15-780: Grad AI
Lecture 20: Monte Carlo methods, Bayesian learning

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

Admin

PDFs of Tuomas's slides will be on web

- Reminder: midterm March 29
  - Tuomas's review session tomorrow, mine yesterday
    1-2PM GHC 9115
- Reminder: project milestone reports due March 31

Review: factor graphs

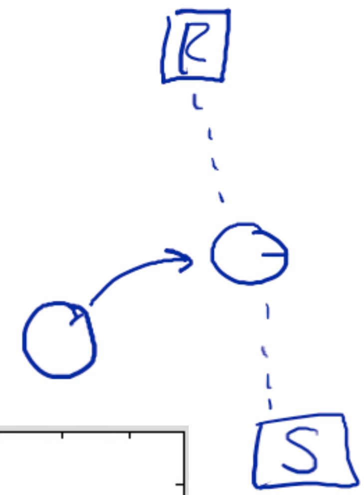
- Undirected, bipartite graph
  - one set of nodes represents variables
  - other set represents factors in probability distribution—tables of nonnegative numbers
  - need to compute normalizer in order to do anything useful
- Can convert back and forth to Bayes nets
- Hard v. soft constraints

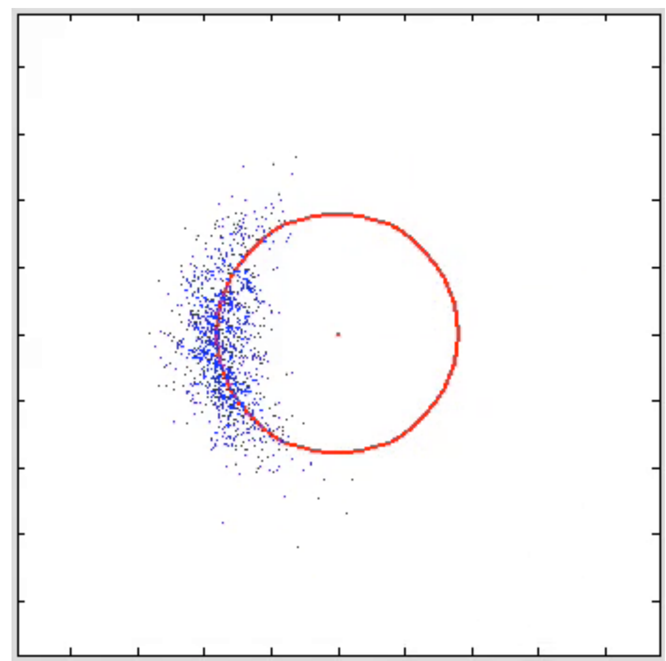
Review: factor graphs

- Graphical test for independence
  - different results from Bayes net, even if we are representing the same distribution
- Inference by dynamic programming
  - instantiate evidence, eliminate nuisance nodes, normalize, answer query
  - elimination order matters
  - treewidth
- Relation to logic

Review: HMMs, DBNs

- Inference over time
  - same graphical template repeated once for each time step—conceptually infinite
- Inference: forward-backward algorithm (special case of belief propagation)

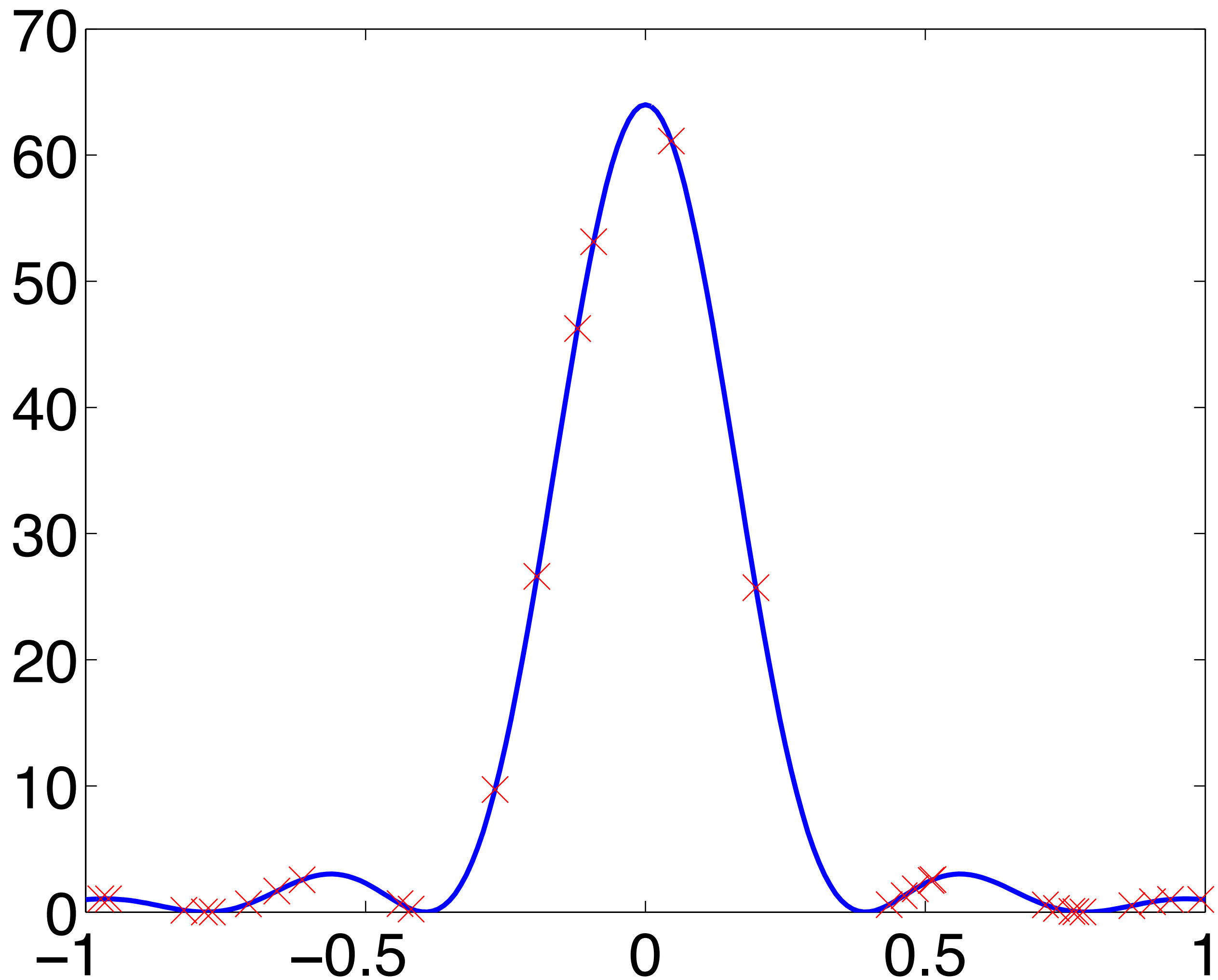




Review: numerical integration

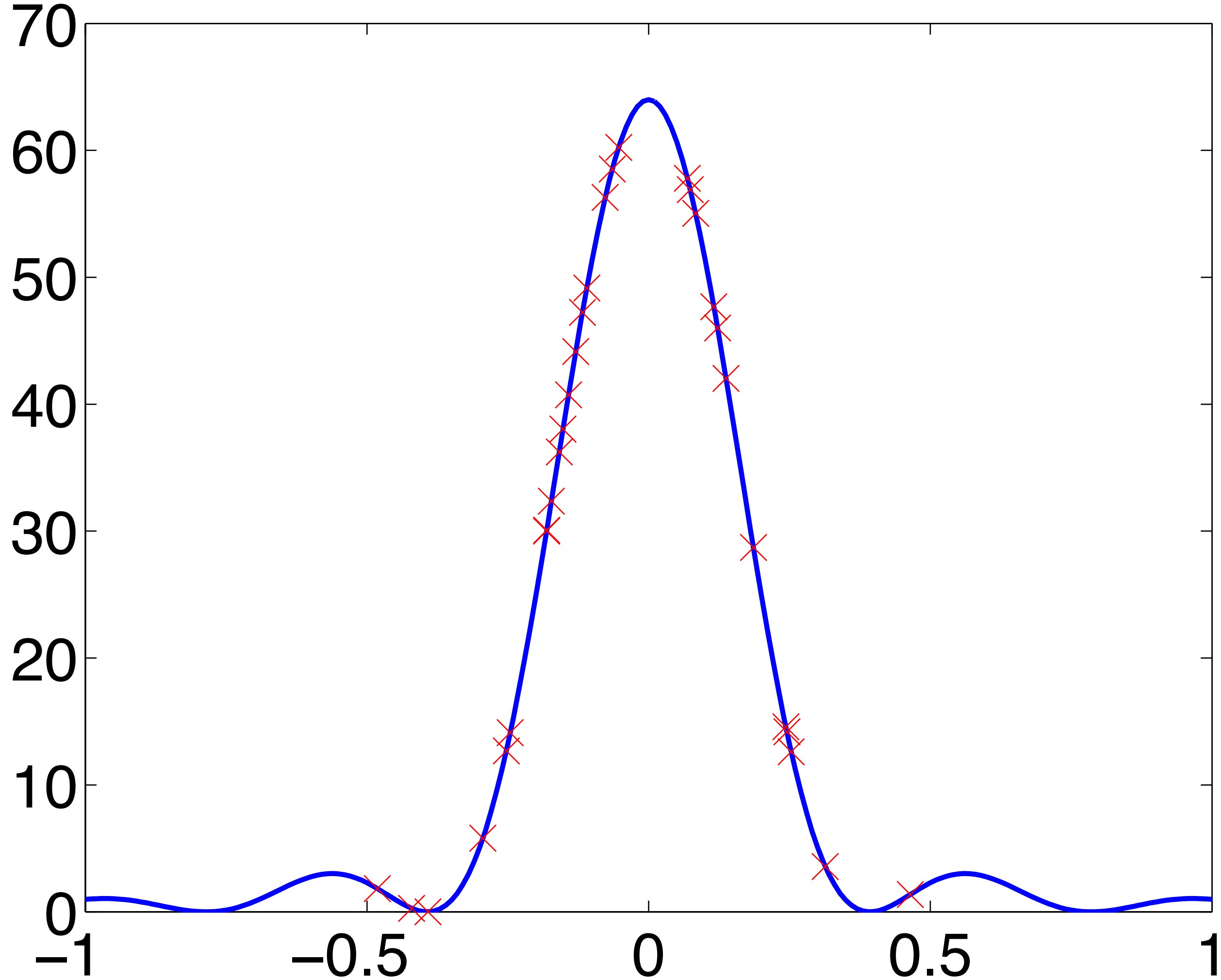
- Integrate a difficult function over a high-dimensional volume
  - narrow, tall peaks contribute most of the integral—difficult search problem
- Central problem for approximate inference
  - e.g., computing normalizing constant in a factor graph

Uniform sampling



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

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

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

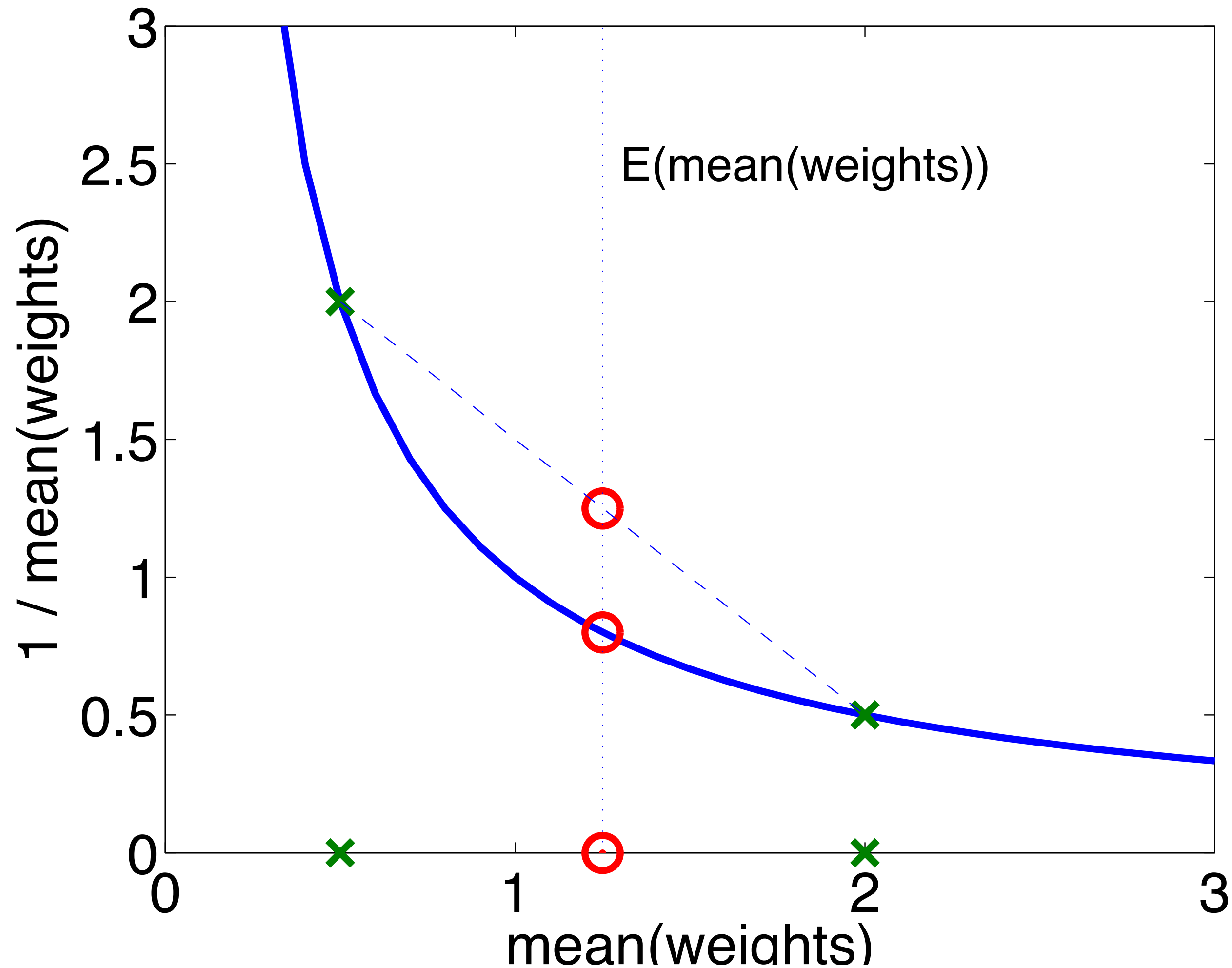
- So, $\bar{w} = \frac{1}{N}\sum_i \hat{w}_i$ is an unbiased estimate of Z

Parallel IS

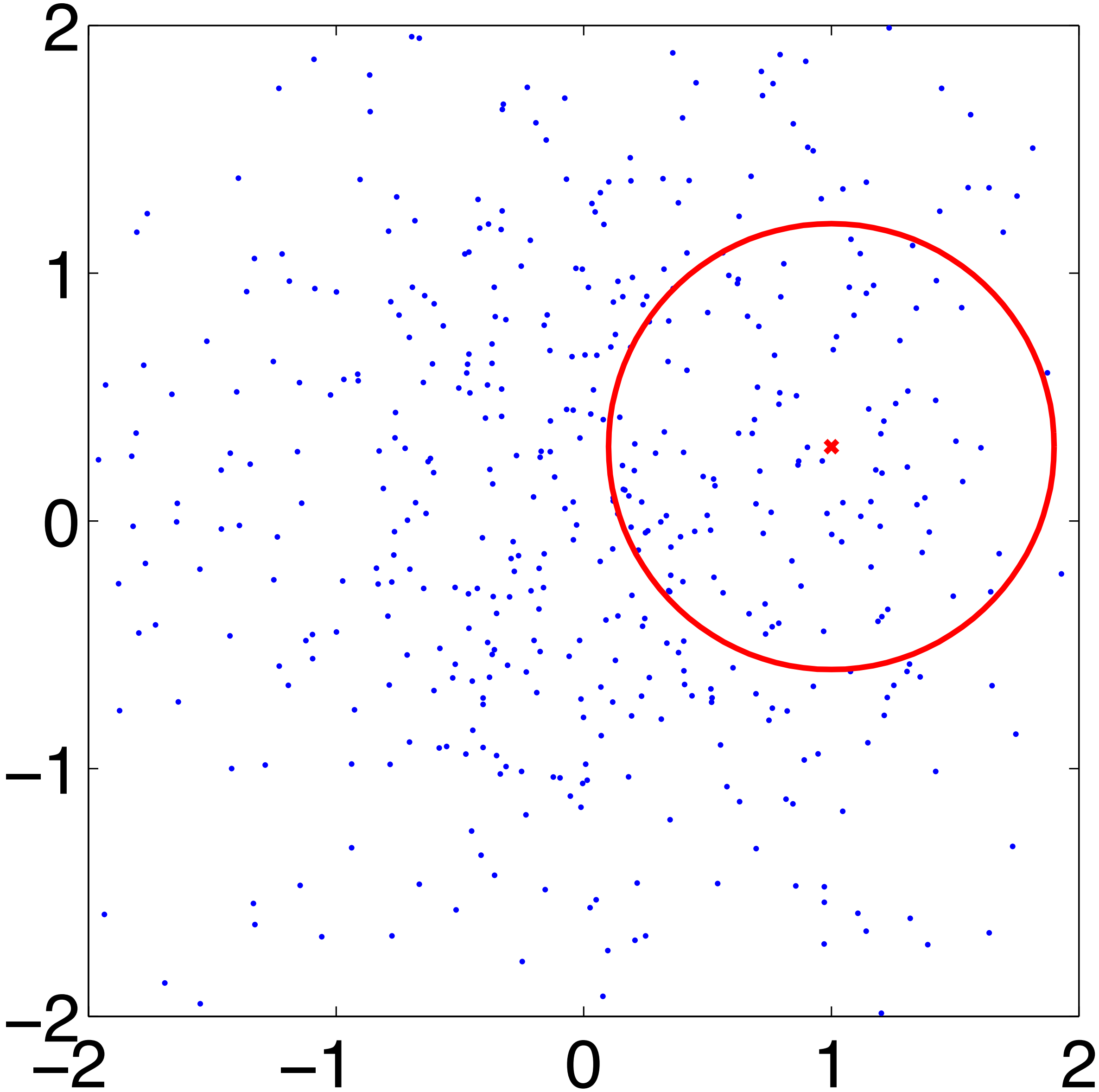
- So, $\hat{w}_i / \bar{w}$ is an estimate of w_i, computed without knowing Z
- Final estimate:

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

Parallel IS is biased

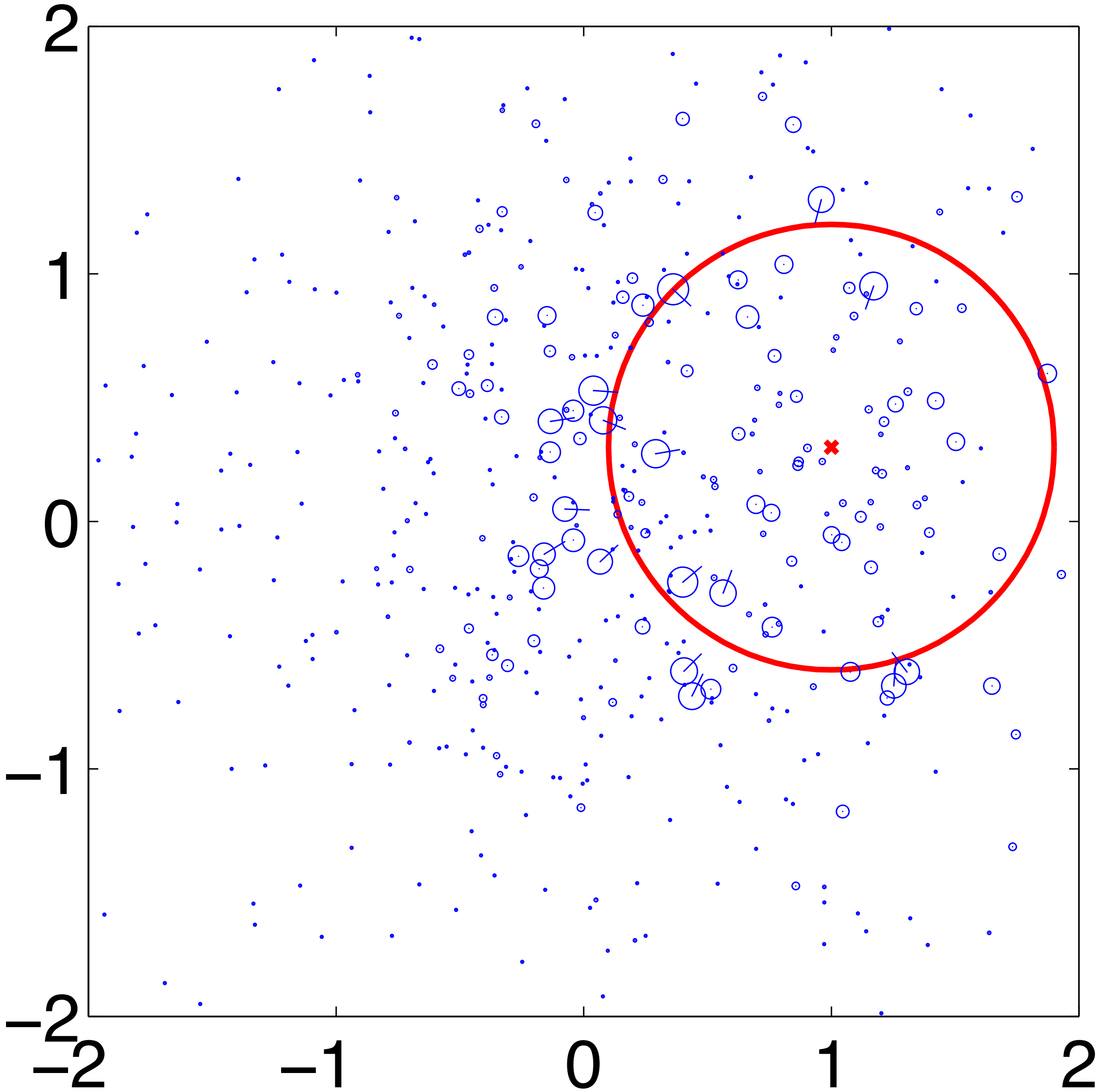


$$E(\bar{W}) = Z, \text{ but } E(1/\bar{W}) \neq 1/Z \text{ in general}$$



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

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



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

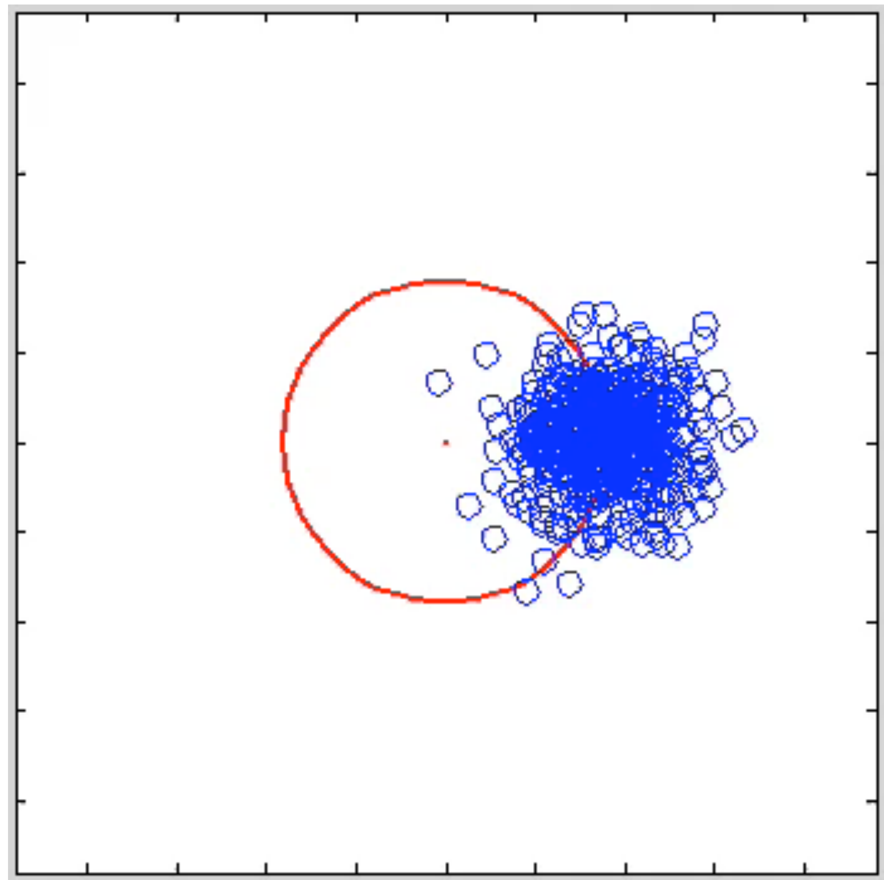
- Design a randomized search procedure M over values of x, which tends to increase R(x) if it is small
- Run M for a while, take resulting x as a sample
- Importance distribution Q(x)?

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

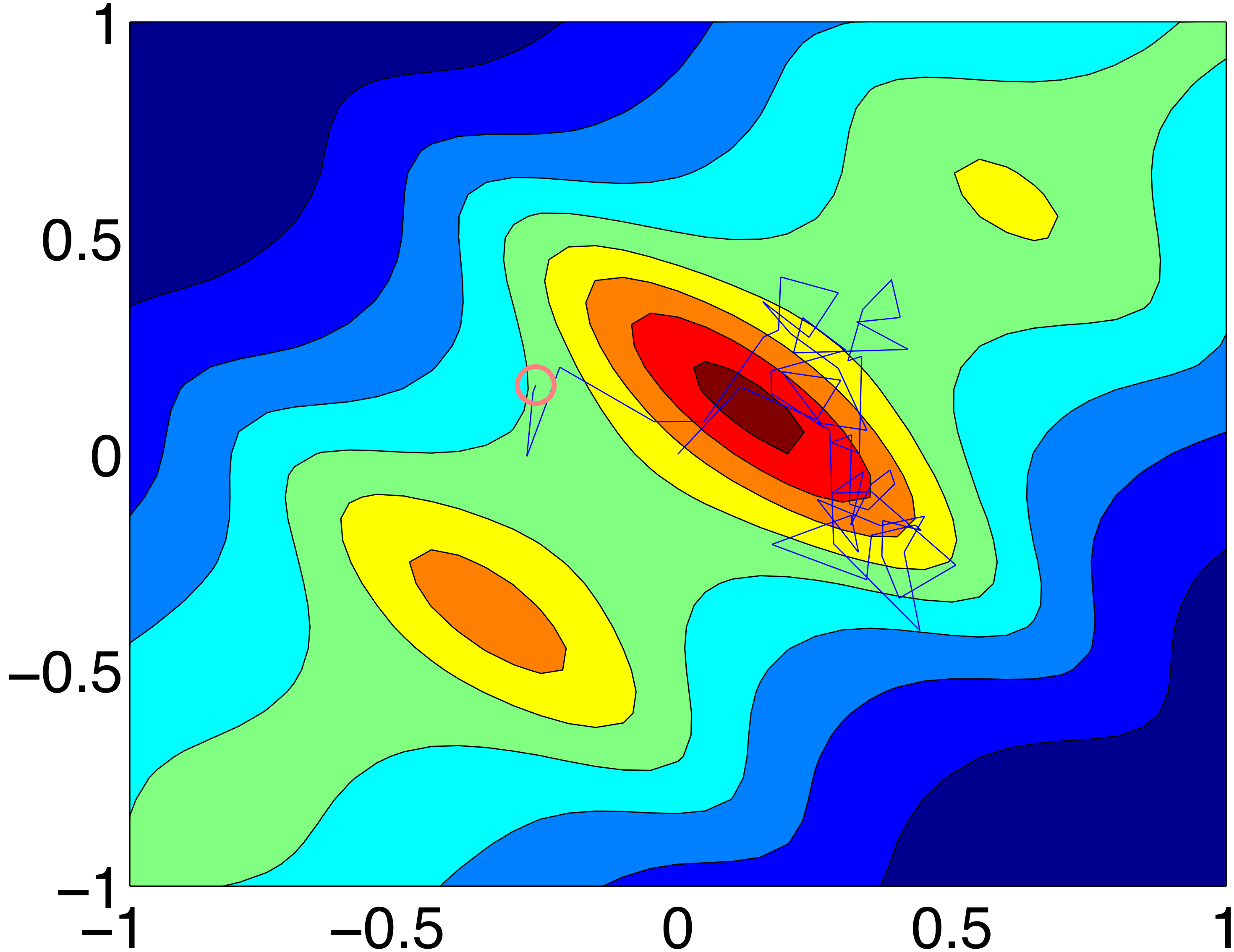
- Sample x' ~ $Q(x' \mid x)$
- Compute p = $\dfrac{R(x')}{R(x)} \dfrac{Q(x' \mid x)}{Q(x \mid x')}$
- With probability min(1, p), set x := x'
- Repeat for T steps; sample is $x_1, \ldots, x_T$ (will usually contain duplicates)

MH 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—we have reached stationary dist'n
- 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, …
- 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

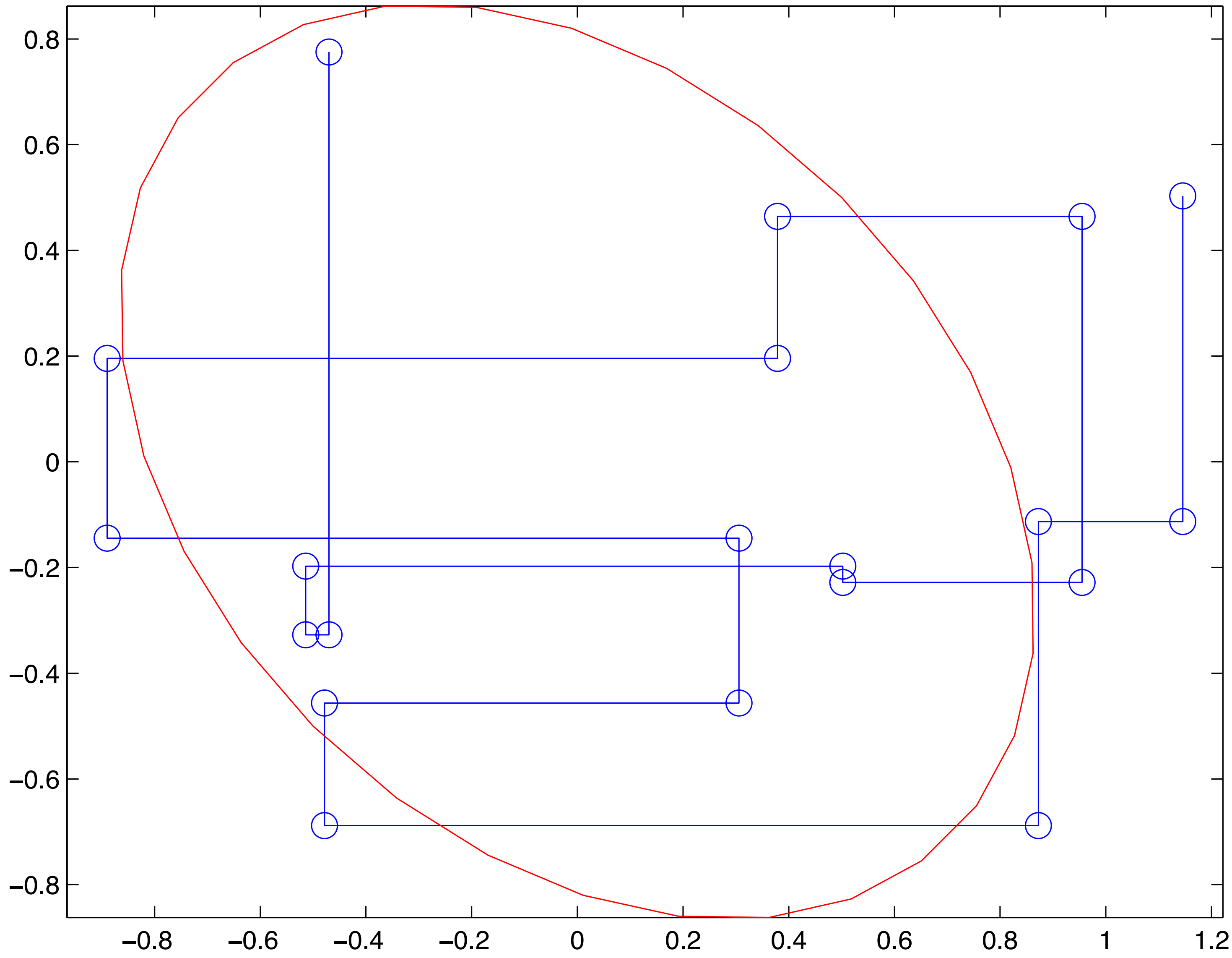
- $g(x) = x^2$
- True $E(g(X)) = 0.28\ldots$
- Proposal: $Q(x' \mid x) = N(x' \mid x, 0.25^2 I)$
- Acceptance rate 55–60%
- After 1000 samples, minus burn-in of 100:

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

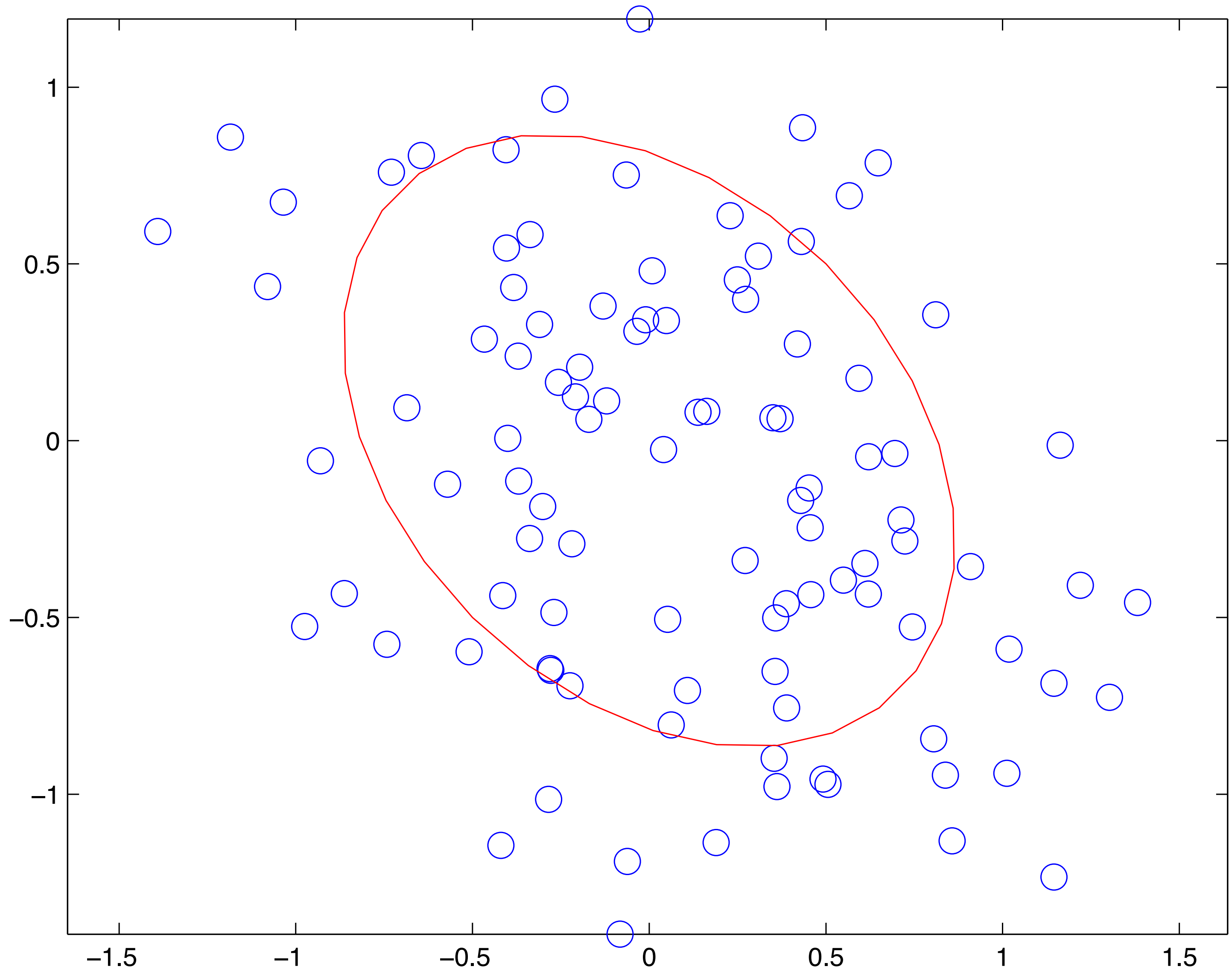
Gibbs sampler

- 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 = $\dfrac{P(x_i', x'_{\neg i})}{P(x_i, x_{\neg i})} \dfrac{P(x_i \mid x'_{\neg i})}{P(x_i' \mid x_{\neg i})}$

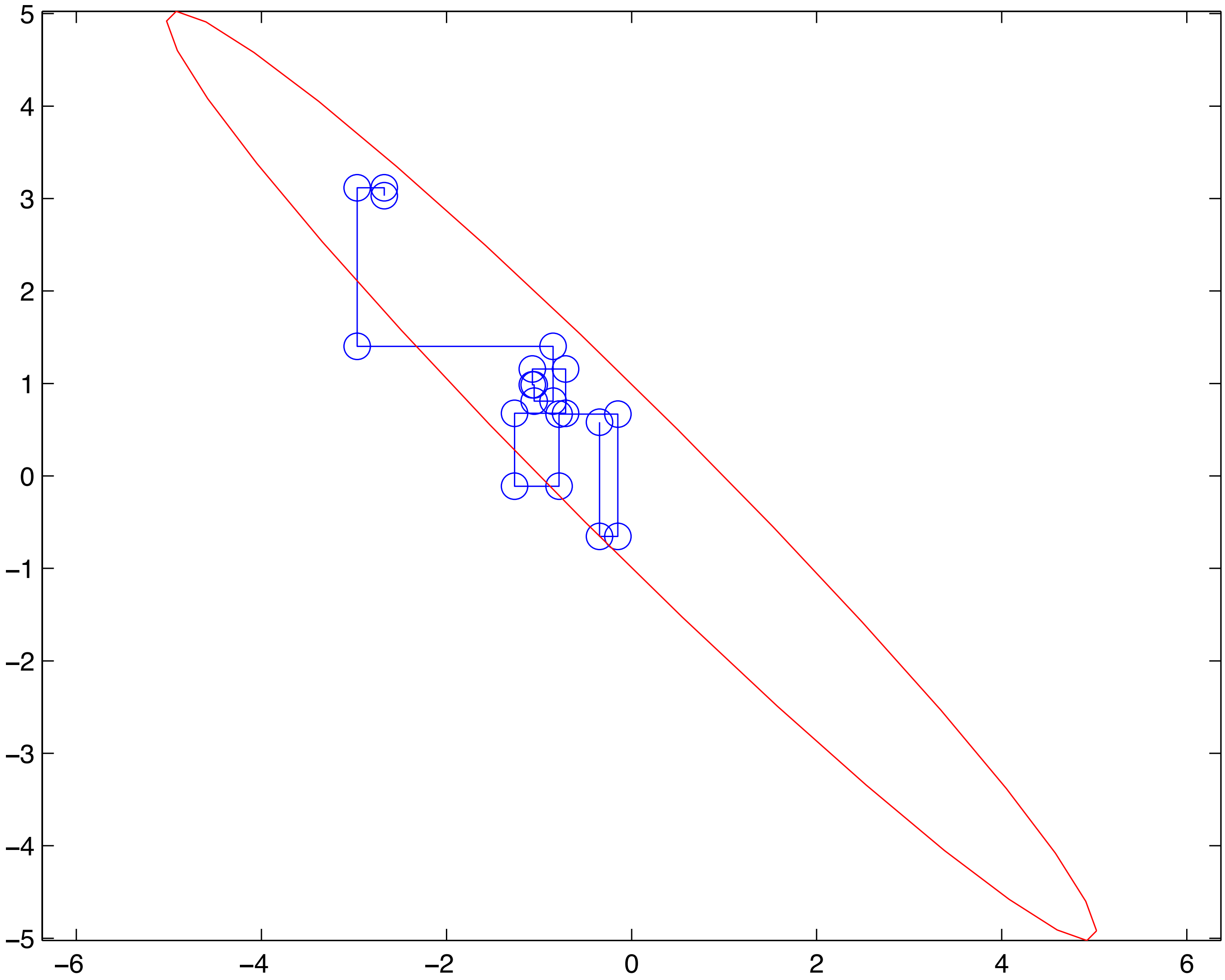
Gibbs derivation

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

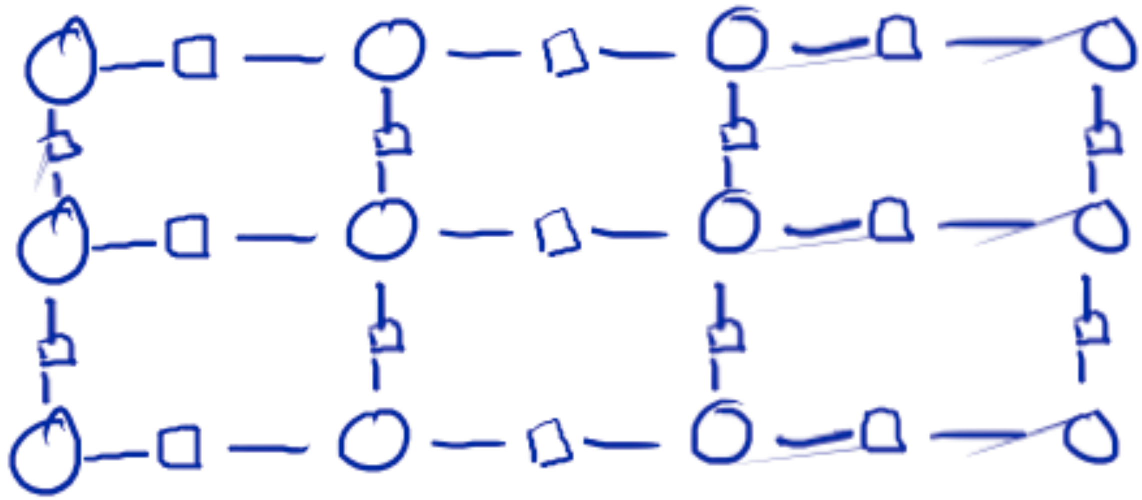
Gibbs in practice

- Proof of p=1 means Gibbs is often easy to implement
- Often works well
  - **if** we choose good blocks (but there may be no good blocking!)
- Fancier version: adaptive blocks, based on current **x**

Gibbs failure example



Sequential sampling



- In an HMM or DBN, to sample $P(\mathbf{X}_T)$, start from $\mathbf{X}_1$ and sample forward step by step
  - $\mathbf{X}_{t+1} \sim P(\mathbf{X}_{t+1} \mid \mathbf{X}_t)$
- $P(\mathbf{X}_{1:T}) = P(\mathbf{X}_1)\, P(\mathbf{X}_2 \mid \mathbf{X}_1)\, P(\mathbf{X}_3 \mid \mathbf{X}_2) \ldots$

Particle filter

- Can sample $\mathbf{x}_{t+1} \sim P(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$ using any algorithm from above
- If we use parallel importance sampling to get N samples at once from each $P(\mathbf{x}_t)$, we get a ***particle filter***
  - also need one more trick: ***resampling***
- Write $\mathbf{x}_{t,i}$ (i = 1…N) for sample at time t

Particle filter

- Want one sample from each of $P(\mathbf{X}_{t+1} \mid \mathbf{x}_{t,i})$
- Have only $Z\,P(\mathbf{X}_{t+1} \mid \mathbf{x}_{t,i})$
- For each i, pick $\mathbf{x}_{t+1,i}$ from proposal $Q(x)$
- Compute unnormalized importance weight

$$\hat{w}_i = ZP(\mathbf{x}_{t+1,i} \mid \mathbf{x}_{t,i}) / Q(\mathbf{x}_{t+1,i})$$

Particle filter

- Normalize weights:

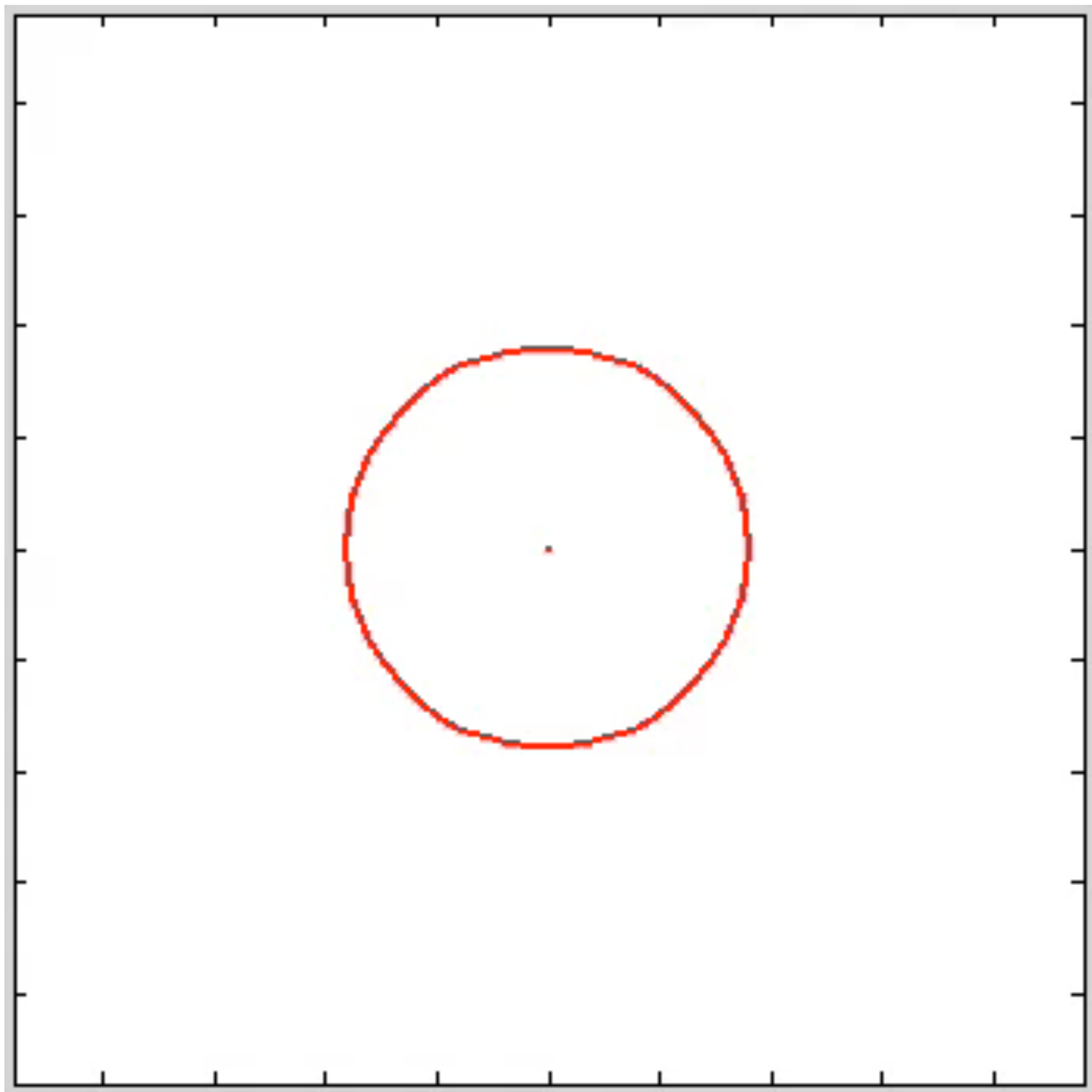
$$\bar{w} = \frac{1}{N} \sum_i \hat{w}_i \qquad w_i = \hat{w}_i / \bar{w}$$

- Now, $(w_i, \mathbf{x}_{t+1,i})$ is an approximate **weighted** sample from $P(\mathbf{X}_{t+1})$
- What will happen if we do this for *T=1, 2, …* ?

Resampling

- To get an unweighted sample, **resample**
- Sample N times (with replacement) from $\mathbf{x}_{t+1,i}$ with probabilities w_i/N
  - alternately: deterministically take $floor(w_i)$ copies of $\mathbf{x}_{t+1,i}$ and sample only from fractional part $[w_i - floor(w_i)]$
- Each $\mathbf{x}_{t+1,i}$ appears w_i times on average, so we're still a sample from $P(\mathbf{X}_{t+1})$

Particle filter example



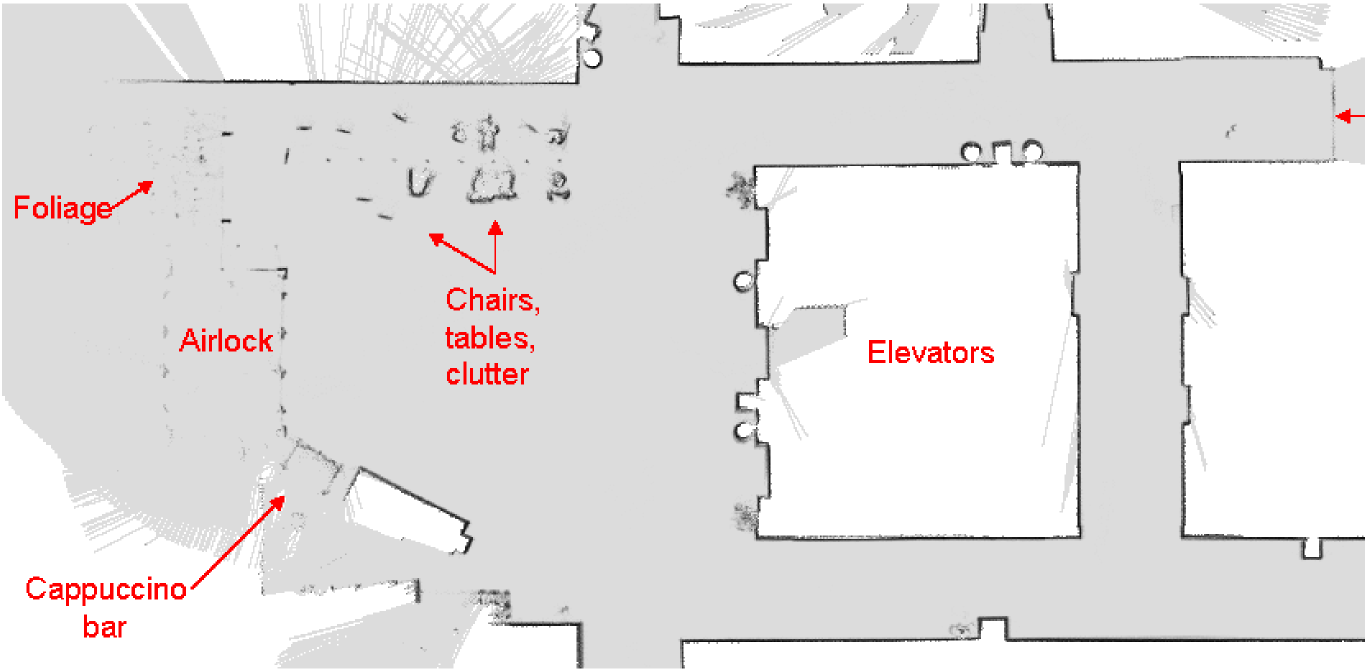
Learning

Learning

- Basic learning problem: given some experience, find a new or improved model
- Experience: a sample $x_1, \ldots, x_N$
- Model: want to predict $x_{N+1}, \ldots$

Example

- Experience = range sensor readings & odometry from robot
- Model = map of the world



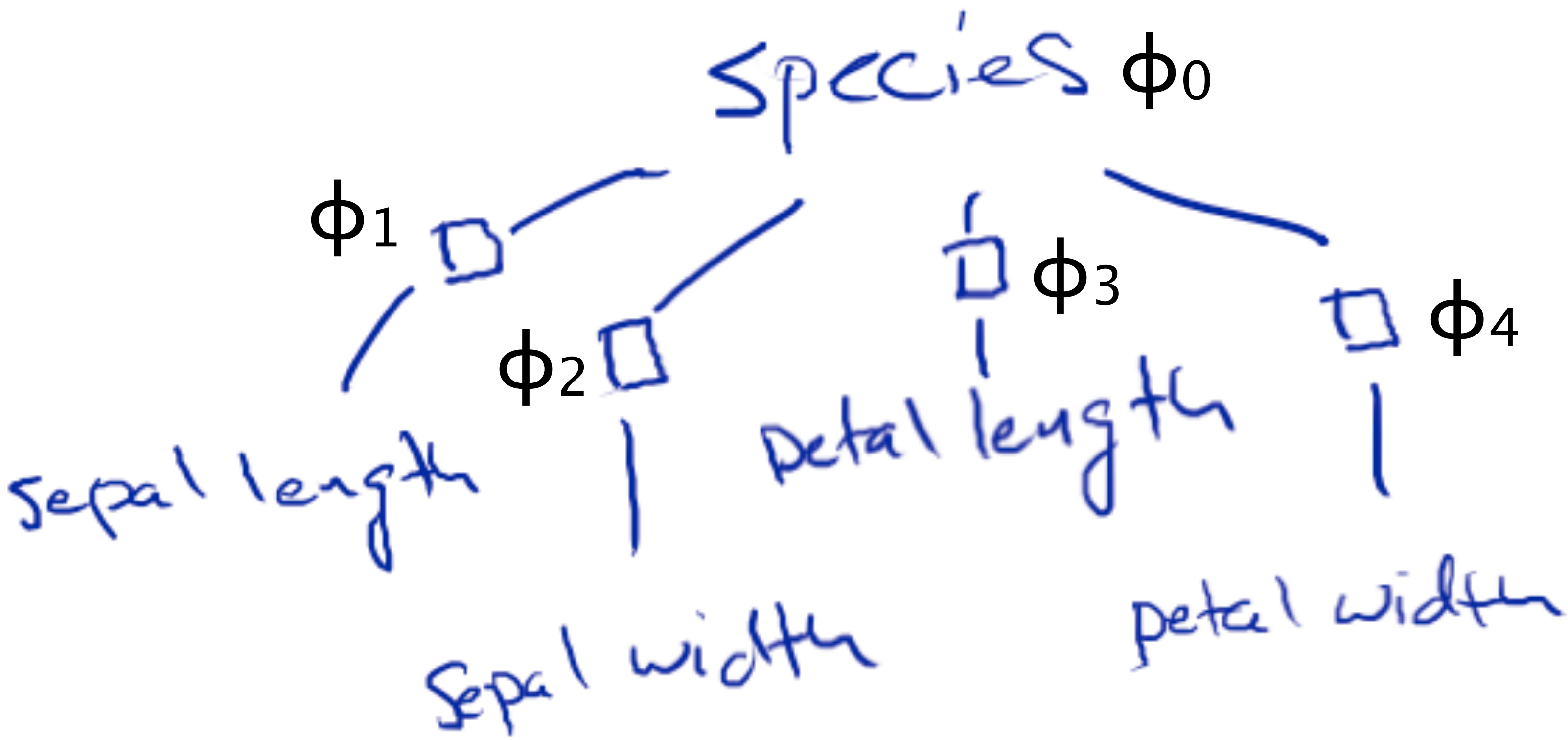
Example

- The “botanist learning problem”
  - Experience = physical measurements of surveyed specimens & expert judgements of their true species
  - Model = factor graph relating species to measurements

Sample data

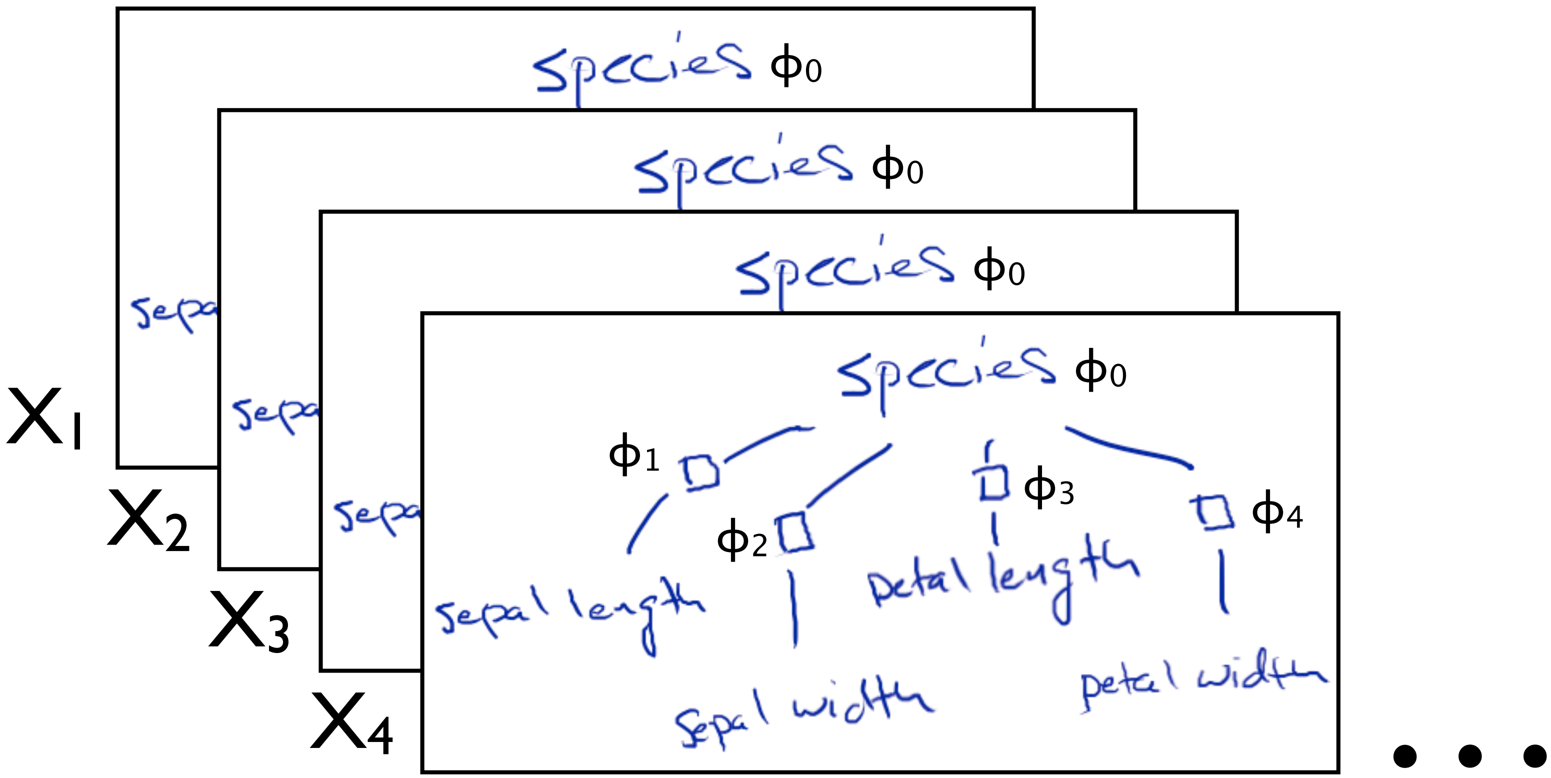
sepal length	sepal width	petal length	petal width	species
5.1	3.5	1.4	0.2	Iris setosa
5.6	3.0	4.5	1.5	Iris versicolor
4.9	3.0	1.4	0.2	Iris setosa
6.4	2.8	5.6	2.1	Iris virginica
5.8	2.7	4.1	1.0	Iris versicolor

Factor graph

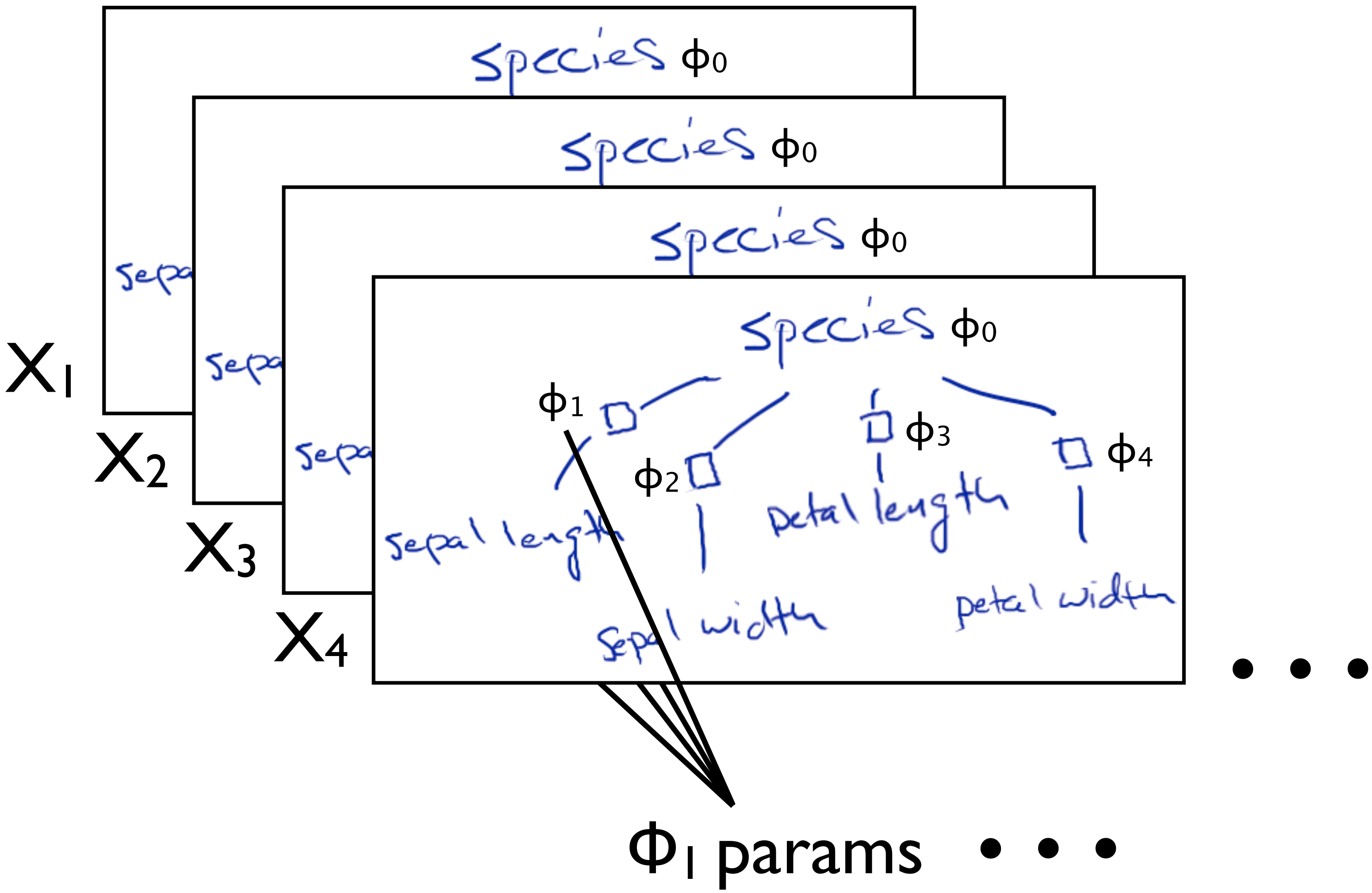


- One of many possible factor graphs
- Values of Φs not shown, but part of model

Factor graph



Factor graph



In general

- For our purposes, a model M is exactly a distribution $P(\mathbf{X} \mid M)$ over possible samples
- When is M ***better than*** M'? When $P(\mathbf{X} \mid M)$ is more accurate than $P(\mathbf{X} \mid M')$.
- Bayes rule encodes this: from ***prior*** $P(M)$ and ***evidence*** $\mathbf{X}$, compute ***posterior*** $P(M \mid \mathbf{X})$
  - $P(M \mid \mathbf{X}) = P(\mathbf{X} \mid M)\, P(M) \,/\, P(\mathbf{X})$
  - better predictions (higher $P(\mathbf{X} \mid M)$) yield higher posterior

Conditional model

- Split variables into $(\boldsymbol{X}, \boldsymbol{Y})$
- Suppose we always observe $\boldsymbol{X}$
- Two ways $P(\boldsymbol{X}, \boldsymbol{Y})$ and $P'(\boldsymbol{X}, \boldsymbol{Y})$ can differ:
  - $P(\boldsymbol{X}) \neq P'(\boldsymbol{X})$, and/or
  - $P(\boldsymbol{Y} \mid \boldsymbol{X}) \neq P'(\boldsymbol{Y} \mid \boldsymbol{X})$
- First way doesn't matter for decisions
- ***Conditional model***: only specifies $P(\boldsymbol{Y} \mid \boldsymbol{X}, M)$

Conditional model example

- Experience = samples of (***X***, ***Y***)
- ***X*** = features of object
- ***Y*** = whether object is a "framling"
- Model = rule for deciding whether a new object is a framling

Sample data & possible model

tall	pointy	blue	framling
T	T	F	T
T	F	F	T
F	T	F	F
T	T	T	F
T	F	F	T

$H = \text{tall} \wedge \neg\text{blue}$

Hypothesis space

- Hypothesis space $\mathscr{H}$ = set of models we are willing to consider
  - for philosophical or computational reasons
- E.g., all factor graphs of a given structure
- Or, all conjunctions of up to two literals
- Prior is a distribution over $\mathscr{H}$

A simple learning algorithm

= Bayes Rule

- Conditional learning: samples $(\boldsymbol{x}_i, y_i)$
- Let $\mathscr{H}$ be a set of propositional formulae
  - $\mathscr{H} = \{ H_1, H_2, \ldots \}$
- H is ***consistent*** if $H(\boldsymbol{x}_i) = y_i$ for all i
- ***Version space*** $V = \{$ all consistent $H \} \subseteq \mathscr{H}$
- ***Version space algorithm***: predict y = majority vote of $H(\boldsymbol{x})$ over all $H \in V$

Framlings

tall	pointy	blue	framling
T	T	F	T
T	F	F	T
F	T	F	F
T	T	T	F
T	F	F	T

- $\mathscr{H}$ = { conjunctions of up to 2 literals } = { T, F, tall, pointy, blue, ¬tall, ¬pointy, ¬blue, tall ∧ pointy, tall ∧ blue, pointy ∧ blue, ¬tall ∧ pointy, … }

Framlings

tall	pointy	blue	framling
T	T	F	T
T	F	F	T
F	T	F	F
T	T	T	F
T	F	F	T

x_1: T, tall, pointy, $\neg$ blue, tall $\wedge$ pointy, tall $\wedge$ $\neg$ blue, pointy $\wedge$ $\neg$ blue

x_2: T, tall, $\neg$ blue, tall $\wedge$ $\neg$ blue

x_3: tall, tall $\wedge$ $\neg$ blue

x_4: tall $\wedge$ $\neg$ blue

$T \to \times$

$T \to \times$

Analysis

- ***Mistake*** = make wrong prediction
- If some H $\in \mathcal{H}$ is always right, eventually we'll eliminate all competitors, and make no more mistakes
- If no H $\in \mathcal{H}$ is always right, eventually V will become empty
  - e.g., if ***label noise*** or ***feature noise***

Analysis

- Suppose | $\mathscr{H}$ | = N
- How many mistakes could we make?

if $\mathcal{H}$ contains consistent

$\lceil \log_2 N \rceil$

Analysis

- Suppose | $\mathscr{H}$ | = N
- How many mistakes could we make?
- Since we predict w/ ***majority*** of V, after any mistake, we eliminate half (or more) of V
- Can't do that more than $\log_2(N)$ times

Discussion

- In example, $N = 20$, $\log_2(N) = 4.32$
- Made only 2 mistakes
- Mistake bound: limits wrong decisions, as desired
- But, required strong assumptions (no noise, true H contained in $\mathcal{H}$)
- Could be very slow!