i^{(l)} | \text{Pa}_i^{(l)}; \theta) \\ &= \sum_l \sum_i \log P(X_i^{(l)} | \text{Pa}_i^{(l)}; \theta_{X_i(\text{Pa}_i)})\end{aligned}$$

Parameter
indepent

$$\begin{aligned}\frac{\partial}{\partial \theta_{X_i(\text{Pa}_i)}} \log P(D | \theta) &= \sum_j \sum_l \frac{\partial}{\partial \theta_{X_i(\text{Pa}_i)}} \log P(X_j^{(l)} | \text{Pa}_j^{(l)}; \theta_{X_j(\text{Pa}_j)}) \\ &= \sum_l \frac{\partial}{\partial \theta_{X_i(\text{Pa}_i)}} \log P(X_i^{(l)} | \text{Pa}_i; \theta_{X_i(\text{Pa}_i)}) \stackrel{!}{=} 0\end{aligned}$$

Problem breaks down into independent subproblems

Learn every CPD independent of others

Algorithm for BN MLE

Given BN structure G

For each variable X_i :

$$\text{learn } \hat{\theta}_{X_i | \text{pa}_i} = \frac{\text{Count}(X_i, \text{pa}_i)}{\text{Count}(\text{pa}_i)}$$

\Rightarrow globally maximum likelihood estimate
for fixed structure G

Learning general BNs

| | Known structure | Unknown structure |
|------------------|--|-------------------|
| Fully observable | Easy! 😊 <i>Get CPDs by counting (for MLE)</i> | ??? |
| Missing data | Hard (EM) | Very hard (later) |

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

MLE for structure learning

- For fixed structure, can compute likelihood of data

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = \sum_{\ell} \sum_i \log P(X_i = x_i^{(\ell)} \mid \text{Pa}_i = \text{pa}_i^{(\ell)}, \theta_{\mathcal{G}}; \mathcal{G})$$

$$\stackrel{\text{MLE}}{=} \sum_i \sum_{x_i} \sum_{\text{pa}_i} \text{Count}(x_i, \text{pa}_i) \log \frac{\hat{P}(x_i, \text{pa}_i)}{\hat{P}(\text{pa}_i)}$$

$$= m \sum_i \sum_{x_i} \sum_{\text{pa}_i} \hat{P}(x_i, \text{pa}_i) \log \frac{\hat{P}(x_i, \text{pa}_i) \hat{P}(x_i)}{\hat{P}(\text{pa}_i) \hat{P}(x_i)}$$

$$= m \sum_i \sum_{x_i} \sum_{\text{pa}_i} \hat{P}(x_i, \text{pa}_i) \log \frac{\hat{P}(x_i, \text{pa}_i)}{\hat{P}(x_i) \hat{P}(\text{pa}_i)} + m \sum_i \sum_{x_i} \sum_{\text{pa}_i} \hat{P}(y_i, \text{pa}_i) \log \hat{P}(y_i)$$

$$= m \sum_i \hat{I}(X_i; \text{Pa}_i) - m \sum_i \hat{H}(X_i)$$

Decomposable score

- Log-data likelihood

$$\log \hat{P}(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = m \sum_i \hat{I}(X_i, \mathbf{Pa}_i) - m \sum_i \hat{H}(X_i)$$

independent of graph structure!

- MLE score decomposes over families of the BN (nodes + parents)
- Score($G ; D$) = $\sum_i \text{FamScore}(X_i \mid \text{Pa}_i; D)$
- Can exploit for computational efficiency!

Finding the optimal MLE structure

- Log-likelihood score:

$$\text{Score}(\mathcal{G}; \mathcal{D}) = \sum_i \widehat{I}(X_i, \mathbf{Pa}_i)$$

- Want $\mathbf{G}^* = \operatorname{argmax}_{\mathbf{G}} \text{Score}(\mathbf{G}; \mathcal{D})$
- Lemma: $\mathbf{G} \subseteq \mathbf{G}' \rightarrow \text{Score}(\mathbf{G}; \mathcal{D}) \leq \text{Score}(\mathbf{G}'; \mathcal{D})$

Complete graph
maximizes log data likelihood !

"Information never hurts"
RV X , $A \subset B$
 $H(X|A) \geq H(X|B)$
 $I(X; A) = H(X) - H(X|A)$
 $\Rightarrow I(X; B) \geq I(X; A)$

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)

Constraint optimization of BN structures

- **Theorem:** for any fixed $d \geq 2$, finding the optimal BN (w.r.t. MLE score) is NP-hard
- What about $d=1$??
- Want to find optimal tree!

Finding the optimal tree BN

- Scoring function

$$\text{Score}(\mathcal{G}; \mathcal{D}) = \sum_i \hat{I}(X_i, \mathbf{Pa}_i)$$

- Scoring a tree

$$\begin{array}{c} \emptyset \rightarrow \emptyset \rightarrow \emptyset \\ \hat{I}(z; y) + \hat{I}(y; z) \end{array} \stackrel{\text{Same Score}}{=} \begin{array}{c} \emptyset \\ \downarrow \quad \downarrow \\ \emptyset \quad \emptyset \end{array} \quad \hat{I}(x; y) + \hat{I}(z; y)$$

b/c: $I(X, Y) = I(Y, X)$

Same skeleton \Rightarrow same score

Finding the optimal tree skeleton

- Can reduce to following problem:
- Given graph $G = (V, E)$, and nonnegative weights w_e for each edge $e=(X_i, X_j)$
 - In our case: $w_e = I(X_i, X_j)$
- Want to find tree $T \subseteq E$ that maximizes $\sum_{e \in T} w_e$
- Maximum spanning tree problem!
- Can solve in time $O(|E| \log |E|)$!

Chow-Liu algorithm

- For each pair X_i, X_j of variables compute

$$\hat{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{m}$$

- Compute mutual information

$$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

- Define complete graph with weight of edge (X_i, X_j) given by the mutual information
- Find maximum spanning tree → skeleton
- Orient the skeleton using breadth-first search

Generalizing Chow-Liu

- Tree-augmented Naïve Bayes Model [Friedman '97]
- If evidence variables are correlated, Naïve Bayes models can be overconfident
- **Key idea:** Learn optimal tree for conditional distribution $P(X_1, \dots, X_n \mid Y)$
- Can do optimally using Chow-Liu (homework! ☺)

Tasks

- Subscribe to Mailing list
<https://utils.its.caltech.edu/mailman/listinfo/cs155>
- Select recitation times
- Read Koller & Friedman Chapter 17.1-17.3, 18.1-2, 18.4.1
- Form groups and think about class projects. If you have difficulty finding a group, email Pete Trautman

Probabilistic Graphical Models

Lecture 5 – Bayesian Learning of
Bayesian Networks

CS/CNS/EE 155
Andreas Krause

Announcements

- Recitations: Every Tuesday 4-5:30 in 243 Annenberg
- Homework 1 out. Due in class Wed Oct 21
- Project proposals due Monday Oct 19

Project proposal

- At most 2 pages. One proposal per project
- due Monday Oct 19
- Please clearly specify
 - What is the idea of this project?
 - Who will be on the team?
 - What data will you use? Will you need time "cleaning up" the data?
 - What code will you need to write? What existing code are you planning to use?
 - What references are relevant? Mention 1-3 related papers.
 - What are you planning to accomplish by the Nov 9 milestone?

Project ideas

- Ideally, do graphical model project related to your research (and, e.g., data that you're working with)
 - Must be a new project started for the class!
- Website has examples for
 - Project ideas
 - Data sets
 - Code

Project ideas

- All projects should involve using PGMs for some data set, and then doing some experiments
- Learning related
 - Experiment with different algorithms for structure / parameter learning
- Inference related
 - Compare different algorithms for exact or approximate inference
- Algorithmic / decision making
 - Experiment with algorithms for value of information, MAP assignment, ...
- Application related
 - Attempt to answer interesting domain-related question using graphical modeling techniques

Data sets

- Some cool data sets made available specifically for this course!!
→ Contact TAs to get access to data.
- Exercise physiological data (collected by John Doyle's group)
 - E.g., do model identification / Bayesian filtering
- Fly data (by Pietro Perona and Michael Dickinson et al.)
 - “Activity recognition” – what are the patterns in fly behavior? Clustering / segmentation of trajectories?
- Urban challenge data (GPS data + LADAR + Vision) by Richard Murray et al.
 - Sensor fusion using DBNs; SLAM
- JPL MER data by Larry Matthies et al.
 - Predict slip based on orbital imagery + GPS tracks
 - Segment images to identify dangerous areas for rover
- LDPC decoding
 - Compare new approximate inference techniques with Loopy-BP
- Other open data sets mentioned on course webpage

Code

- Libraries for graphical modeling by Intel, Microsoft, ...
- Toolboxes
 - computer vision image manipulations
 - Topic modeling
 - Nonparametric Bayesian modeling (Dirichlet processes / Gaussian processes / ...)

Learning general BNs

| | Known structure | Unknown structure |
|------------------|-----------------|-------------------|
| Fully observable | <u>Easy</u> ;) | hard 2. |
| Missing data | hard 3. (EM) | very hard (ast) |

Algorithm for BN MLE

Given BN structure G

For each variable X_i :

$$\text{learn } \hat{\theta}_{X_i | \text{pa}_i} = \frac{\text{Count}(X_i, \text{pa}_i)}{\text{Count}(\text{pa}_i)}$$

\Rightarrow globally maximum likelihood estimate
for fixed structure G

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

MLE for structure learning

- For fixed structure, can compute likelihood of data

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = \sum_{\ell} \sum_i \log P(X_i = x_i^{(\ell)} \mid \text{Pa}_i = \text{pa}_i^{(\ell)}, \theta_{\mathcal{G}}; \mathcal{G})$$

$$\stackrel{\text{MLE}}{=} \sum_i \sum_{x_i} \sum_{\text{pa}_i} \text{Count}(x_i, \text{pa}_i) \log \frac{\hat{P}(x_i, \text{pa}_i)}{\hat{P}(\text{pa}_i)}$$

$$= m \sum_i \sum_{x_i} \sum_{\text{pa}_i} \hat{P}(x_i, \text{pa}_i) \log \frac{\hat{P}(x_i, \text{pa}_i) \hat{P}(x_i)}{\hat{P}(\text{pa}_i) \hat{P}(x_i)}$$

$$= m \sum_i \sum_{x_i} \sum_{\text{pa}_i} \hat{P}(x_i, \text{pa}_i) \log \frac{\hat{P}(x_i, \text{pa}_i)}{\hat{P}(x_i) \hat{P}(\text{pa}_i)} + m \sum_i \sum_{x_i} \sum_{\text{pa}_i} \hat{P}(y_i, \text{pa}_i) \log \hat{P}(y_i)$$

$$= m \sum_i \hat{I}(X_i; \text{Pa}_i) - m \sum_i \hat{H}(X_i)$$

Decomposable score

- Log-data likelihood

$$\log \hat{P}(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G}) = m \sum_i \hat{I}(X_i, \mathbf{Pa}_i) - m \sum_i \hat{H}(X_i)$$

independent of graph structure!

- MLE score decomposes over families of the BN (nodes + parents)
- Score($G ; D$) = $\sum_i \text{FamScore}(X_i \mid \text{Pa}_i; D)$
- Can exploit for computational efficiency!

Finding the optimal MLE structure

- Log-likelihood score:

$$\text{Score}(\mathcal{G}; \mathcal{D}) = \sum_i \widehat{I}(X_i, \mathbf{Pa}_i)$$

- Want $\mathbf{G}^* = \operatorname{argmax}_{\mathbf{G}} \text{Score}(\mathbf{G}; \mathcal{D})$
- Lemma: $\mathbf{G} \subseteq \mathbf{G}' \rightarrow \text{Score}(\mathbf{G}; \mathcal{D}) \leq \text{Score}(\mathbf{G}'; \mathcal{D})$

Complete graph
maximizes log data likelihood !

"Information never hurts"
RV X , $A \subset B$
 $H(X|A) \geq H(X|B)$
 $I(X; A) = H(X) - H(X|A)$
 $\Rightarrow I(X; B) \geq I(X; A)$

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)

Chow-Liu algorithm

- For each pair X_i, X_j of variables compute

$$\hat{P}(x_i, x_j) = \frac{\text{Count}(x_i, x_j)}{m}$$

- Compute mutual information

$$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

- Define complete graph with weight of edge (X_i, X_j) given by the mutual information
- Find maximum spanning tree → skeleton
- Orient the skeleton using breadth-first search

Today: Bayesian learning

- X Bernoulli variable
- Which is better:
 - Observe 1 H and 2 T $\hat{\theta} = \frac{1}{3}$
 - Observe 10 H and 20 T $\hat{\theta} = \frac{1}{3}$
 - Observe 100 H and 200 T $\hat{\theta} = \frac{1}{3}$
- MLE is same in all three cases
- However, should be much more “confident” about MLE if we have more data
 - Want to model distributions over parameters

Bayesian learning

- Make prior assumptions about parameters $P(\theta)$
- Compute posterior

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

Given data D want to predict

$$P(x|D) = \int P(\theta|D) \underbrace{P(x|\theta)}_{d\theta}$$

In MLE

$$P(x|D) \approx P(x|\hat{\theta}) \quad \hat{\theta} = \operatorname{argmax}_{\theta} P(D|\theta)$$

Bayesian Learning for Binomial

$$P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$$

- Likelihood function:

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

- How do we choose prior?
 - Many possible answers...
 - Pragmatic approach: Want computationally “simple” (and still flexible) prior

Conjugate priors

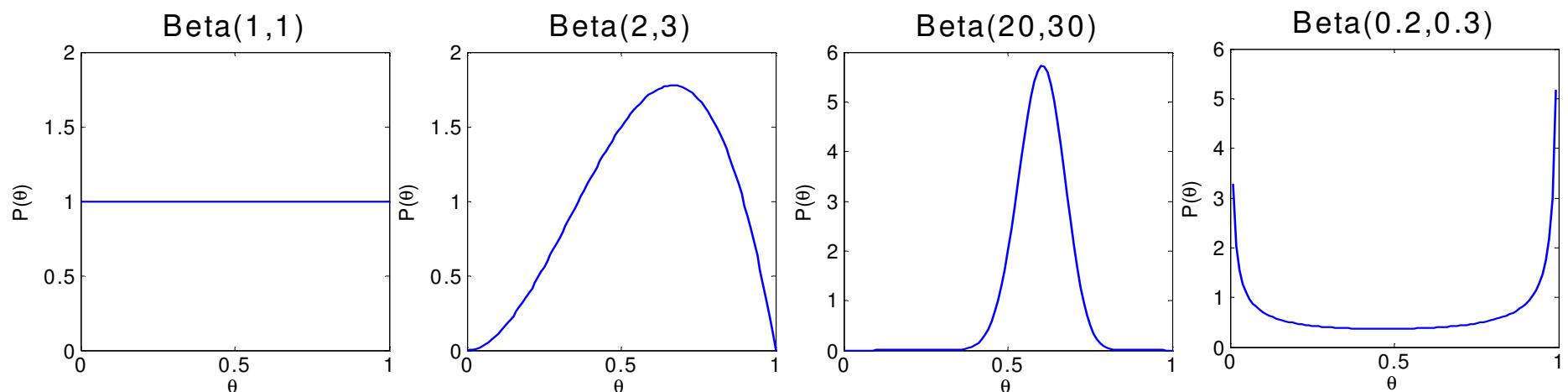
- Consider parametric families of prior distributions:
 - $P(\theta) = f(\theta; \alpha)$
 - α is called “hyperparameters” of prior
- A prior $P(\theta) = f(\theta; \alpha)$ is called **conjugate** for a likelihood function $P(D | \theta)$ if $P(\theta | D) = f(\theta; \alpha')$
 - Posterior has same parametric form
 - Hyperparameters are updated based on data D
- Obvious questions (answered later):
 - How to choose hyperparameters??
 - Why limit ourselves to conjugate priors??

Conjugate prior for Binomial

- Beta distribution

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

Normalization constant



Posterior for Beta prior

- Beta distribution

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

- Likelihood:

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

- Posterior:

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

$$P(\theta \mid \mathcal{D}) = \text{Beta}(\theta; \alpha_H + m_H, \alpha_T + m_T)$$

Bayesian prediction

- Prior $P(\theta) = \text{Beta}(\alpha_H, \alpha_T)$ *Remark:* $P(X=H) = \theta$
- Suppose we observe $D = \{m_H \text{ heads, and } m_T \text{ tails}\}$
- What's $P(X=H | D)$, i.e., prob. that next flip is heads?

$$P(X=H | D) = \underbrace{\int \theta P(\theta | D) d\theta}_{= E[\theta | D]} = \frac{\alpha_H + m_H}{\alpha_H + \alpha_T + m_H + m_T}$$

Prior = Smoothing

$$\mathbb{E}[\theta] = \frac{m_H + \alpha_H}{\underbrace{m_H + m_T}_{m} + \underbrace{\alpha_H + \alpha_T}_{m'}} = \frac{m_H + \gamma m'}{\underbrace{m + m'}_{(\approx)}}$$

- Where $m' = \alpha_H + \alpha_T$, and $\underline{\gamma = \alpha_H / m'}$, $0 \leq \gamma \leq 1$
- m' is called “equivalent sample size”
→ “hallucinated” coin flips

$$\mathbb{E}[\theta] = \frac{m}{m+m'} \underbrace{\frac{m_H}{m}}_{\text{MLE}} + \frac{m'}{m+m'} \underbrace{\gamma}_{\text{prior mean}}$$

$m \rightarrow \infty$ $\mathbb{E}[\theta] \rightarrow \text{MLE}$ Forget prior
 $m = 0$ prior

→ Interpolate between MLE and prior mean

Conjugate for multinomial

- If $X \in \{1, \dots, k\}$ has k states:
- Multinomial likelihood

$$P(\mathcal{D} \mid \theta) = \theta_1^{m_1} \theta_2^{m_2} \dots \theta_k^{m_k}$$

where $\sum_i \theta_i = 1, \theta_i \geq 0$

- Conjugate prior: Dirichlet distribution

$$P(\theta) = \text{Dir}(\theta; \alpha_1, \dots, \alpha_k) = \frac{1}{Z} \prod_i \theta_i^{\alpha_i - 1}$$

- If observe $D = \{m_1 \text{ 1s, } m_2 \text{ 2s, } \dots \text{ } m_k \text{ ks}\}$, then

$$P(\theta \mid \mathcal{D}) = \text{Dir}(\theta; \alpha_1 + m_1, \dots, \alpha_k + m_k)$$

Parameter learning for CPDs

- Parameters $P(X | Pa_X)$
- Have one parameter $\theta_{X | pa_X}$ for each value of parents pa_X

$$P(\theta_{X | Pa_X = u}) = \text{Dir}(\alpha_1, \dots, \alpha_k)$$

$$P(\theta_{X | Pa_X = u_1, \dots, u_N}) = \prod_u P(\theta_{X | Pa_X = u})$$

"local parameter independence"

Parameter learning for BNs

- Each CPD $P(X \mid \text{Pa}_X; \theta_{X|\text{Pa}_X})$ has its own sets of parameters $P(\theta_{X|\text{pa}_X})$
→ Dirichlet distribution
- Want to compute posterior over all parameters

$$P(\theta_{X_1|\text{Pa}_{X_1}}, \dots, \theta_{X_n|\text{Pa}_{X_n}} \mid \mathcal{D})$$

- How can we do this??
- **Crucial assumption:** Prior distribution over parameters factorizes (“parameter independence”)
global

$$P(\theta_{X_1|\text{Pa}_{X_1}}, \dots, \theta_{X_n|\text{Pa}_{X_n}}) = \prod_i P(\theta_{X_i|\text{Pa}_{X_i}})$$

Parameter Independence

- Assume

$$P(\theta_{X_1} | \mathbf{Pa}_{X_1}, \dots, \theta_{X_n} | \mathbf{Pa}_{X_n}) = \prod_i P(\theta_{X_i} | \mathbf{Pa}_{X_i})$$

- Why useful?
- If data is fully observed, then

$$P(\theta_{X_1} | \mathbf{Pa}_{X_1}, \dots, \theta_{X_n} | \mathbf{Pa}_{X_n} \mid \mathcal{D}) = \prod_i P(\theta_{X_i} | \mathbf{Pa}_{X_i} \mid \mathcal{D})$$

- I.e., posterior still independent. Why??

Meta-BN with parameters

Meta-BN

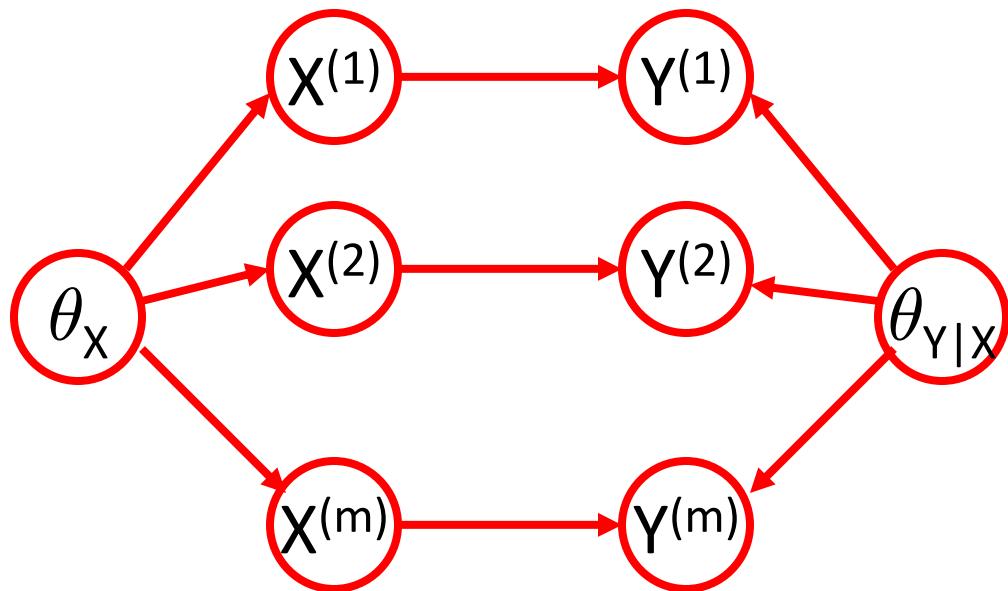
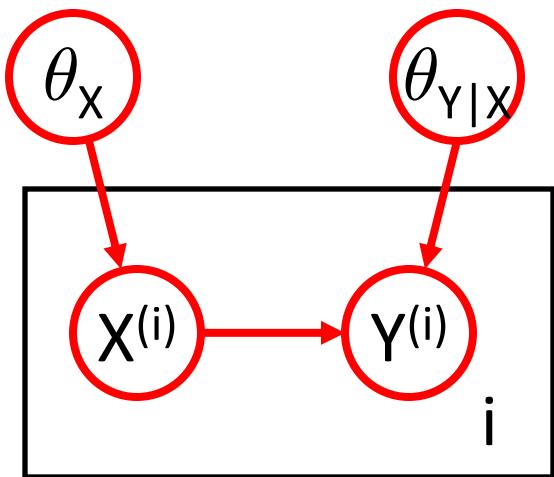


Plate notation



Meta BN contains one copy of original BN per data sample, and one variable for each parameter

Under parameter-independences, data d-separates parameters
Also: Parameters d-separate copies of BN: $P(D, X|\theta) = P(D|s) P(X|s)$

Bayesian learning of Bayesian Networks

- Specifying priors helps overfitting
 - Do not commit to fixed parameter estimate, but maintain distribution
- So far: Know how to specify priors over parameters for fixed structure.
- Why should we commit to fixed structure??
- Fully Bayesian inference

$$P(\underline{\mathbf{X}} \mid \mathcal{D}) \propto \sum_{\mathcal{G}} P(\mathcal{G}) \int P(\theta_{\mathcal{G}} \mid \mathcal{G}) P(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) P(\mathbf{X} \mid \mathcal{D}, \mathcal{G}, \theta_{\mathcal{G}}) d\theta$$

Prior over param. *Likelihood of data* *Likelihood of pred. vars*

$= P(X \mid \mathcal{G}, \theta_{\mathcal{G}})$

Prior over Structure

Fully Bayesian inference

$$P(\mathbf{X} \mid \mathcal{D}) \propto \sum_{\mathcal{G}} P(\mathcal{G}) \int P(\theta_{\mathcal{G}} \mid \mathcal{G}) P(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) P(\mathbf{X} \mid \mathcal{G}, \theta_{\mathcal{G}}) d\theta$$

- $P(\mathcal{G})$: Prior over graphs
 - E.g.: $P(\mathcal{G}) = \exp(-c \text{ Dim}(\mathcal{G}))$
- $\text{Dim}(\mathcal{G}) = \# \text{ free params}$
- Called “Bayesian Model Averaging”
- **Hopelessly intractable for larger models**
- Often: want to pick most likely structure:

$$\mathcal{G}^* = \operatorname{argmax}_{\mathcal{G}} P(\mathcal{G} \mid \mathcal{D}) = \operatorname{argmax}_{\mathcal{G}} \log P(\mathcal{G}) + \log P(\mathcal{D} \mid \mathcal{G})$$

Why do priors help 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:

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

- Why??
- **Theorem:** For Dirichlet priors, and for $m \rightarrow \infty$:

$$\log P(\mathcal{D} \mid \mathcal{G}) \rightarrow \log P(\mathcal{D} \mid \mathcal{G}, \widehat{\theta}_{\mathcal{G}}) - \frac{\log m}{2} \text{Dim}(\mathcal{G}) + \mathcal{O}(1)$$

BIC score

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

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

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

- 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 $\text{Score}(G ; D)$ 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
 - $P(\theta_X | \text{Pa}_X) = \text{Dir}(\alpha, \dots, \alpha)$
- Is this a good choice?

(X)

(Y)

$$P(\theta_Y) \approx \text{Dir}(\alpha, \alpha)$$

\Rightarrow Equir. sample size

$$2\alpha$$

(X) \rightarrow (Y)

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

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

\Rightarrow Equir. sample size
 4α

BDe prior

- Want to ensure “equivalent sample size” m' is constant
- Idea:
 - Define $P'(X_1, \dots, X_n)$
For example: $P'(X_1, \dots, X_n) = \prod_i \text{Uniform}(\text{Val}(X_i))$
 - Choose equivalent sample size m'
 - Set $\alpha_{x_i | \text{pa}_i} = m' P'(x_i, \text{pa}_i)$



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

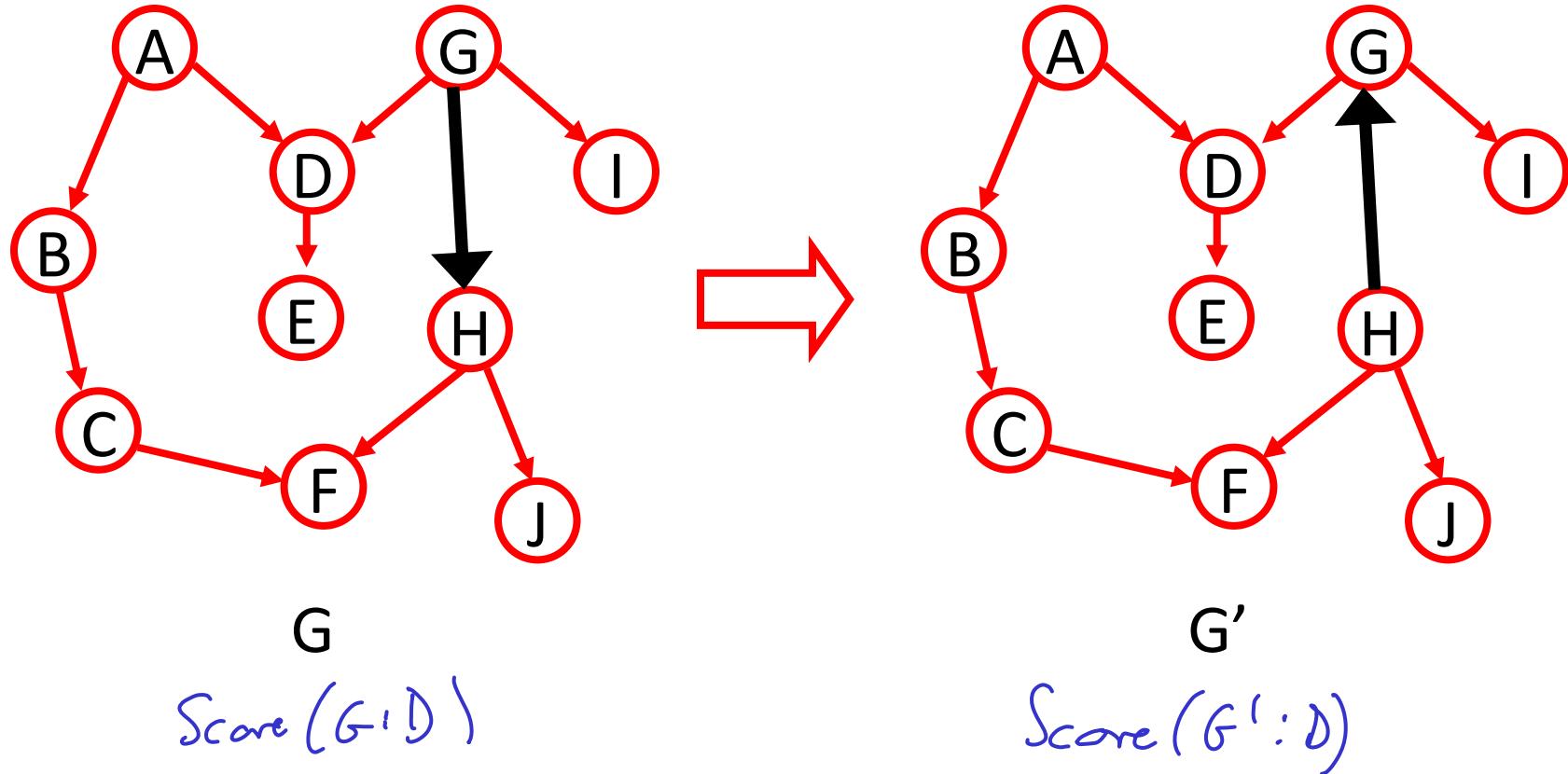
Bayesian structure search

- Given consistent scoring function $\text{Score}(G : D)$, 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



- Want to avoid recomputing the score after each modification!

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:
$$\text{Score}(G ; D) = \sum_i \text{FamScore}(X_i | Pa_i; D)$$
- If G' results from G by modifying a single edge, only need to recompute the score of the affected families!!

What you need to know

- Conjugate priors
 - Beta / Dirichlet
 - Predictions, updating of hyperparameters
- Meta-BN encoding parameters as variables
- Choice of hyperparameters
 - BDe prior
- Decomposability of scores and implications
- Local search

Tasks

- Read Koller & Friedman Chapter 17.4, 18.3-5
- Project proposal due Monday Oct 19 (contact TAs or instructor to discuss ideas)
- Homework 1 due Wednesday Oct 21

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 $P(\theta)$
- Compute posterior

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

Given data D want to predict

$$P(x|D) = \int P(\theta|D) \underbrace{P(x|\theta)}_{d\theta}$$

In MLE

$$P(x|D) \approx P(x|\hat{\theta}) \quad \hat{\theta} = \operatorname{argmax}_{\theta} P(D|\theta)$$

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 $P(D | \theta)$ if $P(\theta | D) = f(\theta; \alpha')$
 - 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

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

- Likelihood:

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

- Posterior:

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

$$P(\theta \mid \mathcal{D}) = \text{Beta}(\theta; \alpha_H + m_H, \alpha_T + m_T)$$

Why do priors help avoid overfitting?

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \underline{\theta}_{\mathcal{G}}) dP(\underline{\theta}_{\mathcal{G}} \mid \mathcal{G})$$

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

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

- Why??
- **Theorem:** For Dirichlet priors, and for $m \rightarrow \infty$:

$$\log \underline{P(\mathcal{D} \mid \mathcal{G})} \rightarrow \log P(\mathcal{D} \mid \mathcal{G}, \widehat{\theta}_{\mathcal{G}}) - \frac{\log m}{2} \text{Dim}(\mathcal{G}) + \mathcal{O}(1)$$

BIC score

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

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

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

- 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 $\text{Score}(G ; D)$ 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
 - $P(\theta_X | \text{Pa}_X) = \text{Dir}(\alpha, \dots, \alpha)$
- Is this a good choice?

(X)

(Y)

$$P(\theta_Y) \approx \text{Dir}(\alpha, \alpha)$$

\Rightarrow Equir. sample size

$$2\alpha$$

(X) \rightarrow (Y)

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

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

\Rightarrow Equir. sample size
 4α

BDe prior

- Want to ensure “equivalent sample size” m' is constant
- Idea:
 - Define $P'(X_1, \dots, X_n)$
For example: $P'(X_1, \dots, X_n) = \prod_i \text{Uniform}(\text{Val}(X_i))$
 - Choose equivalent sample size m'
 - Set $\alpha_{x_i | \text{pa}_i} = m' P'(x_i, \text{pa}_i)$



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

Score consistency

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

$$\begin{array}{ccc} \text{Diagram } G: & \text{Diagram } G^I: & \text{Score consistency: } \\ \text{Score}(G) = \text{Score}(G^I) & \Rightarrow & \text{Score}(G : D) = \text{Score}(G^I : D) \end{array}$$

Diagram G: A directed acyclic graph with three nodes X, Y, Z. Node X is at the top left, Y is at the bottom center, and Z is at the bottom right. Directed edges are X → Y and Y → Z.

Diagram G^I : A directed acyclic graph with three nodes X, Y, Z. Node X is at the top left, Y is at the top right, and Z is at the bottom right. Directed edges are X → Y and Y ← Z.

- K2 prior is inconsistent!
- BDe prior is consistent
- In fact, Bayesian score is consistent \Leftrightarrow 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 $\text{Score}(D : G)$ **decomposes** over the graph:
$$\text{Score}(G ; D) = \sum_i \text{FamScore}(X_i | Pa_i; D)$$
- If G' results from G by modifying a single edge, only need to recompute the score of the affected families!!

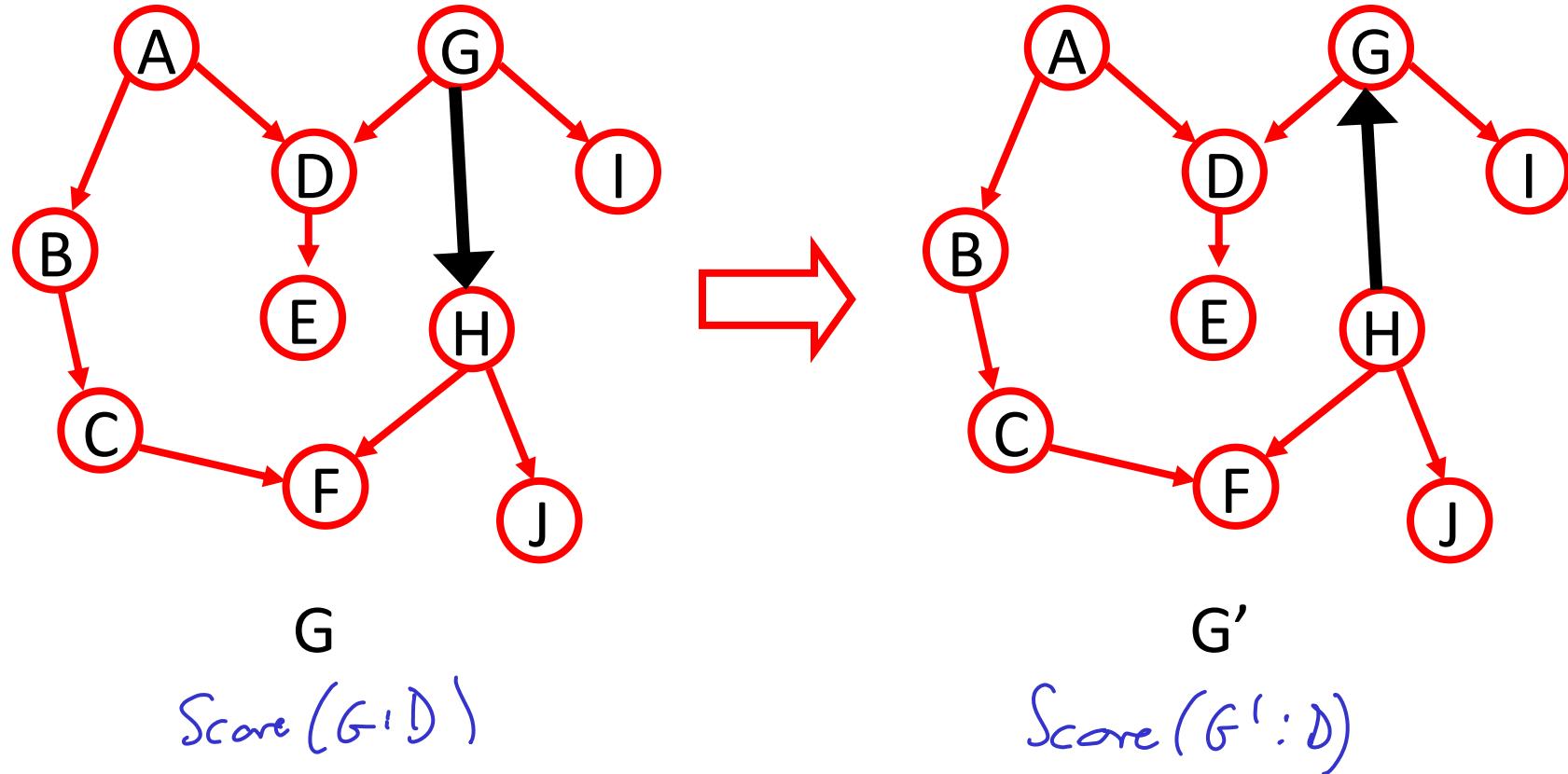
Bayesian structure search

- Given consistent scoring function $\text{Score}(G : D)$, 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



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

Alternative: Fixed order search

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

For $i = 1 : n$

For each $A \subseteq \{X_1, \dots, X_{i-1}\}$

Compute Fam Score ($X_i | A$)

$A^* = \underset{A}{\operatorname{argmax}} \text{ Fam Score} (X_i | A)$

$\text{Pa}_i = A^*$

$\text{Score}(G: D) = \sum_i \text{Fam Score} (X_i | \text{Pa}_i)$

n independent opt problems

\Rightarrow Find optimal structure!

Fixed order for d parents

- Fix ordering
- For each variable X_i
 - For each subset $A \subseteq \{X_1, \dots, X_{i-1}\}$, $|A| \leq d$ compute $\text{FamScore}(X_i | A)$
 - Set $Pa_i = \text{argmax}_A \text{FamScore}(X_i | A)$
- If score is decomposable \rightarrow 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 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

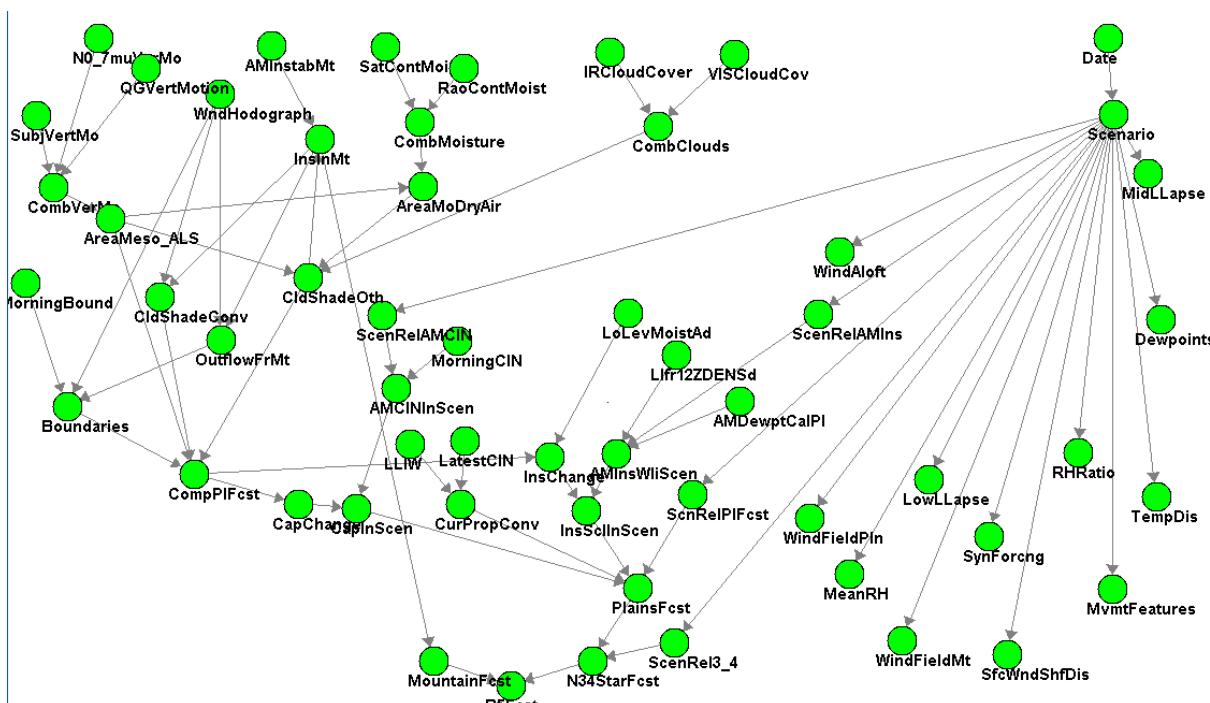
- Conjugate priors
 - Beta / Dirichlet
 - Predictions, updating of hyperparameters
- Meta-BN encoding parameters as variables
- Choice of hyperparameters
 - BDe prior \Rightarrow *Score consistency*
- Decomposability of scores and implications
- Local search
 - On graphs
 - On orderings (optimal for 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.)



HailFinder

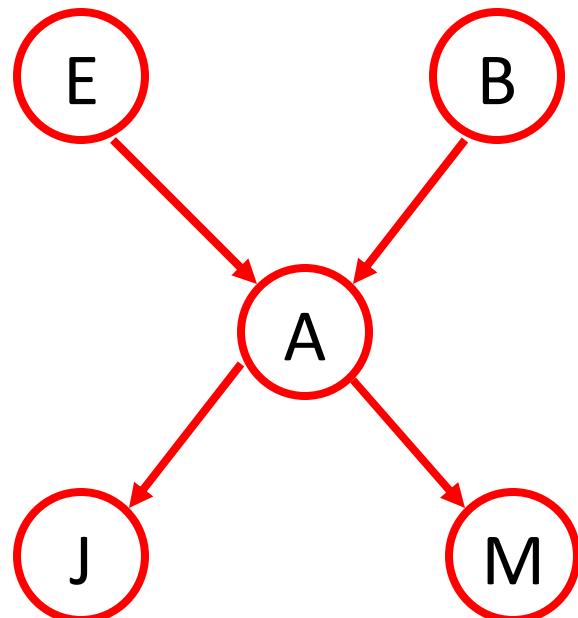
56 vars

~ 3 states each

→ ~ 10^{26} terms
>> 10.000 years
on Top
supercomputers

JavaBayes applet

Typical queries: Conditional distribution



- Compute distribution of some variables given values for others

Observe $M=T$

Compute $P(E=T | M=T)$

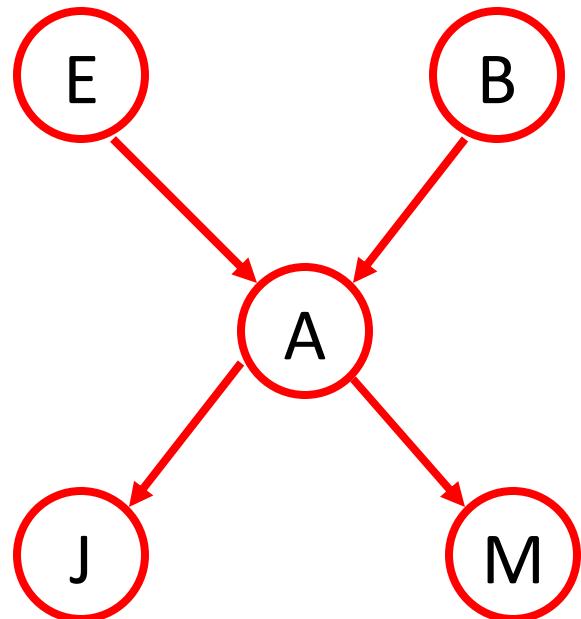
$$P(E=T | M=T) = \frac{P(E=T, M=T)}{P(M=T)}$$

$$P(E=T, M=T) = \sum_b \sum_a \underbrace{\sum_j P(E=T, M=T, B=b, A=a, J=j)}_{P(E)P(B)P(A|EB)P(J|A)P(M|A)}$$

$\overbrace{\hspace{10em}}^{2^3 \text{ terms}}$

Naive approach exponential in # vars ...

Typical queries: Maximization



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

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

- MAP (Maximum a posteriori):
Compute most likely assignment to
some variables

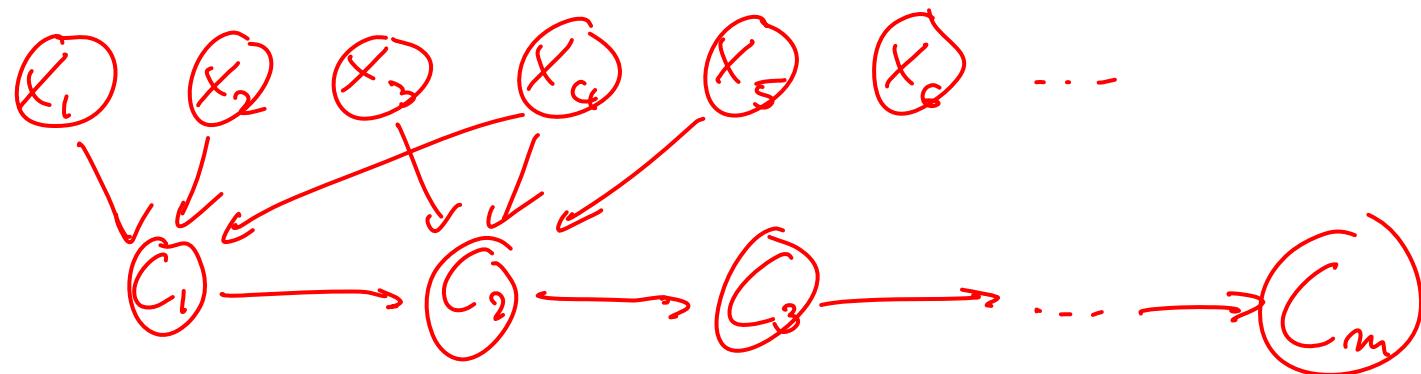
$$e^* = \underset{e}{\operatorname{argmax}} P(J=F, M=T, E=e) = \\ = \underset{e, a, b}{\operatorname{argmax}} \sum_{a, b} P(J=F, M=T, e, a, b)$$

Hardness of computing conditional prob.

- Computing $P(X=x | E=e)$ is NP-hard

- **Proof:** by reduction from 3SAT

Given boolean formula $\varphi = (x_1 \vee x_2 \vee x_4) \wedge (\neg x_3 \vee x_4 \vee x_5) \wedge \dots$
does there exist a sat. assignment to x_1, \dots, x_m



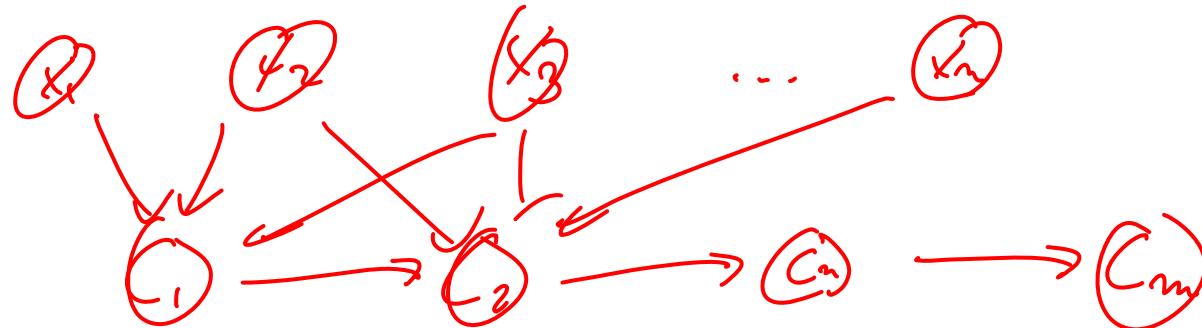
$$P(x_i) = \text{Bern}(0.5)$$

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

$$\varphi \text{ sat.} \Leftrightarrow P(C_m = T) > 0$$

Hardness of computing cond. prob.

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



$$P(C_m = T) = \sum_{x_1 \dots x_m} P(x_1 \dots x_m) \cdot P(C_m = T \mid x_1 \dots x_m)$$

$\frac{1}{2^m}$ | if $x_1 \dots x_m$ sat.
 0 otherwise

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

$\Rightarrow \#P$ hardness

Hardness of inference for general BNs

- Computing conditional distributions:

- Exact solution: #P-complete

- Approximate solution: NP-hard:

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

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

- Maximization:

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

$$\max_{x_1, \dots, x_m} \sum_{e_1, \dots, e_m}$$

- Inference in general BNs is really hard 😞

- Is all hope lost?

Inference

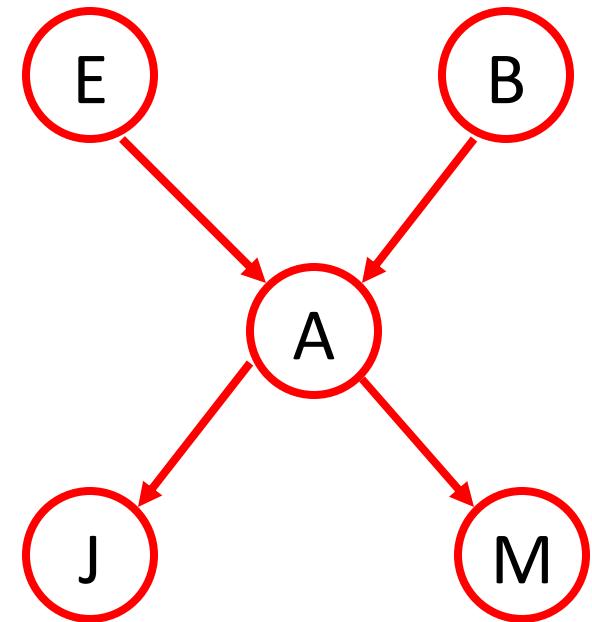
- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations
- For BNs where exact inference is not possible, can use algorithms for **approximate inference** (later this term)

Computing conditional distributions

- Query: $P(X | E=e)$

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

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



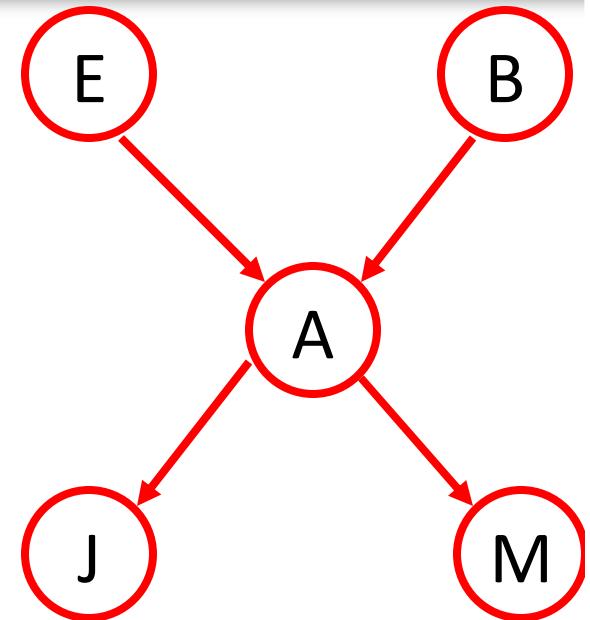
Inference example

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

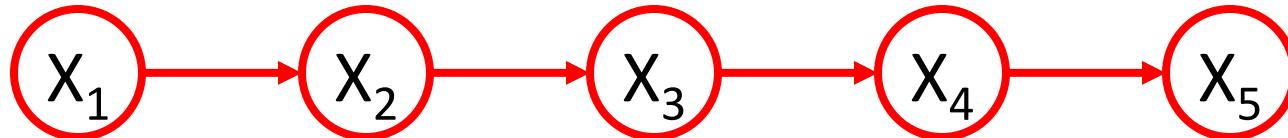
$$= \sum_{a, j, b} P(E, m, a, j, b)$$

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

is generally exponentially many terms



Potential for savings: Variable elimination!



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

distributivity

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

How many additions: 3

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_5 \quad \xrightarrow{g_4(X_3, X_4)} = P(X_5 | X_3)$$

$$X_1 \rightarrow X_2 \rightarrow X_5 \quad \xrightarrow{g_3(X_2, X_5)} = P(X_5 | X_3)$$

$$\xrightarrow{g_2(X_1, X_5)} = P(X_5 | X_1)$$

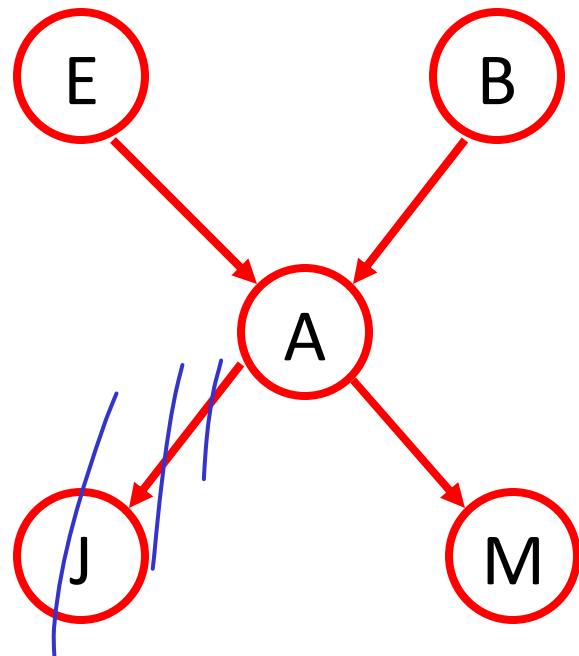
$$X_1 \rightarrow X_5$$

Intermediate solutions are distributions on fewer variables!

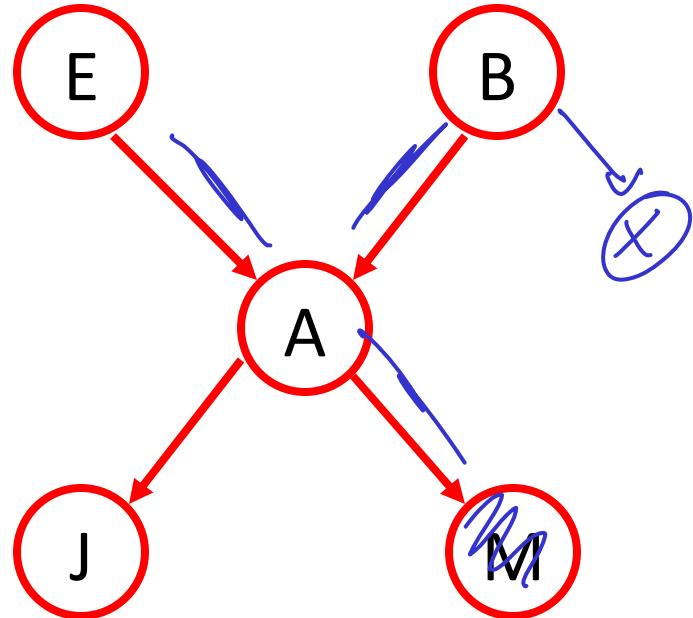
Variable elimination in general graphs

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

$$\begin{aligned} P(E, m) &= \sum_{b,a,j} P(E) P(b) P(a|E,b) P(j|a) P(m|a) \\ &= P(E) \sum_b P(b) \sum_a P(a|E,b) P(m|a) \sum_j P(j|a) \\ &\quad \underbrace{\qquad\qquad\qquad}_{g_A(E, b, m)} \qquad\qquad\qquad \underbrace{\qquad\qquad\qquad}_{g_B(E, m)} \end{aligned}$$



Removing irrelevant variables



$$P(E, m) = \sum_{\dots} \sum_x P(x|b) \quad \sum_d P(d|A)$$

$\underbrace{\quad}_{=1} \qquad \qquad \underbrace{\quad}_{=1}$

Delete nodes not on active trail between query vars.

Variable elimination algorithm

- Given BN and Query $P(X | \underline{E=e})$
- Remove irrelevant variables for $\{X, e\}$
- Choose an ordering of X_1, \dots, X_n
- Set up initial factors: $f_i = P(X_i | \text{Pa}_i)$
- For $i = 1:n$, $X_i \notin \{X, E\}$
 - 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

| | | |
|---|---|---|
| A | T | F |
| B | T | F |
| F | . | . |

| | | |
|---|---|---|
| B | T | F |
| C | T | F |
| F | . | . |

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

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

$$f' = f_1 \cdot f_2$$

$$f'(A, B, C)$$

| BC | TT | TF | FT | FF |
|----|----|----|----|----|
| A | T | . | . | . |
| T | . | . | . | . |
| F | . | . | . | . |

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

Marginalizing factors

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

f_i

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

| A\B | T | F |
|-----|---|---|
| T | | |
| F | | |

$$\begin{array}{c|c} B & g(B) \\ \hline T & \\ F & \end{array}$$

←

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

Tasks

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

Probabilistic Graphical Models

Lecture 7 –Variable Elimination

CS/CNS/EE 155
Andreas Krause

Announcements

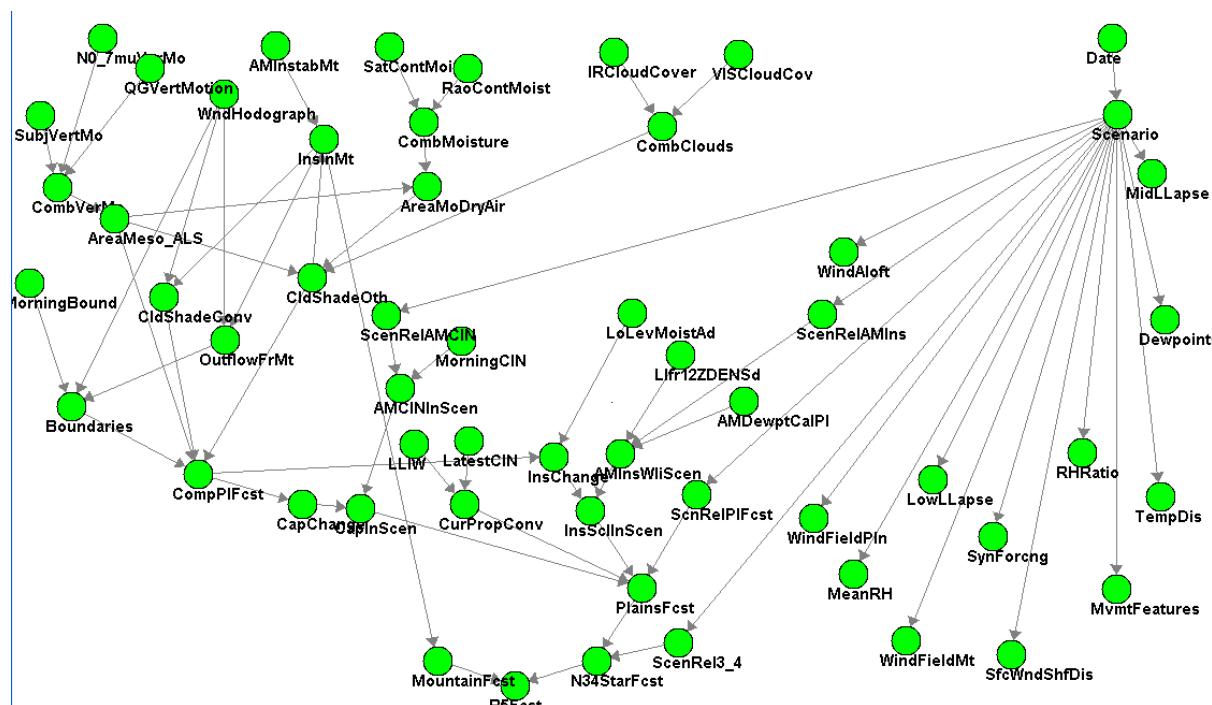
- Homework 1 due today in class
- Will get back to you soon with feedback on project proposals.

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.)



HailFinder

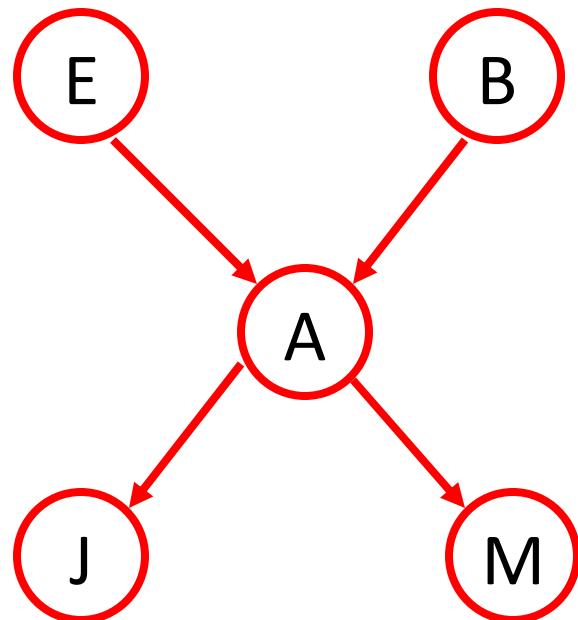
56 vars

~ 3 states each

→ ~ 10^{26} terms
>> 10.000 years
on Top
supercomputers

JavaBayes applet

Typical queries: Conditional distribution



- Compute distribution of some variables given values for others

Observe $M=T$

Compute $P(E=T | M=T)$

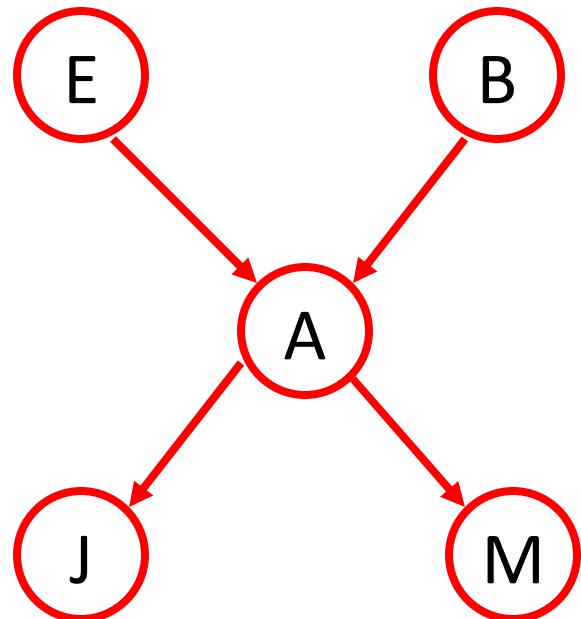
$$P(E=T | M=T) = \frac{P(E=T, M=T)}{P(M=T)}$$

$$P(E=T, M=T) = \sum_b \sum_a \underbrace{\sum_j P(E=T, M=T, B=b, A=a, J=j)}_{P(E)P(B)P(A|EB)P(J|A)P(M|A)}$$

$\overbrace{\quad\quad\quad}^{2^3 \text{ terms}}$

Naive approach exponential in # vars ...

Typical queries: Maximization



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

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

- MAP (Maximum a posteriori):
Compute most likely assignment to
some variables

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

Hardness of inference for general BNs

- Computing conditional distributions:

- Exact solution: #P-complete

- Approximate solution: NP-hard:

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

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

- Maximization:

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

$$\max_{x_1, \dots, x_m} \sum_{e_1, \dots, e_m}$$

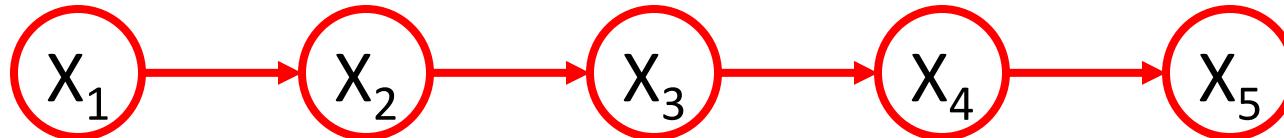
- Inference in general BNs is really hard 😞

- 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)

Potential for savings: Variable elimination!



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

distributivity

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

How many additions: 3

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_5 \quad \xrightarrow{g_4(X_3, X_4)} = P(X_5 | X_3)$$

$$X_1 \rightarrow X_2 \rightarrow X_5 \quad \xrightarrow{g_3(X_2, X_5)} = g_3(X_2, X_5) \quad |$$

$$\xrightarrow{g_2(X_1, X_5)} = g_2(X_1, X_5) \quad |$$

$$\xrightarrow{g_1(X_1)} = g_1(X_1) \quad |$$

Intermediate solutions are distributions on fewer variables!

Variable elimination algorithm

- Given BN and Query $P(X | \underline{E=e})$
- Remove irrelevant variables for $\{X, e\}$
- Choose an ordering of X_1, \dots, X_n
- Set up initial factors: $f_i = P(X_i | \underline{\text{Pa}_i})$
- For $i = 1:n$, $X_i \notin \{X, E\}$
 - 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

| | | |
|---|---|---|
| A | T | F |
| B | T | F |
| F | . | . |

| | | |
|---|---|---|
| B | T | F |
| C | T | F |
| F | . | . |

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

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

$$f' = f_1 \cdot f_2$$

$$f'(A, B, C)$$

| BC | TT | TF | FT | FF |
|----|----|----|----|----|
| A | T | . | . | . |
| I | T | . | . | . |
| F | . | . | . | . |

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

Marginalizing factors

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

f_i

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

| A\B | T | F |
|-----|---|---|
| T | | |
| F | | |

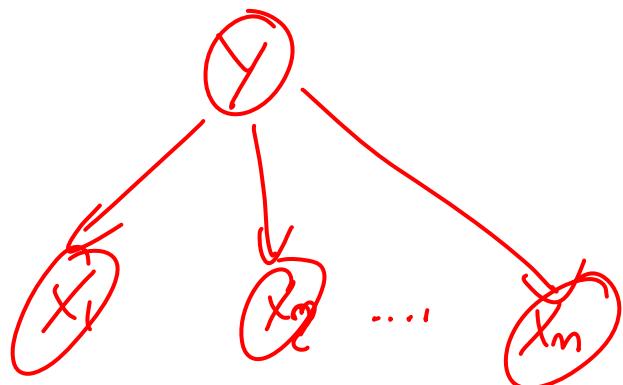
$$\begin{array}{c|c} B & g(B) \\ \hline T & \\ F & \end{array}$$

←

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

Does the order matter?

If elim. X_2 first



$$P(X_m | X_1)$$

$$g = \sum_{X_2} P(X_2 | Y)$$

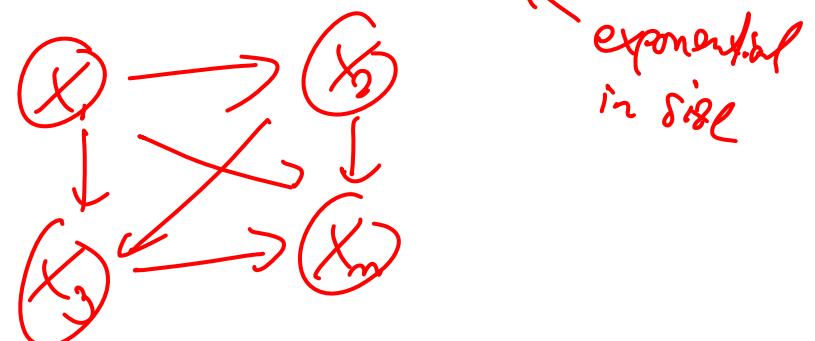
$$\text{Scope}(g) = \{X_2, Y\} = \text{Scope}(P(X_2 | Y))$$

YES !

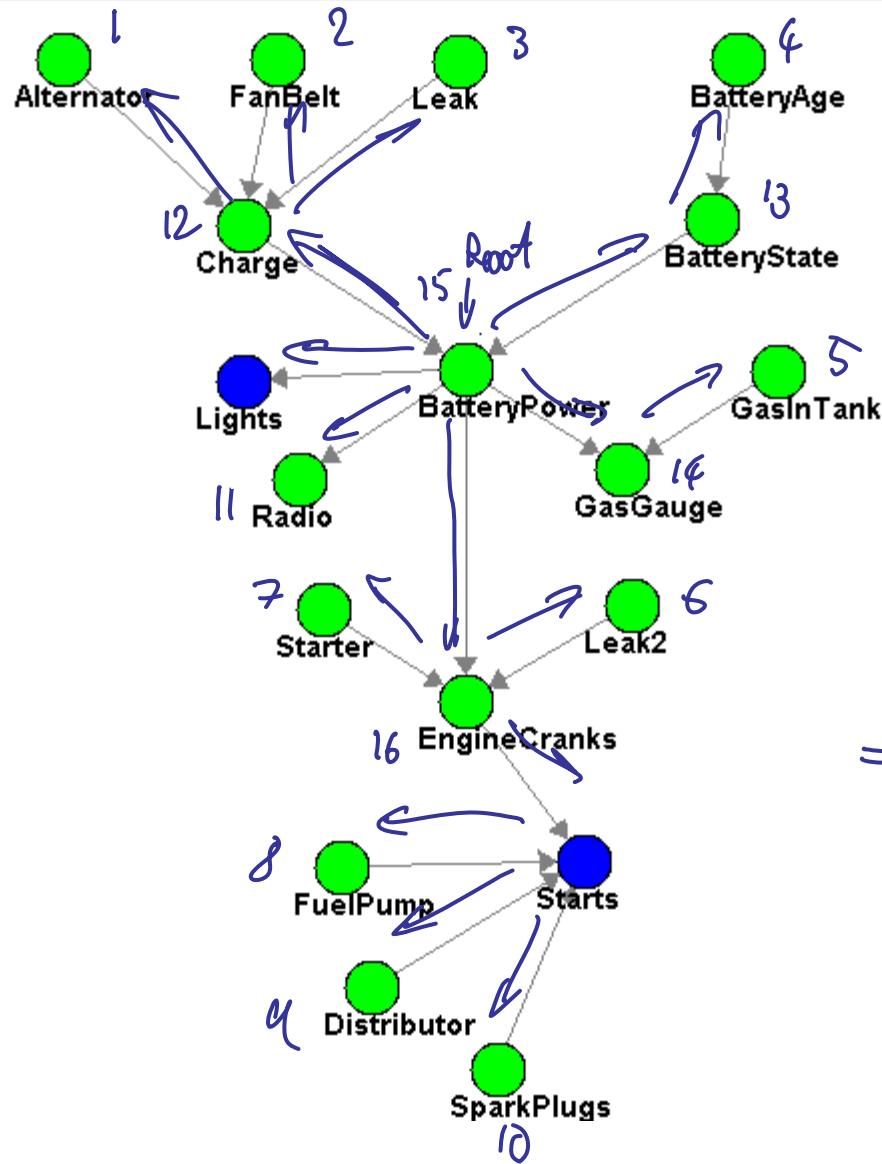
If elim. Y first

$$g = \sum_Y \left(\prod_j P(X_j | Y) \right) P(Y)$$

$$\text{Scope}(g) = \{X_1, \dots, X_m\}$$



Variable elimination for polytrees



BNG Polytree if skeleton (G) is tree

Drop edge directions

Pick some root in skeleton

Direct edges outward from root
not in query

Eliminate vars in reverse topological order of resulting directed tree

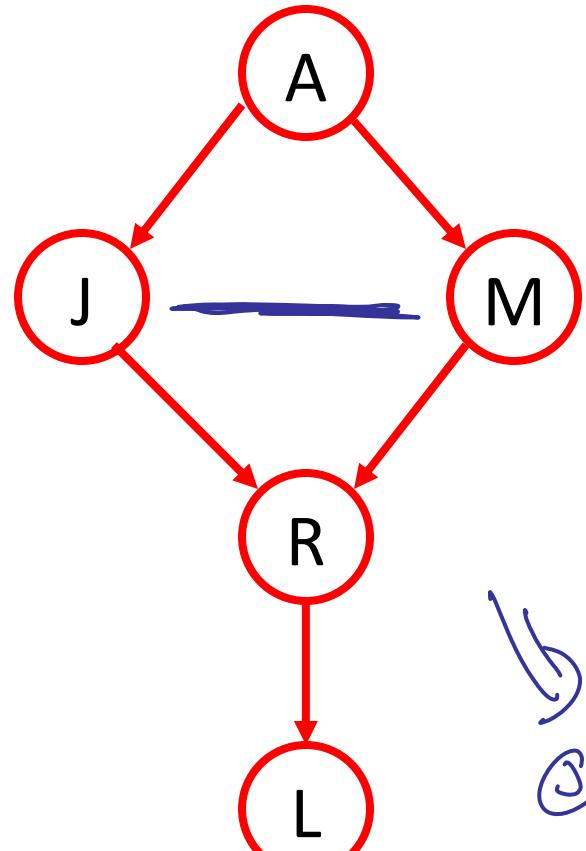
Factor sizes do not increase

\Rightarrow Inference possible in $O(n)$ steps

Complexity of variable elimination

- Tree graphical models
 - Using correct elimination order, factor sizes do not increase!
 - Inference in linear time!!
- General graphical models
 - Ultimately NP-hard..
 - Need to understand what happens if there are loops

Variable elimination with loops



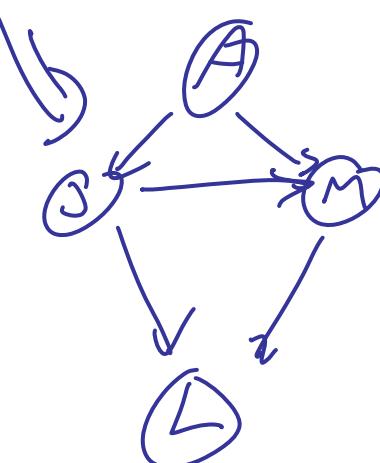
$$P(A, L)$$

Elim. order R, J, M

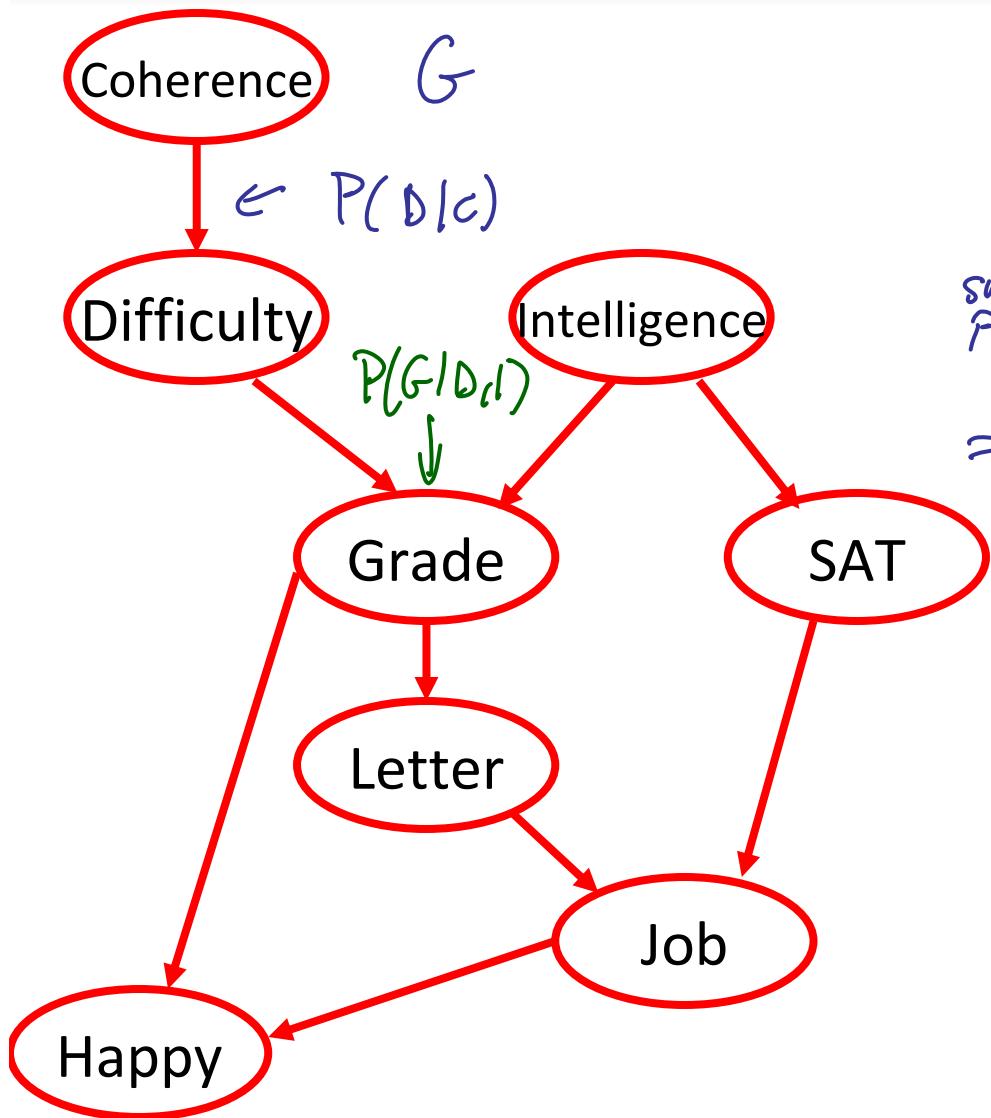
$$P(A, L) = \sum_{r \in j \in m} P(A) P(J|A) P(M|A) P(R|JM) P(L|R)$$

$$= \sum_{j \in m} P(A) P(J|A) P(M|A) \sum_r P(r|J, m) \underbrace{P(L|r)}_g$$

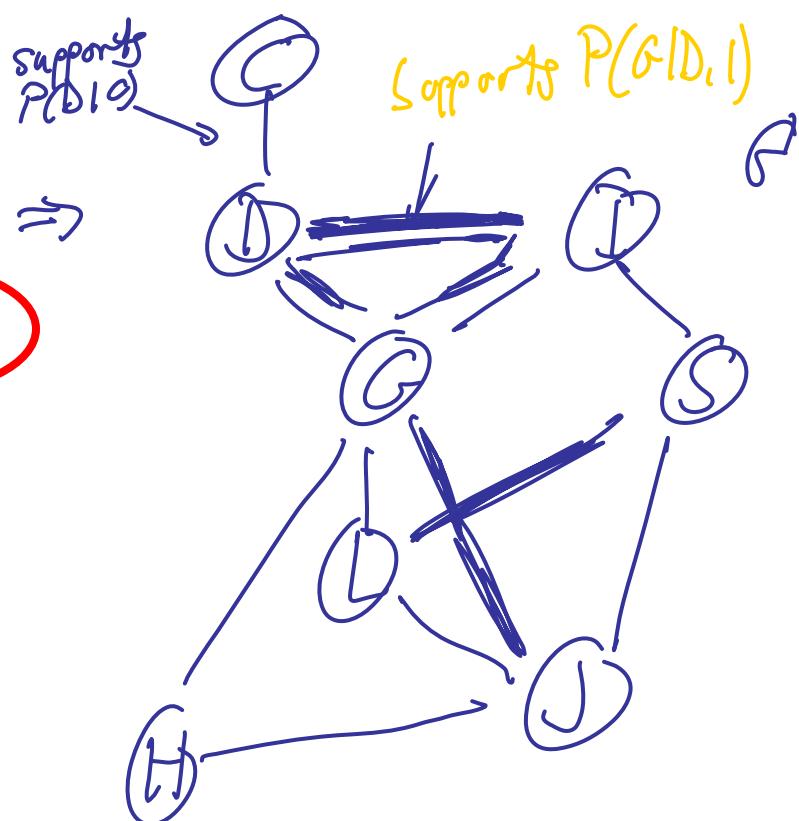
$$\text{Scope } (g) = \{L, J, M\}$$



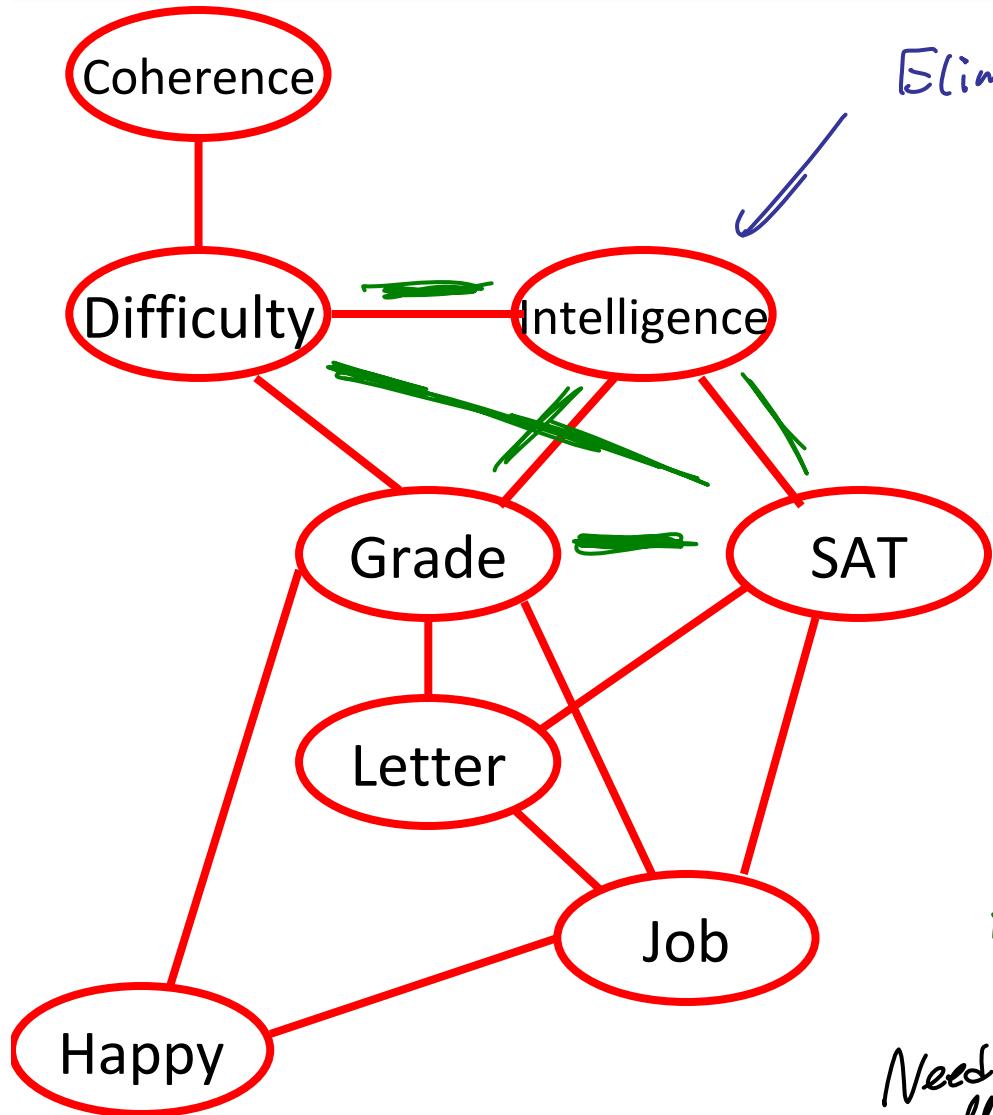
Elimination as graph transformation: Moralization



Want that all factors of G are supported on cliques of G'



Elimination: Filling edges



Eliminate

⇒ Introduce new factor

$$g = \sum_I \prod_{\text{Cliques } C \text{ incident with } I} f_C$$

$$\text{Scope}(g) = \bigcup_{\text{Cliques } C \text{ inc.w } I} \text{Scope}(f_C) \setminus \{I\}$$

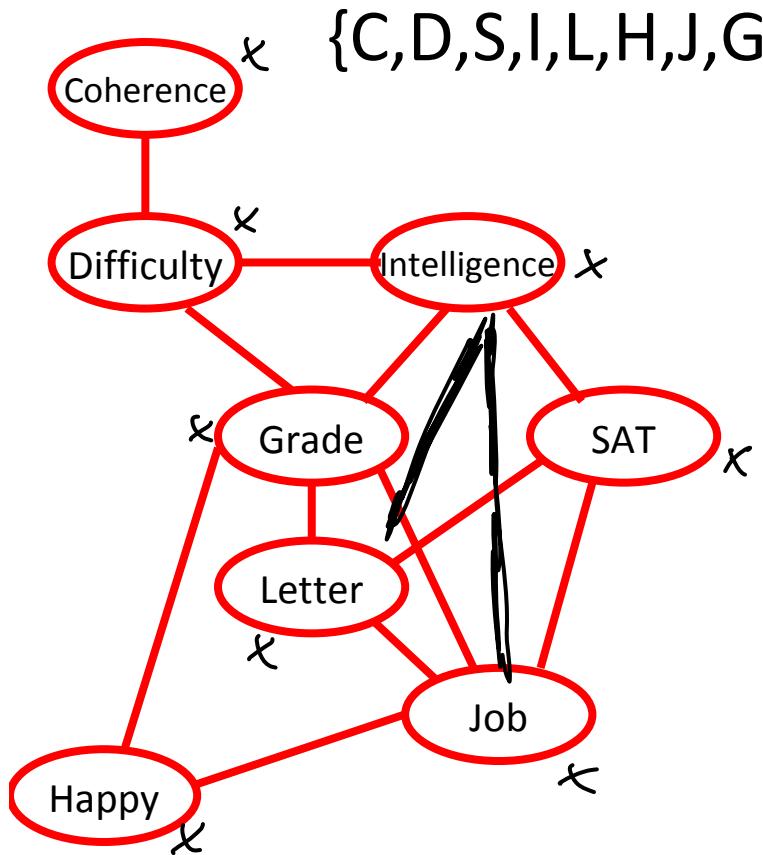
Want to make sure that resulting graph supports g

$$\text{Here: Scope}(g) = \{D, G, S\}$$

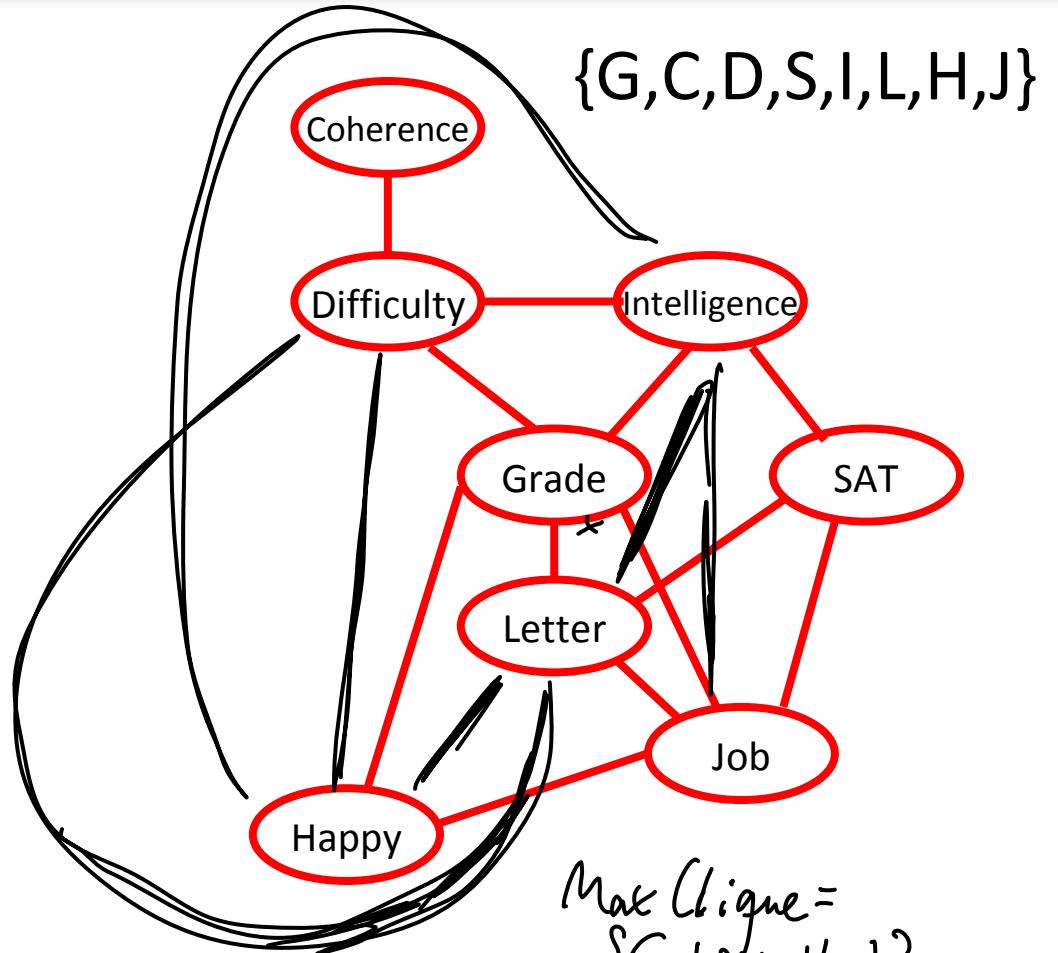
Not supported!

Need to fill in edges by connecting all neighbors of I into clique

Impact of elimination order



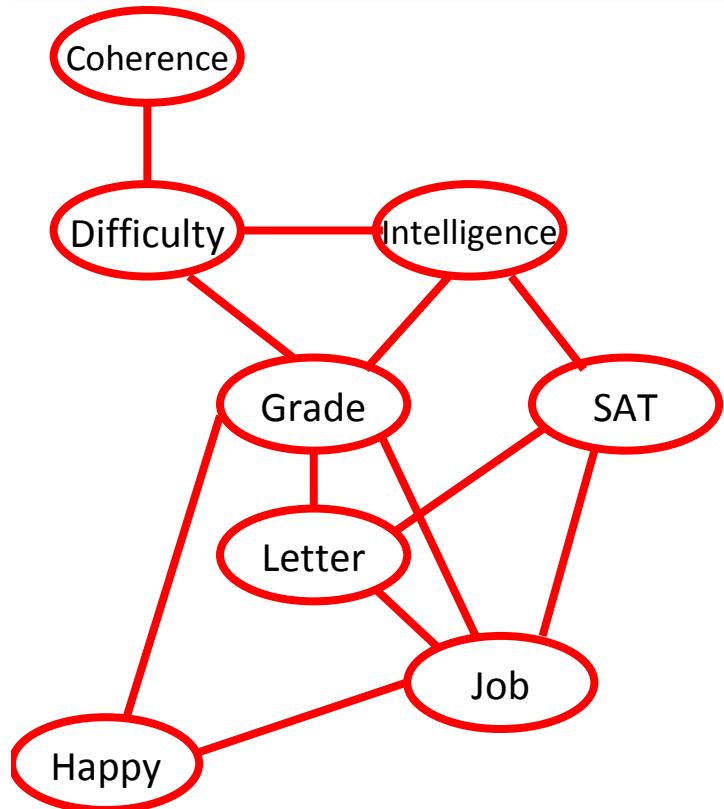
$\text{Max Clique} = \{I, G, L, J\}$
 $\Rightarrow \text{Tree width } 3$



$\text{Max Clique} = \{G, I, D, L, H, J\}$
 $\Rightarrow \text{Tree width } 5$

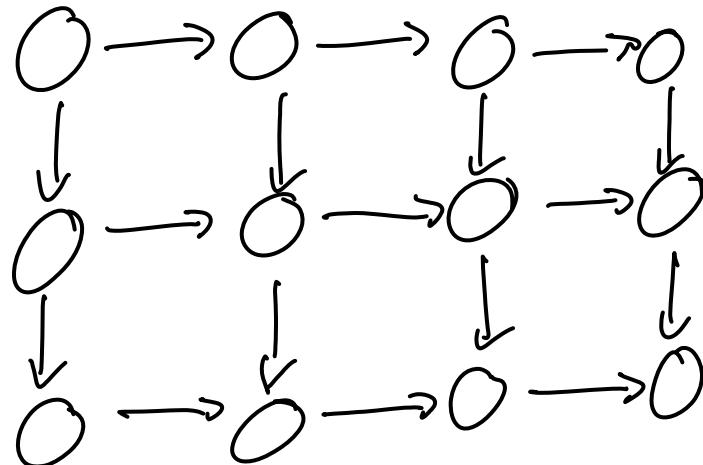
- Different elim. order induce different graphs! Tree width 5

Induced graph and VE complexity

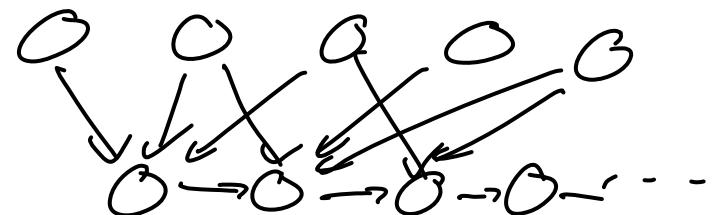


- **Theorem:**
 - All factors arising in VE are defined over cliques (fully connected subgraphs) of the induced graph
 - All maximal cliques of induced graph arise as factors in VE
- Treewidth for ordering = Size of largest clique of induced graph -1
- Treewidth of a graph = minimal treewidth under optimal ordering
- VE exponential in treewidth!

Compact representation → small treewidth?



$n \times n$ grid
treewidth n
 $\frac{\# \text{ parents}}{\text{node}} = 2$



compact
~~# parents~~ ≤ 4

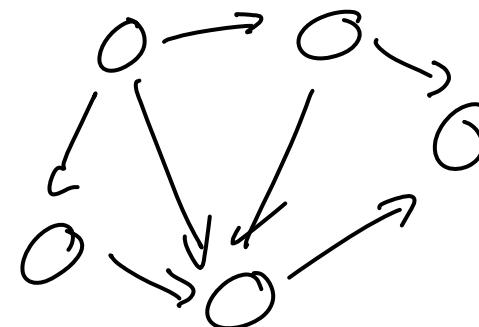
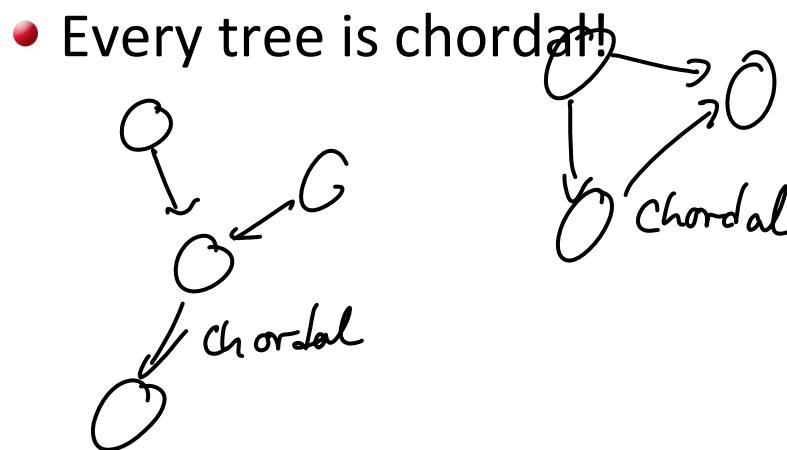
treewidth ??

Finding the optimal elimination order

- **Theorem:** Deciding whether there exists an elimination order with induced width at most K is NP-hard
 - Proof by reduction from MAX-CLIQUE
- In fact, can find elimination order in time exponential in treewidth
- Finding optimal ordering as hard as inference...
- For which graphs can we find optimal elimination order?

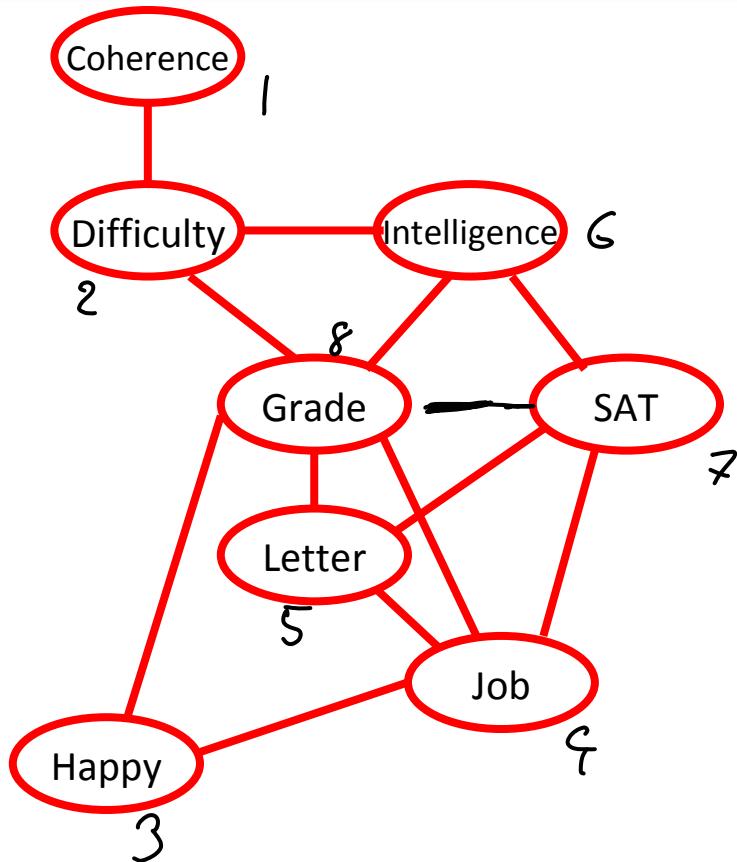
Finding optimal elimination order

- For trees can find optimal ordering (^{poly} saw before)
- A graph is called chordal if every cycle of length ≥ 4 has a chord (an edge between some pair of non-consecutive nodes)



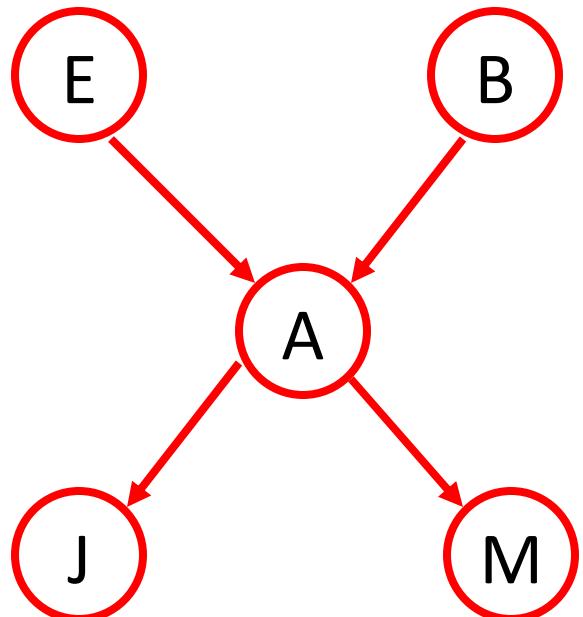
- Can find optimal elimination ordering for chordal graphs

Minimal fill heuristic



- Unmark all nodes
- For $i = 1:n$
 - Find unmarked node X such that adding X adds fewest additional edges
 - Set X as i -th var. in ordering
 - Fill in edges added by eliminating X
 - Mark X
- Often very effective!
- In fact, finds optimal order for chordal graphs!
- May want to weigh by factor size

Maximization queries



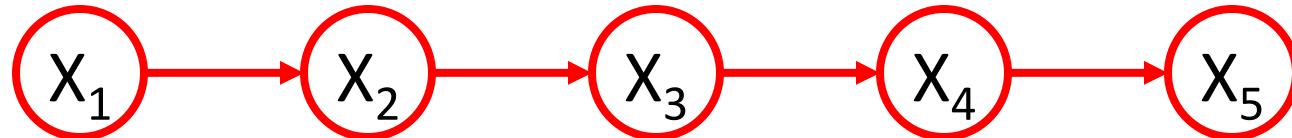
- MPE (Most probable explanation):
Given values for some vars,
compute most likely assignment to
all remaining vars

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

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

$$\underset{x_{\{1\dots n\} \setminus E}}{\operatorname{argmax}} P(x_1, \dots, x_n | e) = \underset{x}{\operatorname{argmax}} P(x_1, \dots, x_n | e)$$

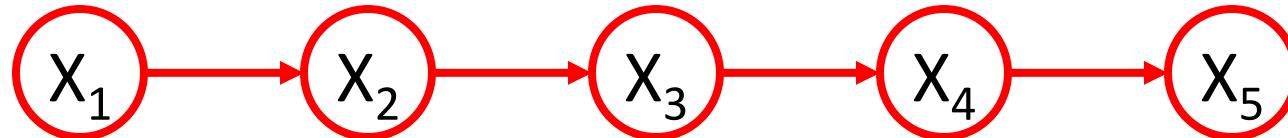
Variable elimination for MPE



$$\begin{aligned} & \max_{x_2, x_3, x_4} P(x_1) P(x_2|x_1) P(x_3|x_2) P(x_4|x_3) P(x_5|x_4) \\ = & P(x_1) \max_{x_2} P(x_2|x_1) \max_{x_3} P(x_3|x_2) \max_{x_4} \underbrace{P(x_4|x_3) P(x_5|x_4)}_{g_4(x_3, x_5)} \\ & \qquad\qquad\qquad \underbrace{g_3(x_2, x_5)}_{g_2(x_1, x_5)} \end{aligned}$$

How can we retrieve the array \max ??

Recovering the MPE



$$x_1^* = \underset{x_1}{\operatorname{argmax}} \ P(x_1) \ g_2(x_1, x_5)$$

$$x_2^* = \underset{x_2}{\operatorname{argmax}} \ \cancel{P(x_2)} \ P(x_2 | x_1^*) \ g_3(x_2, x_5)$$

$$x_3^* = \underset{x_3}{\operatorname{argmax}} \ P(x_3 | x_2^*) \ g_4(x_3, x_5)$$

⋮
⋮

Variable elimination for MPE: Forward pass

- Given BN and MLE query $\max P(x_1, \dots, x_n \mid E=e)$
- Choose an ordering of X_1, \dots, X_n
- Set up initial factors: $f_i = P(X_i \mid Pa_i)$
- For $i = 1:n$, $X_i \notin \{X, E\}$
 - Collect all factors f that include X_i
 - Generate new factor by maximizing over X_i

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

- Add g to set of factors

Variable elimination for MPE: Backward pass

- Variables x_1^*, \dots, x_n^* will contain MPE
- For $i = \underline{n:1}$, $X_i \notin \{X, E\}$
 - Take factors f_1, \dots, f_m used when eliminating X_i
 - Plug in values x_{i+1}^*, \dots, x_n^* into these factors

$$x_i^* = \operatorname{argmax}_{x_i} \prod_j f_j(x_i, x_{i+1}^*, \dots, x_n^*)$$

\uparrow can only depend on x_i

Maximizing factors

$$g = \max_{x_i} \prod_{\substack{j \\ f_j \neq 1}} f_j$$

f'

| | | | |
|---|-----|---|---|
| | A/B | T | F |
| T | | : | : |
| P | | : | : |

$$g = \max_A f'$$

$$g(B) = \max_A f'(A, B)$$

Summary so far

- Variable elimination complexity exponential in induced width for elimination ordering
- Finding optimal ordering is NP-hard
- Many good heuristics
 - Exact for trees, chordal graphs
- Ultimately, inference is NP-hard
- Only difference between cond. prob. queries and MPE is \sum vs. max
- Variable elimination building block for many exact and approximate inference techniques

Tasks

- Homework 2 due in class Wednesday Nov 4

Probabilistic Graphical Models

Lecture 8 – Junction Trees

CS/CNS/EE 155
Andreas Krause

Announcements

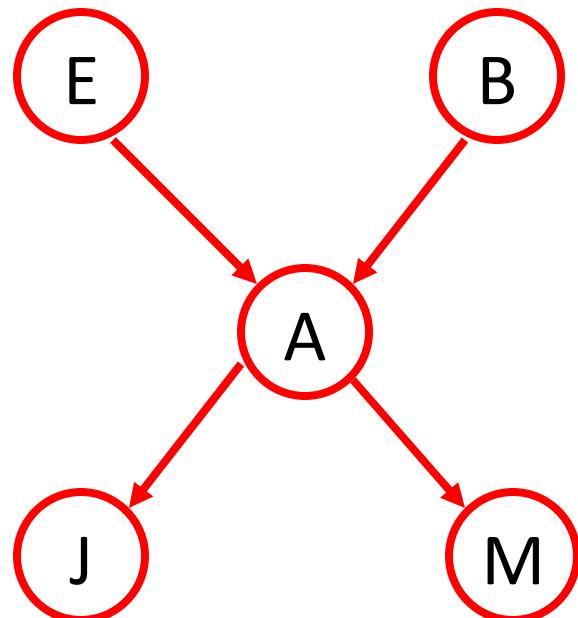
- Homework 2 due next Wednesday (Nov 4) in class
 - Start early!!!
- Project milestones due Monday (Nov 9)
 - 4 pages of writeup, NIPS format
 - <http://nips.cc/PaperInformation/StyleFiles>

Best project award!!

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

Typical queries: Conditional distribution



- Compute distribution of some variables given values for others

Observe $M=T$

Compute $P(E=T | M=T)$

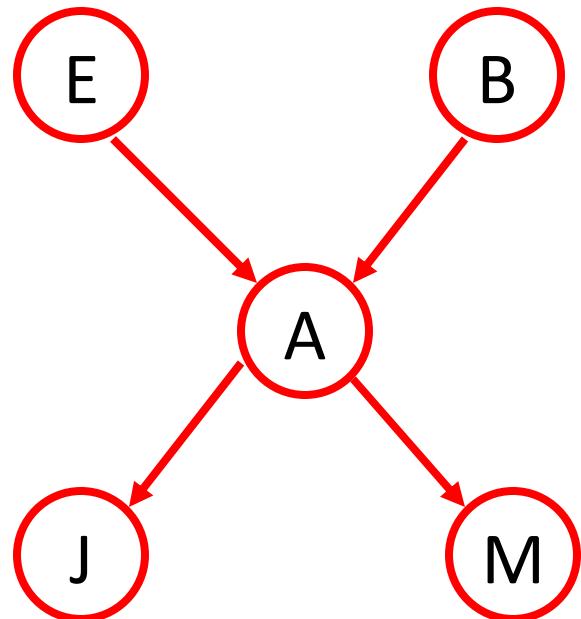
$$P(E=T | M=T) = \frac{P(E=T, M=T)}{P(M=T)}$$

$$P(E=T, M=T) = \sum_b \sum_a \underbrace{\sum_j P(E=T, M=T, B=b, A=a, J=j)}_{P(E)P(B)P(A|EB)P(J|A)P(M|A)}$$

$\overbrace{\quad\quad\quad}^{2^3 \text{ terms}}$

Naive approach exponential in # vars ...

Typical queries: Maximization



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

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

- MAP (Maximum a posteriori):
Compute most likely assignment to
some variables

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

Hardness of inference for general BNs

- Computing conditional distributions:

- Exact solution: #P-complete

- Approximate solution: NP-hard:

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

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

- Maximization:

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

$$\max_{x_1, \dots, x_m} \sum_{e_1, \dots, e_m}$$

- 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)

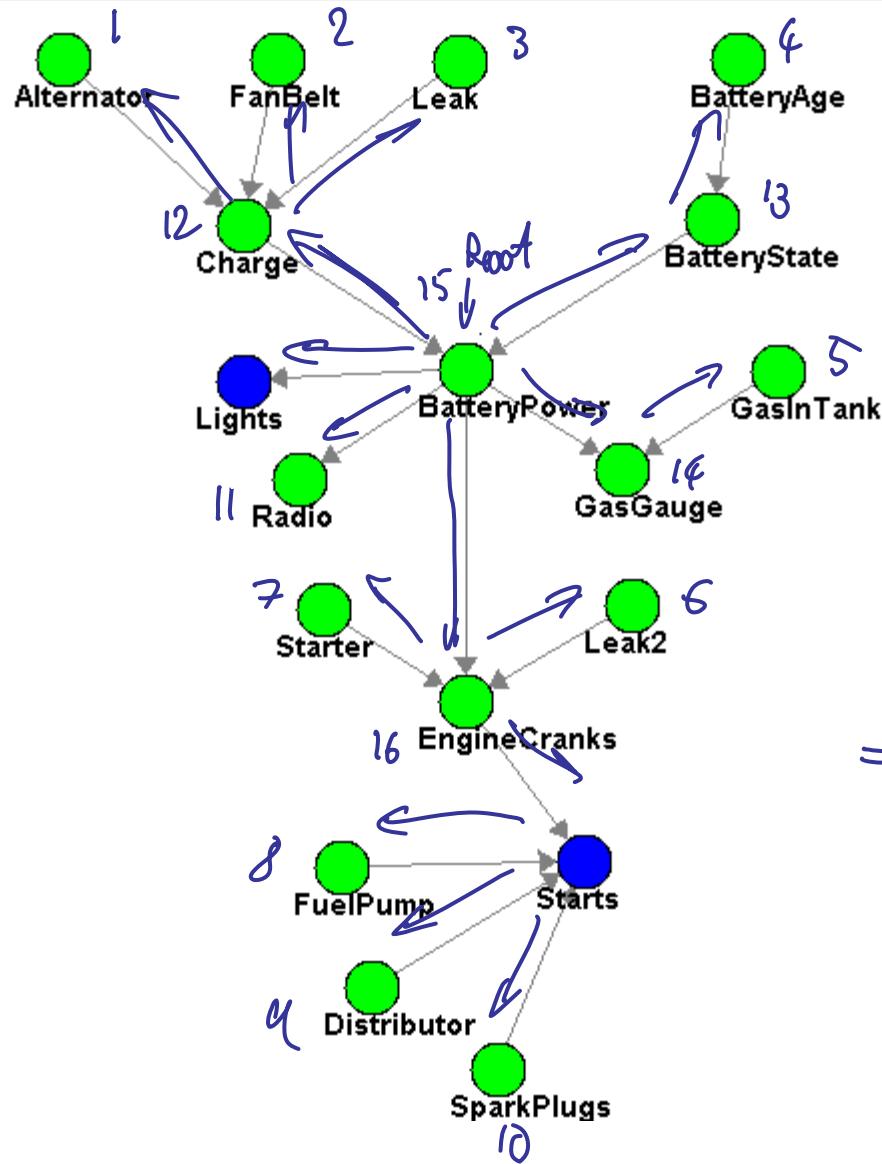
Variable elimination algorithm

- Given BN and Query $P(X | \underline{E=e})$
- Remove irrelevant variables for $\{X, e\}$
- Choose an ordering of X_1, \dots, X_n
- Set up initial factors: $f_i = P(X_i | \underline{\text{Pa}_i})$
- For $i = 1:n$, $X_i \notin \{X, E\}$
 - 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 $P(x | e)$

Variable elimination for polytrees



BNG Polytree if skeleton (G) is tree

Drop edge directions

Pick some root in skeleton

Direct edges outward from root
not in query

Eliminate vars in reverse topological order of resulting directed tree

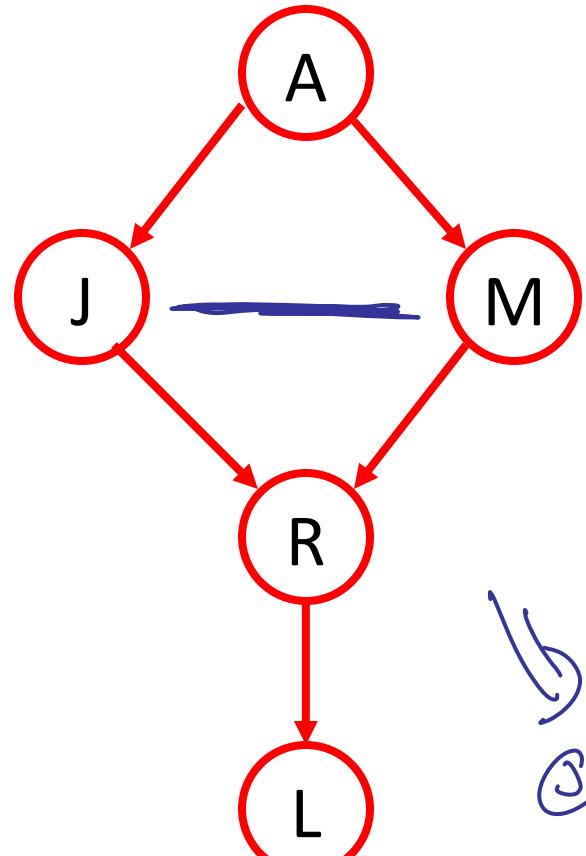
Factor sizes do not increase

\Rightarrow Inference possible in $O(n)$ steps

Complexity of variable elimination

- Tree graphical models
 - Using correct elimination order, factor sizes do not increase!
 - Inference in linear time!!
- General graphical models
 - Ultimately NP-hard..
 - Need to understand what happens if there are loops

Variable elimination with loops



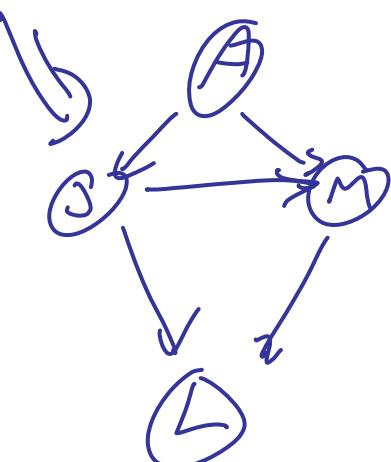
$$P(A, L)$$

Elim. order R, J, M

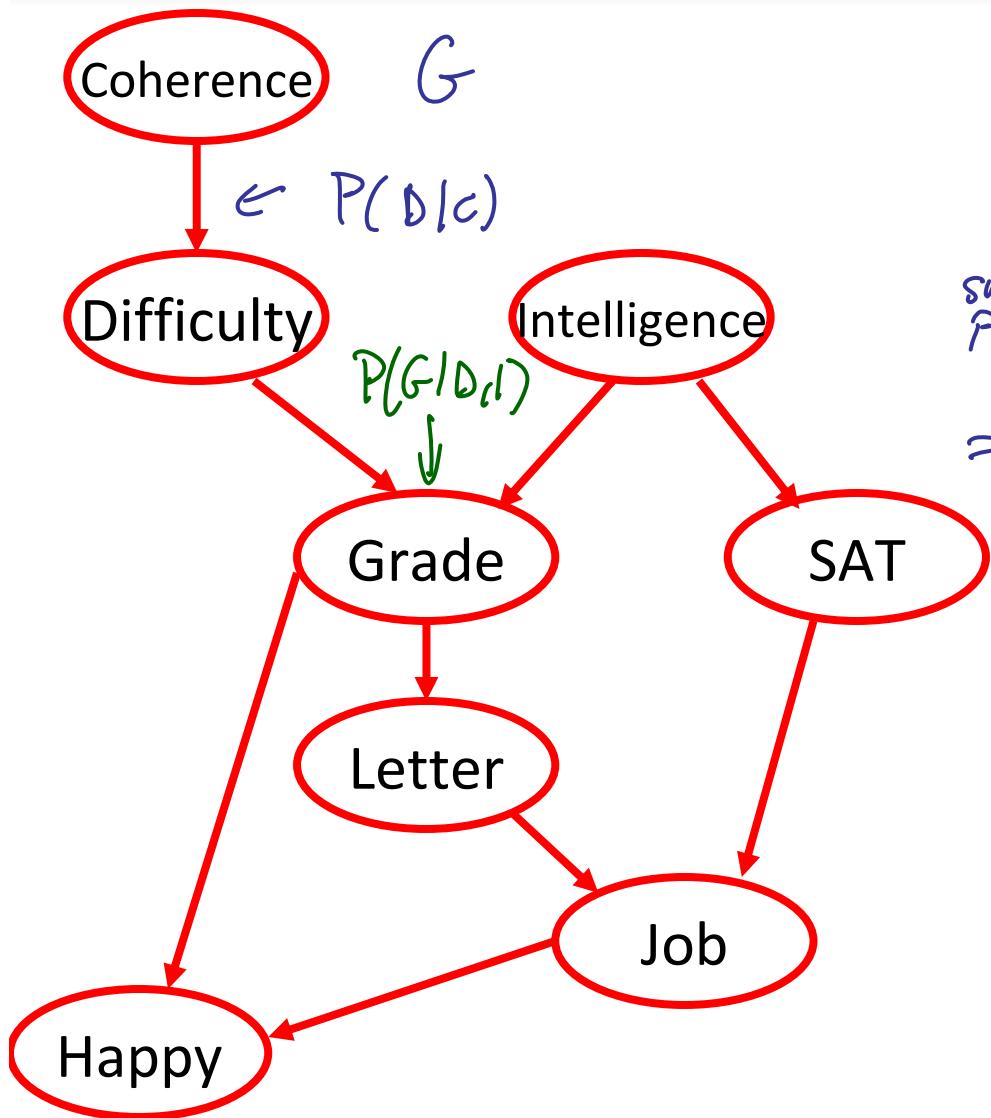
$$P(A, L) = \sum_{r \in j \in m} P(A) P(J|A) P(M|A) P(R|JM) P(L|R)$$

$$= \sum_{j \in m} P(A) P(J|A) P(M|A) \sum_r P(r|J, m) P(L|r)$$

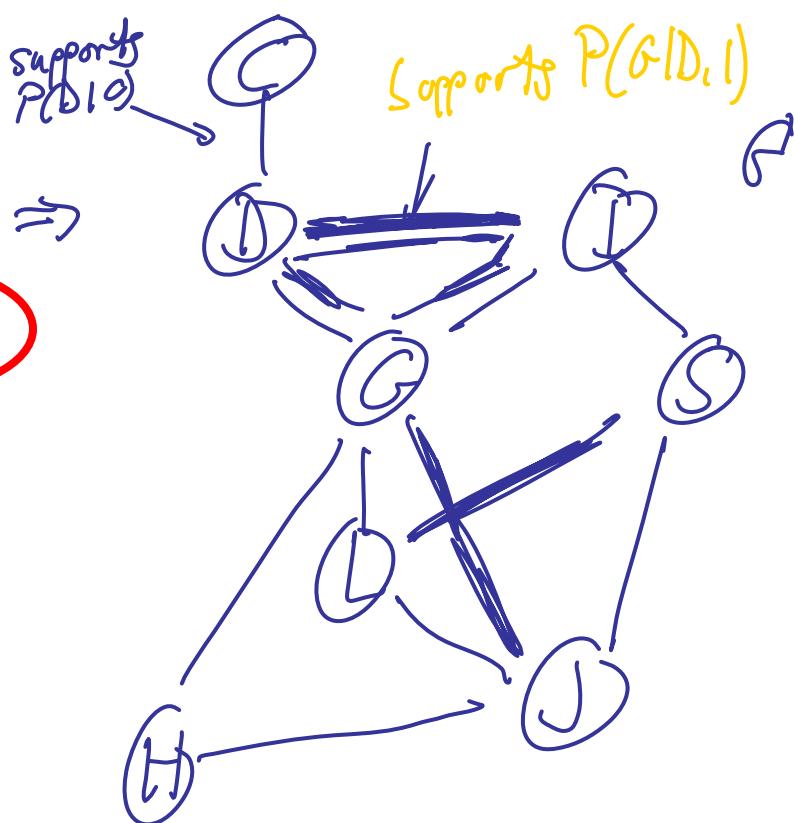
Scope (g) = $\{L, J, M\}$



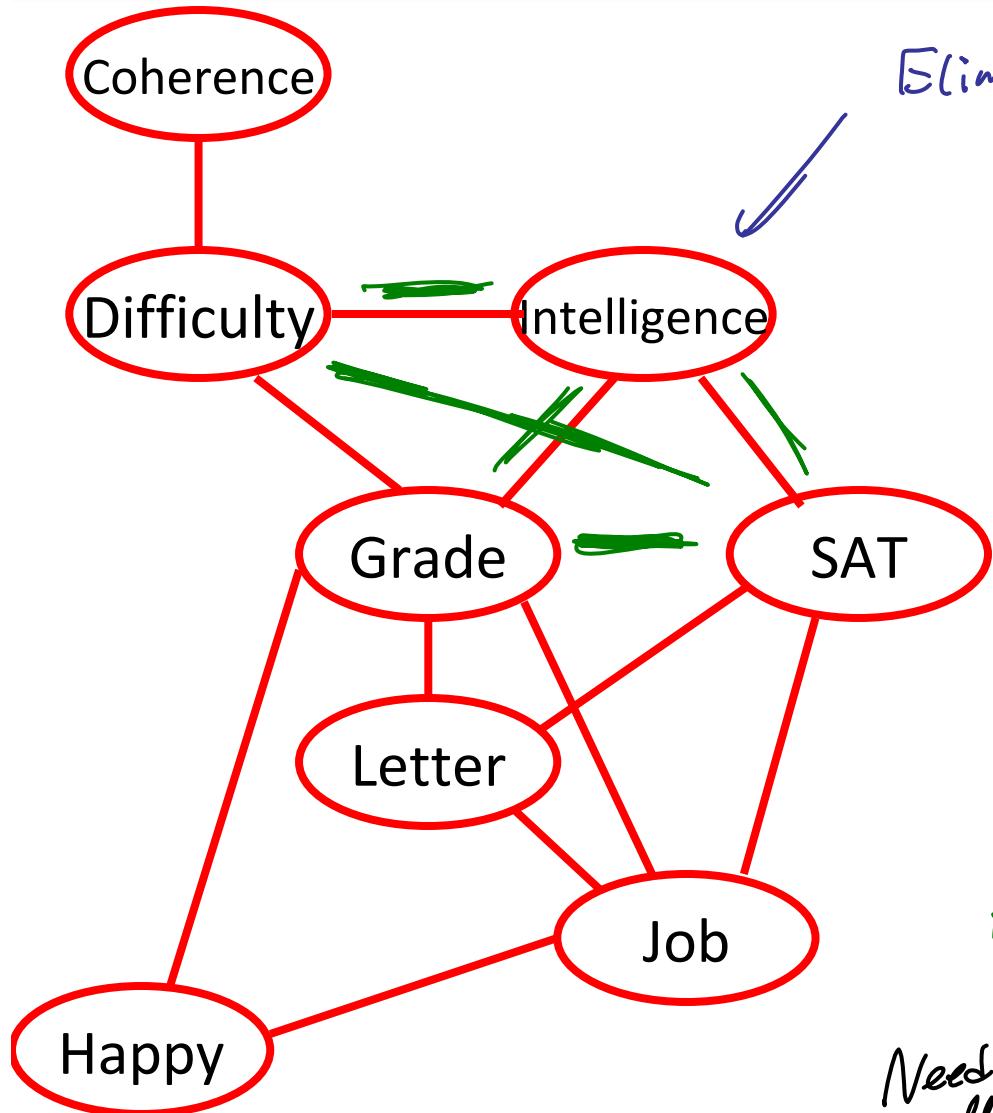
Elimination as graph transformation: Moralization



Want that all factors of G are supported on cliques of G'



Elimination: Filling edges



Eliminate

⇒ Introduce new factor

$$g = \sum_I \prod_{\text{Cliques } C \text{ incident with } I} f_C$$

$$\text{Scope}(g) = \bigcup_{\text{Cliques } C \text{ inc. w/ } I} \text{Scope}(f_C) \setminus \{\emptyset\}$$

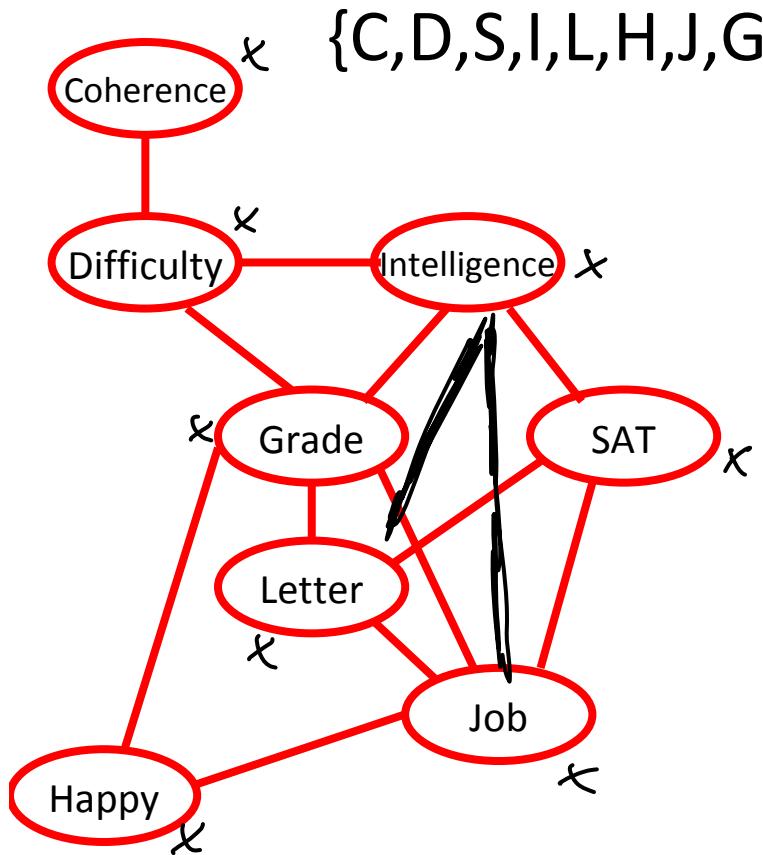
Want to make sure that resulting graph supports g

$$\text{Here: Scope}(g) = \{D, G, S\}$$

Not supported!

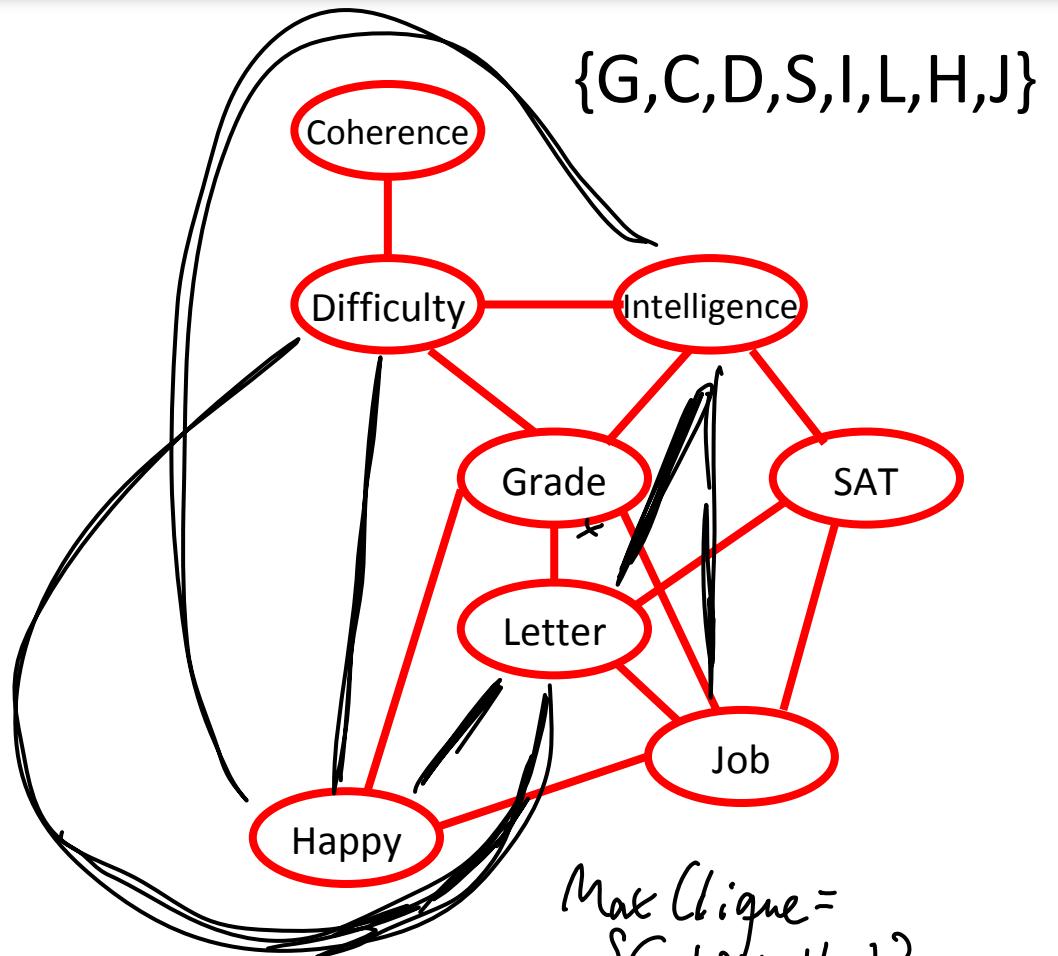
Need to fill in edges by connecting all neighbors of I into clique

Impact of elimination order



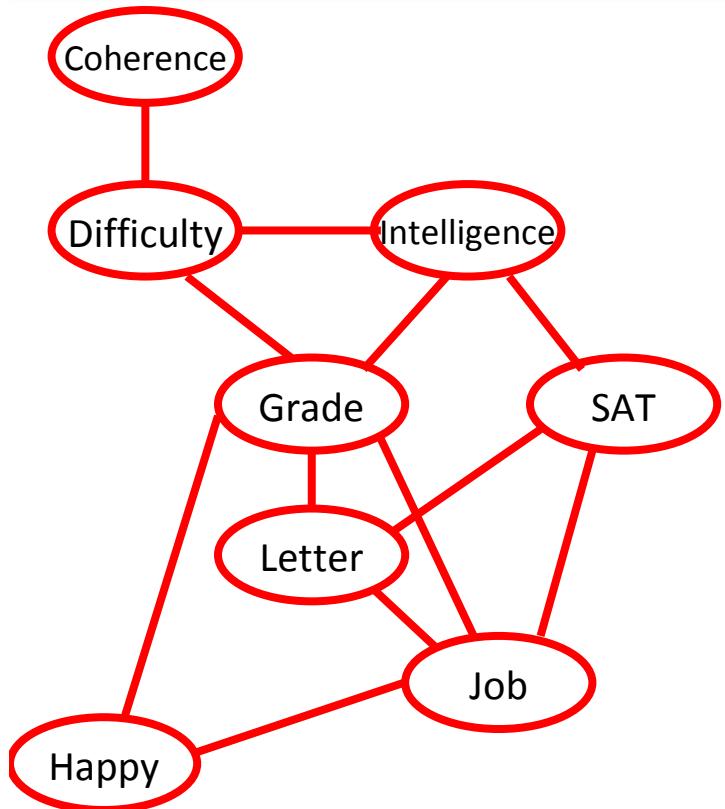
Max Clique = $\{I, G, L, J\}$
 \Rightarrow Tree width 3

- Different elim. order induce different graphs! Tree width 5



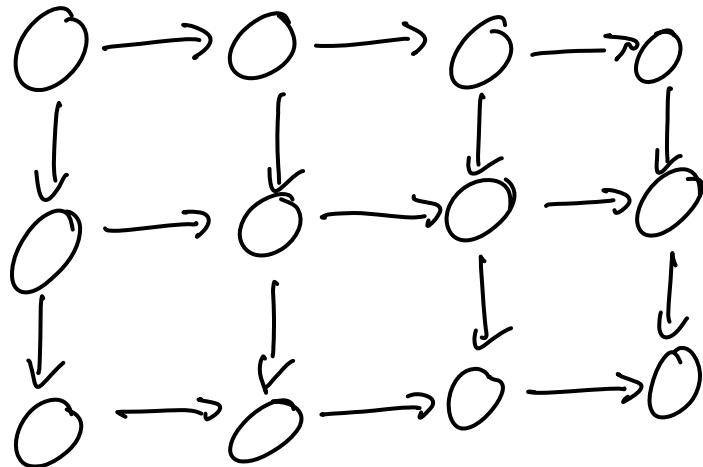
Max Clique =
 $\{G, I, D, L, H, J\}$

Induced graph and VE complexity

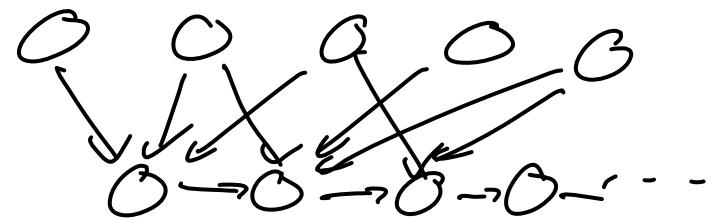


- **Theorem:**
 - All factors arising in VE are defined over cliques (fully connected subgraphs) of the induced graph
 - All maximal cliques of induced graph arise as factors in VE
- Treewidth for ordering = Size of largest clique of induced graph -1
- Treewidth of a graph = minimal treewidth under optimal ordering
- VE exponential in treewidth!

Compact representation → small treewidth?



$n \times n$ grid
treewidth n
 $\frac{\# \text{ parents}}{\text{node}} = 2$



compact
 $\# \text{ parents} \leq 4$

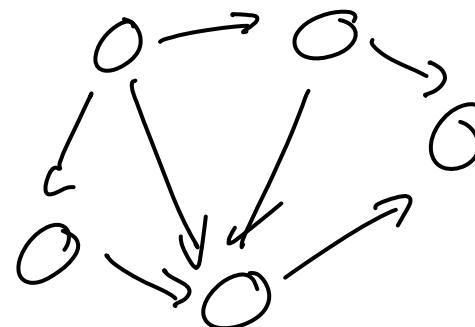
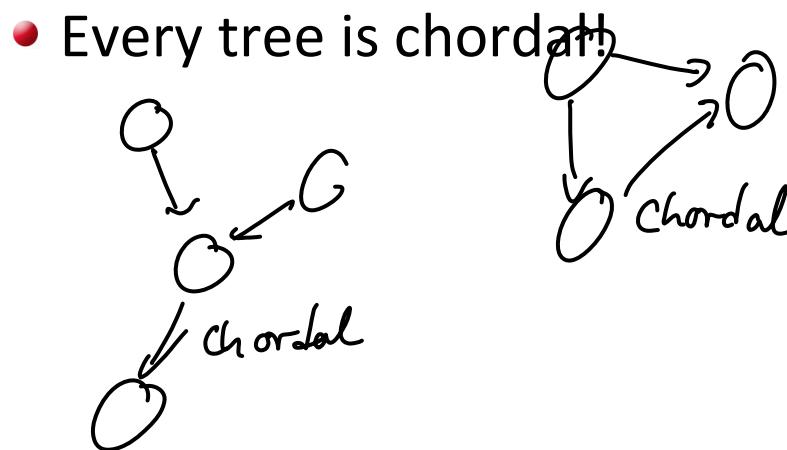
treewidth ??

Finding the optimal elimination order

- **Theorem:** Deciding whether there exists an elimination order with induced width at most K is NP-hard
 - Proof by reduction from MAX-CLIQUE
- In fact, can find elimination order in time exponential in treewidth
- Finding optimal ordering as hard as inference...
- For which graphs can we find optimal elimination order?

Finding optimal elimination order

- For trees can find optimal ordering (^{poly} saw before)
- A graph is called chordal if every cycle of length ≥ 4 has a chord (an edge between some pair of non-consecutive nodes)

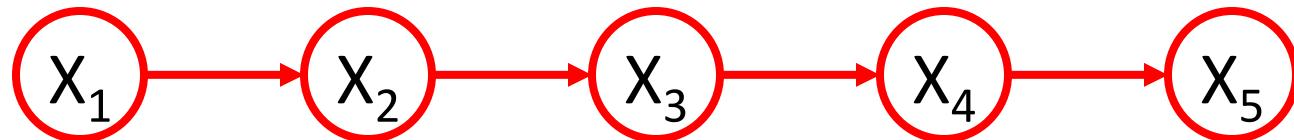


- Can find optimal elimination ordering for chordal graphs

Summary so far

- Variable elimination complexity exponential in induced width for elimination ordering
- Finding optimal ordering is NP-hard
- Many good heuristics
 - Exact for trees, chordal graphs
- Ultimately, inference is NP-hard
- Only difference between cond. prob. queries and MPE is \sum vs. max
- Variable elimination building block for many exact and approximate inference techniques

Answering multiple queries



$$P(x_1, x_5) = P(x_1) \sum_{x_2} P(x_2 | x_1) \sum_{x_3} P(x_3 | x_2) \sum_{x_4} P(x_4 | x_3) P(x_5 | x_4)$$

Can compute in $O(n)$ operations (+, *)

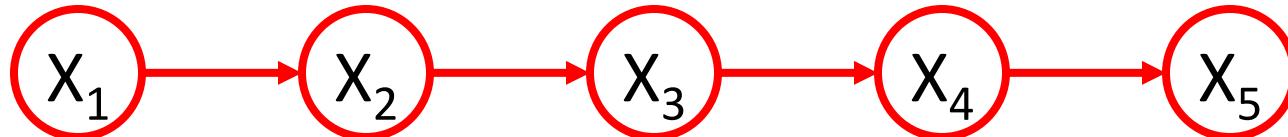
$P(x_2, x_5)$ costs $O(n)$ ops.

$\underbrace{P(x_i, x_5)}_{\text{---}} \text{--- } O(n) \text{ ops}$

If compute $P(x_i | x_j) \forall i \Rightarrow \Theta(n^2)$

Can we reduce this to $O(n)$??

Reusing computation



$$P(x_1, x_5) = P(x_1) \sum_{x_2} P(x_2 | x_1) \sum_{x_3} P(x_3 | x_2) \underbrace{\sum_{x_4} P(x_4 | x_3) P(x_5 | x_4)}_{g_4(x_3, x_5)}$$
$$\underbrace{g_3(x_2, x_5)}_{g_3(x_2, x_5)}$$

$$P(x_2, x_5) = \sum_{x_1} P(x_1) P(x_2 | x_1) \sum_{x_3} P(x_3 | x_2) \underbrace{\sum_{x_4} P(x_4 | x_3) P(x_5 | x_4)}_{g_4(x_3, x_5)}$$

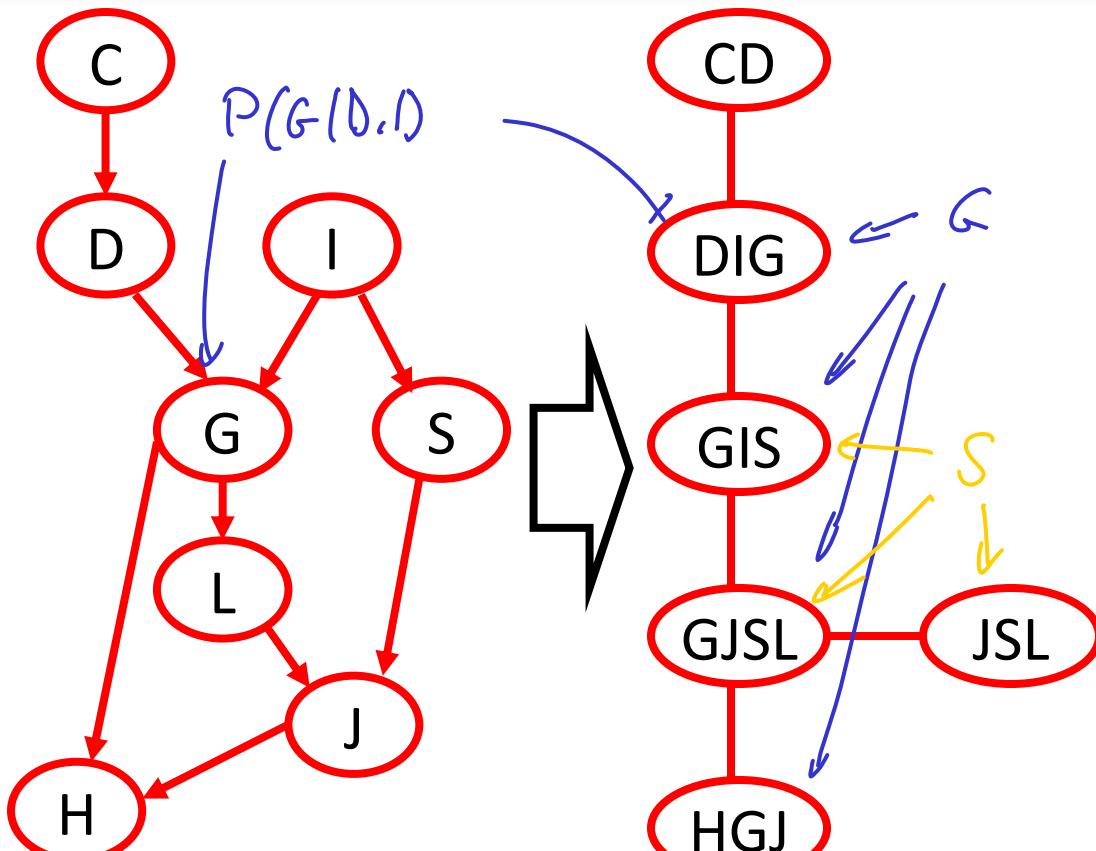
already computed!

Want to "cache" our computations!

Next

- Will learn about algorithm for efficiently computing all marginals $P(X_i \mid E=e)$ given fixed evidence $E=e$
- Need appropriate data structure for storing the computation
→ Junction trees

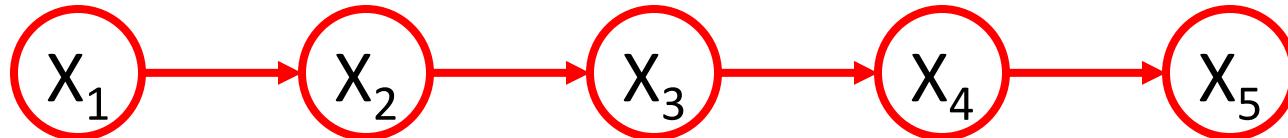
Junction trees



A junction tree for a collection of factors:

- A tree, where each node is a cluster of variables
- Every factor contained in some cluster C_i
- **Running intersection property:** If $X \in C_i$ and $X \in C_j$, and C_m is on the path between C_i and C_j , then $X \in C_m$

VE constructs a junction tree

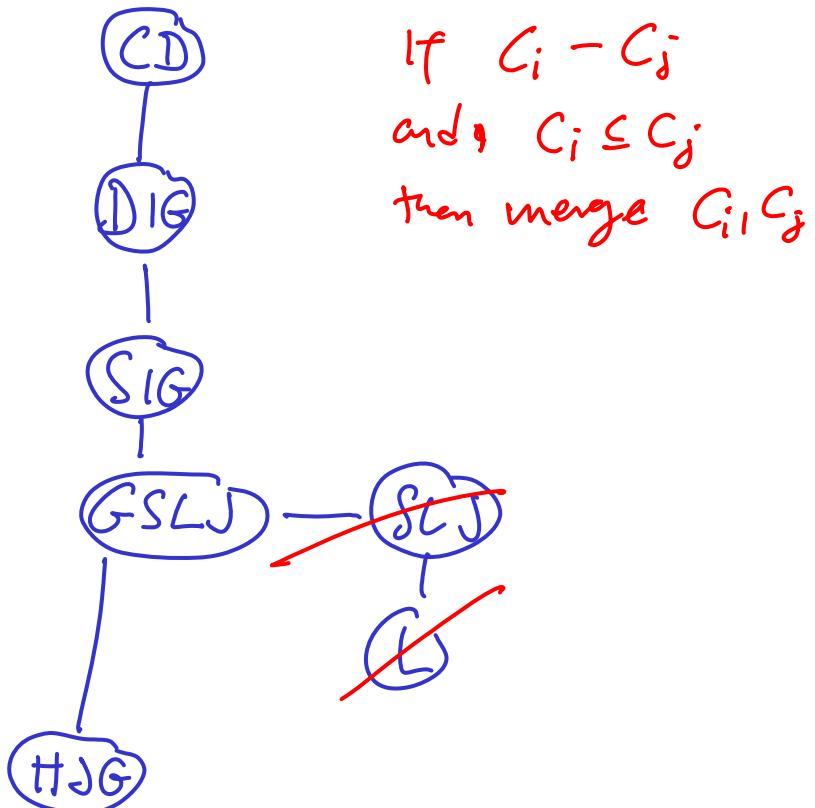
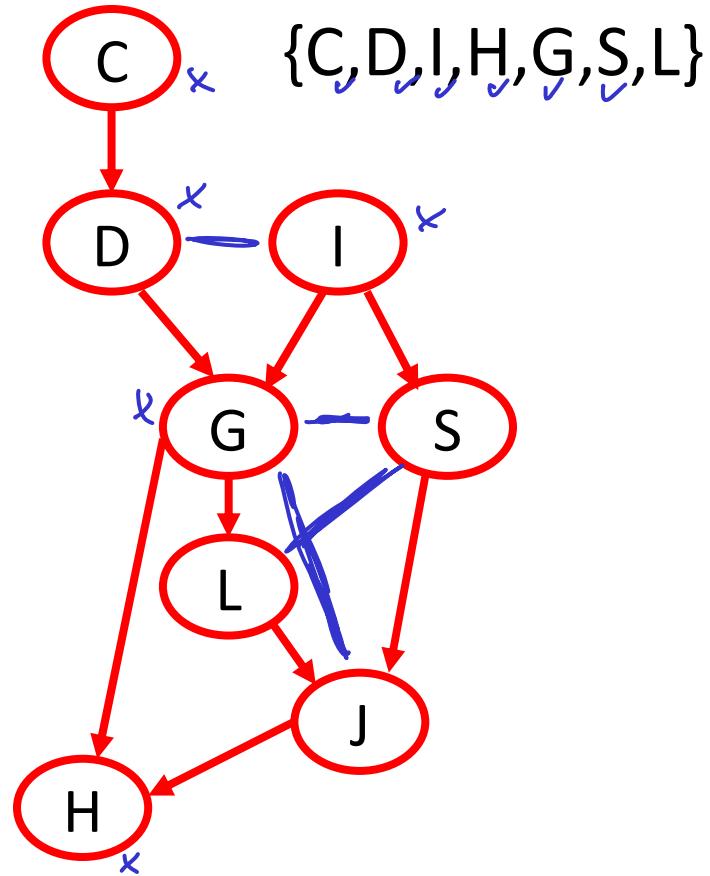


$P(x_1, x_5)$
Ordering
 x_1, x_3, x_2

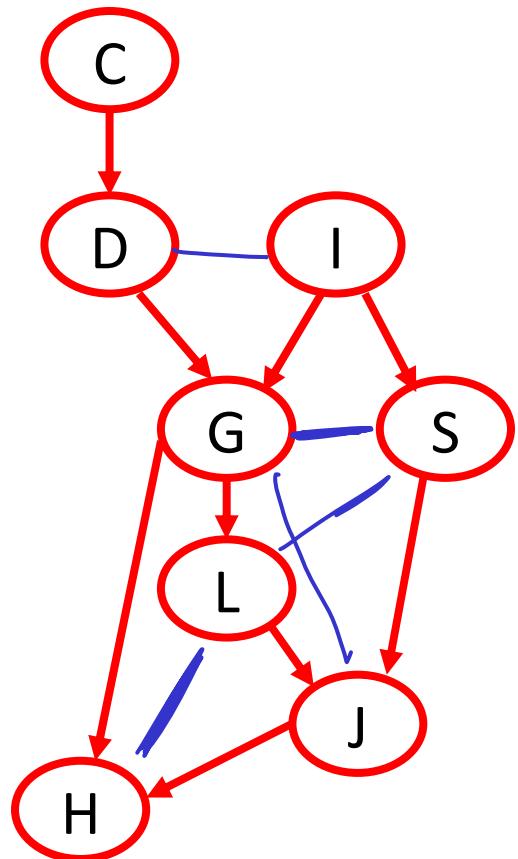
$$P(x_1, x_5) = \underbrace{f_1(x_1, x_2)}_{\text{Ordering } x_1, x_3, x_2} \cdot \underbrace{f_2(x_2, x_3)}_{x_1, x_3, x_2} \cdot \underbrace{f_3(x_3, x_4)}_{x_1, x_3, x_2} \cdot \underbrace{f_4(x_4, x_5)}_{x_1, x_3, x_2}$$

- One clique C_i for each factor f_i created in VE
- C_i connected to C_j if f_i used to generate f_j
- Every factor used only once → Tree
- **Theorem:** resulting tree satisfies RIP

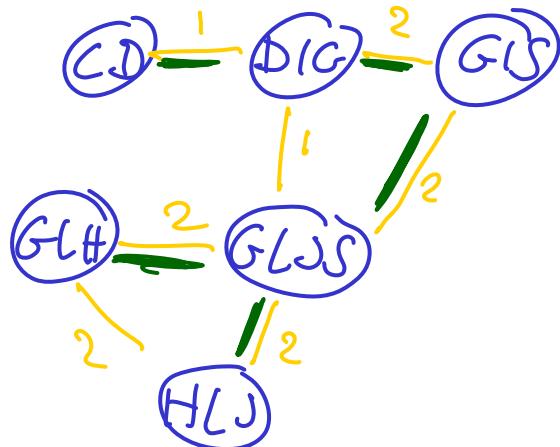
Example: JT from VE



Constructing JT from chordal graph

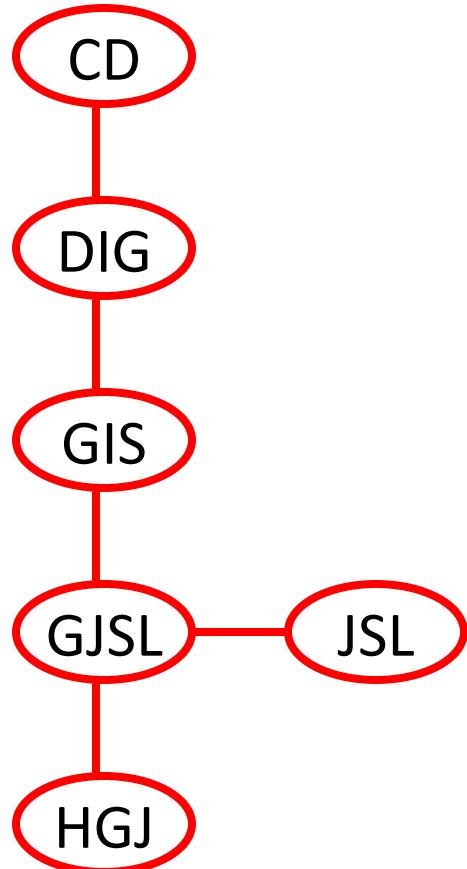


1. Moralize
2. Triangulate (make chordal)
3. Identify max. cliques
4. Connect cliques into undirected graph
 $w(C_i, C_j) = |C_i \cap C_j|$
5. Find Max ST



⇒ Results in valid junction tree

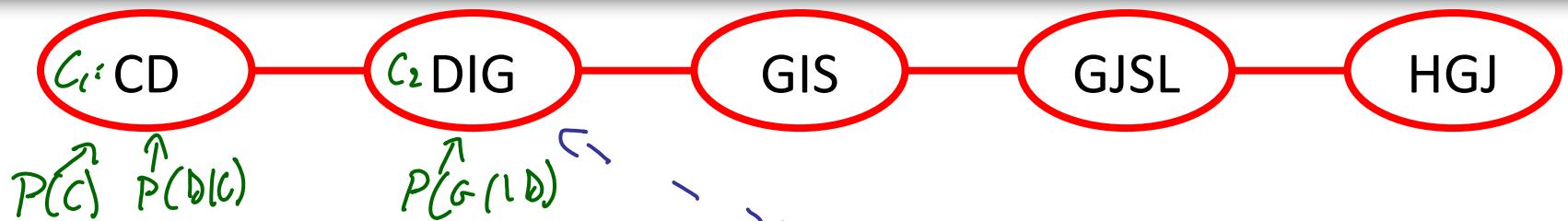
Junction trees and independence



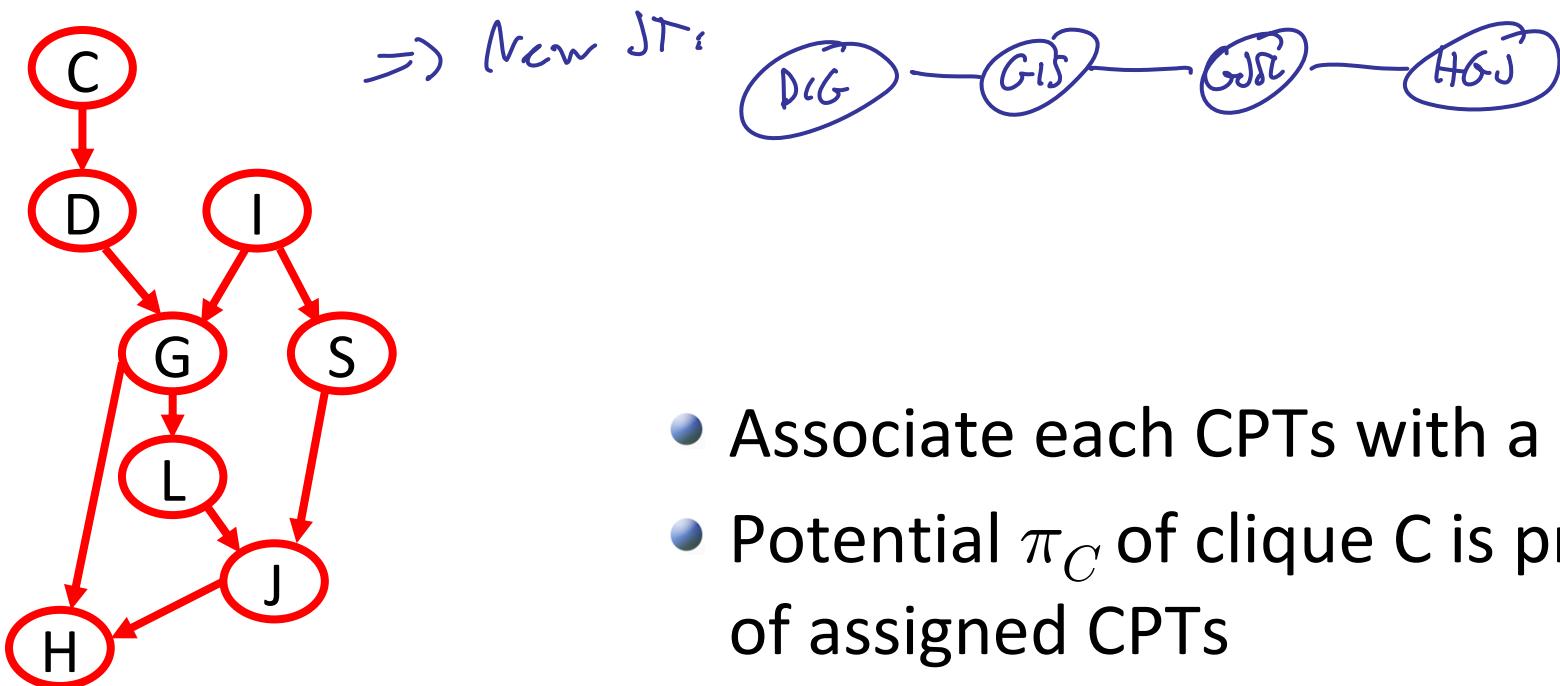
Theorem:

- Suppose
 - T is a junction tree for graph G and factors F
 - Consider edge $C_i - C_j$ with separator $S_{i,j} = \underline{C_i} \cap \underline{C_j}$
 - Variables X and Y on opposite sites of separator
- Then $X \perp Y \mid S_{i,j}$
- Furthermore, $I(T) \subseteq I(G)$

Variable elimination in junction trees



Elim C: Compute new factor $g(D) = \sum_C \Pi_{C_i}(C, D)$



VE as message passing

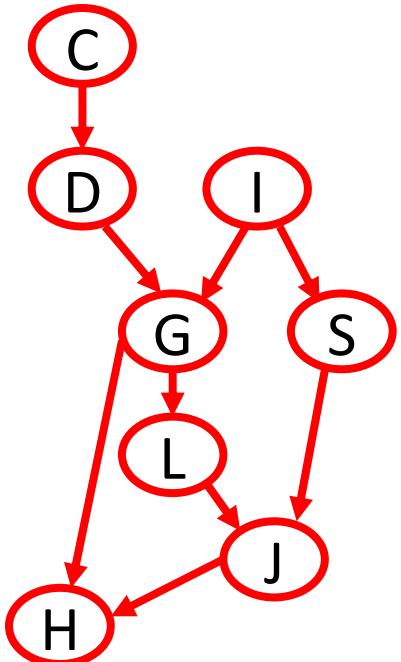
Message: $\delta_{1 \rightarrow 2}(D)$
 $= \sum_C \pi(C, D)$

$\delta_{2 \rightarrow 3}(IG)$
 $= \sum_D \pi(D, IG) \cdot \delta_{1 \rightarrow 2}(D)$

$\delta_{4 \rightarrow 3}(JL)$
 $= \sum_{JL} \pi(GJS) \delta_{5 \rightarrow 4}(GJ)$

$\delta_{5 \rightarrow 4}(GJ)$
 $= \sum_H \pi(H, GJ)$

$P(G) = \sum_S \pi(G, S) \delta_{2 \rightarrow 3}(IG) \delta_{4 \rightarrow 3}(GS)$



VE for computing X_i

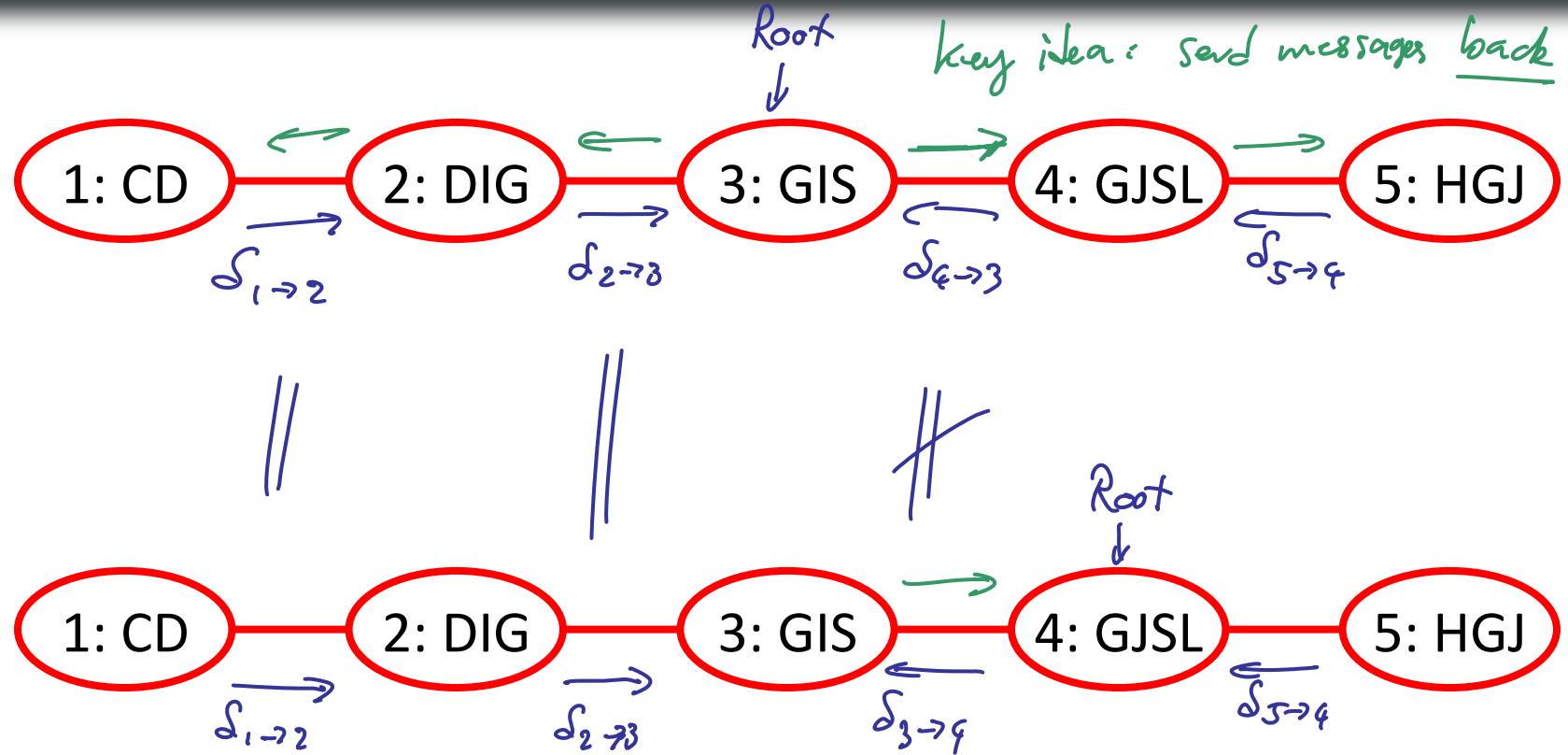
- Pick root (any clique containing X_i)
- Don't eliminate, only send messages recursively from leaves to root
 - Multiply incoming messages with clique potential
 - Marginalize variables not in separator
- Root “ready” when received all messages

Correctness of message passing



- **Theorem:** When root ready (received all messages), all variables in root have correct potentials
 - Follows from correctness of VE
- So far, no gain in efficiency ☹

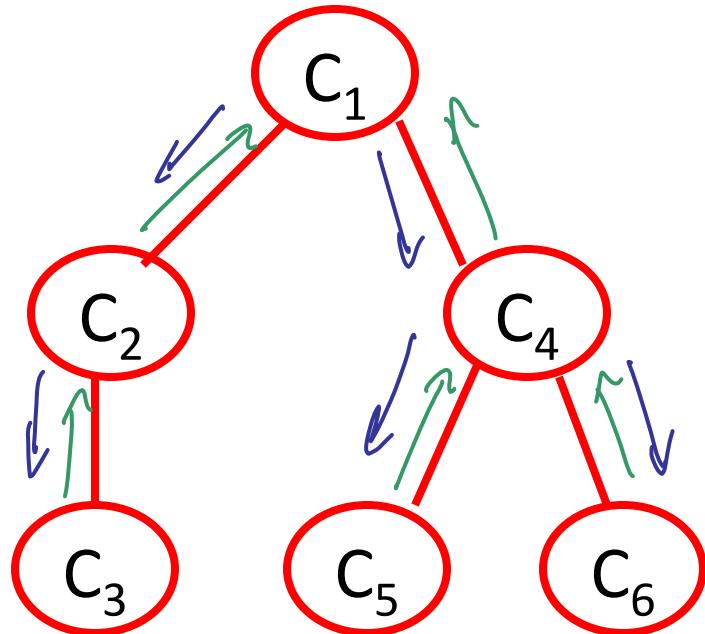
Does the choice of root affect messages?



Instead of $O(n^2) \rightarrow O(n)$

1 message per edge per direction

Shenoy-Shafer algorithm



- Clique i ready if received messages from all neighbors *but 1*
 - Leaves always ready
- While there exists a message $\delta_{i \rightarrow j}$ ready to transmit send message

Complexity? $O(m 2^{\text{treewidth}})$
"Only" 1 msg per edge per direction

Theorem: At convergence,
every clique has correct beliefs

Inference using VE

- Want to incorporate evidence $E=e$
- Multiply all cliques containing evidence variables with indicator potential $\mathbf{1}_e$

$$\textcircled{AB} \quad I_{A=T}(a,b) = \begin{cases} 1 & \text{if } a=T \\ 0 & \text{if } a=F \end{cases}$$

- Perform variable elimination

Summary so far

- Junction trees represent distribution
 - Constructed using elimination order
 - Make complexity of inference explicitly visible
- Can implement variable elimination on junction trees to compute correct beliefs on all nodes
- Now:
 - **Belief propagation** – an important alternative to VE on junction trees.
 - Will later generalize to approximate inference!
 - Key difference: Messages obtained by division rather than multiplication

Probabilistic Graphical Models

Lecture 9 – Undirected Models

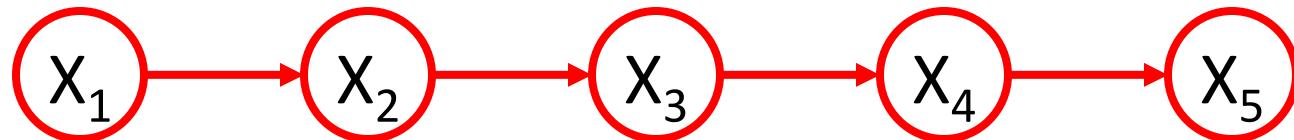
CS/CNS/EE 155
Andreas Krause

Announcements

- Homework 2 due next Wednesday (Nov 4) in class
 - Start early!!!
- Project milestones due Monday (Nov 9)
 - 4 pages of writeup, NIPS format
 - <http://nips.cc/PaperInformation/StyleFiles>

Best project award!!

Answering multiple queries



$$P(x_1, x_5) = P(x_1) \sum_{x_2} P(x_2 | x_1) \sum_{x_3} P(x_3 | x_2) \sum_{x_4} P(x_4 | x_3) P(x_5 | x_4)$$

Can compute in $O(n)$ operations (+, *)

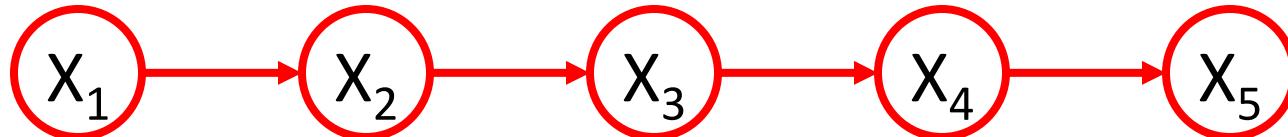
$P(x_2, x_5)$ costs $O(n)$ ops.

$\underbrace{P(x_i, x_5)}_{\text{---}} \text{---} O(n) \text{ ops}$

If compute $P(x_i | x_j) \forall i \Rightarrow \Theta(n^2)$

Can we reduce this to $O(n)$??

Reusing computation



$$P(x_1, x_5) = P(x_1) \sum_{x_2} P(x_2 | x_1) \sum_{x_3} P(x_3 | x_2) \underbrace{\sum_{x_4} P(x_4 | x_3) P(x_5 | x_4)}_{g_4(x_3, x_5)}$$
$$\underbrace{g_3(x_2, x_5)}_{g_3(x_2, x_5)}$$

$$P(x_2, x_5) = \sum_{x_1} P(x_1) P(x_2 | x_1) \sum_{x_3} P(x_3 | x_2) \underbrace{\sum_{x_4} P(x_4 | x_3) P(x_5 | x_4)}_{g_4(x_3, x_5)}$$

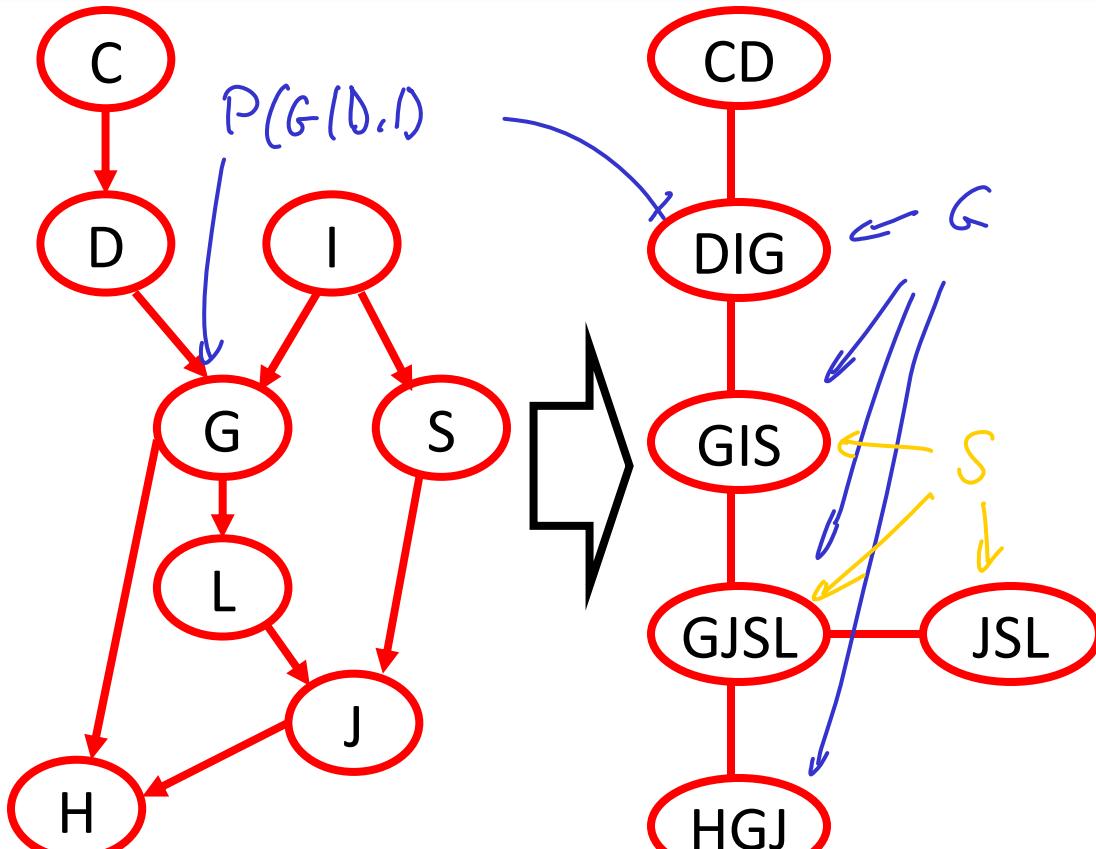
already computed!

Want to "cache" our computations!

Next

- Will learn about algorithm for efficiently computing all marginals $P(X_i \mid E=e)$ given fixed evidence $E=e$
- Need appropriate data structure for storing the computation
→ Junction trees

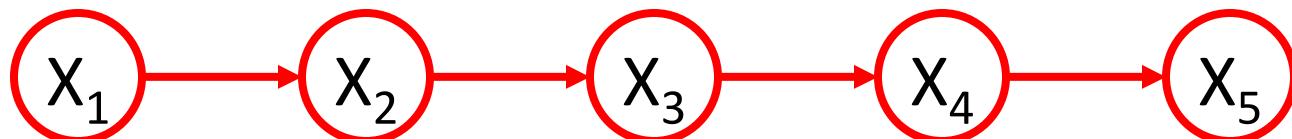
Junction trees



A junction tree for a collection of factors:

- A tree, where each node is a cluster of variables
- Every factor contained in some cluster C_i
- **Running intersection property:** If $X \in C_i$ and $X \in C_j$, and C_m is on the path between C_i and C_j , then $X \in C_m$

VE constructs a junction tree

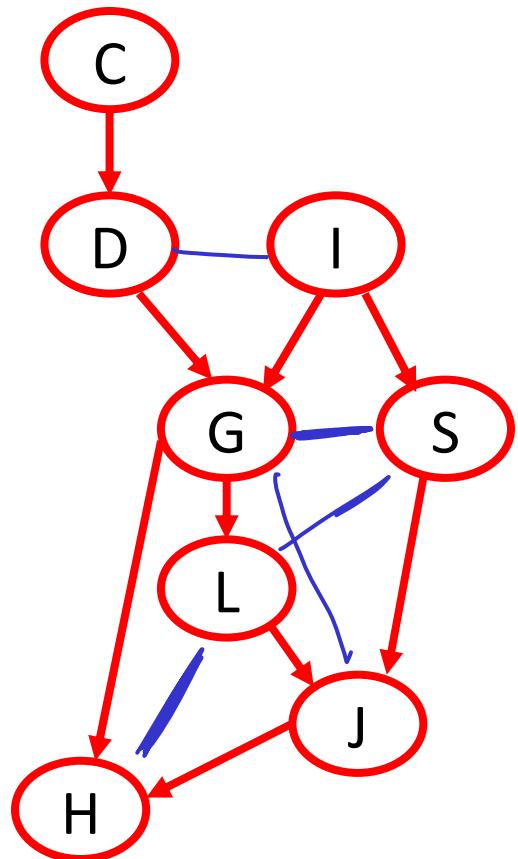


$P(x_1, x_5)$
Ordering
 x_1, x_3, x_2

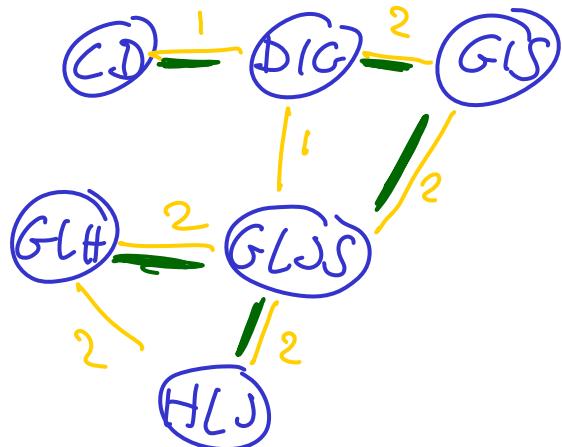
x_1, x_2 - x_2, x_3 - x_3, x_5 - x_4, x_5

- One clique C_i for each factor f_i created in VE
- C_i connected to C_j if f_i used to generate f_j
- Every factor used only once → Tree
- **Theorem:** resulting tree satisfies RIP

Constructing JT from chordal graph

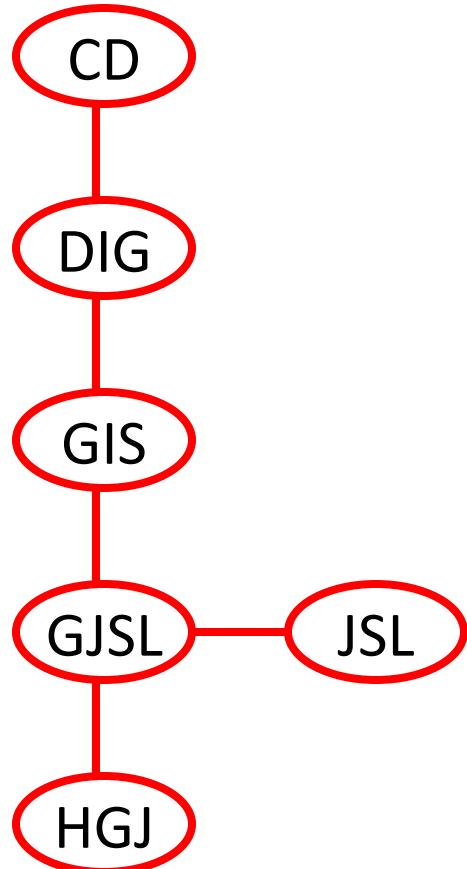


1. Moralize
2. Triangulate (make chordal)
3. Identify max. cliques
4. Connect cliques into undirected graph
 $w(C_i, C_j) = |C_i \cap C_j|$
5. Find Max ST



⇒ Results in valid junction tree

Junction trees and independence



Theorem:

- Suppose
 - T is a junction tree for graph G and factors F
 - Consider edge $C_i - C_j$ with separator $S_{i,j} = \underline{C_i} \cap \underline{C_j}$
 - Variables X and Y on opposite sites of separator
- Then $X \perp Y \mid S_{i,j}$
- Furthermore, $I(T) \subseteq I(G)$

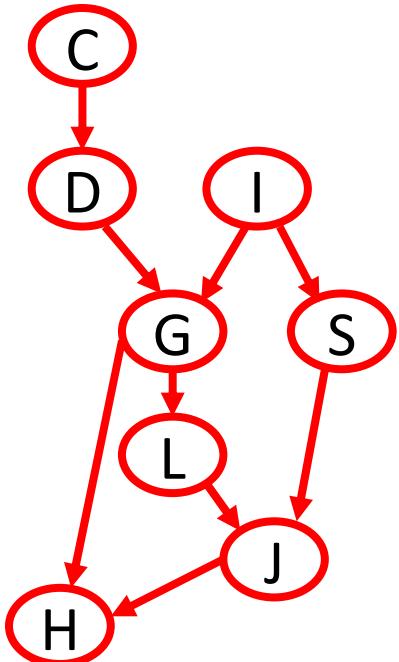
VE as message passing



$$\text{"Message": } \delta_{1 \rightarrow 2}(D) = \sum_C \pi(C, D)$$

$$\begin{aligned} \delta_{2 \rightarrow 3}(I, G) &= \sum_D \pi(D, I, G) \cdot \delta_{1 \rightarrow 2}(D) \\ \delta_{4 \rightarrow 3}(J, L) &= \sum_{JL} \pi(G, J, S) \delta_{3 \rightarrow 4}(JL) \\ \delta_{5 \rightarrow 4}(G, J) &= \sum_H \pi(H, G, J) \end{aligned}$$

$$P(G) = \sum_S \pi(G, S) \delta_{2 \rightarrow 3}(IG) \delta_{4 \rightarrow 3}(GS)$$



VE for computing X_i

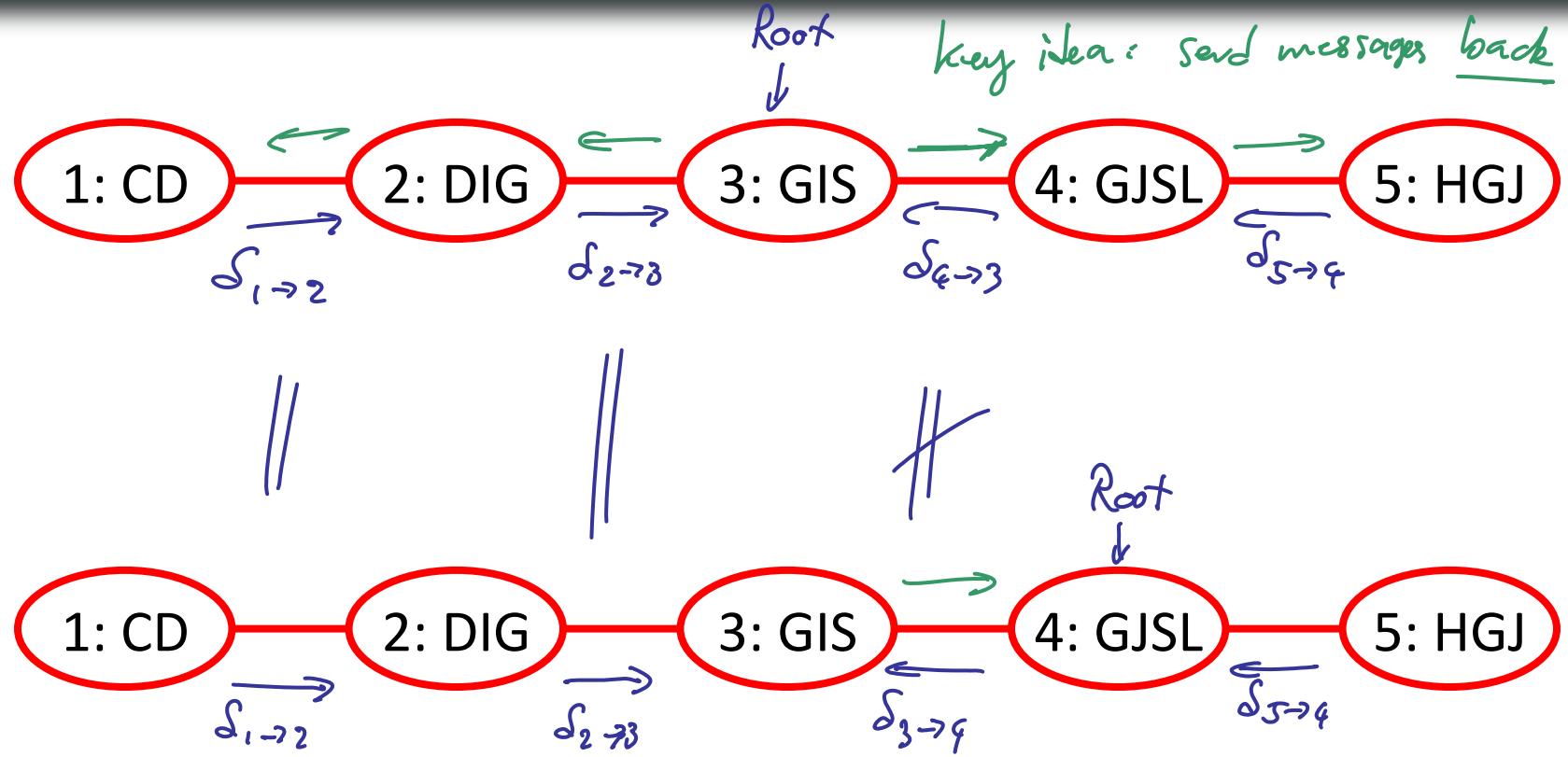
- Pick root (any clique containing X_i)
- Don't eliminate, only send messages recursively from leaves to root
 - Multiply incoming messages with clique potential
 - Marginalize variables not in separator
- Root "ready" when received all messages

Correctness of message passing



- **Theorem:** When root ready (received all messages), all variables in root have correct potentials
 - Follows from correctness of VE
- So far, no gain in efficiency ☹

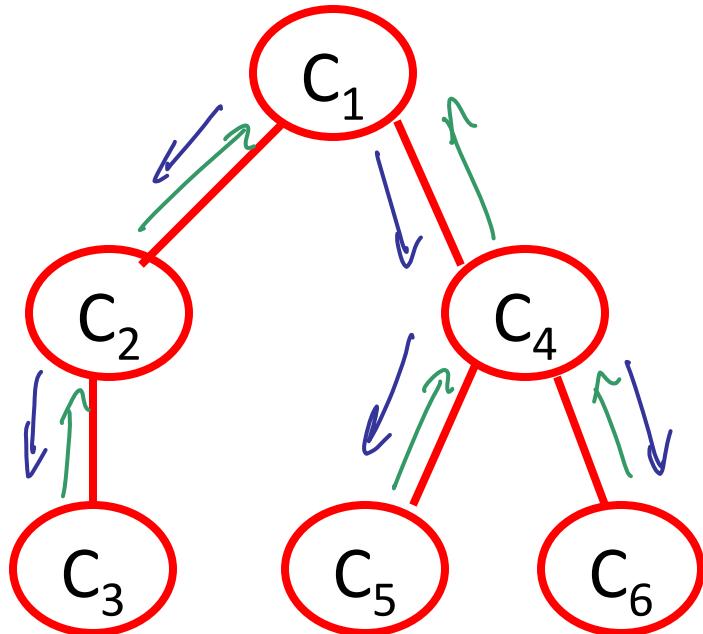
Does the choice of root affect messages?



Instead of $O(n^2) \rightarrow O(n)$

1 message per edge per direction

Shenoy-Shafer algorithm



- Clique i ready if received messages from all neighbors *but 1*
 - Leaves always ready
- While there exists a message $\delta_{i \rightarrow j}$ ready to transmit send message

Complexity? $O(m 2^{\text{treewidth}})$
"Only" 1 msg per edge per direction

Theorem: At convergence,
every clique has correct beliefs

Inference using VE

- Want to incorporate evidence $E=e$
- Multiply all cliques containing evidence variables with indicator potential $\mathbf{1}_e$

$$\textcircled{AB} \quad I_{A=T}(a,b) = \begin{cases} 1 & \text{if } a=T \\ 0 & \text{if } a=F \end{cases}$$

- Perform variable elimination

Summary so far

- Junction trees represent distribution
 - Constructed using elimination order
 - Make complexity of inference explicitly visible
- Can implement variable elimination on junction trees to compute correct beliefs on all nodes
- Now:
 - **Belief propagation** – an important alternative to VE on junction trees.
 - Will later generalize to approximate inference!
 - Key difference: Messages obtained by division rather than multiplication

Message passing by factor division

- Variable elimination:

- Message → Belief

$$\delta_{2 \rightarrow 3}(1G) = \sum_D \Pi_2^{(0)}(D|G) \delta_{1 \rightarrow 2}(D)$$

Belief at 3: $\Pi_3^{(0)}(G|S) \cdot \delta_{2 \rightarrow 3}(1G)$

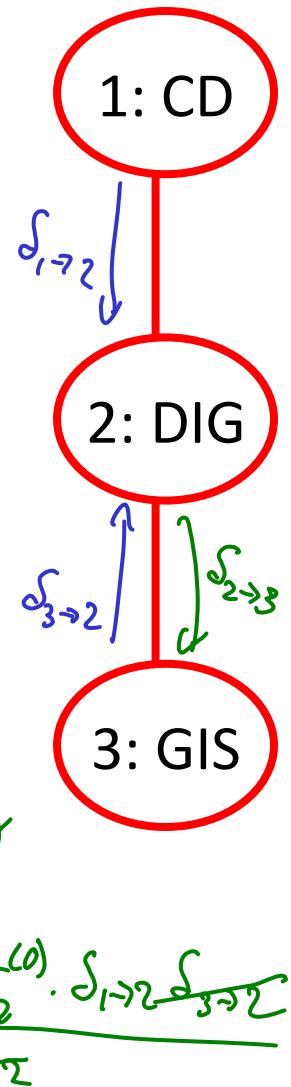
- Factor division:

- Belief → Message

Belief at 2: $\Pi_2^{(0)}(D|G) \cdot \delta_{1 \rightarrow 2}(0) \cdot \delta_{3 \rightarrow 2}(1G)$

Belief about Sep. 1G: $\delta_{2 \rightarrow 3}^{(1)}(1G) = \sum_D \Pi_2^{(1)}(D|G) \quad \leftarrow \text{Send as msg}$

Belief at 3: $\overline{\Pi}_3^{(t+1)}(G|S) = \overline{\Pi}_3^{(0)}(G|S) \cdot \frac{\delta_{2 \rightarrow 3}^{(1)}(1G)}{\delta_{3 \rightarrow 2}(1G)} = \frac{\sum_D \Pi_3^{(0)}(G|S) \cdot \Pi_2^{(0)} \cdot \delta_{1 \rightarrow 2}}{\delta_{3 \rightarrow 2}}$



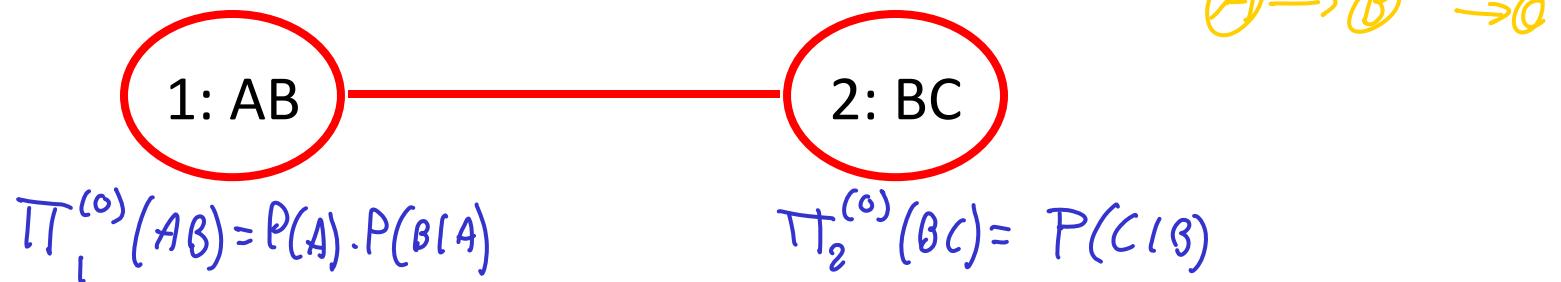
Factor division

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

$$f = \frac{f_1}{f_2} \quad f(A, B, C) = \frac{f_1(A, B, C)}{f_2(B, C)}$$

with the convention that $\frac{0}{0} = 0$

Clique and separator potentials



Belief
about
sep. B

$$\sigma_{1 \rightarrow 2}^{(0)}(B) = \sum_A \Pi_1^{(0)}(AB) \quad \not\equiv \quad \sigma_{2 \rightarrow 1}^{(0)}(B) = \sum_C \Pi_2^{(0)}(BC)$$

At convergence:

$$\Pi_i^{(\pi)}(AB) = P(AB)$$

$$\Pi_2^{(\pi)} = P(BC)$$

$$\sigma_{1 \rightarrow 2}^{(\pi)}(B) = \sum_A \Pi_1^{(\pi)}(AB) = P(B) = \sigma_{2 \rightarrow 1}^{(\pi)} = \mu_{12}(B)$$

Call such a JT with potentials $\Pi_1^{(\pi)} \dots \Pi_n^{(\pi)}$ that agree on separators "calibrated"

Lauritzen-Spiegelhalter algorithm (a.k.a. Belief Propagation)

- Initialize separator potentials μ_{ij}

- One per edge, initialized to 1

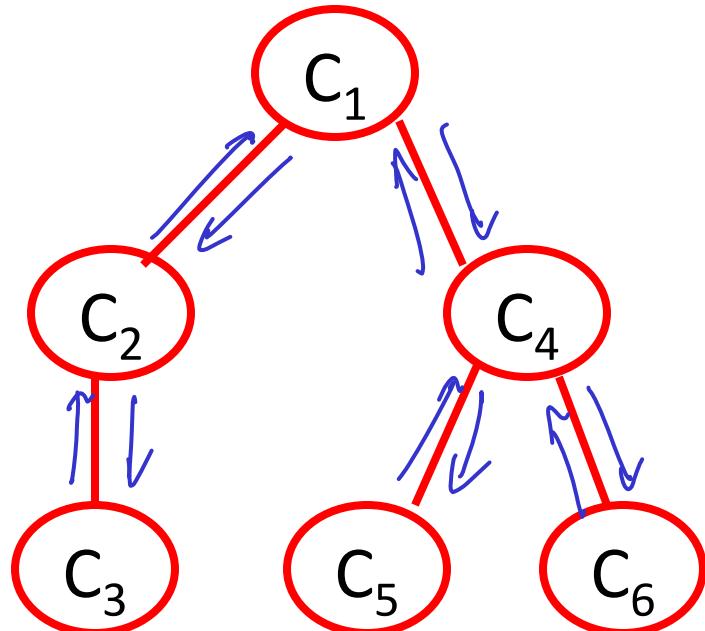
stores last msg across edge $i-j$

- Messages $i \rightarrow j$

$$\sigma_{i \rightarrow j}^t(C_j \cap C_i) = \sum_{C_i \setminus C_j} \Pi_i^{(t)}$$

$$\Pi_j^{(t+1)} = \Pi_j^{(t)} \cdot \frac{\sigma_{i \rightarrow j}^t}{\mu_{ij}}$$

$$\mu_{ij} = \sigma_{i \rightarrow j}^t$$



Correctness of Belief propagation

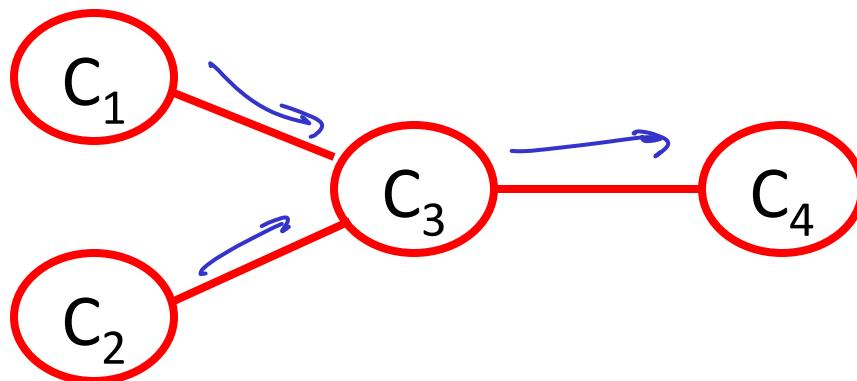
- Complexity linear in #cliques
- **Theorem:**
At convergence, every clique has correct beliefs
(when using correct message order, i.e., leaves to root and back)

→ **Corollary:**

Junction tree is calibrated (cliques agree on separator)

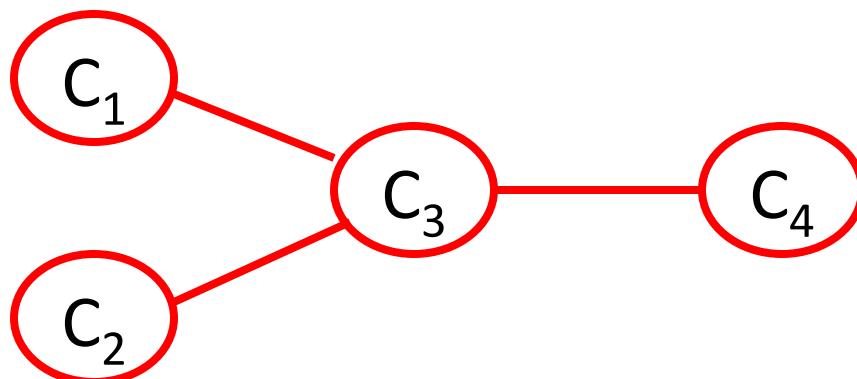
Comparison of VE and BP messages

- Variable elimination



$$\delta_{3 \rightarrow 4} = \sum_{C_3 \setminus C_4} \prod_3^{(o)} \delta_{1 \rightarrow 3} \delta_{2 \rightarrow 3}$$

- Belief propagation



$$G_{3 \rightarrow 4}^{(t)} = \sum_{C_3 \setminus C_4} \prod_3^{(t)}$$

$$\begin{aligned} \prod_4^{(t+1)} &= \prod_4^{(t)} \cdot \frac{\delta_{3 \rightarrow 4}}{m_{34}} \\ m_{34}^{(t+1)} &= G_{3 \rightarrow 4}^{(t)} \end{aligned}$$

Understanding BP

- Junction tree potential

$$\Pi_T(x) = \frac{\prod_i \Pi_{C_i}(C_i)}{\prod_{ij} M_{ij}(C_i \cap C_j)}$$

- Junction tree invariant

$$\Pi_T(x) = P(x)$$

- Theorem: BP maintains Junction tree invariant
→ BP reparametrizes clique and separator potentials

Advantages and disadvantages of junction tree inference

- Advantages

- Can answer multiple queries (for same evidence) efficiently
- Can perform incremental updates

- Disadvantages

- No factors are “deleted”
- Can’t prune away unnecessary variables
- Slower for a single query

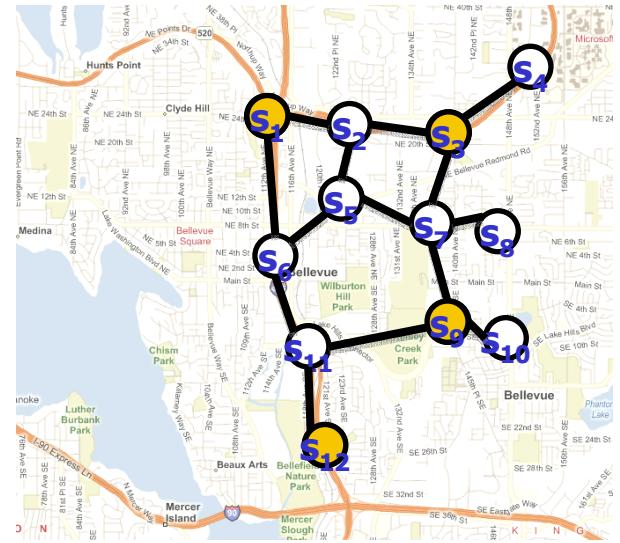
Summary so far

- Bayesian Networks
 - Representation
 - Learning (MLE / Bayesian) with fully observed data
 - Exact Inference
- Next
 - Undirected models
 - Approximate inference
 - Hidden variables

Representing the world using BNs



represent



True distribution P'
with cond. ind. $I(P')$

Bayes net (G, P)
with $I(P)$

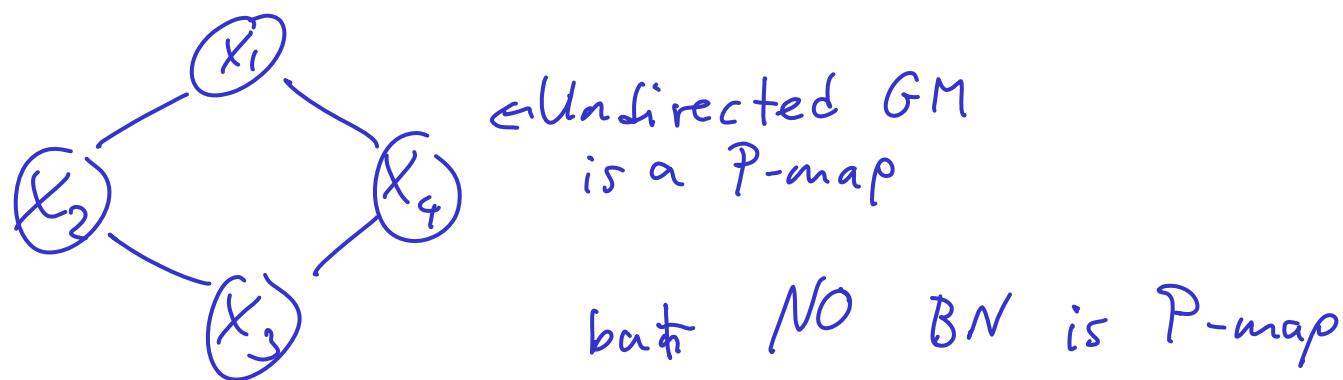
- Want to make sure that $I(P) \subseteq I(P')$
- Ideally: $\underline{I(P) = I(P')}$
- Want BN that **exactly** captures independencies in P' !

Perfect maps

- Minimal I-maps are easy to find, but can contain many unnecessary dependencies.
- A BN structure G is called **P-map** (perfect map) for distribution P if $I(G) = I(P)$
- Does every distribution P have a P-map?

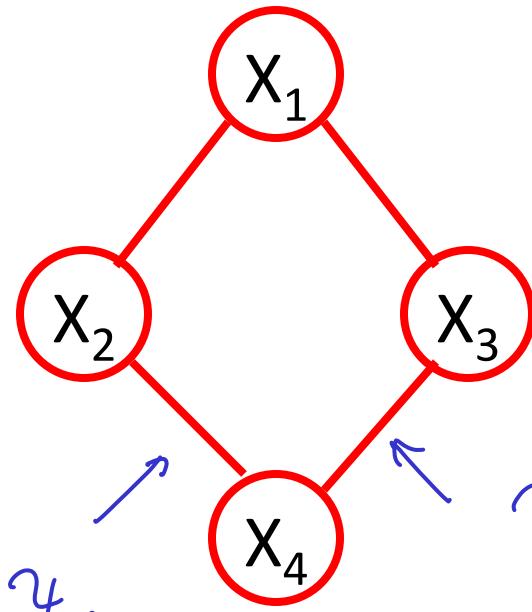
Existence of perfect maps

$$x_1, \dots, x_q \quad x_1 \perp x_3 \mid x_2, x_q$$
$$x_2 \perp x_q \mid x_1, x_3$$



- Will have undirected model as P-map

Undirected Parameterization



Specify factors over cliques in undirected graph

$$\gamma_{34}(x_3, x_4) \geq 0$$

| x_2 | x_4 | $\gamma_{24}(x_2, x_4)$ |
|-------|-------|-------------------------|
| 0 | 0 | 3 |
| 0 | 1 | 17 |
| 1 | 0 | 1 |
| 1 | 1 | 3415 |

$$P(x_1, \dots, x_4) = \frac{1}{2} \gamma_{12}(x_1, x_2) \gamma_{13}(x_1, x_3) \gamma_{24}(x_2, x_4) \gamma_{34}(x_3, x_4)$$

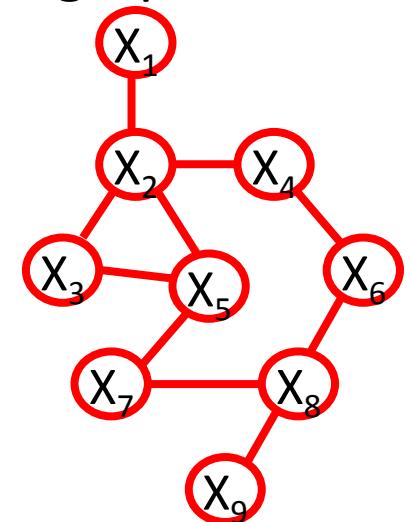
$$Z = \sum_{x_1 \dots x_4} \prod_{ij} \gamma_{ij}(x_i, x_j)$$

Markov Networks

(a.k.a., Markov Random Field, Gibbs Distribution, ...)

- A Markov Network consists of
 - An undirected graph, where each node represents a RV
 - A collection of factors defined over cliques in the graph
- Joint probability

$$P(X) = \frac{1}{Z} \prod_i \psi_i(C_i)$$



- A distribution factorizes over undirected graph G if

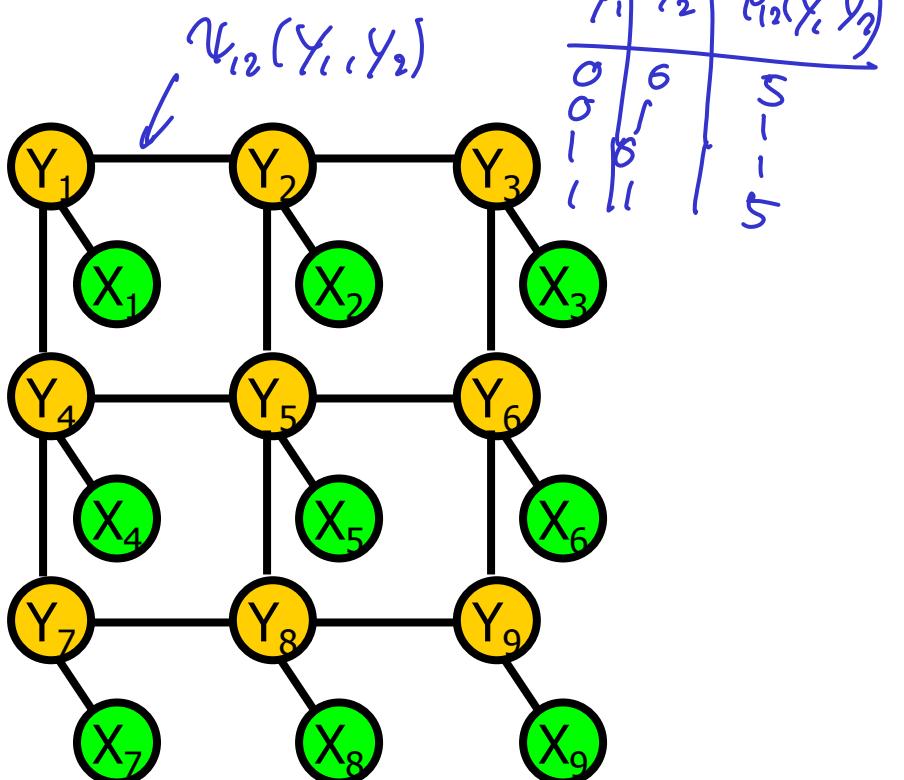
\exists factors $\psi_1 \dots \psi_k$ over cliques of G s.t.

$$P(X) = \frac{1}{Z} \prod_i \psi_i(C_i)$$

Example MN: Image denoising



Markov Network



X_i : noisy pixels

Y_i : “true” pixels

Computing Joint Probabilities

- Computing joint probabilities in BNs

$$P(X_1, \dots, X_m) = \prod_i P(X_i | P_{a_i})$$

$$P(X_1 | X_m)$$

actually comp. $P(X_1, X_m)$

- Computing joint probabilities in Markov Nets

$$P(X_1, \dots, X_m) = \frac{1}{Z} \prod_i \psi_i(C_i)$$

Can do V_F

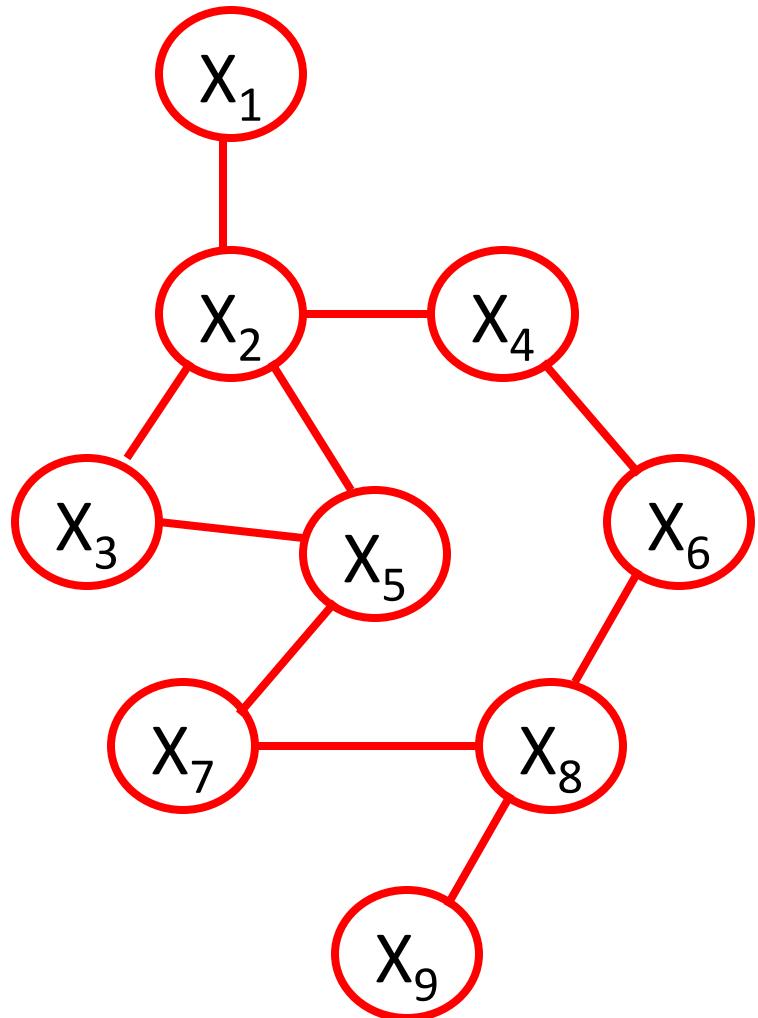
↑
Need to know partition "function" Z

Can compute $\frac{P(X_1, \dots, X_m)}{P(X'_1, \dots, X'_m)} = \frac{\prod_i \psi_i(C_i)}{\prod_i \psi_i(C'_i)}$

Independences in Markov Nets?

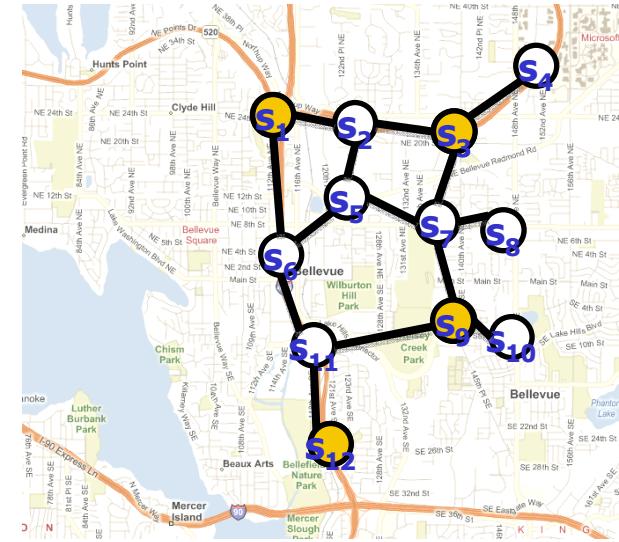
- In Bayes Nets (G, P)
 - Local Markov Assumption: $X \perp \text{NonDesc}(X) \mid \text{Pa}_x$
 - G is I-map for distribution P if Local Markov Assumption holds
 - Factorization Thm: P factorizes over $G \Leftrightarrow G$ is an I-map
 - Global independences: d-separation
 - Completeness and soundness of d-separation
- How about Markov Nets?
 - What's the analog of the Local Markov Assumption?
 - Is there a factorization theorem for Markov Nets?
 - What replaces d-separation?

Local Markov Assumption for MN



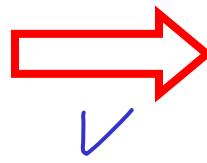
- The **Markov Blanket $\text{MB}(X)$** of a node X is the set of neighbors of X
- Local Markov Assumption:
 $X \perp \text{EverythingElse} \mid \text{MB}(X)$
- $I_{\text{loc}}(G) = \text{set of all local independences}$
- G is called an I -map of distribution P if $I_{\text{loc}}(G) \subseteq I(P)$

Factorization Theorem for Markov Nets “ \Rightarrow ”



True distribution P
can be represented exactly as
a Markov net (G, P)

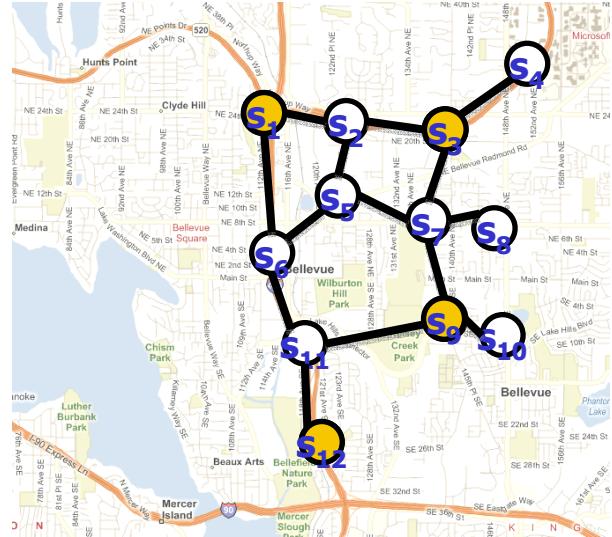
$$P(X_1, \dots, X_n) = \frac{1}{Z} \prod_i \phi_i(C_i)$$



$$I_{loc}(G) \subseteq I(P)$$

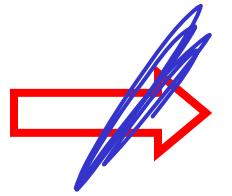
G is an **I-map** of P
(independence map)

Factorization Theorem for Markov Nets “←”



$$I_{loc}(G) \subseteq I(P)$$

G is an **I-map** of **P**
(independence map)



*Not true
in general*



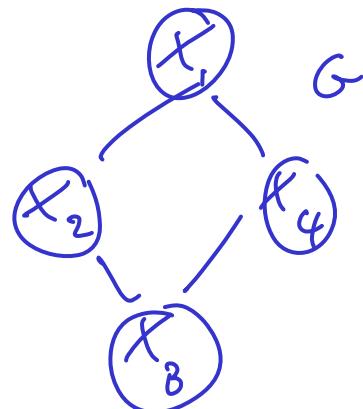
True distribution **P**
can be represented exactly as

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

i.e., **P** can be represented as
a Markov net **(G,P)**

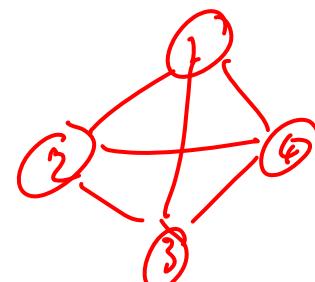
Counterexample

- G an I-map for P does not imply that P factorizes
 - Binary variables X_1, \dots, X_4 .
 - Only positive states
 $(0,0,0,0), (1,0,0,0), (1,1,0,0), (1,1,1,0)$
 $(0,0,0,1), (0,0,1,1), (\underline{0},\underline{1},\underline{1},\underline{1}), (1,\underline{1},\underline{1},\underline{1})$
- each happens with prob $\frac{1}{8}$

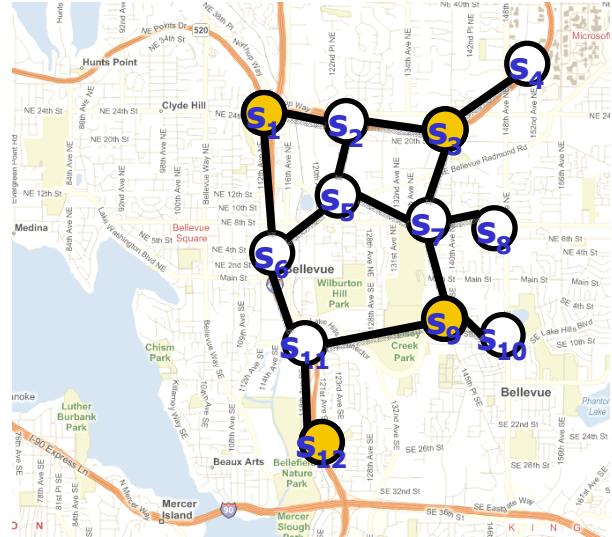


G is I-map for P
 $X_1 \perp X_3 | X_2, X_4$
 Eg.: $X_2 = 1, X_4 = 1$

But to represent P , need fully connected graph

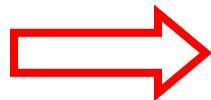


Factorization Theorem for Markov Nets “ \leftarrow ” Hammersley-Clifford Theorem



$$I_{loc}(G) \subseteq I(P)$$

G is an I-map of P
(independence map)
and $P > 0$

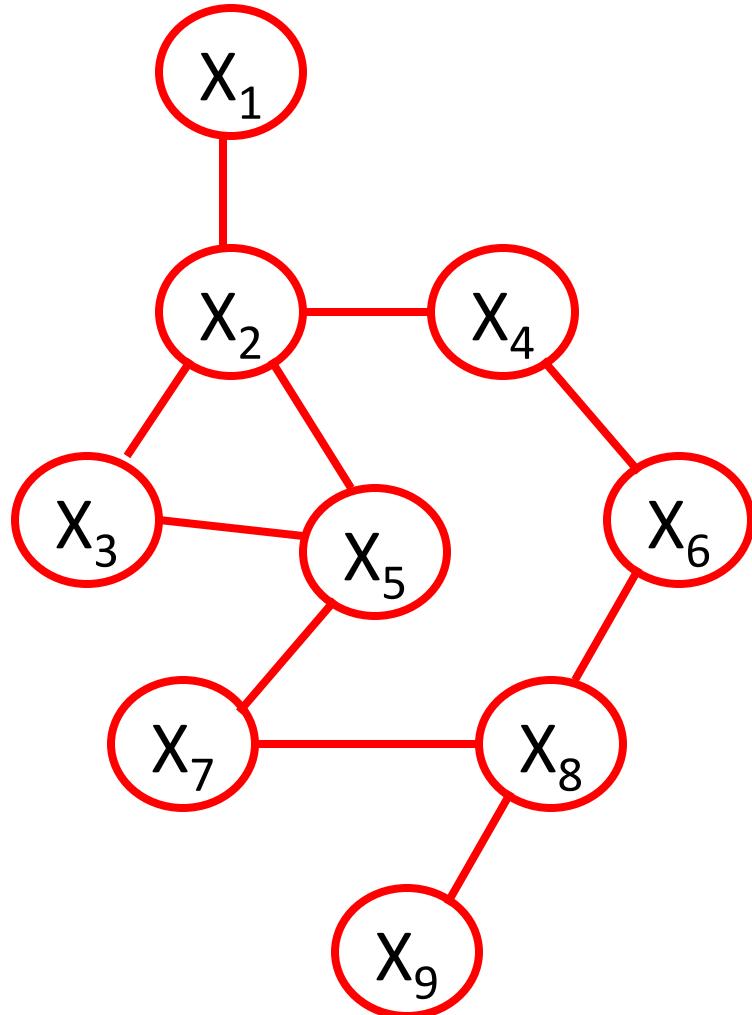


True distribution P
can be represented exactly as

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

i.e., P can be represented as
a Markov net (G, P)

Global independencies



- A trail $X-X_1-\dots-X_m-Y$ is called active for evidence E , if none of $X_1, \dots, X_m \in E$
- Variables X and Y are called **separated** by E if there is no active trail for E connecting X, Y
Write $\text{sep}(X, Y \mid E)$
- $I(G) = \{X \downarrow Y \mid E: \text{sep}(X, Y \mid E)\}$

Soundness of separation

- Know: For positive distributions $P > 0$
 $I_{loc}(G) \subseteq I(P) \Leftrightarrow P$ factorizes over G
- **Theorem:** Soundness of separation
 - For positive distributions $P > 0$
 - $I_{loc}(G) \subseteq I(P) \Leftrightarrow I(G) \subseteq I(P)$
- Hence, separation captures only true independences
- How about $I(G) = I(P)$?

Completeness of separation

Theorem: Completeness of separation

$$I(G) = I(P)$$

for “almost all” distributions P that factorize over G

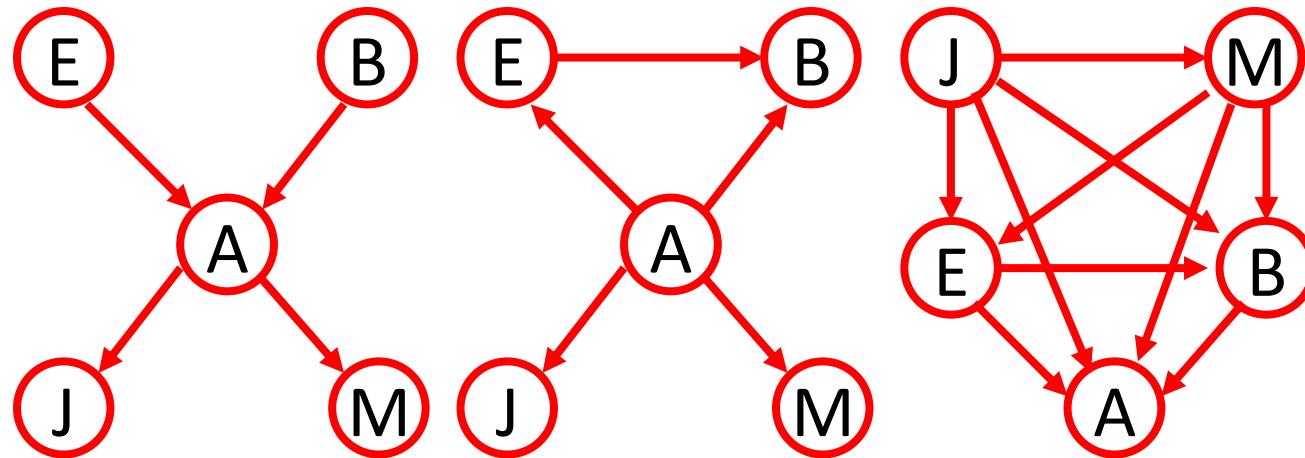
“almost all”: Except for of potential parameterizations of measure 0 (assuming no finite set have positive measure)

Independences in Markov Nets?

- In Bayes Nets (G, P)
 - Local Markov Assumption: $X \perp \text{NonDesc}(X) \mid \text{Pa}_X$
 - G is I-map for distribution P if Local Markov Assumption holds
 - Factorization Thm: P factorizes over $G \Leftrightarrow G$ is an I-map
 - Global independences: d-separation
 - Completeness and soundness of d-separation
- How about Markov Nets?
 - Local Markov Assumption: $X \perp \text{EverythingElse} \mid \text{MB}(X)$
 - Factorization Thm: For positive P , P factorizes $\Leftrightarrow G$ is an I-map
 - Global independences: separation
 - For positive P : separation is complete and sound
- How about minimal I-maps and P-maps??

Minimal I-maps

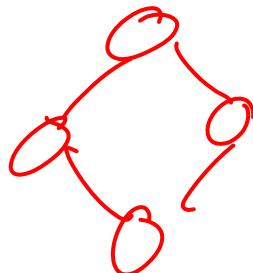
- For BNs: Minimal I-map not unique



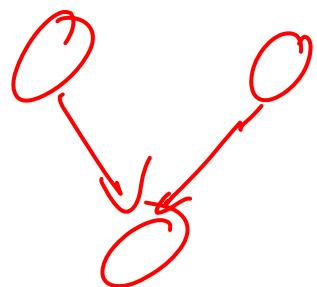
- For MNs: For positive P, minimal I-map is unique!!

P-maps

- Do P-maps always exist?
- For BNs: no



- How about Markov Nets?



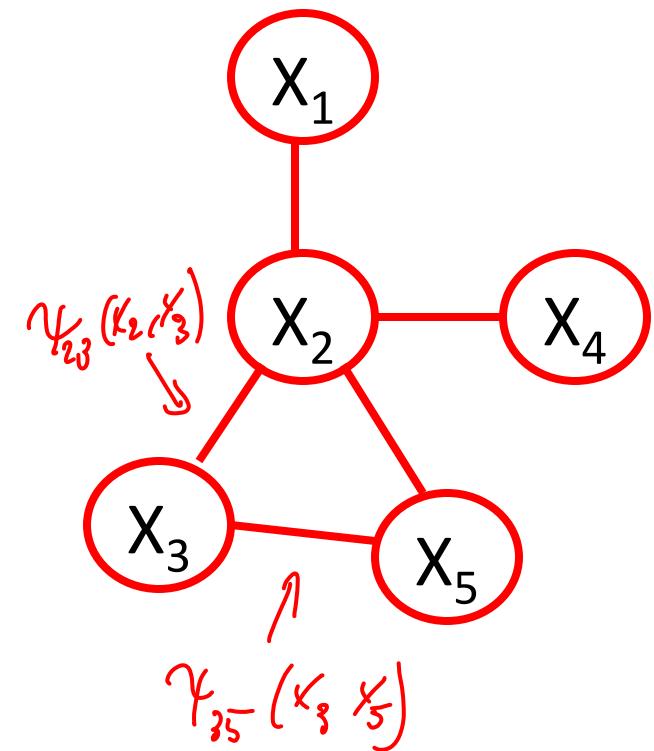
does not have
MN P-map!

Exact inference in MNs

- Variable elimination and junction tree inference work exactly the same way!
 - Need to construct junction trees by obtaining chordal graph through triangulation

Pairwise MNs

- A pairwise MN is a MN where all factors are defined over single variables or pairs of variables
- Can reduce any MN to pairwise MN!



Tasks

- Read Koller & Friedman Chapters 10 and 4.1-4.5

Probabilistic Graphical Models

Lecture 10 – Undirected Models

CS/CNS/EE 155
Andreas Krause

Announcements

- Homework 2 due this Wednesday (Nov 4) in class
- Project milestones due next Monday (Nov 9)
 - About half the work should be done
 - 4 pages of writeup, NIPS format
 - <http://nips.cc/PaperInformation/StyleFiles>

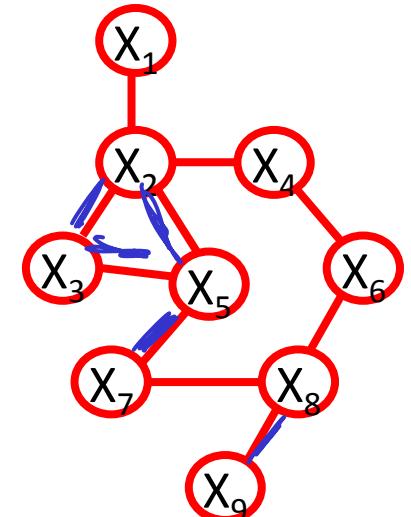
Markov Networks

(a.k.a., Markov Random Field, Gibbs Distribution, ...)

- A Markov Network consists of
 - An undirected graph, where each node represents a RV
 - A collection of factors defined over cliques in the graph
- Joint probability

$$P(X) = \frac{1}{Z} \prod_i \psi_i(C_i)$$

P



- A distribution factorizes over undirected graph G if

\exists factors $\psi_1 \dots \psi_k$ over cliques of G s.t.

$$P(X) = \frac{1}{Z} \prod_i \psi_i(C_i)$$

Computing Joint Probabilities

- Computing joint probabilities in BNs

$$P(X_1, \dots, X_m) = \prod_i P(X_i | P_{a_i})$$

$$P(X_1 | X_m)$$

actually comp. $P(X_1, X_m)$

$$\mathcal{Z} = \sum \prod_i \psi_i(c_i)$$

- Computing joint probabilities in Markov Nets

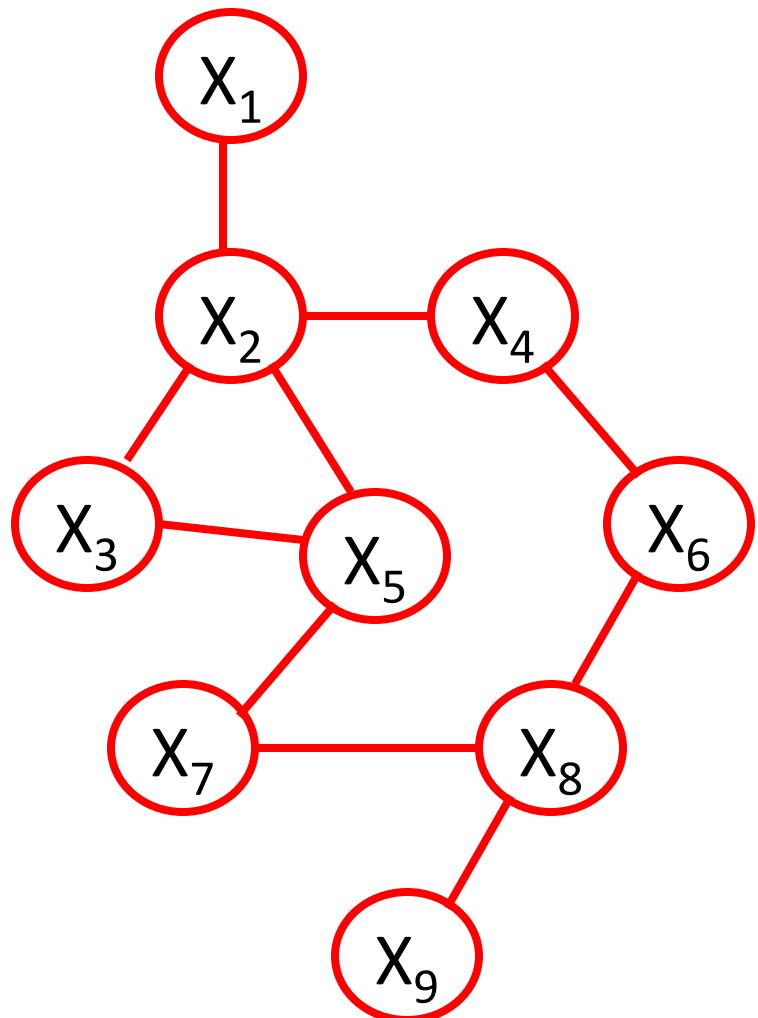
$$P(\underline{X_1}, \dots, \underline{X_m}) = \frac{1}{\mathcal{Z}} \prod_i \psi_i(c_i)$$

↑
Need to know partition "function" \mathcal{Z}

✓ Can do V_E
Variable
elimination

Can compute $\frac{P(X_1, \dots, X_m)}{P(X'_1, \dots, X'_m)} = \frac{\prod_i \psi_i(c_i)}{\prod_i \psi_i(c'_i)}$

Local Markov Assumption for MN



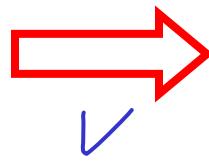
- The **Markov Blanket $\text{MB}(X)$** of a node X is the set of neighbors of X
- Local Markov Assumption:
 $X \perp \text{EverythingElse} \mid \text{MB}(X)$
- $I_{\text{loc}}(G) = \text{set of all local independences}$
- G is called an I -map of distribution P if $I_{\text{loc}}(G) \subseteq I(P)$

Factorization Theorem for Markov Nets “ \Rightarrow ”

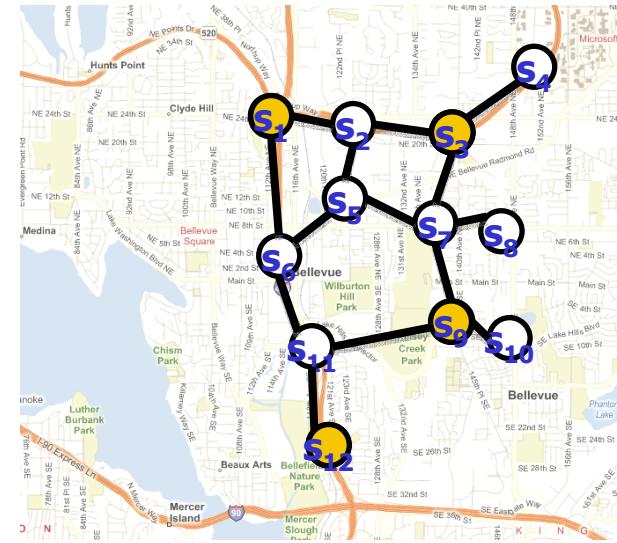


True distribution P
can be represented exactly as
a Markov net (G, P)

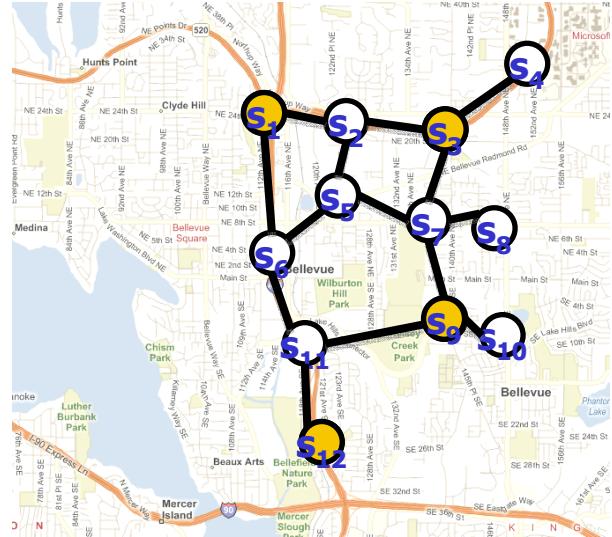
$$P(X_1, \dots, X_n) = \frac{1}{Z} \prod_i \phi_i(C_i)$$



$I_{loc}(G) \subseteq I(P)$
 G is an **I-map** of P
(independence map)

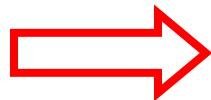


Factorization Theorem for Markov Nets “ \leftarrow ” Hammersley-Clifford Theorem



True distribution P
can be represented exactly as

$$I_{loc}(G) \subseteq I(P)$$

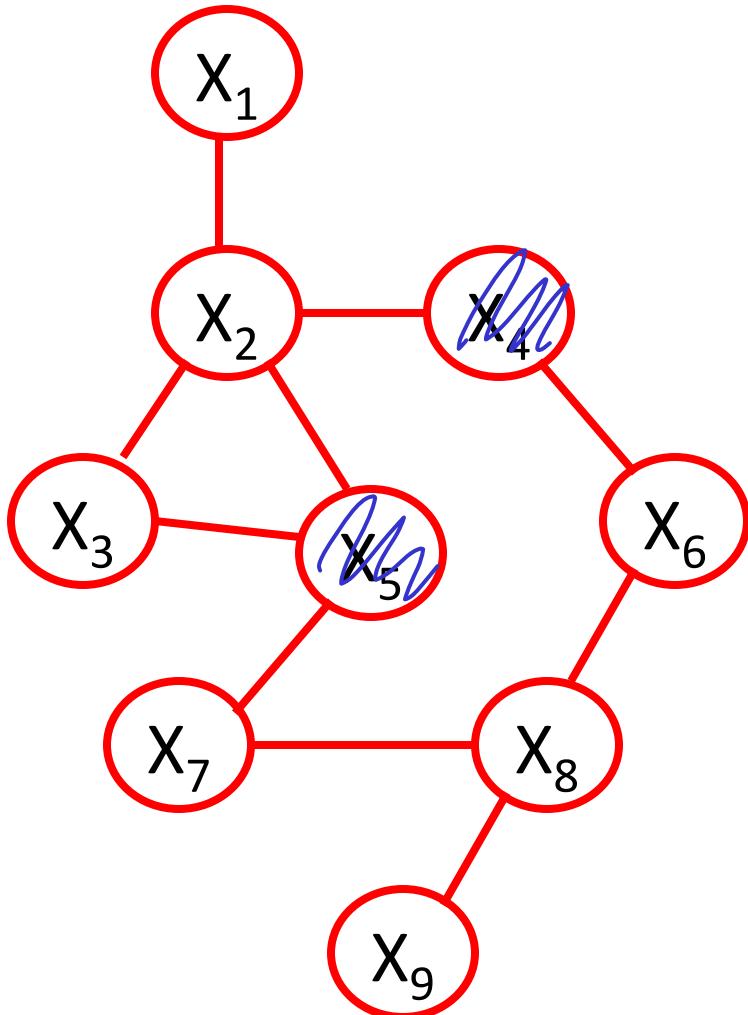


$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}_{X_i})$$

i.e., P can be represented as
a Markov net (G, P)

G is an **I-map** of P
(independence map)
and $P > 0$

Global independencies



- A trail $X-X_1-\dots-X_m-Y$ is called active for evidence E , if none of $X_1, \dots, X_m \in E$
- Variables X and Y are called **separated** by E if there is no active trail for E connecting X, Y
Write $\text{sep}(X, Y \mid E)$
- $I(G) = \{X \downarrow Y \mid E: \text{sep}(X, Y \mid E)\}$

Soundness of separation

- Know: For positive distributions $P > 0$
$$I_{loc}(G) \subseteq I(P) \Leftrightarrow P \text{ factorizes over } G$$
- **Theorem:** Soundness of separation
 - For positive distributions $P > 0$
$$I_{loc}(G) \subseteq I(P) \Leftrightarrow I(G) \subseteq I(P)$$
- Hence, separation captures only true independences
- How about $I(G) = I(P)$?

Completeness of separation

Theorem: Completeness of separation

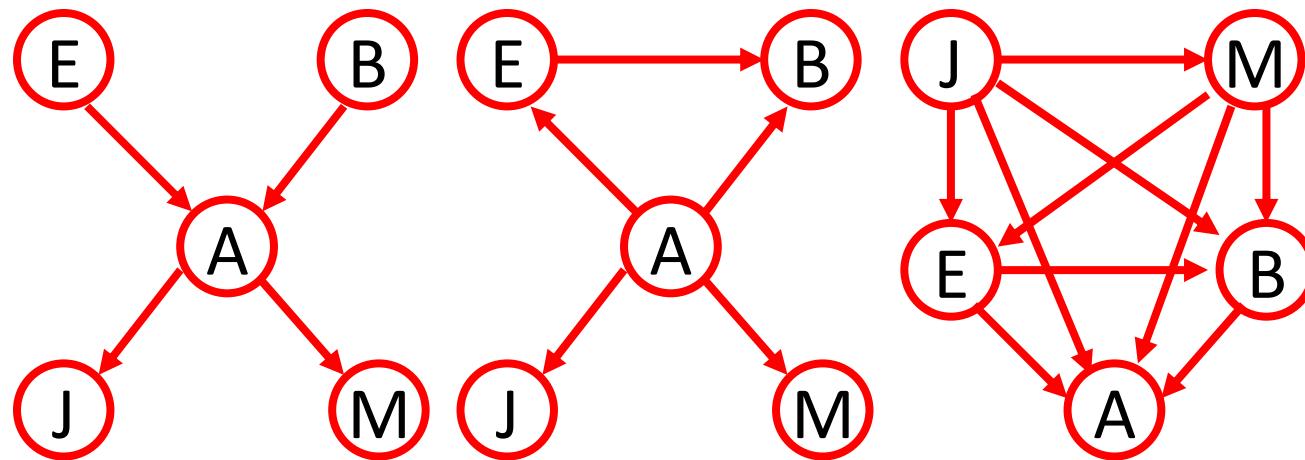
$$I(G) = I(P)$$

for “almost all” distributions P that factorize over G

“almost all”: Except for of potential parameterizations of measure 0 (assuming no finite set have positive measure)

Minimal I-maps

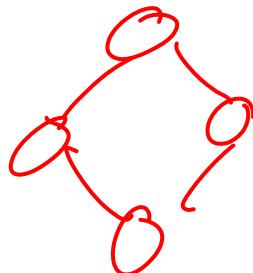
- For BNs: Minimal I-map not unique



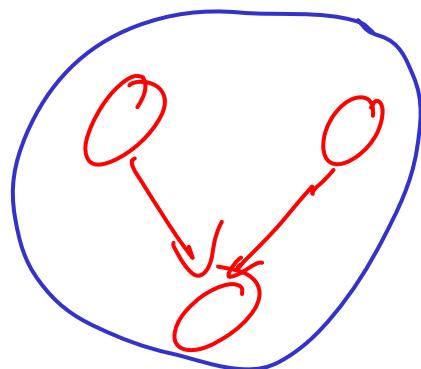
- For MNs: For positive P, minimal I-map is unique!!

P-maps

- Do P-maps always exist?
- For BNs: no



- How about Markov Nets?



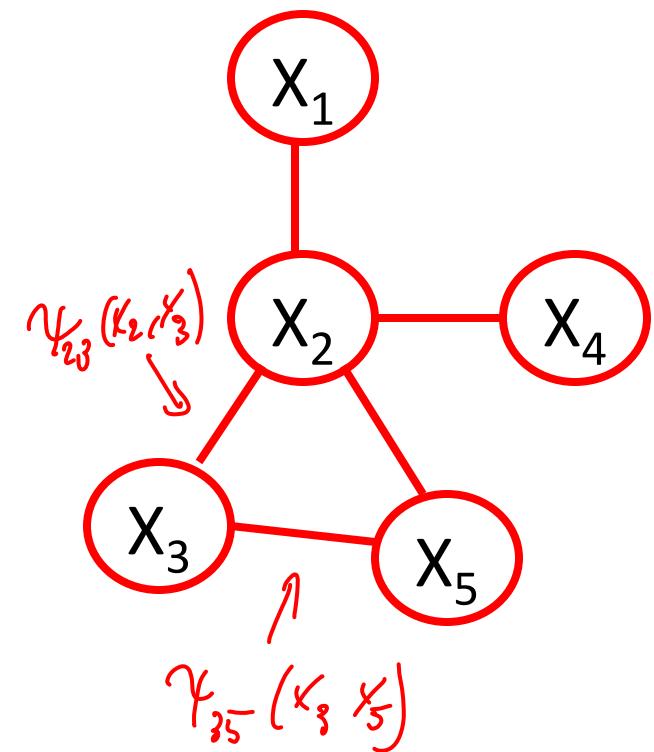
does not have
MN P-map!

Exact inference in MNs

- Variable elimination and junction tree inference work exactly the same way!
 - Need to construct junction trees by obtaining chordal graph through triangulation

Pairwise MNs

- A pairwise MN is a MN where all factors are defined over single variables or pairs of variables
- Can reduce any MN to pairwise MN!



Logarithmic representation

- Can represent any positive distribution in log domain

$$P(x) = \frac{1}{Z} \prod_i \psi_i(c_i)$$

$$\log P(x) = \sum_i \underbrace{\log \psi_i(c_i)}_{\Psi_i(c_i)} - \log Z$$

$$P(x) = \frac{1}{Z} \exp \left(\sum_i \Psi_i(c_i) \right)$$

Log-linear models

- Feature functions $\phi_i(D)$ defined over cliques

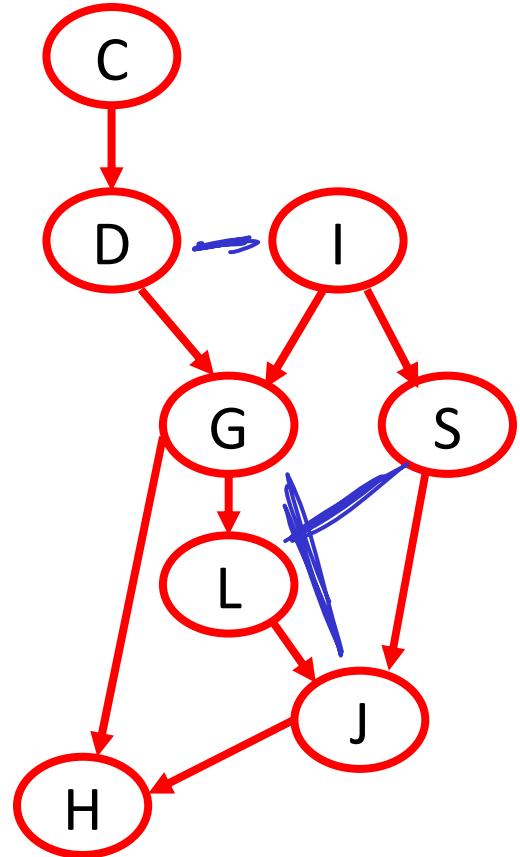
$$\phi_i(x_i, x_{i+1}) = \begin{cases} 1 & \text{if } x_i = x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

- Log linear model over undirected graph G

- Feature functions $\phi_1(D_1), \dots, \phi_k(D_k)$
- Domains D_i can overlap
- Set of weights w_i learnt from data

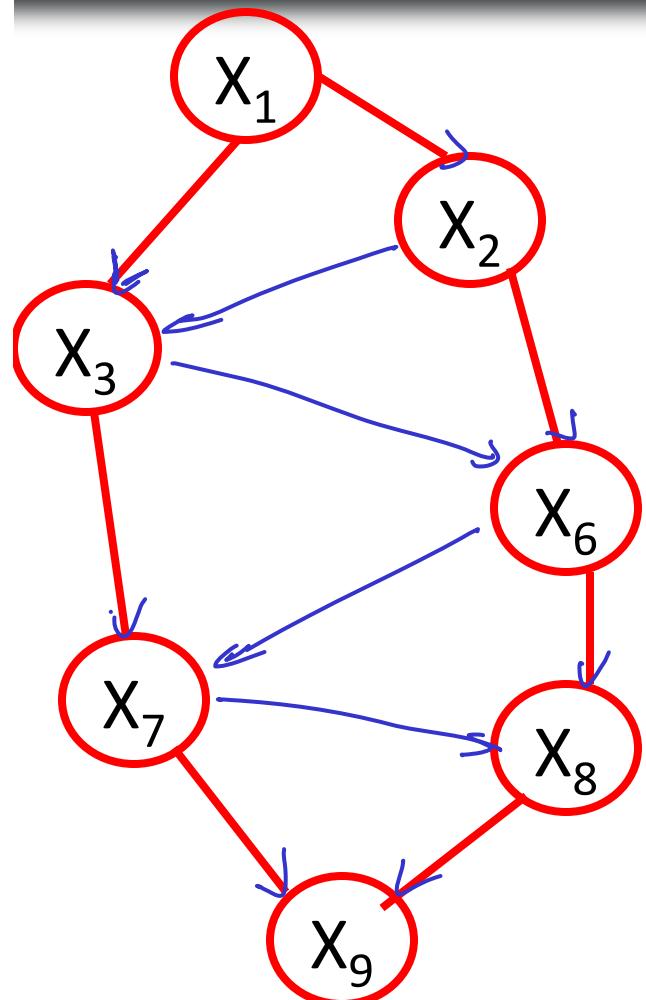
$$P(X) = \frac{1}{Z} \exp \left(\sum_i w_i^T \phi_i(C_i) \right)$$

Converting BNs to MNs



Theorem: Moralized Bayes net is minimal Markov I-map

Converting MNs to BNs



Resulting BN has far fewer cond. independencies than original MN

$$I(G') \leq I(G)$$

\uparrow
BN MN

Theorem: Minimal Bayes I-map for MN must be chordal

So far

- **Markov Network Representation**
 - Local/Global Markov assumptions; Separation
 - Soundness and completeness of separation
- **Markov Network Inference**
 - Variable elimination and Junction Tree inference work exactly as in Bayes Nets
- How about **Learning** Markov Nets?

Parameter Learning for Bayes nets

$$\begin{aligned}\log P(D|\theta) &= \log \prod_l \prod_i P(X_i^{(l)} | \text{Pa}_i^{(l)}; \theta) \\ &= \sum_l \sum_i \log P(X_i^{(l)} | \text{Pa}_i^{(l)}; \theta_{X_i(\text{Pa}_i)})\end{aligned}$$

Parameter
indepent

$$\begin{aligned}\frac{\partial}{\partial \theta_{X_i(\text{Pa}_i)}} \log P(D|\theta) &= \sum_j \sum_l \frac{\partial}{\partial \theta_{X_i(\text{Pa}_i)}} \log P(X_j^{(l)} | \text{Pa}_j^{(l)}; \theta_{X_j(\text{Pa}_j)}) \\ &= \sum_l \frac{\partial}{\partial \theta_{X_i(\text{Pa}_i)}} \log P(X_i^{(l)} | \text{Pa}_i; \theta_{X_i(\text{Pa}_i)}) \stackrel{!}{=} 0\end{aligned}$$

Problem breaks down into independent subproblems

Learn every CPD independent of others

Algorithm for BN MLE

Given BN structure G

For each variable X_i :

$$\text{learn } \hat{\theta}_{X_i | \text{pa}_i} = \frac{\text{Count}(X_i, \text{pa}_i)}{\text{Count}(\text{pa}_i)}$$

\Rightarrow globally maximum likelihood estimate
for fixed structure G

MLE for Markov Nets

- Log likelihood of the data

$$\begin{aligned}\log P(D|\theta) &= \sum_{\ell} \log P(x^{(\ell)}|\theta) \\ &= \sum_{\ell=1}^m \log \frac{1}{Z} \prod_i \psi_i(c_i^{(\ell)}) \\ &= \sum_{\ell} \sum_i \log \psi_i(c_i^{(\ell)}) - m \log Z \\ &= m \sum_i \sum_{c_i} \hat{P}(c_i) \underbrace{\log \psi_i(c_i)}_{\text{in } B \text{ or } \log P(K_i|Pa_i)} - \underbrace{m \log Z}_{\text{not in } B}\end{aligned}$$

$$Z = Z(\theta) = \sum_x \prod_i \psi_i(c_i) \quad \log Z = \log \sum_x \prod_i \psi_i(c_i)$$

\uparrow
 $\text{wrt } \theta$

Log-likelihood doesn't decompose

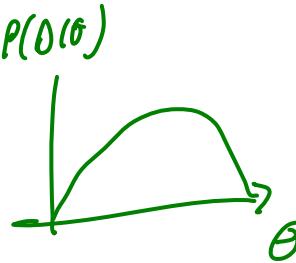
- Log likelihood

$$\log P(\mathcal{D} \mid \theta) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z(\theta)$$

decomposes nicely *does not decompose*

↑

- $I(D \mid \theta)$ is concave function! $\log P(\mathcal{D} \mid \theta)$
No local optima!
Gradient ascent won't get stuck!
- Log Partition function $\log Z(\theta)$ doesn't decompose



Derivative of log-likelihood

$$\log P(\mathcal{D} \mid \theta) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z(\theta)$$

$$\begin{aligned}\frac{\partial \log P(\mathcal{D} \mid \theta)}{\partial \psi_i(c_i)} &= m \underbrace{\sum_j \sum_{c_{ij}} \hat{P}(c_{ij}) \frac{\partial}{\partial \psi_i(c_i)} \log \psi_j(c_{ij})}_{-m \frac{\partial}{\partial \psi_i(c_i)} \log Z(\theta)} \\ &= m \hat{P}(c_i) \frac{1}{\psi_i(c_i)} - m \frac{\partial}{\partial \psi_i(c_i)} \log Z(\theta)\end{aligned}$$

Derivative of log-likelihood

$$\psi_i \quad \begin{array}{c|cc} A & B \\ \hline 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array} \quad \begin{array}{c} \psi_i(A, B) \\ \hline \psi_i(0, 0) \\ \psi_i(0, 1) \\ \psi_i(1, 0) \end{array}$$

$$\therefore \frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(\mathbf{c}_i)} = m \frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - m \frac{\partial \log Z(\theta)}{\partial \psi(\mathbf{c}_i)}$$

$$\frac{\partial \log Z(\theta)}{\partial \psi_i(c_i)} = \frac{\frac{\partial \psi_i(c_i)}{\partial \theta} Z(\theta)}{Z(\theta)} = \frac{\cancel{\sum_j} \frac{\partial P(c_i | \theta)}{\partial \psi_i(c_i)}}{\cancel{\sum_j} \psi_i(c_i)} = \frac{P(c_i | \theta)}{\psi_i(c_i)}$$

$$\begin{aligned} \frac{\partial Z(\theta)}{\partial \psi_i(c_i)} &= \frac{\partial}{\partial \psi_i} \underbrace{\sum_x \prod_j \psi_j(c_j)}_{Z(\theta)} = \sum_x \underbrace{\frac{\partial}{\partial \psi_i} \prod_j \psi_j(c_j)}_0 \\ &\quad \text{if } x \neq c_i \\ &\quad x \text{ inconsistent w/ } c_i \\ &= \sum_{x \sim c_i} \prod_{j \neq i} \psi_j(c_j) \frac{\psi_i(c_i)}{\psi_i(c_i)} \\ &= \frac{Z P(c_i | \theta)}{\psi_i(c_i)} \end{aligned}$$

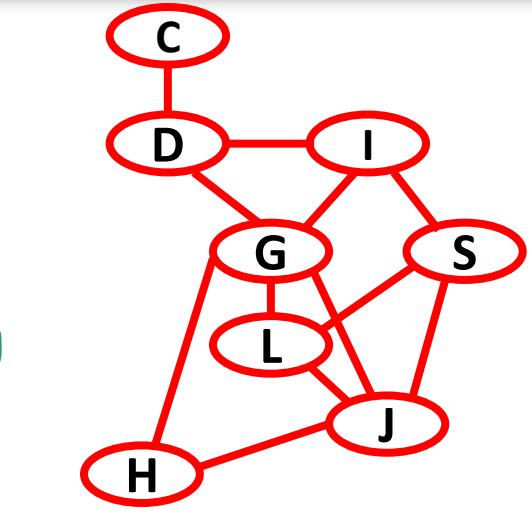
Computing the derivative

- Derivative

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(\mathbf{c}_i)} = m \frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - m \frac{P(\mathbf{c}_i | \theta)}{\psi_i(\mathbf{c}_i)}$$

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(G=1, D=0, I=1)} = m \frac{\hat{P}(1, 0, 1)}{\psi_i(1, 0, 1)} - m \frac{P(1, 0, 1 | \theta)}{\psi_i(1, 0, 1)}$$

- Computing $P(\mathbf{c}_i | \theta)$ requires inference!



Can do this
using VE
Junction tree ...

- Can optimize using conjugate gradient etc.

Alternative approach: Iterative Proportional Fitting (IPF)

- At optimum, it must hold that

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(\mathbf{c}_i)} = m \frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - m \frac{P(\mathbf{c}_i | \theta)}{\psi_i(\mathbf{c}_i)} = 0$$

At opt.: $\frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} = \frac{P(\mathbf{c}_i | \theta)}{\psi_i(\mathbf{c}_i)}$ "Data agrees with model on marginals"

- Solve fixed point equation $\psi_i^{(t)}(\mathbf{c}_i) = 1$

$$\psi_i^{(t+1)}(\mathbf{c}_i) = \psi_i^{(t)}(\mathbf{c}_i) \cdot \frac{\hat{P}(\mathbf{c}_i)}{P(\mathbf{c}_i | \theta)}$$

- Must recompute parameters every iteration

$$P(\mathbf{c}_i | \theta)$$

Parameter learning for log-linear models

- Feature functions $\phi_i(C_i)$ defined over cliques
- Log linear model over undirected graph G
 - Feature functions $\phi_1(C_1), \dots, \phi_k(C_k)$
 - Domains C_i can overlap
- Joint distribution

$$P(X_1, \dots, X_n) = \frac{1}{Z} \exp\left(\underbrace{\sum_i w_i^T \phi_i(C_i)}\right)$$

- How do we get weights w_i ?

Derivative of Log-likelihood 1

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial w_i} = m \underbrace{\sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \frac{\partial w_i^T \phi_i(\mathbf{c}_i)}{\partial w_i}} - m \frac{\partial \log Z(w)}{\partial w_i}$$

$$= m \underbrace{\sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \phi_i(\mathbf{c}_i)} - m \underbrace{\frac{\partial \log Z(w)}{\partial w_i}}$$

$\hat{E}[\phi_i]$

If $\hat{\phi}_i(x_i, x_{i+1}) = \begin{cases} 1 & \text{if } x_i = x_{i+1} \\ 0 & \text{otherwise} \end{cases}$, $\hat{E}[\phi_i] = \frac{\text{Count}(x_i = x_{i+1})}{m}$

Derivative of Log-likelihood 2

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial w_i} = m \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \phi_i(\mathbf{c}_i) - m \frac{\partial \log Z(w)}{\partial w_i}$$

$$\begin{aligned}\frac{\partial}{\partial w_i} \log Z(w) &= \frac{1}{Z(w)} \frac{\partial}{\partial w_i} \sum_x \exp \left(\sum_i w_i^\top \phi_i(c_i) \right) \\ &= \frac{1}{Z(w)} \sum_x \phi_i(c_i) \exp \left(\sum_i w_i^\top \phi_i(c_i) \right) \\ &= \sum_x \phi_i(c_i) \underbrace{\frac{1}{Z} \exp \left(\sum_i w_i^\top \phi_i(c_i) \right)}_{P(x)} \\ &= \sum_{c_i} \phi_i(c_i) P(c_i | w) \\ &= \mathbb{E}_w (\phi_i)\end{aligned}$$

Optimizing parameters

- Gradient of log-likelihood

$$\frac{\partial \log P(\mathcal{D} | w)}{\partial w_i} = m \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \phi_i(\mathbf{c}_i) - m \sum_{\mathbf{c}_i} P(\mathbf{c}_i | w) \phi_i(\mathbf{c}_i)$$

$\underbrace{\hat{E}[\phi_i]}$ $\underbrace{E_w[\phi_i]}$

- Thus, w is MLE $\Leftrightarrow \hat{E}[\phi_i] = E_w[\phi_i]$

Regularization of parameters

- Put prior on parameters w

$$P(w)$$

$$\frac{\partial \log P(\mathcal{D} | w)P(w)}{\partial w_i} = m \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \phi_i(\mathbf{c}_i) - m \sum_{\mathbf{c}_i} P(\mathbf{c}_i | w) \phi_i(\mathbf{c}_i) + \underbrace{\frac{\partial \log P(w)}{\partial w_i}}_{\text{L2}}$$

Prior: $P(w) = \mathcal{N}(w; 0, I) \propto \exp(-\sum_i w_i^2)$

$$(*) \log P(w) = -\sum_i w_i^2$$

$$\frac{\partial}{\partial w_i} \log P(w) = -2w_i$$

Summary: Parameter learning in MN

- MLE in BN is easy (score decomposes)
- MLE in MN requires inference (score doesn't decompose)
- Can optimize using gradient ascent or IPF

Tasks

- Read Koller & Friedman Chapters 20.1-20.3, 4.6.1

Probabilistic Graphical Models

Lecture 11 – CRFs, Exponential
Family

CS/CNS/EE 155
Andreas Krause

Announcements

- Homework 2 due today
- Project milestones due next Monday (Nov 9)
 - About half the work should be done
 - 4 pages of writeup, NIPS format
 - <http://nips.cc/PaperInformation/StyleFiles>

So far

- **Markov Network Representation**
 - Local/Global Markov assumptions; Separation
 - Soundness and completeness of separation
- **Markov Network Inference**
 - Variable elimination and Junction Tree inference work exactly as in Bayes Nets
- How about **Learning** Markov Nets?

MLE for Markov Nets

- Log likelihood of the data

$$\begin{aligned}\log P(D|\theta) &= \sum_{\ell} \log P(x^{(\ell)}|\theta) \\ &= \sum_{\ell=1}^m \log \frac{1}{Z} \prod_i \psi_i(c_i^{(\ell)}) \\ &= \sum_{\ell} \sum_i \log \psi_i(c_i^{(\ell)}) - m \log Z \\ &\doteq m \sum_i \sum_{c_i} \hat{P}(c_i) \underbrace{\log \psi_i(c_i)}_{\text{in } B \text{ or } \log P(K_i|\text{Pa}_i)} - \underbrace{m \log Z}_{\text{not in } B}\end{aligned}$$

$$Z = Z(\theta) = \sum_x \prod_i \psi_i(c_i) \quad \log Z = \log \sum_x \prod_i \psi_i(c_i)$$

\uparrow
 $\text{wrt } \theta$

Log-likelihood doesn't decompose

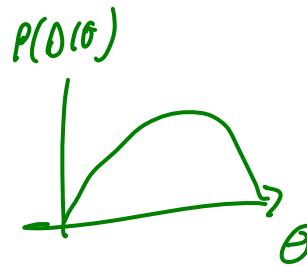
- Log likelihood

$$\log P(\mathcal{D} \mid \theta) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z(\theta)$$

decomposes nicely *does not decompose*

↑

- $I(D \mid \theta)$ is concave function!
No local optima!
Gradient ascent won't get stuck!
- Log Partition function $\log Z(\theta)$ doesn't decompose



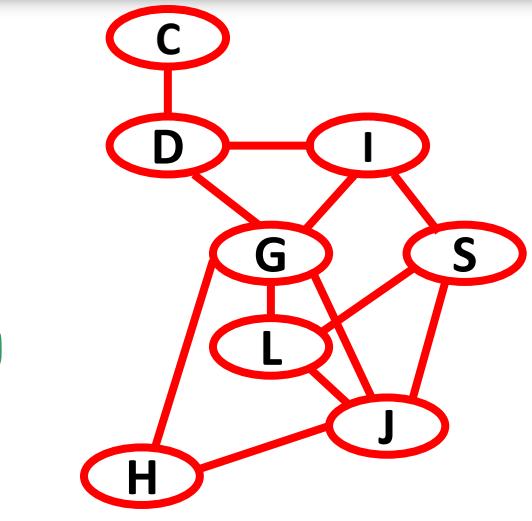
Computing the derivative

- Derivative

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(\mathbf{c}_i)} = m \frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - m \frac{P(\mathbf{c}_i | \theta)}{\psi_i(\mathbf{c}_i)}$$

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(G=1, D=0, I=1)} = m \frac{\hat{P}(1, 0, 1)}{\psi_i(1, 0, 1)} - m \frac{P(1, 0, 1 | \theta)}{\psi_i(1, 0, 1)}$$

- Computing $P(\mathbf{c}_i | \theta)$ requires inference!



Can do this
using VE
Junction tree ...

- Can optimize using conjugate gradient etc.

Alternative approach: Iterative Proportional Fitting (IPF)

- At optimum, it must hold that

$$\frac{\partial \log P(\mathcal{D} | \theta)}{\partial \psi_i(\mathbf{c}_i)} = m \frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - m \frac{P(\mathbf{c}_i | \theta)}{\psi_i(\mathbf{c}_i)} = 0$$

At opt.: $\frac{\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} = \frac{P(\mathbf{c}_i | \theta)}{\psi_i(\mathbf{c}_i)}$ "Data agrees with model on marginals"

- Solve fixed point equation $\psi_i^{(0)}(\mathbf{c}_i) = 1$

$$\psi_i^{(t+1)}(\mathbf{c}_i) = \psi_i^{(t)}(\mathbf{c}_i) \cdot \frac{\hat{P}(\mathbf{c}_i)}{P(\mathbf{c}_i | \theta)}$$

- Must recompute parameters every iteration

$$P(\mathbf{c}_i | \theta)$$

Parameter learning for log-linear models

- Feature functions $\phi_i(C_i)$ defined over cliques
- Log linear model over undirected graph G
 - Feature functions $\phi_1(C_1), \dots, \phi_k(C_k)$
 - Domains C_i can overlap
- Joint distribution

$$P(X_1, \dots, X_n) = \frac{1}{Z} \exp\left(\underbrace{\sum_i w_i^T \phi_i(C_i)}\right)$$

- How do we get weights w_i ?

Optimizing parameters

- Gradient of log-likelihood

$$\frac{\partial \log P(\mathcal{D} | w)}{\partial w_i} = m \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \phi_i(\mathbf{c}_i) - m \sum_{\mathbf{c}_i} P(\mathbf{c}_i | w) \phi_i(\mathbf{c}_i)$$

$\underbrace{\hat{E}[\phi_i]}$ $\underbrace{E_w[\phi_i]}$

- Thus, w is MLE $\Leftrightarrow \hat{E}[\phi_i] = E_w[\phi_i]$

Regularization of parameters

- Put prior on parameters w

$$P(w)$$

$$\frac{\partial \log P(\mathcal{D} | w)P(w)}{\partial w_i} = m \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \phi_i(\mathbf{c}_i) - m \sum_{\mathbf{c}_i} P(\mathbf{c}_i | w) \phi_i(\mathbf{c}_i) + \underbrace{\frac{\partial \log P(w)}{\partial w_i}}_{\text{L2}}$$

Prior: $P(w) = \mathcal{N}(w; 0, I) \propto \exp(-\sum_i w_i^2)$

$$(*) \log P(w) = -\sum_i w_i^2$$

$$\frac{\partial}{\partial w_i} \log P(w) = -2w_i$$

Summary: Parameter learning in MN

- MLE in BN is easy (score decomposes)
- MLE in MN requires inference (score doesn't decompose)
- Can optimize using gradient ascent or IPF

Generative vs. discriminative models

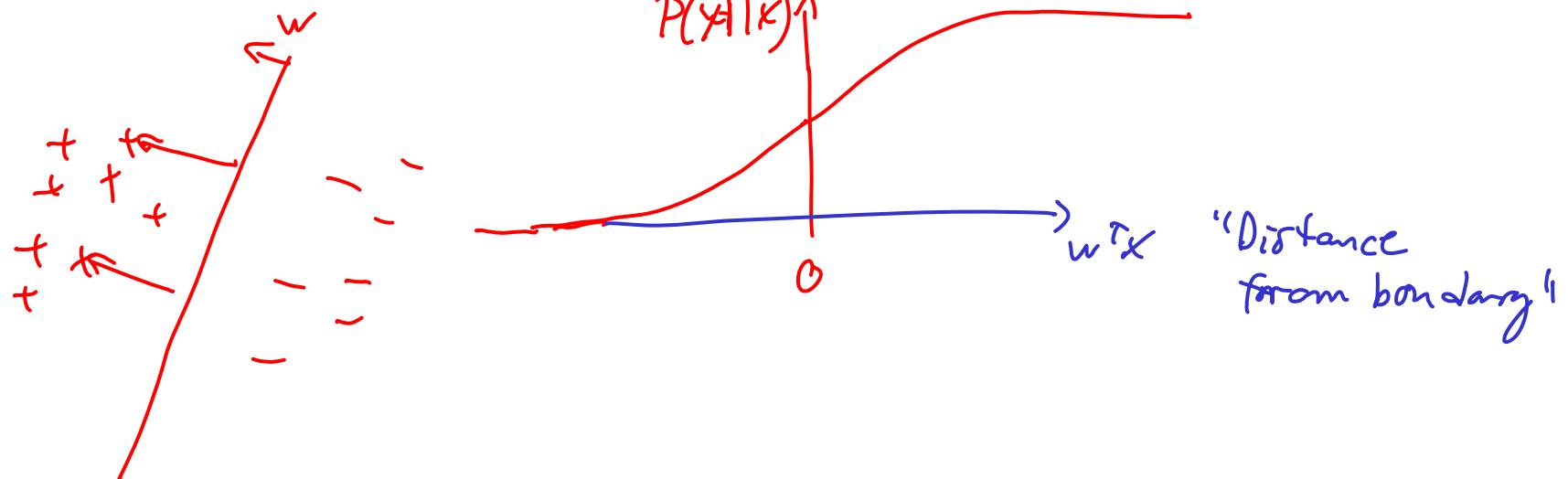
- Often, want to predict Y for inputs X
 - Bayes optimal classifier: Predict according to $P(Y | X)$
- Generative model
 - Model $P(Y)$, $P(X|Y)$
 - Use Bayes' rule to compute $P(Y | X)$
- Discriminative model
 - Model $P(Y | X)$ directly!
 - Don't model distribution $P(X)$ over inputs X
 - Cannot "generate" sample inputs
 - Example: Logistic regression

Example: Logistic Regression

Inputs $x_1 \dots x_m \in \mathbb{R}$

Output $y \in \{0, 1\}$

$$P(Y=1|X) = \frac{1}{1 + \exp(-w^T x)}$$

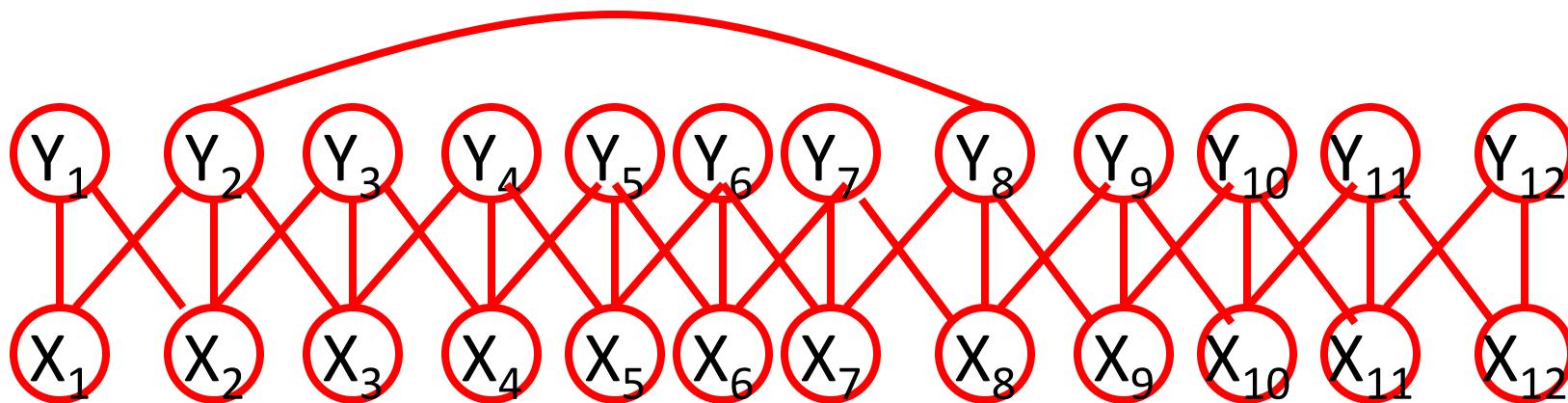


Log-linear conditional random field

- Define log-linear model over outputs Y
 - No assumptions about inputs X
- Feature functions $\phi_i(C_i, x)$ defined over cliques and inputs
 $C_i \subseteq Y$
- Joint distribution

$$P(Y_1, \dots, Y_n \mid x) = \frac{1}{Z(x_{\text{clf}})} \exp\left(\sum_i w_i^T \phi_i(C_i, x)\right)$$

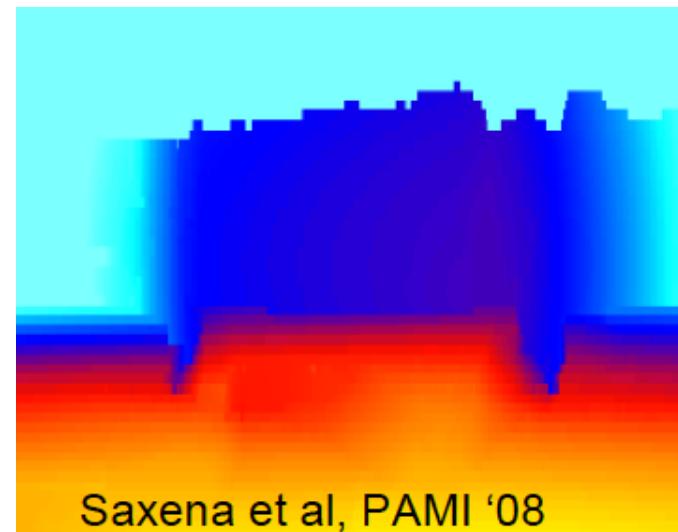
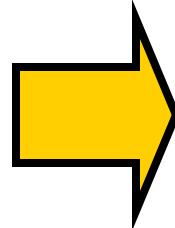
Example: CRFs in NLP



Mrs. Greene spoke today in New York. Green chairs the finance committee

- Classify into Person, Location or Other

Example: CRFs in vision



Parameter learning for log-linear CRF

$$P(Y_1, \dots, Y_n \mid x) = \frac{1}{Z(x)} \exp\left(\sum_i w_i^T \phi_i(C_i, x)\right)$$

- Conditional log-likelihood of data

$$\log P(D_y \mid w, D_x) = \sum_{\ell} \sum_i w_i^T \phi_i(C_i, x^{(\ell)}) - \sum_{\ell} \log Z(x^{(\ell)})$$

- Can maximize using conjugate gradient

Gradient of conditional log-likelihood

- Partial derivative

$$\frac{\partial \log P(\mathcal{D}_Y \mid w, \mathcal{D}_X)}{\partial w_i} = \sum_j \left[\phi_i(\mathbf{c}_i^{(j)}, x^{(j)}) + \underbrace{\sum_{\mathbf{c}_i} P(\mathbf{c}_i \mid w, \underline{x^{(j)}}) \phi_i(\mathbf{c}_i, \underline{x^{(j)}})}_{\text{Reg. Inference}} \right]$$

- Requires one inference per feature and per data point

Can be very expensive

Can do "pseudo" likelihood est. (approximate)

- Can optimize using conjugate gradient

Exponential Family Distributions

- Distributions for log-linear models

$$P(X_1, \dots, X_n) = \frac{1}{Z} \exp\left(\sum_i w_i^T \phi_i(C_i)\right)$$

- More generally: **Exponential family distributions**

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- $h(x)$: Base measure \leftarrow Often constant
- w : natural parameters
- $\phi(x)$: Sufficient statistics
- $A(w)$: log-partition function $\leftarrow A(w) = \log \tau(w)$
- Hereby x can be continuous (defined over any set)

Examples

- Exp. Family:

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- Gaussian distribution

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$h(x) = \frac{1}{\sqrt{2\pi}} \quad -\frac{x^2}{2\sigma^2} + \frac{x\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2}$$

$$\phi(x) = \left[-\frac{x^2}{2}, x\right], \quad A(w) = \frac{\mu^2}{2\sigma^2} - \log \sigma$$

$$w = \left[\frac{1}{\sigma^2}, \frac{\mu}{\sigma^2}\right]$$

$h(x)$: Base measure
 w : natural parameters
 $\phi(x)$: Sufficient statistics
 $A(w)$: log-partition function

- Other examples: Multinomial, Poisson, Exponential, Gamma, Weibull, chi-square, Dirichlet, Geometric, ...

Moments and gradients

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- Correspondence between moments and log-partition function (just like in log-linear models)

$$\frac{\partial A(w)}{\partial w_i} = \int p(x \mid w) \phi_i(x) dx = \mathbb{E}[\phi_i \mid w]$$

$$\frac{\partial^2 A(w)}{\partial w_i \partial w_j} = Cov(\phi_i, \phi_j \mid w)$$

- Can compute moments from derivatives, and derivatives from moments!
- MLE \Leftrightarrow moment matching

Recall: 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 $P(D | \theta)$ if $P(\theta | D) = f(\theta; \alpha')$
 - 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??

Conjugate priors in Exponential Family

$$P(x \mid w) = h(x) \exp(w^T \phi(x) - A(w))$$

Any exponential family likelihood has a conjugate prior

$$P(w|\alpha, \beta) = \exp(\alpha^T w - \beta A(w) - B(\alpha, \beta))$$

Maximum Entropy interpretation

Theorem: Exponential family distributions maximize the entropy over all distributions satisfying

$$\max_p - \int p(x) \log p(x) dx$$

$$\text{such that } t_i = \mathbb{E}[\phi_i(x)] = \int p(x) \phi_i(x) dx$$

Summary exponential family

- Distributions of the form

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- Most common distributions are exponential family
 - Multinomial, Gaussian Poisson, Exponential, Gamma, Weibull, chi-square, Dirichlet, Geometric, ...
 - Log-linear Markov Networks
- All exponential family distributions have conjugate prior in EF
- Moments of sufficient stats = derivatives of log-partition function
- Maximum Entropy distributions (“most uncertain” distributions with specified expected sufficient statistics)

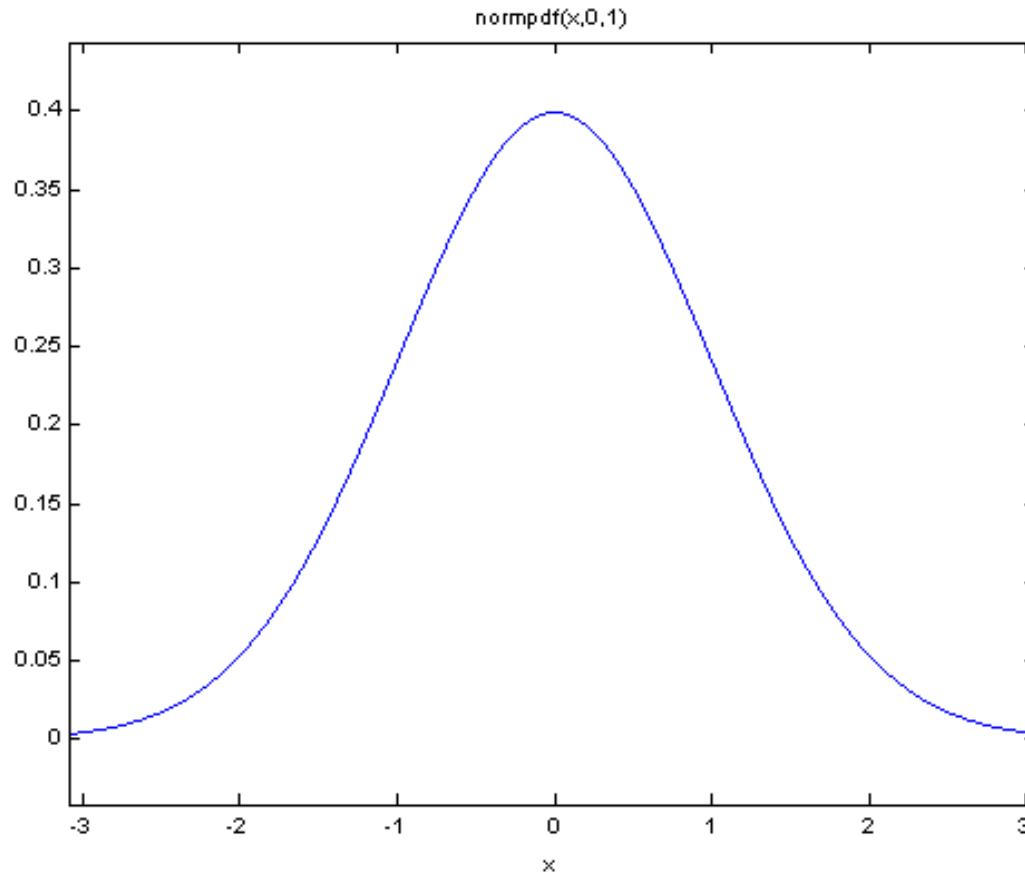
Exponential family graphical models

- So far, only defined graphical models over discrete variables.
- Can define GMs over continuous distributions!
- For exponential family distributions:

$$p(X_1, \dots, X_n) = \prod_i h_i(C_i) \exp\left(\sum_i w_i^T \phi_i(C_i) - A(w)\right)$$
$$\exp A(w) = \int \int \dots \int \prod_i h_i(C_i) \exp\left(\sum_i w_i^T \phi_i(C_i) - A(w)\right) dx_1 \dots dx_n$$

- Can do much of what we discussed (VE, JT, parameter learning, etc.) for such exponential family models
- Important example: **Gaussian Networks**

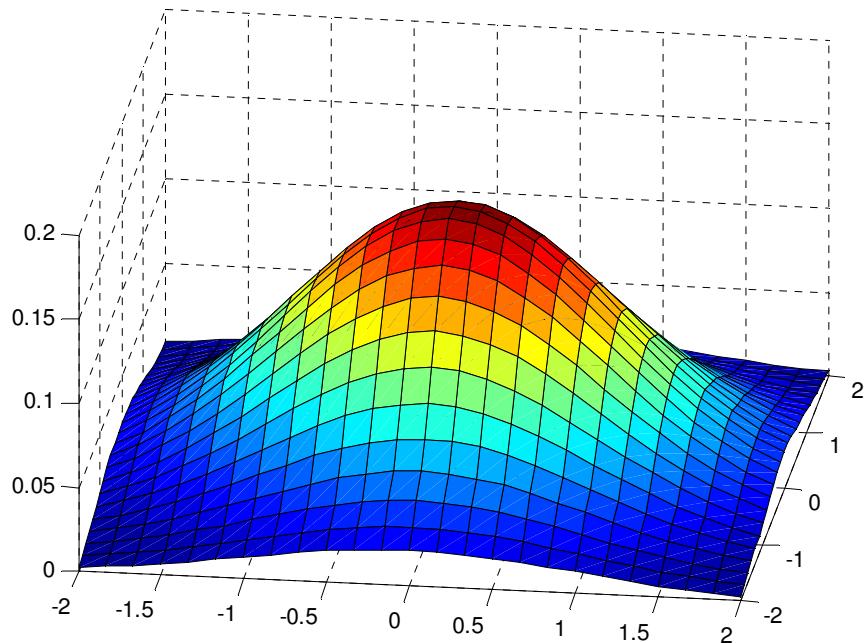
Gaussian distribution



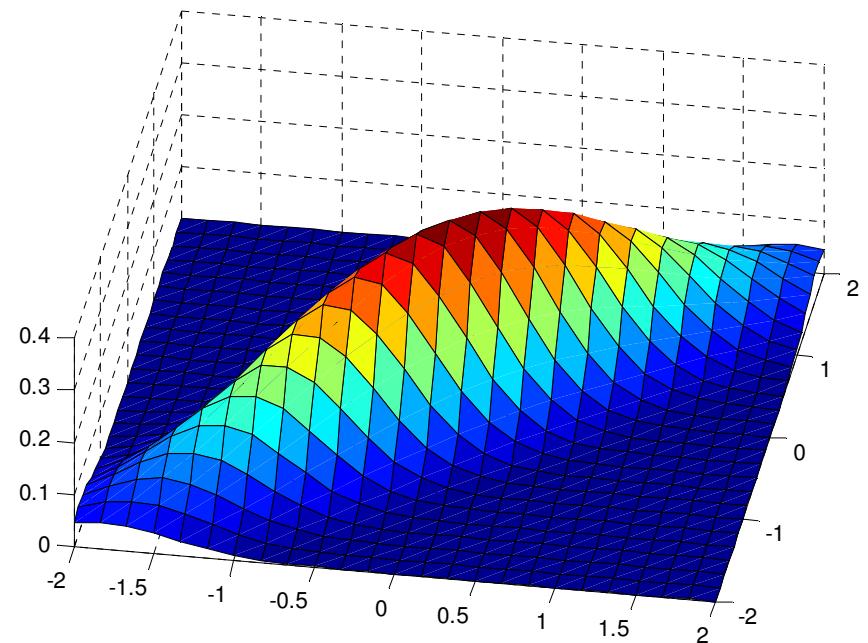
- σ = Standard deviation $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- μ = mean

Bivariate Gaussian distribution

$$\frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^T \underline{\Sigma}^{-1} (x - \mu)\right) \quad \Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix} \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$$

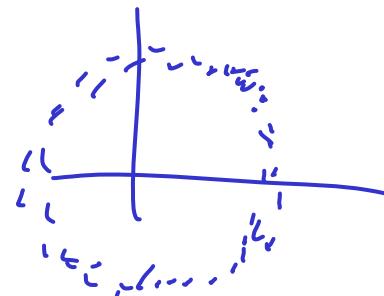
Multivariate Gaussian distribution

$$\mathcal{N}(x; \Sigma, \mu) = \frac{1}{(2\pi)^{n/2} \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \vdots & & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{pmatrix} \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}$$

$\Sigma^\top = \Sigma$
pos. det.: $x^\top \Sigma x \geq 0 \quad \forall x$

- Joint distribution over n random variables $P(X_1, \dots, X_n)$
- $\sigma_{jk} = E[(X_j - \mu_j)(X_k - \mu_k)] = \text{Cov}(X_j, X_k)$
- X_j and X_k independent $\Leftrightarrow \sigma_{jk}=0$



Marginalization

- Suppose $(X_1, \dots, X_n) \sim N(\mu, \Sigma)$

- What is $P(X_1)??$

$$P(X_1) = \iiint \dots \int p(x_1, \dots, x_n) dx_2 \dots dx_n$$

$$P(X_1) = \mathcal{N}(x_1; \mu_1, \sigma_1^2)$$

- More generally: Let $A = \{i_1, \dots, i_k\} \subseteq \{1, \dots, N\}$
- Write $X_A = (X_{i_1}, \dots, X_{i_k})$

- $X_A \sim N(\mu_A, \Sigma_{AA})$

$$\Sigma_{AA} = \begin{pmatrix} \sigma_{i_1}^2 & \sigma_{i_1 i_2} & \dots & \sigma_{i_1 i_k} \\ \vdots & & & \vdots \\ \sigma_{i_k i_1} & \sigma_{i_k i_2} & \dots & \sigma_{i_k}^2 \end{pmatrix}$$

$$\mu_A = \begin{pmatrix} \mu_{i_1} \\ \mu_{i_2} \\ \vdots \\ \mu_{i_k} \end{pmatrix}$$

$$\mu_A = \begin{pmatrix} \mu_{i_1} \\ \mu_{i_2} \\ \vdots \\ \mu_{i_k} \end{pmatrix}$$

Conditioning

- Suppose $(X_1, \dots, X_n) \sim N(\mu, \Sigma)$
- Decompose as (X_A, X_B)
- What is $P(X_A | X_B)??$

$$\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}$$

- $P(X_A = x_A | X_B = x_B) = N(x_A; \mu_{A|B}, \Sigma_{A|B})$ where

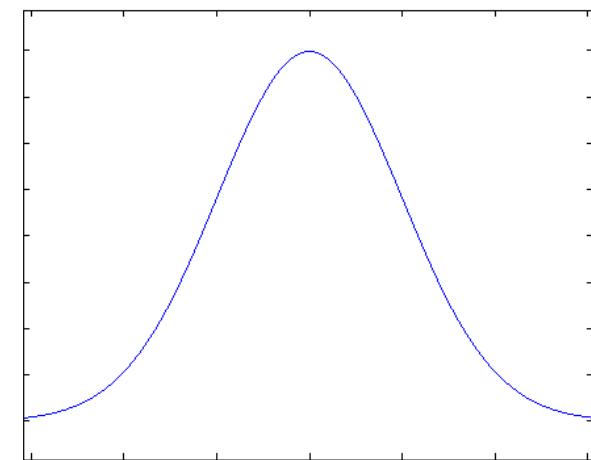
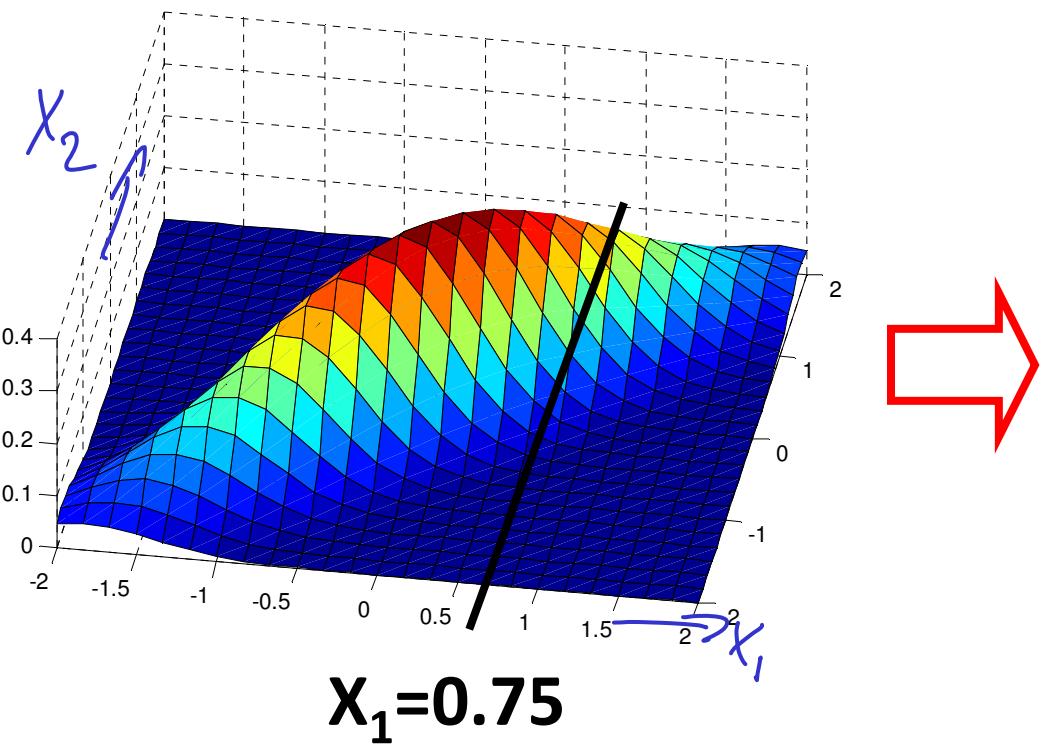
$$\mu_{A|B} = \underline{\mu_A} + \underbrace{\Sigma_{AB} \Sigma_{BB}^{-1}}_W (x_B - \mu_B)$$

$$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB} \Sigma_{BB}^{-1} \Sigma_{BA}$$

- Computable using linear algebra!

Does not depend on x_B

Conditioning



$$P(X_2 | X_1=0.75)$$

Conditional linear Gaussians

$$\mu_{A|B} = \mu_A + \underbrace{\Sigma_{AB}\Sigma_{BB}^{-1}}_w(x_B - \mu_B)$$

$$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$$

$$P(X_A|X_B) = \mathcal{N}(X_A | \mu_{A|B}, \Sigma_{A|B})$$

$$\begin{aligned} X_A &= \mu_A + w X_B - w \mu_B + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \Sigma_{A|B}) \\ &= w X_B + b + \varepsilon \end{aligned}$$

$$w = \Sigma_{AB}\Sigma_{BB}^{-1}$$

$$b = \mu_A - w \mu_B$$

Canonical representation of Gaussians

$$p(X_1, \dots, X_n) = \frac{1}{(2\pi)^{n/2} \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

$$\begin{aligned} & 2 \exp \left(-\frac{1}{2} x^T \Sigma^{-1} x + \frac{1}{2} x^T \Sigma^{-1} \mu + \frac{1}{2} \mu^T \Sigma^{-1} x - \frac{1}{2} \mu^T \Sigma^{-1} \mu \right) \\ &= \exp \left(-\frac{1}{2} x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu - A(\Sigma^{-1}, \mu) \right) \\ &= \exp \left(-\frac{1}{2} x^T \Lambda x + x^T \gamma \right) \end{aligned}$$

precision
matrix

$$\Lambda = \Sigma^{-1}$$

$$\gamma = \Sigma^{-1} \mu$$

Canonical Representation

$$p(X_1, \dots, X_n) = \frac{1}{(2\pi)^{n/2} \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$
$$\propto \exp(\eta^T x - \frac{1}{2} x^T \Lambda x) = \exp \left(\sum_i \eta_i x_i - \frac{1}{2} \sum_{i,j} x_i x_j \Lambda_{i,j} \right)$$

- Multivariate Gaussians in exponential family!
- Standard vs canonical form:

$$\mu = \Lambda^{-1} \eta$$

$$\Sigma = \Lambda^{-1}$$

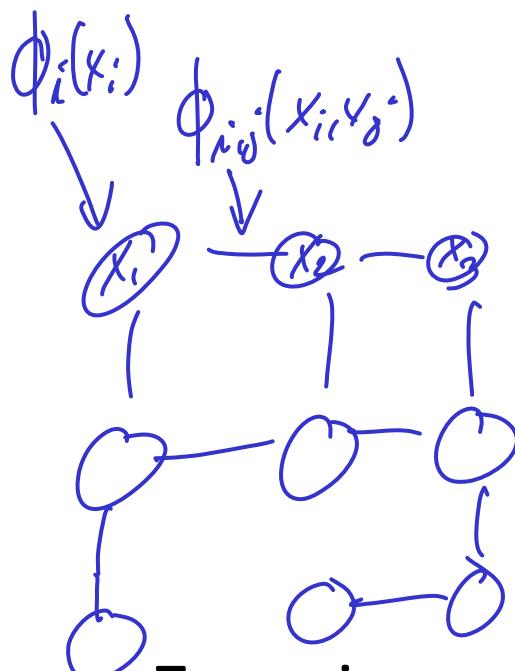
- In standard form: Marginalization is easy
- Will see: In canonical form, multiplication/conditioning is easy!

Gaussian Networks

$$p(X_1, \dots, X_n) \propto \exp\left(-\frac{1}{2} \sum_{i,j} \lambda_{i,j} x_i x_j + \sum_i \eta_i x_i\right)$$

$$= \exp\left(\sum_{i,j} \lambda_{i,j} \phi_{i,j}(x_i, x_j) + \sum_i \eta_i \phi_i(x_i) \right)$$

$$\phi_{i,j}(x_i, x_j) = -\frac{1}{2} x_i x_j \quad \phi_i(x_i) = x_i$$



No edge between x_i, x_j

$$\Leftrightarrow \lambda_{ij} = 0$$

Zeros in precision matrix Λ indicate missing edges in log-linear model!

Inference in Gaussian Networks

- Can compute marginal distributions in $O(n^3)$!
- For large numbers n of variables, still intractable
- If Gaussian Network has low treewidth, can use variable elimination / JT inference!
- Need to be able to multiply and marginalize factors!

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

Multiplying factors in Gaussians

$$P(X_A) = \mathcal{N}(x_A; \Lambda_1, \gamma_1) \quad \begin{matrix} |A|=k, |B|=m \\ \Lambda_1 \in \mathbb{R}^{k \times k} \end{matrix}$$

$$P(X_B|X_A) = \mathcal{N}(x_B|x_A; \Lambda_2, \gamma_2) \leftarrow \begin{matrix} \text{CLG representation} \\ \Lambda_2 \in \mathbb{R}^{m \times m} \end{matrix}$$

$$P(X_A, X_B) = P(X_A) P(X_B|X_A)$$

$$\propto \exp(x_A^\top \Lambda_1 x_A + \gamma_1^\top x_A) \exp((x_A, x_B)^\top \Lambda_2 (x_A, x_B) + \gamma_2^\top (x_A, x_B))$$

$$= \mathcal{N}(x_A, x_B; \Lambda, \gamma)$$

$$\begin{aligned} \Lambda &= \Lambda_1 + \Lambda_2 = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \Lambda_2 & 0 \\ 0 & 0 \end{pmatrix} \\ \gamma &= \gamma_1 + \gamma_2 \end{aligned}$$

Marginalizing in canonical form

- Recall conversion formulas

- $\mu = \Lambda^{-1} \eta$
- $\Sigma = \Lambda^{-1}$

- Marginal distribution

$$\eta_A^m = \eta_A - \Lambda_{AB} \Lambda_{BB}^{-1} \eta_B$$

$$\Lambda_{AA}^m = \Lambda_{AA} - \Lambda_{AB} \Lambda_{BB}^{-1} \Lambda_{BA}$$

Variable elimination

$$\begin{aligned} P(X_1, \dots, X_m) &= P(X_1) \underset{\lambda_1}{P(X_2 | X_1)} \dots \underset{\lambda_{m-1}}{P(X_m | X_{m-1})} \\ &= \mathcal{N}(\cdot; \Lambda, \gamma) \end{aligned}$$

$$\Lambda = \begin{pmatrix} \Lambda_1 & & & \\ & \ddots & & \\ & & \Lambda_2 & \\ & & & \ddots & \\ & & & & \Lambda_m \end{pmatrix}.$$

block diagonal

- In Gaussian Markov Networks, Variable elimination = Gaussian elimination (fast for low bandwidth = low treewidth matrices)

Tasks

- Read Koller & Friedman Chapters 4.6.1, 8.1-8.3

Probabilistic Graphical Models

Lecture 12 – Dynamical Models

CS/CNS/EE 155
Andreas Krause

Announcements

- Homework 3 out tonight
 - Start early!!
- Project milestones due today
 - Please email to TAs

Parameter learning for log-linear models

- Feature functions $\phi_i(C_i)$ defined over cliques
- Log linear model over undirected graph G
 - Feature functions $\phi_1(C_1), \dots, \phi_k(C_k)$
 - Domains C_i can overlap
- Joint distribution

$$P(X_1, \dots, X_n) = \frac{1}{Z} \exp\left(\underbrace{\sum_i w_i^T \phi_i(C_i)}\right)$$

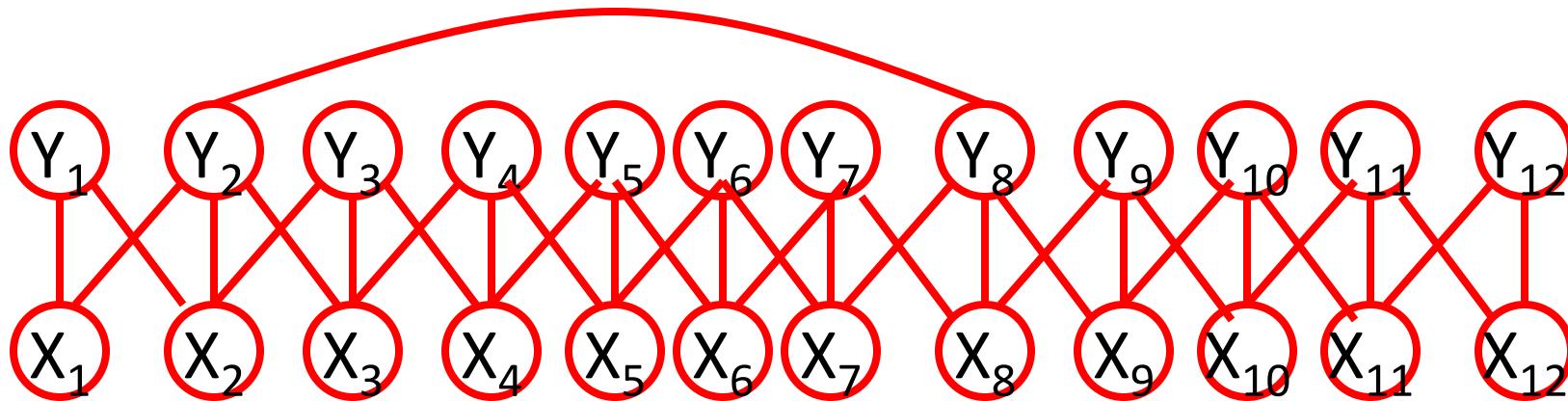
- How do we get weights w_i ?

Log-linear conditional random field

- Define log-linear model over outputs Y
 - No assumptions about inputs X
- Feature functions $\phi_i(C_i, x)$ defined over cliques and inputs
 $C_i \subseteq Y$
- Joint distribution

$$P(Y_1, \dots, Y_n \mid x) = \frac{1}{Z(x_{\text{clf}})} \exp\left(\sum_i w_i^T \phi_i(C_i, x)\right)$$

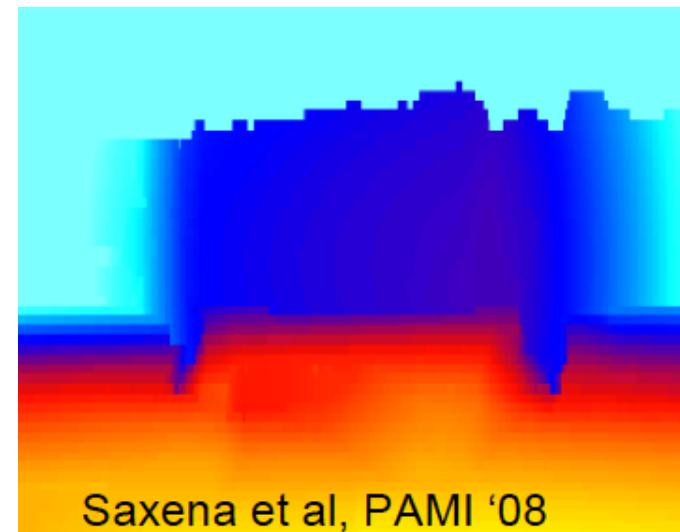
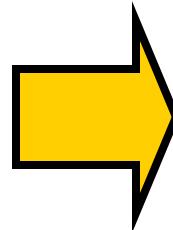
Example: CRFs in NLP



Mrs. Greene spoke today in New York. Green chairs the finance committee

- Classify into Person, Location or Other

Example: CRFs in vision



Gradient of conditional log-likelihood

- Partial derivative

$$\frac{\partial \log P(\mathcal{D}_Y \mid w, \mathcal{D}_X)}{\partial w_i} = \sum_j \left[\phi_i(\mathbf{c}_i^{(j)}, x^{(j)}) + \underbrace{\sum_{\mathbf{c}_i} P(\mathbf{c}_i \mid w, \underline{x^{(j)}}) \phi_i(\mathbf{c}_i, \underline{x^{(j)}})}_{\text{Reg. Inference}} \right]$$

- Requires one inference per feature and per data point

Can be very expensive

Can do "pseudo" likelihood est. (approximate)

- Can optimize using conjugate gradient

Exponential Family Distributions

- Distributions for log-linear models

$$P(X_1, \dots, X_n) = \frac{1}{Z} \exp\left(\sum_i w_i^T \phi_i(C_i)\right)$$

- More generally: **Exponential family distributions**

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- $h(x)$: Base measure \leftarrow Often constant
- w : natural parameters
- $\phi(x)$: Sufficient statistics
- $A(w)$: log-partition function $\leftarrow A(w) = \log \tau(w)$
- Hereby x can be continuous (defined over any set)

Examples

- Exp. Family:

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- Gaussian distribution

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$h(x) = \frac{1}{\sqrt{2\pi}} \quad -\frac{x^2}{2\sigma^2} + \frac{x\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2}$$

$$\phi(x) = \left[-\frac{x^2}{2}, x\right], \quad A(w) = \frac{\mu^2}{2\sigma^2} - \log \sigma$$

$$w = \left[\frac{1}{\sigma^2}, \frac{\mu}{\sigma^2}\right]$$

$h(x)$: Base measure

w : natural parameters

$\phi(x)$: Sufficient statistics

$A(w)$: log-partition function

- Other examples: Multinomial, Poisson, Exponential, Gamma, Weibull, chi-square, Dirichlet, Geometric, ...

Moments and gradients

$$P(x) = h(x) \exp(w^T \phi(x) - A(w))$$

- Correspondence between moments and log-partition function (just like in log-linear models)

$$\frac{\partial A(w)}{\partial w_i} = \int p(x \mid w) \phi_i(x) dx = \mathbb{E}[\phi_i \mid w]$$

$$\frac{\partial^2 A(w)}{\partial w_i \partial w_j} = Cov(\phi_i, \phi_j \mid w)$$

- Can compute moments from derivatives, and derivatives from moments!
- MLE \Leftrightarrow moment matching

Conjugate priors in Exponential Family

$$P(x | w) = h(x) \exp(w^T \phi(x) - A(w))$$

Any exponential family likelihood has a conjugate prior

$$P(w|\alpha, \beta) = \exp(\underline{\alpha}^T w - \underline{\beta} A(w) - \underline{B(\alpha, \beta)})$$

$$P(x|w) P(w|\alpha, \beta) \propto \exp(w^T (\phi(x) + \alpha) - (\beta + 1) A(w))$$

Exponential family graphical models

- So far, only defined graphical models over discrete variables.
- Can define GMs over continuous distributions!
- For exponential family distributions:

$$p(X_1, \dots, X_n) = \prod_i h_i(C_i) \exp\left(\sum_i w_i^T \phi_i(C_i) - A(w)\right)$$
$$\exp A(w) = \int \int \dots \int \prod_i h_i(C_i) \exp\left(\sum_i w_i^T \phi_i(C_i) - A(w)\right) dx_1 \dots dx_n$$

- Can do much of what we discussed (VE, JT, parameter learning, etc.) for such exponential family models
- Important example: **Gaussian Networks**

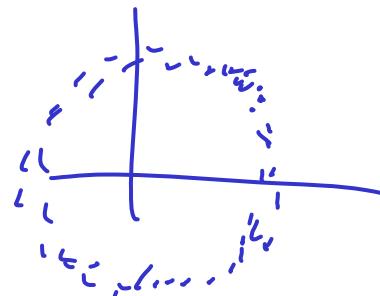
Multivariate Gaussian distribution

$$\mathcal{N}(x; \Sigma, \mu) = \frac{1}{(2\pi)^{n/2} \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \vdots & & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{pmatrix} \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}$$

$\Sigma^\top = \Sigma$
pos. det.: $x^\top \Sigma x \geq 0 \quad \forall x$

- Joint distribution over n random variables $P(X_1, \dots, X_n)$
- $\sigma_{jk} = E[(X_j - \mu_j)(X_k - \mu_k)] = \text{Cov}(X_j, X_k)$
- X_j and X_k independent $\Leftrightarrow \sigma_{jk}=0$



Marginalization

- Suppose $(X_1, \dots, X_n) \sim N(\mu, \Sigma)$

- What is $P(X_1)??$

$$P(X_1) = \underbrace{\int \int \int \dots}_{\text{red line}} p(x_1, \dots, x_n) dx_2 \dots dx_n$$

$$P(X_1) = \mathcal{N}(x_1; \mu_1, \sigma_1^2)$$

- More generally: Let $A = \{i_1, \dots, i_k\} \subseteq \{1, \dots, N\}$
- Write $X_A = (X_{i_1}, \dots, X_{i_k})$

- $X_A \sim N(\mu_A, \Sigma_{AA})$

$$\Sigma_{AA} = \begin{pmatrix} \sigma_{i_1}^2 & \sigma_{i_1 i_2} & \dots & \sigma_{i_1 i_k} \\ \vdots & & & \vdots \\ \sigma_{i_k i_1} & \sigma_{i_k i_2} & \dots & \sigma_{i_k}^2 \end{pmatrix}$$

$$\mu_A = \begin{pmatrix} \mu_{i_1} \\ \mu_{i_2} \\ \vdots \\ \mu_{i_k} \end{pmatrix}$$

$$\mu_A = \begin{pmatrix} \mu_{i_1} \\ \mu_{i_2} \\ \vdots \\ \mu_{i_k} \end{pmatrix}$$

Conditioning

- Suppose $(X_1, \dots, X_n) \sim N(\mu, \Sigma)$
- Decompose as (X_A, X_B)
- What is $P(X_A | X_B)??$

$$\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}$$

- $P(X_A = x_A | X_B = x_B) = N(x_A; \mu_{A|B}, \Sigma_{A|B})$ where

$$\mu_{A|B} = \underline{\mu_A} + \underbrace{\Sigma_{AB} \Sigma_{BB}^{-1}}_W (x_B - \mu_B)$$

$$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB} \Sigma_{BB}^{-1} \Sigma_{BA}$$

- Computable using linear algebra!

Does not depend on x_B

Conditional linear Gaussians

$$\mu_{A|B} = \mu_A + \underbrace{\Sigma_{AB}\Sigma_{BB}^{-1}}_w(x_B - \mu_B)$$

$$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$$

$$x_{A|B} = \mu_A + w x_B - w \mu_B + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \Sigma_{A|B})$$

$$= \underbrace{w x_B}_{} + b + \varepsilon \leftarrow \text{noise}$$

$$w = \Sigma_{AB}\Sigma_{BB}^{-1}$$

$$b = \mu_A - w \mu_B$$

Canonical Representation

$$p(X_1, \dots, X_n) = \frac{1}{(2\pi)^{n/2} \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$
$$\propto \exp(\eta^T x - \frac{1}{2} x^T \underline{\Lambda} x) = \exp \left(\sum_i \eta_i x_i - \frac{1}{2} \sum_{i,j} x_i x_j \underline{\Lambda}_{ij} \right)$$

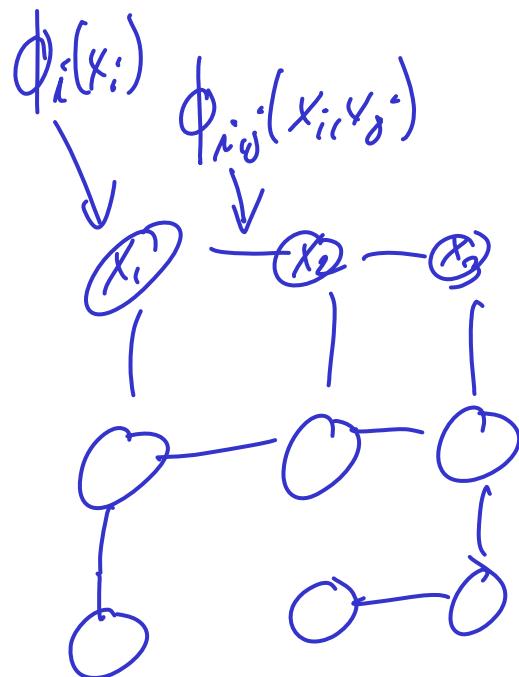
- Multivariate Gaussians in exponential family!
- Standard vs canonical form:

$$\mu = \underline{\Lambda}^{-1} \eta$$

$$\Sigma = \underline{\Lambda}^{-1}$$

Gaussian Networks

$$p(X_1, \dots, X_n) \propto \exp\left(-\frac{1}{2} \sum_{i,j} \lambda_{i,j} x_i x_j + \sum_i \eta_i x_i\right)$$



$$= \exp\left(\sum_{i,j} \lambda_{i,j} \phi_{i,j}(x_i, x_j) + \sum_i \eta_i \phi_i(x_i) \right)$$

$$\phi_{i,j}(x_i, x_j) = -\frac{1}{2} x_i x_j \quad \phi_i(x_i) = x_i$$

No edge between x_i, x_j

$$\Leftrightarrow \lambda_{ij} = 0$$

Zeros in precision matrix Λ indicate missing edges in log-linear model!

Inference in Gaussian Networks

- Can compute marginal distributions in $O(n^3)$!
- For large numbers n of variables, still intractable
- If Gaussian Network has low treewidth, can use variable elimination / JT inference!
- Need to be able to multiply and marginalize factors!

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

Multiplying factors in Gaussians

$$P(X_A) = \mathcal{N}(x_A; \Lambda_1, \gamma_1) \quad \begin{matrix} |A|=k, |B|=m \\ \Lambda_1 \in \mathbb{R}^{k \times k} \end{matrix}$$

$$P(X_B | X_A) = \mathcal{N}(x_B | x_A; \Lambda_2, \gamma_2) \leftarrow \begin{matrix} \text{CLG representation} \\ \Lambda_2 \in \mathbb{R}^{m \times m} \end{matrix}$$

$$P(X_A, X_B) = P(X_A) P(X_B | X_A)$$

$$\propto \exp(x_A^\top \Lambda_1 x_A + \gamma_1^\top x_A) \exp((x_A, x_B)^\top \Lambda_2 (x_A, x_B) + \gamma_2^\top) \\ = \mathcal{N}(x_A, x_B; \Lambda, \gamma)$$

$$\Lambda = \Lambda_1 + \Lambda_2 = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \Lambda_2 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\gamma = \gamma_1 + \gamma_2$$

Conditioning in canonical form

- Joint distribution $(X_A, X_B) \sim N(\eta_{AB}, \Lambda_{AB})$
- Conditioning: $P(X_A \mid X_B = x_B) = N(x_A; \eta_{A|B=x_B}, \Lambda_{A|B=x_B})$

$$\begin{matrix} \Lambda_{AA} & \Lambda_{AB} \\ \Lambda_{BA} & \Lambda_{BB} \end{matrix}$$

$$\eta_{A|B=x_B} = \eta_A - \Lambda_{AB}x_B$$

$$\underline{\Lambda_{A|B=x_B}} = \underline{\Lambda_{AA}}$$

Marginalizing in canonical form

- Recall conversion formulas

- $\mu = \Lambda^{-1} \eta$
- $\Sigma = \Lambda^{-1}$

- Marginal distribution

$$\eta_A^m = \eta_A - \Lambda_{AB} \Lambda_{BB}^{-1} \eta_B$$

$$\Lambda_{AA}^m = \Lambda_{AA} - \Lambda_{AB} \Lambda_{BB}^{-1} \Lambda_{BA}$$

Standard vs. canonical form

Standard form

Marginalization

$$\underline{\mu_A^m = \mu_A}$$

$$\underline{\Sigma_{AA}^m = \Sigma_{AA}}$$

Canonical form

$$\eta_A^m = \eta_A - \Lambda_{AB}\Lambda_{BB}^{-1}\eta_B$$

$$\Lambda_{AA}^m = \Lambda_{AA} - \Lambda_{AB}\Lambda_{BB}^{-1}\Lambda_{BA}$$

Conditioning

$$\mu_{A|B=x_B} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B)$$

$$\Sigma_{A|B=x_B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$$

$$\eta_{A|B=x_B} = \eta_A - \Lambda_{AB}x_B$$

$$\Lambda_{A|B=x_B} = \underline{\Lambda_{AA}}$$

- In standard form, marginalization is easy
- In canonical form, conditioning is easy!

Variable elimination

$$P(X_1, \dots, X_m) = P_{\substack{X_1 \\ \vdots \\ X_m}}(X_1) P_{\substack{X_2 \\ \vdots \\ X_m}}(X_2 | X_1) \dots P_{\substack{X_m \\ \vdots \\ X_m}}(X_m | X_{m-1})$$

$\circlearrowleft \rightarrow \circlearrowleft \rightarrow \circlearrowleft \rightarrow \circlearrowleft$

$$= \mathcal{N}(\cdot; \Lambda, \gamma)$$

$$\Lambda = \begin{pmatrix} \Lambda_1 & & & \\ & \ddots & & \\ & & \Lambda_2 & \\ & & & \ddots & \\ & & & & \Lambda_n \end{pmatrix}$$

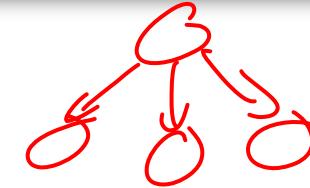
block diagonal

- In Gaussian Markov Networks, Variable elimination = Gaussian elimination (fast for low bandwidth = low treewidth matrices)

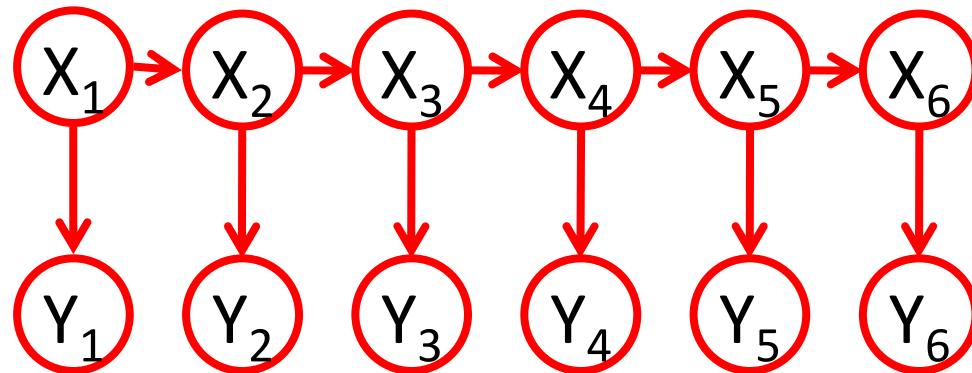
Dynamical models

HMMs / Kalman Filters

- Most famous Graphical models:
 - Naïve Bayes model
 - Hidden Markov model
 - Kalman Filter
- Hidden Markov models
 - Speech recognition
 - Sequence analysis in comp. bio
- Kalman Filters control
 - Cruise control in cars
 - GPS navigation devices
 - Tracking missiles..
- Very simple models but very powerful!!

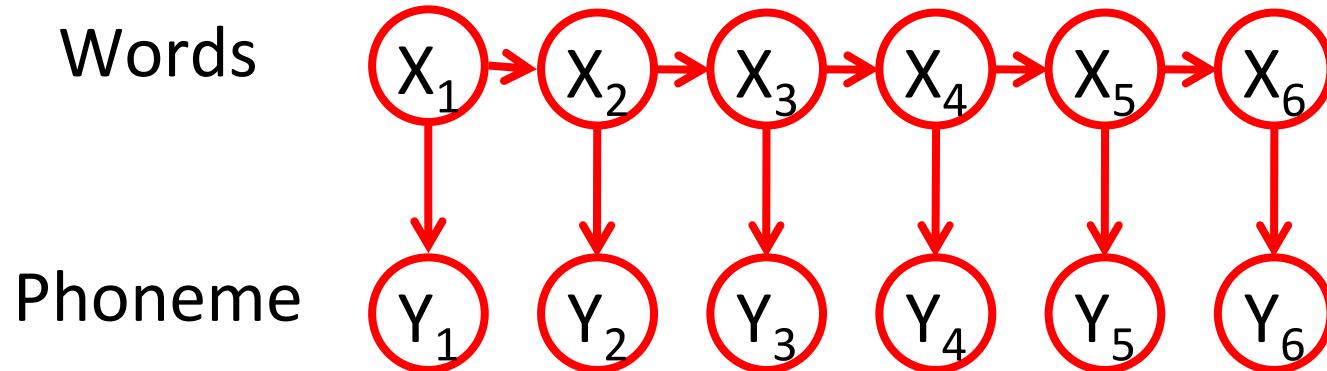


HMMs / Kalman Filters



- X_1, \dots, X_T : Unobserved (hidden) variables
- Y_1, \dots, Y_T : Observations
- HMMs: X_i Multinomial, Y_i arbitrary
- Kalman Filters: X_i, Y_i Gaussian distributions
 - Non-linear KF: X_i Gaussian, Y_i arbitrary

HMMs for speech recognition

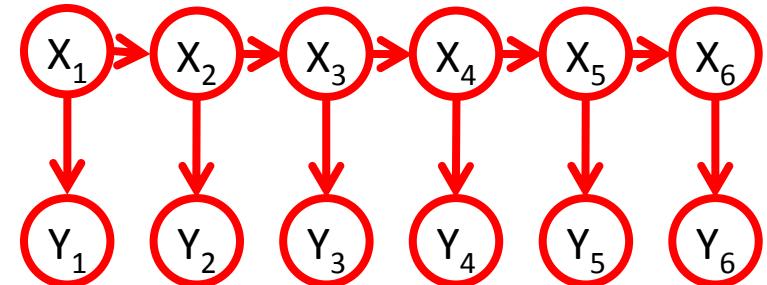


“He ate the cookies on the couch”

- Infer spoken words from audio signals

Hidden Markov Models

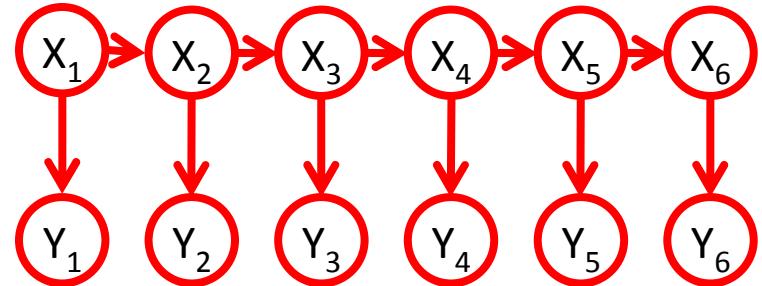
- Inference:
 - In principle, can use VE, JT etc.
 - New variables X_t, Y_t at each time step → need to rerun



- Bayesian Filtering:
 - Suppose we already have computed $P(X_t | y_{1,\dots,t})$
 - Want to efficiently compute $P(X_{t+1} | y_{1,\dots,t+1})$

Bayesian filtering

- Start with $P(X_1)$
- At time t
 - Assume we have $P(X_t | y_{1..t-1})$
 - Condition: $P(X_t | y_{1..t})$



$$P(X_t | y_{1..t}) \propto P(X_t | y_{1..t-1}) \underbrace{P(Y_t | X_t, y_{1..t-1})}_{\text{cond. ind. } P(Y_t | X_t)}$$

- Prediction: $P(X_{t+1}, X_t | y_{1..t})$

$$P(X_{t+1}, X_t | y_{1..t}) = P(X_t | y_{1..t}) \cdot \underbrace{P(X_{t+1} | X_t, y_{1..t})}_{= P(X_{t+1} | X_t)} \quad \text{"Roll up"}$$

- Marginalization: $P(X_{t+1} | y_{1..t})$

$$P(X_{t+1} | y_{1..t}) = \sum_{X_K} P(X_{t+1}, X_t | y_{1..t})$$

Parameter learning in HMMs

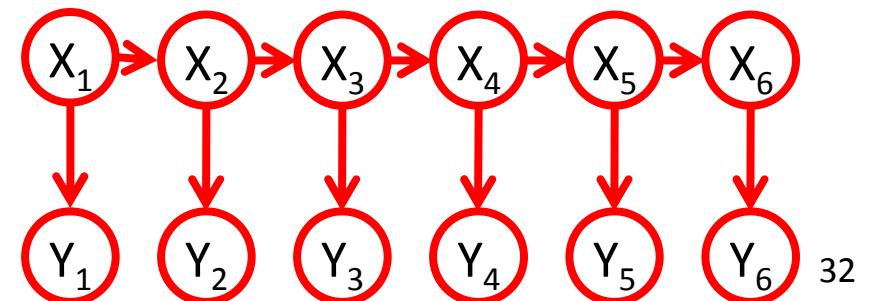
- Assume we have labels for hidden variables
- Assume stationarity
 - $P(X_{t+1} | X_t)$ is same over all time steps
 - $P(Y_t | X_t)$ is same over all time steps
 - Violates parameter independence (\rightarrow parameter “sharing”)
- Example: compute parameters for $P(X_{t+1}=x | X_t=x')$

$$\begin{aligned}\log P(x_1, \dots, x_T, y_1, \dots, y_T | \theta) &= \log P(x_1) \prod_{t=2}^T P(x_t | x_{t-1}, \theta_1) \prod_{t=1}^T P(y_t | x_t, \theta_2) \\ &= \log P(x_1 | \theta_1) + \sum_t \log P(x_t | x_{t-1}, \theta_1) + \sum_t \log P(y_t | x_t, \theta_2) \\ \theta_{x_t=x | x_{t-1}=x'} &= \frac{\text{Count}(x', x)}{T-1}\end{aligned}$$

- What if we don't have labels for hidden vars?
 \rightarrow Use EM (later this course)

Kalman Filters (Gaussian HMMs)

- X_1, \dots, X_T : Location of object being tracked
- Y_1, \dots, Y_T : Observations
- $P(X_1)$: Prior belief about location at time 1
- $P(X_{t+1} | X_t)$: “Motion model”
 - How do I expect my target to move in the environment?
 - Represented as CLG: $X_{t+1} = A X_t + N(0, \Sigma_M)$
- $P(Y_t | X_t)$: “Sensor model”
 - What do I observe if target is at location X_t ?
 - Represented as CLG: $Y_t = H X_t + N(0, \Sigma_O)$

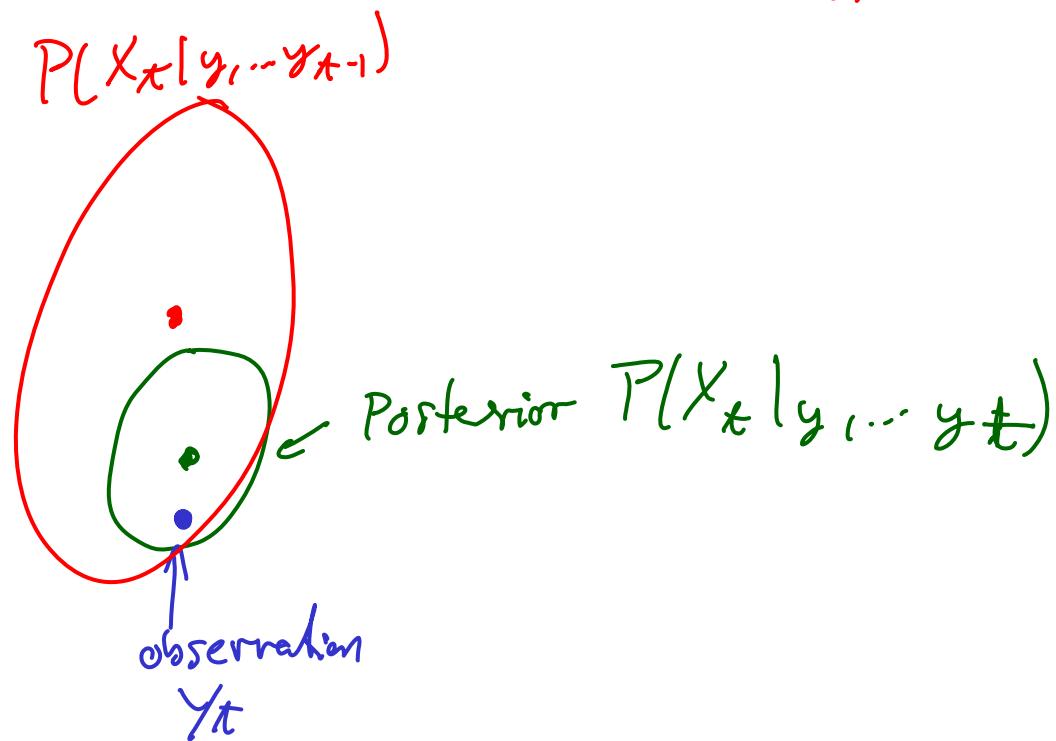


Understanding Motion model

$$x_t = \begin{pmatrix} l_t \\ v_t \end{pmatrix}$$
$$x_{t+1} = Ax_t + \text{noise}$$
$$A = \begin{pmatrix} 1 & dt \\ 0 & 1 \end{pmatrix}$$
$$P(x_1) \quad P(x_2) \quad P(x_3) \quad P(x_4)$$
$$= \int P(x_2|x_1) P(x_1) dx_1$$

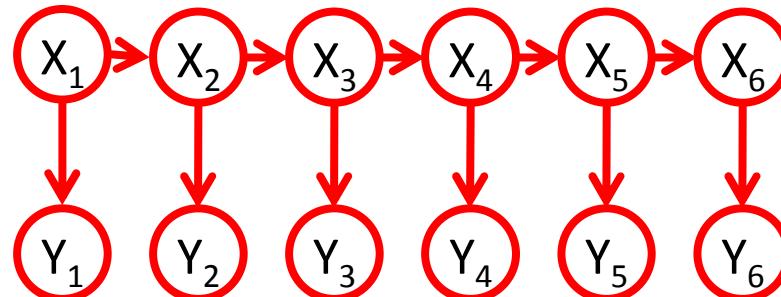
Understanding sensor model

$$X_t = \begin{pmatrix} L_t \\ V_t \end{pmatrix} \quad Y_t = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{\text{only obs. location}} X_t + \text{noise}$$



Bayesian Filtering for KFs

- Can use Gaussian elimination to perform inference in “unrolled” model



- Start with prior belief $P(X_1)$
- At every timestep have belief $P(X_t | y_{1:t-1})$
 - Condition on observation: $P(X_t | y_{1:t}) \xleftarrow{\text{Multiply likelihood}} P(X_t | y_{1:t-1})$ "sensor model"
 - Predict (multiply motion model): $P(X_{t+1}, X_t | y_{1:t}) \xleftarrow{\text{Multiply motion model}} P(X_{t+1} | X_t, y_{1:t})$
 - “Roll-up” (marginalize prev. time): $P(X_{t+1} | y_{1:t})$

$$\mathcal{N}(x_t; \gamma_t, \Lambda_t)$$

Implementation

- Current belief: $P(x_t | y_{1:t-1}) = \mathcal{N}(x_t; \eta_{x_t}, \Lambda_{x_t})$
- Multiply sensor and motion model

Motion model
 $P(x_{t+1} | x_t) \leftarrow$ Represented as CLG, canonical params
 Λ_m, η_m

$$\begin{aligned} P(x_{t+1}, x_t | y_{1..t}) &= P(x_t | y_{1..t}) P(x_{t+1} | x_t) \\ &= \mathcal{N}(\cdot; \eta_{x_t} + \eta_m, \Lambda_{x_t} + \Lambda_m) \end{aligned}$$

- Marginalize $P(x_{t+1} | y_{1..t}) = \mathcal{N}(\cdot; \eta_A^m, \Lambda_{AA}^m)$

$$\eta_A^m = \eta_A - \Lambda_{AB} \Lambda_{BB}^{-1} \eta_B$$

$$A = x_{t+1} | y_{1..t}$$

$$\Lambda_{AA}^m = \Lambda_{AA} - \Lambda_{AB} \Lambda_{BB}^{-1} \Lambda_{BA}$$

$$B = x_t | y_{1..t}$$

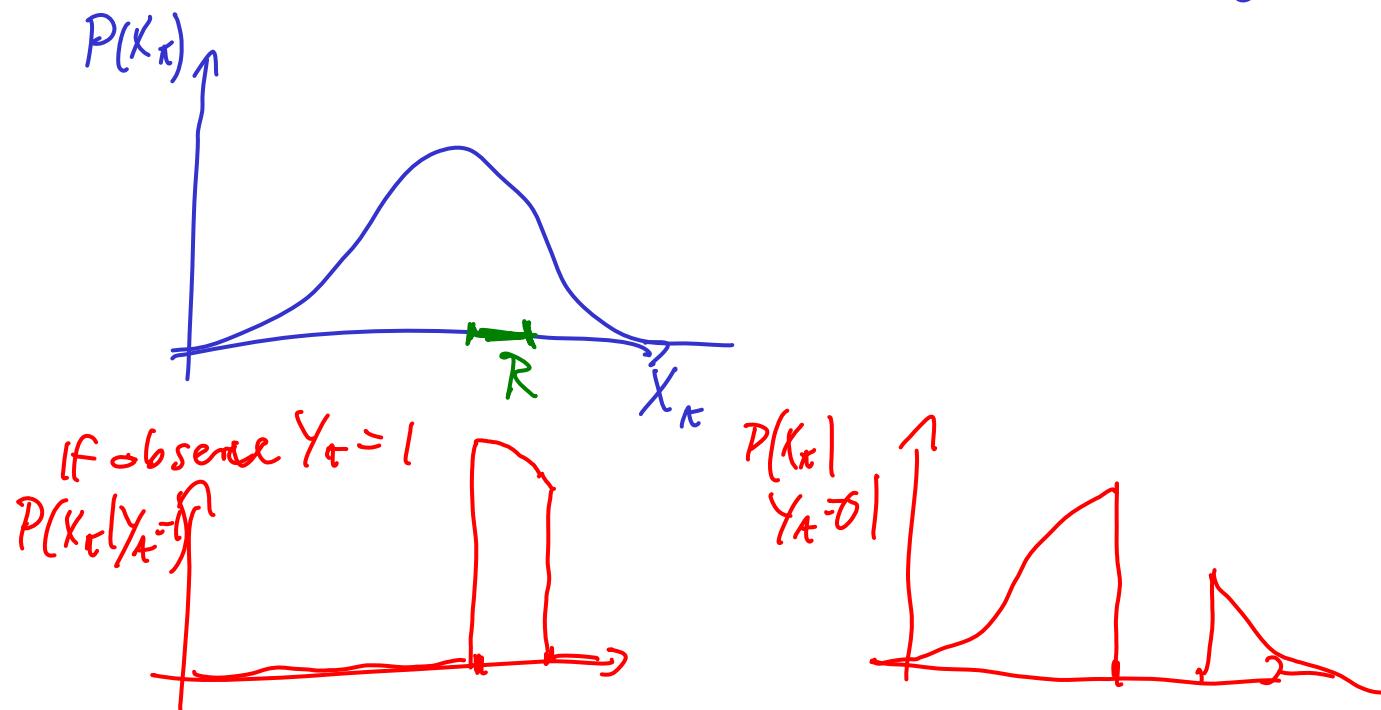
What if observations not “linear”?

- Linear observations:

- $Y_t = H X_t + \text{noise}$

- Nonlinear observations:

“Motion detector”: $Y_t = 1 \text{ if } X_t \in R$
 $= 0 \text{ otherwise}$



Incorporating Non-gaussian observations

- Nonlinear observation $\rightarrow P(Y_t | X_t)$ not Gaussian
- First approach: Approximate $P(Y_t | X_t)$ as CLG
 - Linearize $P(Y_t | X_t)$ around current estimate $E[X_t | y_{1..t-1}]$
 - Known as Extended Kalman Filter (EKF)
 - Can perform poorly if $P(Y_t | X_t)$ highly nonlinear
- Second approach: Approximate $P(Y_t, X_t)$ as Gaussian
 - Takes correlation in X_t into account
 - After obtaining approximation, condition on $Y_t=y_t$ (now a “linear” observation)

Finding Gaussian approximations

- Need to find Gaussian approximation of $P(X_t, Y_t)$
- How?
 - Gaussians in Exponential Family \rightarrow Moment matching!!

- $E[Y_t] = \int y P(y_t) dy_t = \int y \underbrace{P(y_t | x_t)}_{\text{nonlinear}} \underbrace{P(x_t)}_{\text{Gaussian}} dx_t dy_t$
- $E[Y_t^2] = \int y^2 P(y_t | x_t) P(x_t) dx_t dy_t$
- $E[X_t Y_t] = \int x y P(y_t | x_t) P(x_t) dx_t dy_t$

Linearization by integration

- Need to integrate product of Gaussian with arbitrary function
- Can do that by numerical integration
 - Approximate integral as weighted sum of evaluation points

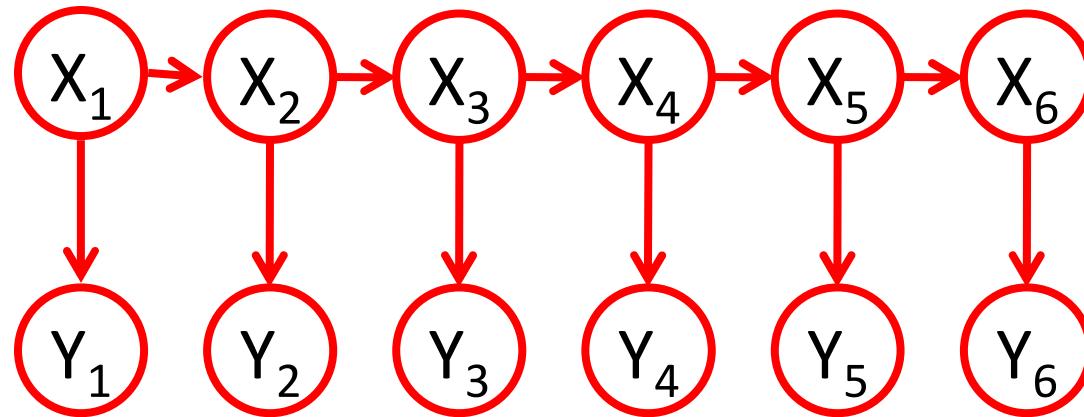
$$\int f(x,y) \rho(x) dx dy \approx \sum_i w_i f(x^{(i)}, y^{(i)})$$

↑ How should we choose ?

- Gaussian quadrature defines locations and weights of points
- For 1 dim: **Exact** for polynomials of degree D if choosing 2D points using Gaussian quadrature
- For higher dimensions: Need exponentially many points to achieve exact evaluation for polynomials
- Application of this is known as “Unscented” Kalman Filter (UKF)

Factored dynamical models

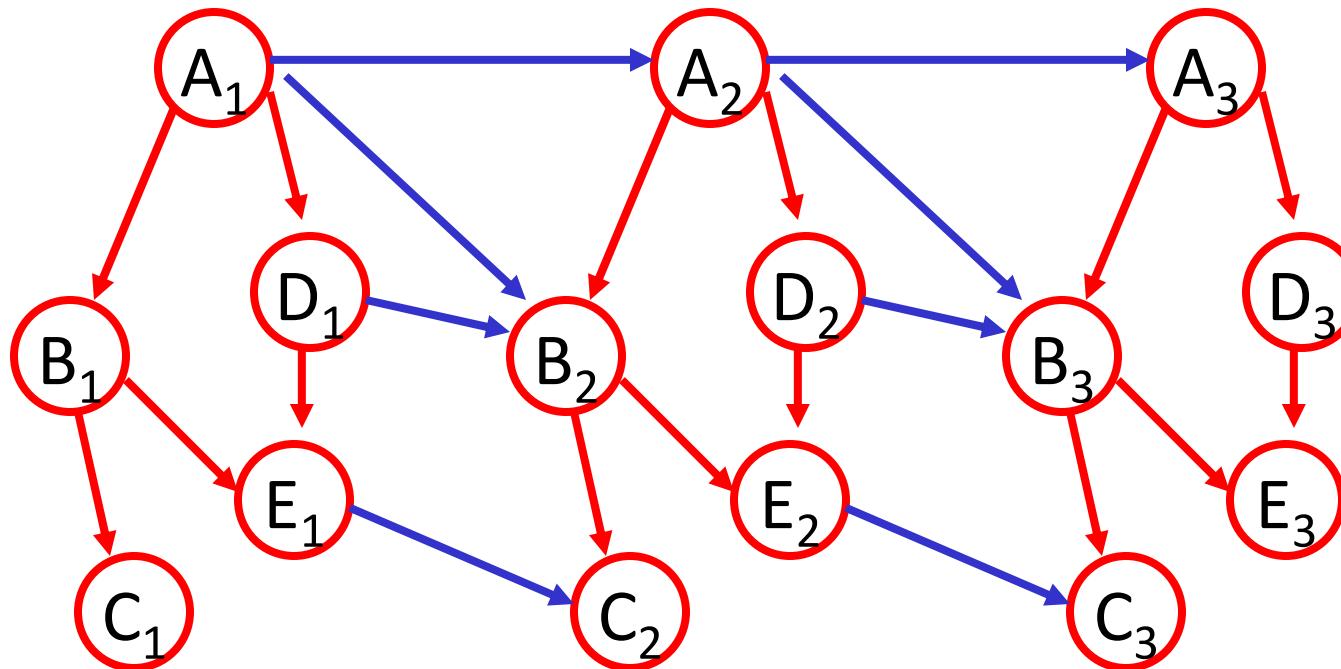
- So far: HMMs and Kalman filters



- What if we have more than one variable at each time step?
 - E.g., temperature at different locations, or road conditions in a road network?
 - ➔ Spatio-temporal models

Dynamic Bayesian Networks

- At every timestep have a Bayesian Network



- Variables at each time step t called a “slice” S_t
- “Temporal” edges connecting S_{t+1} with S_t

Tasks

- Read Koller & Friedman Chapters 6.2.3, 15.1

Probabilistic Graphical Models

Lecture 13 – Loopy Belief Propagation

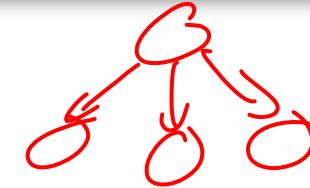
CS/CNS/EE 155
Andreas Krause

Announcements

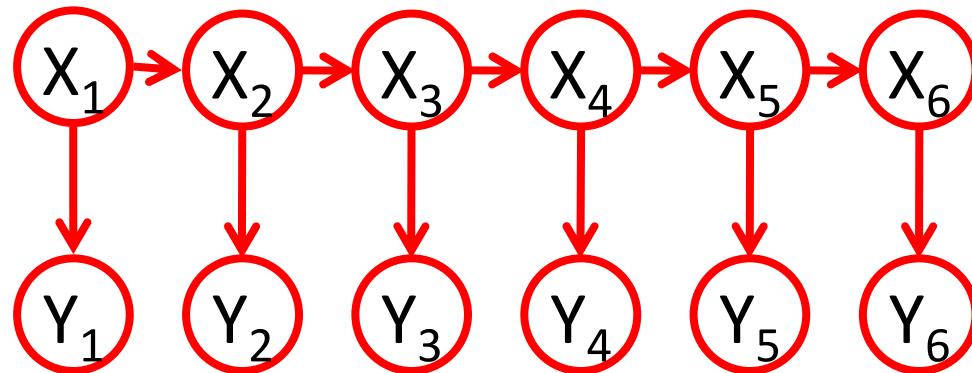
- Homework 3 out
 - Lighter problem set to allow more time for project
- Next Monday: Guest lecture by **Dr. Baback Moghaddam from the JPL Machine Learning Group**
- PLEASE fill out feedback forms
 - This is a new course
 - Your feedback can have major impact in future offerings!!

HMMs / Kalman Filters

- Most famous Graphical models:
 - Naïve Bayes model
 - Hidden Markov model
 - Kalman Filter
- Hidden Markov models
 - Speech recognition
 - Sequence analysis in comp. bio
- Kalman Filters control
 - Cruise control in cars
 - GPS navigation devices
 - Tracking missiles..
- Very simple models but very powerful!!



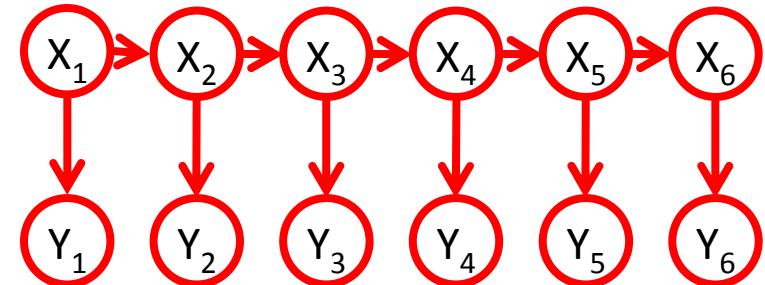
HMMs / Kalman Filters



- X_1, \dots, X_T : Unobserved (hidden) variables
- Y_1, \dots, Y_T : Observations
- HMMs: X_i Multinomial, Y_i arbitrary
- Kalman Filters: X_i, Y_i Gaussian distributions
 - Non-linear KF: X_i Gaussian, Y_i arbitrary

Hidden Markov Models

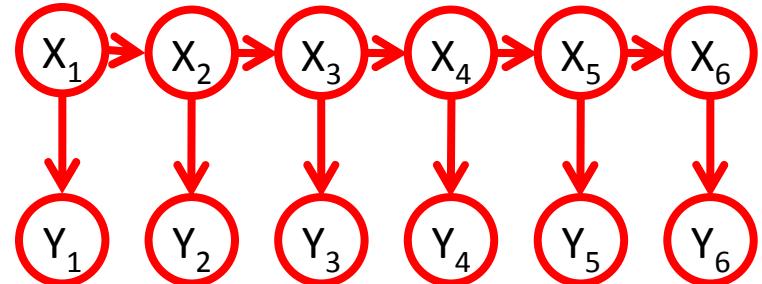
- Inference:
 - In principle, can use VE, JT etc.
 - New variables X_t, Y_t at each time step → need to rerun



- Bayesian Filtering:
 - Suppose we already have computed $P(X_t | y_{1,\dots,t})$
 - Want to efficiently compute $P(X_{t+1} | y_{1,\dots,t+1})$

Bayesian filtering

- Start with $P(X_1)$
- At time t
 - Assume we have $P(X_t | y_{1..t-1})$
 - Condition: $P(X_t | y_{1..t})$



$$P(X_t | y_{1..t}) \propto P(X_t | y_{1..t-1}) \underbrace{P(Y_t | X_t, y_{1..t-1})}_{\text{cond. ind. } P(Y_t | X_t)}$$

- Prediction: $P(X_{t+1}, X_t | y_{1..t})$

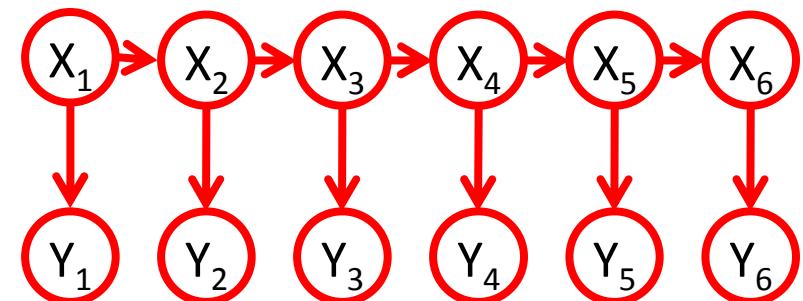
$$P(X_{t+1}, X_t | y_{1..t}) = P(X_t | y_{1..t}) \cdot \underbrace{P(X_{t+1} | X_t, y_{1..t})}_{= P(X_{t+1} | X_t)} \quad \text{"Roll up"}$$

- Marginalization: $P(X_{t+1} | y_{1..t})$

$$P(X_{t+1} | y_{1..t}) = \sum_{X_K} P(X_{t+1}, X_t | y_{1..t})$$

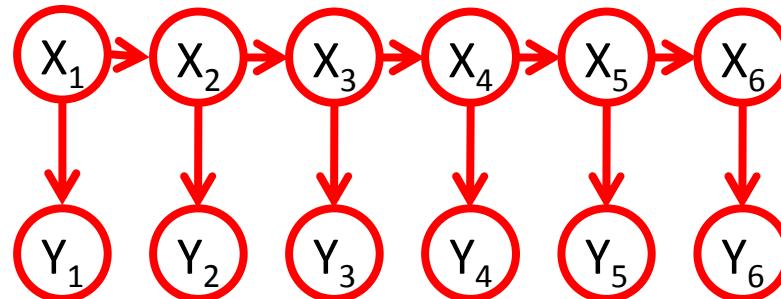
Kalman Filters (Gaussian HMMs)

- X_1, \dots, X_T : Location of object being tracked
- Y_1, \dots, Y_T : Observations
- $P(X_1)$: Prior belief about location at time 1
- $P(X_{t+1} | X_t)$: “Motion model”
 - How do I expect my target to move in the environment?
 - Represented as CLG: $X_{t+1} = A X_t + N(0, \Sigma_M)$
- $P(Y_t | X_t)$: “Sensor model”
 - What do I observe if target is at location X_t ?
 - Represented as CLG: $Y_t = H X_t + N(0, \Sigma_O)$



Bayesian Filtering for KFs

- Can use Gaussian elimination to perform inference in “unrolled” model



- Start with prior belief $P(X_1)$
- At every timestep have belief $P(X_t | y_{1:t-1})$
 - Condition on observation: $P(X_t | y_{1:t})$ *Multiply likelihood*
 - Predict (multiply motion model): $P(X_{t+1}, X_t | y_{1:t})$ *Multiply motion model*
 - “Roll-up” (marginalize prev. time): $P(X_{t+1} | y_{1:t})$

$$\mathcal{N}(x_t; \gamma_t, \Lambda_t)$$

“sensor model”

Multiply likelihood

“motion model”

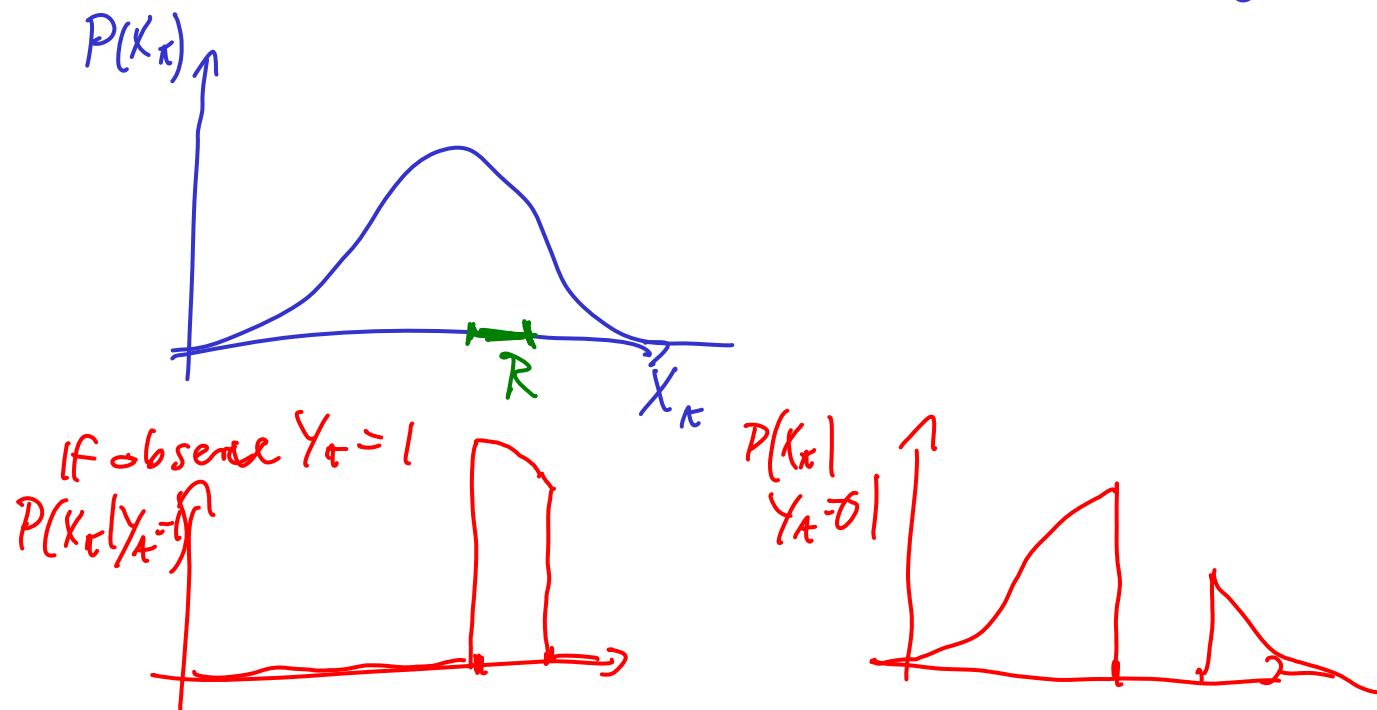
What if observations not “linear”?

- Linear observations:

- $Y_t = H X_t + \text{noise}$

- Nonlinear observations:

“Motion detector”: $Y_t = 1 \text{ if } X_t \in R$
 $= 0 \text{ otherwise}$

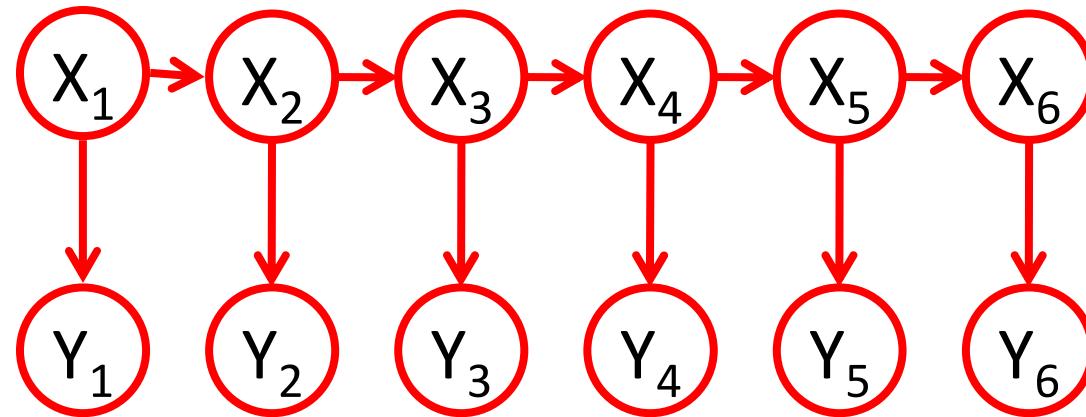


Incorporating Non-gaussian observations

- Nonlinear observation → $P(Y_t | X_t)$ not Gaussian 😞
- Make it Gaussian! ☺
- First approach: Approximate $P(Y_t | X_t)$ as CLG
 - Linearize $P(Y_t | X_t)$ around current estimate $E[X_t | y_{1..t-1}]$
 - Known as Extended Kalman Filter (EKF)
 - Can perform poorly if $P(Y_t | X_t)$ highly nonlinear
- Second approach: Approximate $P(Y_t, X_t)$ as Gaussian
 - Takes correlation in X_t into account
 - After obtaining approximation, condition on $Y_t=y_t$ (now a “linear” observation)

Factored dynamical models

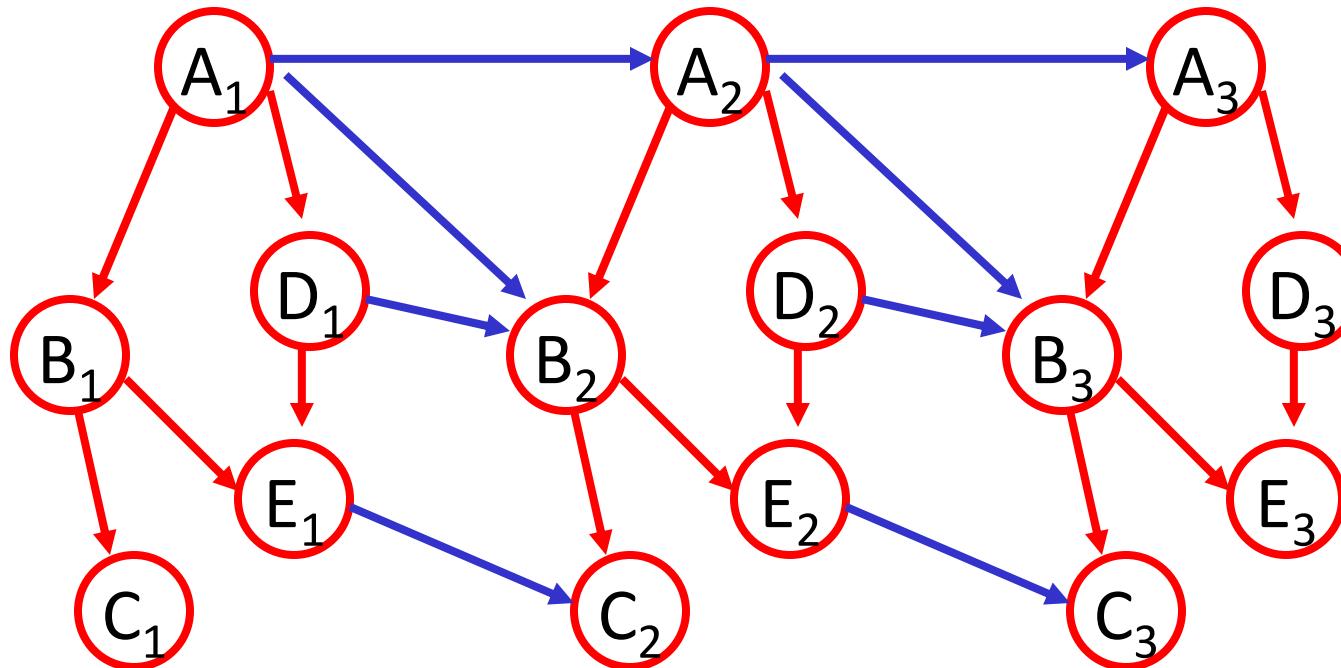
- So far: HMMs and Kalman filters



- What if we have more than one variable at each time step?
 - E.g., temperature at different locations, or road conditions in a road network?
 - ➔ Spatio-temporal models

Dynamic Bayesian Networks

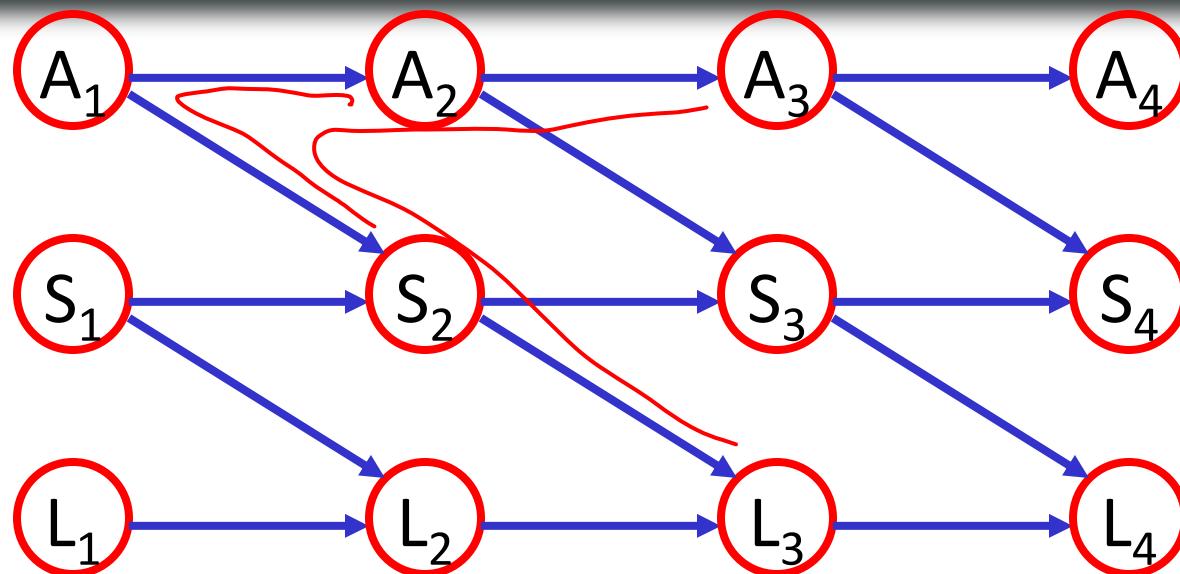
- At every timestep have a Bayesian Network



$$S_t = \{A_1, B_1, \dots, E_t\}$$

- Variables at each time step t called a “slice” S_t
- “Temporal” edges connecting S_{t+1} with S_t

Flow of influence in DBNs



acceleration

speed

location

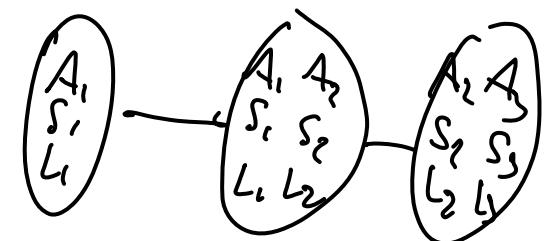
$$A_1 \perp S_1 \checkmark$$

$$A_1 \perp L_1 \checkmark$$

$$A_2 \perp S_2 \times$$

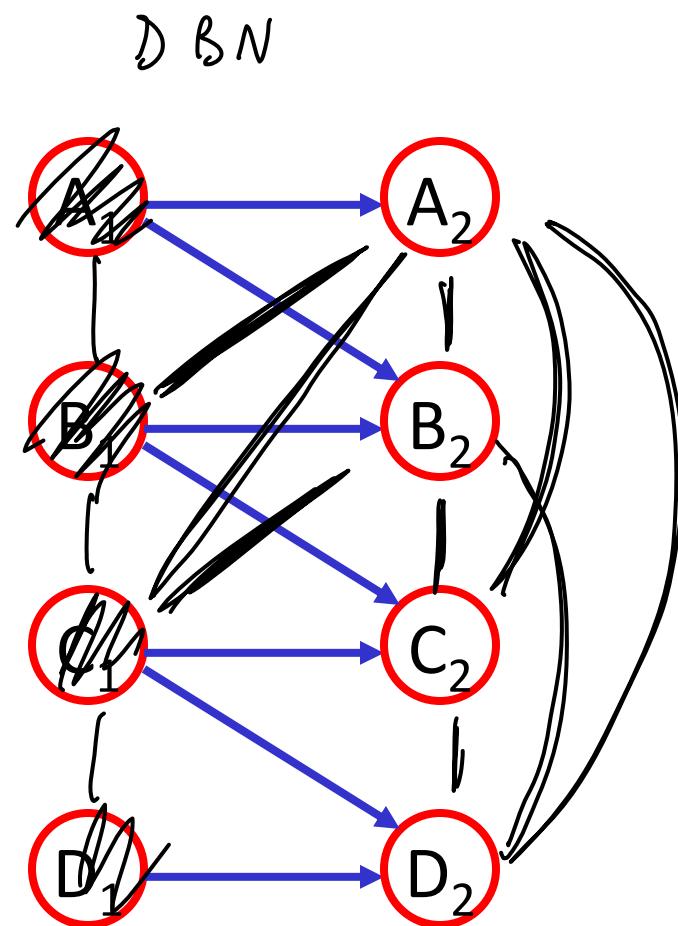
$$A_2 \perp L_2 \checkmark$$

$$A_3 \perp L_3 \times$$



- Can we do efficient filtering in BNs?

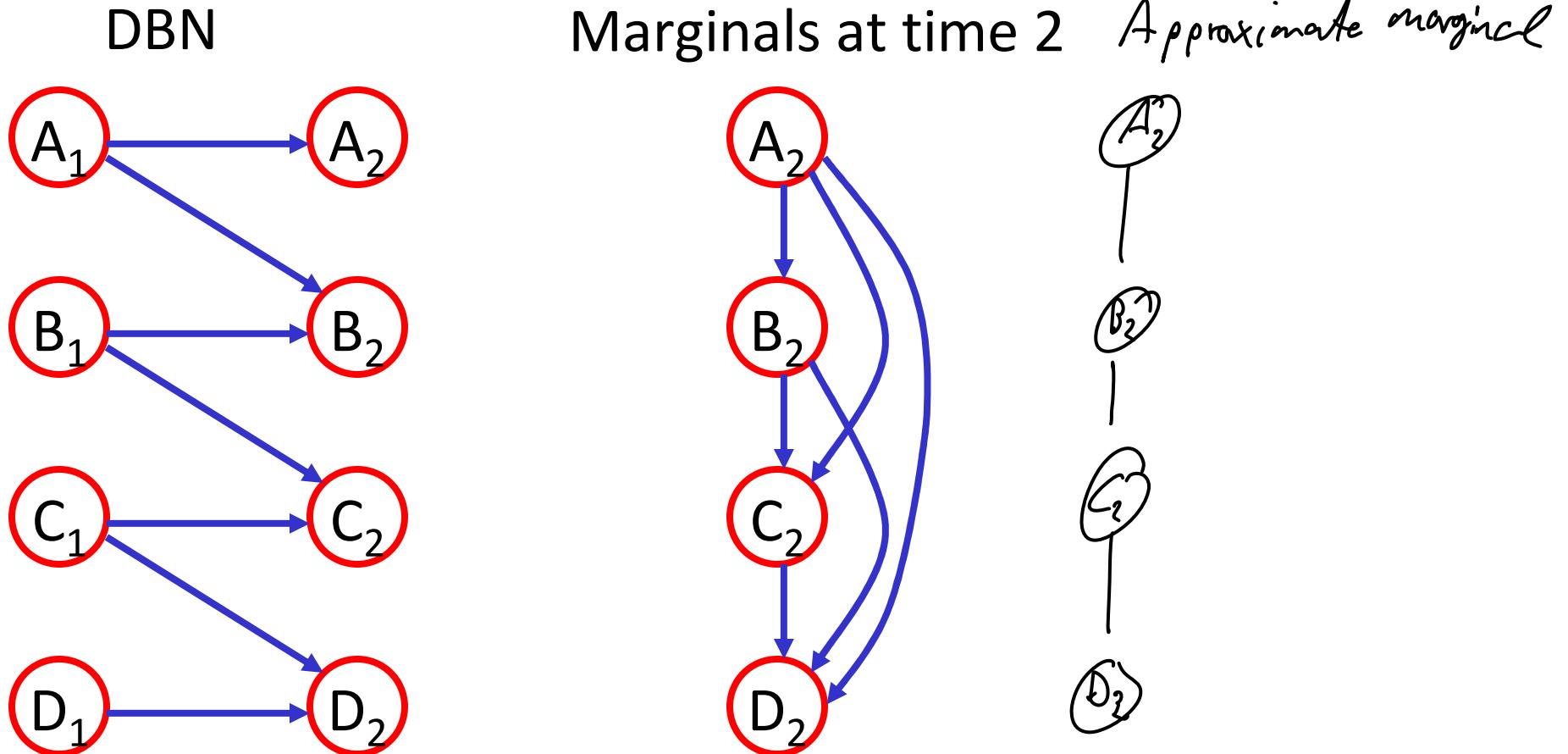
Efficient inference in DBNs?



$$P(A_2, B_2, C_2, D_2)$$

fully connected ::

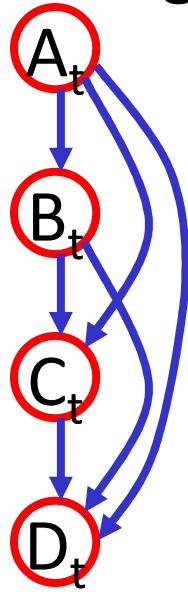
Approximate inference in DBNs?



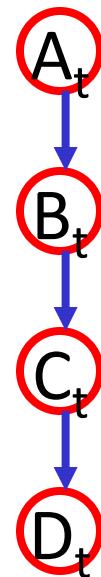
How can we find principled approximations that still allow efficient inference??

Assumed Density Filtering

True marginal



Approximate marginal



Formally:

$$Q^* = \underset{Q}{\operatorname{argmin}} \text{KL}(P \| Q)$$

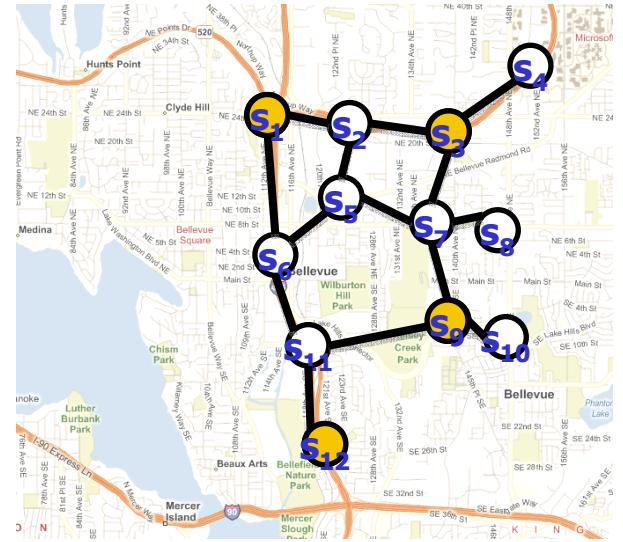
more later

- True marginal $P(X_t)$ fully connected
- Want to find “simpler” distribution $Q(X_t)$ such that $P(X_t) \approx Q(X_t)$
- Optimize over parameters of Q to make Q as “close” to P as possible
- Similar to incorporating non-linear observations in KF!
- More details later (variational inference)!

Big picture summary



represent



States of the world,
sensor measurements, ...

Graphical model

- Want to choose a model that ...
 - represents** relevant statistical dependencies between variables
 - we can use to make **inferences** (make predictions, etc.)
 - we can **learn** from training data

What you have learned so far

- **Representation**
 - Bayesian Networks
 - Markov Networks
 - Conditional independence is key
- **Inference**
 - Variable Elimination and Junction tree inference
 - Exact inference possible if graph has low treewidth
- **Learning**
 - **Parameters:** Can do MLE and Bayesian learning in Bayes Nets and Markov Nets if data fully observed
 - **Structure:** Can find optimal tree

Representation

- Conditional independence = Factorization
 - Represent factorization/independence as graph
 - Directed graphs: Bayesian networks
 - Undirected graphs: Markov networks
 - Typically, assume factors in exponential family (e.g., Multinomial, Gaussian, ...)
-
- So far, we assumed **all variables** in the model are **known**
 - In practice
 - Existence of variables can depend on data
 - Number of variables can grow over time
 - We might have hidden (unobserved variables)!

Inference

- Key idea: Exploit factorization (distributivity)
- Complexity of inference depends on treewidth of underlying model
 - Junction tree inference “only” exponential in treewidth
- In practice, often have high treewidth
 - Always high treewidth in DBNs
 - Need approximate inference

Learning

- Maximum likelihood estimation
 - In BNs: independent optimization for each CPT (decomposable score)
 - In MNs: Partition function couples parameters, but can do gradient ascent (no local optima!)
- Bayesian parameter estimation
 - Conjugate priors convenient to work with
- Structure learning
 - NP-hard in general
 - Can find optimal tree (Chow Liu)
- So far: Assumed all variables are observed
 - In practice: often have missing data

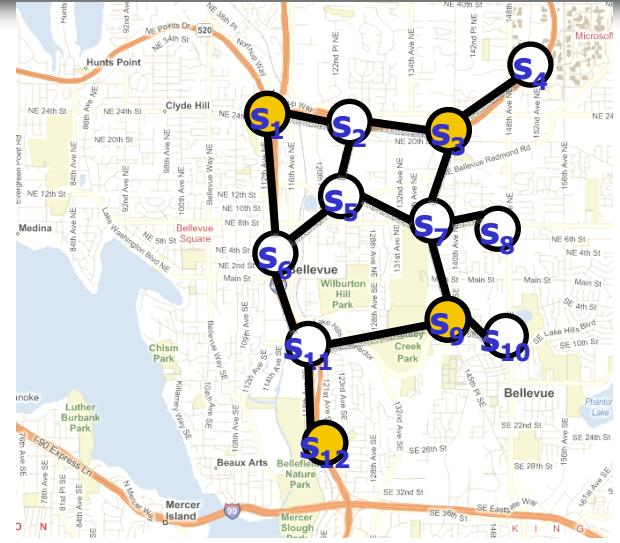
The “light” side

- Assumed
 - everything fully observable
 - low treewidth
 - no hidden variables
- Then everything is nice 😊
 - Efficient exact inference in large models
 - Optimal parameter estimation without local minima
 - Can even solve some structure learning tasks exactly

The “dark” side



represent



States of the world,
sensor measurements, ...

Graphical model

- In the real world, these assumptions are often violated..
- Still want to use graphical models to solve interesting problems..

Remaining Challenges

- Representation:
 - Dealing with **hidden variables**
- **Approximate inference** for high-treewidth models
- Dealing with **missing data**
- This will be focus of remaining part of the course!

Recall: Hardness of inference

- Computing conditional distributions:
 - Exact solution: **#P-complete**
 - Approximate solution: **NP-hard**
- Maximization:
 - MPE: **NP-complete**
 - MAP: **NP^{PP}-complete**

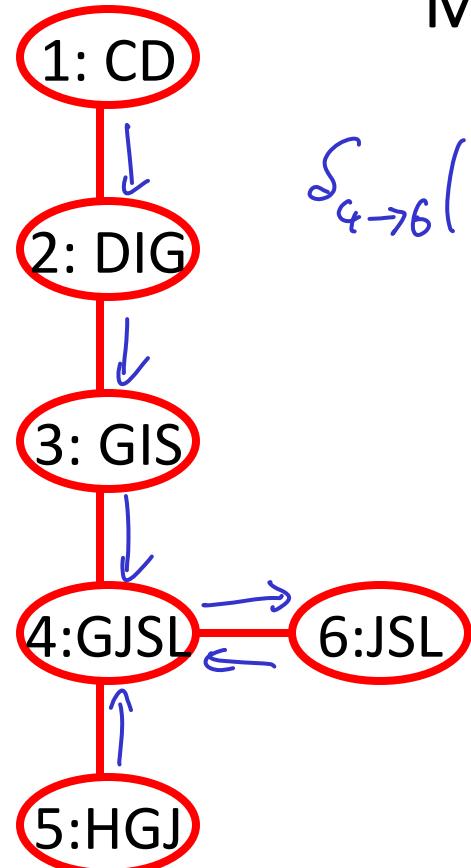
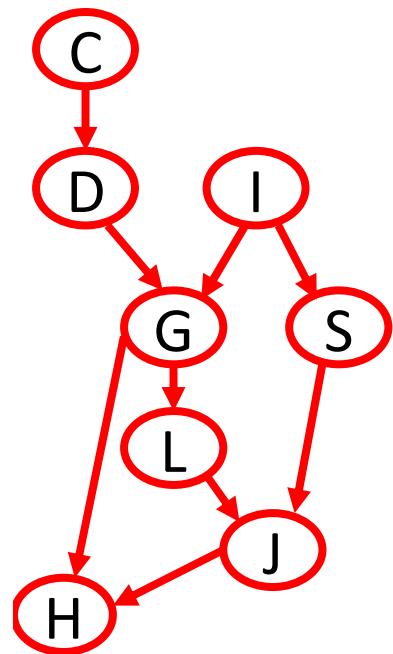
Inference

- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations
 - Whenever the graph is low treewidth
 - Whenever there is **context-specific independence**
 - Several other special cases
- For BNs where exact inference is not possible, can use algorithms for **approximate inference**
 - Coming up now!

Approximate inference

- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation (today!)
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

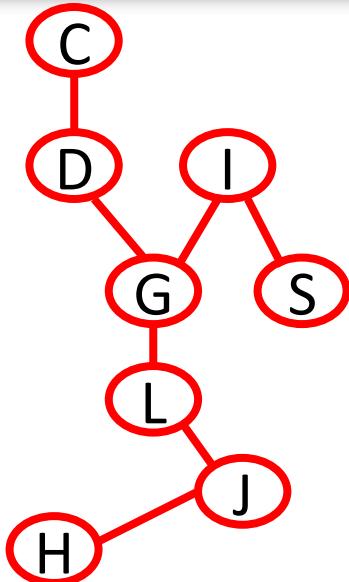
Recall: Message passing in Junction trees



Messages between clusters:

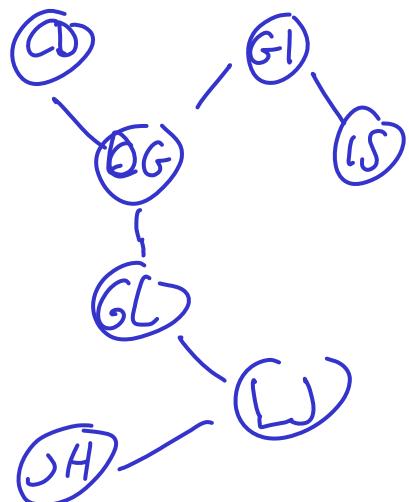
$$\delta_{4 \rightarrow 6}(j, \delta_L) = \sum_g \pi(g, j, \delta_L) \cdot \delta_{3 \rightarrow 4}(g, j) \cdot \delta_{5 \rightarrow 4}(g, j)$$

BP on Tree Pairwise Markov Nets



- Suppose graph is given as tree pairwise Markov net
- Don't need a junction tree!
 - Graph is already a tree!
- Example message:

$$\delta_{G \rightarrow L}(L) = \sum_g \pi_{GL}(g, L) \cdot \overline{\pi}_G(g) \delta_{D \rightarrow G}(g) \delta_{I \rightarrow G}(g)$$

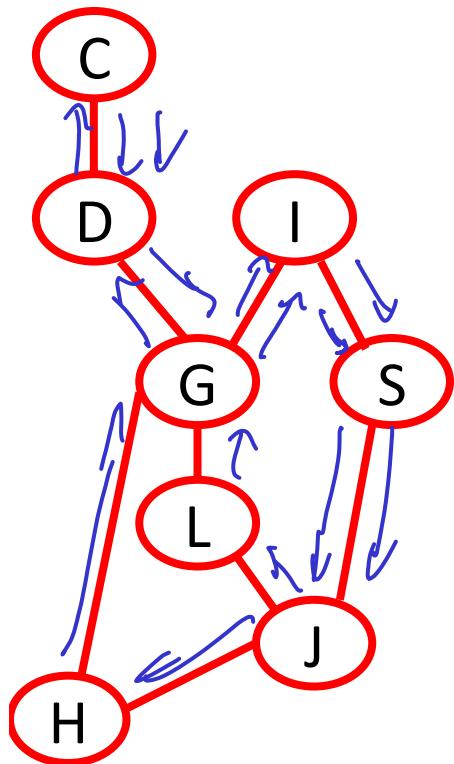


- More generally:

$$\delta_{i \rightarrow j}(x_j) = \sum_{x_i} \pi_{i \rightarrow j}(x_i, x_j) \overline{\pi}_i(x_i) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \rightarrow i}(x_i)$$

- **Theorem:** For trees, get correct answer!

Loopy BP on arbitrary pairwise MNs



- What if we apply BP to a graph with loops?
 - Apply BP and hope for the best..

$$\delta_{i \rightarrow j}(X_j) = \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \rightarrow i}(x_i)$$

- Will not generally converge..
- If it converges, will not necessarily get correct marginals

$$\hat{P}(x_i) \propto \prod_{s \in N(i)} \delta_{s \rightarrow i}(x_i)$$

- However, in practice, answers often still useful!

Practical aspects of Loopy BP

- Messages product of numbers ≤ 1

$$\delta_{i \rightarrow j}(X_j) = \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \rightarrow i}(x_i)$$

- On loopy graphs, repeatedly multiply same factors
→ products converge to 0 (numerical problems)
- Solution:

- Renormalize!

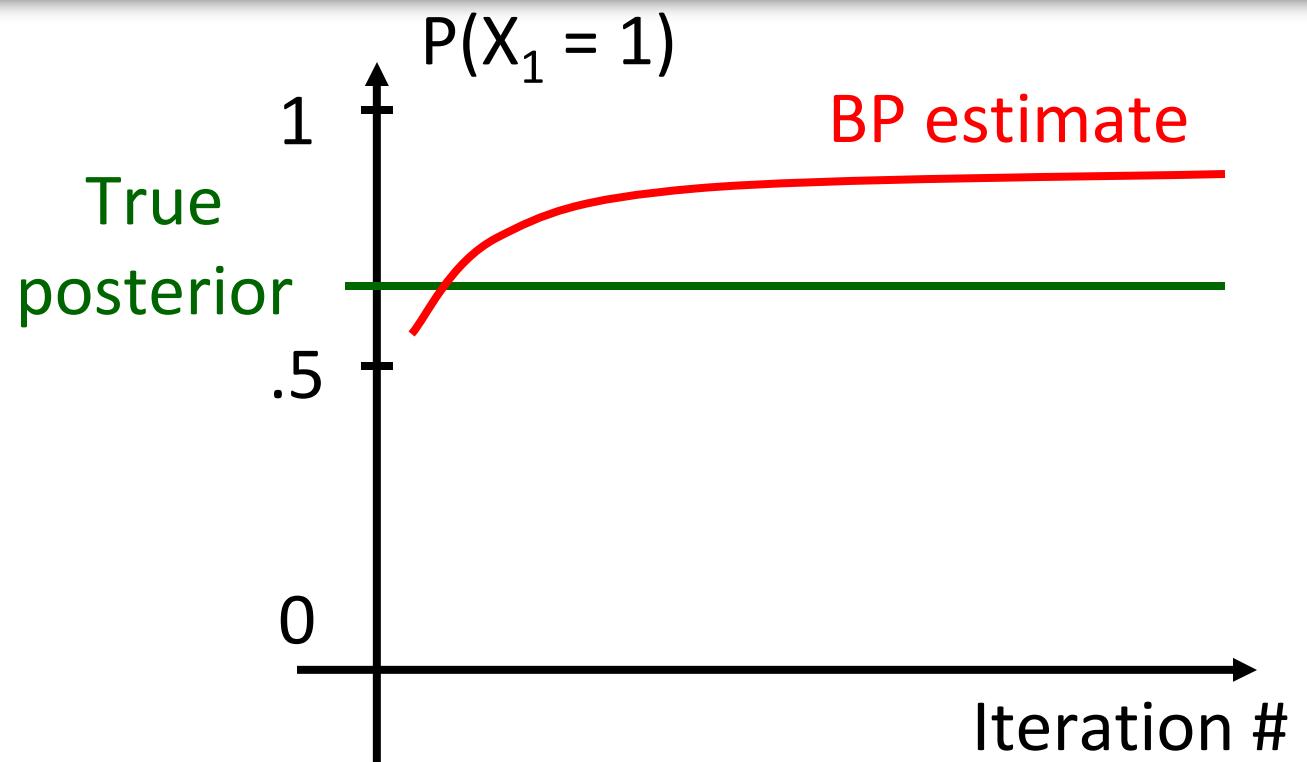
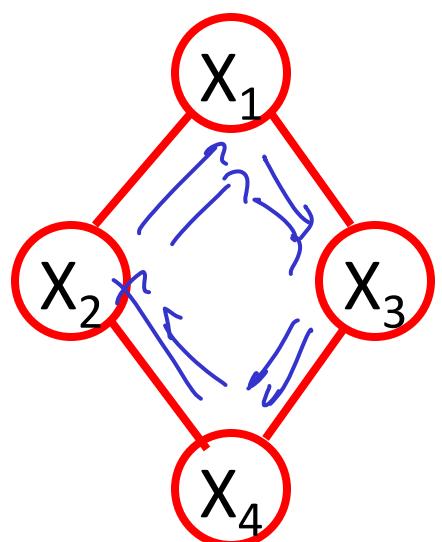
$$\delta_{i \rightarrow j}(X_j) = \frac{1}{Z_{i \rightarrow j}} \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \rightarrow i}(x_i)$$

- Does not affect outcome:

$$\hat{P}(x_i) \propto \overline{\prod_{s \in N(i)}} \delta_{s \rightarrow i}(x_i) \cdot (Z_{i \rightarrow j})$$

Normalization doesn't matter

Behavior of BP



- Loopy BP multiplies same potentials multiple times
→ BP often overconfident

When do we stop?

- Messages

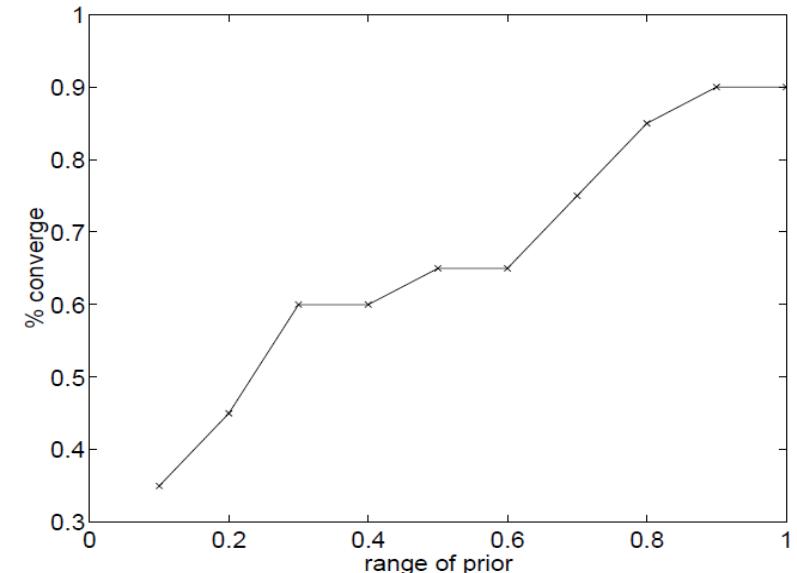
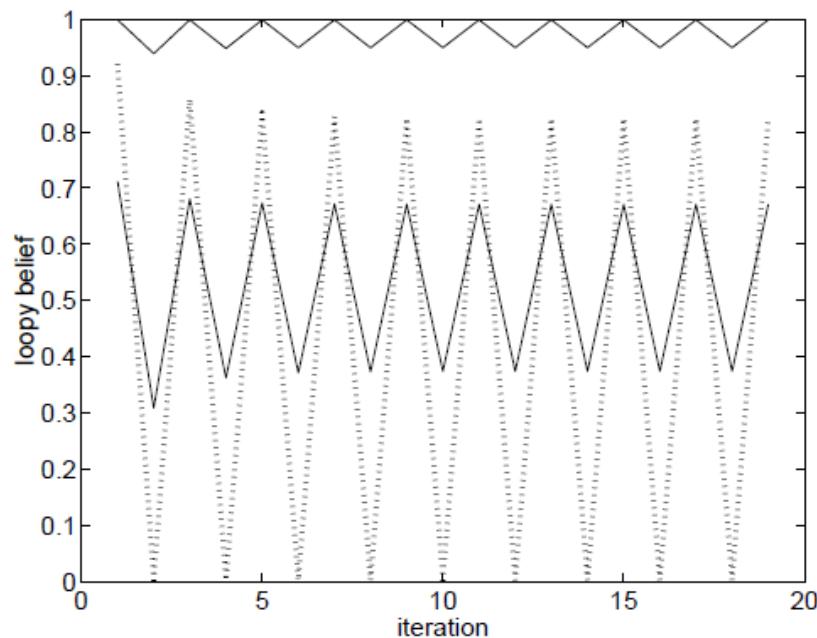
$$\delta_{i \rightarrow j}^{(t+1)}(X_j) = \frac{1}{Z_{i \rightarrow j}} \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \rightarrow i}^{(t)}(x_i)$$

Stop if messages "don't change much"

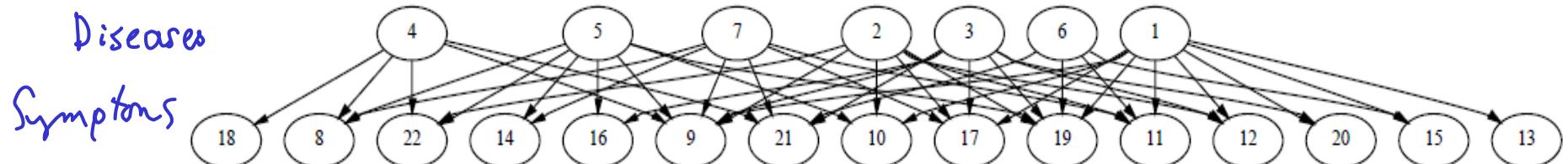
$$|\delta_{i \rightarrow j}^{(t+1)} - \delta_{i \rightarrow j}^{(t)}| \leq \varepsilon \quad \forall i, j$$

Does Loopy BP always converge?

- No! Can oscillate!
- Typically, oscillation the more severe the more “deterministic” the potentials



Graphs from K. Murphy UAI '99



What can we do to make BP converge?

Damping:

$$\hat{\delta}_{i \rightarrow j}^{(t+1)} = (1-\alpha) \hat{\delta}_{i \rightarrow j}^{(t+1)} + \alpha \hat{\delta}_{i \rightarrow j}^{(t)}$$

↑
what I sad

↑
"Correct" BP
message

↑
what I sent
last time

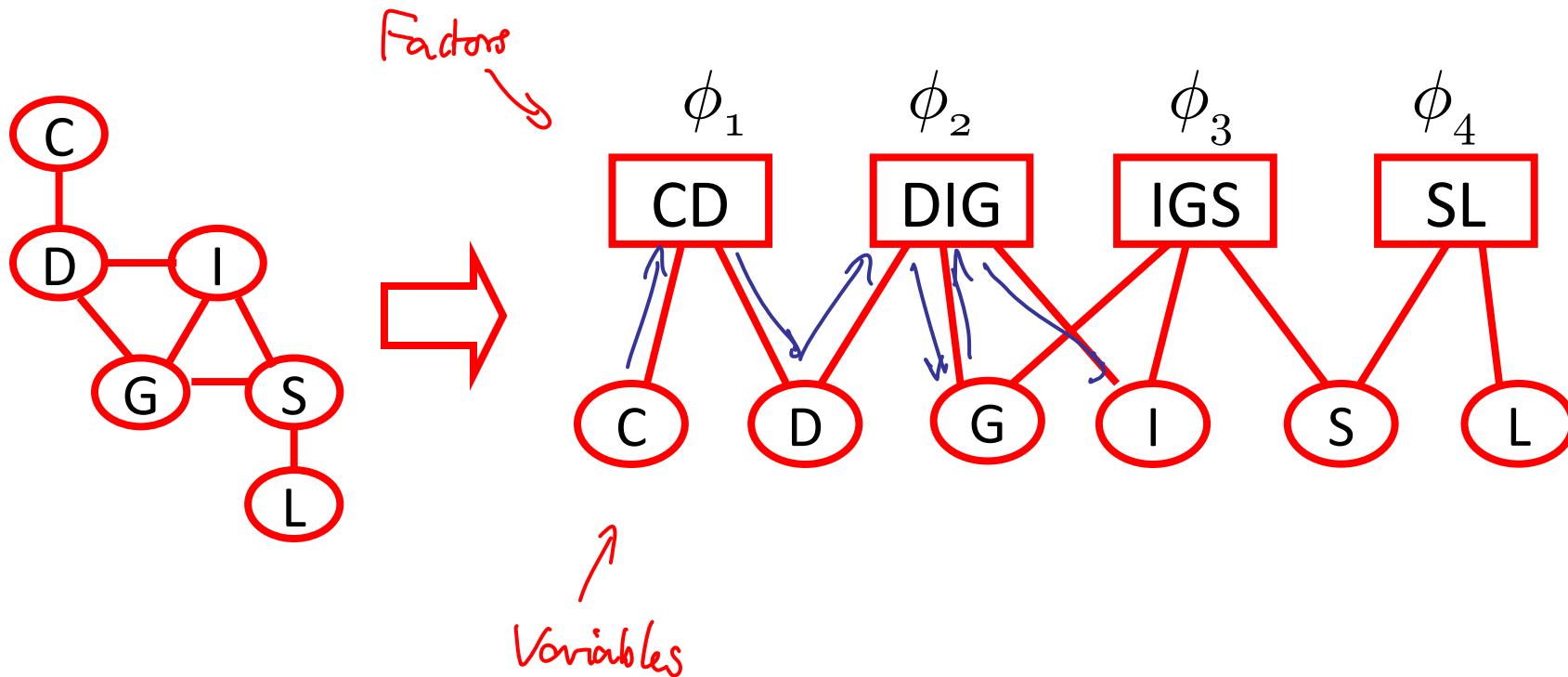
If we need to dampen, answer will most likely be bad

Can we prove convergence of BP?

- Yes, for special types of graphs (e.g., random graphs arising in coding)
- Sometimes can prove that message update “contracts”

What if we have non-pairwise MNs?

- Two approaches:
 - Convert to pairwise MN (possibly exponential blowup)
 - Perform BP on **factor graph**



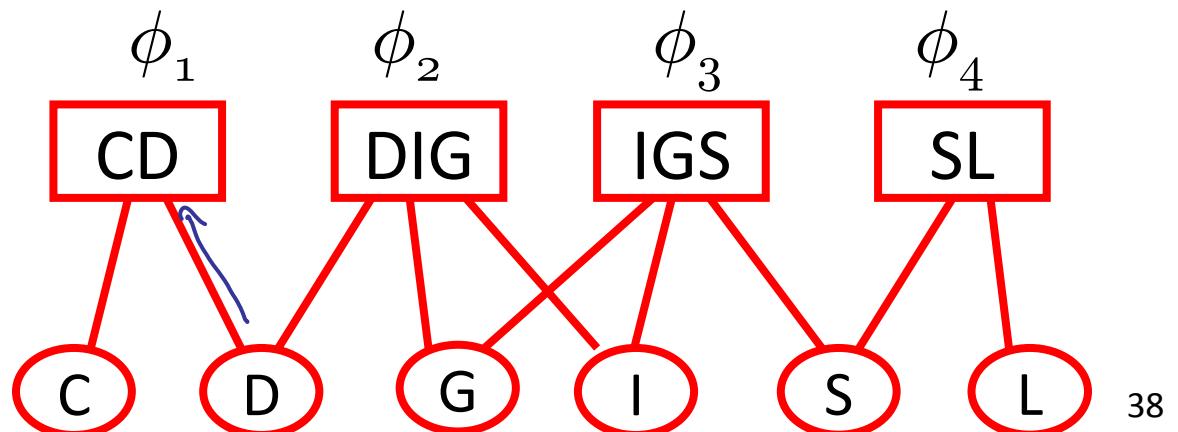
BP on factor graphs

- Messages from nodes to factors

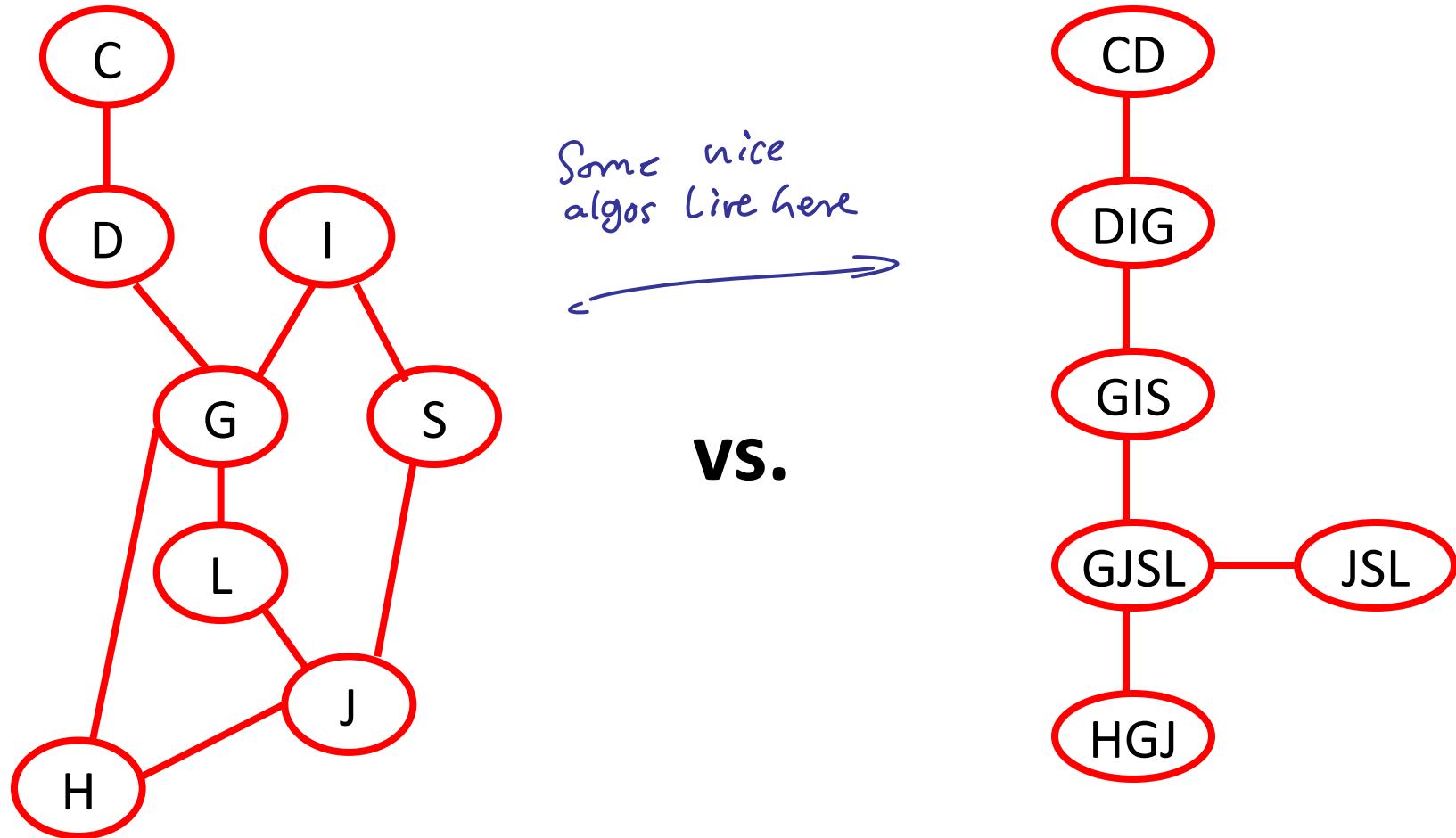
$$\delta_{x \rightarrow \phi}(x) = \frac{1}{Z} \prod_{\phi' \in N(x) \setminus \{\phi\}} \delta_{\phi' \rightarrow x}(x)$$

- Messages from factors to nodes

$$\delta_{\phi \rightarrow x}(x) = \frac{1}{Z} \sum_{x_\phi \sim x} \phi(x_\phi) \prod_{x' \in N(\phi) \setminus \{x\}} \delta_{x' \rightarrow \phi}(x)$$



Loopy BP vs Junction tree



Both BP and JT inference are “ends of a spectrum”

Other message passing algorithms

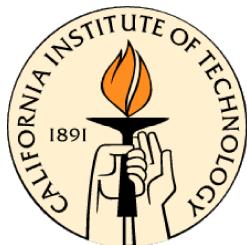
- Gaussian Belief propagation
- BP based on particle filters (see sampling)
- Expectation propagation
- ...

Variational Mean Field for Graphical Models

CS/CNS/EE 155

Baback Moghaddam
Machine Learning Group

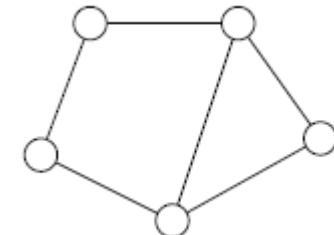
baback@jpl.nasa.gov



Approximate Inference

- Consider general UGs (*i.e.*, not tree-structured)

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \exp \{ \theta_C(x_C) \}$$



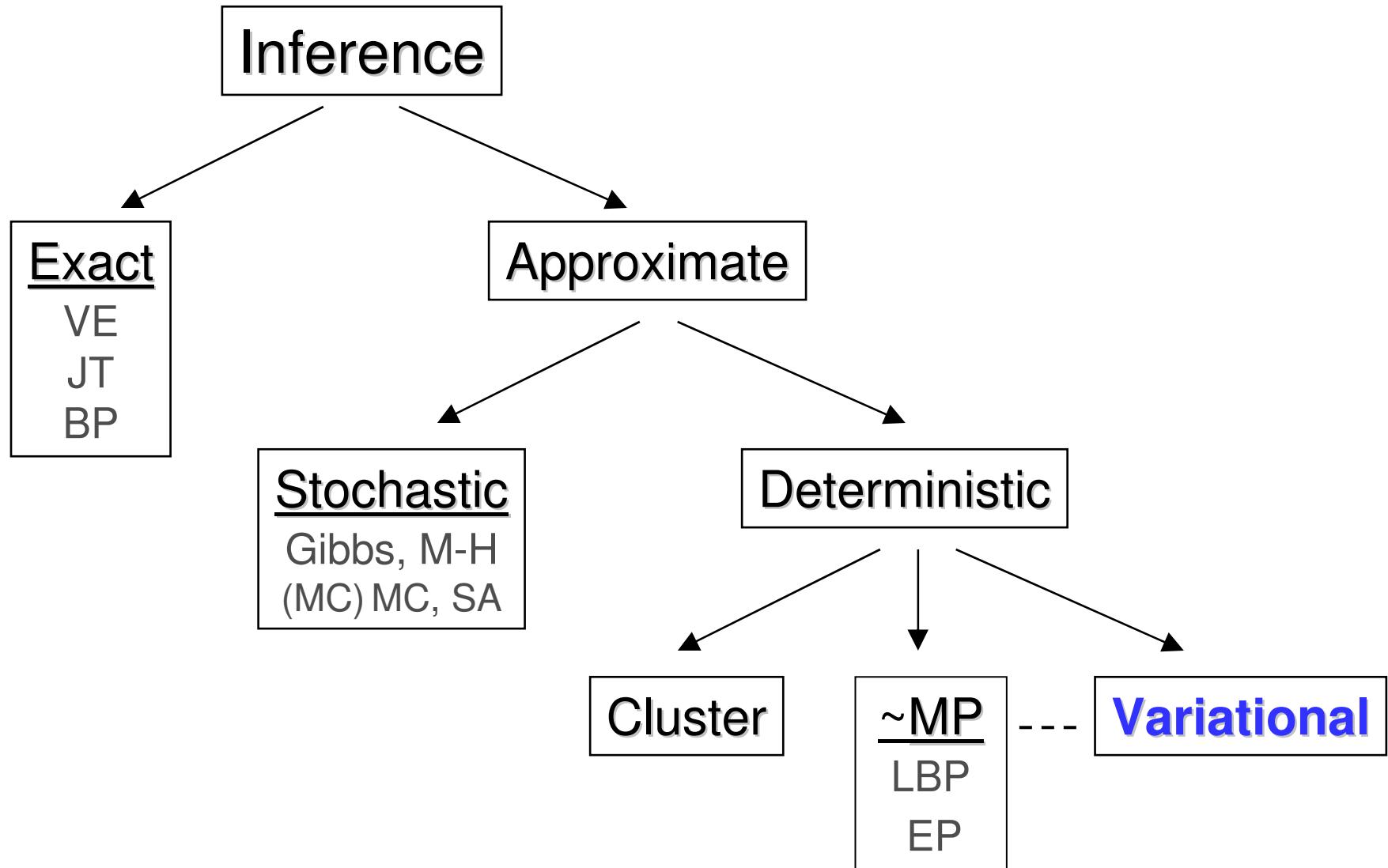
- All basic computations are intractable (for large G)

- likelihoods & partition function $Z = \sum_{x \in \mathcal{X}^N} \prod_{C \in \mathcal{C}} \exp \{ \theta_C(x_C) \}$

- marginals & conditionals $p(X_s = x_s) = \sum_{x_t, t \neq s} \prod_{C \in \mathcal{C}} \exp \{ \theta_C(x_C) \}$

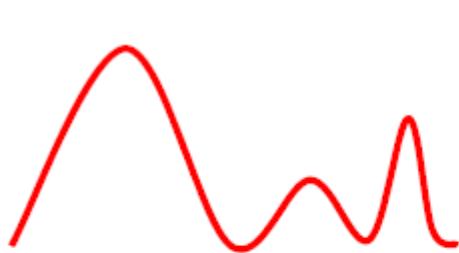
- finding modes $\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{X}^N} p(\mathbf{x}) = \arg \max_{\mathbf{x} \in \mathcal{X}^N} \prod_{C \in \mathcal{C}} \exp \{ \theta_C(x_C) \}$

Taxonomy of Inference Methods

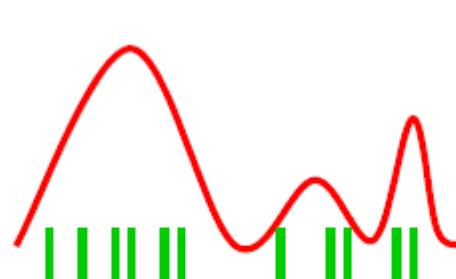


Approximate Inference

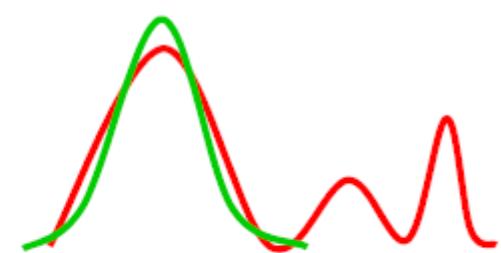
- Stochastic (Sampling)
 - Metropolis-Hastings, Gibbs, (Markov Chain) Monte Carlo, etc
 - Computationally *expensive*, but is “exact” (in the limit)
- Deterministic (Optimization)
 - Mean Field (MF), Loopy Belief Propagation (LBP)
 - Variational Bayes (VB), Expectation Propagation (EP)
 - Computationally *cheaper*, but is not exact (gives bounds)



True distribution



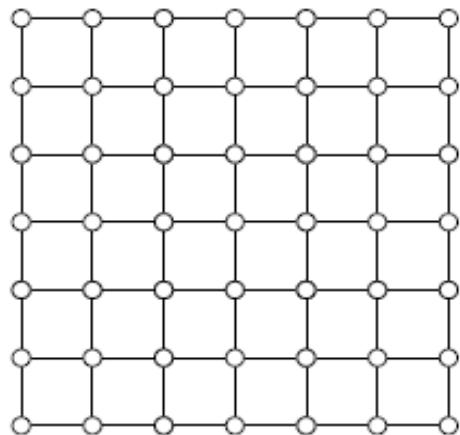
Monte Carlo



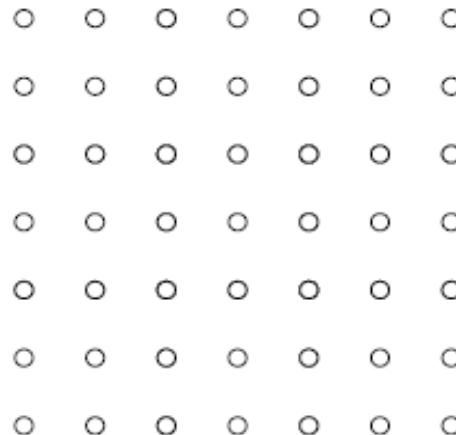
VB / Loopy BP / EP

Mean Field : Overview

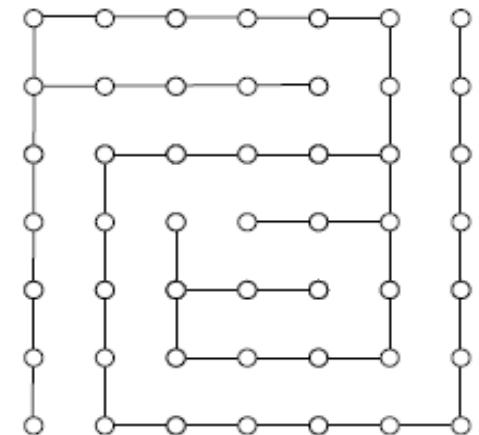
- General idea
 - approximate $p(x)$ by a simpler factored distribution $q(x)$
 - minimize “distance” $D(p \parallel q)$ - e.g., Kullback-Liebler



(Naïve) MF H_0



structured MF H_s



$$p(x) \propto \prod_c \phi_c(x_c)$$

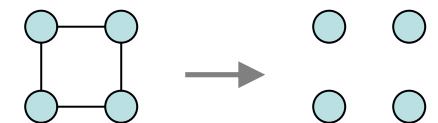
$$q(x) \propto \prod_i q_i(x_i)$$

$$q(x) \propto q_A(x_A) q_B(x_B)$$

Mean Field : Overview

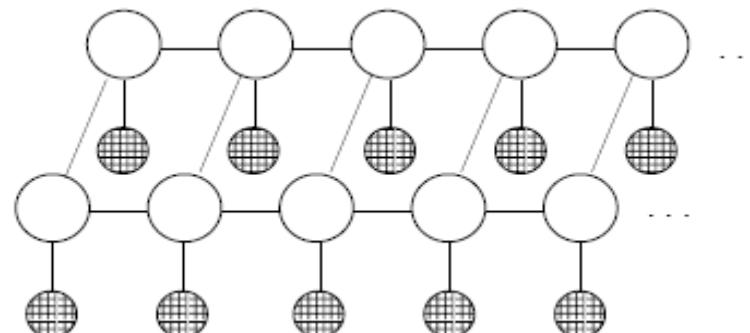
- Naïve MF has roots in *Statistical Mechanics* (1890s)
 - physics of spin glasses (Ising), ferromagnetism, etc
 - why is it called “Mean Field” ?

with full factorization : $E[x_i x_j] = E[x_i] E[x_j]$

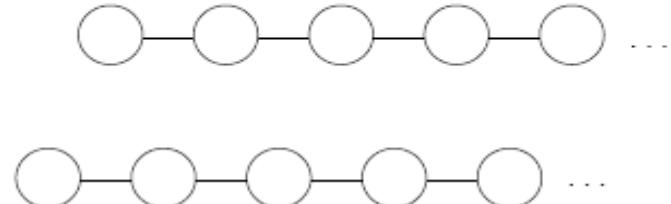


- Structured MF is more “modern”

Coupled HMM



Structured MF approximation
(with tractable chains)



KL Projection $D(Q||P)$

- Infer hidden h given visible v (clamp v nodes with δ 's)

$$P(h|v) = \prod_{c \in C(G)} f_c(h_c), \quad Q(h) = \prod_{c \in C(G')} q_c(h_c)$$

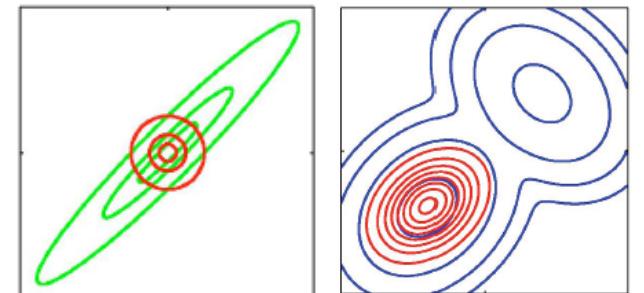
- **Variational** : optimize KL globally

$$\min_Q D(Q||P) = \sum_h Q(h) \ln \frac{Q(h)}{P(h|v)}$$

$P = 0$ forces $Q = 0$

the right density form for Q “falls out”

KL is *easier* since we’re taking $E[.]$ wrt simpler Q
 Q seeks mode with the largest mass (not height)
so it will tend to *underestimate* the support of P



KL Projection $D(P||Q)$

- Infer hidden h given visible v (clamp v nodes with δ 's)

$$P(h|v) = \prod_{c \in C(G)} f_c(h_c), \quad Q(h) = \prod_{c \in C(G')} q_c(h_c)$$

- Expectation Propagation (EP) : optimize KL locally

$$\min_{q_c} D(P||Q) = \sum_h P(h|v) \ln \frac{P(h|v)}{Q(h)}$$

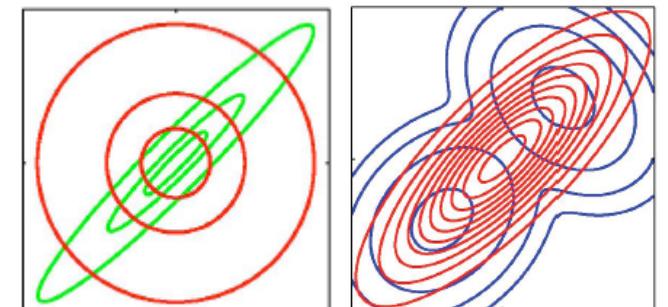
this KL is *harder* since we're taking $E[.]$ wrt P

no nice global solution for Q “falls out”

must sequentially tweak each q_c (match moments)

Q covers *all* modes so it *overestimates* support

$P > 0$ forces $Q > 0$



α - divergences

- The 2 basic KL divergences are *special cases* of

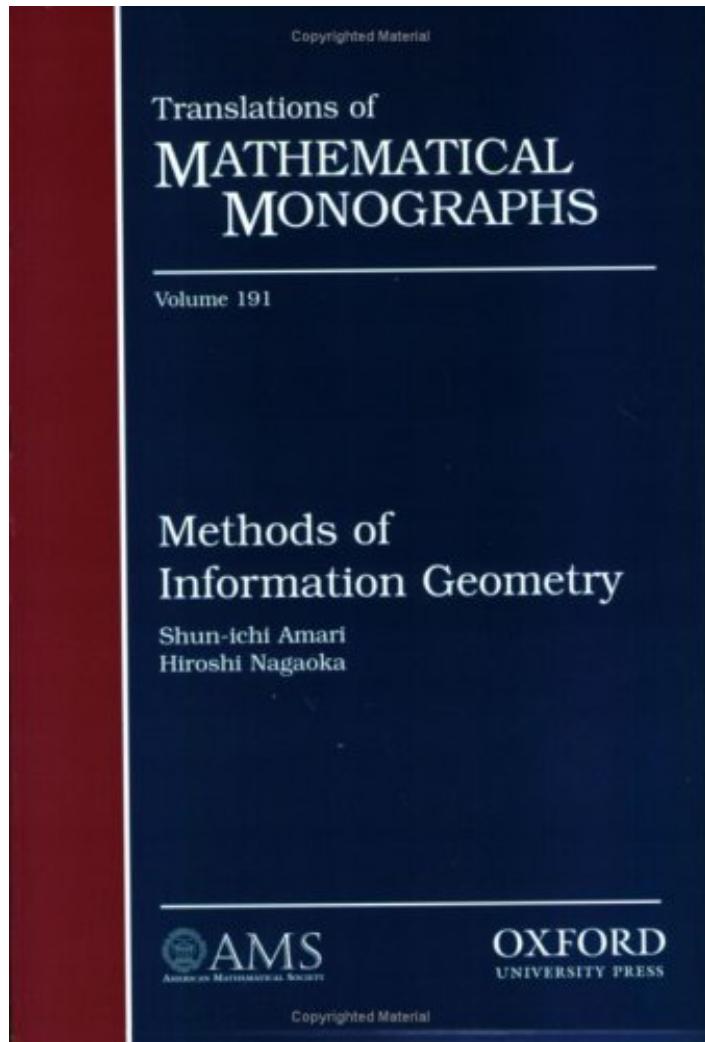
$$D_\alpha(p \parallel q) = \frac{4}{1 - \alpha^2} \left(1 - \int p(x)^{(1+\alpha)/2} q(x)^{(1-\alpha)/2} dx \right)$$

- $D_\alpha(p \parallel q)$ is non-negative and 0 iff $p = q$
 - when $\alpha \rightarrow -1$ we get $KL(P \parallel Q)$
 - when $\alpha \rightarrow +1$ we get $KL(Q \parallel P)$
 - when $\alpha = 0$ $D_0(P \parallel Q)$ is proportional to Hellinger's distance (metric)

$$D_H(p \parallel q) = \int (p(x)^{1/2} - q(x)^{1/2})^2 dx$$

So many variational approximations must exist, one for each α !

for more on α -divergences



Shun-ichi Amari



for specific examples of $\alpha = \pm 1$

See Chapter 10

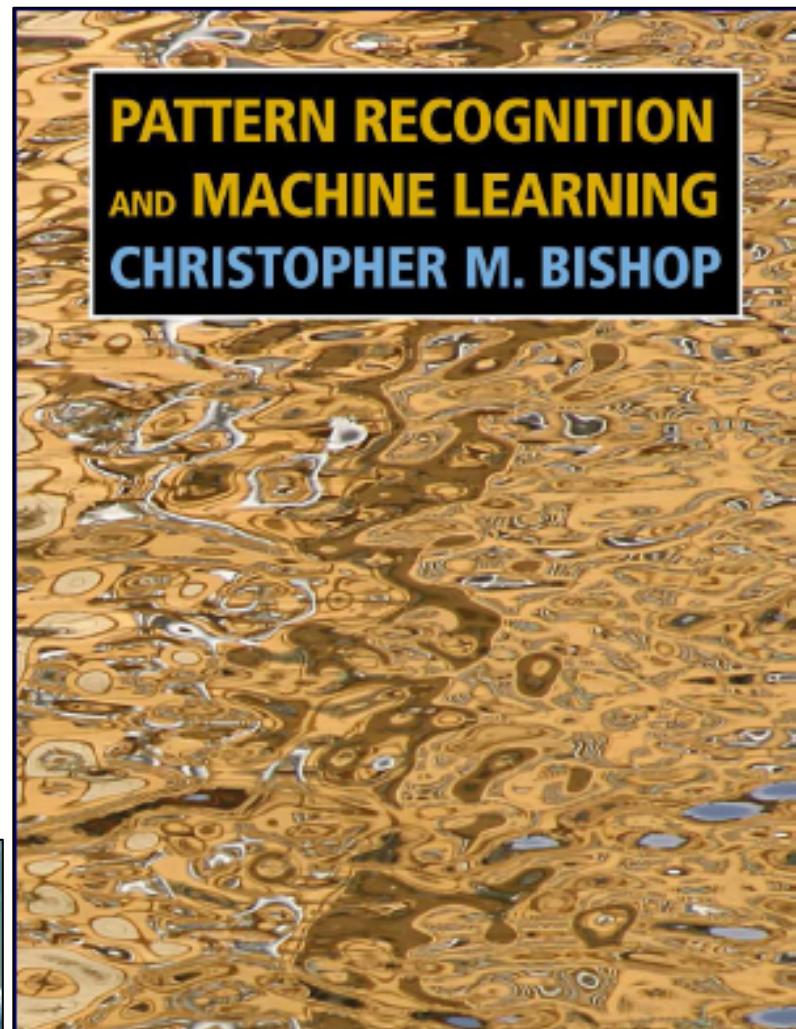
Variational Single Gaussian

Variational Linear Regression

Variational Mixture of Gaussians

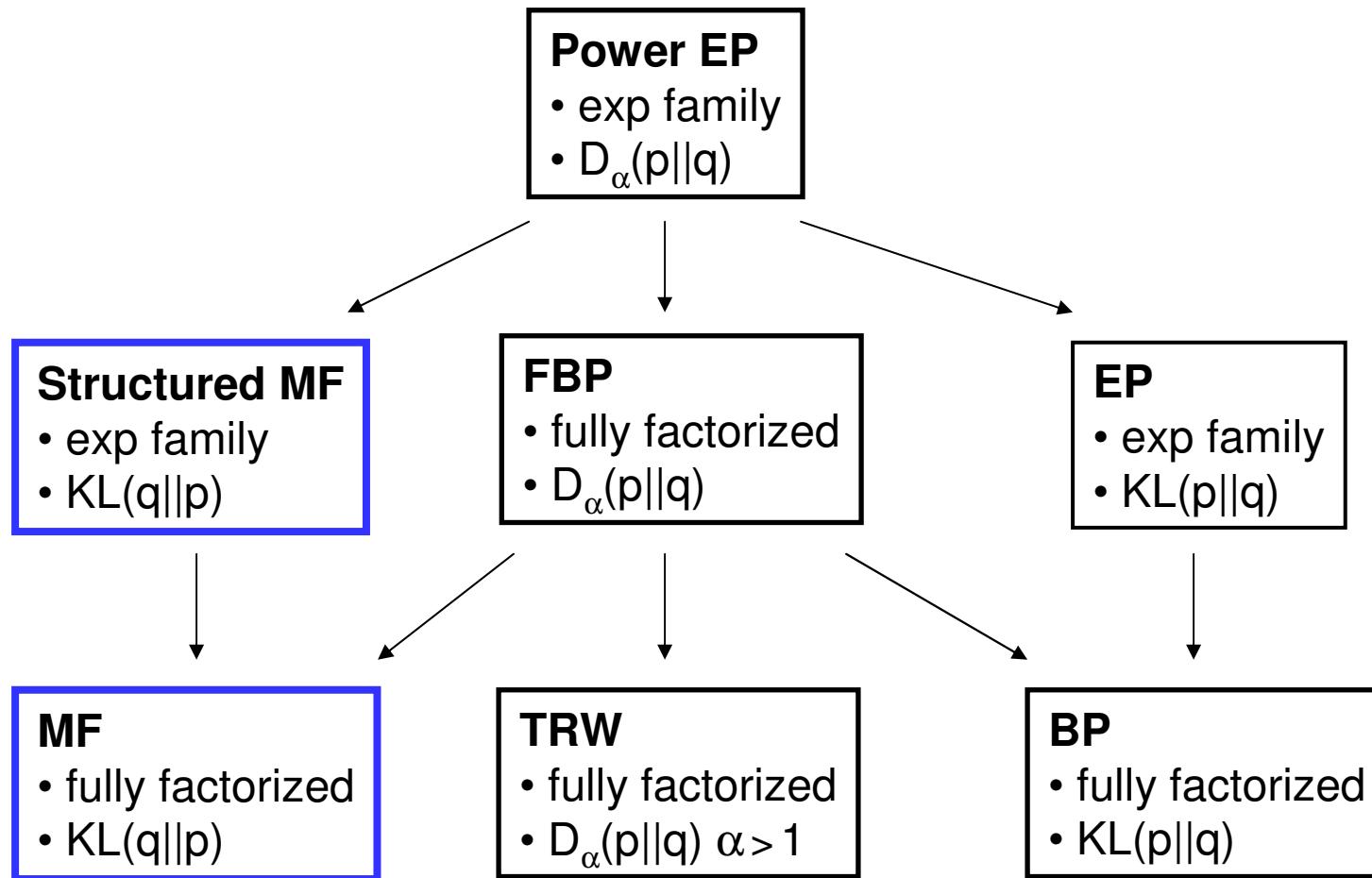
Variational Logistic Regression

Expectation Propagation ($\alpha = -1$)



Hierarchy of Algorithms

(based on α and structuring)



by Tom Minka

Variational MF

$$p(x) = \frac{1}{Z} \prod_c \gamma_c(x_c) = \frac{1}{Z} e^{\psi(x)} \quad \psi(x) = \sum_c \log(\gamma_c(x_c))$$

$$\begin{aligned}\log Z &= \log \int e^{\psi(x)} dx \\ &= \log \int Q(x) \frac{e^{\psi(x)}}{Q(x)} dx \quad \text{Jensen's} \geq E_Q \log [e^{\psi(x)} / Q(x)] \\ &= \sup_Q E_Q \log [e^{\psi(x)} / Q(x)] \\ &= \sup_Q \{ E_Q [\psi(x)] + H[Q(x)] \}\end{aligned}$$

Variational MF

$$\log Z \geq \sup_Q \{ E_Q[\psi(x)] + H[Q(x)] \}$$

Equality is obtained for $Q(x) = P(x)$ (all Q admissible)

Using *any* other Q yields a lower bound on $\log Z$

The slack in this bound is KL-divergence $D(Q||P)$

Goal: restrict Q to a *tractable subclass* \mathbf{Q}
optimize with \sup_Q to tighten this bound

note we're (also) maximizing entropy $H[Q]$

Variational MF

$$\log Z \geq \sup_Q \{ E_Q[\psi(x)] + H[Q(x)] \}$$

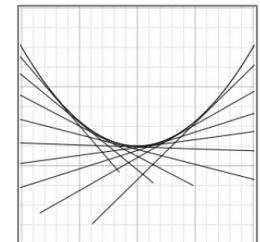
Most common specialized family :

“log-linear models” $\psi(x) = \sum_c \theta_c \phi_c(x_c) = \theta^T \phi(x)$

linear in parameters θ (natural parameters of EFs)

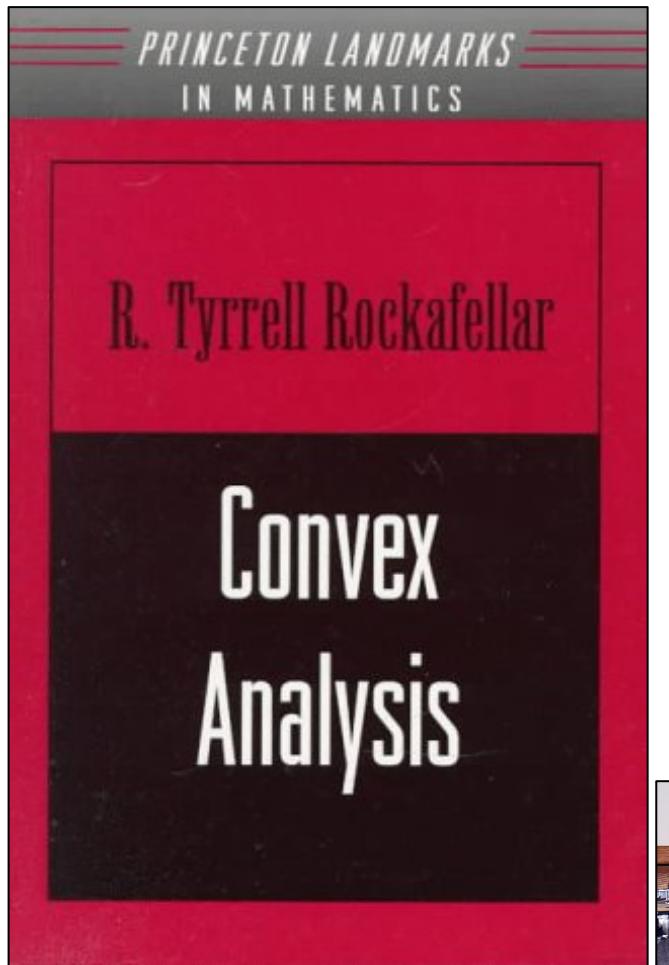
clique potentials $\phi(x)$ (sufficient statistics of EFs)

Fertile ground for plowing Convex Analysis

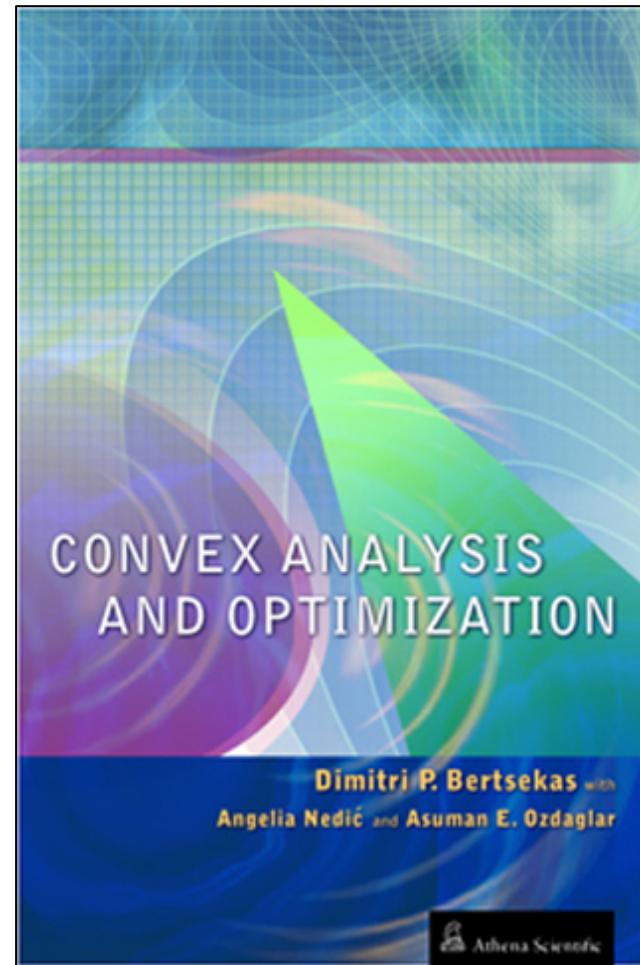


Convex Analysis

The Old Testament



The New Testament

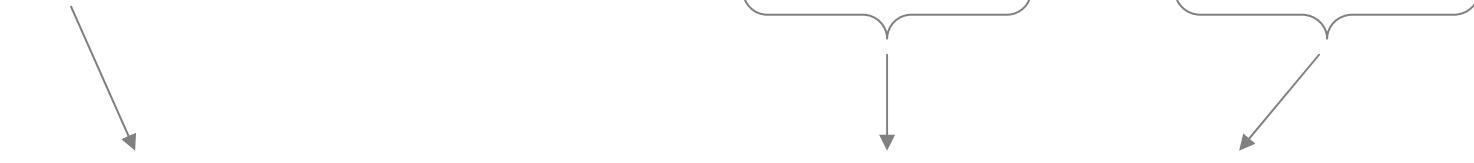


Variational MF for EF

$$\log Z \geq \sup_Q \{ E_Q [\underbrace{\psi(x)}_{\text{}}] + H[Q(x)] \}$$

$$\log Z \geq \sup_Q \{ E_Q [\underbrace{\theta^T \phi(x)}_{\text{}}] + H[Q(x)] \}$$

$$\log Z \geq \sup_Q \{ \theta^T E_Q [\phi(x)] + H[Q(x)] \}$$



$$A(\theta) \geq \sup_{\mu \in M} \{ \theta^T \mu - A^*(\mu) \}$$

EF
notation

M = set of all *moment* parameters realizable under subclass \mathbf{Q}

Variational MF for EF

So it looks like we are just optimizing a concave function (linear term + negative-entropy) over a convex set

Wait ... that doesn't sound so hard! Yet it is **hard** ... Why?

1. graph probability (being a *measure*) requires a very large number of **marginalization** constraints for *consistency* (leads to a typically beastly marginal polytope M in the discrete case)

e.g., a complete 7-node graph's polytope has over 10^8 facets !

In fact, optimizing just the linear term alone can be hard

2. exact computation of **entropy** $-A^*(\mu)$ is highly non-trivial (hence the famed Bethe & Kikuchi approximations)

Gibbs Sampling for Ising

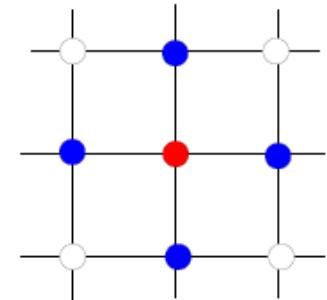
- Binary MRF $G = (V, E)$ with pairwise clique potentials

$$p(\mathbf{x}; \theta) \propto \exp \left\{ \sum_{s \in V} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t \right\}$$

- 1. pick a node s at random
 2. sample $u \sim \text{Uniform}(0,1)$
 3. update node s :

$$x_s^{(m+1)} = \begin{cases} 1 & \text{if } u \leq \{1 + \exp[-(\theta_s + \sum_{t \in \mathcal{N}(s)} \theta_{st} x_t^{(m)})]\}^{-1} \\ 0 & \text{otherwise} \end{cases}$$

- 4. goto step 1



a slower stochastic version of ICM

Naive MF for Ising

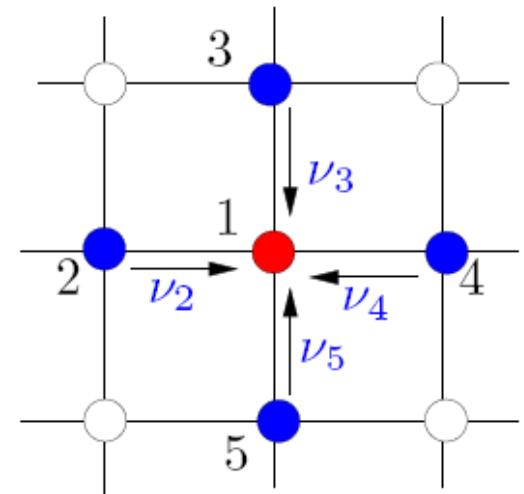
- use a variational mean parameter at each site $\nu_s \in (0, 1)$

- 1. pick a node s at random
2. update its parameter :

$$\nu_s \leftarrow \left\{ 1 + \exp[-(\theta_s + \sum_{t \in \mathcal{N}(s)} \theta_{st} \nu_t)] \right\}^{-1}$$

- 3. goto step 1

- deterministic “loopy” message-passing
- how well does it work? depends on θ



Graphical Models as EF

- $G(V, E)$ with nodes $X_s \in \{0, 1, \dots, m_s - 1\}$
- sufficient stats :
 $\mathbb{I}_j(x_s) \quad \text{for } s = 1, \dots, n, \quad j \in \mathcal{X}_s$
 $\mathbb{I}_{jk}(x_s, x_t) \quad \text{for } (s, t) \in E, \quad (j, k) \in \mathcal{X}_s \times \mathcal{X}_t$
- clique potentials $\theta_s(x_s) := \sum_{j \in \mathcal{X}_s} \theta_{s;j} \mathbb{I}_j(x_s)$ likewise for θ_{st}
- probability $p(\mathbf{x}; \theta) \propto \exp \left\{ \sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st}(x_s, x_t) \right\}$
- log-partition $A(\theta) = \log \sum_{\mathbf{x} \in \mathcal{X}^n} \exp \left\{ \sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st}(x_s, x_t) \right\}$

- mean parameters
 $\mu_s(x_s) := \sum_{j \in \mathcal{X}_s} \mu_{s;j} \mathbb{I}_j(x_s)$
 $\mu_{st}(x_s, x_t) := \sum_{(j,k) \in \mathcal{X}_s \times \mathcal{X}_t} \mu_{st;jk} \mathbb{I}_{jk}(x_s, x_t)$

Variational Theorem for EF

- For any mean parameter μ where $\theta(\mu)$ is the corresponding natural parameter

$$A^*(\mu) = \begin{cases} -H(p_{\theta(\mu)}) & \text{if } \mu \in \mathcal{M}^\circ \quad \text{in relative interior of } M \\ +\infty & \text{if } \mu \notin \overline{\mathcal{M}}. \quad \text{not in the closure of } M \end{cases}$$

- the log-partition function has this variational representation

$$A(\theta) = \sup_{\mu \in \mathcal{M}} \{ \langle \theta, \mu \rangle - A^*(\mu) \}$$

- this supremum is achieved at the moment-matching value of μ

$$\mu = \int_{\mathcal{X}^m} \phi(x) p_\theta(x) \nu(dx) = \mathbb{E}_\theta[\phi(X)] = \nabla A(\theta(\mu))$$

Legendre-Fenchel Duality

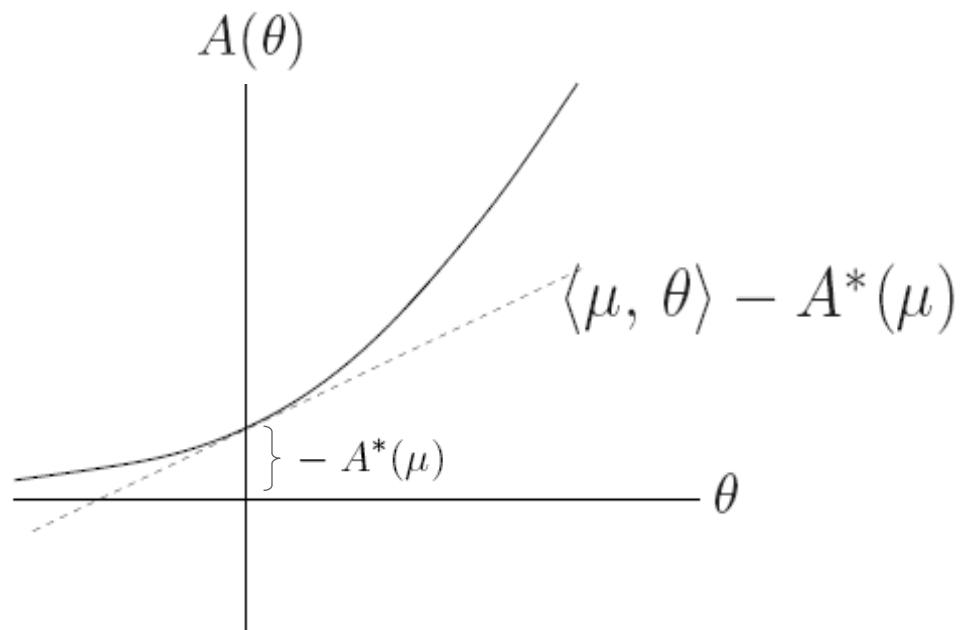
- **Main Idea:** (convex) functions can be “supported” (lower-bounded) by a continuum of lines (hyperplanes) whose intercepts create a *conjugate dual* of the original function (and vice versa)

conjugate dual of A

$$A^*(\mu) := \sup_{\theta \in \Omega} \{ \langle \mu, \theta \rangle - A(\theta) \}$$

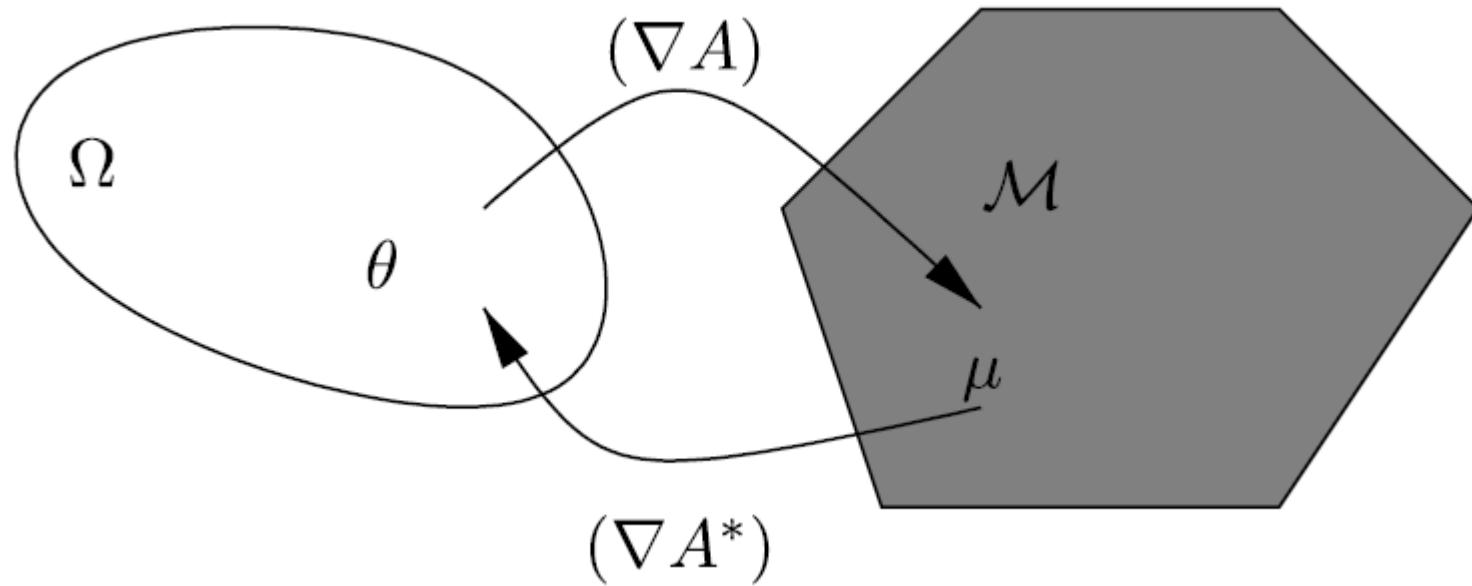
conjugate dual of A^*

$$A(\theta) = \sup_{\mu \in \mathcal{M}} \{ \langle \theta, \mu \rangle - A^*(\mu) \}$$



Note that $A^{**} = A$ (iff A is convex)

Dual Map for EF



Two equivalent parameterizations of the EF

Bijective mapping between Ω and the *interior* of M

Mapping is defined by the gradients of A and its dual A^*

Shape & complexity of M depends on X and size and structure of G

Marginal Polytope

- $G(V, E)$ = graph with discrete nodes
- Then M = convex hull of all $\phi(x)$

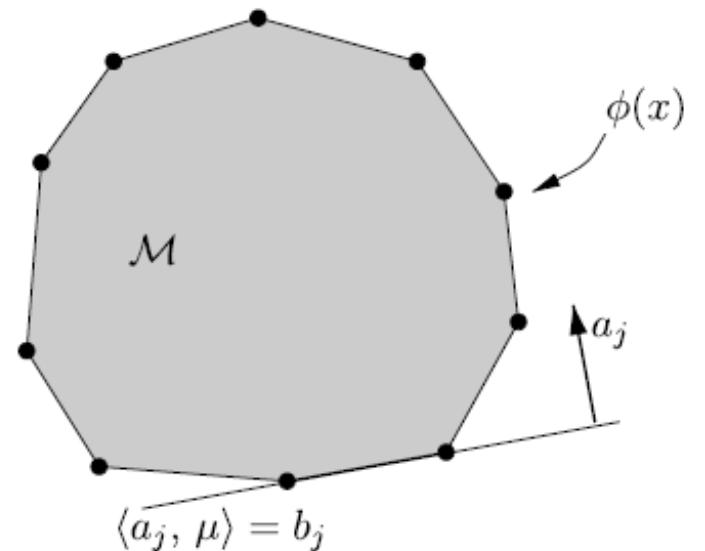
$$\mathcal{M} := \{\mu \in \mathbb{R}^d \mid \exists p \text{ s.t. } \mathbb{E}_p[\phi(X)] = \mu\}$$

- equivalent to intersecting half-spaces $a^T \mu > b$

- difficult to characterize for large G

- hence difficult to optimize over

- interior of M is 1-to-1 with Ω



The Simplest Graph

(x)

- $G(V, E) =$ a single Bernoulli node $\phi(x) = x$
- density $p(x; \theta) \propto \exp\{\theta x\}$
- log-partition $A(\theta) = \log[1 + \exp(\theta)]$ (of course we knew this)
- we know A^* too, but let's solve for it variationally

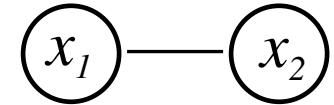
$$A^*(\mu) := \sup_{\theta \in \mathbb{R}} \{\mu\theta - \log[1 + \exp(\theta)]\}$$

- differentiate à stationary point $\mu = \exp(\theta)/[1 + \exp(\theta)]$
- rearrange to $\theta(\mu) := \log[\mu/(1 - \mu)]$, substitute into A^*

$$\begin{aligned} A^*(\mu) &= \mu \log[\mu/(1 - \mu)] - \log \left[1 + \frac{\mu}{1 - \mu} \right] \\ &= \mu \log \mu + (1 - \mu) \log(1 - \mu), \end{aligned}$$

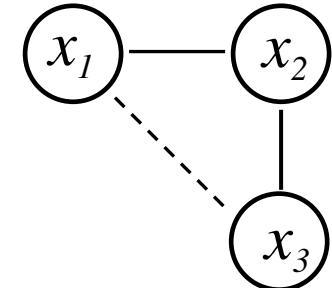
Note: we found *both* the mean parameter and the lower bound using the variational method

The 2nd Simplest Graph



- $G(V, E) = 2$ connected Bernoulli nodes $\phi(x) = \{x_1, x_2, x_1x_2\}$
- $p(x; \theta) \propto \exp \{\theta_1 x_1 + \theta_2 x_2 + \theta_{12} x_1 x_2\}$
- moments
$$\begin{aligned}\mu_1 &= \mathbb{E}[X_1] &= p(x_1 = 1) \\ \mu_2 &= \mathbb{E}[X_2] &= p(x_2 = 1) \\ \mu_{12} &= \mathbb{E}[X_1 X_2] &= p(x_1 = 1, x_2 = 1)\end{aligned}$$
- moment constraints
$$\begin{array}{lcl} \mu_1 & \geq & \mu_{12} \\ \mu_2 & \geq & \mu_{12} \\ \mu_{12} & \geq & 0 \\ 1 + \mu_{12} & \geq & \mu_1 + \mu_2 \end{array}$$
- $$\begin{aligned}A^*(\mu) &= -H(p(x; \mu)) \\ &= \sum_{x_1, x_2} p(x; \mu) \log p(x; \mu) \\ &= \mu_{12} \log \mu_{12} + (\mu_1 - \mu_{12}) \log (\mu_1 - \mu_{12}) + (\mu_2 - \mu_{12}) \log (\mu_2 - \mu_{12}) \\ &\quad + (1 + \mu_{12} - \mu_1 - \mu_2) \log (1 + \mu_{12} - \mu_1 - \mu_2)\end{aligned}$$
- variational problem $A(\theta) = \max \{\theta_1 \mu_1 + \theta_2 \mu_2 + \theta_{12} \mu_{12} - A^*(\mu)\}$
- solve (it's still easy!) $\hat{\mu}_1(\theta) = \frac{\exp\{\theta_1\} + \exp\{\theta_1 + \theta_2 + \theta_{12}\}}{1 + \exp\{\theta_1\} + \exp\{\theta_2\} + \exp\{\theta_1 + \theta_2 + \theta_{12}\}}$

The 3rd Simplest Graph



3 nodes \rightarrow 16 constraints

of constraints blows up real fast:

7 nodes \rightarrow 200,000,000+ constraints

hard to keep track of valid μ 's
(i.e., the full shape and extent of M)

no more checking our results against
closed-forms expressions that we
already knew in advance!

unless G remains a tree, entropy A^*
will not decompose nicely, etc

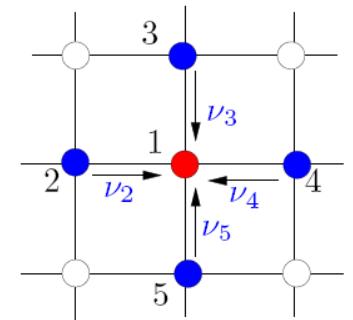
Variational MF for Ising

- tractable subgraph $H = (V, \emptyset)$
- fully-factored distribution $p(\mathbf{x}; \theta) = \prod_{s \in V} p(x_s; \theta_s)$
- moment space $\mathcal{M}_{tr}(G; H) = \{\mu \mid \boxed{\mu_{st} = \mu_s \mu_t}, \mu_s \in [0, 1]\}$
- entropy is *additive* : $- \sum_{s \in V} \mu_s \log \mu_s + (1 - \mu_s) \log(1 - \mu_s)$
- variational problem for $A(\theta)$

$$\max_{\mu_s \in [0, 1]} \left\{ \sum_{s \in V} \theta_s \mu_s + \sum_{(s, t) \in E} \theta_{st} \boxed{\mu_s \mu_t} - \left[\sum_{s \in V} \mu_s \log \mu_s + (1 - \mu_s) \log(1 - \mu_s) \right] \right\}$$

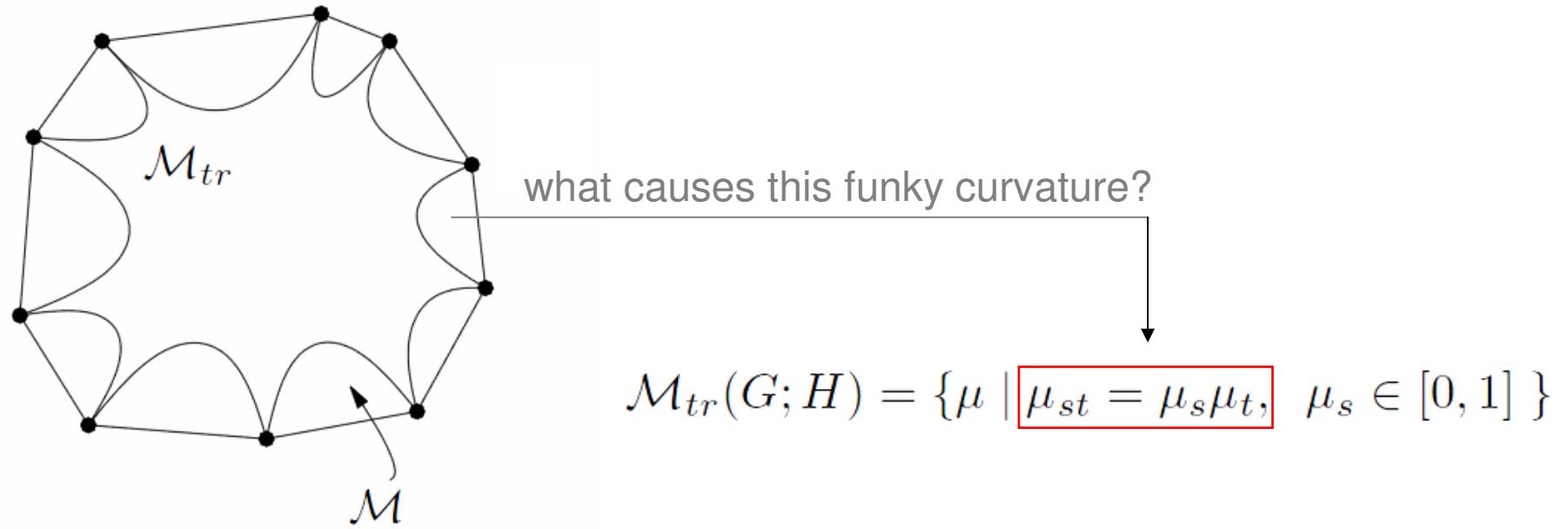
- using coordinate ascent :

$$\mu_s \leftarrow \left\{ 1 + \exp \left[- \left(\theta_s + \sum_{t \in \mathcal{N}(s)} \theta_{st} \mu_t \right) \right] \right\}^{-1}$$



Variational MF for Ising

- M_{tr} is a *non-convex* inner approximation $M_{tr} \subset M$



- optimizing over M_{tr} must then yield a *lower bound*

$$A(\theta) \geq \sup_{\tilde{\mu} \in \mathcal{M}_{tr}} \{ \langle \theta, \tilde{\mu} \rangle - A^*(\tilde{\mu}) \}$$

Factorization with Trees

- suppose we have a tree $G = (V, T)$
- useful factorization for trees

$$p(\mathbf{x}; \theta) = \prod_{s \in V} \mu_s(x_s) \prod_{(s,t) \in E} \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s) \mu_t(x_t)}$$

- entropy becomes

$$-A^*(\mu) = \mathbb{E}_\mu[-\log p_\mu(X)] = \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} I_{st}(\mu_{st})$$

- singleton terms $H_s(\mu_s) := - \sum_{x_s \in \mathcal{X}_s} \mu_s(x_s) \log \mu_s(x_s)$

- pairwise terms $I_{st}(\mu_{st}) := \sum_{(x_s, x_t) \in \mathcal{X}_s \times \mathcal{X}_t} \mu_{st}(x_s, x_t) \log \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s) \mu_t(x_t)}$
Mutual Information

Variational MF for Loopy Graphs

- pretend entropy factorizes like a tree (Bethe approximation)
- define *pseudo* marginals $\{\tau_s, s \in V\}$ $\{\tau_{st}, (s, t) \in E\}$

must impose these
normalization
and marginalization
constraints

$$\left\{ \begin{array}{l} \sum_{x_s} \tau_s(x_s) = 1 \\ \sum_{x'_t} \tau_{st}(x_s, x'_t) = \tau_s(x_s), \quad \forall x_s \in \mathcal{X}_s, \\ \sum_{x'_s} \tau_{st}(x'_s, x_t) = \tau_t(x_t), \quad \forall x_t \in \mathcal{X}_t. \end{array} \right.$$

- define local polytope $L(G)$ obeying these constraints

- note that $M(G) \subseteq L(G)$ for any G

with equality *only* for trees : $M(G) = L(G)$

Variational MF for **Loopy** Graphs

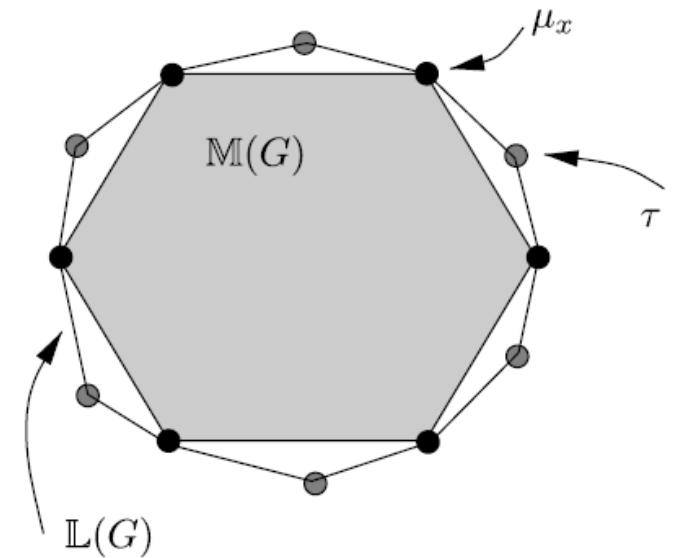
$L(G)$ is an *outer* polyhedral approximation

solving this **Bethe Variational Problem** we get the **LBP** eqs !

$$\max_{\tau \in \text{LOCAL}(G)} \left\{ \langle \theta, \tau \rangle + \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} I_{st}(\tau_{st}) \right\}$$

so fixed points of **LBP** are the stationary points of the **BVP**

$$-A_{Bethe}^*(\mu) = \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} I_{st}(\mu_{st})$$



this not only illuminates what was originally an educated “hack” (**LBP**)
but suggests new convergence conditions and improved algorithms (**TRW**)

see ICML'2008 Tutorial



Graphical models and variational methods: Message-passing, convex relaxations, and all that

Martin Wainwright

Department of Statistics, and

Department of Electrical Engineering and Computer Science,
UC Berkeley, Berkeley, CA USA

Email: `wainwrig@{stat, eecs}.berkeley.edu`

For further information (tutorial slides, papers, course lectures), see:
`www.eecs.berkeley.edu/~wainwrig/`

Summary

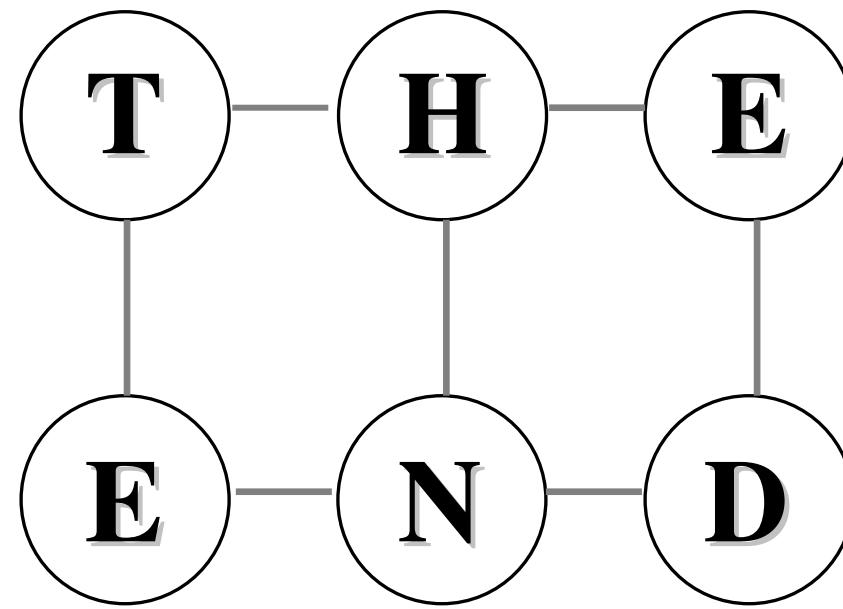
- SMF can also be cast in terms of “Free Energy” etc
- Tightening the var bound = min KL divergence
- Other schemes (e.g, “Variational Bayes”) = SMF
 - with additional conditioning (hidden, visible, parameter)
- Solving variational problem gives both μ and $A(\theta)$
- Helps to see problems through lens of Var Analysis

Matrix of Inference Methods

| | Exact | Deterministic approximation | Stochastic approximation |
|----------|---|--|---|
| | Chain (online) | Low treewidth | High treewidth |
| Discrete | BP = forwards Boyen-Koller (ADF), beam search | VarElim, Jtree, recursive conditioning | Loopy BP, mean field, structured variational, EP, graph-cuts Gibbs |
| Gaussian | BP = Kalman filter | Jtree = sparse linear algebra | Loopy BP Gibbs |
| Other | EKF, UKF, moment matching (ADF) Particle filter | EP, EM, VB, NBP, Gibbs | EP, variational EM, VB, NBP, Gibbs |

BP = Belief Propagation, EP = Expectation Propagation, ADF = Assumed Density Filtering, EKF = Extended Kalman Filter, UKF = unscented Kalman filter, VarElim = Variable Elimination, Jtree= Junction Tree, EM = Expectation Maximization, VB = Variational Bayes, NBP = Non-parametric BP

by Kevin Murphy



Probabilistic Graphical Models

Lecture 15 – Inference as Optimization

CS/CNS/EE 155
Andreas Krause

Announcements

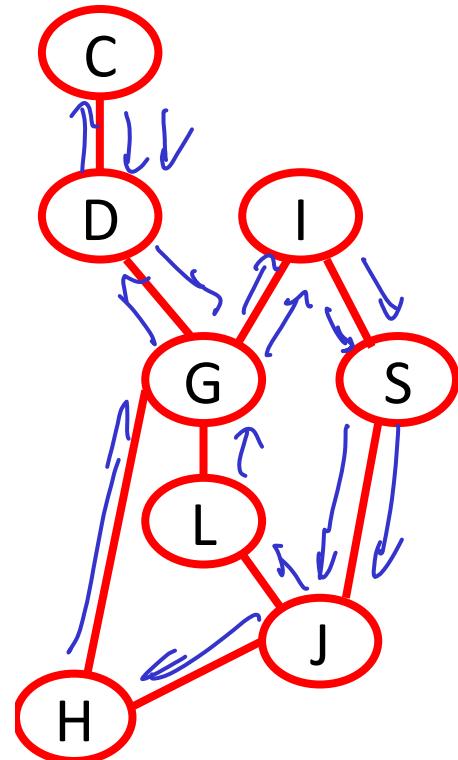
- Homework 3 due next Monday (Nov 23)
- Project poster session on Friday December 4 (tentative)
- Final writeup (8 pages NIPS format) due Dec 9

Approximate inference

- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

Loopy BP on arbitrary pairwise MNs

- What if we apply BP to a graph with loops?
 - Apply BP and hope for the best..



$$\delta_{i \rightarrow j}(X_j) = \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \rightarrow i}(x_i)$$

- Will not generally converge.. 😞
- If it converges, will not necessarily get correct marginals 😞

- However, in practice, answers often still useful!

Approximate inference

- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation (today!)
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
 - Assumed density filtering / expectation propagation
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

Variational approximation

- Graphical model with intractable (high-treewidth) joint distribution $P(X_1, \dots, X_n)$
- Want to compute posterior distributions

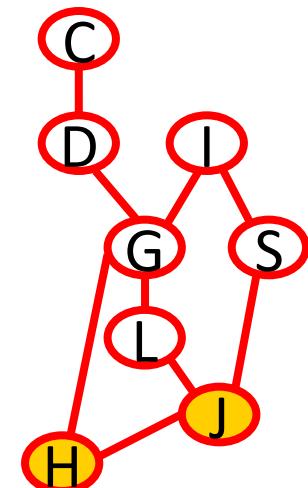
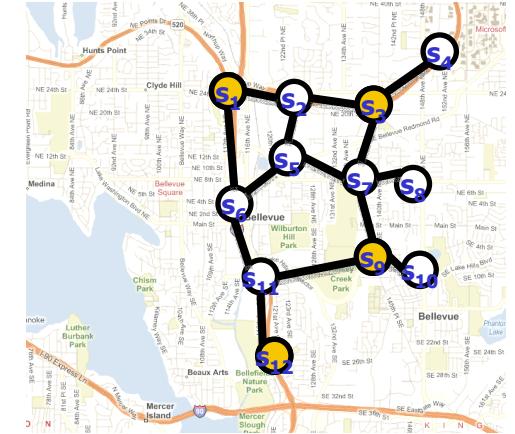
$$\begin{aligned} & P(X_3, X_4 \mid X_1 = x_1, X_7 = x_2) \\ & \propto P(X_3, X_4, X_1 = x_1, X_7 = x_2) = \sum_{X_5} \sum_{X_6} \dots P(X_1, X_2, \dots, X_n) \end{aligned}$$

- Computing posterior exactly is intractable
- **Key idea:** Approximate posterior with *simpler* distribution that's *as close* to P as possible

Why should we hope that we can find a simple approximation?

- Prior distribution is complicated
 - Need to describe all possible states of the world (and relationships between variables)
- Posterior distribution is often simple:
 - Have made many observations → less uncertainty
 - Variables can become “almost independent”
- For now: Represent posterior as undirected model (and instantiate observations)

$$P(X_1, \dots, X_n \mid obs) = \frac{1}{Z} \prod_j \Psi_j(\mathbf{C}_j)$$



Variational approximation

- **Key idea:** Approximate posterior with simpler distribution that's as close as possible to P
 - What is a “simple” distribution?
 - What does “as close as possible” mean?
- **Simple** = efficient inference
 - Typically: factorized (fully independent, chain, tree, ...)
 - Gaussian approximation
- **As close as possible** = KL divergence (typically)
 - Other distance measures can be used too, but more challenging to compute

Kullback-Leibler (KL) divergence

- Distance between distributions

$$D(P||Q) = \int P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})} d\mathbf{x}$$

- Properties:
 - $D(P || Q) \geq 0$
 - $P(x)=Q(x)$ almost everywhere $\Leftrightarrow D(P || Q) = 0$
- In general, $D(P || Q) \neq D(Q || P)$

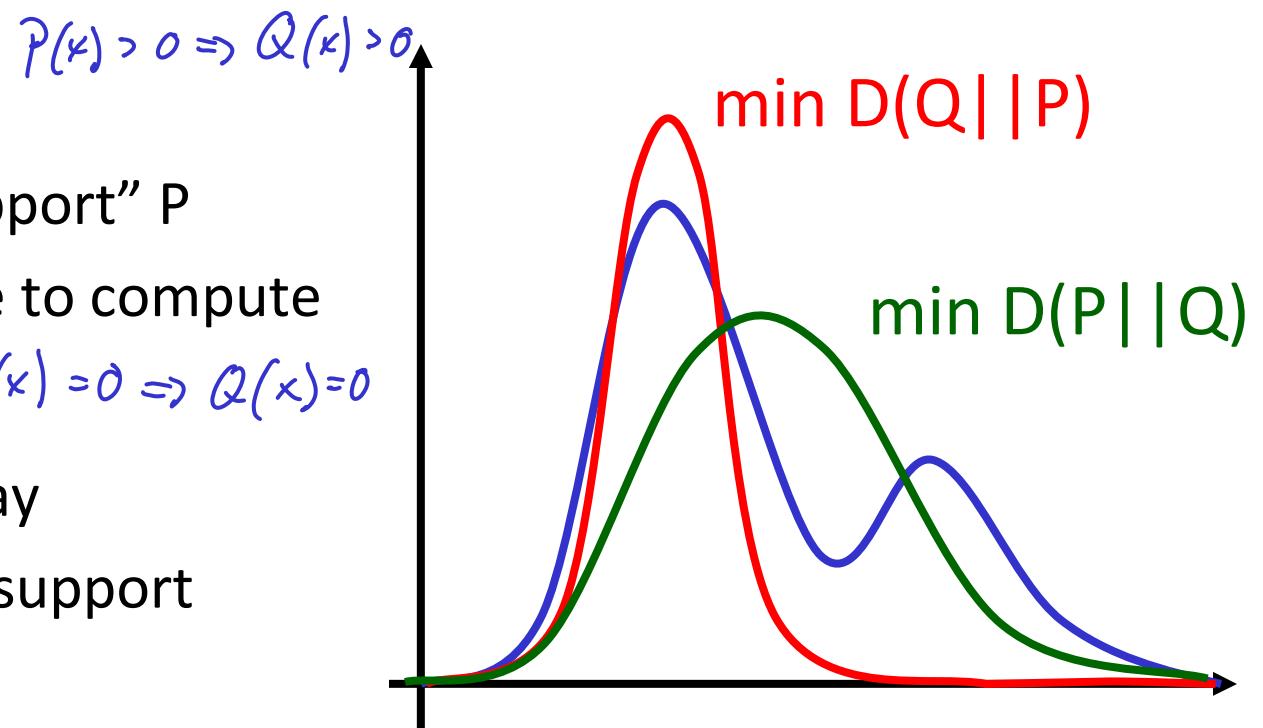
- P determines when difference is important

$$P(x) = 0, Q(x) \neq 0 \Rightarrow 0 \cdot \log \frac{0}{\varepsilon} = 0$$

$$P(x) = \varepsilon, Q(x) > 0 \Rightarrow \varepsilon \cdot \log \frac{\varepsilon}{0} = \infty$$

Finding simple approximate distributions

- KL divergence not symmetric; need to choose directions
- P: true distribution; Q: our approximation
- $D(P \parallel Q)$
 - The “right” way
 - Q chosen to “support” P
 - Often intractable to compute
- $D(Q \parallel P)$
 - The “reverse” way
 - Underestimates support (overconfident)
 - Will be tractable to compute
- Both special cases of α -divergence



“Simple” distributions

- Simplest distribution: Q fully factorized

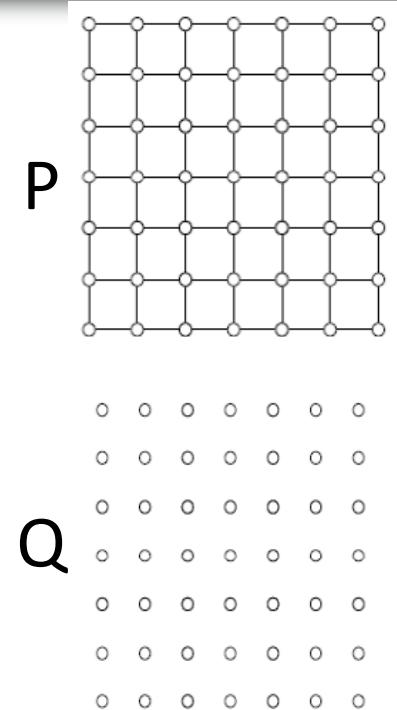
- $Q(X_1, \dots, X_n) = \prod_i Q_i(X_i)$

- M = {Q: Q fully factorized}
= {Q: $Q(X) = \prod_i Q_i(X_i)$ }

$$Q^* = \underset{Q \in \mathcal{M}}{\operatorname{argmin}} D(Q || P)$$

$D(P || Q)$

- Can also find more structured approximations
 - Chains: $Q(X_1, \dots, X_n) = \prod_i Q_i(X_i | X_{i-1})$
 - Trees
 - Any distributions one can do efficient inference on

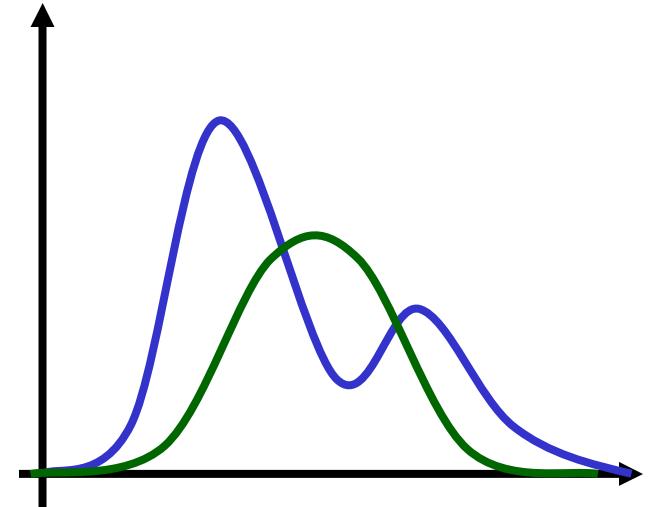


Mean field approximation the “right way”

$$\begin{aligned} D(P \parallel Q) &= \sum_x P(x) \log \frac{P(x)}{Q(x)} \\ &= \underbrace{\sum_x P(x) \log P(x)}_{\text{constant}} - \underbrace{\sum_x P(x) \log Q(x)}_{(*)} \end{aligned}$$

$$\begin{aligned} (*) &= \sum_x P(x) \log \prod_i Q_i(x_i) \\ &= \sum_x P(x) \sum_i \log Q_i(x_i) \\ &= \sum_i \sum_x P(x_i) \log Q_i(x_i) \\ &= \sum_i \left(\sum_{x_i} P(x_i) \log Q_i(x_i) \right) \left(\sum_{x_{i+1}, \dots, x_m} P(x) \right) \end{aligned}$$

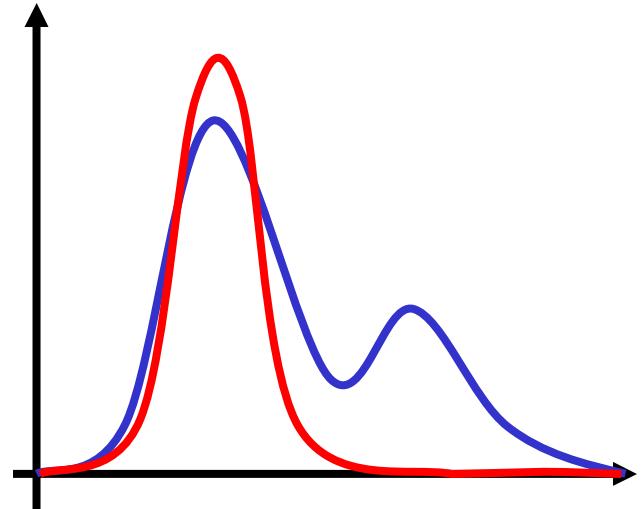
Need $P(x_i)$
Intractable!



Mean field approximation the reverse way

$$D(Q \parallel P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)}$$

$$= \underbrace{\sum_x Q(x) \log Q(x)}_{(1)} - \underbrace{\sum_x Q(x) \log P(x)}_{(2)}$$



$$(1) = -H(Q) = \sum_x Q(x) \log \prod_i Q_i(x_i)$$

$$= \sum_i \sum_x Q(x) \log Q_i(x_i)$$

$$= \sum_i \sum_{x_{\neq i}} Q_i(x_i) \log Q_i(x_i) \underbrace{\left(\sum_{x_{\neq i}} Q(x_{\neq i}) \right)}_{= 1} = -\sum_i H(Q_i)$$

$$(2) = \sum_x Q(x) \log \frac{1}{Z} \prod_i \gamma_i(c_i) = -\underbrace{\sum_x Q(x) \log Z}_{-\log Z} + \underbrace{\sum_x Q(x) \sum_i \log \gamma_i(c_i)}_{(2')}$$

$$(2') = \sum_i \sum_x Q(x) \log \underbrace{\gamma_i(c_i)}_{\text{dys or small} \atop \# \text{ward}} = \sum_i \sum_{c_i} \underbrace{Q(c_i)}_{\text{Need marginal for } c_i} \log \gamma_i(c_i)$$

Reverse KL for fully factorized case

$$D(Q||P) = - \sum_i \underbrace{\sum_x Q(x) \log \Psi_i(x)}_{\mathbb{E}_Q[\log \Psi_i]} - \sum_i H(Q_i) + \underbrace{\ln Z}_{\text{constant}}$$

$$\underbrace{\ln Z}_{\text{constant}} = \underbrace{D(Q||P)}_{\rightarrow \min} + \underbrace{\sum_i H(Q_i)}_{\rightarrow \max} + \sum_i \mathbb{E}_Q[\log \Psi_i]$$
$$F(Q; P)$$

KL and the partition function

Suppose $P(X_1, \dots, X_n) = Z^{-1} \prod_i \Psi_i(C_i)$ is Markov Network

Theorem: For any Q (not necessarily fully factorized)

$$\ln Z = \cancel{F[P; Q]} + D(Q||P)$$

Hereby, $F[P; Q]$ is the following energy functional

$$F[P; Q] = \sum_i \mathbb{E}_{\color{red}Q}[\ln \Psi_i] + H(Q)$$

Reverse KL vs. log-partition function

$$\ln Z = \cancel{F[P; Q]} + \cancel{D(Q||P)} \geq 0 \quad F[P; Q] = \sum_i \mathbb{E}[\ln \Psi_i] + H(Q)$$

Maximizing energy functional \Leftrightarrow Minimizing reverse KL

Corollary:

Energy function is lower bound on log partition function

$$P(x) = \frac{1}{Z} \prod_i \Psi_i(\zeta_i)$$

$$P(x) \leq \frac{1}{F(P, Q)} \prod_i \Psi_i(\xi_i)$$

Implies upper bound on event probabilities!

Optimizing for mean field approximation

- Want to solve $\max_Q F[P; Q] = \max_Q \sum_j \mathbb{E}_Q[\ln \Psi_j] + \sum_i H(Q_i)$

$$\text{s.t. } \sum_{x_i} Q_i(x_i) = 1$$

- Solved via Lagrange multipliers: There exist $\lambda_1, \dots, \lambda_n$
s.t. optimization of $(*)$ is equivalent to

$$\max_Q \sum_j \mathbb{E}_Q[\ln \Psi_j] + \sum_j H(Q_j) + \sum_j \lambda_j \left[\sum_{x_j} Q_j(x_j) - 1 \right]$$

Differentiate and set to 0!

↙ Minimum, Maximum or saddle point

Theorem: Q stationary point iff for each i and x_i :

$$Q_i(x_i) = \frac{1}{Z_i} \exp\left(\sum_j \mathbb{E}[\ln \Psi_j \mid x_i]\right)$$

Fixed point iteration for MF

- Initialize factors $Q^{(0)}_i$ arbitrarily; $t=0$
- Until converged, do
 - $t \leftarrow t+1$
 - For each variable i and each assignment x_i , do

$$Q_i(x_i)^{(t+1)} = \frac{1}{Z_i} \exp\left(\sum_j \underbrace{\mathbb{E}_{Q^{(t)}} [\ln \Psi_j \mid x_i]}_{Z_i}\right)$$

$$Z_i = \sum_{x_i} Q_i(x_i)$$

- Guaranteed to converge! 😊
- Gives both approx. distribution Q and lower bound on $\ln Z$
- Can get stuck in local optimum 😞

Computing updates

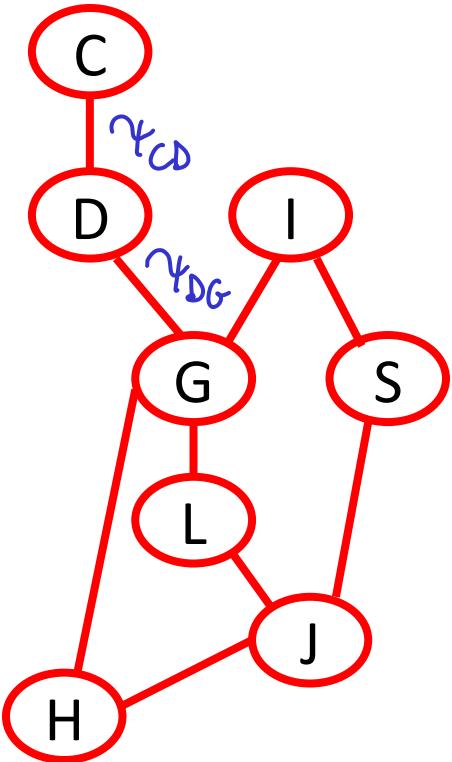
Need to compute

$$Q_i(x_i)^{(t+1)} = \frac{1}{Z_i} \exp\left(\sum_j \underline{\mathbb{E}_{Q^{(t)}} [\ln \Psi_j \mid x_i]}\right)$$

Must compute expected log potentials: $\mathbb{E}_Q [\ln \Psi_j \mid x_i]$

$$\begin{aligned} (\star) &= \sum_x Q^{(t)}(x \mid x_i) \ln \underbrace{\Psi_j(x)}_{C_j \text{ in } x_i} = \sum_{C_j \text{ in } x_i} Q^{(t)}(C_j \mid x_i) \ln \Psi_j(C_j) \\ &= \Psi_j(C_j) &= \prod_{k \in C_j \setminus \{x_i\}} Q_k(x_k) \end{aligned}$$

Example iteration



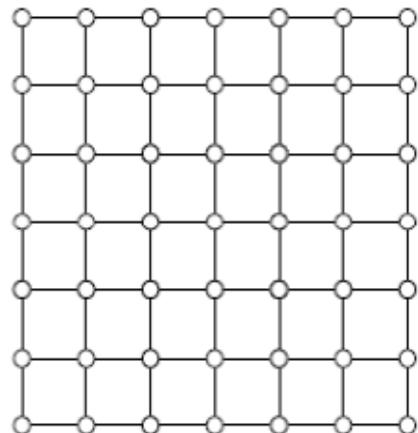
$$\begin{aligned}
 & \mathbb{E}_Q [\ln \psi_{DG} | x_D=1] \\
 &= \underbrace{\sum_{x_G} Q(x_G | x_D=1) \ln \psi_{DG}(x_D=1, x_G)}_{= Q_G(x_G)} \\
 & Q_D(x_D=1) = \frac{1}{Z_D} \exp \left(\mathbb{E}_Q [\ln \psi_{DG} | x_D=1] + \mathbb{E}_Q [\ln \psi_{CG} | x_D=1] \right)
 \end{aligned}$$

Structured mean field

Goal of variational inference:

Approximate complex distribution by simple distribution

True dist.



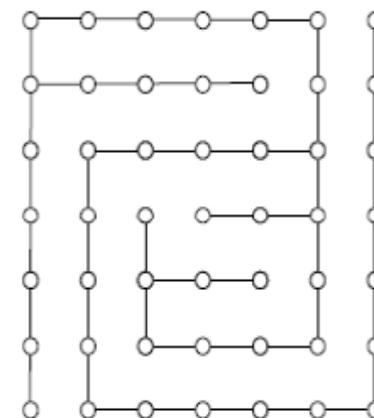
$$p(x) \propto \prod_c \phi_c(x_c)$$

**Fully-factorized
mean field**



$$q(x) \propto \prod_i q_i(x_i)$$

**Structured
mean field**



$$q(x) \propto q_A(x_A) q_B(x_B)$$

Structured mean-field approximations

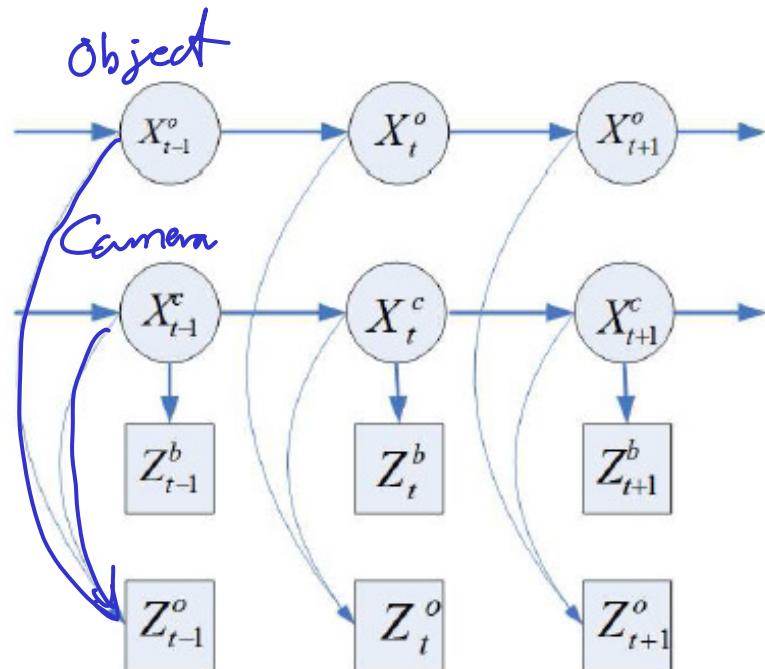
- Can get better approximations using **structured approximations**:

$$\max_{Q \in \mathcal{M}} F[P; Q] = \max_{Q \in \mathcal{M}} \sum_j \mathbb{E}_Q[\ln \Psi_j] + H(Q)$$

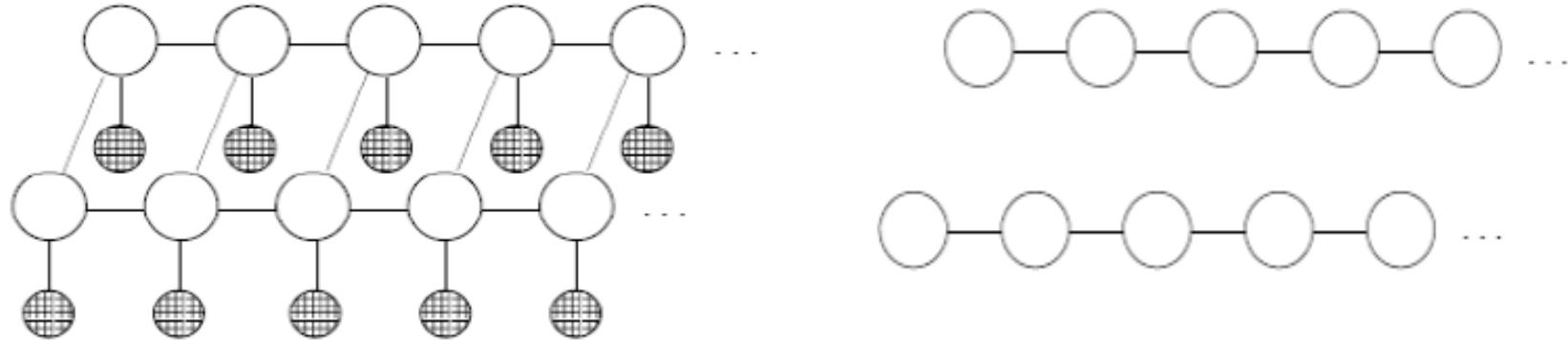
- Only need to be able to compute energy functional
- Can do whenever we can perform efficient inference in Q (e.g., chains, trees, low-treewidth models)
 - Update equations look similar as for fully-factorized case (see reading)

Example: Factorial HMM

- Simultaneous tracking and camera registration
- State space decomposed into object location and camera parameters



Variational approximations for FHMMs



$$\max_{Q \in \mathcal{M}} F[P; Q] = \max_{Q \in \mathcal{M}} \sum_j \mathbb{E}_Q[\ln \Psi_j] + H(Q)$$

- Approximate posterior by independent chains

$$\mathcal{M} = \left\{ Q : Q(\mathbf{X}) = \prod_c \prod_t Q_{c,t}(X_{c,t} | X_{c,t-1}) \right\}$$

Summary: Variational inference

- Approximate complex (intractable) distribution by simpler distribution that is “as close as possible”
- **Simple** = tractable (efficient inference)
- **Closeness** = Reverse KL (efficient to compute)
- Interpretation: Optimize **lower bound** on the log-partition function
 - Implies upper bound on event probabilities
- Efficient algorithm that’s guaranteed to converge (in contrast to Loopy BP..), but possibly to local optimum

Approximate inference

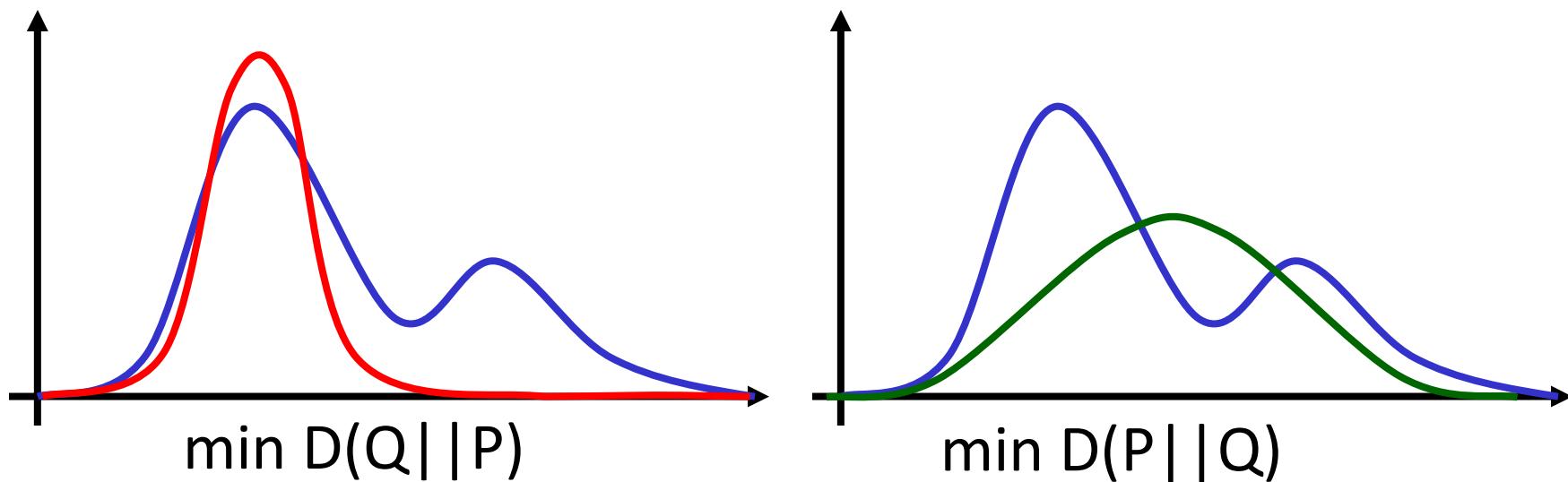
- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation (today!)
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
 - Assumed density filtering / expectation propagation
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

KL-divergence the “right” way:

- Find distribution $Q^* \in \mathcal{M}$:

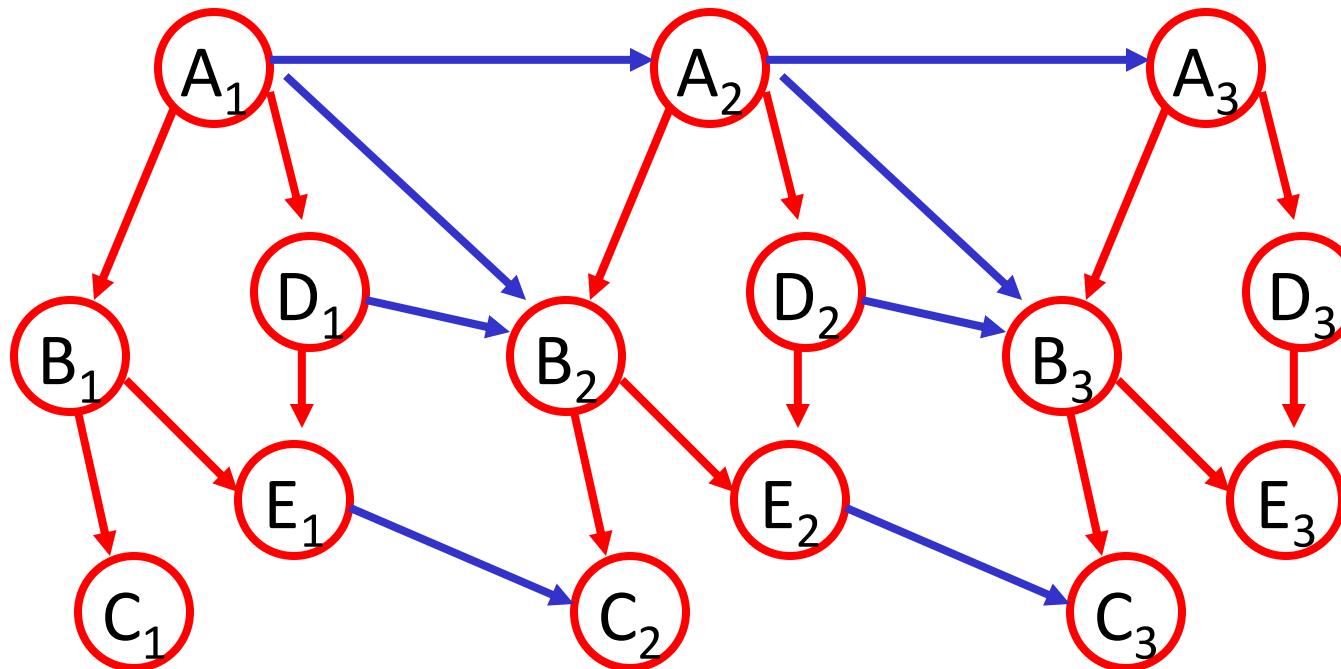
$$Q^* = \underset{Q \in \mathcal{M}}{\operatorname{argmin}} D(P || Q)$$

- In some applications, can compute $D(P || Q)$
 - Important example: Assumed density filtering in DBNs



Recall: Dynamic Bayesian Networks

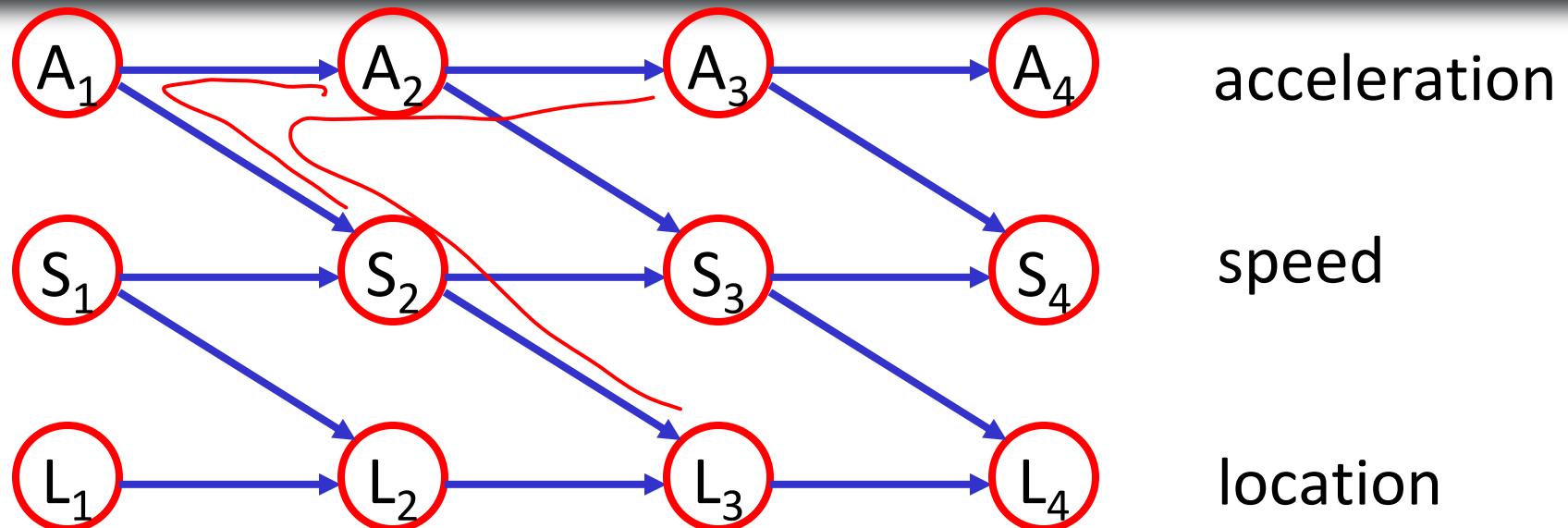
- At every timestep have a Bayesian Network



$$S_t = \{A_1, B_1, \dots, E_t\}$$

- Variables at each time step t called a “slice” S_t
- “Temporal” edges connecting S_{t+1} with S_t

Flow of influence in DBNs



$$A_1 \perp S_1 \checkmark$$

$$A_2 \perp S_2 \times$$

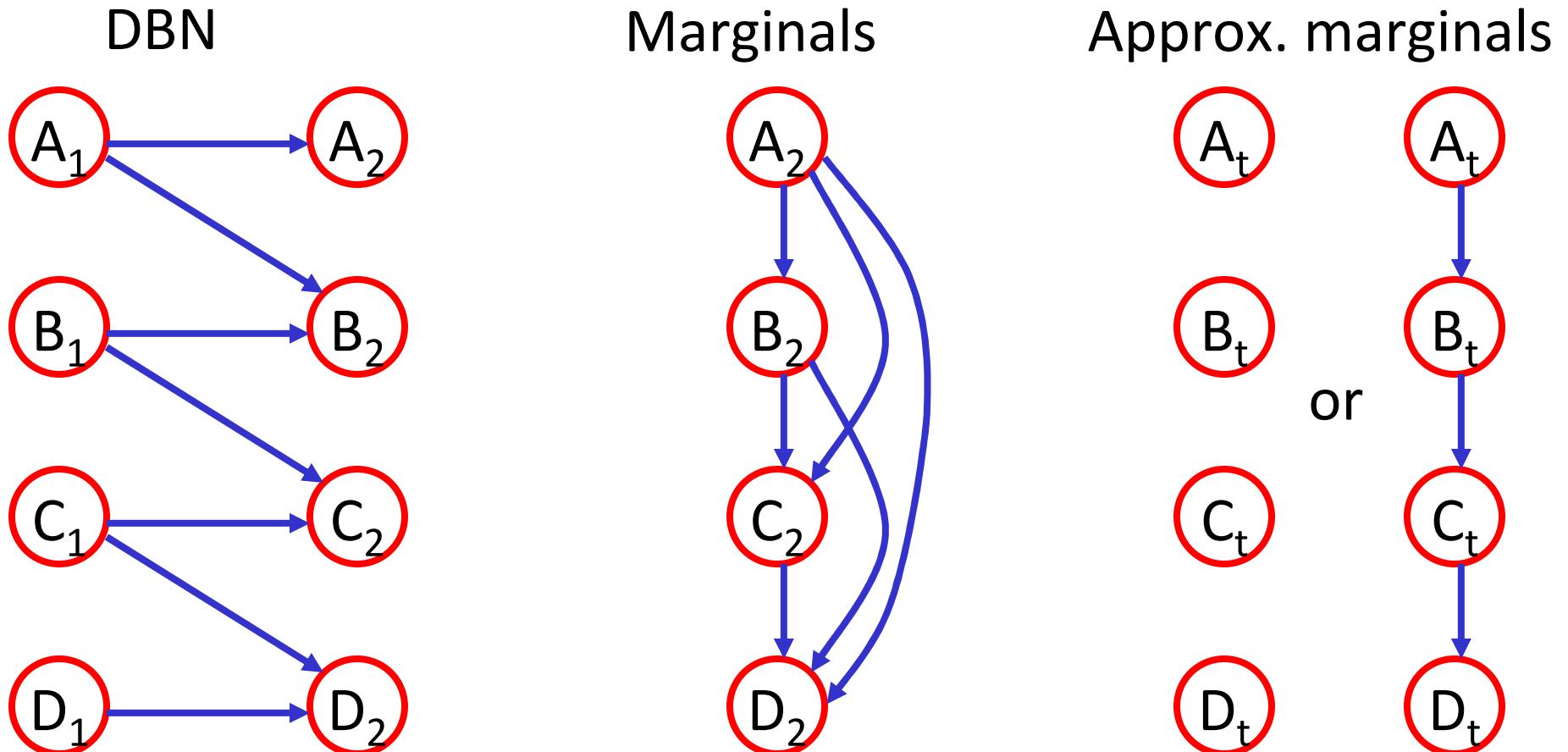
$$A_1 \perp L_1 \checkmark$$

$$A_2 \perp L_2 \checkmark$$

$$A_3 \perp L_3 \times$$

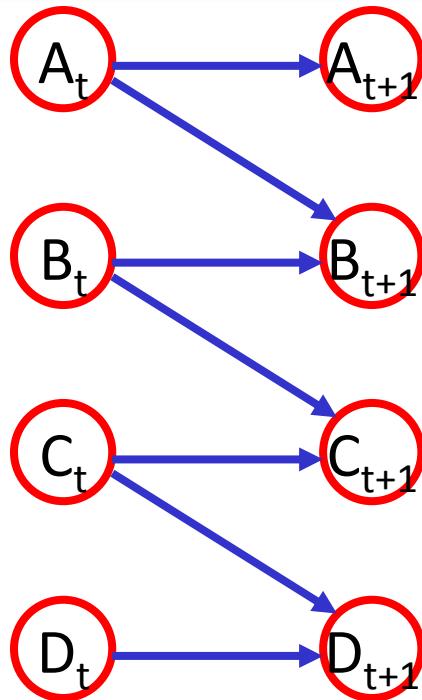
- Can we do efficient filtering in BNs?

Approximate inference in DBNs?



Want to find **tractable** approximation to marginals
that's **as close** to true marginals as possible

Assumed Density Filtering



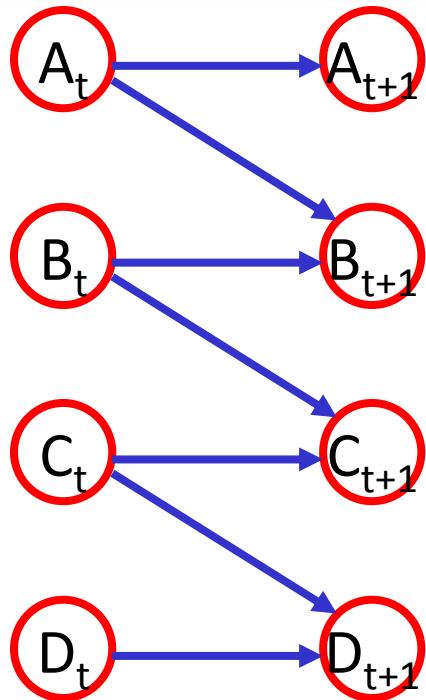
$$\begin{aligned}
 D(P(S_{t+1}) || Q(S_{t+1})) &= \sum_{S_{t+1}} P(S_{t+1}) \log \frac{P(S_{t+1})}{Q(S_{t+1})} \\
 &= \text{const} - \sum_{S_{t+1}} P(S_{t+1}) \log Q(S_{t+1})
 \end{aligned}$$

(x)

- Assume distribution $P(S_t)$ for slice t factorizes
- $P(S_{t+1})$ is fully connected ☹
- Want to compute best-approximation Q^* for $P(S_{t+1})$

$$Q^* = \operatorname{argmin} D(P || Q)$$

Assumed Density Filtering

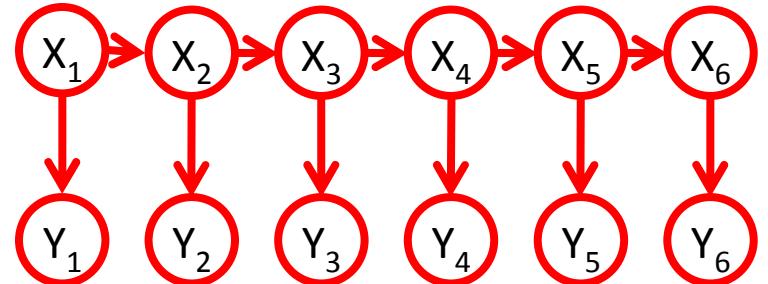


$$\begin{aligned}
 & \sum_{S_{t+1}} P(S_{t+1}) \log \underbrace{Q(S_{t+1})}_{= \prod_i Q_i(S_{i,t+1})} \\
 & = \sum_i \sum_{S_{i,t+1}} P(S_{i,t+1}) \log Q_i(S_{i,t+1}) \\
 & = \sum_i \sum_{S_{i,t+1}} \underbrace{P(S_{i,t+1})}_{\text{E.g., } P(A_{t+1})} \log Q_i(S_{i,t+1}) \\
 & \quad \text{Can compute expectations efficiently}
 \end{aligned}$$

Get optimal Q^* by setting $Q_i^*(S_{i,t+1}) = P(S_{i,t+1})$

Recall: Bayesian filtering

- Start with $P(X_1)$
- At time t
 - Assume we have $P(X_t | y_{1..t-1})$
 - Condition: $P(X_t | y_{1..t})$



$$P(X_t | y_{1..t}) \propto P(X_t | y_{1..t-1}) \underbrace{P(Y_t | X_t, y_{1..t-1})}_{\text{cond. ind. } P(Y_t | X_t)}$$

- Prediction: $P(X_{t+1}, X_t | y_{1..t})$

$$P(X_{t+1}, X_t | y_{1..t}) = P(X_t | y_{1..t}) \cdot \underbrace{P(X_{t+1} | X_t, y_{1..t})}_{= P(X_{t+1} | X_t)}$$

- Marginalization: $P(X_{t+1} | y_{1..t})$

$$P(X_{t+1} | y_{1..t}) = \sum_{X_K} P(X_{t+1}, X_t | y_{1..t})$$

Assumed Density Filtering

- Start with $P(S_1)$
- At every time step t : tractable approximation Q_t
 $Q_t(S_t) \approx P(S_t | O_{1:t-1})$
- Condition on observation $O_t \subseteq S_t$: $Q_t(S_t | O_t)$
- Predict: multiply transition model to get $Q_t(S_{t+1}, S_t | O_t)$
$$Q_t(S_{t+1}, S_t | O_t) = Q_t(S_t | O_t) P(S_{t+1} | S_t)$$
- Marginalize S_t
 - This is intractable (connects all variables in S_{t+1})
 - Approximate $Q_t(S_{t+1} | O_t)$ by Q^* s.t.
$$Q^* = \operatorname{argmin}_Q D(Q_t(S_{t+1}) || Q(S_{t+1}))$$
 - This is done by matching moments:
for discrete models, ensure that $Q_{t+1}(s_{t+1}) = Q_t(s_{t+1} | o_t)$

Summary of Assumed Density Filtering

- Variational inference technique for dynamical Bayesian Networks
- Find tractable approximation for each time slice that minimizes KL divergence (in the “right” way)
- Can show that errors don’t add up too much
- Examples:
 - Tractable inference in DBNs
 - Unscented Kalman Filter

Summary: Inference as optimization

- Approximate intractable distribution by a **tractable** one
- **Optimize parameters** of the distribution to make approximation as tight as possible
- Common distance measure: KL-divergence (both ways)
 - Special case of α -divergence
- Can get upper bounds on event probabilities, etc.

Probabilistic Graphical Models

Lecture 16 – Sampling

CS/CNS/EE 155
Andreas Krause

Announcements

- Homework 3 due today
- Project poster session on Friday December 4 (tentative)
- Final writeup (8 pages NIPS format) due Dec 9

Approximate inference

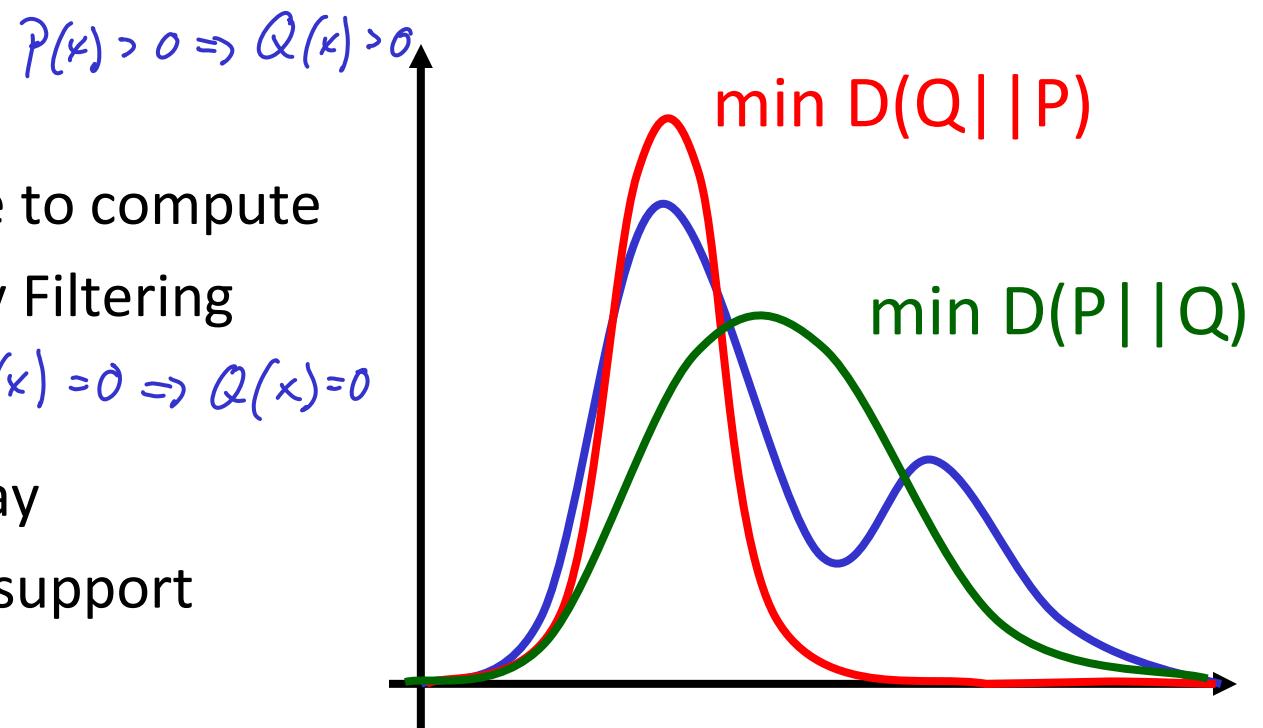
- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation (today!)
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
 - Assumed density filtering / expectation propagation
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

Variational approximation

- **Key idea:** Approximate posterior with simpler distribution that's as close as possible to P
 - What is a “simple” distribution?
 - What does “as close as possible” mean?
- **Simple** = efficient inference
 - Typically: factorized (fully independent, chain, tree, ...)
 - Gaussian approximation
- **As close as possible** = KL divergence

Finding simple approximate distributions

- KL divergence not symmetric; need to choose directions
- P: true distribution; Q: our approximation
- $D(P \parallel Q)$
 - The “right” way
 - Often intractable to compute
 - Assumed Density Filtering
- $D(Q \parallel P)$
 - The “reverse” way
 - Underestimates support (overconfident)
 - Mean field approximation
- Both special cases of α -divergence



Approximate inference

- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation (today!)
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
 - Assumed density filtering / expectation propagation
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

Sampling based inference

- So far: deterministic inference techniques
 - Loopy belief propagation
 - (Structured) mean field approximation
 - Assumed density filtering
- Will now introduce stochastic approximations
 - Algorithms that “randomize” to compute expectations
 - In contrast to the deterministic methods, can sometimes get approximation guarantees
 - More exact, but slower than deterministic variants

Computing expectations

- Often, we're not necessarily interested in computing marginal distributions, but certain expectations:
- Moments (mean, variance, ...)

$$\mathbb{E}_P[X^k] = \int x^k P(x) dx$$

- Event probabilities

$$P(\underline{X > c}) = \mathbb{E}_P[\underline{I_{X>c}}] = \int [x > c] P(x) dx$$

Sample approximations of expectations

- $\underline{x_1, \dots, x_N}$ samples from RV X
- Law of large numbers:

$$\underline{\mathbb{E}_P[f(X)]} = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N f(x_i)$$

- Hereby, the convergence is with probability 1
(almost sure convergence)
- Finite samples: $\underline{\mathbb{E}_P[f(X)]} \approx \frac{1}{N} \sum_{i=1}^N f(x_i)$

How many samples do we need?

- Hoeffding inequality

Suppose f is bounded in $[0, C]$. Then

$$P\left(\left|\mathbb{E}_P[f(X)] - \underbrace{\frac{1}{N} \sum_{i=1}^N f(x_i)}_{\text{sample mean}}\right| > \varepsilon\right) \leq \underbrace{2 \exp(-2N\varepsilon^2/C^2)}_{\text{error probability}}$$

- Thus, probability of error decreases exponentially in N !

want error ε with probability $1-\delta$

$$2 \exp(-2N\varepsilon^2/C^2) < \delta$$

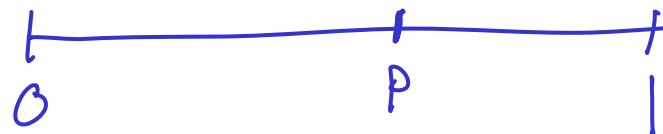
$$\begin{aligned} 2N\varepsilon^2/C^2 &> \log \frac{2}{\delta} \\ N &> \frac{1}{2} \cdot \frac{1}{\varepsilon^2} \cdot C^2 \cdot \log \frac{2}{\delta} \end{aligned}$$

- Need to be able to draw samples from P

Sampling from a Bernoulli distribution

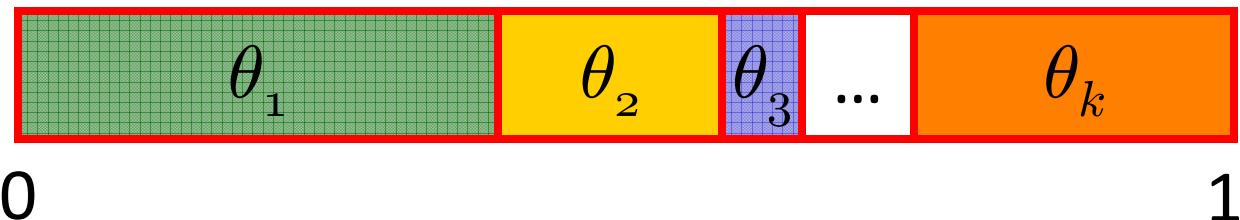
- $X \sim \text{Bernoulli}(p)$
- How can we draw samples from X ?

Assume we can draw uniform distribution $[0,1]$



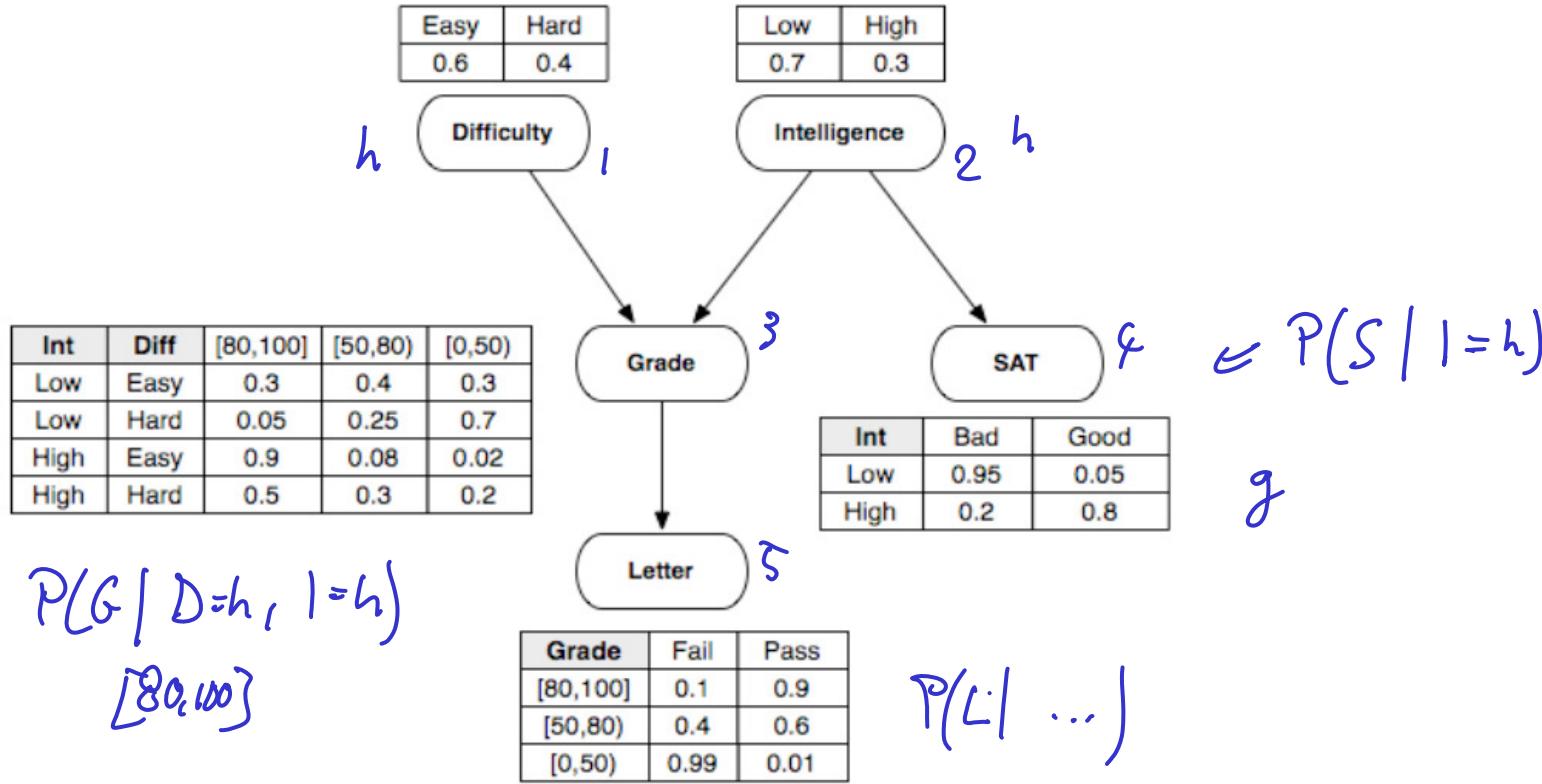
Sampling from a Multinomial

- $X \sim \text{Mult}([\theta_1, \dots, \theta_k])$
where $\theta_i = P(X=i)$; $\sum_i \theta_i = 1$



- Function $g: [0,1] \rightarrow \{1, \dots, k\}$ assigns state $g(x)$ to each x
- Draw sample from uniform distribution on $[0,1]$
- Return $g^{-1}(x)$

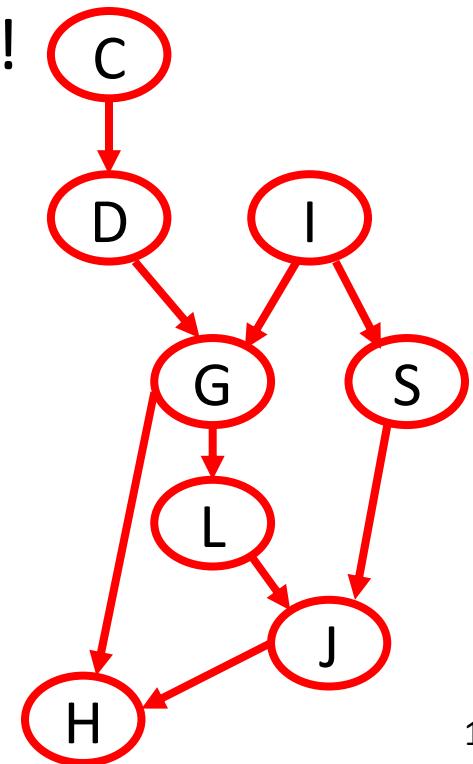
Forward sampling from a BN



Monte Carlo sampling from a BN

- Sort variables in topological ordering X_1, \dots, X_n
- For $i = 1$ to n do
 - Sample $x_i \sim P(X_i \mid X_1=x_1, \dots, X_{i-1}=x_{i-1})$ $= P(X_i \mid \text{Pa}_{X_i})$

- Works even with high-treewidth models!



Computing probabilities through sampling

- Want to estimate probabilities

- Draw N samples from BN

- Marginals

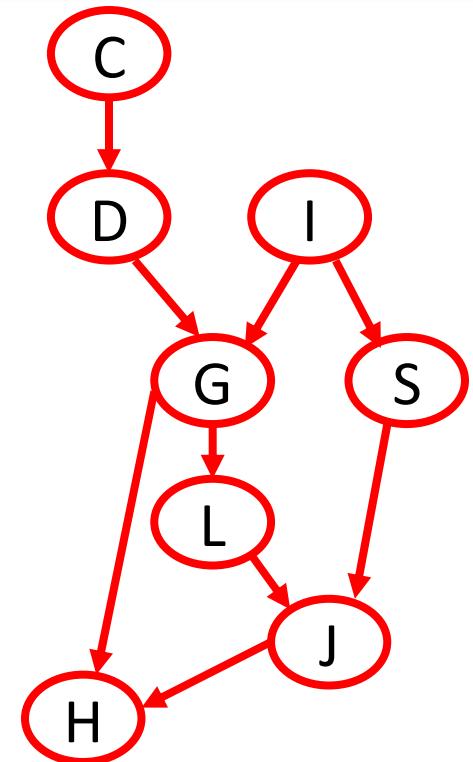
$$P(H=y) = \mathbb{E}_p [I_{H=y}] = \sum_x P(x) \cdot I_{H=y}(x)$$

$\approx \frac{1}{N} \sum_{i=1}^N I_{H=y}(x^{(i)}) = \frac{\text{Count}(H=y)}{N}$

- Conditionals

$$P(D=h | H=m) = \frac{P(D=h, H=m)}{P(H=m)} = \frac{\text{Count}(D=h, H=m)}{\text{Count}(H=m)}$$

Rejection sampling



Rejection sampling

- Collect samples over all variables

$$\widehat{P}(\mathbf{X}_A = \mathbf{x}_A \mid \mathbf{X}_B = \mathbf{x}_B) \approx \frac{\text{Count}(\mathbf{x}_A, \mathbf{x}_B)}{\text{Count}(\mathbf{x}_B)}$$

- Throw away samples that disagree with \mathbf{x}_B
- Can be problematic if $P(\mathbf{X}_B = \mathbf{x}_B)$ is rare event

Sample complexity for probability estimates

- Absolute error:

$$\text{Prob}\left(\left|\hat{P}(\mathbf{x}) - P(\mathbf{x})\right| > \varepsilon\right) \leq 2 \exp(-2N\varepsilon^2)$$

- Relative error:

$$\text{Prob}\left(\hat{P}(\mathbf{x}) < (1 + \varepsilon)P(\mathbf{x})\right) \leq 2 \exp(-NP(\mathbf{x})\varepsilon^2/3)$$

Estimating low probability events
is hard

Sampling from rare events

- Estimating conditional probabilities $P(X_A \mid X_B = x_B)$ using rejection sampling is hard!
 - The more observations, the unlikelier $P(X_B = x_B)$ becomes
- Want to directly sample from posterior distribution!

Sampling from intractable distributions

- Given unnormalized distribution

$$P(X) \propto Q(X) = P(X, X_{obs} = x_{obs})$$

$$P(X_A | X_B = x_B) \propto P(X_A, X_B = x_B)$$

- Q(X) efficient to evaluate, but normalizer intractable
- For example, $Q(X) = \prod_j \Psi(C_j)$
- Want to sample from $P(X) = \frac{1}{Z} Q(X)$
- Ingenious idea:**
Can create Markov chain that is efficient to simulate
and that has stationary distribution $P(X)$

Markov Chains

- A Markov chain is a sequence of RVs, X_1, \dots, X_N, \dots with
 - Prior $P(X_1)$
 - Transition probabilities $\underline{P(X_{t+1} | X_t)}$
- A Markov Chain with $P(X_{t+1} | X_t) > 0$ has a unique **stationary distribution** $\mu(x)$, such that for all x
$$\lim_{N \rightarrow \infty} P(X_N = x) = \underline{\mu(x)}$$



The stationary distribution is independent of $P(X_1)$

Simulating a Markov Chain

- Can sample from a Markov chain as from a BN:
- Sample $x_1 \sim P(X_1)$
- Sample $x_2 \sim P(X_2 \mid X_1=x_1)$
- ...
- Sample $x_N \sim P(X_N \mid X_{N-1}=x_{N-1})$
- ...
- If simulated “sufficiently long”, sample X_N is drawn from a distribution “very close” to stationary distribution μ

Markov Chain Monte Carlo

- Given an unnormalized distribution $Q(x)$
- Want to design a Markov chain with stationary distribution

$$\pi(x) = 1/Z Q(x)$$

- Need to specify transition probabilities $P(x | x')$!

Detailed balance equation

- A Markov Chain satisfies the **detailed balance equation** for unnormalized distribution Q if for all x, x' :

$$Q(x) P(x' | x) = Q(x') P(x | x')$$

- In this case, the Markov chain has stationary distribution $\frac{1}{Z} Q(x)$

$$\underbrace{\frac{1}{Z} Q(x)}_{\mu(x)} = \frac{1}{Z} \sum_{x'} P(x' | x) Q(x) = \frac{1}{Z} \sum_{x'} Q(x') P(x | x') = \sum_{x'} \mu(x') P(x | x')$$

Designing Markov Chains

1) Proposal distribution $R(X' | X)$

- Given $X_t = x$, sample “proposal” $x' \sim R(X' | X=x)$
- Performance of algorithm will strongly depend on R

2) Acceptance distribution:

- Suppose $X_t = x$
- With probability $\alpha = \min \left\{ 1, \frac{Q(x')R(x | x')}{Q(x)R(x' | x)} \right\}$
set $X_{t+1} = x'$
- With probability $1-\alpha$, set $X_{t+1} = x$

Theorem [Metropolis, Hastings]: The stationary distribution is $Z^{-1} Q(x)$

- Proof: Markov chain satisfies detailed balance condition!

MCMC for Graphical Models

- Random vector $X=(X_1, \dots, X_n)$ is high-dimensional
- Need to specify proposal distributions $R(x' | x)$ over such random vectors
 - x^{old} : old state
 - x' : proposed state, $x' \sim R(X' | X=x)$
- Examples

$$- R(x'|x) = R(x')$$

$$- R(x'|x) \quad x'_i \sim R_i(x'_i | x) \\ x'_{\neg i} = x_{\neg i}$$

Gibbs sampling

- Start with initial assignment $\underline{x^{(0)}}$ to all variables
- For $\underline{t = 1}$ to ∞ do
 - Set $x^{(t)} = \underline{x^{(t-1)}}$
 - For each variable X_i
 - Set $v_i = \text{values of all } x^{(t)} \text{ except } x_i$
 - Sample $x_i^{(t)}$ from $P(X_i | v_i)$
- Gibbs sampling satisfies detailed balance equation for P
- **Key challenge:** Computing conditional distributions $P(X_i | v_i)$

Computing $P(X_i \mid v_i)$

$$Q(x) = \prod_i \psi_i(c_i)$$

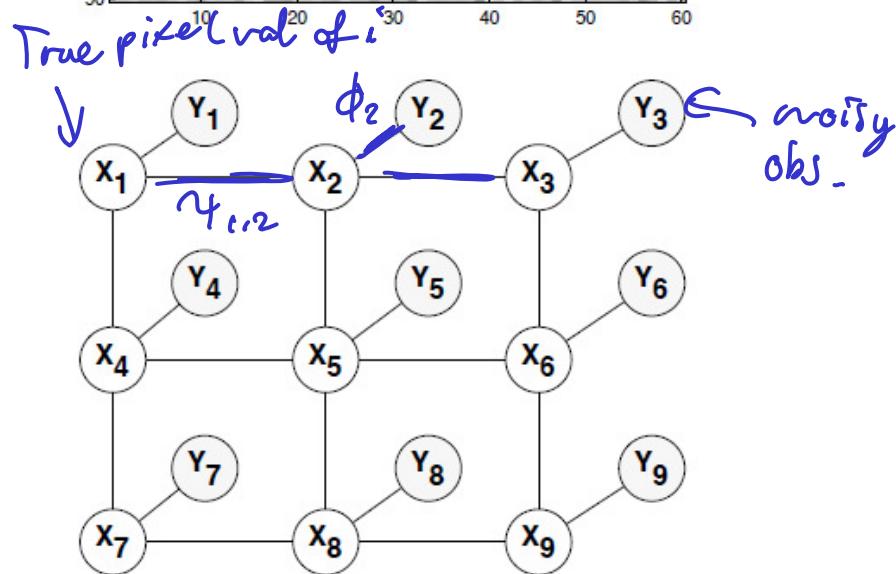
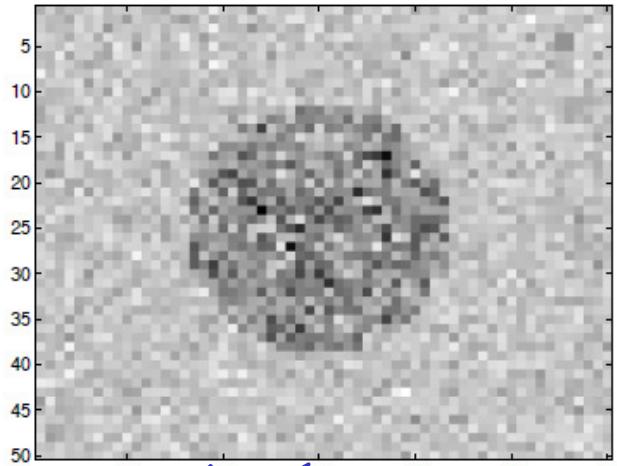
$$P(X_i \mid X_1 \dots X_{i-1}, X_{i+1} \dots X_m) = \frac{P(X_1 \dots X_m)}{P(X_1 \dots X_{i-1}, X_{i+1} \dots X_m)}$$

$$= \frac{\cancel{\prod_i Q(X_1 \dots X_m)}}{\sum_{X_i} \cancel{\prod_i Q(X_1 \dots X_m)}} = \frac{\prod_j \psi_j(c_j)}{\sum_{X_i} \prod_j \psi_j(c_j)}$$

$$= \frac{\prod_{j \in N(i)} \psi_j(s_j)}{\sum_{X_i} \prod_{j \in N(i)} \psi_j(c_j)}$$

↑
all factors that contain X_i

Example: (Simple) image segmentation



$$P(x) = \frac{1}{Z} \prod_i \Phi(x_i) \prod_{(j,k) \in E} \Psi(x_j, x_k)$$

$$\Phi(x_i) = \exp \left\{ -\frac{(y_i - \mu_{x_i})^2}{2\sigma_{x_i}^2} \right\}$$

$\mu_{x_i} > \text{mean}$
for true
pix. val x_i

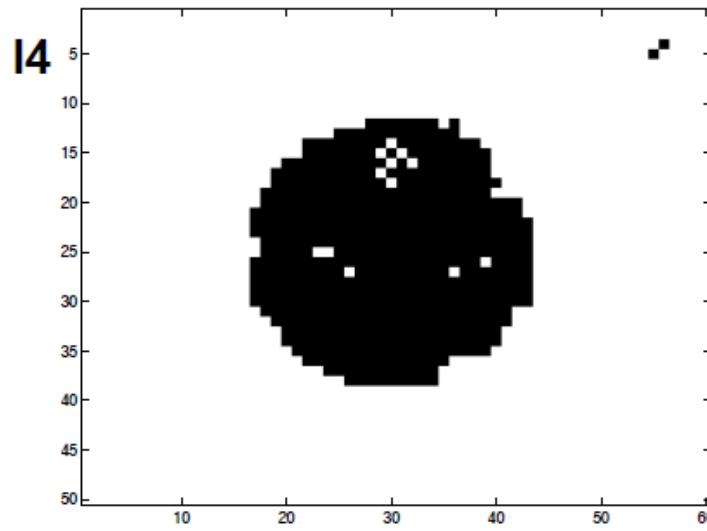
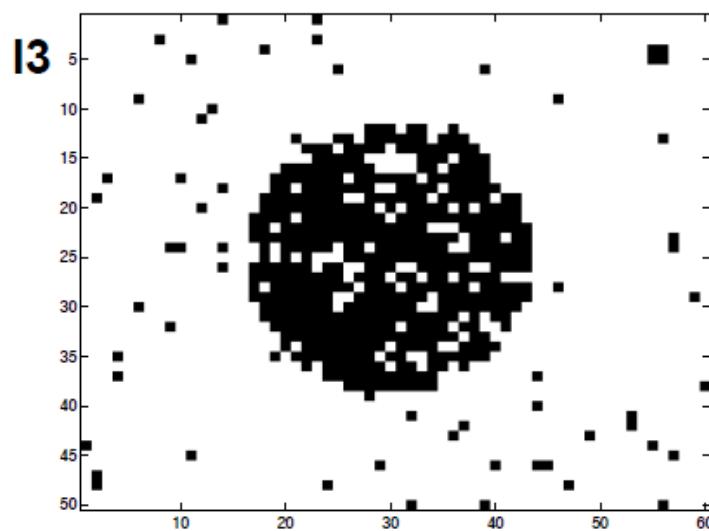
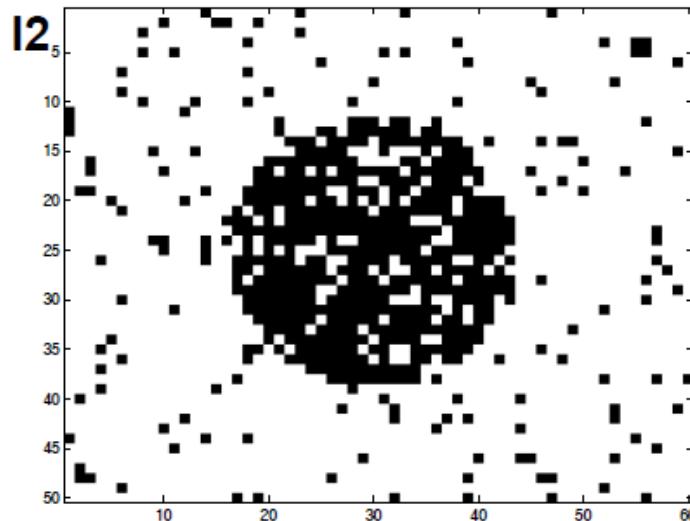
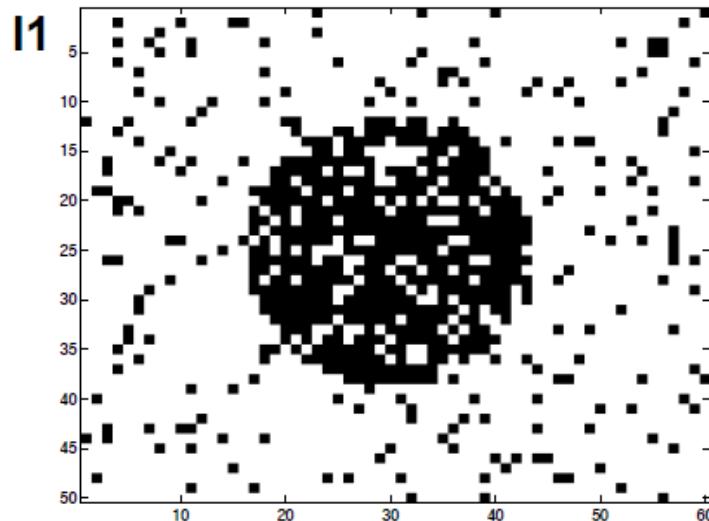
$$\Psi(x_i, x_j) = \exp \left\{ -\beta(x_i - x_j)^2 \right\}$$

Gibbs sampling :

$$P(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) =$$

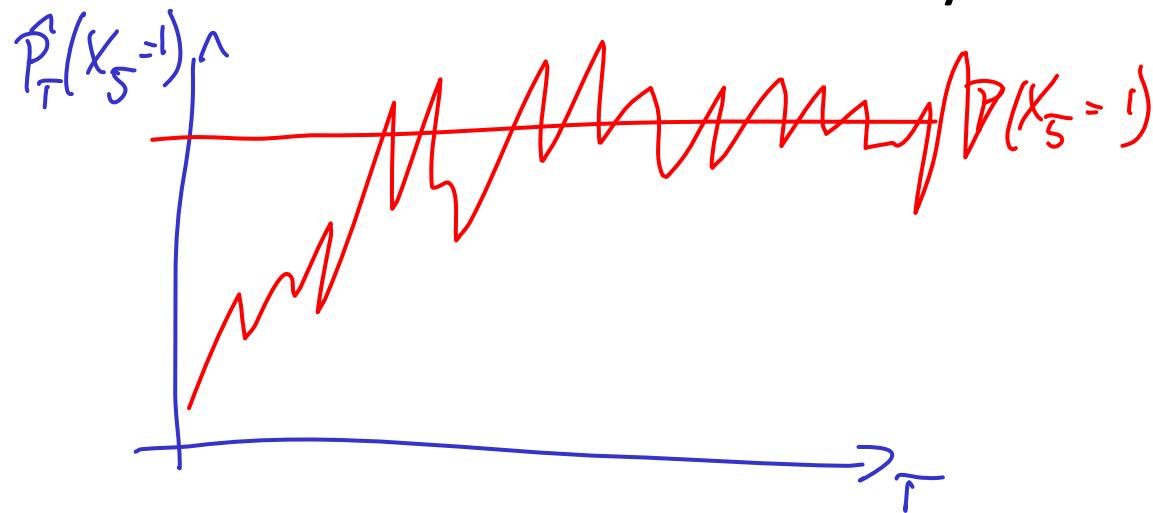
$$= \frac{\phi_i(x_i) \prod_{j \in N(i)} \Psi_{ij}(x_i, x_j)}{\sum_{k_i} \phi_i(k_i) \prod_{j \in N(i)} \Psi_{ij}(k_i, k_j)}$$

Gibbs Sampling iterations



Convergence of Gibbs Sampling

- When are we close to stationary distribution?



Summary of Sampling

- Randomized approximate inference for computing expectations, (conditional) probabilities, etc.
- Exact in the limit
 - But may need ridiculously many samples
- Can even directly sample from intractable distributions
 - Disguise distribution as stationary distribution of Markov Chain
 - Famous example: Gibbs sampling

Probabilistic Graphical Models

Lecture 17 – EM

CS/CNS/EE 155
Andreas Krause

Announcements

- Project poster session on **Thursday Dec 3, 4-6pm in Annenberg 2nd floor atrium!**
 - Easels, poster boards and cookies will be provided!
- Final writeup (8 pages NIPS format) due Dec 9

Approximate inference

- Three major classes of general-purpose approaches
- **Message passing**
 - E.g.: Loopy Belief Propagation (today!)
- **Inference as optimization**
 - Approximate posterior distribution by simple distribution
 - Mean field / structured mean field
 - Assumed density filtering / expectation propagation
- **Sampling based inference**
 - Importance sampling, particle filtering
 - Gibbs sampling, MCMC
- Many other alternatives (often for special cases)

Sample approximations of expectations

- $\underline{x_1, \dots, x_N}$ samples from RV X
- Law of large numbers:

$$\underline{\mathbb{E}_P[f(X)]} = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N f(x_i)$$

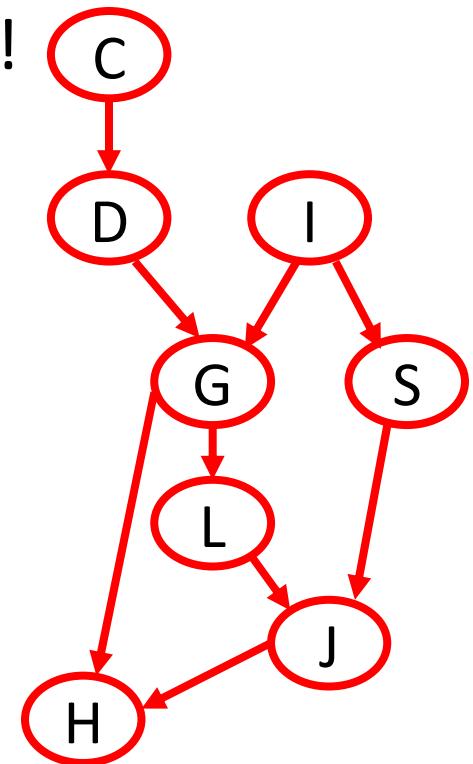
- Hereby, the convergence is with probability 1
(almost sure convergence)
- Finite samples:

$$\mathbb{E}_P[f(x)] \approx \frac{1}{N} \sum_{i=1}^N f(x_i)$$

Monte Carlo sampling from a BN

- Sort variables in topological ordering X_1, \dots, X_n
- For $i = 1$ to n do
 - Sample $x_i \sim P(X_i \mid X_1=x_1, \dots, X_{i-1}=x_{i-1})$ $= P(X_i \mid \text{Pa}_{X_i})$

- Works even with high-treewidth models!



Computing probabilities through sampling

- Want to estimate probabilities

- Draw N samples from BN

- Marginals

$$P(H=y) = \mathbb{E}_p [I_{H=y}] = \sum_x P(x) \cdot I_{H=y}(x)$$

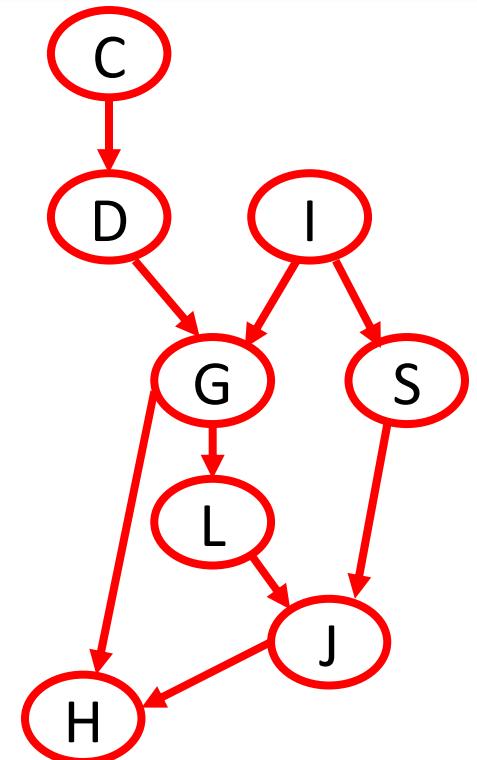
$\approx \frac{1}{N} \sum_{i=1}^N I_{H=y}(x^{(i)}) = \frac{\text{Count}(H=y)}{N}$

- Conditionals

$$P(D=h | H=m) = \frac{P(D=h, H=m)}{P(H=m)} = \frac{\text{Count}(D=h, H=m)}{\text{Count}(H=m)}$$

Rejection sampling

- Rejection sampling problematic for rare events



Sampling from intractable distributions

- Given unnormalized distribution

$$P(X) \propto Q(X) = P(X, X_{obs} = x_{obs})$$

$$P(X_A | X_B = x_B) \propto P(X_A, X_B = x_B)$$

- Q(X) efficient to evaluate, but normalizer intractable
- For example, $Q(X) = \prod_j \Psi(C_j)$
- Want to sample from $P(X) = \frac{1}{Z} Q(X)$
- Ingenious idea:**
Can create Markov chain that is efficient to simulate
and that has stationary distribution $P(X)$

Markov Chain Monte Carlo

- Given an unnormalized distribution $Q(x)$
- Want to design a Markov chain with stationary distribution

$$\pi(x) = 1/Z Q(x)$$

- Need to specify transition probabilities $P(x | x')$!

Designing Markov Chains

1) Proposal distribution $R(X' | X)$

- Given $X_t = x$, sample “proposal” $x' \sim R(X' | X=x)$
- Performance of algorithm will strongly depend on R

2) Acceptance distribution:

- Suppose $X_t = x$
- With probability $\alpha = \min \left\{ 1, \frac{Q(x')R(x | x')}{Q(x)R(x' | x)} \right\}$
set $X_{t+1} = x'$
- With probability $1-\alpha$, set $X_{t+1} = x$

Theorem [Metropolis, Hastings]: The stationary distribution is $Z^{-1} Q(x)$

- Proof: Markov chain satisfies detailed balance condition!

Gibbs sampling

- Start with initial assignment $\underline{x^{(0)}}$ to all variables
- For $t = 1$ to ∞ do
 - Set $x^{(t)} = \underline{x^{(t-1)}}$
 - For each variable X_i
 - Set v_i = values of all $x^{(t)}$ except x_i
 - Sample $x_i^{(t)}$ from $P(X_i | v_i)$
- Gibbs sampling satisfies detailed balance equation for P
- Can efficiently compute conditional distributions $P(X_i | v_i)$ for graphical models

Summary of Sampling

- Randomized approximate inference for computing expectations, (conditional) probabilities, etc.
- Exact in the limit
 - But may need ridiculously many samples
- Can even directly sample from intractable distributions
 - Disguise distribution as stationary distribution of Markov Chain
 - Famous example: Gibbs sampling

Summary of approximate inference

- Deterministic and randomized approaches
- Deterministic
 - Loopy BP
 - Mean field inference
 - Assumed density filtering
- Randomized
 - Forward sampling
 - Markov Chain Monte Carlo
 - Gibbs Sampling

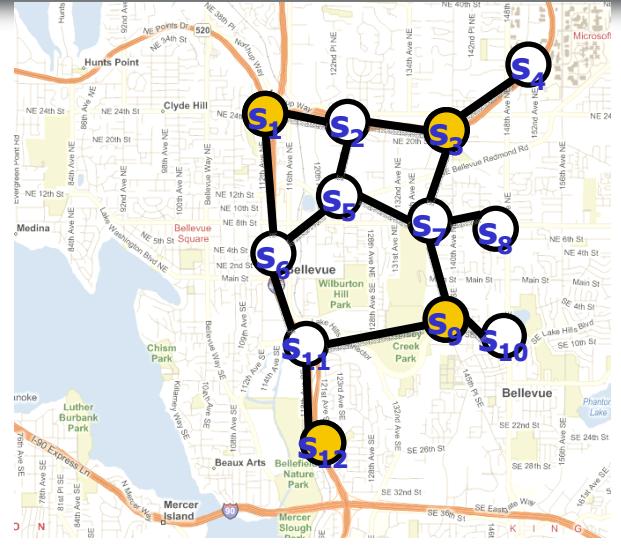
Recall: The “light” side

- Assumed
 - everything fully observable
 - low treewidth
 - no hidden variables
- Then everything is nice 😊
 - Efficient exact inference in large models
 - Optimal parameter estimation without local minima
 - Can even solve some structure learning tasks exactly

The “dark” side



represent



States of the world,
sensor measurements, ...

Graphical model

- In the real world, these assumptions are often violated..
- Still want to use graphical models to solve interesting problems..

Remaining Challenges

- Inference
 - **Approximate inference** for high-treewidth models
- Learning
 - Dealing with **missing data**
- Representation
 - Dealing with **hidden variables**

Learning general BNs

| | Known structure | Unknown structure |
|------------------|-----------------|-------------------|
| Fully observable | Easy! | Hard |
| Missing data | <i>Today</i> | |

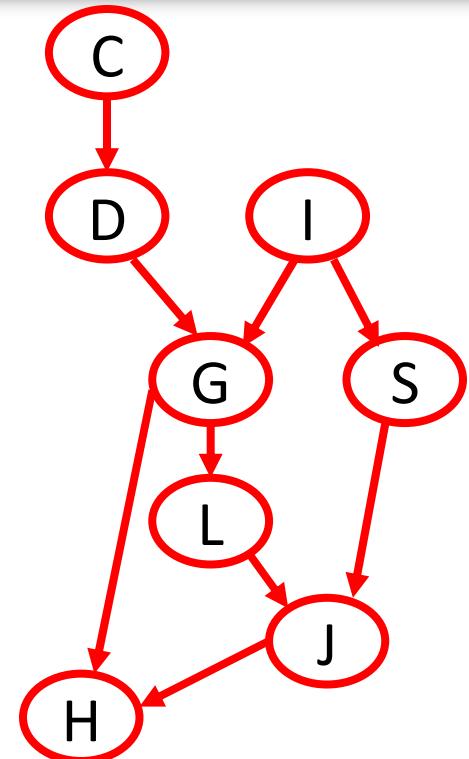
Dealing with missing data

- So far, have assumed all variables are observed in each training example

$$x^{(i)} = [G=g, D=d, G=h, l=l, s=s, \dots]$$
$$x^{(i+1)} = -$$

- In practice, often have missing data
 - Some variables may never be observed
 - Missing variables may be different for each example

$$x^{(i)} = [G=g, S=s, L=? , L=? , \dots]$$
$$x^{(i+1)} = -$$



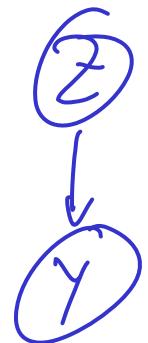
Gaussian Mixture Modeling

$$x = [y_1, z]$$

$$x^{(1)} = [0.1, 0.15, \text{blue}]$$

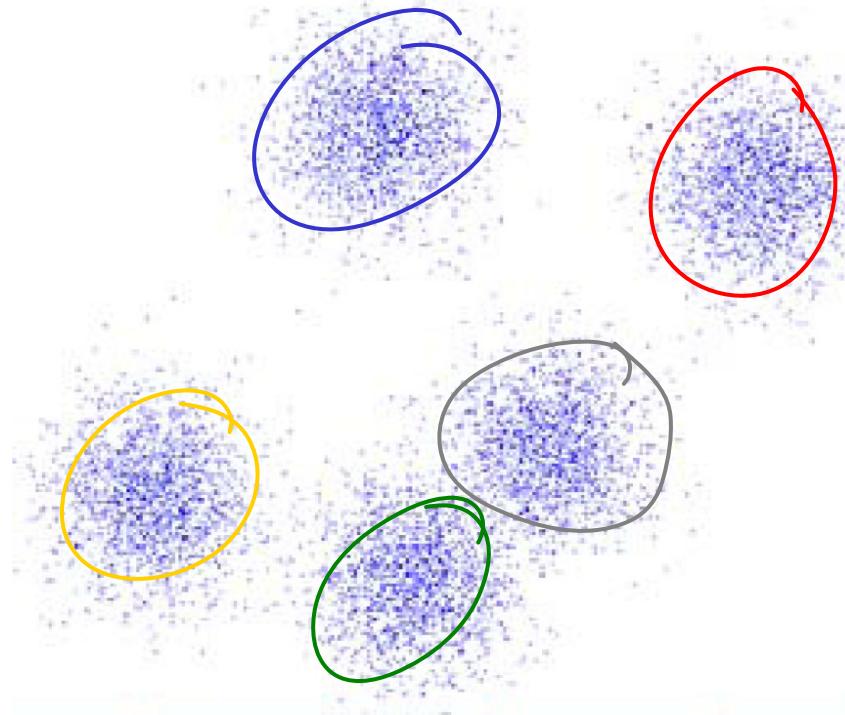
$$x^{(2)} = [-0.2, -0.2, \text{blue}]$$

$$x^{(3)} = [0.7, 0.7, \text{green}]$$



$$z \in \{1, \dots, k\}$$

$$P(y_i | z=j) = \mathcal{N}(y_i; \mu_j, \Sigma_j)$$



Learning with missing data

- Suppose \mathbf{X} is observed variables, \mathbf{Z} hidden variables
- Training data: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$
- Marginal likelihood:

$$\begin{aligned}\ell(D_x; \theta) &= \sum_{j=1}^m \log P(x^{(j)}; \theta) \\ &= \sum_{j=1}^m \log \sum_z P(x^{(j)}, z; \theta)\end{aligned}$$

$P = \prod \gamma$
 $\log P = \sum \log \gamma$

- Marginal likelihood doesn't decompose

Intuition: EM Algorithm

- Iterative algorithm for parameter learning in case of missing data
- EM Algorithm
 - Expectation Step: “Hallucinate” hidden values
 - Maximization Step: Train model as if data were fully observed
 - Repeat
- Will converge to local maximum

E-Step:

- x : observed data; z : hidden data
- “Hallucinate” missing values by computing distribution over hidden variables using current parameter estimate:
- For each example $x^{(j)}$, compute:

$$Q^{(t+1)}(z \mid x^{(j)}) = P(z \mid x^{(j)}, \underline{\theta^{(t)}})$$

Current parameter estimate

Towards M-step: Jensen inequality

- Marginal likelihood doesn't decompose

$$\ell(\mathbf{x}; \theta) = \sum_j \log \sum_{\mathbf{z}} P(\mathbf{x}^{(j)}, \mathbf{z}; \theta)$$


- **Theorem [Jensen's inequality]:**
For any distribution $P(\mathbf{z})$ and function $f(\mathbf{z})$,

$$\log \sum_{\mathbf{z}} P(\mathbf{z}) f(\mathbf{z}) \geq \sum_{\mathbf{z}} P(\mathbf{z}) \log f(\mathbf{z})$$

$$\log \left(\mathbb{E}_P[f(z)] \right) \geq \mathbb{E}_P[\log f(z)]$$

Lower-bounding marginal likelihood

- Jensen's inequality: $\log \sum_{\mathbf{z}} P(\mathbf{z}) f(\mathbf{z}) \geq \sum_{\mathbf{z}} P(\mathbf{z}) \log f(\mathbf{z})$
- From E-step: $Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) = P(\mathbf{z} \mid \mathbf{x}^{(j)}, \theta^{(t)})$

$$\begin{aligned}
 \ell(\mathbf{x}; \theta) &= \sum_j \log \sum_{\mathbf{z}} P(\mathbf{x}^{(j)}, \mathbf{z}; \theta) \\
 &= \sum_j \log \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \underbrace{\frac{P(\mathbf{x}^{(j)}, \mathbf{z}; \theta)}{Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}; \theta)}}_{f(\mathbf{z})} \\
 &\Rightarrow \sum_j \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \log \frac{P(\mathbf{x}^{(j)}, \mathbf{z}; \theta)}{Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}; \theta)} \\
 &= \sum_j \left\{ Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \log P(\mathbf{x}^{(j)}, \mathbf{z}; \theta) + H(Q^{(t+1)}) \cdot m \right\}
 \end{aligned}$$

Lower bound on marginal likelihood

- Bound of marginal likelihood with hidden variables

$$\ell(\mathbf{x}; \theta) \geq \sum_{j=1}^m \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \log P(\mathbf{z}, \mathbf{x}^{(j)} \mid \theta) + mH(Q^{(t+1)})$$

constant

- Recall: Likelihood in fully observable case:

$$\ell(\mathbf{x}; \theta) \geq \sum_{j=1}^m \log P(\mathbf{x}^{(j)} \mid \theta)$$

- Lower-bound interpreted as “weighted” data set

| x | z | $Q(z x)$ | fully obs: | | |
|------------|-----|----------|------------|--|--|
| [0..1..2] | 1 | .9 | 1 | | |
| [0..1..2] | 2 | .1 | 0 | | |
| [.5..1..4] | 1 | .3 | 0 | | |
| [.5..1..4] | 2 | .7 | 1 | | |
| | | | | | |

M-step: Maximize lower bound

- Lower bound:

$$\ell(\mathbf{x}; \theta) \geq \sum_{j=1}^m \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \log P(\mathbf{z}, \mathbf{x}^{(j)} \mid \theta) + mH(Q^{(t+1)})$$

- Choose $\theta^{(t+1)}$ to maximize lower bound

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \sum_{j=1}^m \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \log P(\mathbf{z}, \mathbf{x}^{(j)} \mid \theta)$$

- Use expected sufficient statistics (counts). Will see:
 - Whenever we used Count(x,z) in fully observable case, replace by $E_{Q^{t+1}}[\text{Count}(\mathbf{x}, \mathbf{z})]$

Coordinate Ascent Interpretation

- Define energy function

$$F[Q, \theta] = \sum_{j=1}^m \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}^{(j)}) \log P(\mathbf{z}, \mathbf{x}^{(j)} \mid \theta) + mH(Q)$$

- For any distribution Q and parameters θ :

$$\ell(\mathbf{x}; \theta) \geq F[Q, \theta]$$

- EM algorithm performs coordinate ascent on F :

$$Q^{(t+1)} = \operatorname{argmax}_Q F[Q, \theta^{(t)}]$$

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} F[Q^{(t+1)}, \theta]$$

- Monotonically converges to local maximum

EM for Gaussian Mixtures

$$\text{E-Step} \\ Q^{(t+1)}(z|x^{(t)})$$

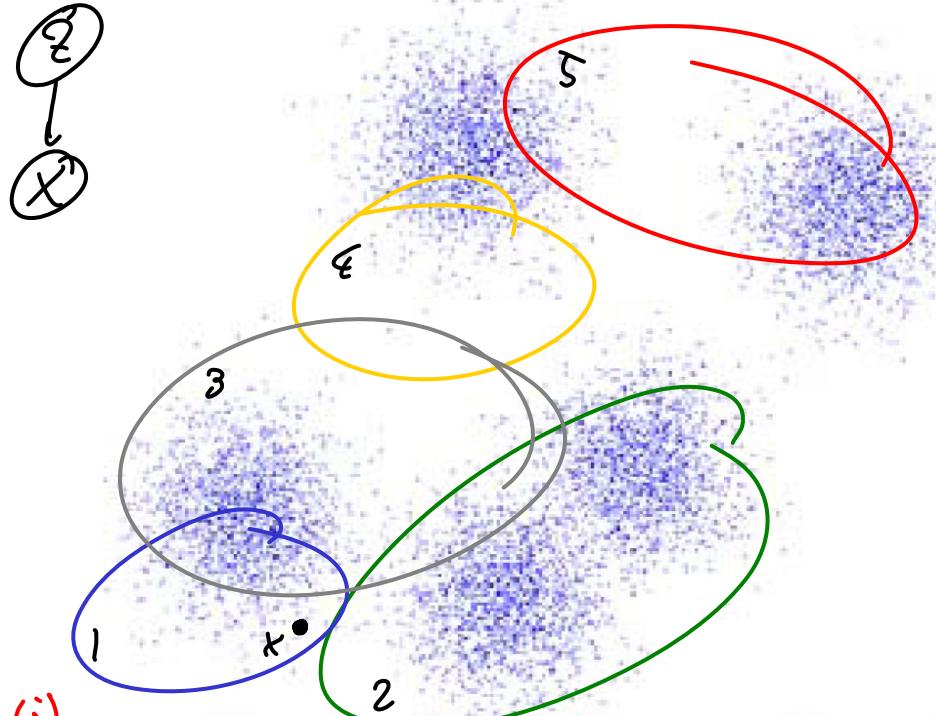
$$= P(z=z|x^{(t)}; \theta^{(t)})$$

$$Q^{(1)}(z=1 | x=[.1, .2]) = .4 \\ \begin{matrix} 2 \\ 1 \\ 3 \end{matrix} \quad \begin{matrix} .3 \\ .3 \end{matrix}$$

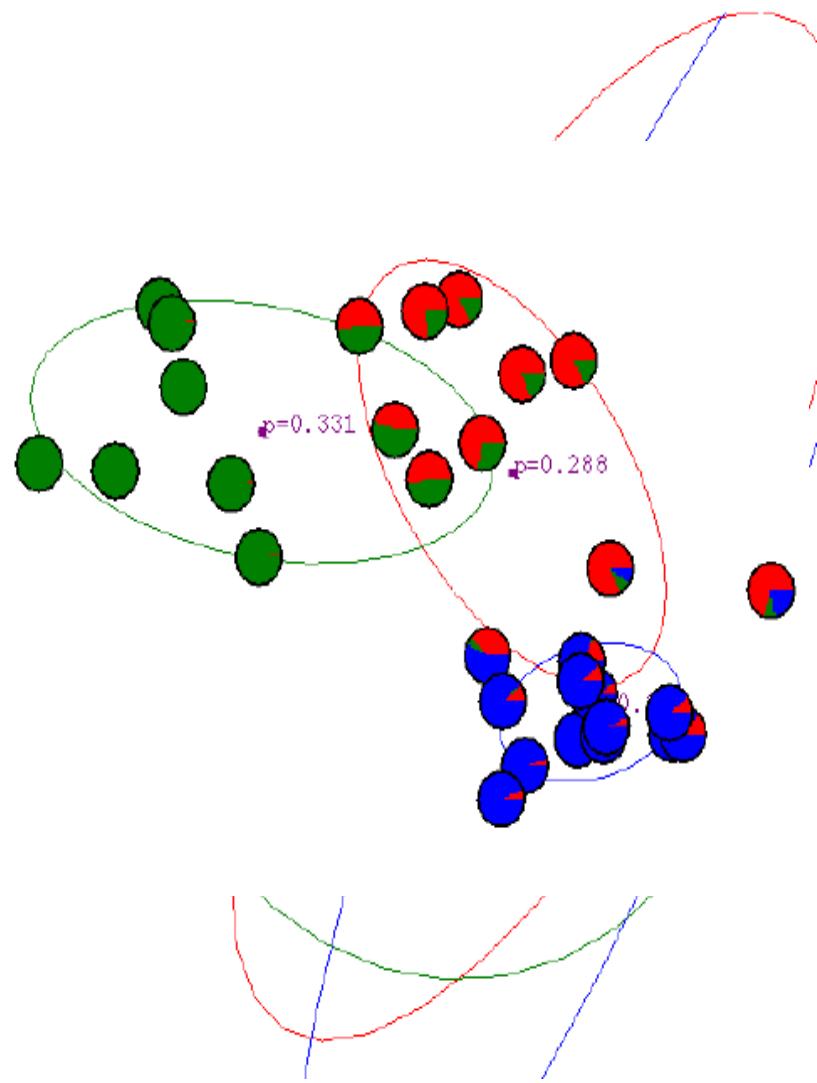
$$P(z|x) \propto P(x|z)P(z)$$

$$\text{M-Step: } \mu_i^{(t+1)} = \frac{\sum_{j=1}^m Q(z=i|x^{(t)}) \cdot x^{(t)}}{\sum_{j=1}^m Q(z=i|x^{(t)})}$$

$$\Sigma_i^{(t+1)} = \dots$$



EM Iterations [by Andrew Moore]



EM in Bayes Nets

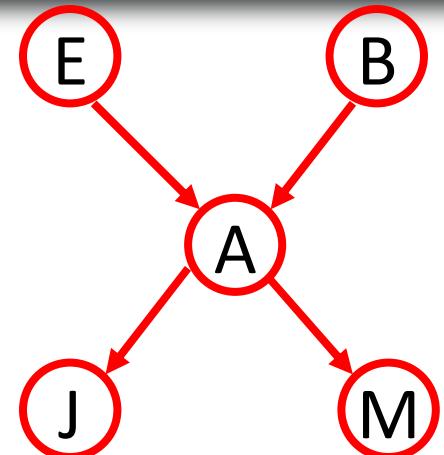
- Complete data likelihood

$$\ell(D; \theta) = \sum_j \log P(e^{(j)} | \theta) \cdot P(b^{(j)} | \theta) \cdot P(a^{(j)} | \theta), \dots$$

$$= \sum_j \log \prod_i P(X_i | Pa_i)$$

$$= \sum_0 \sum_i \log P(X_i | Pa_i)$$

Decomposes
Can optimize each CPT independently!

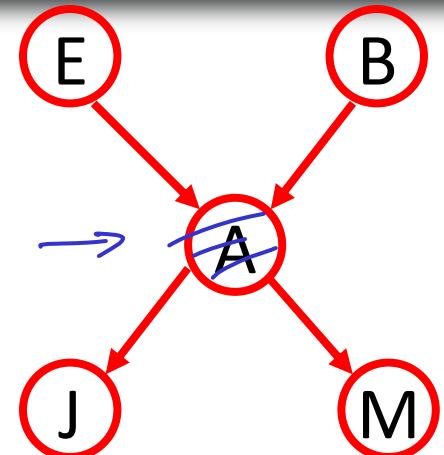


EM in Bayes Nets

- Incomplete data likelihood

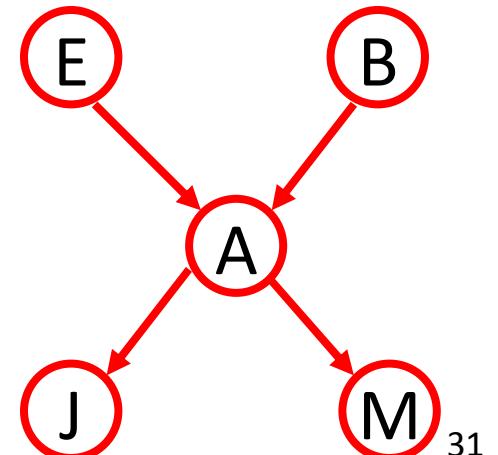
$$l(D; \theta) = \sum_j \log \sum_a P(e^{(j)} | \theta) P(b^{(j)} | \theta) P(a^{(j)} | e, b)$$

Does not decompose \vdash



E-Step for BNs

- Need to compute $Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) = P(\mathbf{z} \mid \mathbf{x}^{(j)}, \theta^{(t)})$
- For fixed \mathbf{z}, \mathbf{x} : Can compute using inference
- Naively specifying full distribution would be intractable
Have to use $Q^{(t+1)}$ "implicitly" (as needed)



M-step for BNs

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \sum_{j=1}^m \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}^{(j)}) \log P(\mathbf{z}, \mathbf{x}^{(j)} \mid \theta)$$

- Can optimize each CPT independently!

- MLE in fully observed case:

$$\hat{\theta}_{x|\mathbf{pa}_x} = \frac{\text{Count}(x, \mathbf{pa}_x)}{\text{Count}(\mathbf{pa}_x)}$$

- MLE with hidden data:

$$\hat{\theta}_{x|\mathbf{pa}_x}^{(t+1)} = \frac{\mathbb{E}_{Q^{(t+1)}}[\text{Count}(x, \mathbf{pa}_x)]}{\mathbb{E}_{Q^{(t+1)}}[\text{Count}(\mathbf{pa}_x)]}$$

Computing expected counts

$$\hat{\theta}_{x|\mathbf{pa}_x}^{(t+1)} = \frac{\mathbb{E}_{Q^{(t+1)}}[\text{Count}(x, \mathbf{pa}_x)]}{\mathbb{E}_{Q^{(t+1)}}[\text{Count}(\mathbf{pa}_x)]}$$

- Suppose we observe $O=o$
- Variables A hidden

Partition A into $A_o, A_h : A_o \cap O ; A_h \cap O = \emptyset$

$$\mathbb{E}_Q[\text{Count}(A_o = a_o^i, A_h = a_h^i)]$$

$$= \sum_j I[a_o^{(j)} = a_o^i] \cdot \underbrace{Q(a_h | O=o^{(j)})}_{\text{To evaluate need to perform inference}}$$

| Inference per data point

Learning general BNs

| | Known structure | Unknown structure |
|------------------|-----------------|-------------------|
| Fully observable | Easy! | Hard (2.) |
| Missing data | EM | Now |

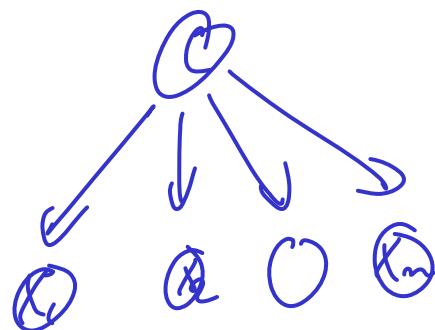
Structure learning with hidden data

- Fully observable case:
 - $\text{Score}(D;G) = \text{likelihood of data under most likely parameters}$
 - Decomposes over families
$$\text{Score}(D;G) = \sum_i \text{FamScore}_i(X_i \mid \text{Pa}_{X_i})$$
 - Can recompute score efficiently after adding/removing edges
- Incomplete data case:
 - $\text{Score}(D;G) = \text{lower bound from EM}$
 - Does not decompose over families
 - Search is very expensive
- Structure-EM: Iterate
 - Computing of expected counts
 - Multiple iterations of structure search for fixed counts
- Guaranteed to monotonically improve likelihood score

Hidden variable discovery

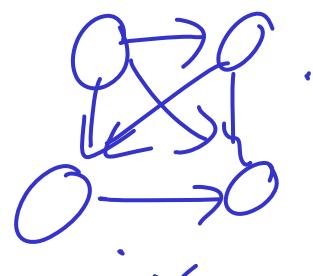
- Sometimes, “invention” of a hidden variable can drastically simplify model

“True” words

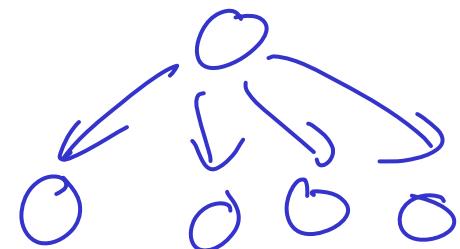


We only know about/
model
 x_1, \dots, x_m

Best fit to data:

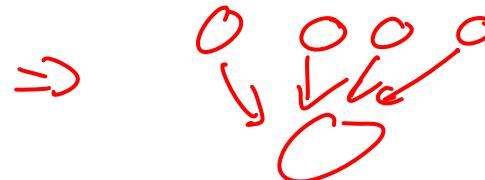


“Guess” existence of
hidden variable &
and run structure EM
⇒ (hopefully) “recover”



But: Can't identify common effects

$\text{O } \text{O } \text{O } \text{O}$
⇒ Strong limits to identifiability



Learning general BNs

| | Known structure | Unknown structure |
|------------------|-----------------|---------------------|
| Fully observable | Easy! | Hard (2.) |
| Missing data | EM | Structure-EM |

Probabilistic Graphical Models

Lecture 1 – Introduction

CS/CNS/EE 155
Andreas Krause

One of the **most exciting advances** in
machine learning (AI, signal processing,
coding, control, ...) in the last decades



How can we gain
global insight based on
local observations?

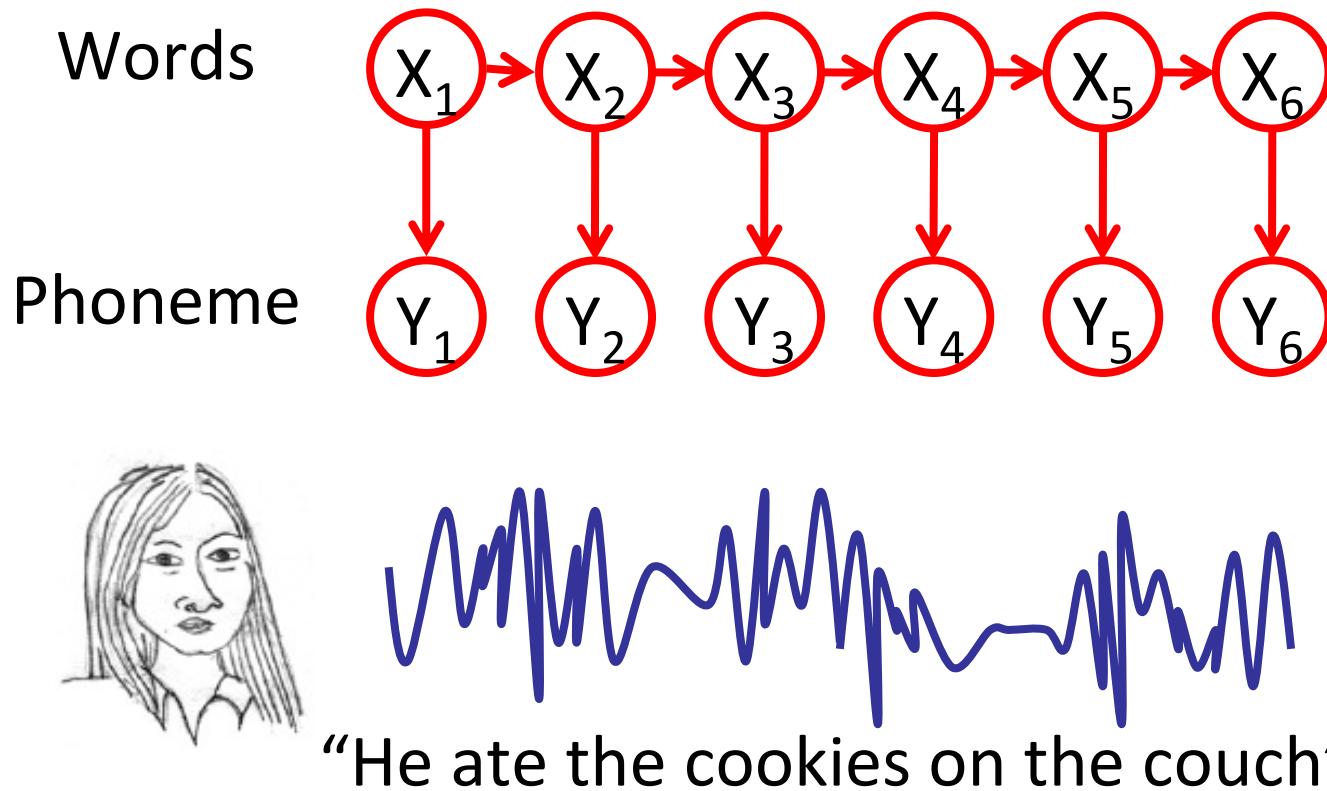
Key idea:

- **Represent** the world as a collection of random variables $X_1, \dots X_n$ with joint distribution $P(X_1, \dots, X_n)$
- **Learn** the distribution from data
- Perform “**inference**” (compute conditional distributions $P(X_i | X_1 = x_1, \dots, X_m = x_m)$)

Applications

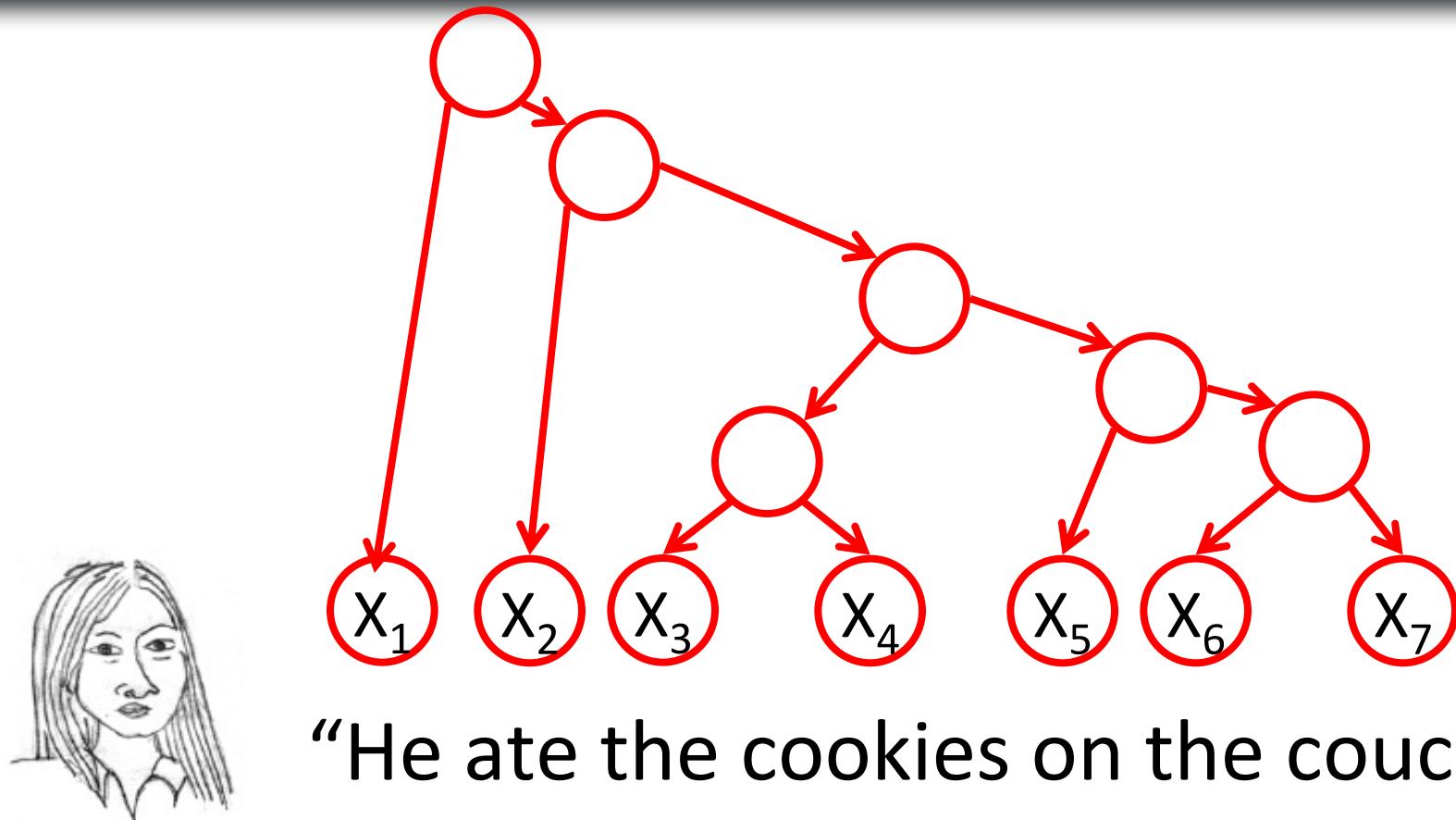
Natural Language Processing

Speech recognition

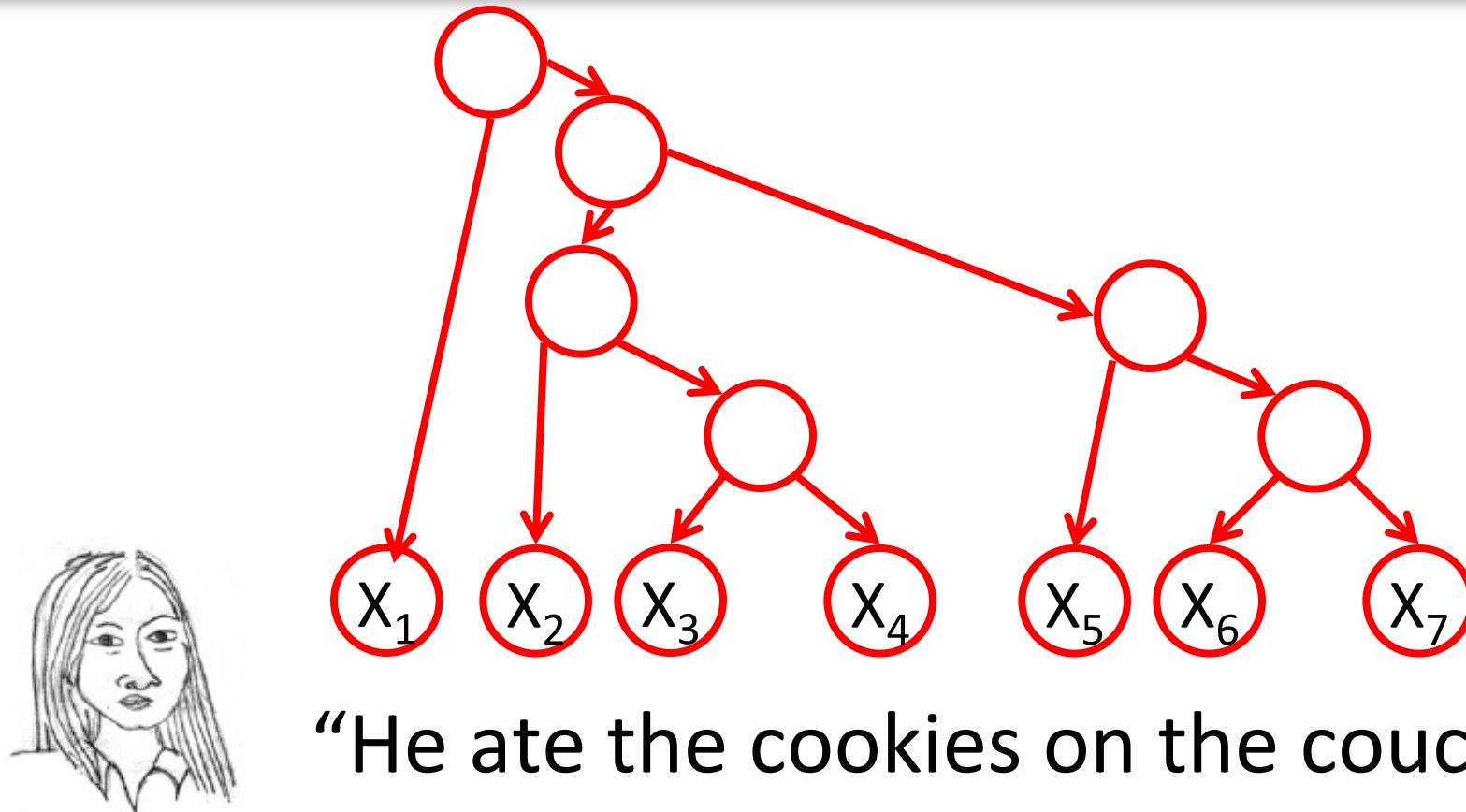


- Infer spoken words from audio signals
- “Hidden Markov Models”

Natural language processing



Natural language processing

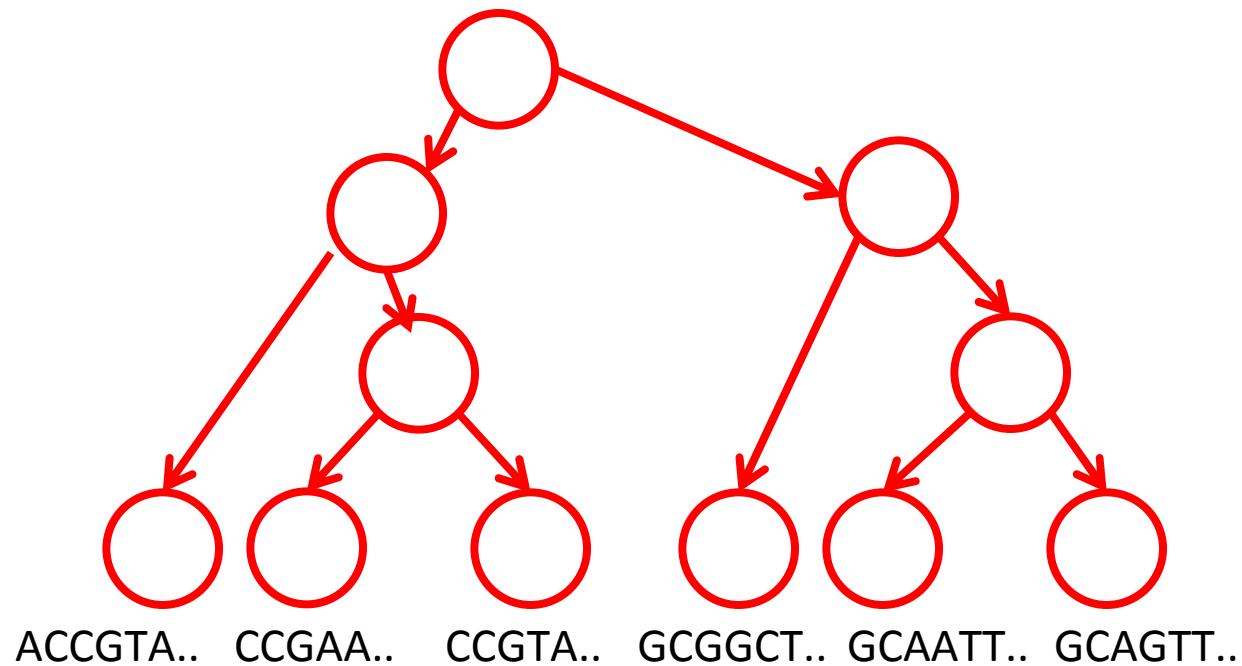


“He ate the cookies on the couch”

- Need to deal with ambiguity!
- Infer grammatical function from sentence structure
- “Probabilistic Grammars”

Evolutionary biology

[Friedman et al.]



- Reconstruct phylogenetic tree from current species (and their DNA samples)

Applications

Computer Vision

Image denoising

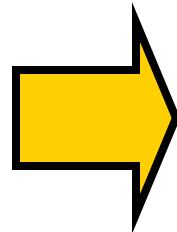
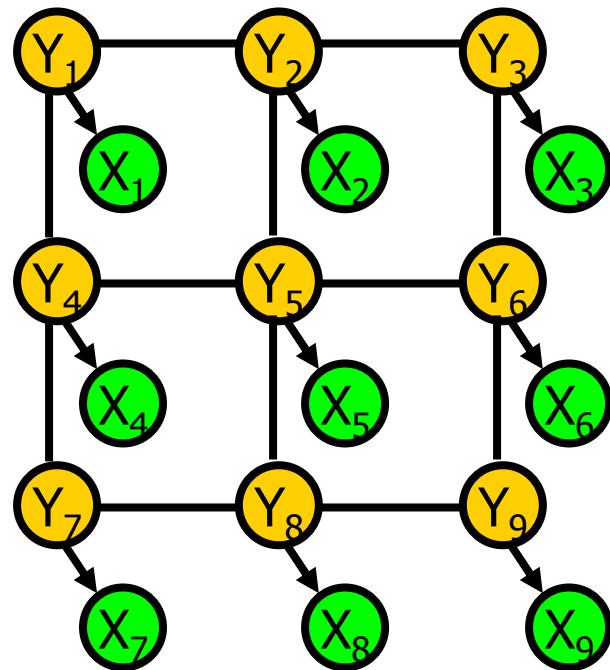


Image denoising

Markov Random Field



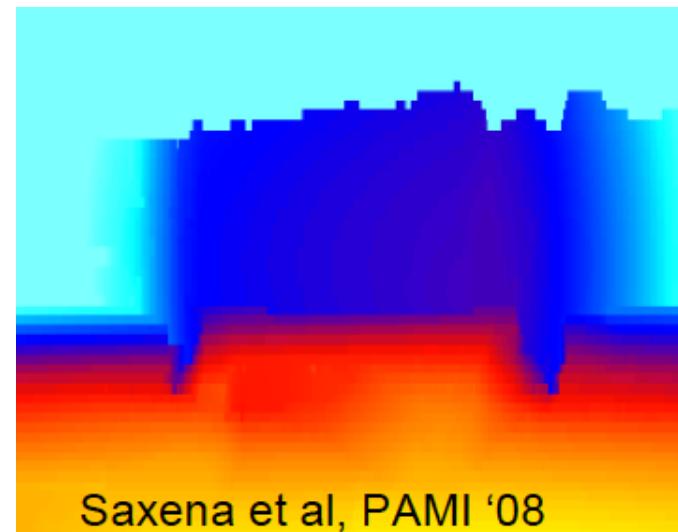
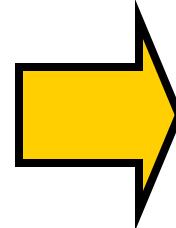
X_i : noisy pixels

Y_i : “true” pixels

12

12

Make3D

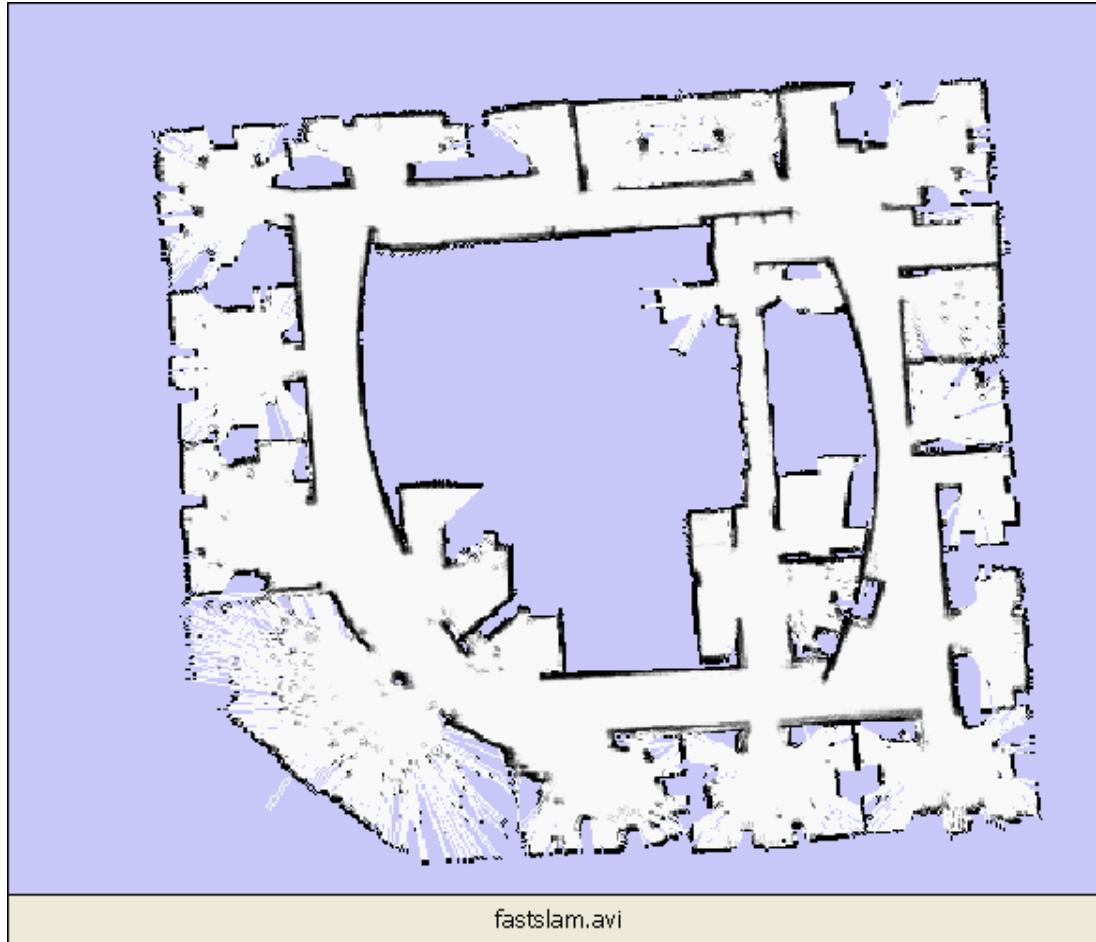


- Infer depth from 2D images
- “Conditional random fields”

Applications

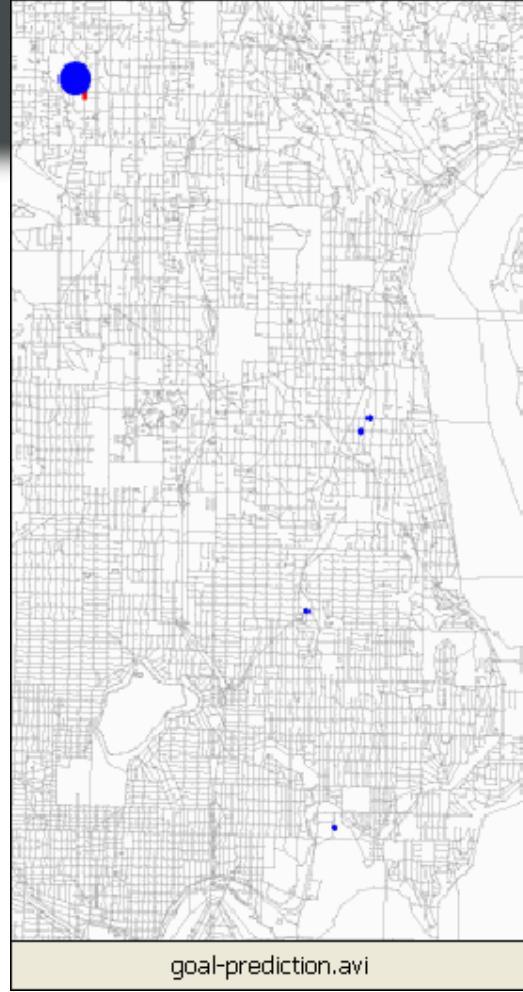
State estimation

Robot localization & mapping



D. Haehnel,
W. Burgard,
D. Fox, and
S. Thrun.
IROS-03.

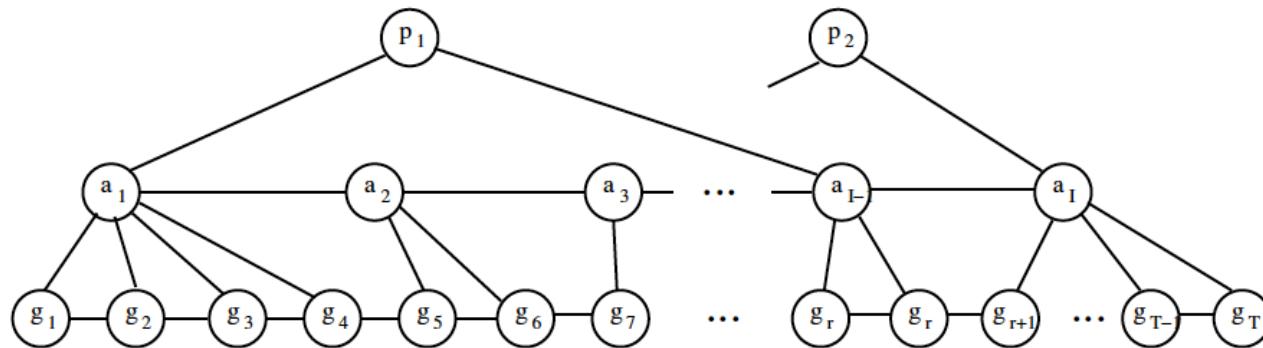
- Infer both location and map from noisy sensor data
- Particle filters



Activity recognition

L. Liao, D. Fox, and H. Kautz. AAAI-04

Predict “goals” from raw GPS data
“Hierarchical Dynamical
Bayesian networks”

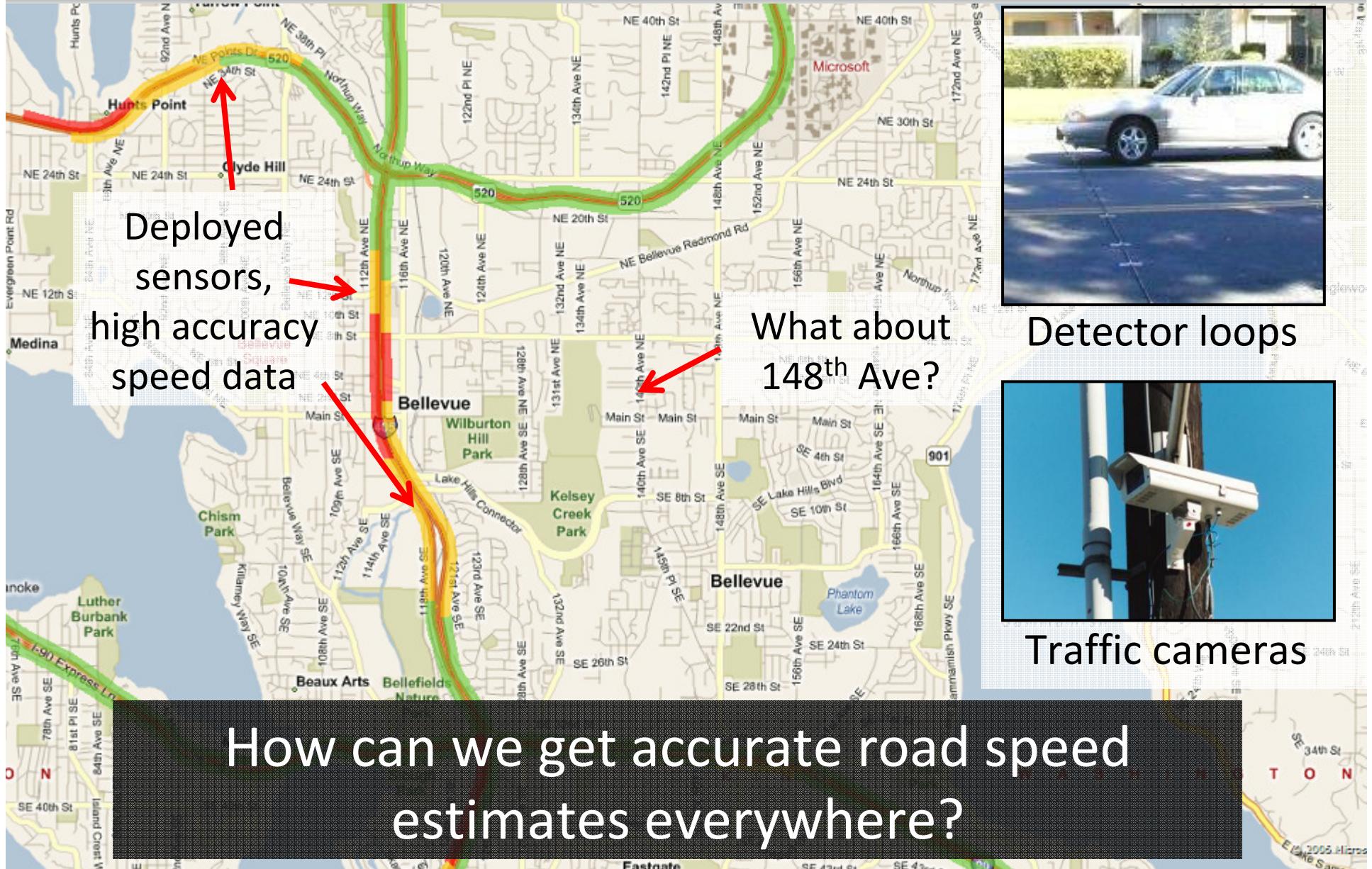


Significant places
home, work, bus stop, parking lot, friend

Activity sequence
walk, drive, visit, sleep, pickup, get on bus

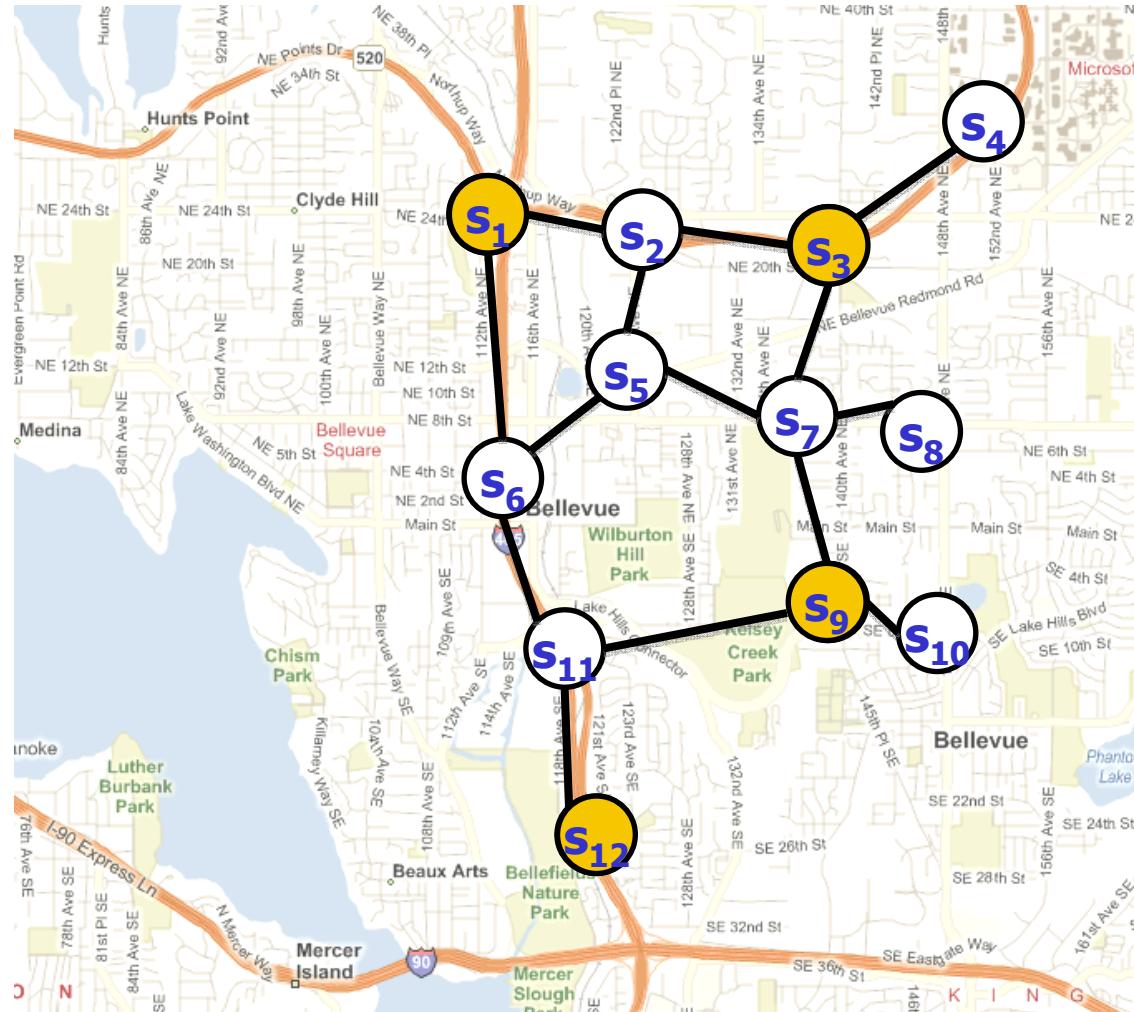
GPS trace
association to street map

Traffic monitoring



Cars as a sensor network

[Krause, Horvitz et al.]



- (Normalized) speeds as random variables
- Joint distribution allows modeling correlations
- Can predict unmonitored speeds from monitored speeds using $P(S_5 | S_1, S_9)$

Applications

Structure Prediction

Collaborative Filtering and Link Prediction

People you may know

L. Brouwer

T. Riley

invite | x

invite | x

See more »

NETFLIX

Browse DVDs Watch Instantly Your Queue Movies You'll ❤️

Suggestions (731) Rate Movies Taste Preferences

Suggestions in All Genres

Your Recent History [\(What's this?\)](#)

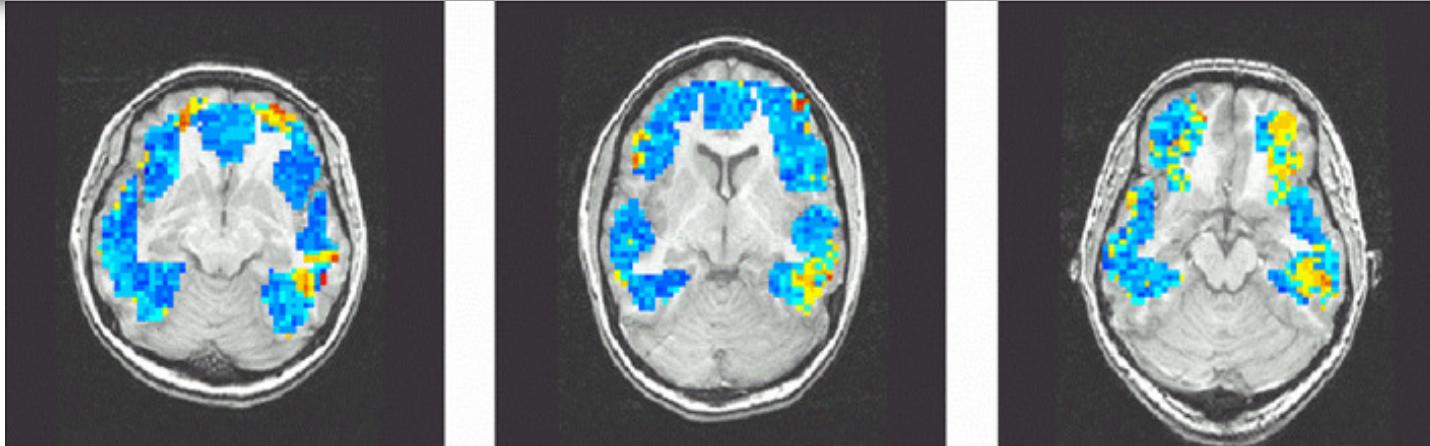
Recently Viewed Items

[Probabilistic Graphical Models: Principles and...](#)
by Daphne Koller

Continue shopping: Customers Who Bought Items In Your Recent History Also Bought

- Predict “missing links”, ratings...
- “Collective matrix factorization”, Relational models

Analyzing fMRI data



Mitchell et al.,
Science, 2008



- Predict activation patterns for nouns
- Predict connectivity (Pittsburgh Brain Competition)

Other applications

- Coding (LDPC codes, ...)
- Medical diagnosis
- Identifying gene regulatory networks
- Distributed control
- Computer music
- Probabilistic logic
- Graphical games
-

MANY MORE!!

Key challenges:

How do we

... **represent** such probabilistic models?

(distributions over vectors, maps, shapes,
trees, graphs, functions...)

... perform **inference** in such models?

... **learn** such models from data?

Syllabus overview

- We will study Representation, Inference & Learning
- First in the simplest case
 - Only discrete variables
 - Fully observed models
 - Exact inference & learning
- Then generalize
 - Continuous distributions
 - Partially observed models (hidden variables)
 - Approximate inference & learning
- Learn about algorithms, theory & applications

Overview

- Course webpage
 - <http://www.cs.caltech.edu/courses/cs155/>
- Teaching assistant: Pete Trautman
[\(trautman@cds.caltech.edu\)](mailto:trautman@cds.caltech.edu)
- Administrative assistant: Sheri Garcia
[\(sherি@cs.caltech.edu\)](mailto:sherি@cs.caltech.edu)

Background & Prerequisites

- Basic probability and statistics
- Algorithms
- CS 156a or permission by instructor
- Please fill out the questionnaire about background
(not graded ☺)
- Programming assignments in MATLAB.
- Do we need a MATLAB review recitation? *No.*

Coursework

- Grading based on
 - 4 homework assignments (one per topic) (40%)
 - Course project (40%)
 - Final take home exam (20%)
- 3 late days
- Discussing assignments allowed, but everybody must turn in their own solutions
- Start early! ☺ 

Course project

- “Get your hands dirty” with the course material
- Implement an algorithm from the course or a paper you read and apply it to some data set
- Ideas on the course website (soon)
- Application of techniques you learnt to your own research is encouraged
- Must be something new (e.g., not work done last term)

Project: Timeline and grading

- Small groups (2-3 students)
 - October 19: Project proposals due (1-2 pages); feedback by instructor and TA
 - November 9: Project milestone
 - December 4: Project report due; poster session
-
- Grading based on quality of poster (20%), milestone report (20%) and final report (60%)

Review: Probability

- This should be familiar to you...
- Probability Space (Ω, \mathcal{F}, P)
 - Ω : set of “atomic events”
 - $\mathcal{F} \subseteq 2^\Omega$: set of all (non-atomic) events
 - \mathcal{F} is a σ -Algebra
(closed under complements and countable unions)
 - $P: \mathcal{F} \rightarrow [0,1]$ probability measure
For $\omega \in \mathcal{F}$, $P(\omega)$ is the probability that event ω happens

$$\begin{aligned}\alpha &\in \mathcal{F} \\ \Omega \setminus \alpha &\in \mathcal{F}\end{aligned}$$

Interpretation of probabilities

- Philosophical debate..
- Frequentist interpretation
 - $P(\alpha)$ is relative frequency of α in repeated experiments
 - Often difficult to assess with limited data
- Bayesian interpretation
 - $P(\alpha)$ is “degree of belief” that α will occur
 - Where does this belief come from?
 - Many different flavors (subjective, pragmatic, ...)
- Most techniques in this class can be interpreted either way.

Independence of events

- Two events $\alpha, \beta \in F$ are independent if

$$P(\alpha \cap \beta) = P(\alpha) P(\beta)$$

- A collection S of events is independent, if for any subset $\alpha_1, \dots, \alpha_n \in S$ it holds that

$$P(\alpha_1, \dots, \alpha_m) = P(\alpha_1) P(\alpha_2) \cdot \dots \cdot P(\alpha_m)$$

~~Q: $\forall i, j : P(\alpha_i \cap \alpha_j) = P(\alpha_i) P(\alpha_j)$~~
 ~~$\Rightarrow \alpha_1, \dots, \alpha_m$ independent?~~ No.

Conditional probability

- Let α, β be events, $P(\beta) > 0$
- Then:

$$P(\alpha | \beta) = \frac{P(\alpha \cap \beta)}{P(\beta)}$$

Most important rule #1:

- Let $\alpha_1, \dots, \alpha_n$ be events, $P(\alpha_i) > 0$

- Then

$$P(\alpha_1 \cap \dots \cap \alpha_n) = P(\alpha_1) \cdot P(\alpha_2 | \alpha_1) \cdot \dots \cdot P(\alpha_n | \alpha_1 \dots \alpha_{n-1})$$

Chain rule

Most important rule #2:

- Let α, β be events with prob. $P(\alpha) > 0, P(\beta) > 0$
- Then

$$P(\alpha | \beta) = \frac{P(\alpha \cap \beta)}{P(\beta)} = \frac{P(\beta | \alpha) \cdot P(\alpha)}{P(\beta)}$$

$$P(\beta) = P(\beta | \alpha) \cdot P(\alpha) + P(\beta | \neg \alpha) \cdot P(\neg \alpha)$$

Bayes' rule



Random variables

- Events are cumbersome to work with.
- Let D be some set (e.g., the integers)
- A **random variable** X is a mapping $X: \Omega \rightarrow D$
- For some $x \in D$, we say
 $P(\underline{X = x}) = P(\{\omega \in \Omega: X(\omega) = x\})$
“probability that variable X assumes state x ”
- Notation: $\text{Val}(X) = \text{set } D \text{ of all values assumed by } X.$

Examples

- Bernoulli distribution: “(biased) coin flips”

$$D = \{H, T\}$$

Specify $P(X = H) = p$. Then $P(X = T) = 1-p$.

Write: $X \sim \text{Ber}(p)$;

- Multinomial distribution: “(biased) m-sided dice”

$$D = \{1, \dots, m\}$$

Specify $P(X = i) = p_i$, s.t. $\sum_i p_i = 1$

Write: $X \sim \text{Mult}(p_1, \dots, p_m)$

Multivariate distributions

- Instead of random variable, have random vector

$$\mathbf{X}(\omega) = [X_1(\omega), \dots, X_n(\omega)]$$

- Specify $P(X_1=x_1, \dots, X_n=x_n)$

- Suppose all X_i are Bernoulli variables.
- How many parameters do we need to specify?

$$\begin{array}{cccc|c} x_1 & x_2 & \dots & x_m \\ 0 & 0 & & 0 \\ 0 & & & ? \\ 0 & . & 0 & ? \\ 0 & & 1 & ? \\ 0 & & 0 & ? \end{array}$$

$2^m - 1$

Rules for random variables

- Chain rule

$$P(X_1 \dots X_n) = P(X_1) P(Y_2|X_1) \dots P(X_n|X_1 \dots X_{n-1})$$

- Bayes' rule

$$P(x|y) = \frac{P(y|x) P(x)}{P(y)}$$

How do we get $P(y)$?

Marginal distributions

- Suppose, X and Y are RVs with distribution $P(X,Y)$

X : Intelligence

Y : Grade

| X\Y | | VH | |
|-----|---|-----|------|
| | | H | L |
| A | V | 0.7 | 0.15 |
| | L | 0.1 | 0.05 |

$$P(\text{Grade} = A) = .85$$

Marginal distributions

- Suppose we have joint distribution $P(X_1, \dots, X_n)$
- Then

$$P(X_i = x_i) = \sum_{x_1} \sum_{x_2} \dots \sum_{x_{i-1}} \sum_{x_{i+1}} \dots \sum_{x_n} P(x_1, \dots, x_n)$$

- If all X_i binary: How many terms?

$$2^{n-1}$$

Independent RVs

- What if RVs are independent?

RVs X_1, \dots, X_n are independent, if for any assignment

$$P(X_1=x_1, \dots, X_n=x_n) = P(x_1) P(x_2) \dots P(x_n)$$

$$\Leftrightarrow \{w: X_i(w) = x_i\} \quad \forall i, \quad x_i \in \text{Val}(X_i) \quad \text{indep.}$$

- How many parameters are needed in this case? $\underline{\underline{n}} \quad n < 2^n$

$$X_i, X_j \text{ indep} \Rightarrow P(X_i | X_j) = P(X_i)$$

- Independence too strong assumption... Is there something weaker?

Key concept: Conditional independence

- Events α, β conditionally independent given γ if

$$P(\alpha \cap \beta | \gamma) = P(\alpha | \gamma) P(\beta | \gamma)$$

- Random variables X and Y cond. indep. given Z if
for all $x \in \text{Val}(X)$, $y \in \text{Val}(Y)$, $Z \in \text{Val}(Z)$

$$P(X = x, Y = y | Z = z) = P(X = x | Z = z) P(Y = y | Z = z)$$

- If $P(Y=y | Z=z) > 0$, that's equivalent to

$$P(X = x | Z = z, Y = y) = P(X = x | Z = z)$$

Similarly for sets of random variables X, Y, Z

We write: $P \models X \perp Y | Z$

Why is conditional independence useful?

- $P(X_1, \dots, X_n) = P(X_1) P(X_2 | X_1) \dots P(\underbrace{X_n | X_1, \dots, X_{n-1}}_{n-1})$

How many parameters?

$$2^0 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1$$

- Now suppose $X_1 \dots X_{i-1} \perp X_{i+1} \dots X_n | X_i$ for all i

Then

$$P(X_1, \dots, X_n) = \underbrace{P(X_1)}_1 \cdot \underbrace{P(X_2 | X_1)}_2 \cdot \underbrace{P(X_3 | X_2)}_2 \cdot \dots \cdot \underbrace{P(X_n | X_{n-1})}_2$$

$$2^{n-1} < 2^n$$

How many parameters?

Exponential reduction in # params

- Can we compute $P(X_n)$ more efficiently?

Yes (often)

Properties of Conditional Independence

- **Symmetry**

- $X \perp Y | Z \Rightarrow Y \perp X | Z$

- **Decomposition**

- $X \perp Y, W | Z \Rightarrow X \perp Y | Z$

- **Contraction**

"Converse Decomposition"

- $(X \perp Y | Z) \wedge (X \perp W | Y, Z) \Rightarrow X \perp Y, W | Z$

- **Weak union**

- $X \perp Y, W | Z \Rightarrow X \perp Y | Z, W$

- **Intersection**

- $(X \perp Y | Z, W) \wedge (X \perp W | Y, Z) \Rightarrow X \perp Y, W | Z$

- Holds only if distribution is positive, i.e., $P > 0$

Key questions

- How do we specify distributions that satisfy particular independence properties?
 → **Representation**
- How can we exploit independence properties for efficient computation?
 → Inference
- How can we identify independence properties present in data?
 → Learning

Will now see examples: Bayesian Networks

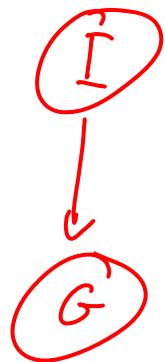
Bayesian networks

- A powerful class of probabilistic graphical models
- Compact parametrization of high-dimensional distributions
- In many cases, efficient exact inference possible
- Many applications
 - Natural language processing
 - State estimation
 - Link prediction
 - ...
- Demo..

Key idea

- Conditional parametrization
(instead of joint parametrization)
- For each RV, specify $P(X_i | X_A)$ for set X_A of RVs
- Then use chain rule to get joint parametrization
- Have to be careful to guarantee legal distribution...

Example: 2 variables

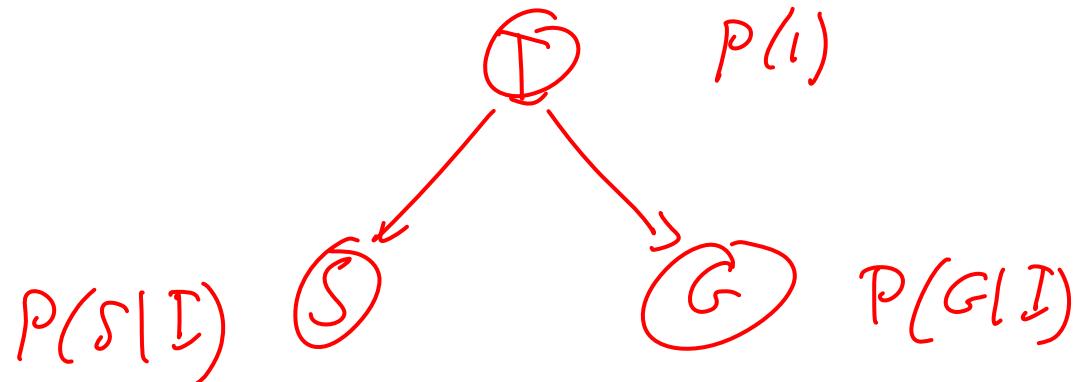


$$P(I = VH) = 0.8$$

$$P(G|I)$$

| | | |
|----|------------|------------|
| | A | B |
| VH | 0.8 | 0.2 |
| H | 0.6 | 0.4 |
| | $\sum_i=1$ | $\sum_j=1$ |

Example: 3 variables



$$P(I, S, G) = P(I) P(G|I) P(S|I)$$

Example: Naïve Bayes models

- Class variable Y
- Evidence variables X_1, \dots, X_n
- Assume that $X_A \perp X_B \mid Y$
for all subsets X_A, X_B of $\{X_1, \dots, X_n\}$
- Conditional parametrization:
 - Specify $P(Y)$
 - Specify $P(X_i \mid Y)$
- Joint distribution

$$P(X_1, \dots, X_n, Y) = P(Y) \prod_i P(X_i \mid Y)$$

What you need to know

- Basic probability
- Independence and conditional independence
- Chain rule & Bayes' rule
- Naïve Bayes models

Tasks

- By tomorrow (October 1, 4pm): hand in questionnaire about background to Sheri Garcia
- Read Chapter 2 in Koller & Friedman
- Start thinking about project teams and ideas (proposals due October 19)