Graphical models

Review

$\mathbb{P}[(x \vee y \vee \bar{z}) \wedge (\bar{y} \vee \bar{u}) \wedge (z \vee w) \wedge (z \vee u \vee v)]$

- Dynamic programming on graphs
  - variable elimination example
- Graphical model = graph + model
  - e.g., Bayes net: DAG + CPTs
  - e.g., rusty robot
- Benefits:
  - fewer parameters, faster inference
  - some properties (e.g., some conditional independences) depend only on graph

Review

- Blocking

- Explaining away

Rains --> Wet --> Rusty
vs Rains --> Wet (shaded) --> Rusty

Rains --> Wet <-- Outside
vs Rains --> Wet (shaded) <-- Outside

d-separation

- General graphical test: "d-separation"
  - d = dependence
- $X \perp Y \mid Z$ when there are no active paths between X and Y given Z
  - activity of path depends on conditioning variable/set Z
- Active paths of length 3 ($W \notin$ conditioning set):

active paths

X --> W --> Y
X <-- W <-- Y
X <-- W --> Y
X --> Z <-- Y
X --> W <-- Y *if* W --> ... --> Z

Longer paths

- Node X is active (wrt path P) if:

  and inactive o/w

- (Undirected) path is active if ***all*** intermediate nodes are active

active if
unshaded and path arrows are >>, <<, or <>
shaded (or descendant shaded) and arrows >< (collider)

longer paths:
active when *all* intermediate nodes are active

example: shade Rusty; are M and O indep?
no: active path thru Ru and W

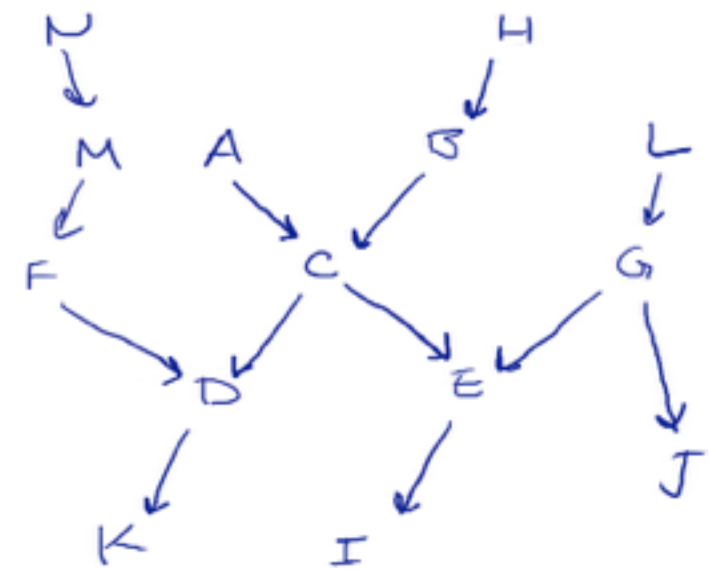
Algorithm: $X \perp Y \mid \{Z_1, Z_2, \ldots\}$?

- For each Z_i:
  - mark self and ancestors by traversing parent links
- Breadth-first search starting from X
  - traverse edges only if they can be part of an active path
    - use "ancestor of shaded" marks to test activity
  - prune when we visit a node for the second time from the same direction (from children or from parents)
- If we reach Y, then X and Y are dependent given $\{Z_1, Z_2, \ldots\}$ — else, conditionally independent

test activity:
  e.g., coming in from child; if node is marked, can't leave by parents
  e.g., coming in from parent; if node is unmarked, can't leave by parent

Markov blanket

- Markov blanket of C = minimal set of obs'ns to make C independent of rest of graph



MB(C) = A..G
 = parents, children, co-parents
 = enough to ensure no active paths to C
AB block from above; DE block to below; conditioning on DE makes C depend on FG, so need them too

Learning fully-observed Bayes nets

P(M) =

P(Ra) =

P(O) =

P(W | Ra, O) =

P(Ru | M, W) =

M	Ra	O	W	Ru
T	F	T	T	F
T	T	T	T	T
F	T	T	F	F
T	F	F	F	T
F	F	T	F	T

P(M) = 3/5
P(Ra) = 2/5
P(O) = 4/5
P(W|Ra, O):
TT: 1/2 TF: 0/0 !
FT: 1/2 FF: 1/1 ?
P(Ru|M, W):
TT: 1/2 TF: 1/1 ?
FT: 0/0 ! FF: 1/2

note division by zero, extreme probabilities --> Laplace smoothing

Limitations of counting

- Works ***only*** when all variables are observed in all examples
- If there are ***hidden*** or ***latent*** variables, more complicated algorithm (expectation-maximization or spectral)
  - or use a toolbox!

EM: alternately infer distribution for latent nodes, maximize likelihood given that distribution

we'll discuss later in course

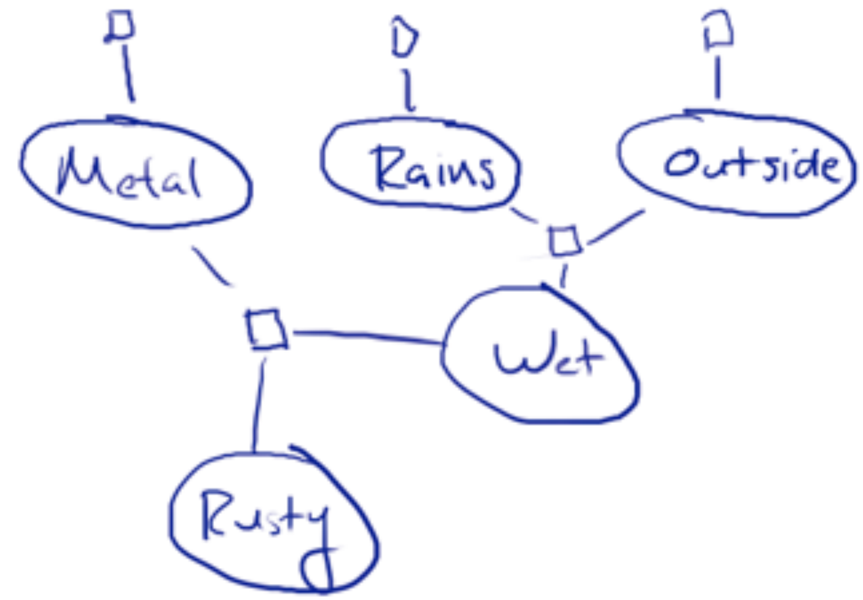
Factor graphs

- Another common type of graphical model
- *Undirected, bipartite* graph instead of DAG
- Like Bayes net:
  - can represent any distribution
  - can infer conditional independences from graph structure
  - but some distributions have more faithful representations in one formalism or the other

more faithful: more of the conditional independences follow from graph structure

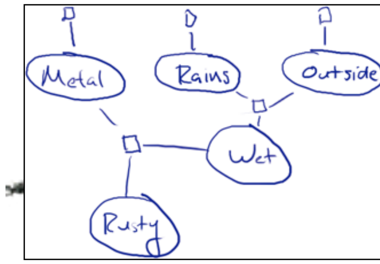
more faithful as Bayes net: e.g., rusty robot
more faithful as factor graph:
 e.g., node with a lot of neighbors, but simple (factored) structure of joint potential
 e.g., any graph with only pairwise potentials but bigger cliques
 e.g., cycles (ring in factor graph -> chorded ring -> chain junction tree of treewidth 2)

Rusty robot: factor graph

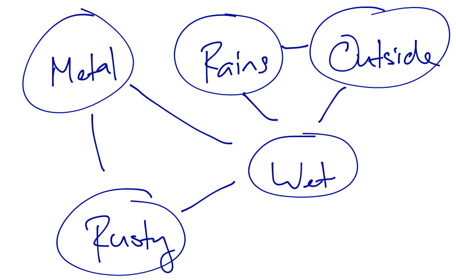


P(M) P(Ra) P(O) P(W|Ra,O) P(Ru|M,W)

node = RV
draw squares for factors (also “potentials”)
    phi_{M, Ra, O, W, Ru}
draw arcs: factor mentions variable
defn: neighbor set
    nbr(phi_W) = {W, Ra, O}
    nbr(W) = {phi_W, phi_Ru}



Conventions



Markov random field

- Don't need to show unary factors—why?
  - can usually be collapsed into other factors
  - don't affect structure of dynamic programming
- Show factors as cliques

another convention: instead of a factor, draw a clique
    e.g.: binary factors are just edges (no little square)

MRFs: lose some information relative to factor graphs
    e.g., distinction between A — B — C — A and a factor on ABC

Non-CPT factors

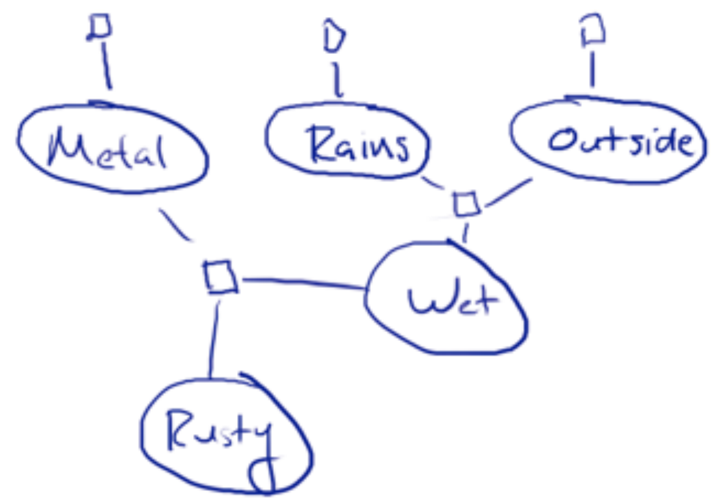
- Just saw: easy to convert Bayes net → factor graph
- In general, factors need not be CPTs: any nonnegative #s allowed
  - higher # → this combination more likely
- In general, P(A, B, …) =
- Z =

normalizing constant: to compensate for sum of P-tilde not being 1
P = (1/Z) P-tilde(A, B, ...)
P-tilde = prod_{i in factor nodes} phi_i(nbr(i))
Z = sum_A sum_B ... P-tilde(A, B, ...)

Independence

- Just like Bayes nets, there are graphical tests for independence and conditional independence
- Simpler, though:
  - Cover up all observed nodes
  - Look for a path

Independence example



Are M and O dependent? (y)
given Ru? (y)
given W? (n)

Note: some answers different than we got from Bayes net representation! Fewer conditional independences: e.g., in Bayes net, M ⊥ O

What gives?

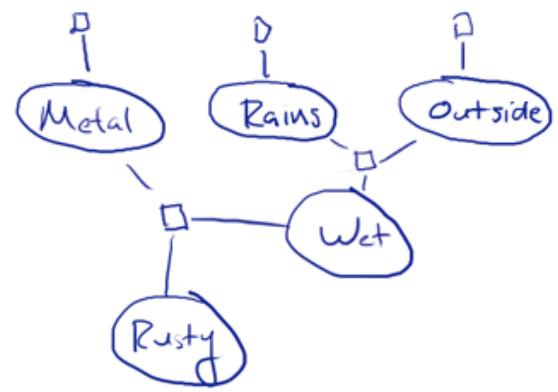
- Take a Bayes net, list (conditional) independences
- Convert to a factor graph, list (conditional) independences
- Are they the same list?
- What happened?

same list? No! Fewer CIs in factor graph
e.g., M&O dep in factor graph, but not in BNet

went away? No, since it's the same distribution
instead, turned into "accidental" CIs
factor graph doesn't force factors to be CPTs

Inference: same kind of DP as before

$$P(M, Ra, O, W, Ru) = \phi_1(M)\,\phi_2(Ra)\,\phi_3(O)\,\phi_4(Ra, O, W)\,\phi_5(M, W, Ru)\,/\,Z$$



$\phi_1(M)$ = T 0.9, F 0.1

$\phi_2(Ra)$ = T 0.7, F 0.3

$\phi_3(O)$ = T 0.2, F 0.8

$\phi_4(Ra, O, W)$ =

TTT	0.9
TTF	0.1
TFT	0.1
TFF	0.9
FTT	0.1
FTF	0.9
FFT	0.1
FFF	0.9

$\phi_5(M, W, Ru)$ =

TTT	0.8
TTF	0.2
TFT	0.1
TFF	0.9
FTT	0
FTF	1
FFT	0
FFF	1

- Typical Q: given Ra=F, Ru=T, what is P(W)?

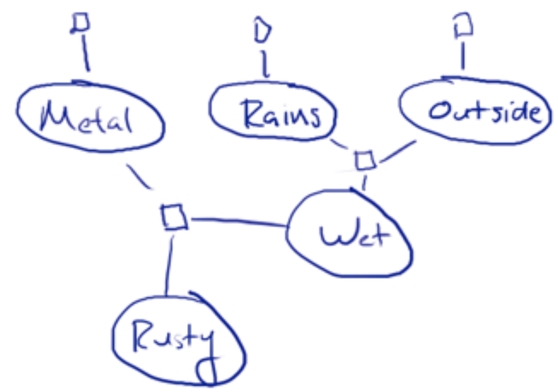
label: evidence, query (evidence is shaded)
everything else: nuisance

we will go through these steps: instantiate evidence, eliminate nuisance nodes, normalize, answer query

P(Metal) = 0.9
P(Rains) = 0.7
P(Outside) = 0.2
P(Wet | Rains, Outside)
TT: 0.9 TF: 0.1
FT: 0.1 FF: 0.1
P(Rusty | Metal, Wet) =
TT: 0.8 TF: 0.1
FT: 0 FF: 0

Incorporate evidence

$$P(M, Ra, O, W, Ru) = \phi_1(M)\,\phi_2(Ra)\,\phi_3(O)\,\phi_4(Ra, O, W)\,\phi_5(M, W, Ru) / Z$$



$\phi_1(M) =$

T	0.9
F	0.1

$\phi_2(Ra) =$

T	0.7
F	0.3

$\phi_3(O) =$

T	0.2
F	0.8

$\phi_4(Ra, O, W) =$

TTT	0.9
TTF	0.1
TFT	0.1
TFF	0.9
FTT	0.1
FTF	0.9
FFT	0.1
FFF	0.9

$\phi_5(M, W, Ru) =$

TTT	0.8
TTF	0.2
TFT	0.1
TFF	0.9
FTT	0
FTF	1
FFT	0
FFF	1

Condition on Ra=F, Ru=T

note: Z = 1 before evidence (since we converted from Bayes net) but evidence will change Z
can't answer *any* questions w/o Z
new Z will be a result of inference
(goal: get it w/ less than exponential work)

change LHS to P(M, O, W | Ra=T, Ru=F)
cross out Ra = T in Phi2, Phi4
cross out Ru = F in Phi5
cross out Ra as arg in Phi2, Phi4
cross out Ru as arg in Phi5
note: changed 3-arg to 2-arg potentials
cross out Phi2 (incorporate into Z)

Eliminate nuisance nodes

$P(M, O, W \mid Ra = T, Ru = F) = \phi_1(M)\, \phi_2(Ra)\, \phi_3(O)\, \phi_4(Ra, O, W)\, \phi_5(M, W, Ru) / Z$

- Remaining nodes: M, O, W
- Query: P(W)
- So, O&M are nuisance—marginalize away
- Marginal =

marginal = sum_M sum_O P(M, O, W | Ra=T, Ru=F)
 = sum_M sum_O phi1(M) phi3(O) phi4(O,W) phi5(M,W) / Z

Elimination order

$$\sum_M \sum_O \phi_1(M)\phi_3(O)\phi_4(O,W)\phi_5(M,W)/Z$$

- Sum out nuisance variables in turn
- Can do it in any order, but some orders may be easier than others—do O then M

$\phi_3(O) =$ T 0.2, F 0.8

$\phi_4(X, O, W) =$

TT	0.1
TF	0.9
FT	0.1
FF	0.9

$\phi_1(M) =$ T 0.9, F 0.1

$\phi_6(W) =$ T 0.1, F 0.9

$\phi_5(M, W, X) =$

TT	0.8
TF	0.1
FT	0
FF	0

$\phi_7(W) =$

T: $0.1 \times 0.9 \times 0.8 + 0.1 \times 0.1 \times 0 = .072$

F: $0.9 \times 0.9 \times 0.1 + 0.9 \times 0.1 \times 0 = .081$

$$P(W \mid Ra=T, Ru=F) = \frac{.072}{.072 + .081} = 8/17$$

move sum over O in:
sum_W phi1(M) phi5(M,W) sum_O phi3(O) phi4(O,W)
= sum_W phi1(M) phi5(M,W) phi6(W)
phi6(W) = sum_O phi3(O) phi4(O,W)
  T: 0.02+0.08 = 0.1
  F: 0.18+0.72 = 0.9

sum_M phi1(M) phi5(M,W) phi6(W)
phi7(W) =
  T: 0.1*0.9*0.8 + 0.1*0.1*0 = .072
  F: 0.9*0.9*0.1 + 0.9*0.1*0 = .081

renormalize: P(W) = T:8/17, F:9/17
    this is the answer!
    note: it’s easy to renorm now

FLOPs: 10, then 3 for renorm (+ earlier 2 = 15)
compare to full table method:
    8 relevant entries (M, O, W for Ra=T, Ru=F)
    4 mults each (5 phis): 32 flops
    normalize: sum (7 flops), divide (8 flops): 15 flops
    total = 47

Discussion

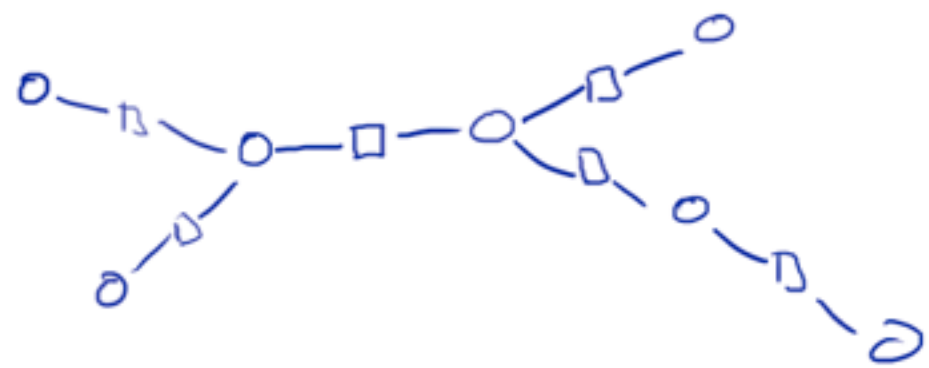
- Directed v. undirected: advantages to both
- Normalization
- Each elimination introduces a new table (all current neighbors of eliminated variable), makes some old tables irrelevant
- Each elim. order introduces different tables
- Some tables bigger than others
  - FLOP count; treewidth

importance of norm const: if we don't know it, need to compute it
Bnets: Z = 1 to start, so can answer some questions w/o inference; but once we've instantiated evidence, have a general factor graph (i.e., normalization is required)
Factor graphs: usually any question requires inference

Treewidth examples

Chain

Tree

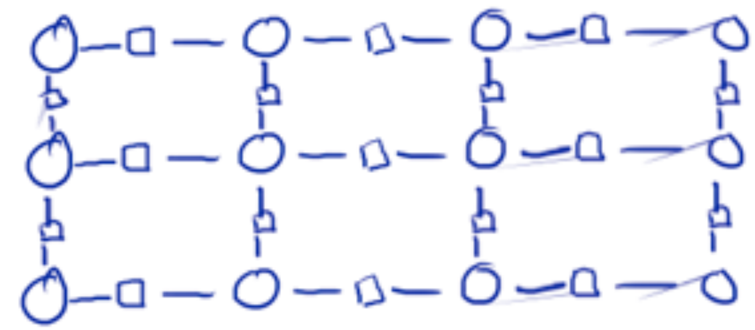


chain, tree = 1

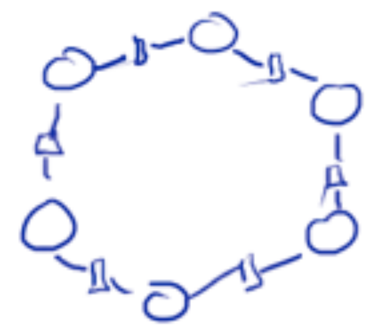
- chain = special case of tree
- tree: eliminate any leaf

Treewidth examples

Parallel chains



Cycle



parallel chains: #rows
eliminate down each column -- form a factor of #rows+1 just before eliminating last element of column
Cycle: 2
eliminate anything; we form a factor of size 3, then get back to a smaller cycle

Inference in general models

- Prior + evidence → (marginals of) posterior
  - several examples so far, but no general algorithm
- General algorithm: ***message passing***
  - aka ***belief propagation***
  - build a *junction tree*, instantiate evidence, pass messages (*calibrate*), read off answer, eliminate nuisance variables
- Share work of building JT among multiple queries
  - there are many possible JTs; different ones are better for different queries, so might want to build several

prior: a GM
evidence: observations at some nodes
posterior: resulting distribution after conditioning (incl renormalizing)

JT: also called “clique tree”—as with many other related problems, finding best JT for a given graphical model is NP-hard

BP: refers to “instantiate evidence, pass messages, read off answer” [but often building a JT and eliminating nuisance vars are assumed when we’re doing BP]

Better than variable elimination

- Suppose we want all 1-variable marginals
  - Could do N runs of variable elimination
  - Or: BP simulates N runs for the price of 2
- Further reading: Kschischang et al., "Factor Graphs and the Sum-Product Algorithm"

  www.comm.utoronto.ca/frank/papers/KFL01.pdf
- Or, Daphne Koller's book

or take the "graphical models" course...

What you need to understand

- How expensive will inference be?
  - what tables will be built and how big are they?
- What does a message represent and why?

each factor: a source of evidence (similar to a term in likelihood)
message: summary of evidence from one part of tree

Junction tree

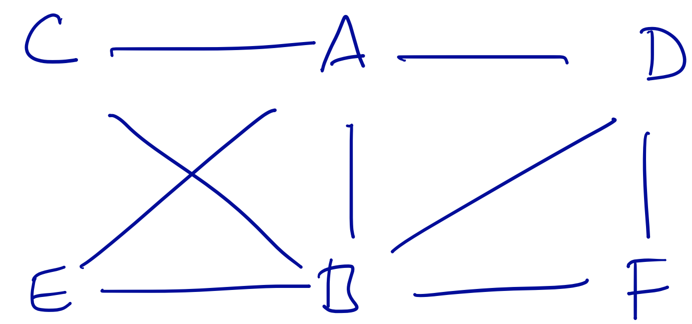
(aka clique tree, aka join tree)

- Represents the tables that we build during elimination
  - many JTs for each graphical model
  - many-to-many correspondence w/ elimination orders
- A junction tree for a model is:
  - a tree
  - whose nodes are sets of variables ("cliques")
  - that contains a node for each of our factors
  - that satisfies ***running intersection property*** (below)

nodes are cliques:
these are the tables we build

a node for each factor:
factor is a subset of that node's clique

Example network



- Elimination order: CEABDF
- Factors: ABC, ABE, ABD, BDF

Building a junction tree

(given an elimination order)

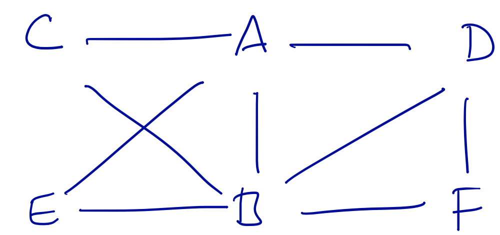
- $S_0 \leftarrow \varnothing$, $V \leftarrow \varnothing$ *[S = table args; V = visited]*
- For i = 1…n: *[elimination order]*
  - $T_i \leftarrow S_{i-1} \cup (\mathrm{nbr}(X_i) \setminus V)$ *[extend table to unvisited nbrs]*
  - $S_i \leftarrow T_i \setminus \{X_i\}$ *[marginalize out X_i]*
  - $V \leftarrow V \cup \{X_i\}$ *[mark X_i visited]*
- Build a junction tree from values S_i, T_i:
  - nodes: local maxima of T_i ($T_i \not\subseteq T_j$ for $j \neq i$)
  - edges: local minima of S_i (after a run of marginalizations without adding new nodes)

after for loop, it should be clear that each value of S or T corresponds to a table we have to reason about during variable elimination

if you've heard the phrase "moralize and triangulate", that's essentially what we're doing here

Example

CEABDF



T1 = CAB
S1 = AB
T2 = EAB
S2 = AB
T3 = ABD
S3 = BD
T4 = BDF
S4 = DF
T5 = DF
S5 = F
T6 = F
S6 = {}

messages: AB, AB, BD (the smaller marginal tables we multiply into larger (node) tables)
note: we can delay multiplying in messages

Edges, cont'd

- Pattern: $T_i \ldots S_{j-1}\ T_j \ldots S_{k-1}\ T_k \ldots$
- Pair each T with its following S (e.g., T_i w/ S_{j-1})
- Can connect T_i to T_k iff $k>i$ and $S_{j-1} \subseteq T_k$
- Subject to this constraint, free to choose edges
  - always OK to connect in a line, but may be able to skip

S increases and decreases in size: might add a lot of nodes at once (for a high-degree X) then eliminate several in a row without adding more (if all their neighbors are already in S)

T are local maxima, S are local minima

Running intersection property

- Once a node X is added to T, it stays in T until eliminated, then never appears again
- In JT, this means all sets containing X form a connected region of tree
  - true for all X = running intersection property

*** mark where we use RIP later

*** note: largest clique is size treewidth+1

Moralize & triangulate

Instantiate evidence

- For each factor:
  - fix known arguments
  - assign to some clique containing all non-fixed arguments

define sepset ***

Pass messages (belief propagation)

Read off answer

- Find some subtree that contains all variables of interest
- Compute distribution over variables mentioned in this subtree
- Marginalize (sum out) nuisance variables

depending on query and JT, might have a lot of nuisance variables

*** make a running example?

Hard v. soft factors

Hard

X

Y		0	1	2
	0	0	0	0
	1	0	0	1
	2	0	1	1

Soft

X

Y		0	1	2
	0	1	1	1
	1	1	1	3
	2	1	3	3

number = degree to which event is more or less likely
must be nonnegative

0 = hard constraint

can combine hard & soft (some numbers zero, others positive and varying)

hard factors can lead to complications (e.g., impossible to satisfy all constraints; e.g., Koller ex 4.4 (may not be able to factor according to a graph that matches our actual set of independences, i.e., failure of Hammersley-Clifford))

we'll mostly be using soft factors

Factor graph → Bayes net

- Conversion possible, but more involved
  - Each representation can handle ***any*** distribution
  - But, size/complexity of graph may differ
- 2 cases for conversion:
  - without adding nodes:
  - adding nodes:

Without adding nodes: #P-complete (i.e., we think exp-time)
Adding nodes: poly-time, but we get a bigger Bayes net
won't cover algorithm

chordal graphs are precisely the graphs that turn directly into both factor graphs / MRFs and Bayes nets