15-780: Grad AI
Lecture 19: Graphical models, Monte Carlo methods

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman

Admin

Review W eve
5-ish

- Reminder: midterm March 29
- Reminder: project milestone reports due March 31

Review: scenarios

- Converting QBF+ to PBI/MILP by scenarios
  - Replicate decision variables for each scenario
  - Replicate clauses: share first stage vars; set scenario vars by scenario index; replace decision vars by replicates
  - Sample random scenarios
- Example: PSTRIPS

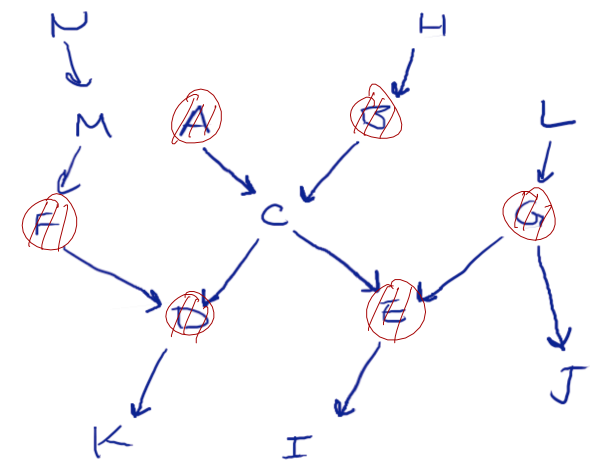
Review: dynamic programming

- Solving #SAT by dynamic programming (variable elimination)
  - repeatedly move sums inward, combine tables, sum out
  - treewidth and runtime/space

Review: graphical models

- Bayes net = DAG + CPTs
  - For each RV (say X), there is one CPT specifying P(X | pa(X))
  - Can simulate with propositional logic + random causes
- Inference: similar to #SAT DP—move sums inward
  - Can do partly analytically
  - Allows us to prove independences and conditional ind's from DAG alone

Review: graphical models

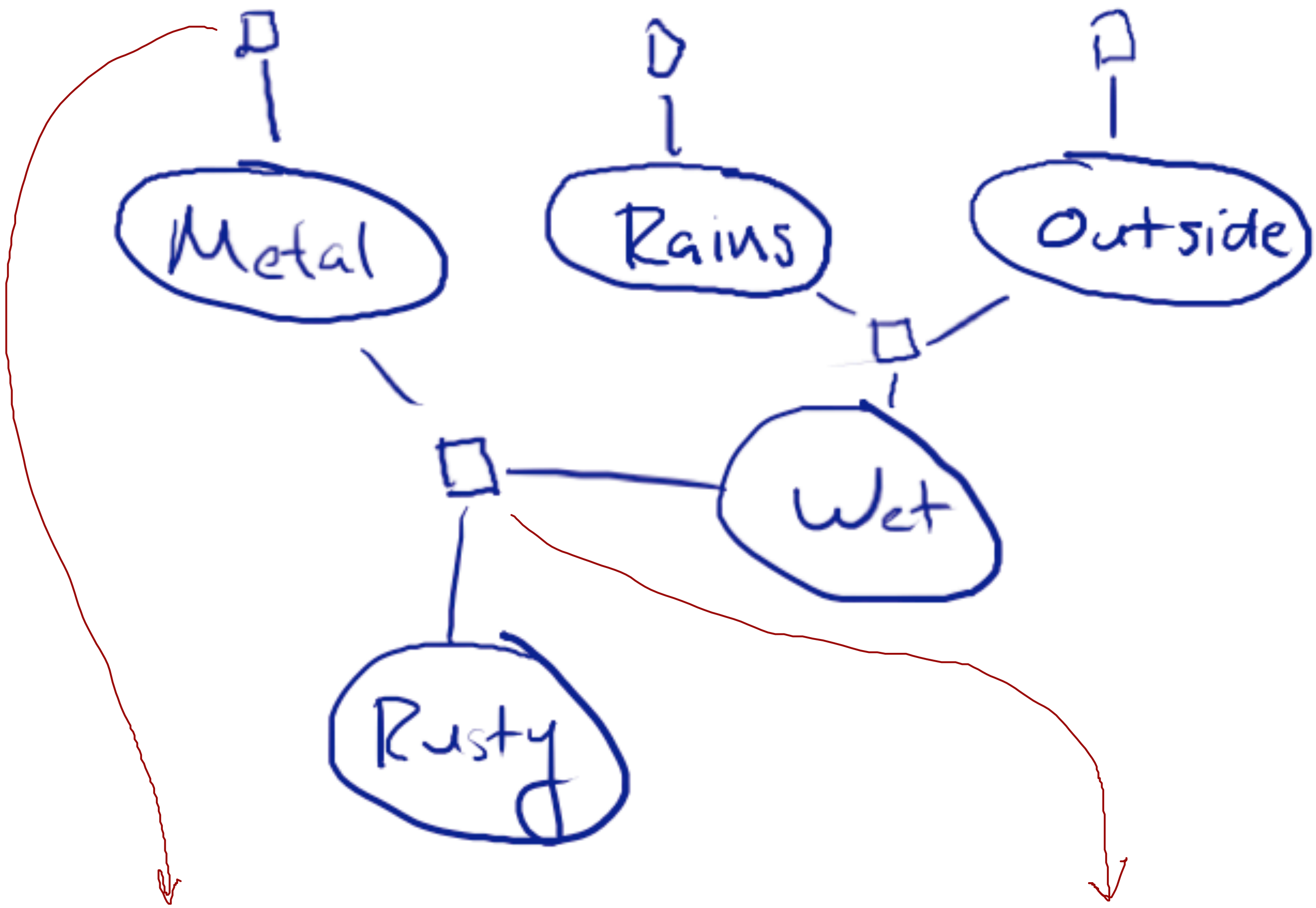


- Blocking, explaining away
- Markov blanket
- Learning: counting, Laplace smoothing
  - if hidden variables: take 10-708 or use a toolbox

Factor graphs

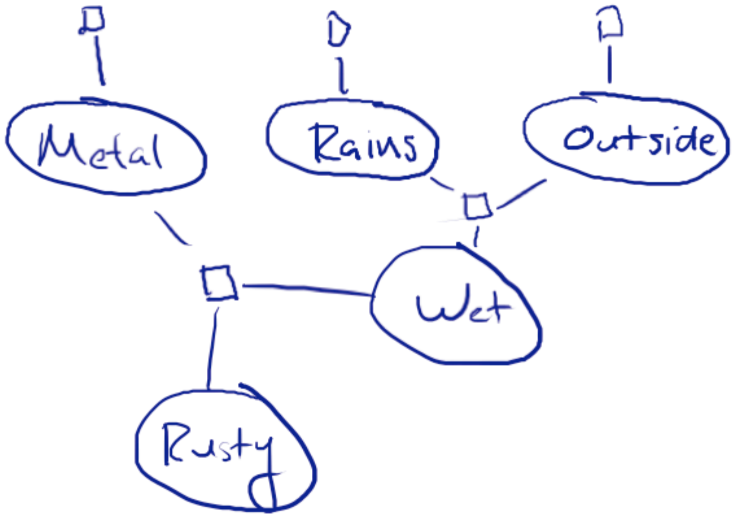
- Another common type of graphical model
- Uses ***undirected, bipartite*** graph instead of DAG

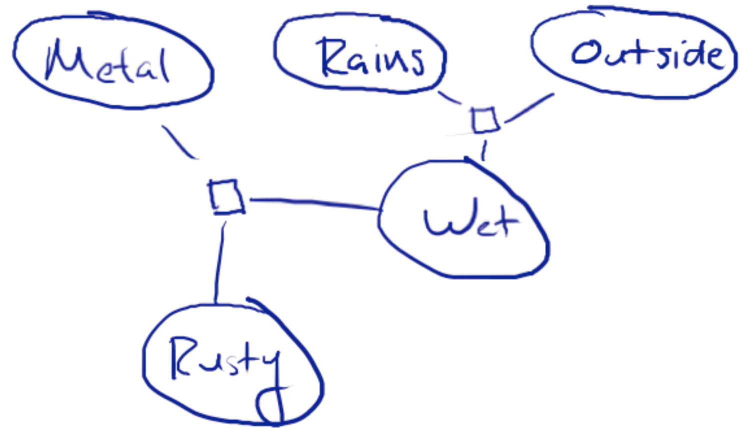
Rusty robot: factor graph



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

Convention





- Don't need to show unary factors
- Why? They don't affect algorithms below.

Non-CPT factors

- Just saw: easy to convert Bayes net → factor graph
- In general, factors need not be CPTs: any nonnegative #s allowed
- In general, P(A, B, …) = $\frac{1}{Z} \prod_{i \in \text{factors}} \phi_i(X_{nbr(i)})$
- Z = $\sum_{x_1} \sum_{x_2} \sum_{x_3} \cdots \prod_i \phi_i(X_{nbr(i)})$

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

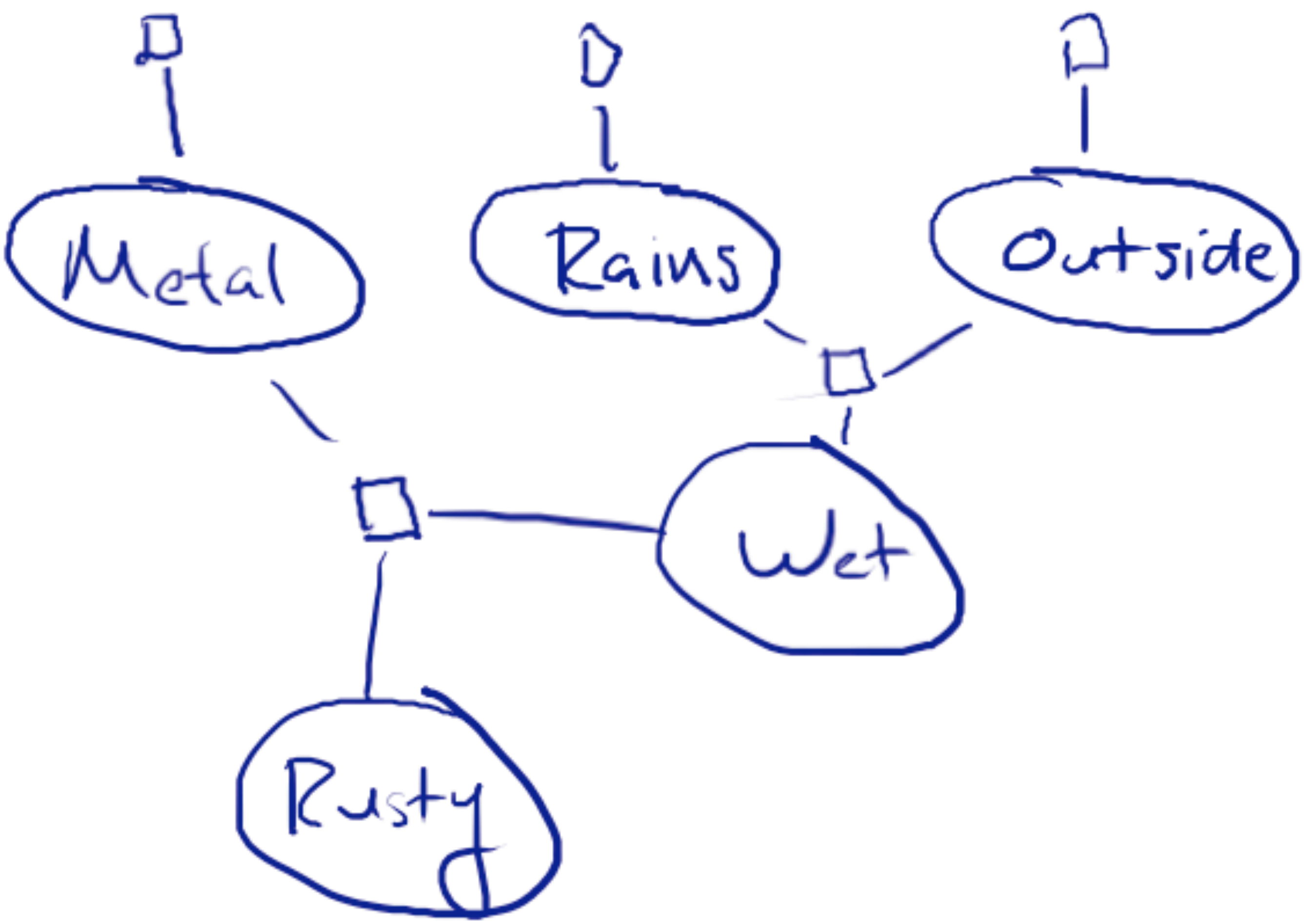
Factor graph → Bayes net

- 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: #P-complete
  - adding nodes: linear time

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



$M \not\perp O$

$M \not\perp O \mid Ru$

$M \perp O \mid W$

Modeling independence

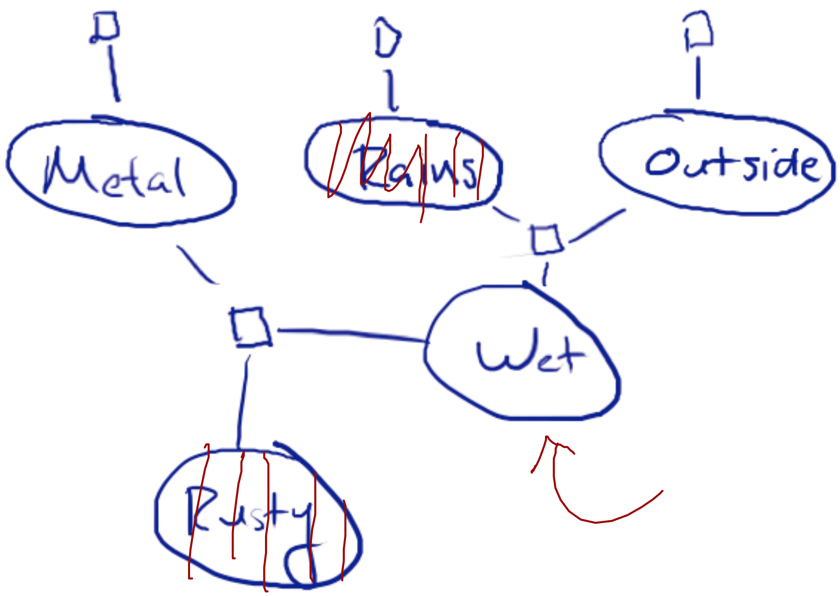
- Take a Bayes net, list the (conditional) independences
- Convert to a factor graph, list the (conditional) independences
- Are they the same list? No
- What happened?

Inference

- Inference: prior + evidence → posterior
- We gave examples of inference in a Bayes net, but not a general algorithm
- Reason: general algorithm uses factor-graph representation
- Steps: instantiate evidence, eliminate nuisance nodes, normalize, answer query

Inference

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



$\phi_1(M) =$

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

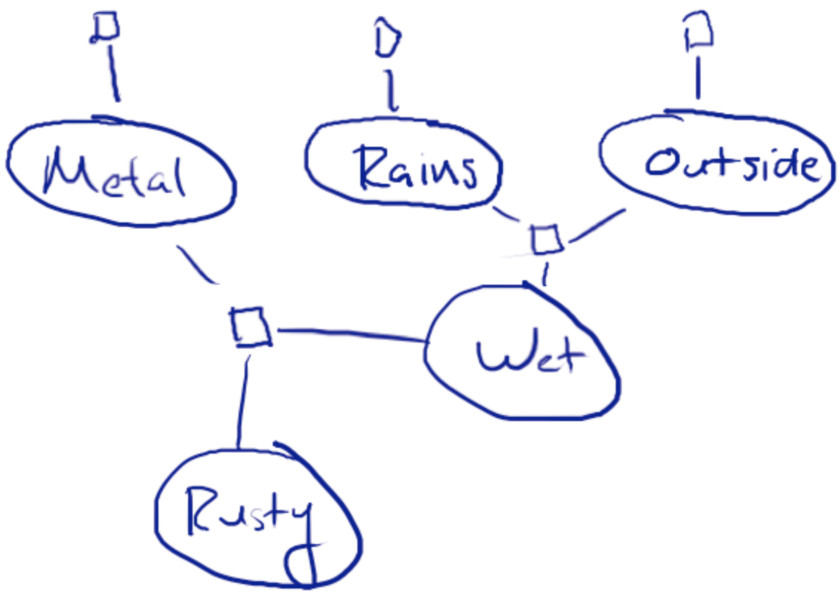
$\phi_5(M, W, Ru) =$

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)?

Incorporate evidence

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



$\phi_1(M) =$

T	0.9
F	0.1

$\cancel{\phi_2(Ra)} =$

~~T~~	~~0.7~~
~~F~~	~~0.3~~

$\phi_3(O) =$

T	0.2
F	0.8

$\phi_4(\cancel{Ra}, O, W) =$

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

$\phi_5(M, W, \cancel{Ru}) =$

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

Eliminate nuisance nodes

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

- Remaining nodes: M, O, W
- Query: P(W)
- So, O&M are nuisance—marginalize away
- Marginal = $\frac{1}{Z} \sum_{O} \sum_{M} \phi_1(M)\,\phi_3(O)\,\phi_4(F, O, W)\,\phi_5(M, W, T)$

Elimination order

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

$$\bar\phi_6(W)$$

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

$\phi_3(O) =$

T	0.2
F	0.8

$\phi_4(\cancel{M}, O, W) =$

TT	0.1
TF	0.9
FT	0.1
FF	0.9

$\phi_6(O,W)$

T T	.02
T F	.18
F T	.08
F F	.72

$\bar\phi_6(W)$

T	.1
F	.9

FLOPS
$4+2=6$

One last elimination

$$P(W \mid \ldots) = \bar{\Phi}_6(W)\, \bar{\Phi}_7(W)\, \frac{1}{Z}$$

$\Phi_7(M, W)$

T T	.72
T F	.09
F T	0
F F	0

$\bar{\Phi}_7(W)$

T	.72
F	.09

$P(W \mid \ldots)$

.072 / Z

.081 / Z

8/17

9/17

FLOPS

4 + 2 + 2 + 3

= 11

$\Phi_1(M) =$

T	0.9
F	0.1

$\bar{\Phi}_6(W) =$

T	0.1
F	0.9

$\Phi_5(M, W, \cancel{R}) =$

T T	0.8
T F	0.1
F T	0
F F	0

Checking our work

- http://www.aispace.org/bayes/version5.1.6/bayes.jnlp

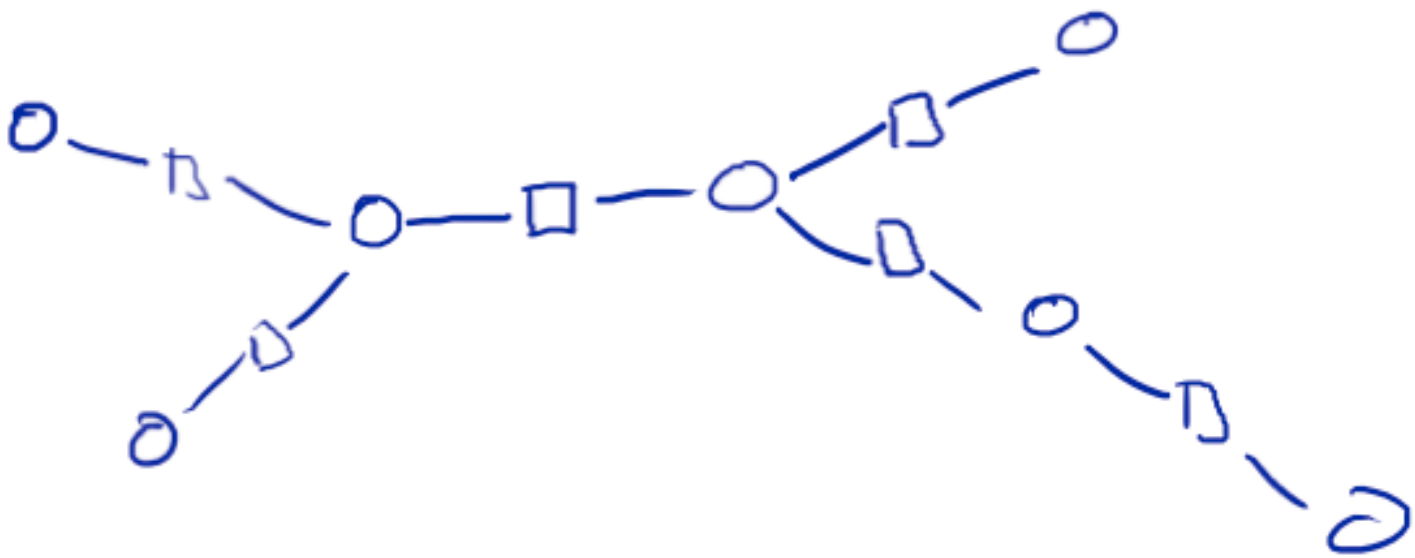
Discussion

- Steps: instantiate evidence, eliminate nuisance nodes, normalize, answer query
  - each elimination introduces a new table, makes some old tables irrelevant
- Normalization
- Each elim. order introduces different tables
  - some tables bigger than others
- FLOP count; treewidth

Treewidth examples

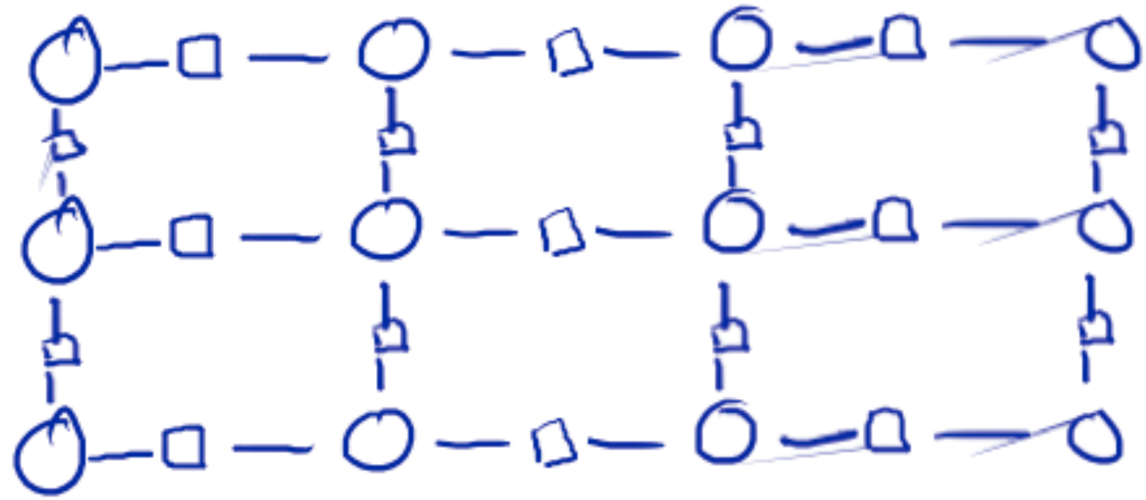
Chain

Tree

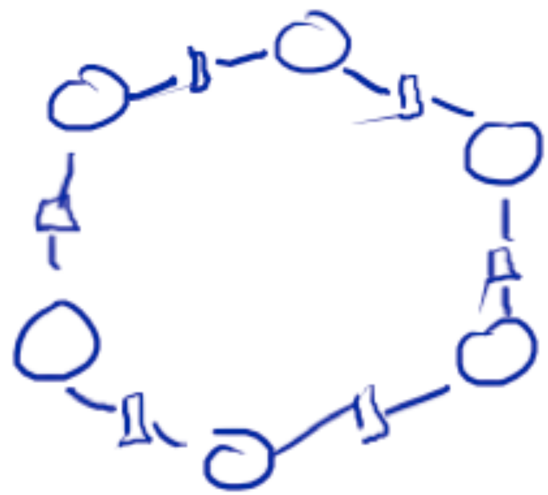


Treewidth examples

Parallel chains



Cycle



Discussion

- Several relationships between GMs and logic (similar DP algorithm, use of independent choices + logical consequences to represent a GM, factor graph with 0-1 potentials = CSP, MAP assignment = ILP)
- Directed v. undirected: advantages to both
- Lifted reasoning
  - Propositional logic + objects = FOL
  - FO GMs are a current hot topic of research (plate models, MLNs, ICL)—not solved yet!

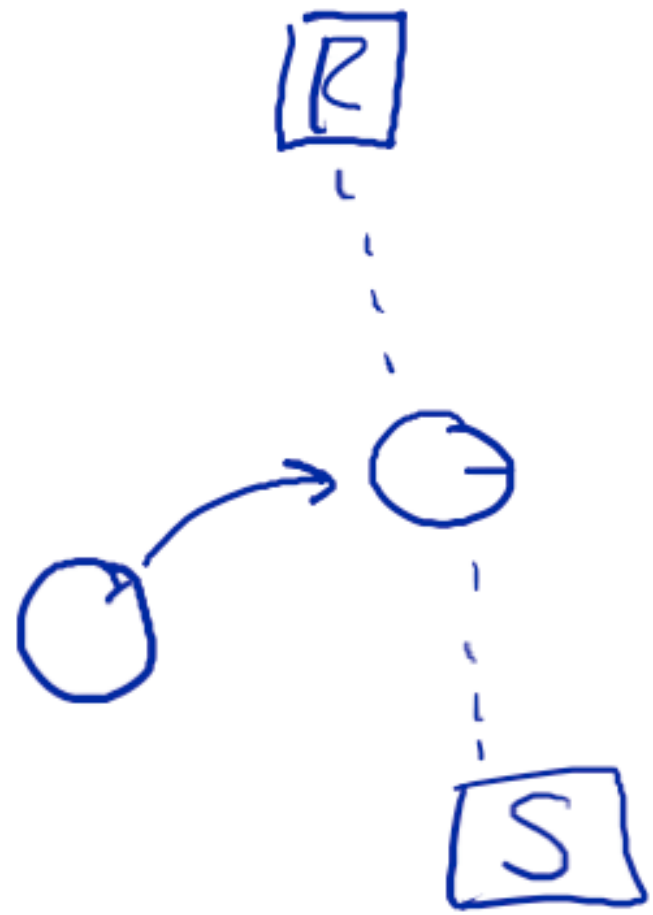
Discussion: belief propagation

- Suppose we want all 1-variable marginals
- Could do N runs of variable elimination
- Or: the BP algorithm simulates N runs for the price of 2
- For details: Kschischang et al. reading

HMMs and DBNs

Inference over time

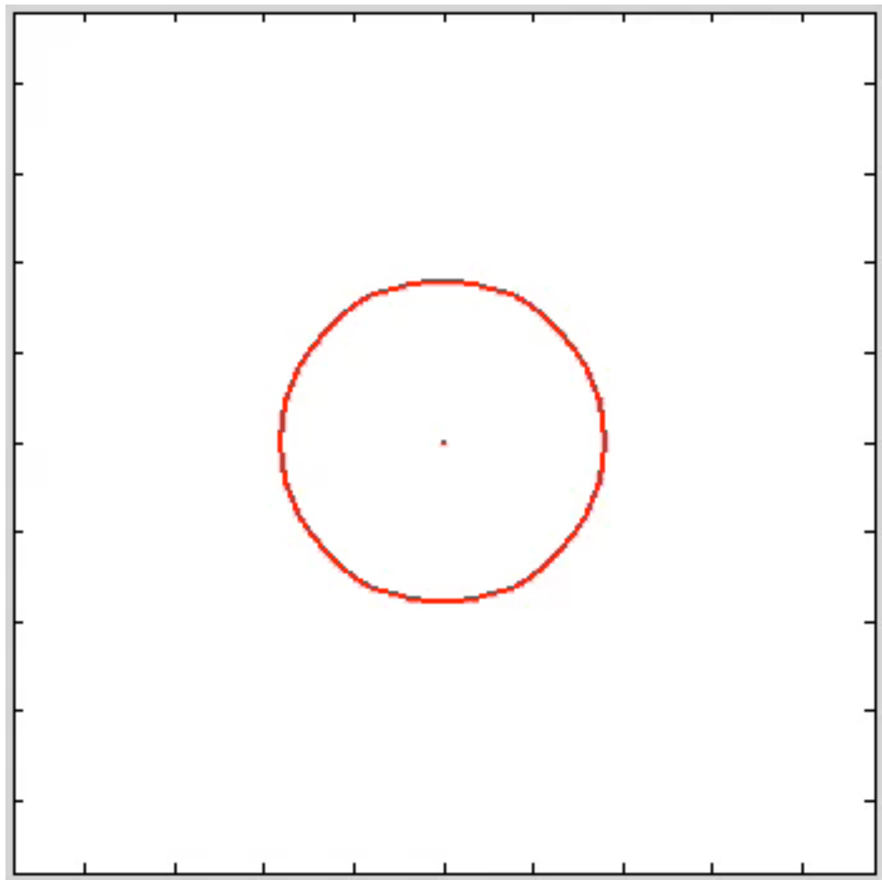
- Consider a robot:
  - true state (x, y, θ)
  - controls (v, w)
  - N range sensors (here N=2: r, s)



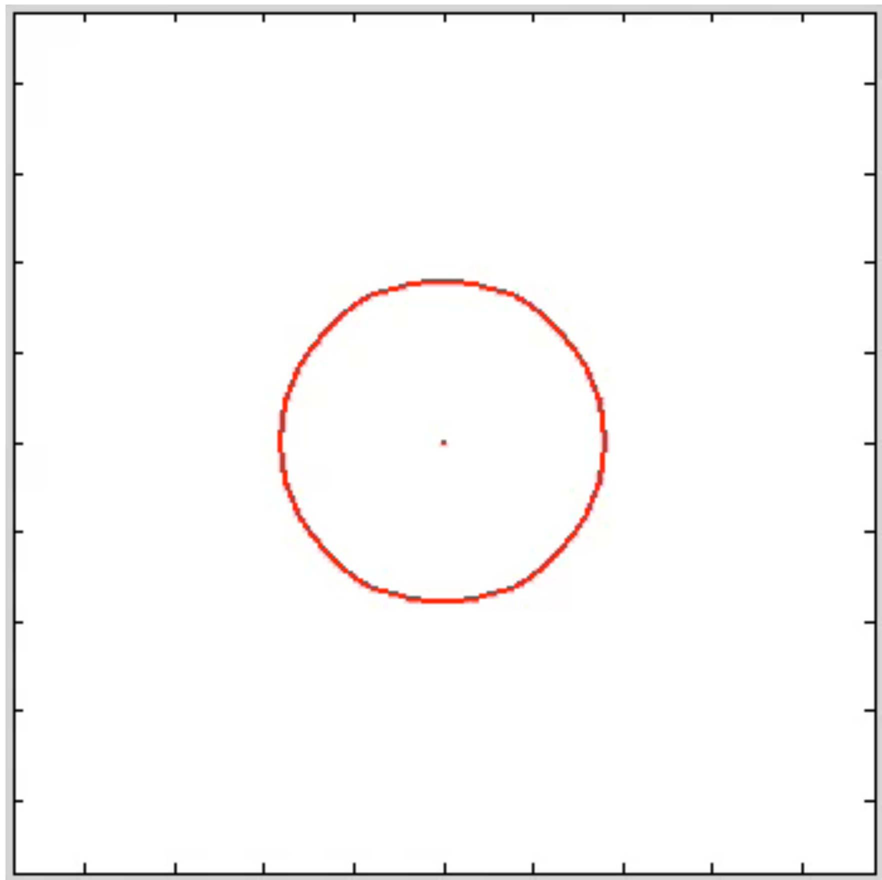
Model

$$
\begin{aligned}
x_{t+1} &= x_t + v_t \cos\theta_t + \text{noise} \\
y_{t+1} &= y_t + v_t \sin\theta_t + \text{noise} \\
\theta_{t+1} &= \theta_t + w_t + \text{noise} \\
r_t &= \sqrt{(x_t - x^R)^2 + (y_t - y^R)^2} + \text{noise} \\
s_t &= \sqrt{(x_t - x^S)^2 + (y_t - y^S)^2} + \text{noise}
\end{aligned}
$$

Model of x, y, θ (r, s unobserved)

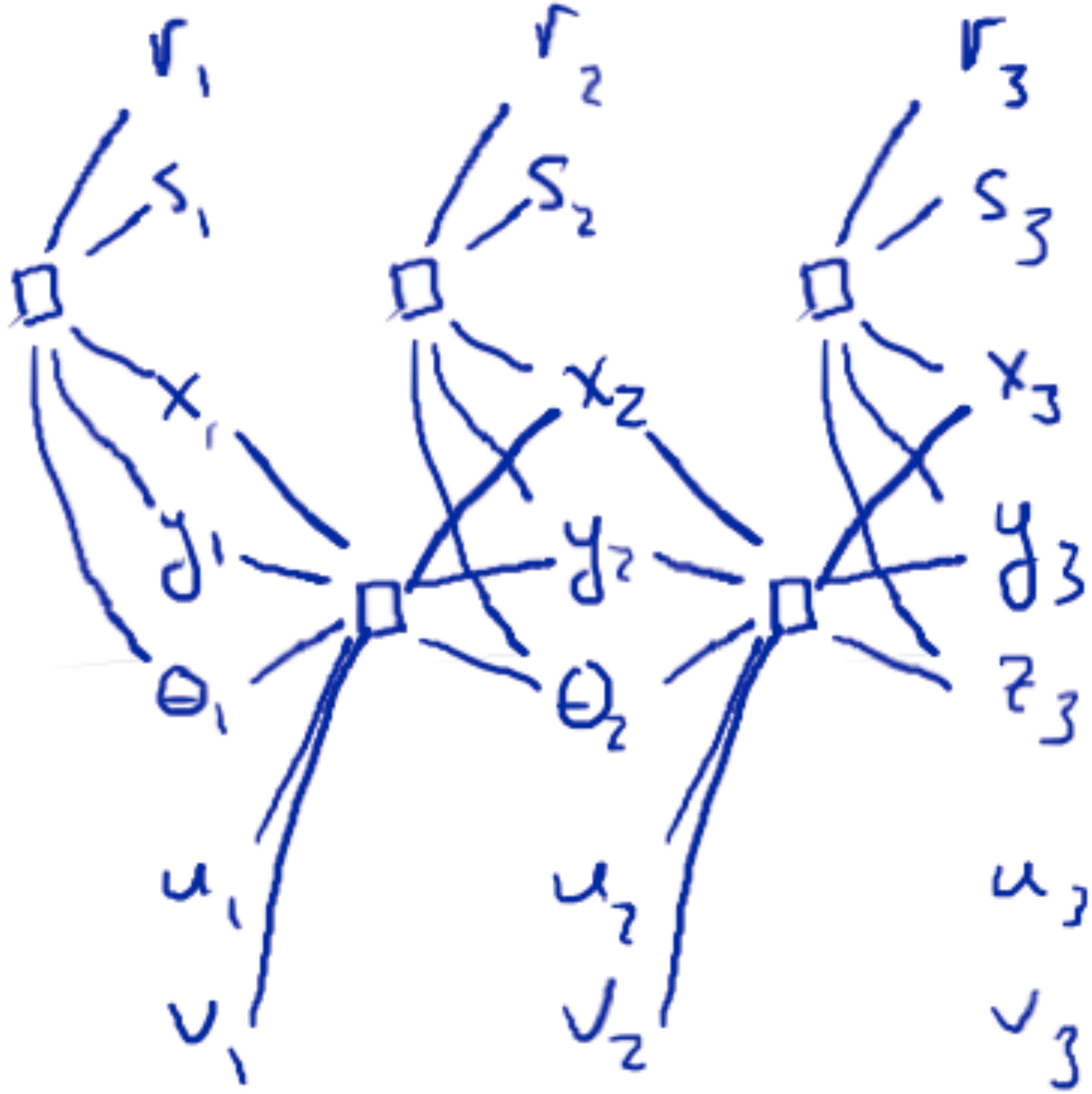


Goal: inference over time



- N=1 sensor, repeatedly observe range = 1m + noise

Factor graph



Dynamic Bayes Network

- DBN: factor graph composed of a single structural unit repeated over time
  - conceptually infinite to right, but in practice cut off at some maximum T
- Factors **must** be conditional distributions

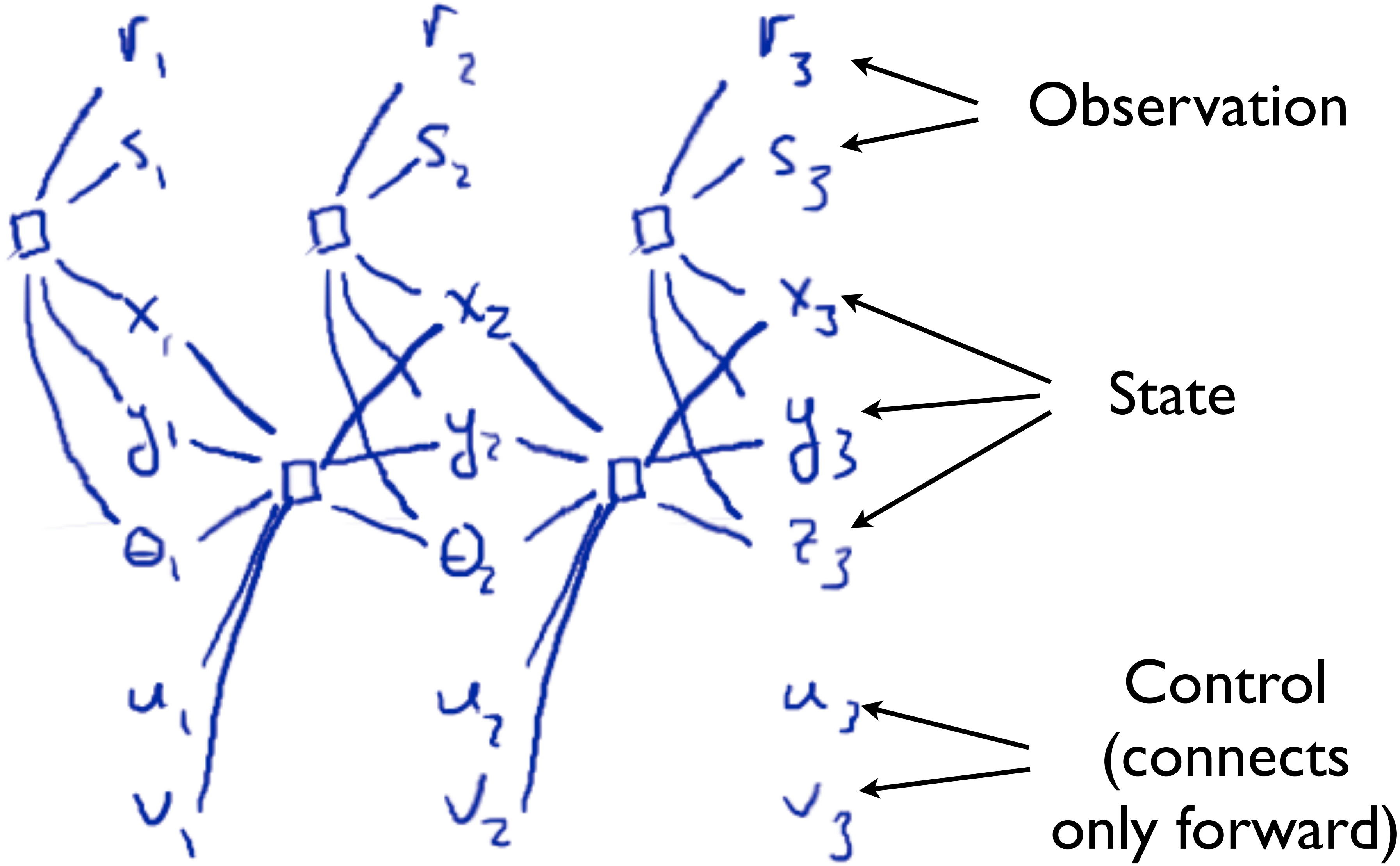
Should be replaced by

```
\begin{array}{rrcl}
\forall x_t, y_t, \theta_t, u_t, v_t& \sum_{x_{t+1},y_{t+1},\theta_{t+1}} \phi(x_t, y_t, \theta_t, u_t, v_t, x_{t+1},y_{t+1},\theta_{t+1}) &=& 1\\[2ex]
\forall x_t, y_t, \theta_t& \sum_{r_{t},s_{t}} \phi(x_t, y_t, \theta_t, r_t, s_t) &=& 1
\end{array}
```

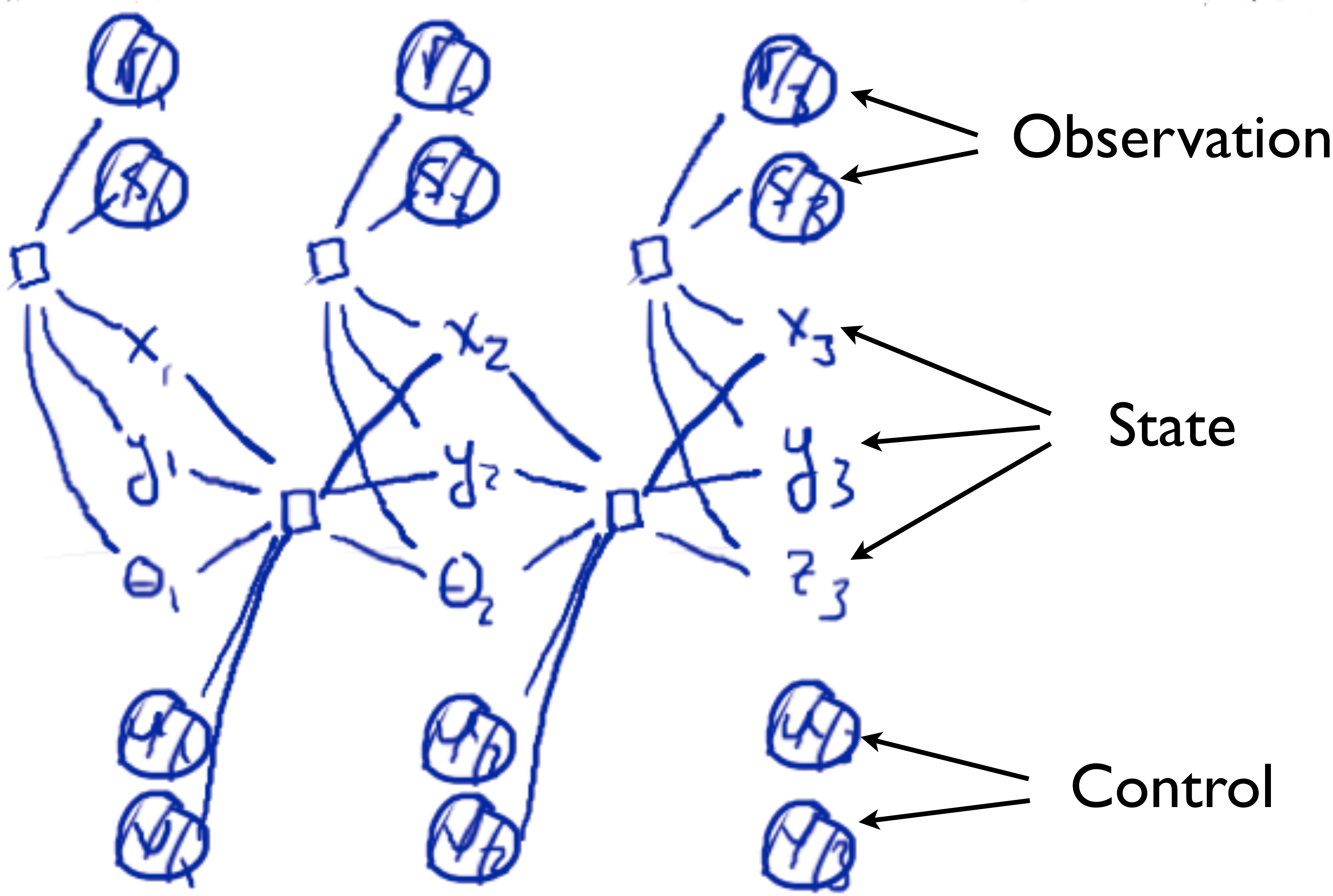
(see unannotated slides for a latex'd version)

y_t

Three kinds of variable

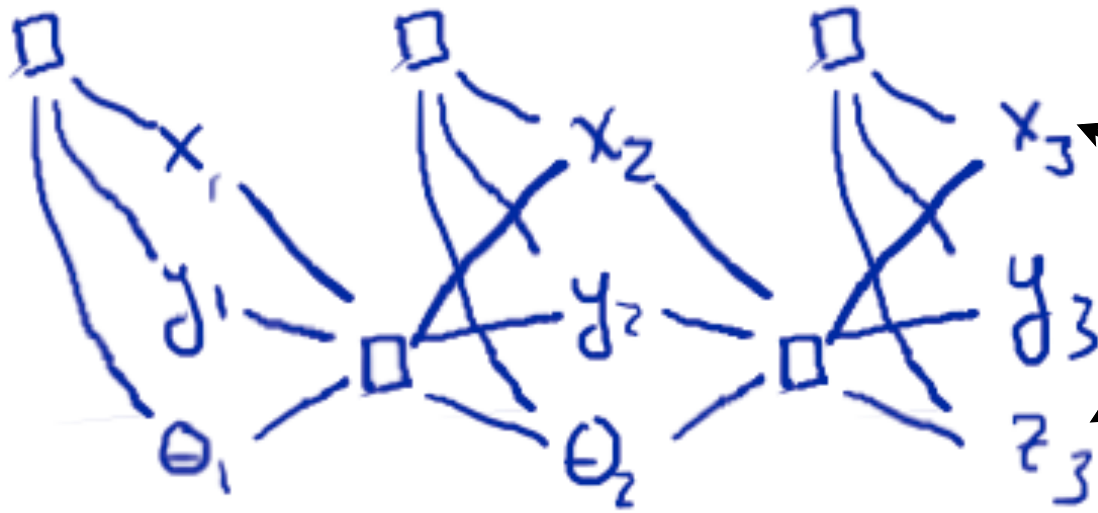


Condition on obs, do(control)



Condition on obs, do(control)

Observation



State

Control

Simplified version

- State: $x_t \in \{1, 2, 3\}$
- Observation: $y_t \in \{L, H\}$
- Control: just one (i.e., no choice)—"keep going"

Hidden Markov Models

- This is an HMM—a DBN with:
    - one state variable
    - one observation variable

Potentials

X_t \ X_{t+1}	1	2	3
1	0.7	0.3	0
2	0.3	0.3	0.3
3	0	0.3	0.7

X_t \ Y_t	L	H
1	0.67	0.33
2	0.5	0.5
3	0.33	0.67

HMM inference

- Condition on $y_1 = H, y_2 = H, y_3 = L$
- What is $P(X_2 \mid HHL)$?

HMM factors after conditioning

x_1	ϕ_1
1	.33
2	.5
3	.67

x_2	ϕ_2
1	.33
2	.5
3	.67

x_3	ϕ_3
1	.67
2	.5
3	.33

ϕ_4

x_1 \ x_2			
	.67	.33	0
	.33	.33	.33
	0	.33	.67

ϕ_5

x_2 \ x_3			
	.67	.33	0
	.33	.33	.33
	0	.33	.67

Eliminate x_1 and x_3

$\phi_1 \phi_4$ α_{12}

$$
\begin{array}{c|ccc}
 & & x_2 & \\
\hline
 & 2/9 & 1/9 & 0 \\
x_1 & 1/6 & 1/6 & 1/6 \\
 & 0 & 2/9 & 4/9
\end{array}
\rightarrow
\begin{array}{c}
x_2 \\
\hline
7/18 \\
1/2 \\
11/18
\end{array}
$$

$\phi_3 \phi_5$ β_{23}

$$
\begin{array}{c|ccc}
 & & x_3 & \\
\hline
 & 4/9 & 1/6 & 0 \\
x_2 & 2/9 & 1/6 & 1/9 \\
 & 0 & 1/6 & 2/9
\end{array}
\rightarrow
\begin{array}{c}
x_2 \\
\hline
11/18 \\
1/2 \\
7/18
\end{array}
$$

Multiply remaining potentials and renormalize

α_{12}

X_2
7/18
1/2
11/18

β_{23}

X_2
1/18
1/2
7/18

$\alpha_{12}\,\beta_{23}\,\phi_2$
.079
.125
.158

/Z

.22

.34

.44

Forward-backward

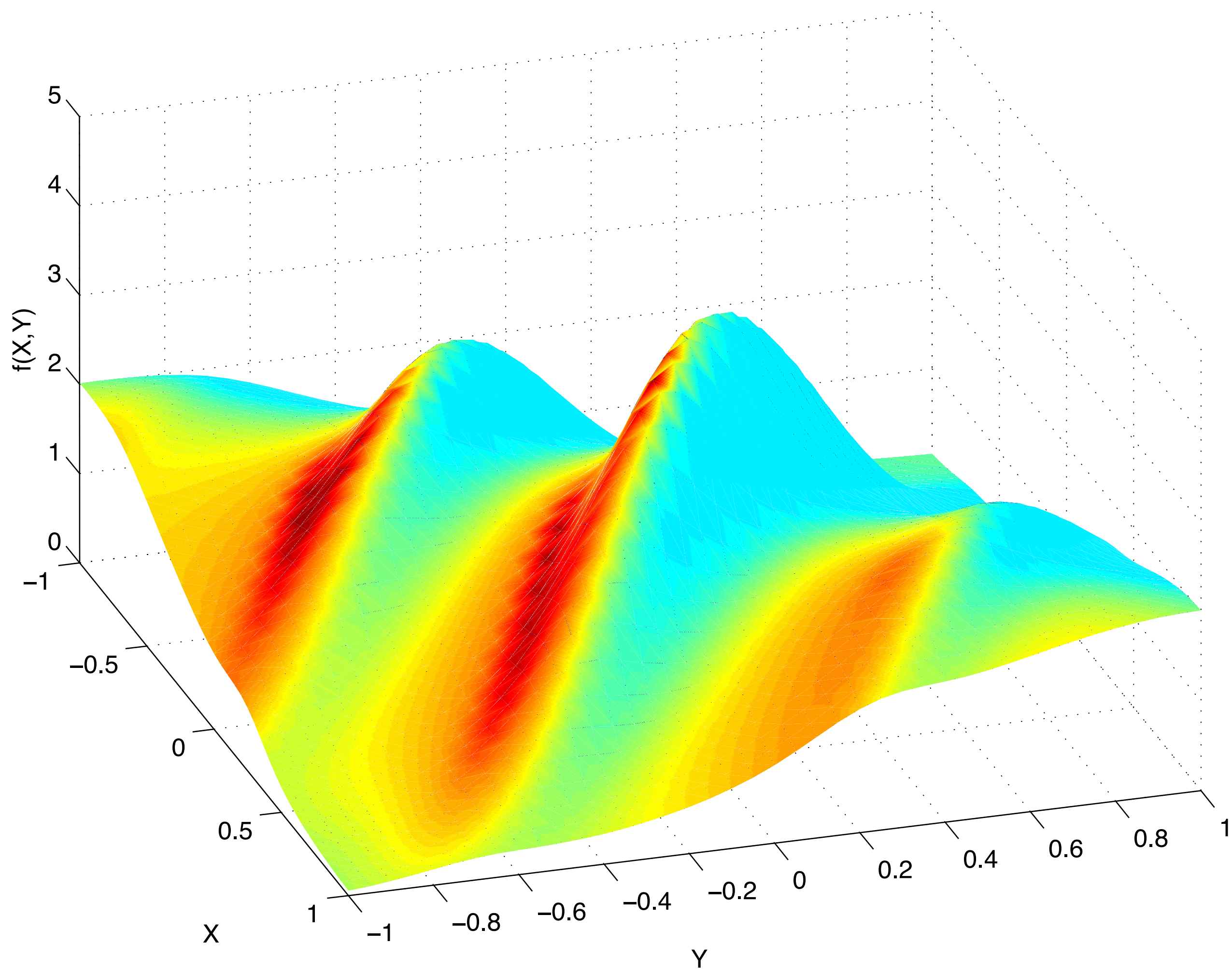
- You may recognize the above as the forward-backward algorithm
- Special case of dynamic programming / variable elimination / belief propagation

Approximate Inference

Most of the time…

- Treewidth is big
- Variables are high-arity or continuous
- Can’t afford exact inference
- Need numerical integration (and/or summation)
- We’ll look at randomized algorithms

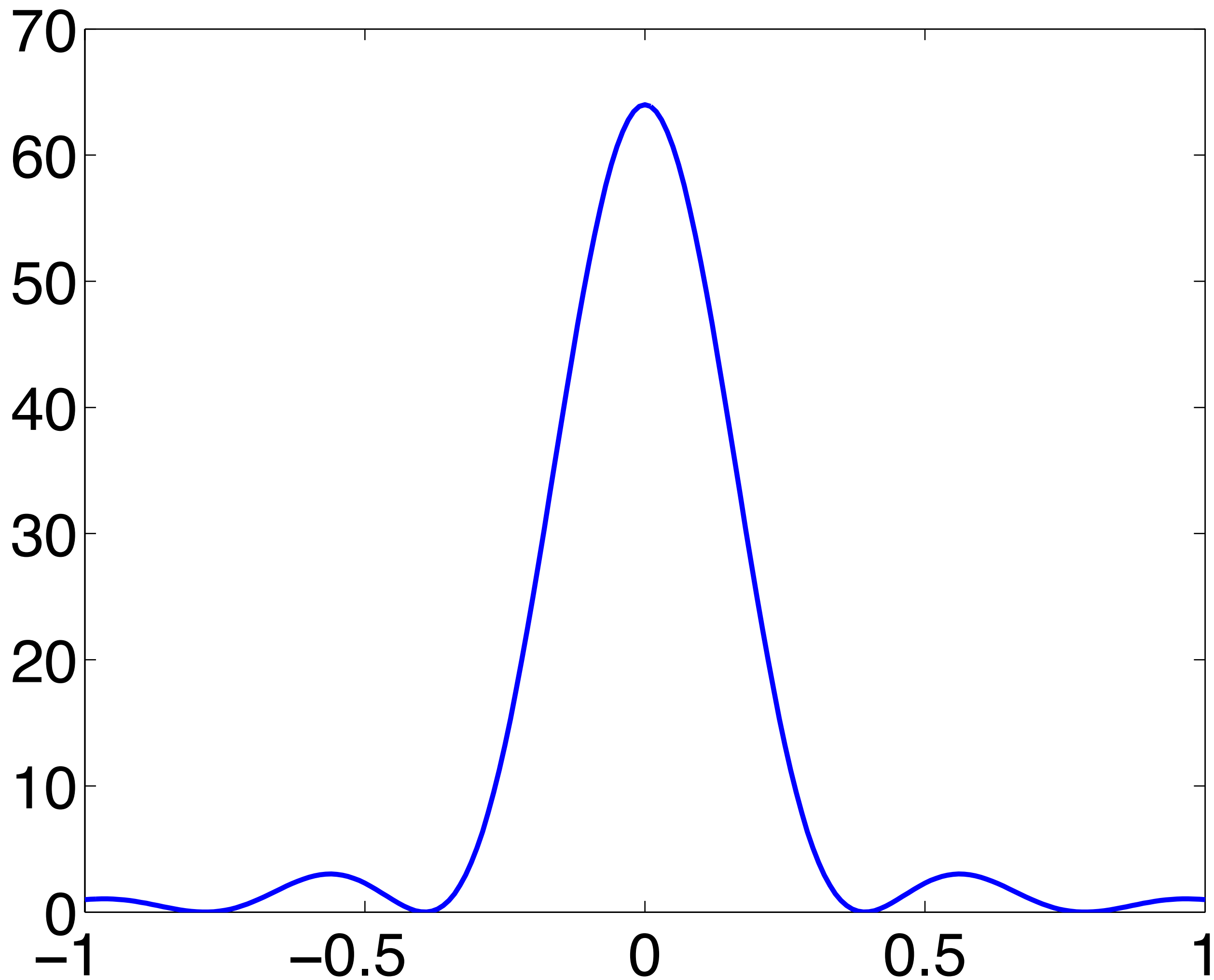
Numerical integration



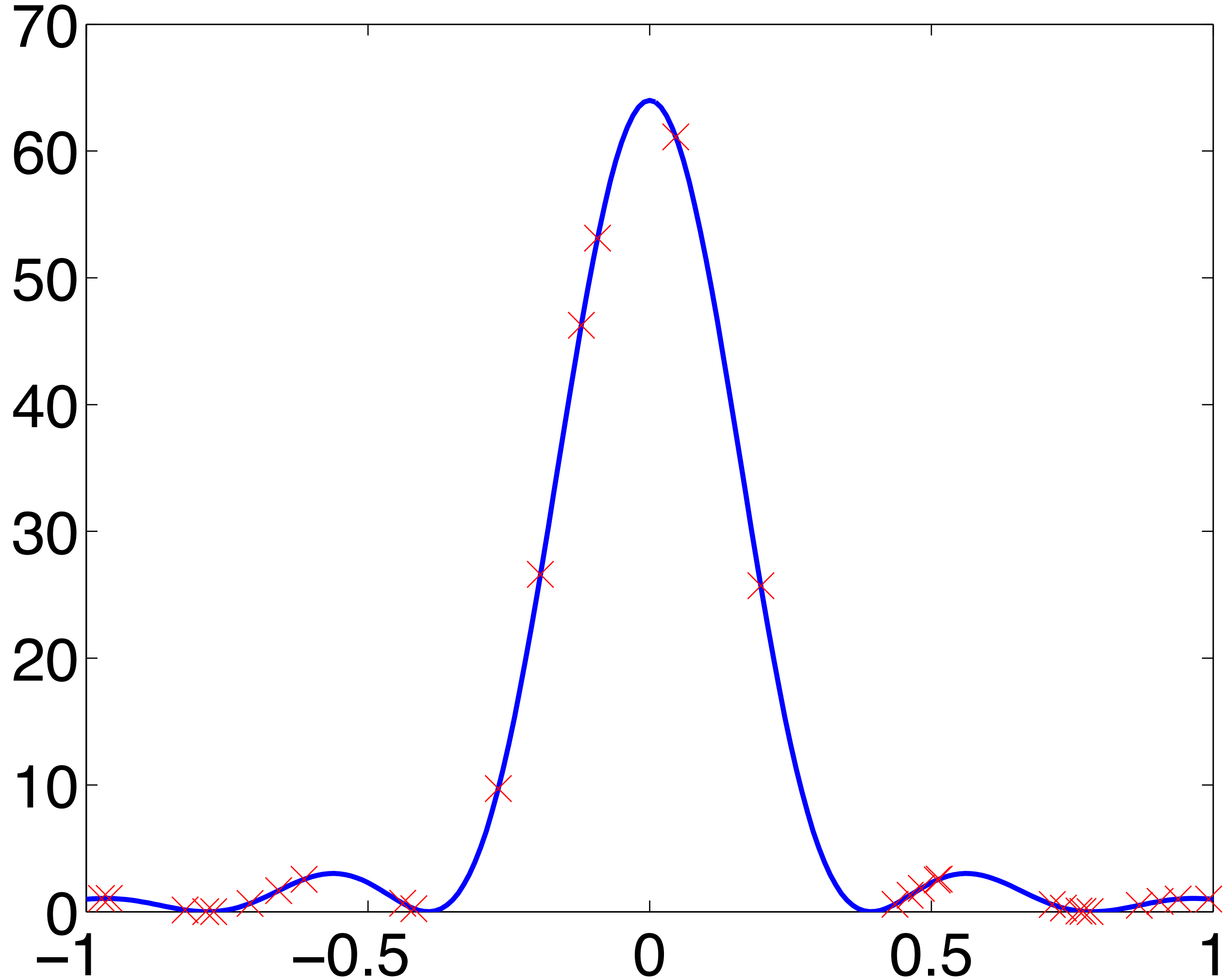
Integration in 1000s of dims

Eliazar and Parr, IJCAI-03

Simple 1D problem



Uniform sampling



$$\frac{2}{N}\sum_i f(x_i)$$

Uniform sampling

$$\begin{aligned} E(f(X)) &= \int P(x)f(x)dx \\ &= \frac{1}{V}\int f(x)dx \end{aligned}$$

- So, V E(f(X)) is desired integral
- But standard deviation can be big
- Can reduce it by averaging many samples
- But only at rate 1/sqrt(N)

Importance sampling

- Instead of x ~ uniform, use x ~ Q(x)
- Q = importance distribution
- Should have Q(x) large where f(x) is large
- Problem:

$$E_Q(f(X)) = \int Q(x)f(x)dx$$

Importance sampling

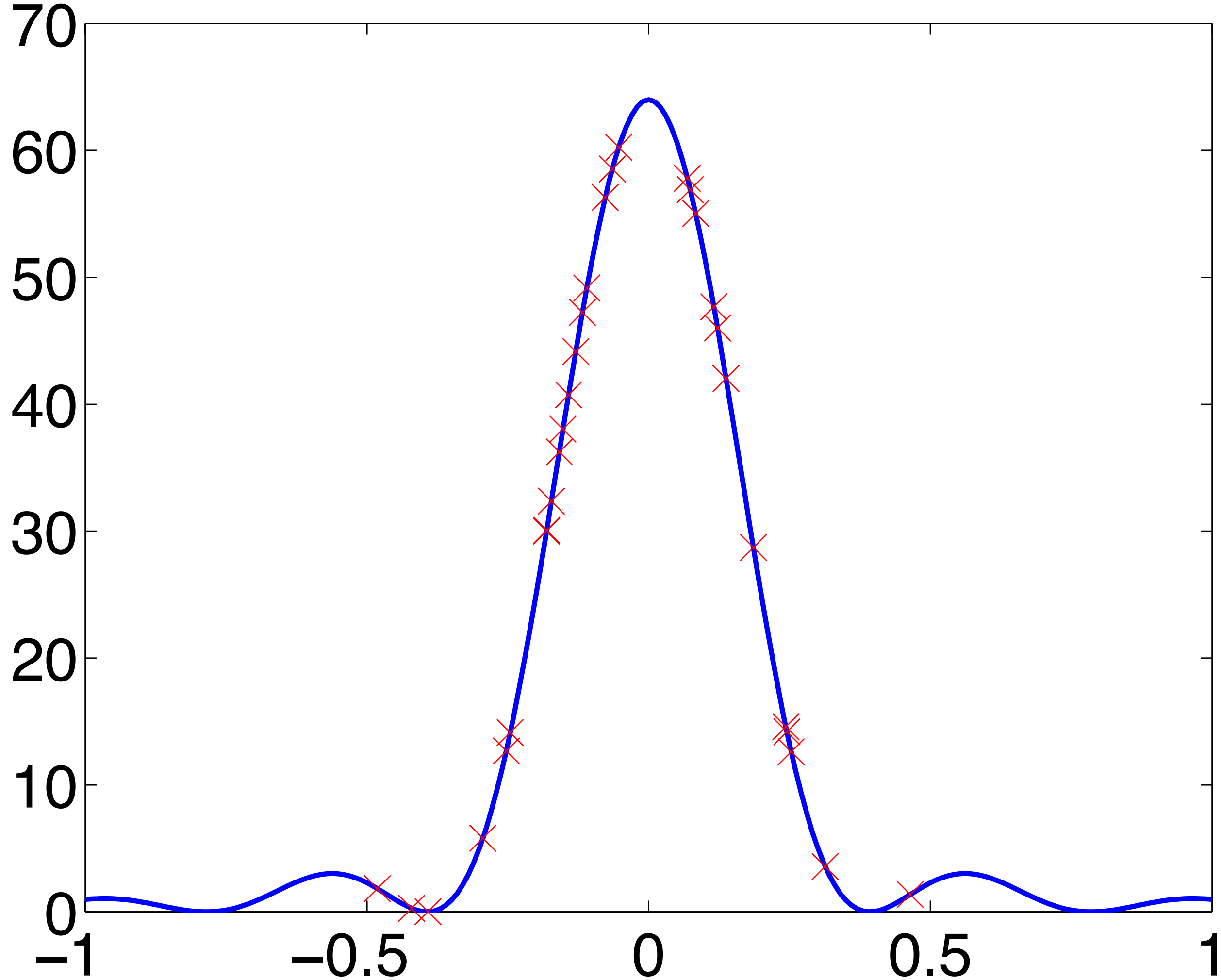
$$h(x) \equiv f(x)/Q(x)$$

$$\begin{aligned} E_Q(h(X)) &= \int Q(x)h(x)dx \\ &= \int Q(x)f(x)/Q(x)dx \\ &= \int f(x)dx \end{aligned}$$

Importance sampling

- So, take samples of $h(X)$ instead of $f(X)$
- $w_i = 1/Q(x_i)$ is **importance weight**
- $Q = 1/V$ yields uniform sampling

Importance sampling



Variance

- How does this help us control variance?
- Suppose f big ==> Q big
- And Q small ==> f small
- Then $h = f/Q$ never gets too big
- Variance of each sample is lower ==> need fewer samples
- A good Q makes a good IS

Importance sampling, part II

- Suppose

$$
\begin{aligned}
f(x) &= R(x)g(x) \\
\int f(x)dx &= \int R(x)g(x)dx \\
&= \mathbb{E}_R[g(x)]
\end{aligned}
$$

Importance sampling, part II

- Use importance sampling w/ proposal $Q(X)$:
  - Pick N samples x_i from $Q(X)$
  - Average $w_i\, g(x_i)$, where $w_i = R(x_i)/Q(x_i)$ is importance weight

$$
\begin{aligned}
\mathbb{E}_Q(W g(X)) &= \int Q(x) \frac{R(x)}{Q(x)} g(x) \\
&= \int R(x) g(x) dx \\
&= \int f(x) dx
\end{aligned}
$$

Parallel IS

- Now suppose $R(x)$ is unnormalized (e.g., represented by factor graph)—know only $Z\,R(x)$
- Pick N samples x_i from proposal $Q(X)$
- If we knew $w_i = R(x_i)/Q(x_i)$, could do IS
- Instead, set

$$\hat{w}_i = ZR(x_i)/Q(x_i)$$

Parallel IS

$$\begin{aligned}\mathbb{E}(\hat{W}) &= \int Q(x)\frac{ZR(x)}{Q(x)}dx \\ &= \int ZR(x)dx \\ &= Z\end{aligned}$$

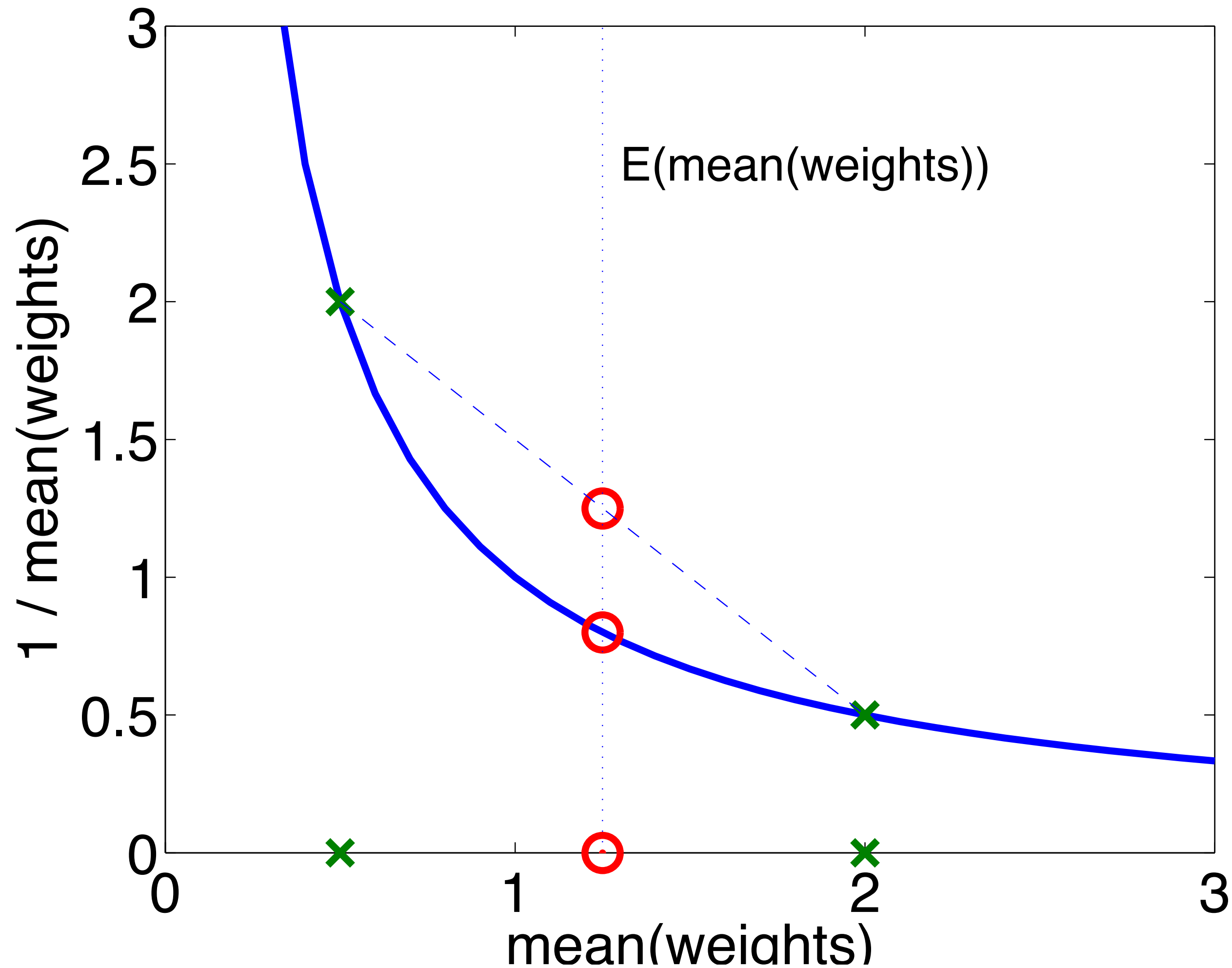
- So, $\bar{w} = \frac{1}{N}\sum_i \hat{w}_i$ is an unbiased estimate of Z

Parallel IS

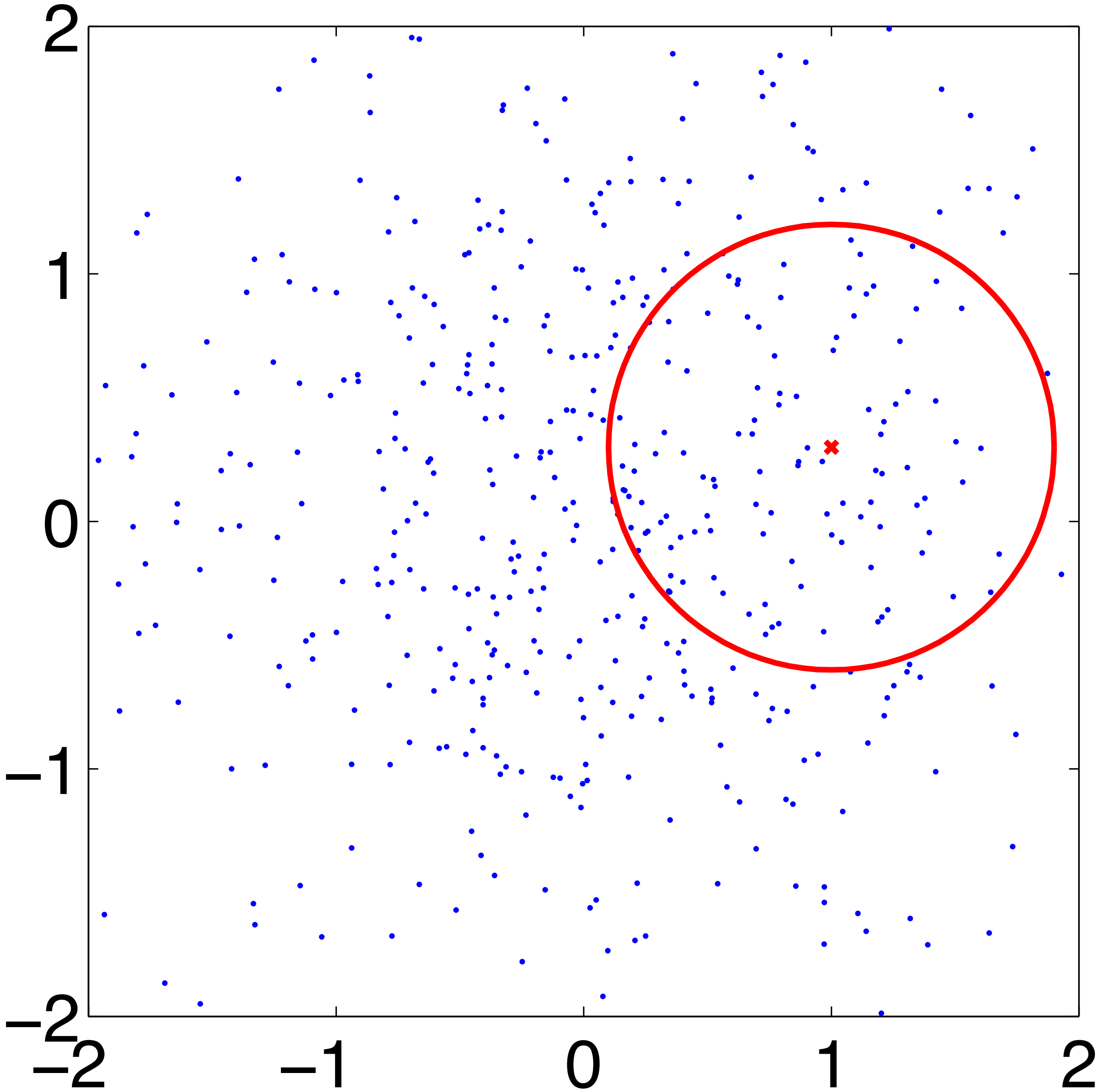
- So, $\hat{w}_i / \bar{w}$ is an estimate of w_i, computed without knowing Z
- Final estimate:

$$\int f(x)dx \approx \frac{1}{n} \sum_i \frac{\hat{w}_i}{\bar{w}} g(x_i)$$

Parallel IS is biased

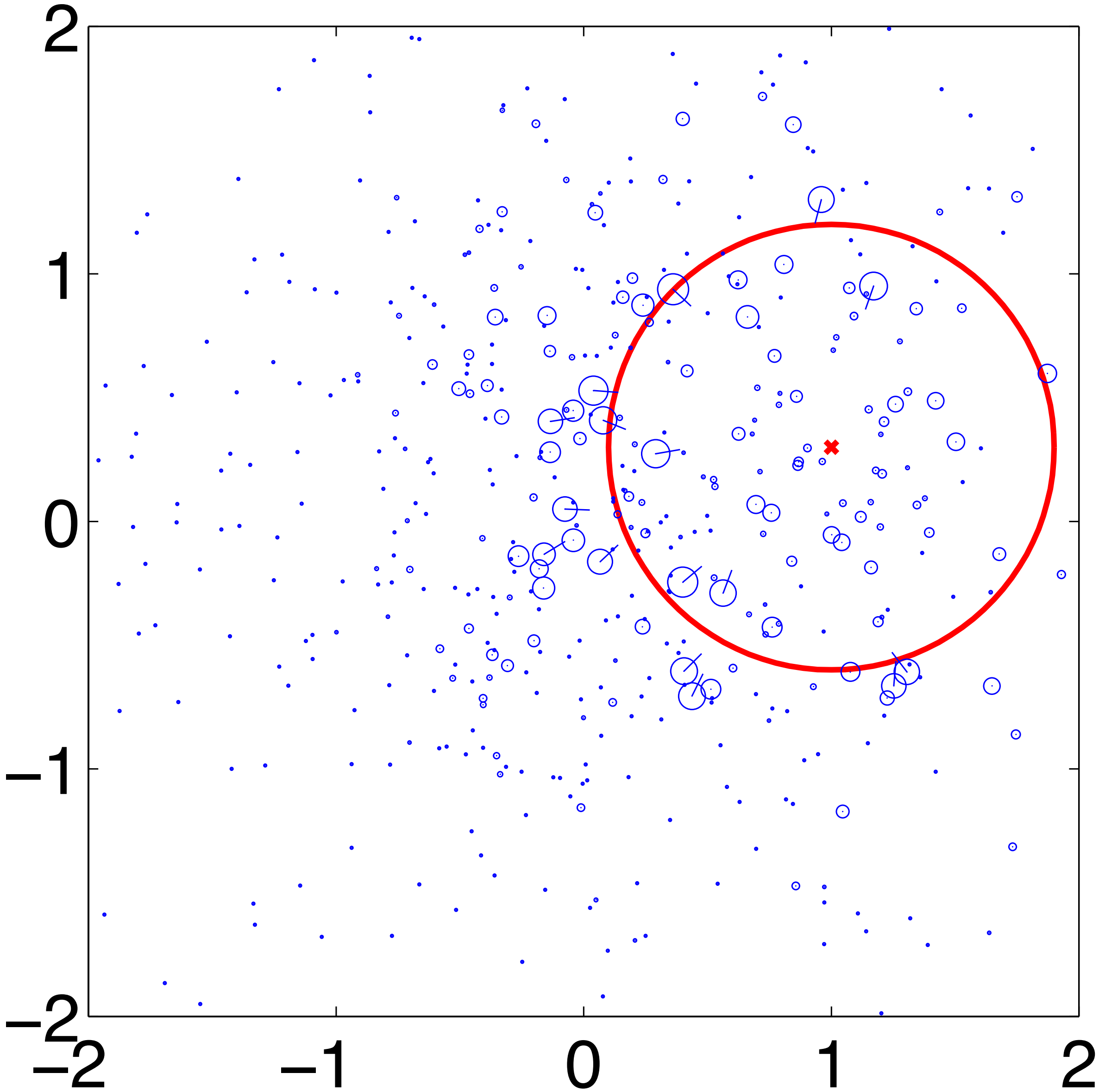


$$E(\bar{W}) = Z, \text{ but } E(1/\bar{W}) \neq 1/Z \text{ in general}$$



$$Q : (X, Y) \sim N(1, 1) \qquad \theta \sim U(-\pi, \pi)$$

$$f(x, y, \theta) = Q(x, y, \theta) P(o = 0.8 \mid x, y, \theta)/Z$$



Posterior $E(X, Y, \theta) = (0.496, 0.350, 0.084)$

MCMC

Integration problem

- Recall: wanted

$$\int f(x)dx = \int R(x)g(x)dx$$

- And therefore, wanted good importance distribution Q(x) (close to R)

Back to high dimensions

- Picking a good importance distribution is hard in high-D
- Major contributions to integral can be hidden in small areas
  - recall, want (R big ==> Q big)
- Would like to search for areas of high R(x)
- But searching could bias our estimates

Markov-Chain Monte Carlo

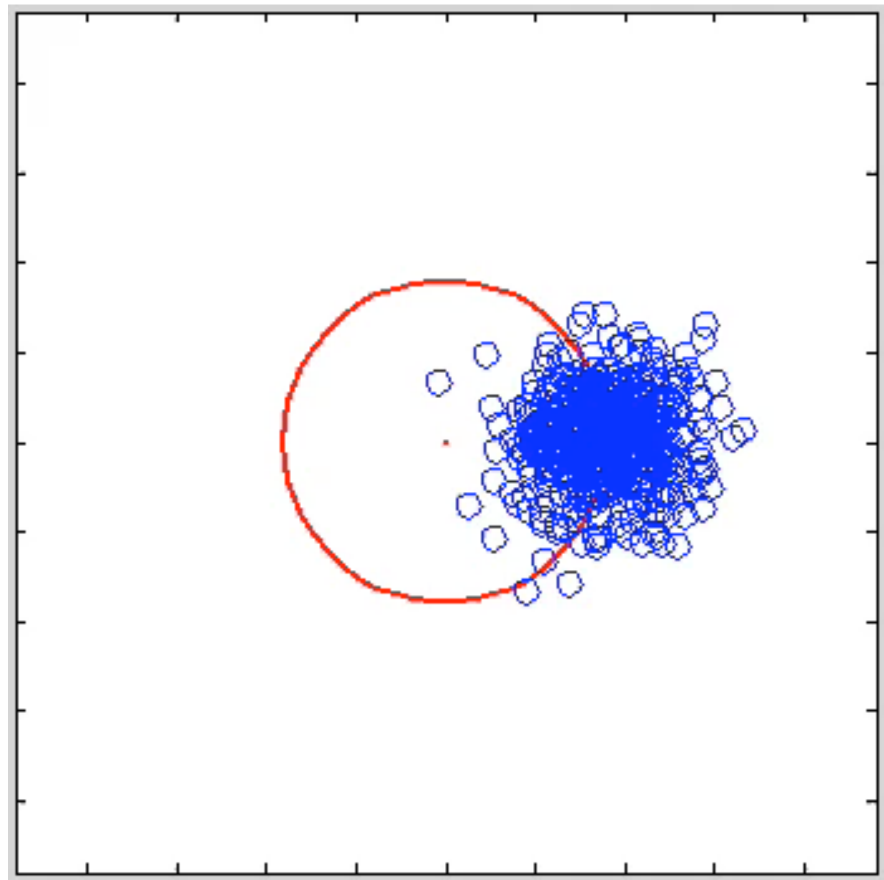
- Design a randomized search procedure M over values of x, which tends to increase R(x) if it is small
- Run M for a while, take resulting x as a sample
- Importance distribution Q(x)?

Markov-Chain Monte Carlo

- Design a randomized search procedure M over values of x, which tends to increase R(x) if it is small
- Run M for a while, take resulting x as a sample
- Importance distribution Q(x)?
  - Q = stationary distribution of M…

Stationary distribution

- Run HMM or DBN for a long time; stop at a random point
- Do this again and again
- Resulting samples are from stationary distribution



Designing a search chain

$$\int f(x)dx \quad = \quad \int R(x)g(x)dx$$

- Would like Q(x) = R(x)
  - makes importance weight = 1
- Turns out we can get this exactly, using **Metropolis-Hastings**

Metropolis-Hastings

- Way of designing chain w/ $Q(x) = R(x)$
- Basic strategy: start from arbitrary x
- Repeatedly tweak x to get x'
- If $R(x') \geq R(x)$, move to x'
- If $R(x') << R(x)$, stay at x
- In intermediate cases, randomize

Proposal distribution

- Left open: what does "tweak" mean?
- Parameter of MH: $Q(x' \mid x)$
  - one-step proposal distribution
- Good proposals explore quickly, but remain in regions of high $R(x)$
- Optimal proposal?

MH algorithm

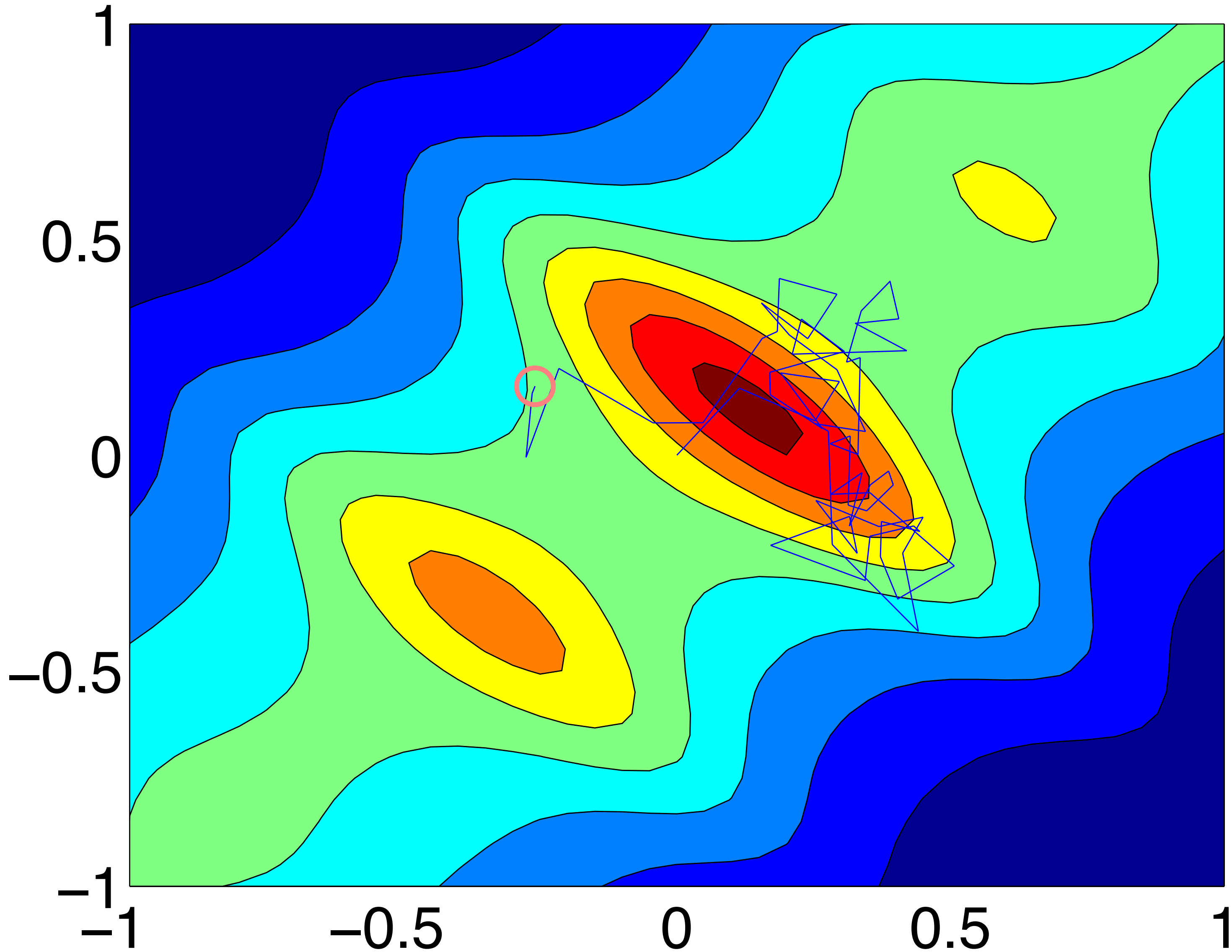
- Sample x' ~ Q(x' | x)
- Compute p = $$\frac{R(x')}{R(x)} \frac{Q(x' \mid x)}{Q(x \mid x')}$$
- With probability min(1, p), set x := x'
- Repeat for T steps; sample is $x_1, \ldots, x_T$ (will usually contain duplicates)

MH algorithm

note: we don't need to know Z

- Sample x' ~ Q(x' | x)
- Compute p = $\dfrac{R(x')}{R(x)} \dfrac{Q(x' \mid x)}{Q(x \mid x')}$
- With probability min(1, p), set x := x'
- Repeat for T steps; sample is $x_1, \ldots, x_T$ (will usually contain duplicates)

MH example



Acceptance rate

- Moving to new x’ is **accepting**
- Want **acceptance rate** (avg p) to be large, so we don’t get big runs of the same x
- Want Q(x’ | x) to move long distances (to explore quickly)
- Tension between Q and P(accept):

$$p = \frac{R(x')}{R(x)} \frac{Q(x' \mid x)}{Q(x \mid x')}$$

Mixing rate, mixing time

- If we pick a good proposal, we will move rapidly around domain of R(x)
- After a short time, won't be able to tell where we started
- This is short **mixing time** = # steps until we can't tell which starting point we used
- **Mixing rate** = 1 / (mixing time)

MH estimate

- Once we have our samples $x_1, x_2, \ldots$
- Optional: discard initial "burn-in" range
  - allows time to reach stationary dist'n
- Estimated integral:

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

In example

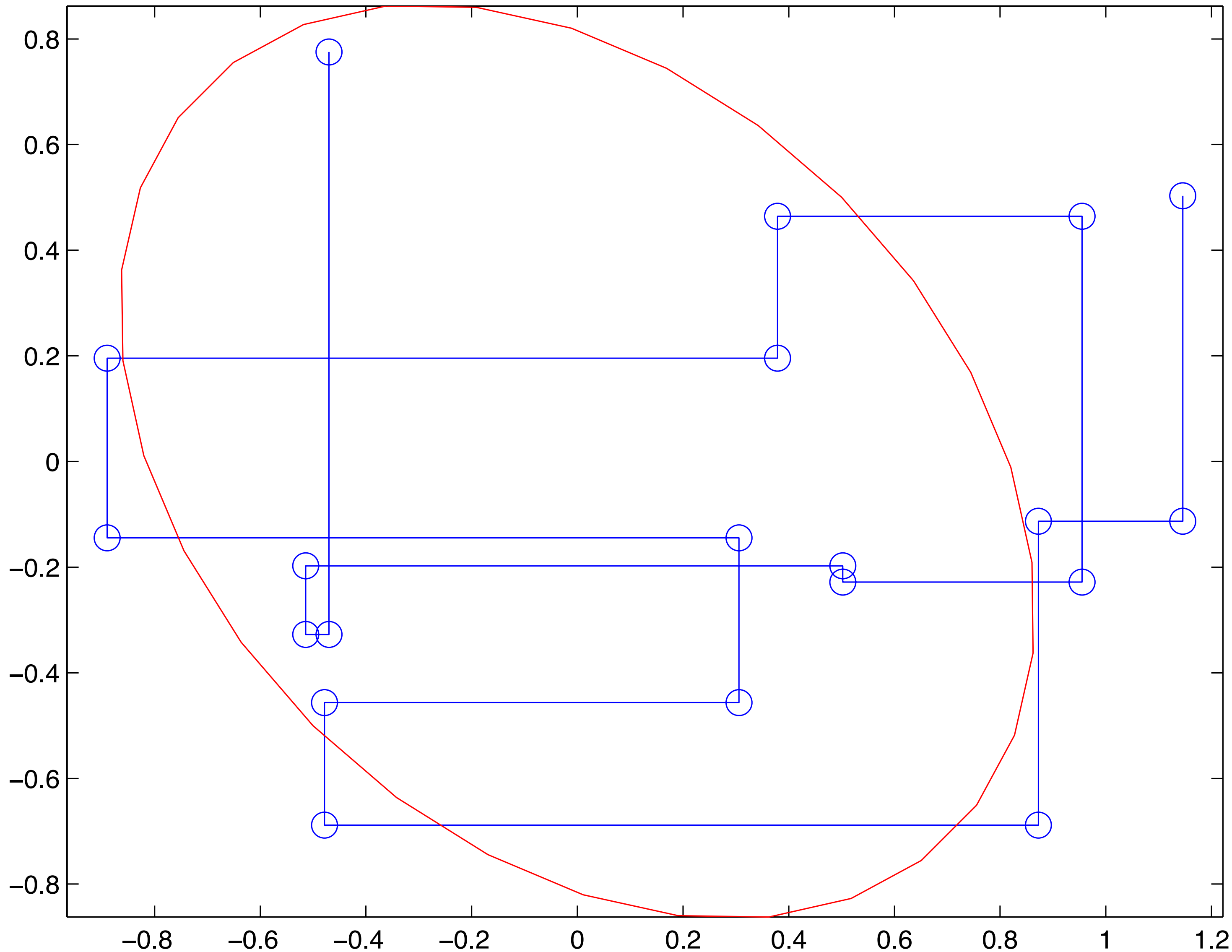
- $g(x) = x^2$
- True $E(g(X)) = 0.28\ldots$
- Proposal: $Q(x' \mid x) = N(x' \mid x, 0.25^2 I)$
- Acceptance rate 55–60%
- After 1000 samples, minus burn-in of 100:

```
final estimate 0.282361
final estimate 0.271167
final estimate 0.322270
final estimate 0.306541
final estimate 0.308716
```

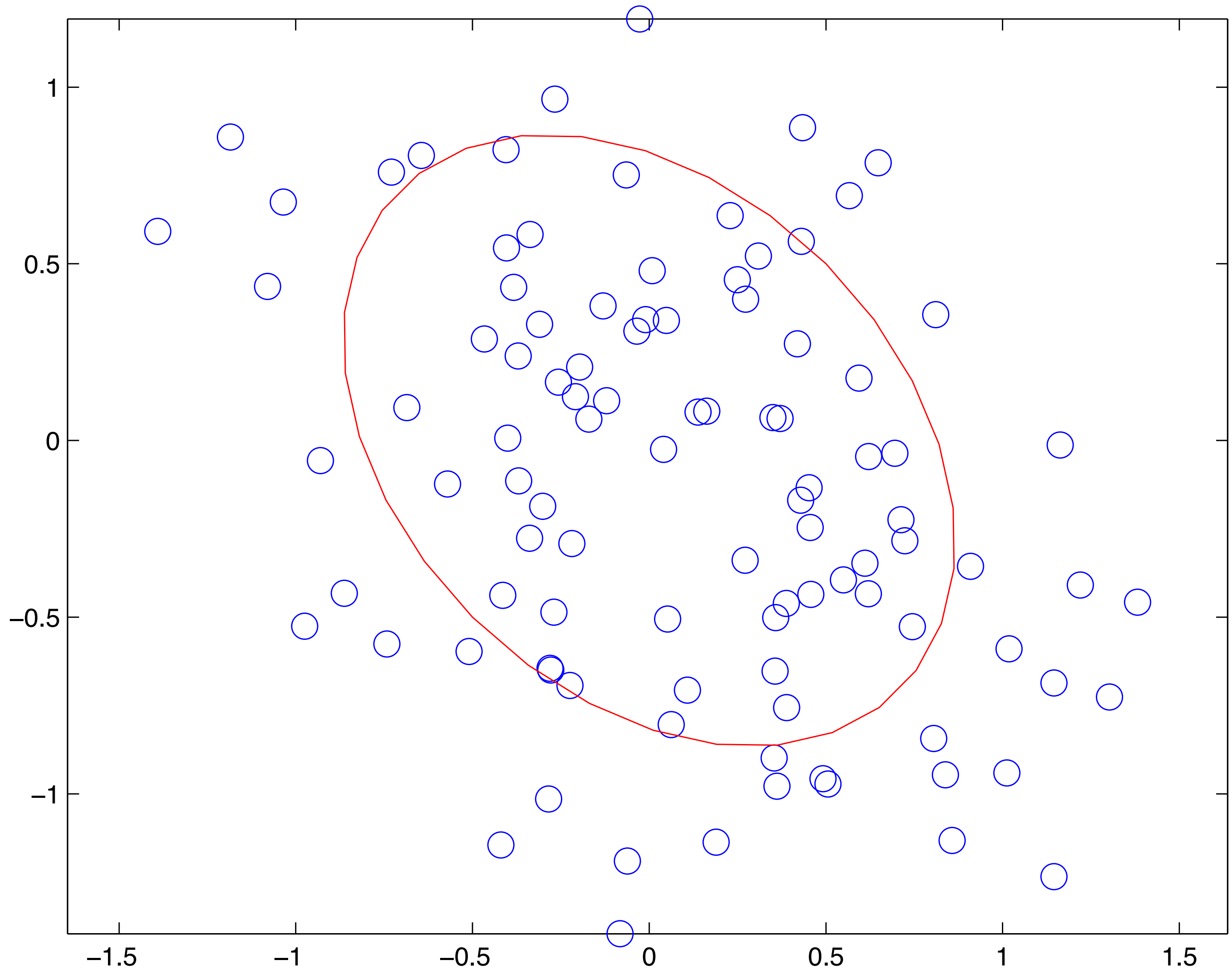
Gibbs sampler

- Special case of MH
- Divide $\mathbf{x}$ into blocks of r.v.s B(1), B(2), …
- Proposal Q:
  - pick a block i uniformly (or round robin, or any other schedule)
  - sample $\mathbf{x}_{B(i)} \sim P(\mathbf{x}_{B(i)} \mid \mathbf{x}_{\neg B(i)})$

Gibbs example



Gibbs example



Why is Gibbs useful?

- For Gibbs, p = $$\frac{P(x'_i, x'_{\neg i})}{P(x_i, x_{\neg i})} \frac{P(x_i \mid x'_{\neg i})}{P(x'_i \mid x_{\neg i})}$$

Gibbs derivation

$$
\begin{aligned}
& \frac{P(x'_i, x'_{\neg i})}{P(x_i, x_{\neg i})} \frac{P(x_i \mid x'_{\neg i})}{P(x'_i \mid x_{\neg i})} \\
= \quad & \frac{P(x'_i, x_{\neg i})}{P(x_i, x_{\neg i})} \frac{P(x_i \mid x_{\neg i})}{P(x'_i \mid x_{\neg i})} \\
= \quad & \frac{P(x'_i, x_{\neg i})}{P(x_i, x_{\neg i})} \frac{P(x_i, x_{\neg i})/P(x_{\neg i})}{P(x'_i, x_{\neg i})/P(x_{\neg i})} \\
= \quad & 1
\end{aligned}
$$