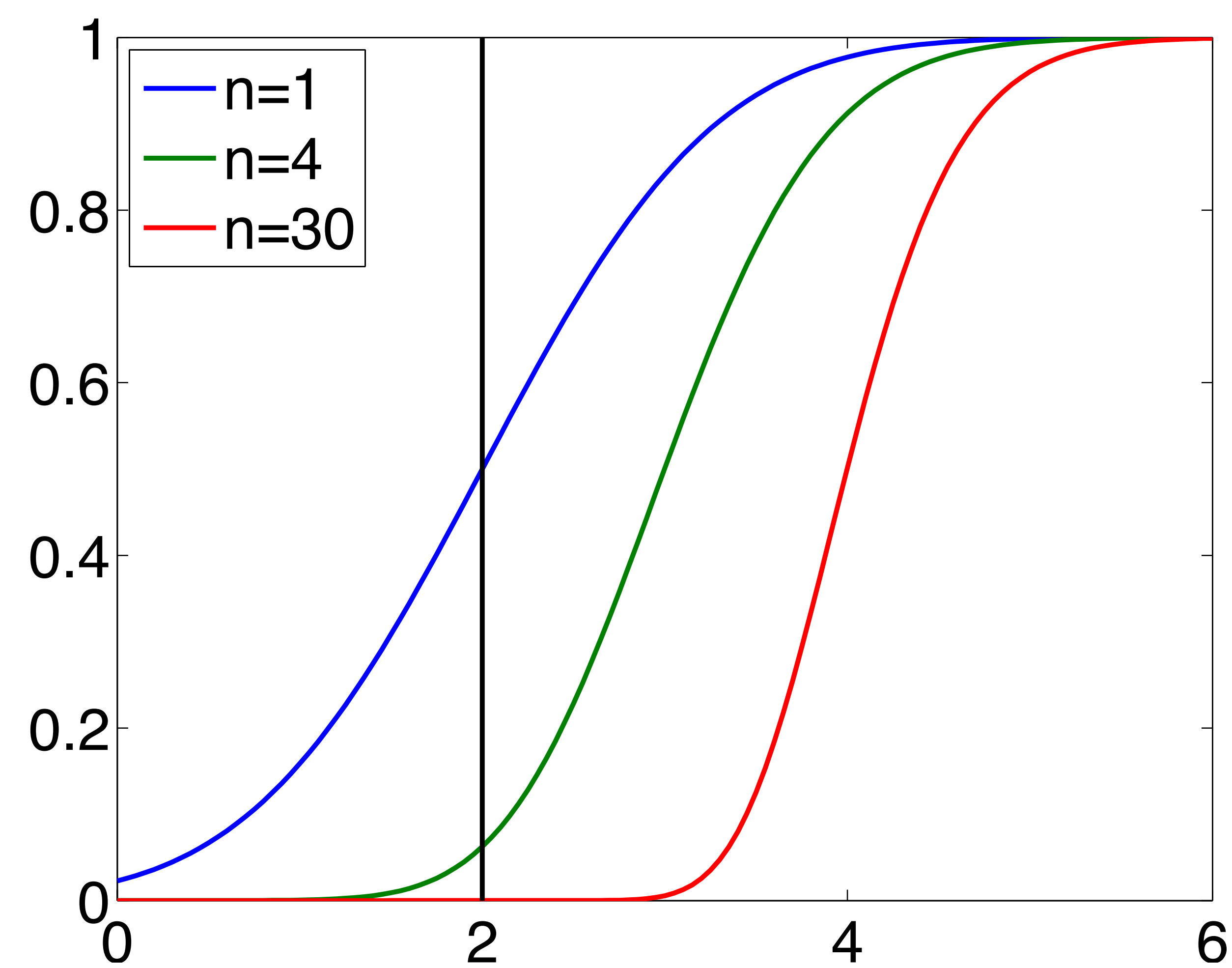
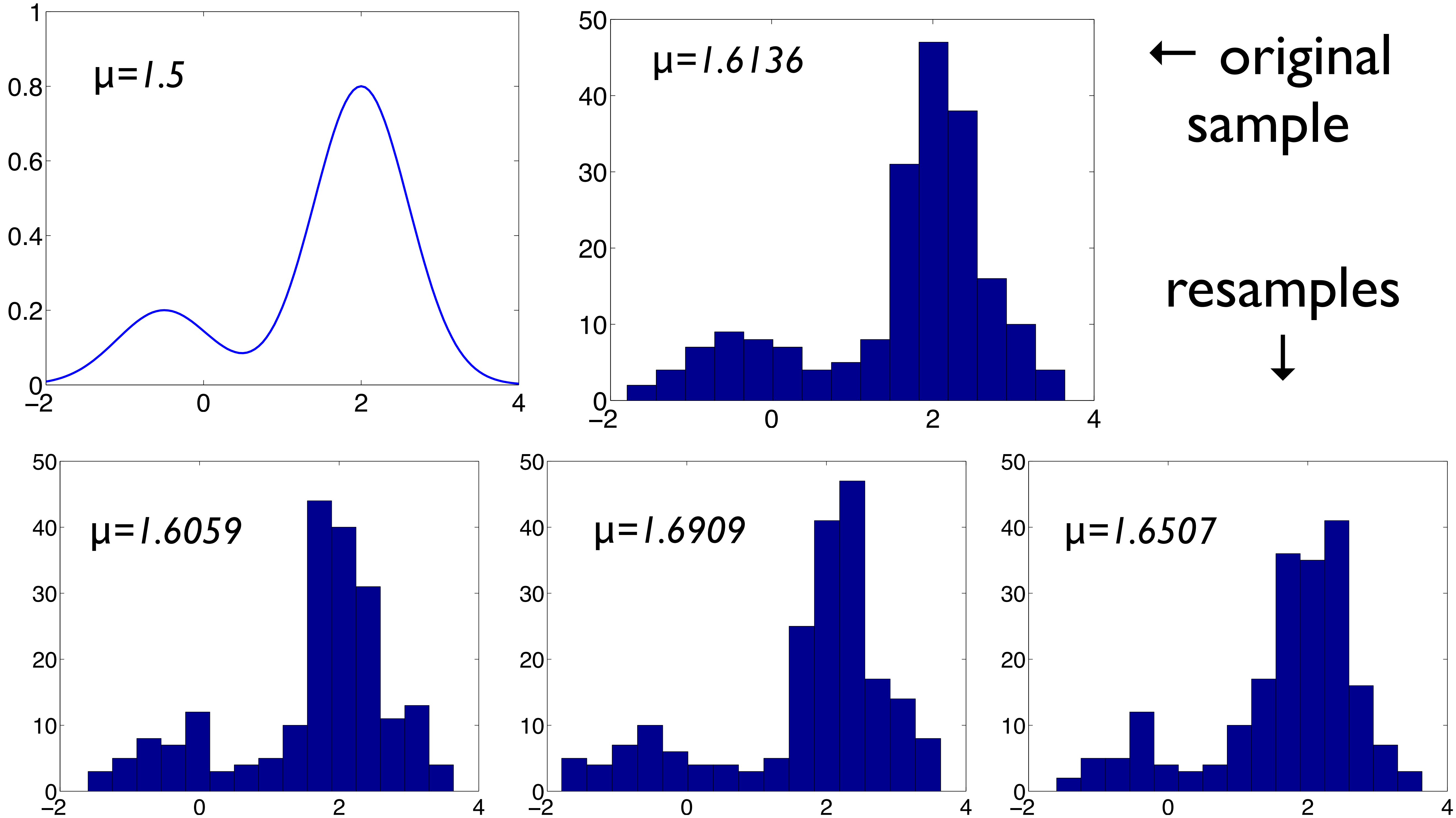
Review

- Selection bias, overfitting
- Bias v. variance v. residual
- Bias-variance tradeoff
  - Cramér-Rao bound

CDF of max of n samples of
N(μ=2, σ^2=1)
[representing error estimates for n models]



Review: bootstrap



Repeat 100k times: est. stdev of \hat\mu = 0.0818
compare to true stdev, .0825

Cross-validation

- Used to estimate classification error, RMSE, or similar error measure of an algorithm
- Surrogate sample: exactly the same as $x_1, \ldots, x_N$ except for train-test split
- k-fold CV:
  - randomly permute $x_1, \ldots x_N$
  - split into *folds*: first N/k samples, second N/k samples, …
  - train on k–1 folds, measure error on remaining fold
  - repeat k times, with each fold being holdout set once

f = function from whole sample to single number = train model on k−1 folds then evaluate error on remaining one

CV: uses sample splitting idea twice
first: split into train & validation
second: repeat to estimate variability
only the second is approximated

k = N: leave−one−out CV (LOOCV)

Cross-validation: caveats

- Original sample might not be i.i.d.
- Size of surrogate sample is wrong:
  - want to estimate error we'd get on a sample of size N
  - actually use samples of size N(k–1)/k
- Failure of i.i.d, even if original sample was i.i.d.

two of these are potentially optimistic; middle one is conservative (but usually pretty small effect)

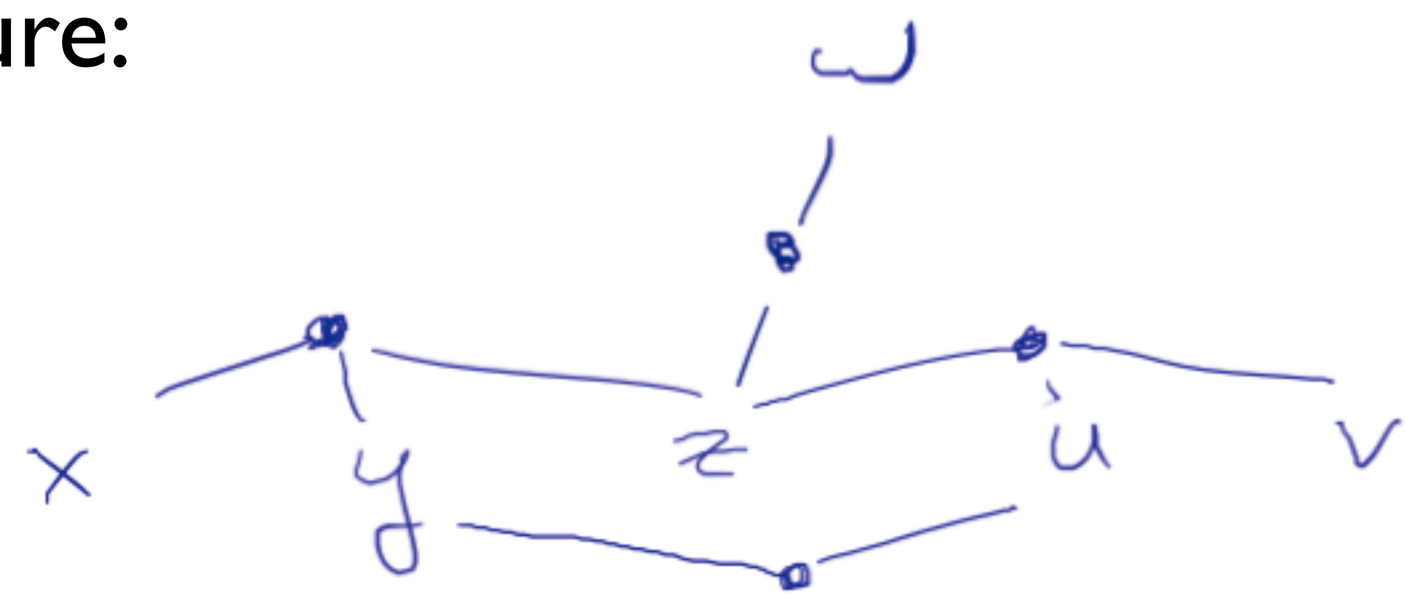
Graphical models

Dynamic programming on a graph

- Probability calculation problem (all binary vars, p=0.5):

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

- Essentially an instance of #SAT
- Structure:



===

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

Variable elimination

$$\sum_x \sum_y \sum_z \sum_u \sum_v \sum_w A(xyz)\, B(yu)\, C(zw)\, D(zuv)$$

A

TTT	1
TTF	1
TFT	1
TFF	1
FTT	1
FTF	1
FFT	0
FFF	1

B

TT	0
TF	1
FT	1
FF	1

C

TT	1
TF	1
FT	1
FF	0

D

TTT	1
TTF	1
TFT	1
TFF	1
FTT	1
FTF	1
FFT	1
FFF	0

(leaving off normalizer of 1/2^6)
move in sum over w: get sum_w C(zw) = table
  E(z): 1: 2, 0: 1
move in sum over v: get sum_uv D(zuv) = table
  F(zu): 11: 2, 10: 2, 01: 2, 00: 1
move in sum over u: get sum_u B(yu) F(zu)
  BF(yzu): (0 1 0 1 1 1 1 1) * (2 2 2 1 2 2 2 1)
    = 0 2 0 1 2 2 2 1
  sum over u: G(yz) = 2 1 4 3
write out EGA(xyz): (2 1 2 1 2 1 2 1) * (2 1 4 3 2 1 4 3) * A
  = (4 1 8 3 4 1 0 3)
sum over xyz: 24 satisfying assignments

Variable elimination

In general

- Pick a variable ordering
- Repeat: say next variable is z
  - move sum over z inward as far as it goes
  - make a new table by multiplying all old tables containing z, then summing out z
  - arguments of new table are "neighbors" of z
- Cost: O(size of biggest table * # of sums)
  - sadly: biggest table can be exponentially large
  - but often not: low-treewidth formulas

neighbors: share a table
note that vars can become neighbors when we delete old tables and add a new table

treewidth = #args of largest table - 1
(for best elimination ordering)

Why did we do this?

- A simple graphical model!
- Graphical model = graphical representation + statistical model
  - in our example: graph of clauses & variables, plus coin flips for variables

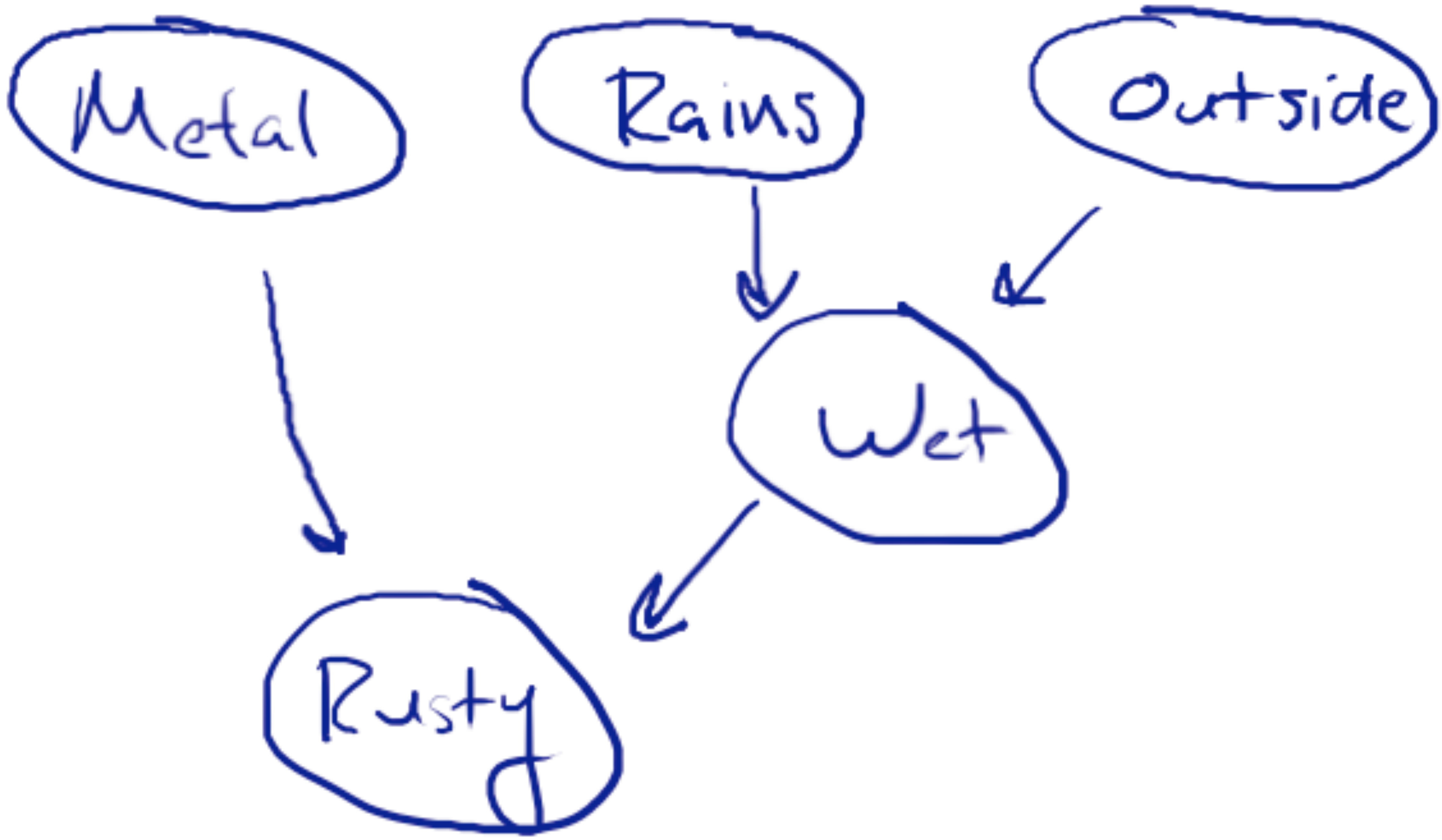
Why do we need graphical models?

- Don't want to write a distribution as a big table
  - Gets unwieldy fast!
  - E.g., 10 RVs, each w/ 10 settings
  - Table size = 10^{10}
- Graphical model: way to write distribution compactly using diagrams & numbers
- Typical GMs are huge (10^{10} is a small one), but we'll use tiny ones for examples

Bayes nets

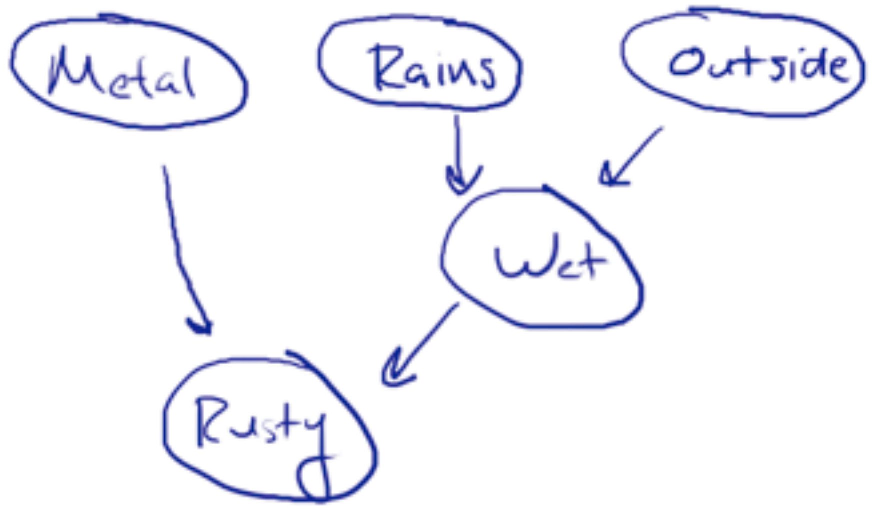
- Best-known type of graphical model
- Two parts: DAG and CPTs

Rusty robot: the DAG



node = RV
arcs: indicate probabilistic dependence
    rusty: metal, wet
    wet: rains, outside
define: pa(X) = parent set
    e.g., pa(rusty) = metal, wet

Rusty robot: the CPTs

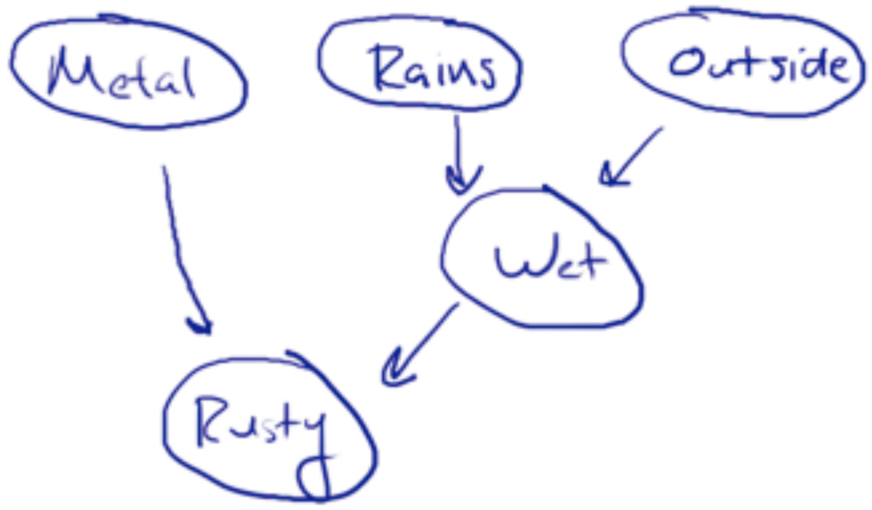


P(Metal) = 0.9
P(Rains) = 0.7
P(Outside) = 0.2
P(Wet | Rains, Outside)
TT: 0.9 TF: 0.1
FT: 0.1 FF: 0.1
P(Rusty | Metal, Wet) =
TT: 0.8 TF: 0.1
FT: 0 FF: 0

- For each RV (say X), there is one CPT specifying P(X | pa(X))

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

Interpreting it



P(RVs) = prod_{X in RVs} P(X | pa(X))
P(M, Ra, O, W, Ru) = P(M)P(Ra)P(O)P(W|Ra,O)P(Ru|M,W)

Write out part of table:

Met	Rai	Out	Wet	Rus	P(...)
F	F	F	F	F	.1*.3*.8*.9*1 = .0216
F	F	F	F	T	.1*.3*.8*.9*0 = 0
...					
T	T	T	T	T	.9*.7*.2*.9*.8 = 0.0907

Note: 11 numbers (instead of 2^5 - 1 = 31)
just gets better as #RVs increases

Benefits

- 11 v. 31 numbers
- Fewer parameters to learn
- Efficient ***inference*** = computation of marginals, conditionals ⇒ posteriors

Inference Qs

- Is Z > 0?
- What is P(E)?
- What is $P(E_1 \mid E_2)$?
- Sample a random configuration according to P(.) or P(. | E)
- Hard part: taking sums over r.v.s (e.g., sum over all values to get normalizer)

Z = 0: probabilities undefined

why is Z hard? exponentially many configurations

other than Z, it's just a bunch of table lookups

Inference example

- P(M, Ra, O, W, Ru) =

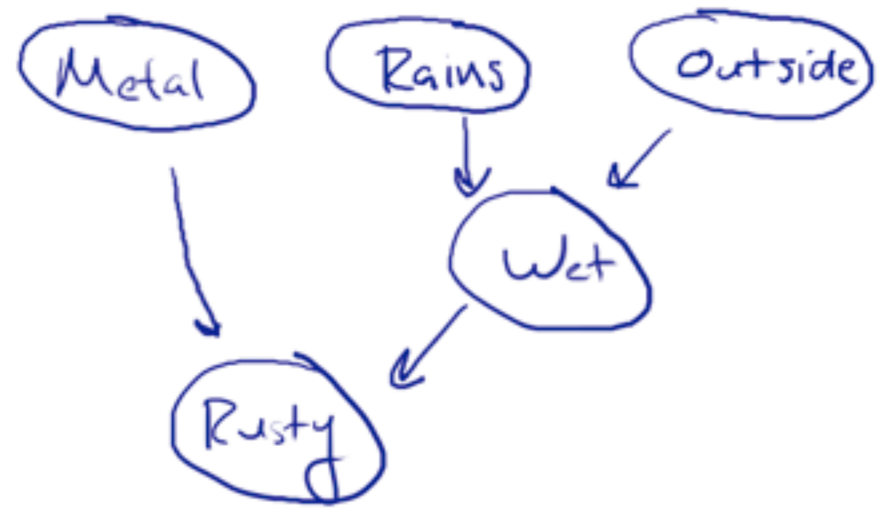
  P(M) P(Ra) P(O) P(W|Ra,O) P(Ru|M,W)

- Find marginal of M, O

sum_Ra in 0,1 sum_W in 0,1 sum_Ru in 0,1
P(M) P(Ra) P(O) P(W|Ra,O) P(Ru|M,W)
= sum_Ra sum_W P(M) P(Ra) P(O) P(W|Ra,O) sum_Ru P(Ru|M,W)
= sum_Ra sum_W P(M) P(Ra) P(O) P(W|Ra,O)
= sum_Ra P(M) P(Ra) P(O) sum_W P(W|Ra,O)
= sum_Ra P(M) P(Ra) P(O)
= P(M) P(O)
note: so far, no actual arithmetic (all analytic, true for *any* CPTs)
now can write P(M, O) using 4 multiplications (using CPTs)
.9, .7 (.63 .07 .27 .03)
note: M & O are independent

Independence

- Showed M ⊥ O
- Any other independences?



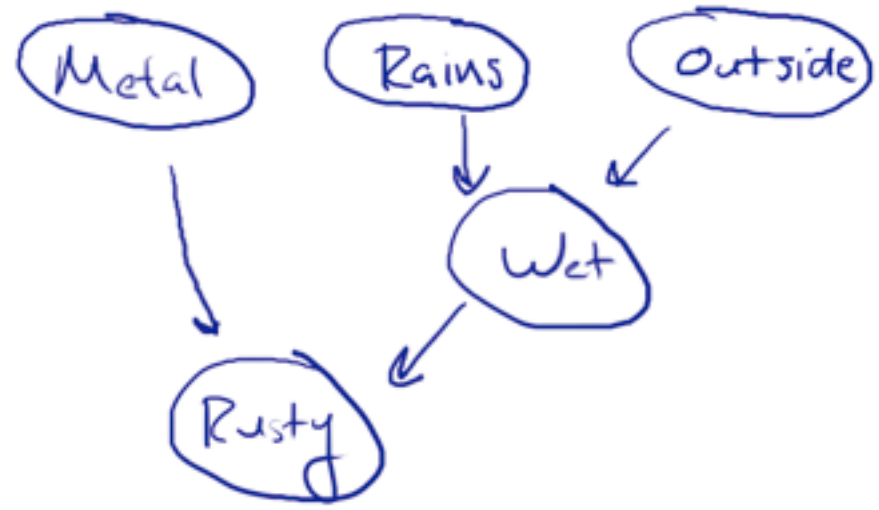
- Didn't use CPTs: some independences depend only on graph structure
- May also be "accidental" independences
  - i.e., depend on values in CPTs

note new symbol ⊥
M ⊥ R   R ⊥ O
M ⊥ W

didn't use CPTs ==> these hold for *all* CPTs
    depend only on graph structure

accidental = depend on values in CPTs
    e.g.: P(W I Ra, O) = .3 .3 .3 .3 yields W ⊥ Ra, O
    note that even a tiny change in CPT voids this

Conditional independence



- How about O, Ru? O Ru
- Suppose we know we're not wet
- P(M, Ra, O, W, Ru) =
  - P(M) P(Ra) P(O) P(W|Ra,O) P(Ru|M,W)
- Condition on W=F, find marginal of O, Ru

O not indep Ru

sum_M sum_Ra P(M) P(Ra) P(O) P(W=FIRa,O) P(RuIM,W=F) / P(W=F)
 = [sum_Ra P(Ra) P(O) P(W=FIRa,O)] [sum_M P(M) P(RuIM,W=F) / P(W=F)]
 = factored!

O ! Ru I W=F
again, true no matter what CPTs are

Conditional independence

- This is generally true
  - conditioning can make or break independences
  - many conditional independences can be derived from graph structure alone
  - accidental ones often considered less interesting
- We derived them by looking for factorizations
  - turns out there is a purely graphical test
  - one of the key contributions of Bayes nets

less interesting: *except* context-specific

Example: blocking

- Shaded = observed (by convention)

Rains --> Wet --> Rusty
P(Ra) P(W | Ra) P(Ru | W)

Rains --> Wet (shaded) --> Rusty
P(Ra) P(W=T | Ra) P(Ru | W=T) / P(W=T)
[P(Ra) P(W=T | Ra)] [P(Ru | W=T) / P(W=T)]

Ra ⊥ Ru | W

Example: explaining away

- Intuitively:

Rains --> Wet <-- Outside
already showed Ra ! O
sum_W P(Ra) P(O) P(W | Ra, O) = P(Ra) P(O)

Rains --> Wet (shaded) <-- Outside
P(Ra) P(O) P(W = F | Ra, O) / P(W=F)
became dependent! Ra not indep O | W

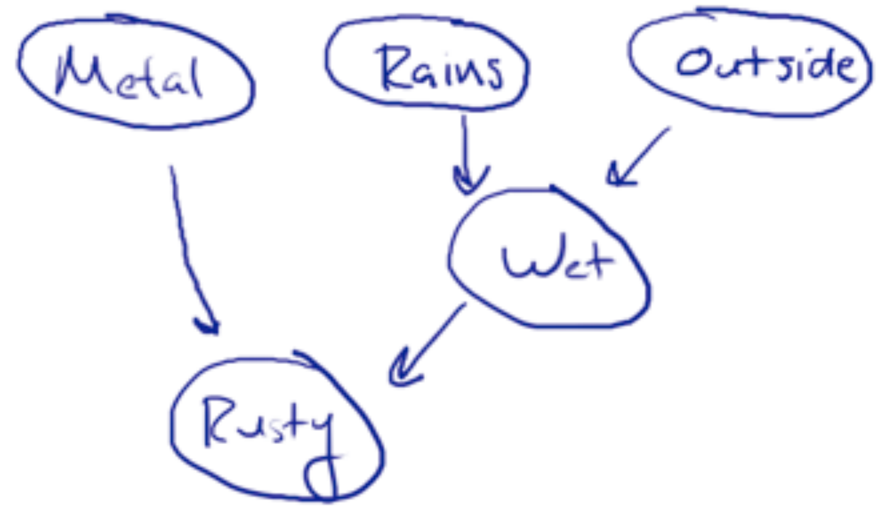
intuitively: If we know we're not wet, suppose we find out it's raining: then we know we're probably not outside

d-separation

- General graphical test: "d-separation"
  - d = dependence
- $X \perp Y \mid Z$ when there are no active paths between X and Y
- 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 is active if:

  and inactive o/w

- Path is active if intermediate nodes are

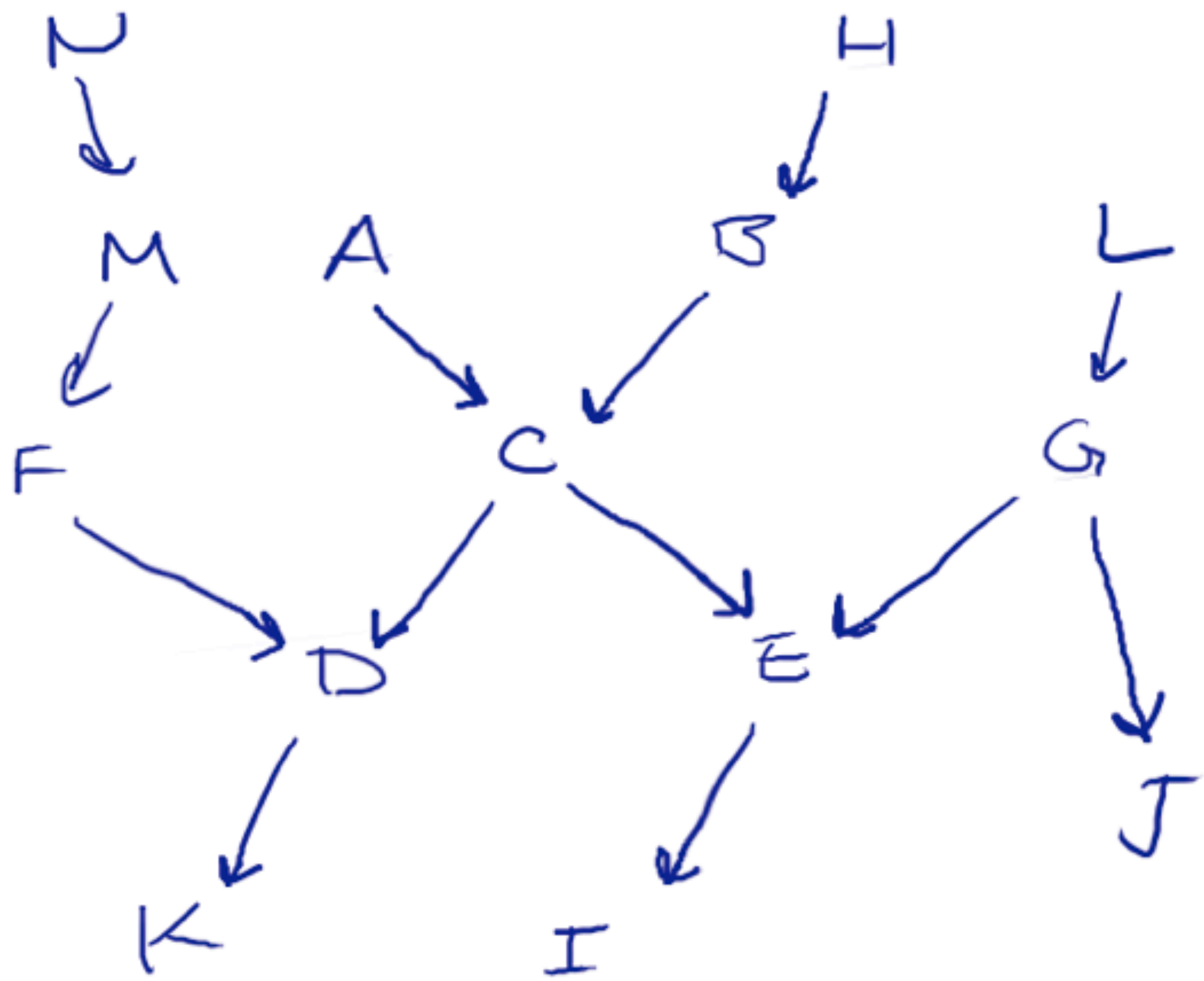
active if
unshaded and arrows are >>, <<, or <>
shaded (or descendant shaded) and arrows >< (collider)

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

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

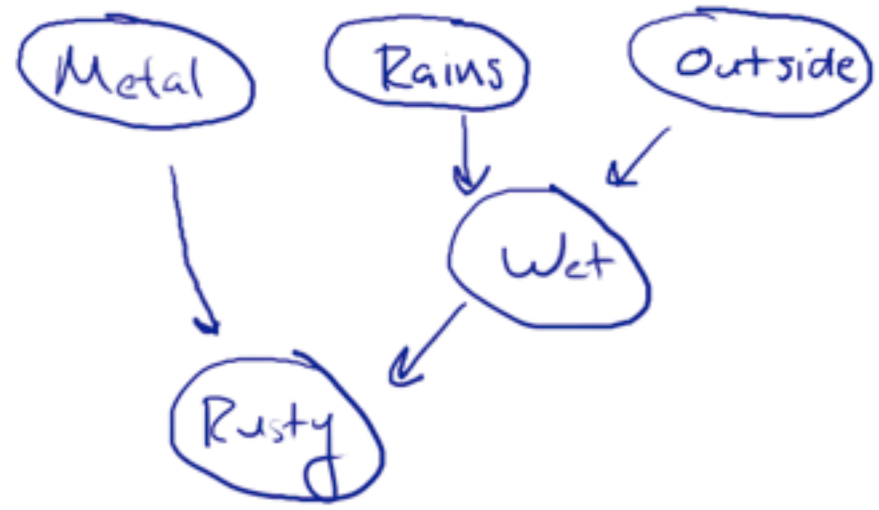
Markov blanket

- 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 --> Laplace smoothing
note extreme probabilities

Limitations of counting

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