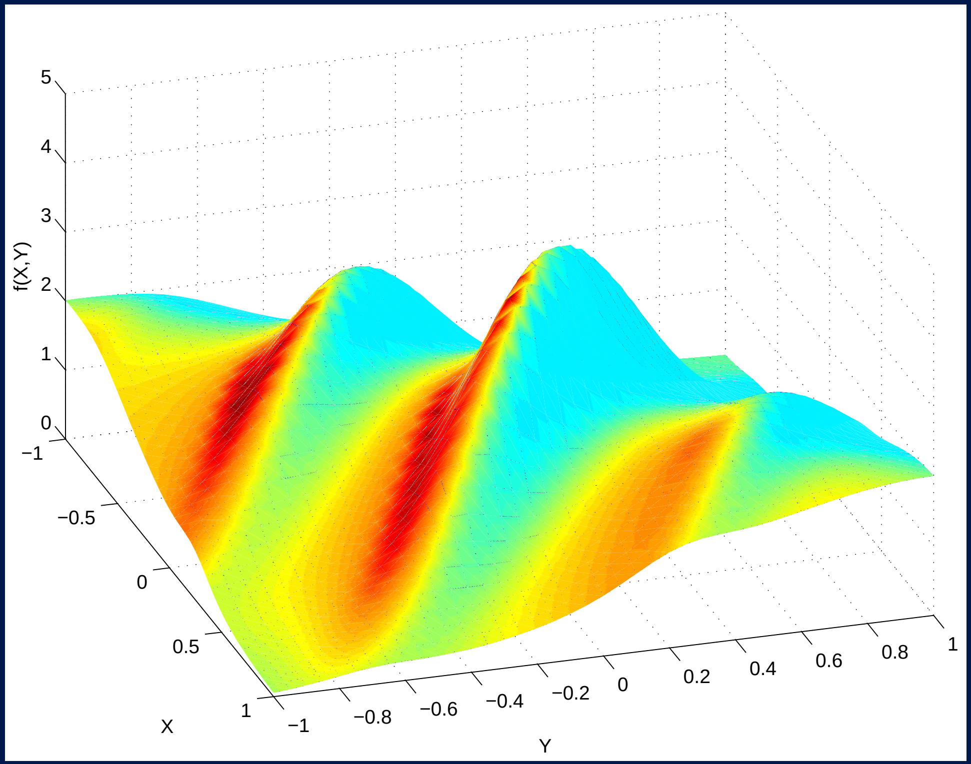
Monte Carlo Methods

Geoff Gordon
ggordon@cs.cmu.edu

February 9, 2006

Numerical integration problem



$$\int_{x \in \mathcal{X}} f(x)dx$$

Used for: function approximation

$$f(x) \approx \alpha_1 f_1(x) + \alpha_2 f_2(x) + \dots$$

Orthonormal system: $\int f_i(x)^2 dx = 1$ and $\int f_i(x) f_j(x) dx = 0$

- Fourier ($\sin x$, $\cos x$, ...)
- Chebyshev (1, x, $2x^2 - 1$, $4x^3 - 3x$, ...)
- ...

Coefficients are

$$\alpha_i = \int f(x) f_i(x) dx$$

Used for: optimization

Optimization problem: minimize $T(x)$ for $x \in \mathcal{X}$

Assume unique global optimum x^*

Define Gibbs distribution with temperature $1/\beta$ for $\beta > 0$:

$$P_\beta(x) = \frac{1}{Z(\beta)} \exp(-\beta T(x))$$

As $\beta \to \infty$, have $E_{x \sim P_\beta}(x) \to x^*$

Simulated annealing: track $E_\beta(x) = \int x P_\beta(x) dx$ as $\beta \to \infty$

Used for: Bayes net inference

Undirected Bayes net on $x = x_1, x_2, \ldots$:

$$P(x) = \frac{1}{Z} \prod_j \phi_j(x)$$

Typical inference problem: compute $E(x_i)$

Belief propagation is fast if argument lists of ϕ_js are small and form a junction tree

If not, MCMC

Used for: SLAM

Used for

Image segmentation

Tracking radar/sonar returns

Outline

Uniform sampling, importance sampling

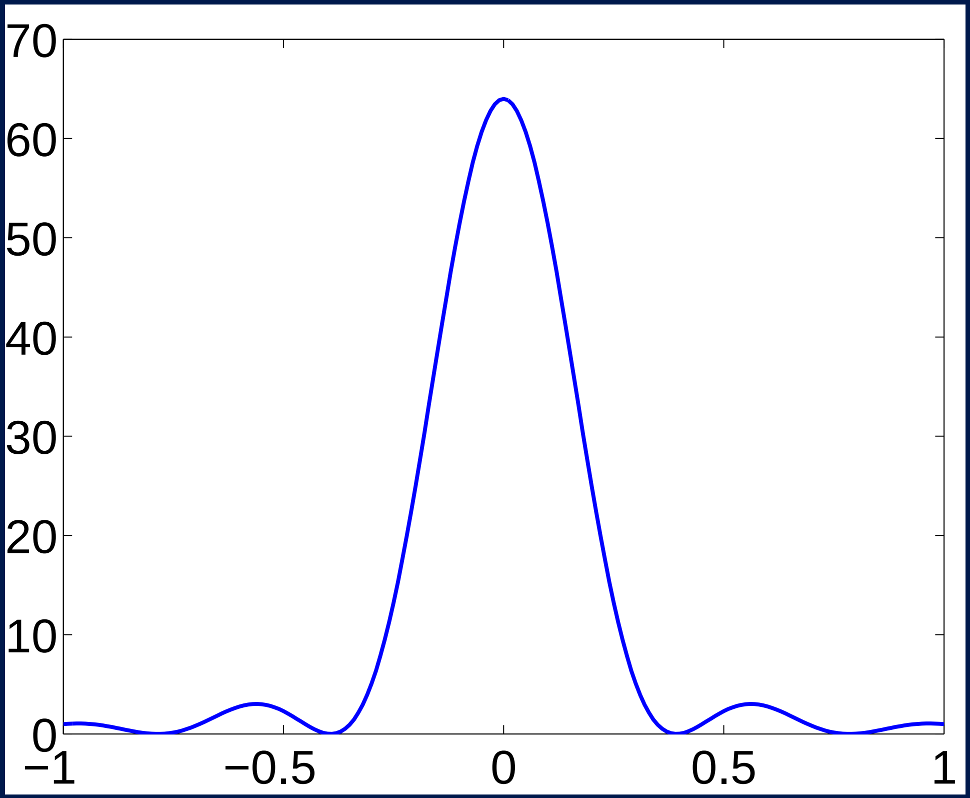
MCMC and Metropolis-Hastings algorithm

What if $f(x)$ has internal structure:

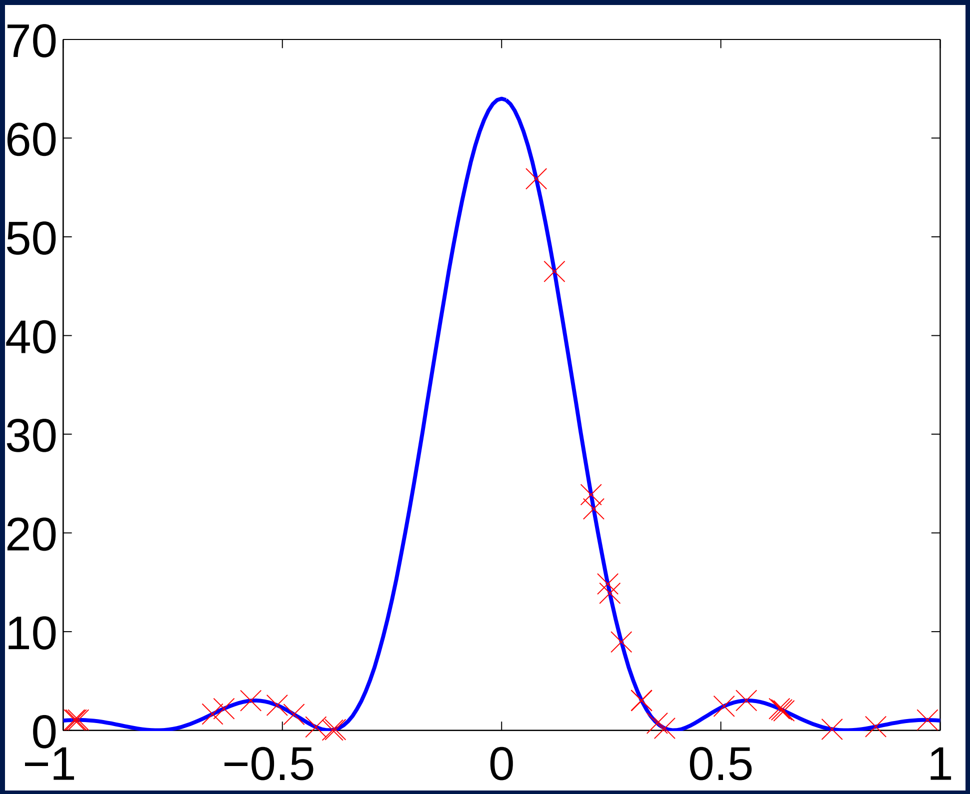
- SIS, SIR (particle filter)
- Gibbs sampler

Combining SIR w/ MCMC

A one-dimensional problem



Uniform sampling



true integral 24.0; uniform sampling 14.7 w/ 30 samples

Uniform sampling

Pick an x uniformly at random from $\mathcal{X}$

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

where V is volume of $\mathcal{X}$

So $E(V f(x)) =$ desired integral

But variance can be big (esp. if V large)

Uniform sampling

Do it a bunch of times: pick x_i, compute

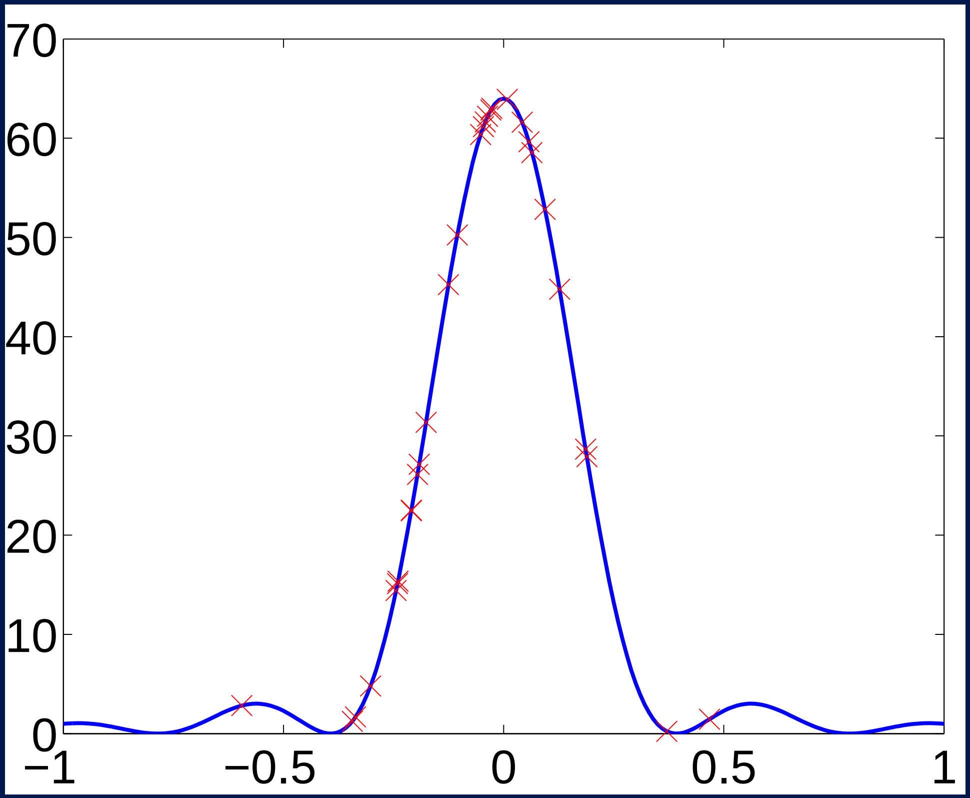
$$\frac{V}{n}\sum_{i=1}^{n} f(x_i)$$

Same expectation, lower variance

Variance decreases as $1/n$ (standard dev $1/\sqrt{n}$)

Not all that fast; limitation of most MC methods

Nonuniform sampling



true integral 24.0; importance sampling ($Q = N(0, 0.25^2)$) 25.8

Importance sampling

Suppose we pick x nonuniformly, $x \sim Q(x)$

$Q(x)$ is *importance distribution*

Use Q to (approximately) pick out areas where f is large

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

Not what we want

Importance sampling

Define $g(x) = f(x)/Q(x)$

Now

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

Importance sampling

So, sample x_i from Q, take average of $g(x_i)$:

$$\frac{1}{n}\sum_i f(x_i)/Q(x_i)$$

$w_i = 1/Q(x_i)$ is *importance weight*

Uniform sampling is just importance sampling with Q = uniform = $1/V$

Parallel importance sampling

Suppose $f(x) = P(x)g(x)$

Desired integral is $\int f(x)dx = E_P(g(x))$

But suppose we only know $g(x)$ and $\lambda P(x)$

Parallel importance sampling

Pick n samples x_i from proposal $Q(x)$

If we could compute importance weights $w_i = P(x_i)/Q(x_i)$, then

$$E_Q\left[w_i g(x_i)\right] = \int Q(x)\frac{P(x)}{Q(x)}g(x)dx$$
$$= \int f(x)dx$$

so $\frac{1}{n}\sum_i w_i g(x_i)$ would be our IS estimate

Parallel importance sampling

Assign raw importance weights $\hat{w}_i = \lambda P(x_i)/Q(x_i)$

$$\begin{aligned} E(\hat{w}_i) &= \int Q(x)(\lambda P(x)/Q(x))dx \\ &= \lambda \int P(x)dx \\ &= \lambda \end{aligned}$$

So w_i is an unbiased estimate of λ

Define $\bar{w} = \frac{1}{n}\sum_i w_i \quad \Rightarrow$ also unbiased, but lower variance

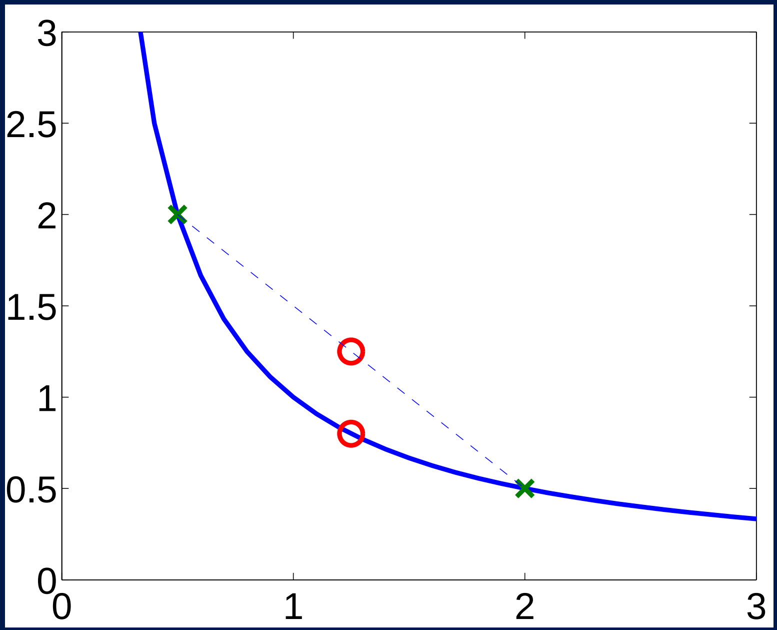
Parallel importance sampling

$\widehat{w}_i/\bar{w}$ is approximately w_i, but computed without knowing λ

So, make the estimate

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

Parallel IS bias

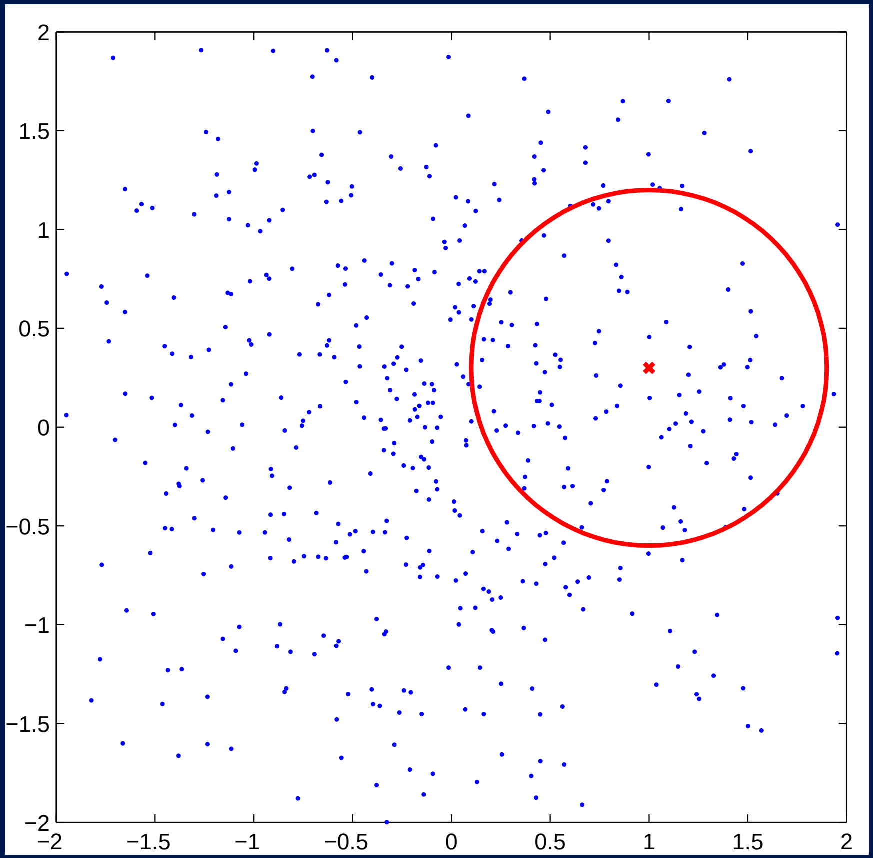


Parallel IS is biased

$E(\bar{w}) = \lambda$, but $E(1/\bar{w}) \neq 1/\lambda$ in general

Bias $\rightarrow$ 0 as $n \rightarrow \infty$, since variance of $\bar{w} \rightarrow 0$

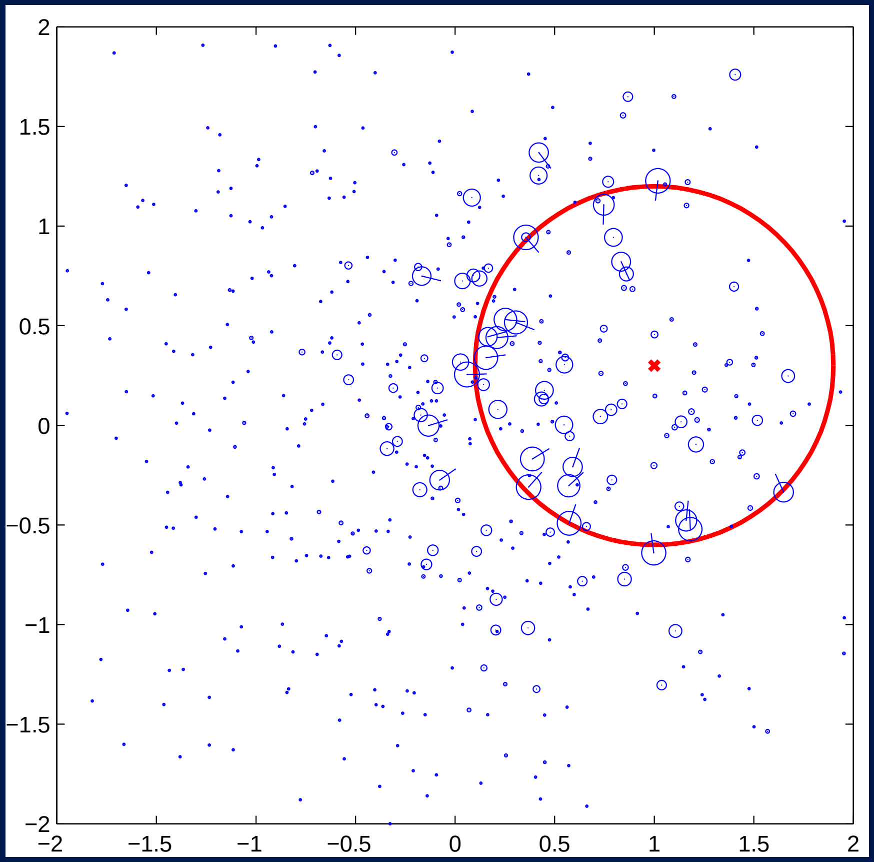
Parallel IS example



$$Q : (x, y) \sim N(1, 1) \qquad \theta \sim U(-\pi, \pi)$$

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

Parallel IS example



Posterior $E(x, y, \theta) = (0.496, 0.350, 0.084)$

Back to n dimensions

Picking a good sampling distribution becomes hard in high-d

Major contribution to integral can be hidden in small areas

Danger of missing these areas $\Rightarrow$ need to search for areas of large $f(x)$

Naively, searching could bias our choice of x in strange ways, making it hard to design an unbiased estimator

Markov chain Monte-Carlo

Design a Markov chain M whose moves tend to increase $f(x)$ if it is small

This chain encodes a search strategy: start at an arbitrary x, run chain for a while to find an x with reasonably high $f(x)$

For x found by an arbitrary search algorithm, don't know what importance weight we should use to correct for search bias

For x found by M after sufficiently many moves, can use stationary distribution of M, $P_M(x)$, as importance weight

Picking P_M

MCMC works well if $f(x)/P_M(x)$ has low variance

$f(x) \gg P_M(x)$ means there's a region of comparatively large $f(x)$ that we don't sample enough

$f(x) \ll P_M(x)$ means we waste samples in regions where $f(x) \approx 0$

So, e.g., if $f(x) = g(x)P(x)$, could ask for $P_M = P$

Metropolis-Hastings

Way of getting chain M with desired P_M

Basic strategy: start from arbitrary x

Repeatedly tweak x a little to get x'

If $P_M(x') \geq P_M(x)\alpha$, move to x'

If $P_M(x') \ll P_M(x)\alpha$, stay at x

In intermediate cases, randomize

Proposal distributions

MH has one parameter: how do we tweak x to get x'

Encoded in one-step proposal distribution $Q(x' \mid x)$

Good proposals explore quickly but remain in regions of high $P_M(x)$

Optimal proposal: $P(x' \mid x) = P_M(x')$ for all x

Metropolis-Hastings algorithm

MH transition probability $T_M(x' \mid x)$ is defined as follows:

Sample $x' \sim Q(x' \mid x)$

Compute $p = \frac{P_M(x')}{P_M(x)} \frac{Q(x|x')}{Q(x'|x)} = \frac{P_M(x')}{P_M(x)} \alpha$

With probability p, set $x \leftarrow x'$

Repeat

Stop after, say, t steps (possibly $\ll t$ distinct samples)

Metropolis-Hastings notes

Only need P_M up to constant factor—nice for problems where normalizing constant is hard

Efficiency determined by

- how fast $Q(x' \mid x)$ moves us around
- how high acceptance probability p is

Tension between fast Q and high p

Metropolis-Hastings proof

Given $P_M(x)$ and $T_M(x' \mid x)$

Want to show P_M is stationary distribution for T_M

Based on "detailed balance" condition

$$P_M(x)T_M(x' \mid x) = P_M(x')T_M(x \mid x') \qquad \forall x, x'$$

Detailed balance implies

$$\begin{aligned} \int P_M(x)T_M(x' \mid x)dx &= \int P_M(x')T_M(x \mid x')dx \\ &= P_M(x') \int T_M(x \mid x')dx \\ &= P_M(x') \end{aligned}$$

So, if we can show detailed balance we are done

Proving detailed balance

Want to show $P_M(x)T_M(x' \mid x) = P_M(x')T_M(x \mid x')$ for $x \neq x'$

$$P_M(x)T_M(x' \mid x) = P_M(x)Q(x' \mid x) \max\left(1, \frac{P_M(x')}{P_M(x)} \frac{Q(x \mid x')}{Q(x' \mid x)}\right)$$

$$P_M(x')T_M(x \mid x') = P_M(x')Q(x \mid x') \max\left(1, \frac{P_M(x)}{P_M(x')} \frac{Q(x' \mid x)}{Q(x \mid x')}\right)$$

Exactly one of the two max statements chooses 1

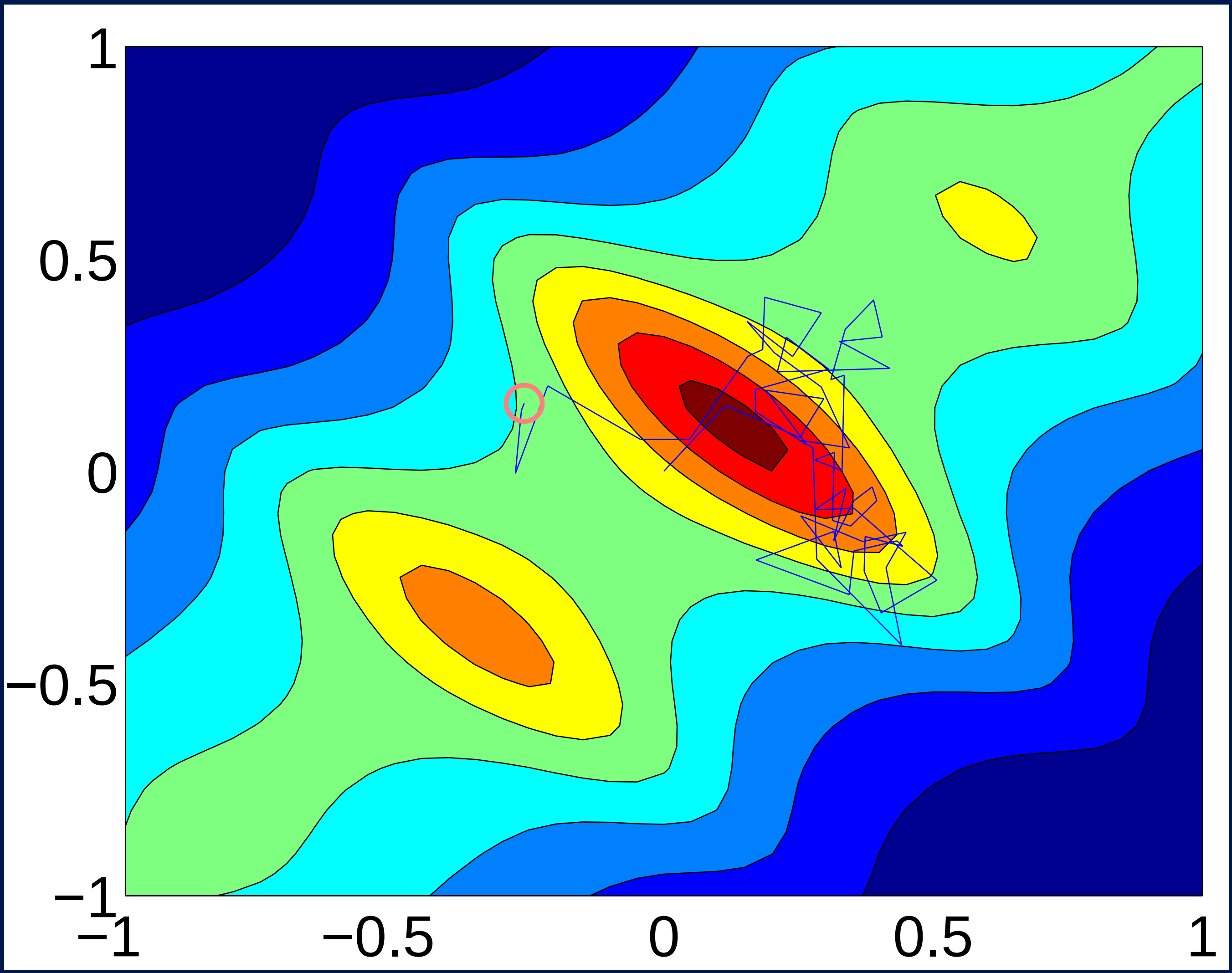
Wlog, suppose it's the first

Detailed balance

$$
\begin{aligned}
P_M(x) T_M(x' \mid x) &= P_M(x) Q(x' \mid x) \\
P_M(x') T_M(x \mid x') &= P_M(x') Q(x \mid x') \frac{P_M(x)}{P_M(x')} \frac{Q(x' \mid x)}{Q(x \mid x')} \\
&= P_M(x) Q(x' \mid x)
\end{aligned}
$$

So, P_M is stationary distribution of Metropolis-Hastings sampler

Metropolis-Hastings example



MH example accuracy

True $E(x^2) \approx 0.28$

$\sigma = 0.25$ in proposal leads to 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
```

Structure in $f(x)$

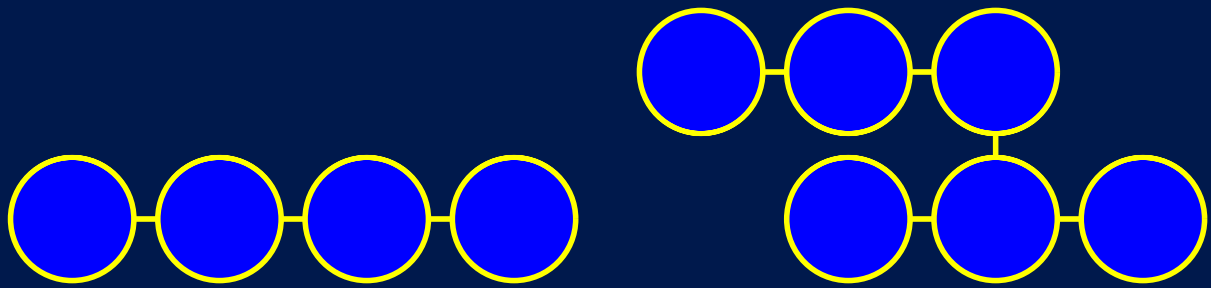
Suppose $f(x) = g(x)P(x)$ as above

And suppose $P(x)$ can be factored, e.g.,

$$P(x) = \frac{1}{Z}\phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{245}(x_2, x_4, x_5)\ldots$$

Then we can take advantage of structure to sample from P efficiently and compute $E_P(g(x))$

Linear or tree structure



$$P(x) = \frac{1}{Z}\phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4)$$

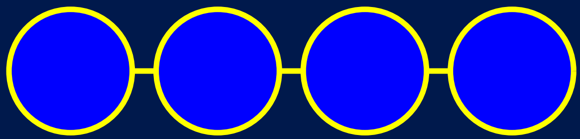
Pick a node as root arbitrarily

Sample a value for root

Sample children conditional on parents

Repeat until we have sampled all of x

Sequential importance sampling



Assume a chain graph $x_1 \ldots x_T$ (tree would be fine too)

Want to estimate $E(x_T)$

Can evaluate but not sample from $P(x_1)$, $P(x_{t+1} \mid x_t)$

Sequential importance sampling

Suppose we have proposals $Q(x_1)$, $Q(x_{t+1} \mid x_t)$

Sample $x_1 \sim Q(x_1)$, compute weight $w_1 = P(x_1)/Q(x_1)$

Sample $x_2 \sim Q(x_2 \mid x_1)$, weight $w_2 = w_1 \cdot P(x_2 \mid x_1)/Q(x_2 \mid x_1)$

$\ldots$ continue until last variable x_T

Weight w_T at final step is $P(x)/Q(x)$

Problems with SIS

w_T often has really high variance

We often only know $P(x_{t+1} \mid x_t)$ up to a constant factor

For example, in an HMM after conditioning on observation y_{t+1},

$$P(x_{t+1} \mid x_t, y_{t+1}) = \frac{1}{Z} P(x_{t+1} \mid x_t) P(y_{t+1} \mid x_{t+1})$$

Parallel SIS

Apply parallel IS trick to SIS:

- Generate n SIS samples x^i with weights w^i
- Normalize w^i so $\sum_i w^i = n$

Gets rid of problem of having to know normalized Ps

Introduces a bias which $\rightarrow$ 0 as $n \rightarrow \infty$

Still not practical (variance of w^i)

Sequential importance resampling

SIR = particle filter, sample = particle

Run SIS, keep weights normalized to sum to n

Monitor variance of weights

If too few particles get most of the weight, *resample* to fix it

Resampling reduces variance of final estimate, but increases bias due to normalization

Resampling

After normalization, suppose a particle has weight $0 < w < 1$

Set its weight to 1 w/ probability w, or to 0 w/ probability $1 - w$

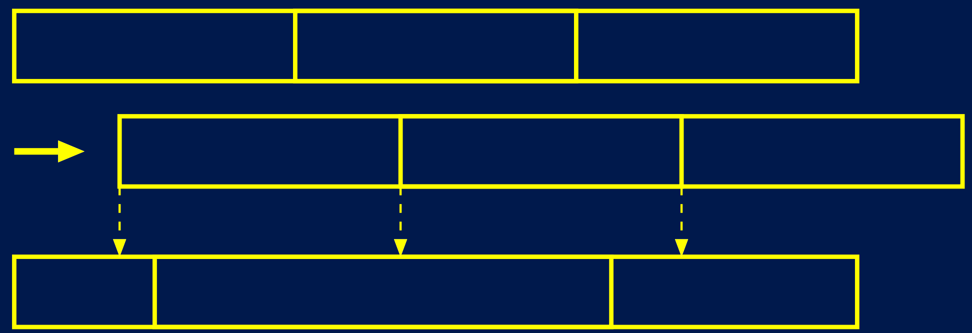
$E(\texttt{weight})$ is still w, but can throw particle away if weight is 0

Resampling

A particle with weight $w \geq 1$ will get $\lfloor w \rfloor$ copies for sure, plus one with probability $w - \lfloor w \rfloor$

Total number of particles is $\approx n$

Can make it exactly n:

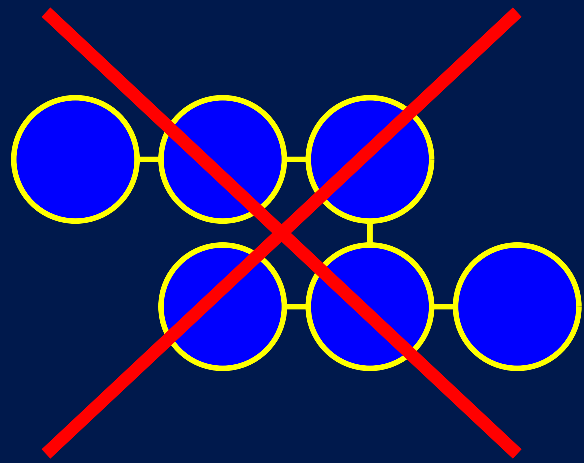


High-weight particles are replicated at expense of low-weight ones

SIR example

[DC factored filter movie]

Gibbs sampler



Recall

$$P(x) = \frac{1}{Z}\phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{245}(x_2, x_4, x_5)\dots$$

What if we don't have a nice tree structure?

Gibbs sampler

MH algorithm for sampling from $P(x)$

Proposal distribution: pick an i at random, resample x_i from its conditional distribution holding $x_{\neg i}$ fixed

That is, $Q(x, x') = 0$ if x and x' differ in more than one component

If x and x' differ in component i,

$$Q(x' \mid x) = \frac{1}{n} P(x'_i \mid x_{\neg i})$$

Gibbs acceptance probability

MH acceptance probability is

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

For Gibbs, suppose we are resampling x_1 which participates in $\phi_7(x_1, x_4)$ and $\phi_9(x_1, x_3, x_6)$

$$\frac{P(x')}{P(x)} = \frac{\phi_7(x_1', x_4)\phi_9(x_1', x_3, x_6)}{\phi_7(x_1, x_4)\phi_9(x_1, x_3, x_6)}$$

First factor is easy

Gibbs acceptance probability

Second factor:

$$\frac{Q(x \mid x')}{Q(x' \mid x)} = \frac{P(x_1 \mid x'_{\neg 1})}{P(x'_1 \mid x_{\neg 1})}$$

$P(x'_1 \mid x_{\neg 1})$ is simple too:

$$P(x'_1 \mid x_{\neg 1}) = \frac{1}{Z}\phi_7(x'_1, x_4)\phi_9(x'_1, x_3, x_6)$$

So

$$\frac{Q(x \mid x')}{Q(x' \mid x)} = \frac{\phi_7(x_1, x_4)\phi_9(x_1, x_3, x_6)}{\phi_7(x'_1, x_4)\phi_9(x'_1, x_3, x_6)}$$

Better yet

$$\frac{P(x')}{P(x)} = \frac{\phi_7(x'_1, x_4)\phi_9(x'_1, x_3, x_6)}{\phi_7(x_1, x_4)\phi_9(x_1, x_3, x_6)}$$

$$\frac{Q(x \mid x')}{Q(x' \mid x)} = \frac{\phi_7(x_1, x_4)\phi_9(x_1, x_3, x_6)}{\phi_7(x'_1, x_4)\phi_9(x'_1, x_3, x_6)}$$

The two factors cancel!

So $p = 1$: always accept

Gibbs in practice

Simple to implement

Often works well

Common failure mode: knowing $x_{\neg i}$ "locks down" x_i

Results in slow mixing, since it takes a lot of low-probability moves to get from x to a very different x'

Locking down

E.g., handwriting recognition: "antidisestablishmen?arianism"

Even if we do propose and accept "antidisestablishmenqarianism", likely to go right back

E.g., image segmentation: if all my neighboring pixels in an 11×11 region are background, I'm highly likely to be background as well

E.g., HMMs: knowing x_{t-1} and x_{t+1} often gives a good idea of x_t

Sometimes conditional on values of other variables: ai? $\mapsto$ $\{$aid, ail, aim, air$\}$ but th? $\mapsto$ the (and maybe thy or tho)

Worked example

[switch to Matlab]

Related topics

Reversible-jump MCMC

- for when we don't know the dimension of x

Rao-Blackwellization

- hybrid between Monte-Carlo and exact
- treat some variables exactly, sample over rest

Swendsen-Wang

- modification to Gibbs that mixes faster in locked-down distributions

Data-driven proposals: EKPF, UPF