

Bayes Nets IV: Sampling

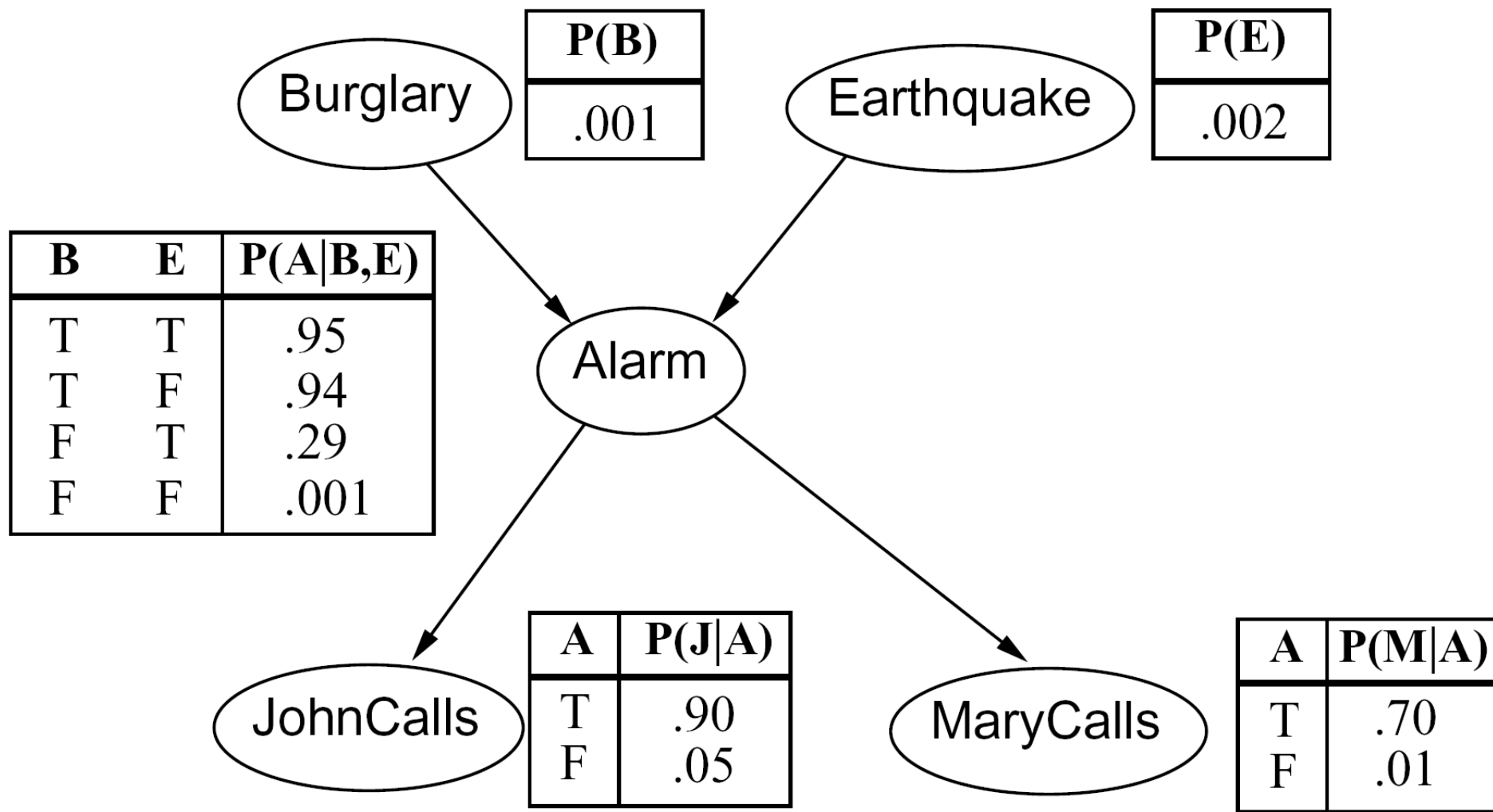
Many slides courtesy of
Dan Klein, Stuart Russell,
or Andrew Moore

CS 5300 / CS 6300
Artificial Intelligence
Spring 2010

Hal Daumé III
hal@cs.utah.edu

www.cs.utah.edu/~hal/courses/2010S_AI

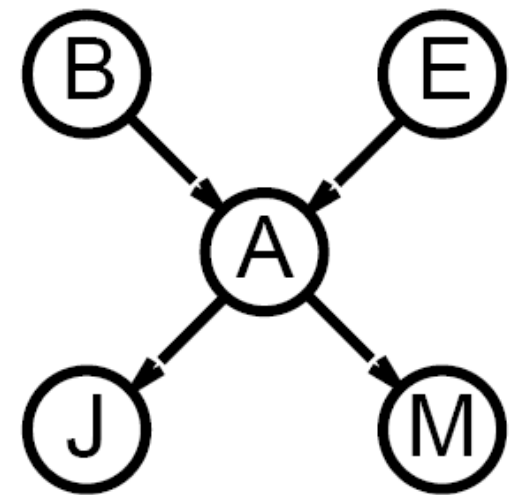
Reminder: Alarm Network



Inference by Enumeration

- Given unlimited time, inference in BNs is easy
- Recipe:
 - State the marginal probabilities you need
 - Figure out ALL the atomic probabilities you need
 - Calculate and combine them
- Example:

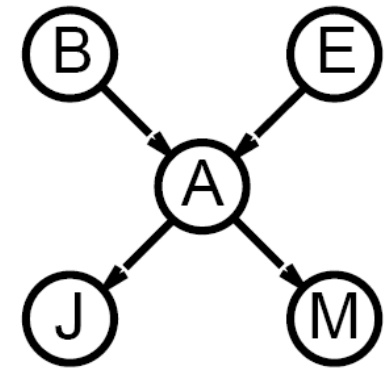
$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)}$$



General Variable Elimination

- Query: $P(Q|E_1 = e_1, \dots, E_k = e_k)$
- Start with initial factors:
 - Local CPTs (but instantiated by evidence)
- While there are still hidden variables (not Q or evidence):
 - Pick a hidden variable H
 - Join all factors mentioning H
 - Project out H
- Join all remaining factors and normalize

Example



$$P(B|j, m) \propto P(B, j, m)$$

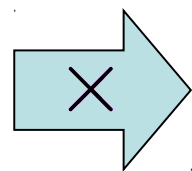
$P(B)$	$P(E)$	$P(A B, E)$	$P(j A)$	$P(m A)$
--------	--------	-------------	----------	----------

Choose A

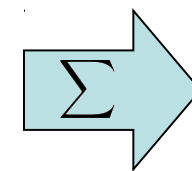
$$P(A|B, E)$$

$$P(j|A)$$

$$P(m|A)$$



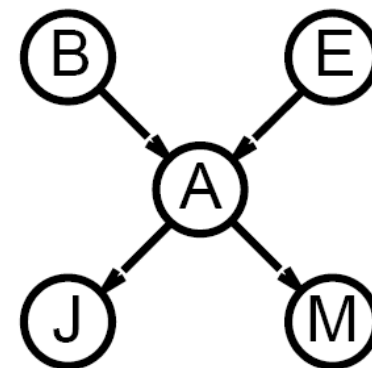
$$P(j, m, A|B, E)$$



$$P(j, m|B, E)$$

$P(B)$	$P(E)$	$P(j, m B, E)$
--------	--------	----------------

Example



$$P(B) \quad P(E) \quad P(j, m|B, E)$$

Choose E

$$\begin{array}{c} P(E) \\ P(j, m|B, E) \end{array} \xrightarrow{\times} P(j, m, E|B) \xrightarrow{\Sigma} P(j, m|B)$$

$$P(B) \quad P(j, m|B)$$

Finish with B

$$\begin{array}{c} P(B) \\ P(j, m|B) \end{array} \xrightarrow{\times} P(j, m, B) \xrightarrow{\text{Normalize}} P(B|j, m)$$

Variable Elimination

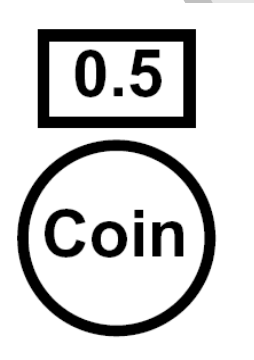
- What you need to know:
 - Should be able to run it on small examples, understand the factor creation / reduction flow
 - Better than enumeration: VE caches intermediate computations
 - Saves time by marginalizing variables as soon as possible rather than at the end
 - Polynomial time for tree-structured graphs – sound familiar?
 - We will see special cases of VE later
 - You'll have to implement the special cases

- Approximations
 - Exact inference is slow, especially with a lot of hidden nodes
 - Approximate methods give you a (close, wrong?) answer, faster

Sampling

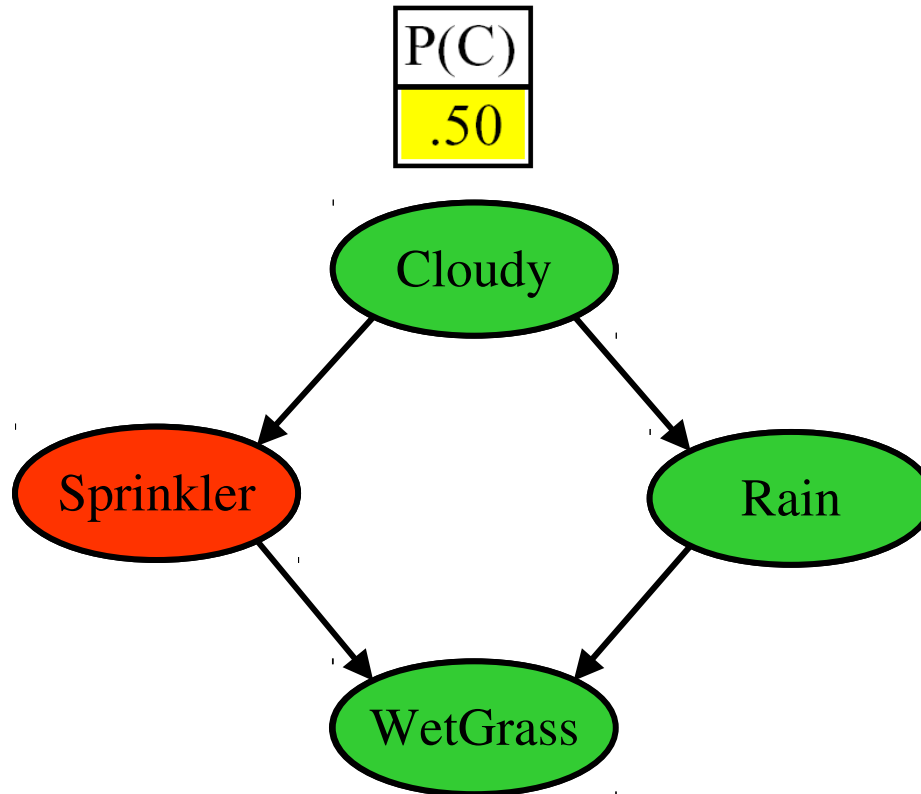
- Basic idea:
 - Draw N samples from a sampling distribution S
 - Compute an approximate posterior probability
 - Show this converges to the true probability P

- Outline:
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples



Prior Sampling

C	P(S C)
T	.10
F	.50



C	P(R C)
T	.80
F	.20

S	R	P(W S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Prior Sampling

- This process generates samples with probability

$$S_{PS}(x_1 \dots x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(X_i)) = P(x_1 \dots x_n)$$

...i.e. the BN's joint probability

- Let the number of samples of an event be

$$N_{PS}(x_1 \dots x_n)$$

- Then

$$\begin{aligned} \lim_{N \rightarrow \infty} \hat{P}(x_1, \dots, x_n) &= \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n) / N \\ &= S_{PS}(x_1, \dots, x_n) \\ &= P(x_1 \dots x_n) \end{aligned}$$

- I.e., the sampling procedure is **consistent**

Example

- We'll get a bunch of samples from the BN:

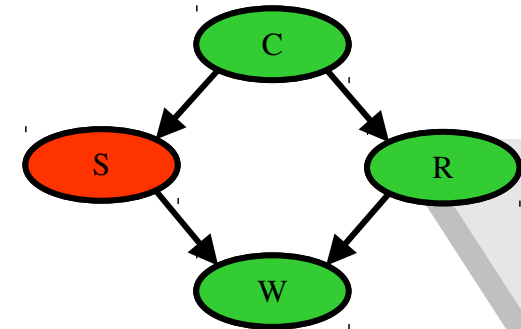
C, \neg S, r, W

C, S, r, W

\neg C, S, r, \neg W

C, \neg S, r, W

\neg C, S, \neg r, W



- If we want to know $P(W)$

- We have counts $\langle w:4, \neg w:1 \rangle$

- Normalize to get $P(W) = \langle w:0.8, \neg w:0.2 \rangle$

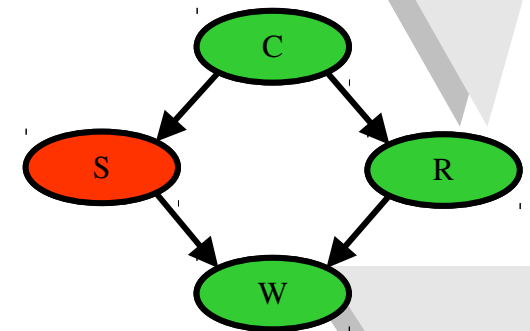
- This will get closer to the true distribution with more samples

- Can estimate anything else, too

- What about $P(C | \neg r)$? $P(C | \neg r, \neg w)$?

Rejection Sampling

- Let's say we want $P(C)$
 - No point keeping all samples around
 - Just tally counts of C outcomes
- Let's say we want $P(C | s)$
 - Same thing: tally C outcomes, but ignore (reject) samples which don't have $S=s$
 - This is rejection sampling
 - It is also consistent (correct in the limit)



$C, \neg S, r, W$

C, S, r, W

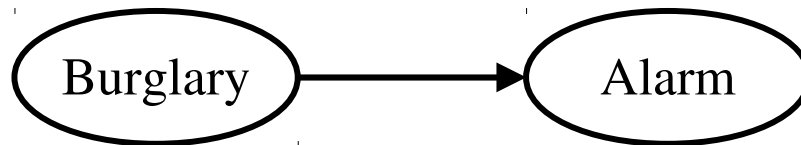
$\neg C, S, r, \neg W$

$C, \neg S, r, W$

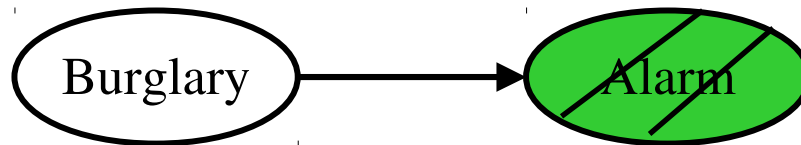
$\neg C, S, \neg r, W$

Likelihood Weighting

- Problem with rejection sampling:
 - If evidence is unlikely, you reject a lot of samples
 - You don't exploit your evidence as you sample
 - Consider $P(B|a)$



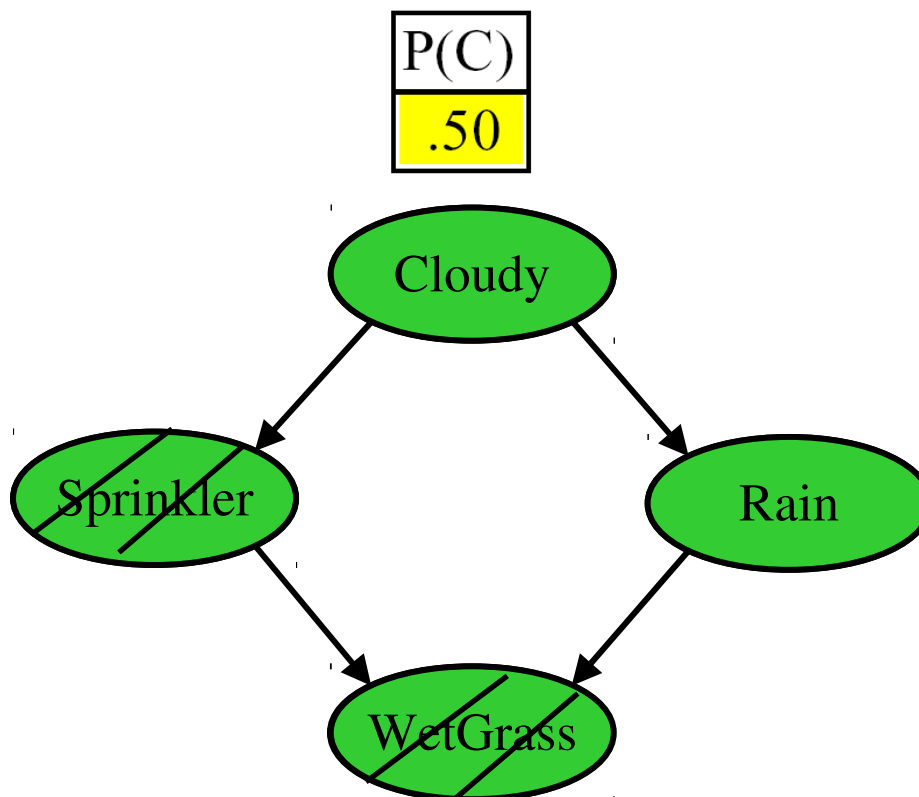
- Idea: fix evidence variables and sample the rest



- Problem: sample distribution not consistent!
- Solution: weight by probability of evidence given parents

Likelihood Sampling

C	P(S C)
T	.10
F	.50



C	P(R C)
T	.80
F	.20

S	R	P(W S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$$W = 1.0 * 0.1 * 0.99$$

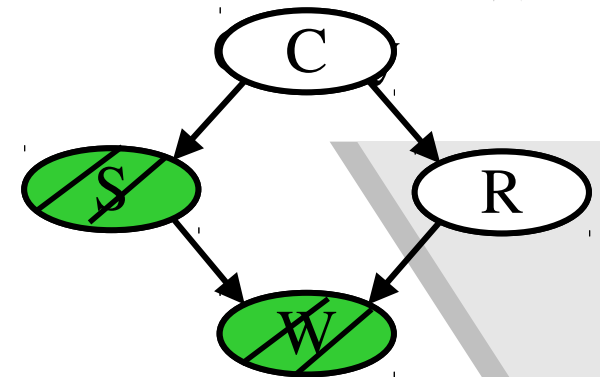
Likelihood Weighting

- Sampling distribution if z sampled and e fixed evidence

$$S_{WS}(z, e) = \prod_{i=1}^l P(z_i | \text{Parents}(Z_i))$$

- Now samples have weights

$$w(z, e) = \prod_{i=1}^m P(e_i | \text{Parents}(E_i))$$

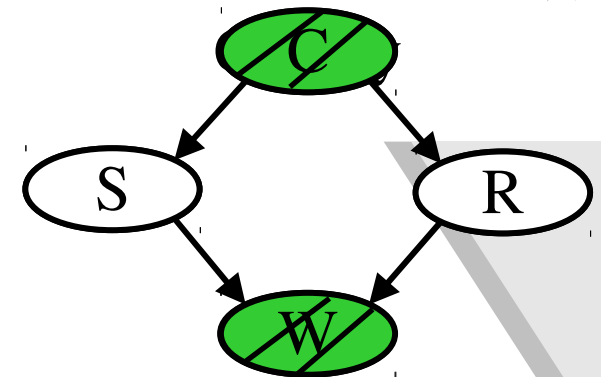


- Together, weighted sampling distribution is

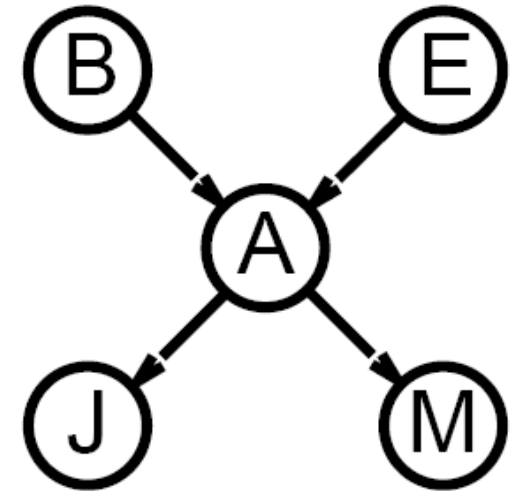
$$\begin{aligned}
 S_{WS}(z, e)w(z, e) &= \prod_{i=1}^m P(e_i | \text{Parents}(E_i)) \prod_{i=1}^m P(e_i | \text{Parents}(E_i)) \\
 &= P(z, e)
 \end{aligned}$$

Likelihood Weighting

- Note that likelihood weighting doesn't solve all our problems
- Rare evidence is taken into account for downstream variables, but not upstream ones
- A better solution is Markov-chain Monte Carlo (MCMC), more advanced
- We'll return to sampling for robot localization and tracking in dynamic BNs



Inference = Integration



- Most inference problems can be cast as an integration problem with respect to the model's probability distribution: p

- In this case, we have $p(B, E, A, J, M) = p(X)$ where $X = \{B, E, A, J, M\}$

Indicator function

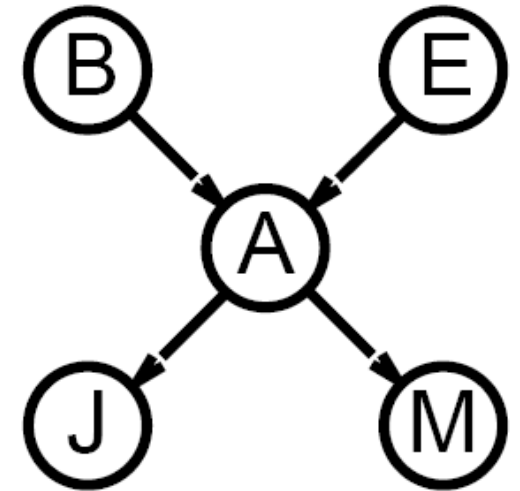
- Suppose we want: $\int dx \ p(x) \mathbf{1}[B=b]$
 - $p(B=b) \implies \int dx \ p(x) \mathbf{1}[B=b] \mathbf{1}[A=\neg a]$
 - $p(B=b, A=\sim a) \implies \int dx \ p(x) \mathbf{1}[B=b] \mathbf{1}[A=\neg a]$
 - $p(B=b|A=\sim a) \implies \frac{\int dx \ p(x) \mathbf{1}[B=b] \mathbf{1}[A=\neg a]}{\int dx \ p(x) \mathbf{1}[A=\neg a]}$

Inference = Integration

- Most inference problems can be cast as an integration problem with respect to the model's probability distribution: p
- In this case, we have $p(B, E, A, J, M) = p(X)$ where $X = \{B, E, A, J, M\}$
- We want to compute something of the form:

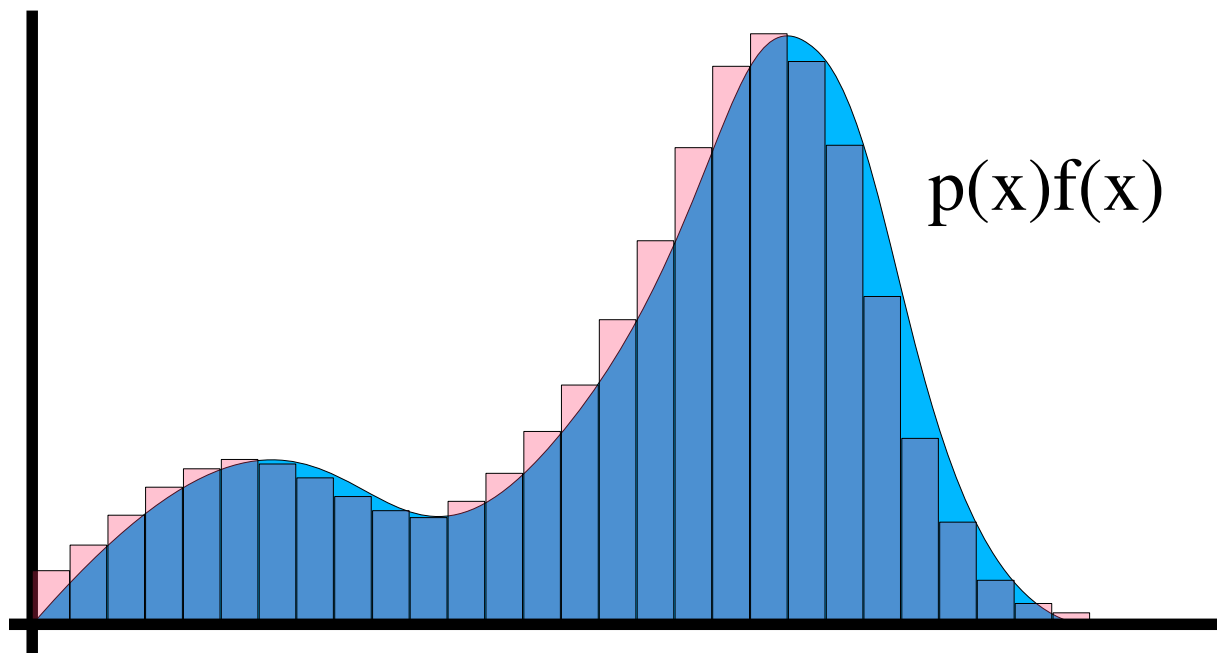
$$F = \int dx \ p(x) f(x) = \mathbf{E}_{x \sim p} [f(x)]$$

Just an expectation!



Integration by Summation

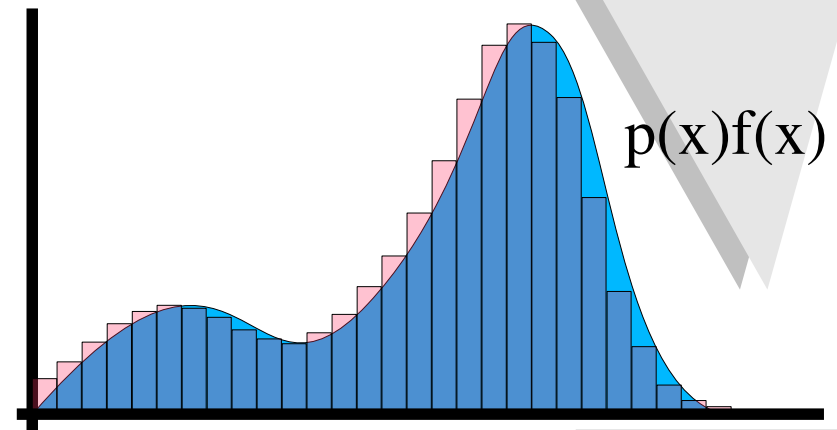
- Remember your 9th grade math:



$$F = \int_X dx p(x)f(x) \approx \frac{1}{R} \sum_{x \in R} p(x)f(x)$$

Integration by Summation

- Pros:
 - Easy to implement
 - Arbitrarily accurate
- Cons:
 - Only works for doubly-bounded regions
 - Intractable for >1 or >2 dimensions
 - Difficult to choose granularity

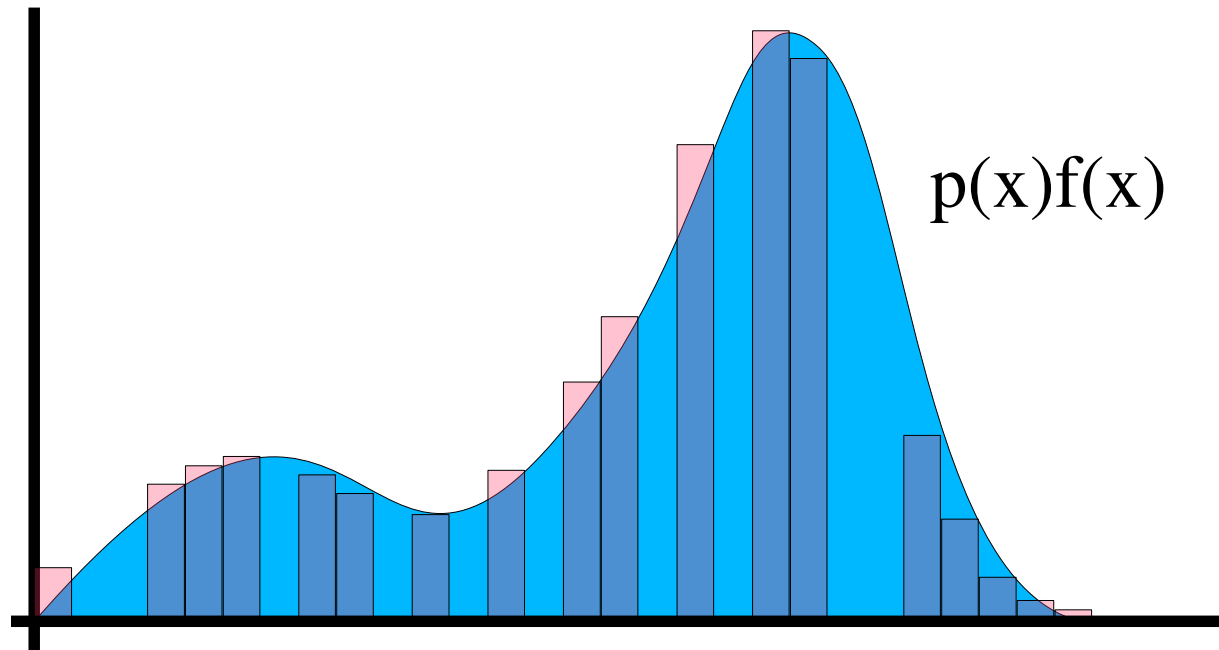


$$F = \int_x dx p(x)f(x) \approx \frac{1}{R} \sum_{x \in R} p(x)f(x)$$

- Idea: let's choose R differently

Monte Carlo Integration

- Uniform sampling:
 - Let R be a (multi)set of points drawn uniformly at random

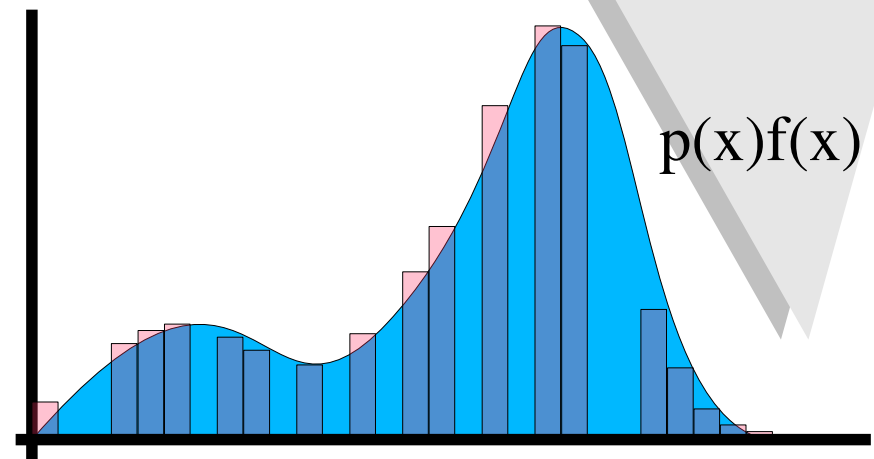


$$F = \int_{\mathcal{X}} dx p(x) f(x) \approx \frac{1}{R} \sum_{x \in R} p(x) f(x)$$

Uniform Sampling

- Pros:
 - Can now work in arbitrarily high dimensions (in theory)
 - Choice is now size of R , not the width of windows

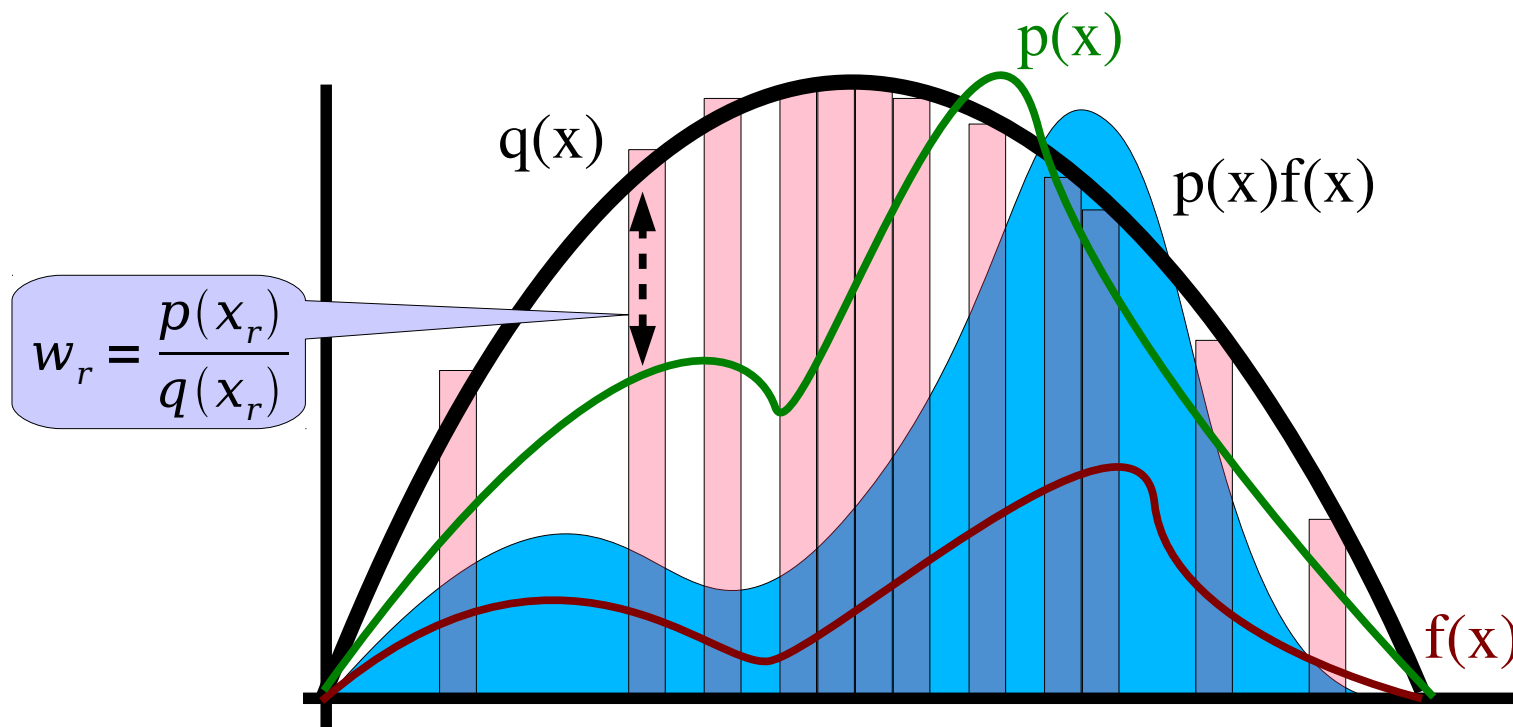
- Cons:
 - Number of samples required to get near the mode of a spiky distribution is huge: $R \sim 2^{D/2}$
 - True distribution is rarely uniform



$$F = \int_x dx p(x) f(x) \approx \frac{1}{R} \sum_{x \in R} p(x) f(x)$$

Importance Sampling

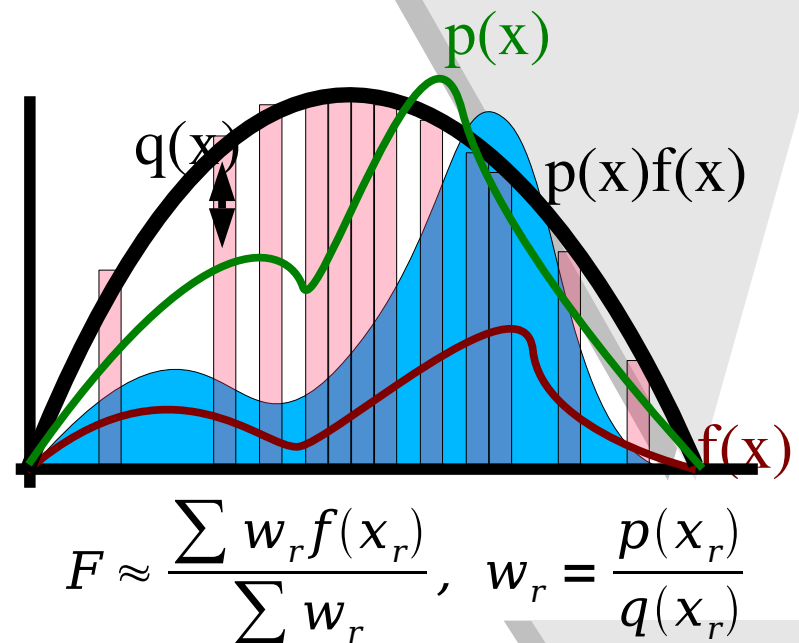
- Let R be a set of points drawn from a proposal distribution q



$$F = \int_X dx p(x) f(x) \approx \frac{\sum_r w_r f(x_r)}{\sum_r w_r}, \quad w_r = \frac{p(x_r)}{q(x_r)}$$

Importance Sampling

- Pros:
 - If q can be constructed similar to p , then good samples can be had
 - Can scale better than uniform sampling (not saying much)
- Cons:
 - Very sensitive to choice of q
 - Hard to evaluate whether it has converged
 - Still a lot of samples required:



IS: $R \sim \exp \sqrt{2D}$

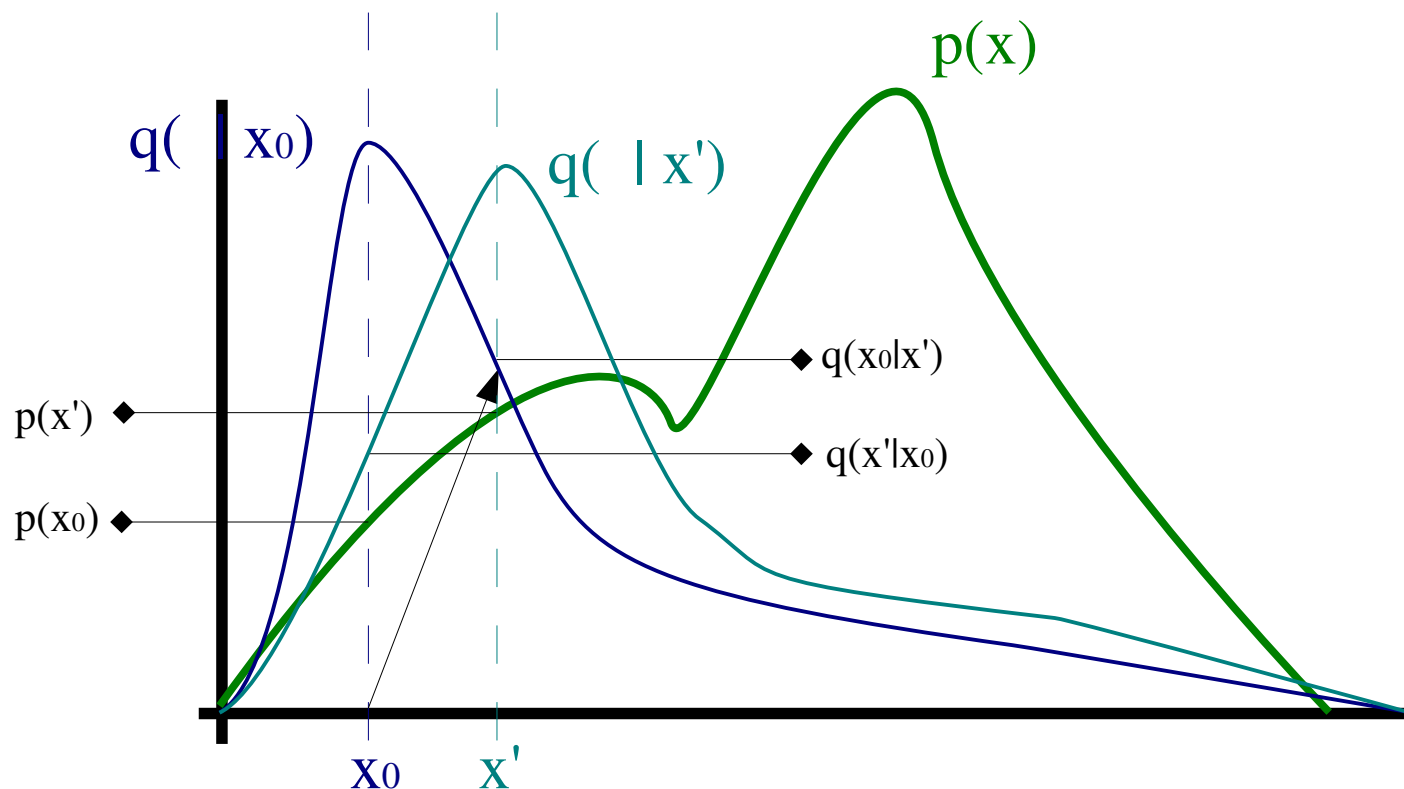
US: $R \sim 2^{D/2}$

Markov Chain Monte Carlo

- Monte Carlo methods suffer because the proposal density needs to be similar to the true density everywhere
- MCMC methods get around this problem by changing the proposal density after each sample
- General framework:
 - Choose a proposal density $q(\cdot | x)$ parameterized by location x
 - Initialize state x arbitrarily
 - Repeatedly sample by:
 - Propose a new state x' from $q(x' | x)$
 - Either accept or reject this new state
 - If accepted, set $x = x'$

Metropolis-Hastings Sampling

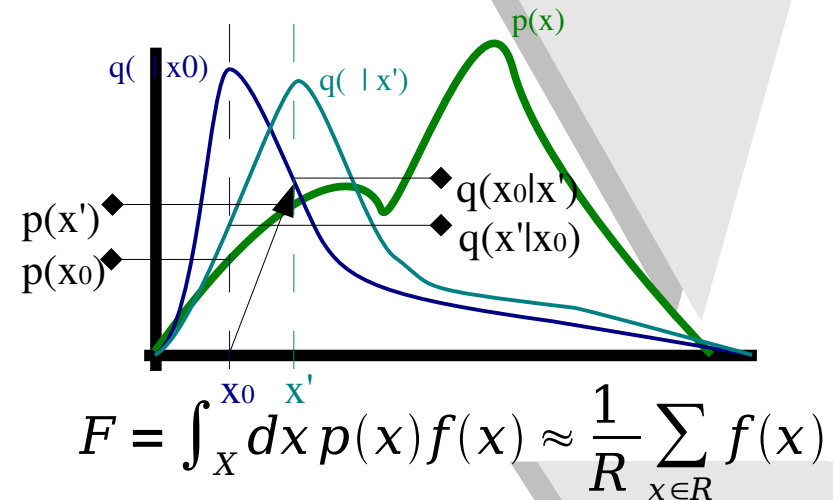
- Accept new states with probability: $\min\left\{1, \frac{p(x')}{p(x)} \frac{q(x|x')}{q(x'|x)}\right\}$



$$F = \int_{\mathcal{X}} dx p(x) f(x) \approx \frac{1}{R} \sum_{x \in R} f(x)$$

Metropolis-Hastings Sampling

- Pros:
 - No longer need to specify a universally good proposal distribution; only locally good
 - Simple proposal distributions can go far



- Cons:
 - Hard to tell how far to space samples:
 - Suppose we use spherical proposals and, then we need at least

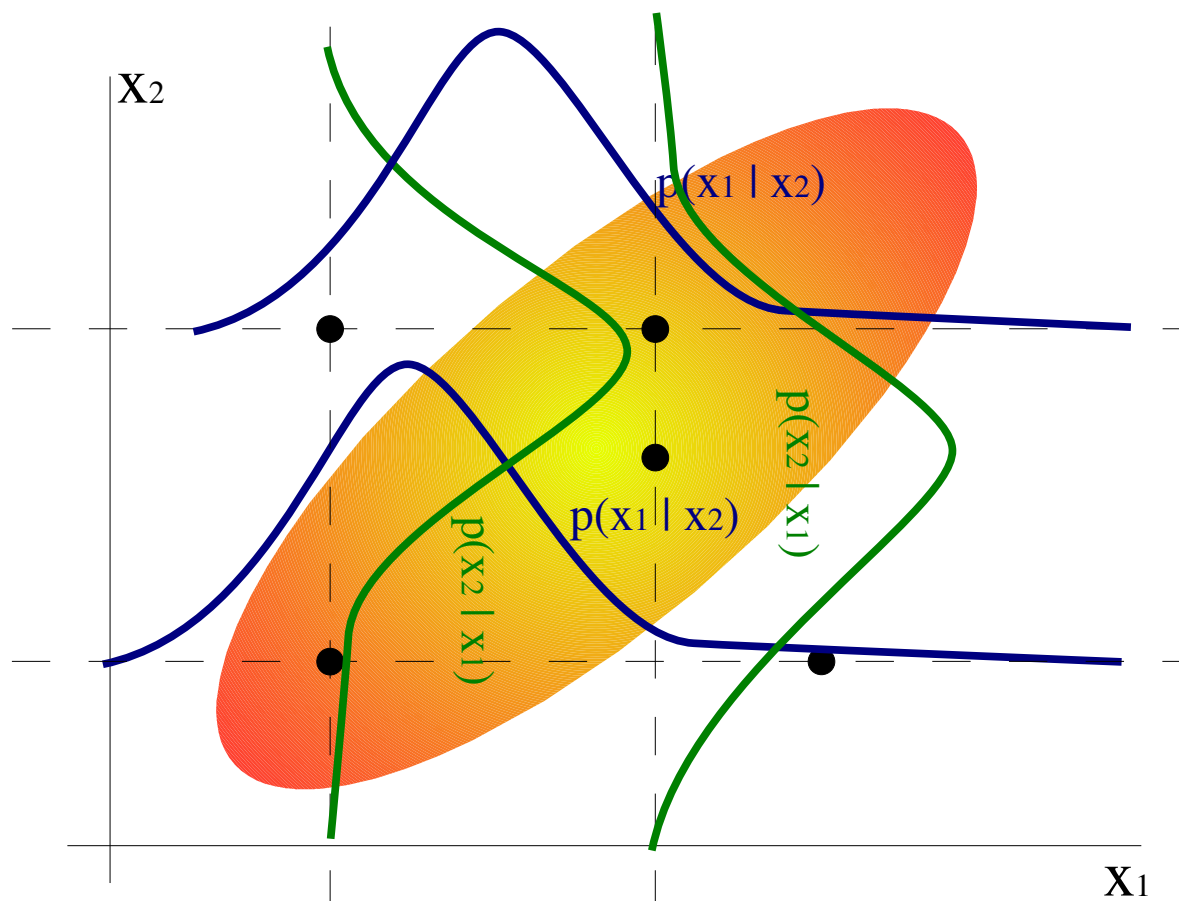
$$N \geq (\sigma_{max} / \sigma_{min})^2$$

where *sigmas* are lengths of the major density in *p*

- Auto-correlation to track dependencies

Gibbs Sampling

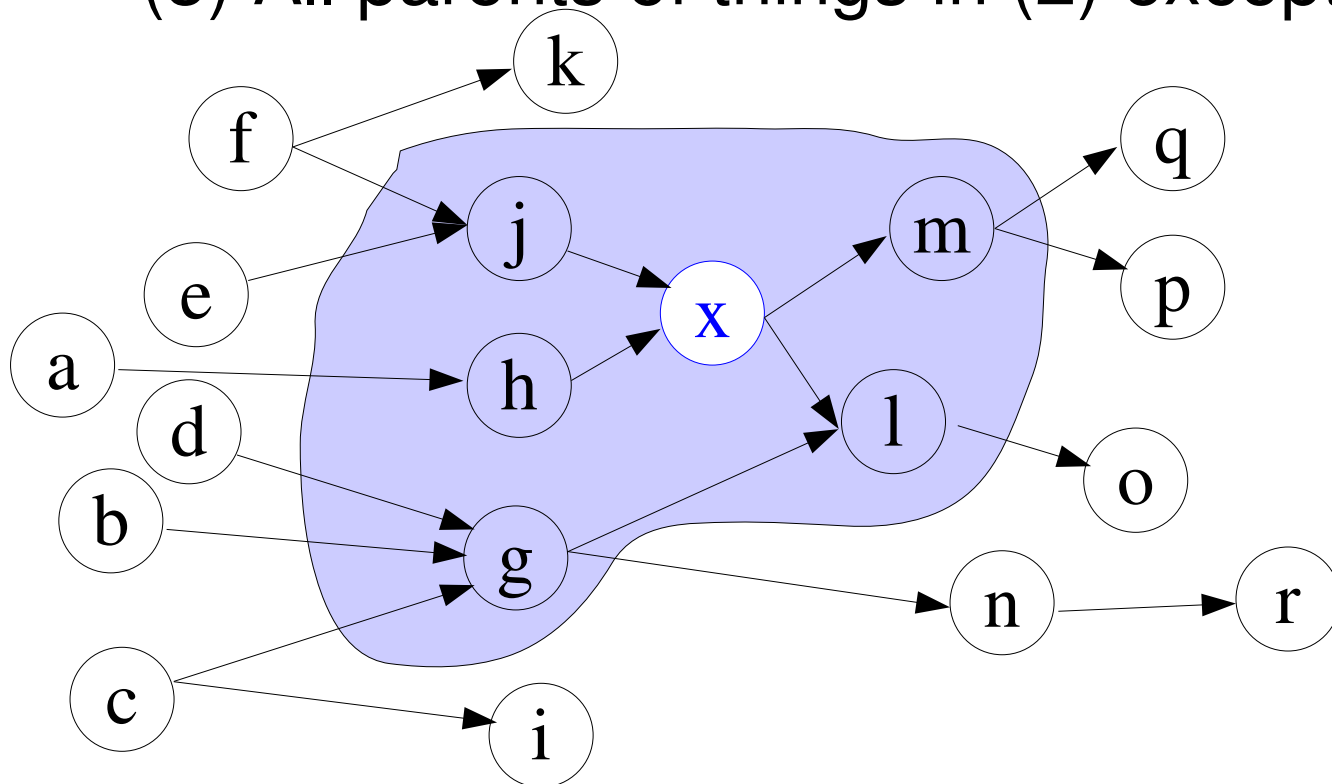
- Defined only for multidimensional problems
- Useful when you can take out one variable and explicitly sample the rest



$$F = \int_{\mathbf{x}} d\mathbf{x} p(\mathbf{x}) f(\mathbf{x}) \approx \frac{1}{R} \sum_{\mathbf{x} \in R} f(\mathbf{x})$$

Markov Blankets

- Given a graphical model and a node x the Markov blanket of x consists of:
 - (1) All parents of x
 - (2) All children of x
 - (3) All parents of things in (2) except x



Gibbs Sampling

- Typically our r.v.s are:

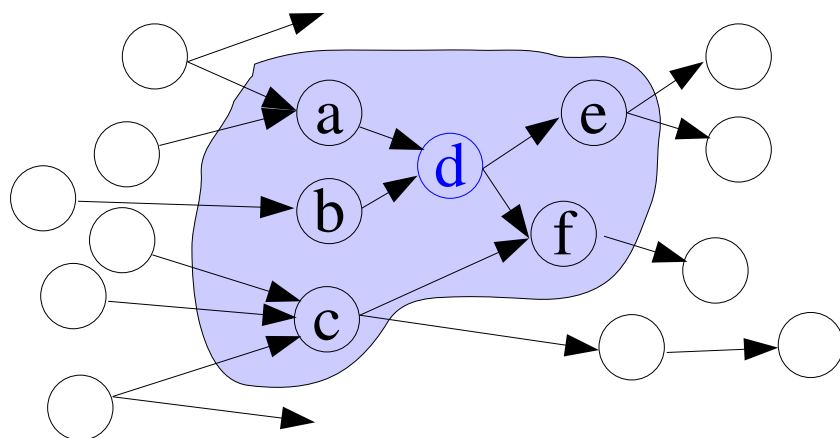
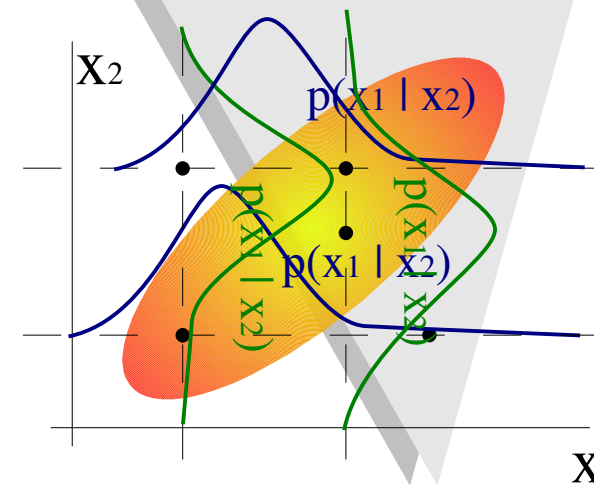
$$\bar{x} = \langle x_1, \dots, x_N \rangle$$

- For each i , we draw a sample from:

$$p(x_i | x_{-i}) = p(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_D)$$

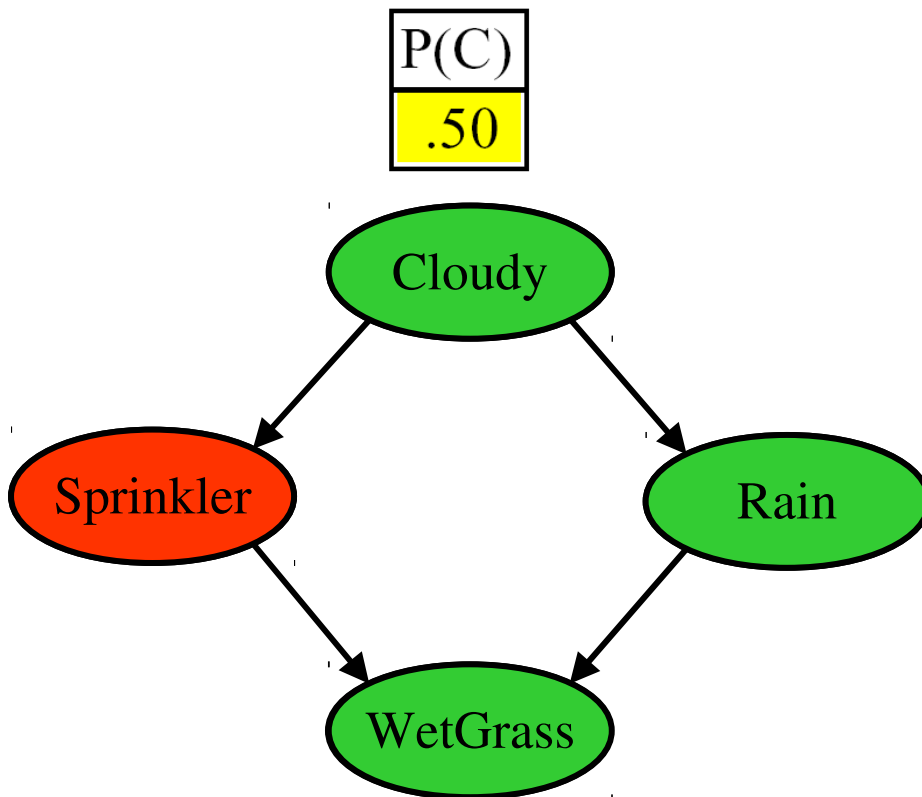
- Only depends on the *Markov blanket*:

$$p(x_i | x_{-i}) = p(x_i | par(x_i)) \prod_{j: x_j \in par(x_i)} p(x_j | par(x_j))$$



Gibbs Sampling

C	P(S C)
T	.10
F	.50



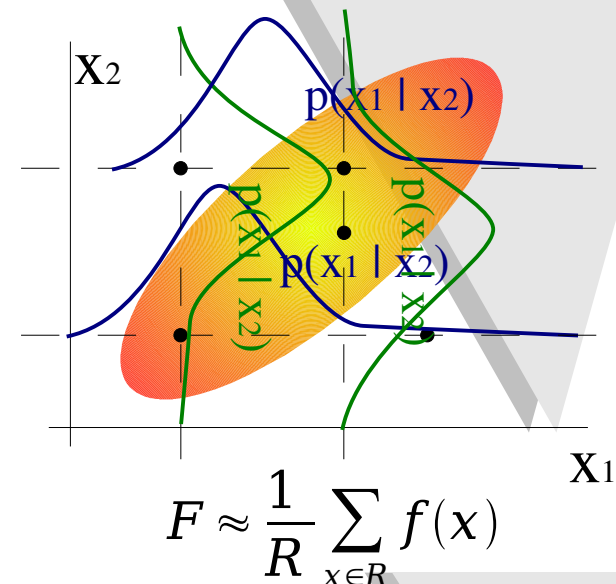
C	P(R C)
T	.80
F	.20

S	R	P(W S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Gibbs Sampling

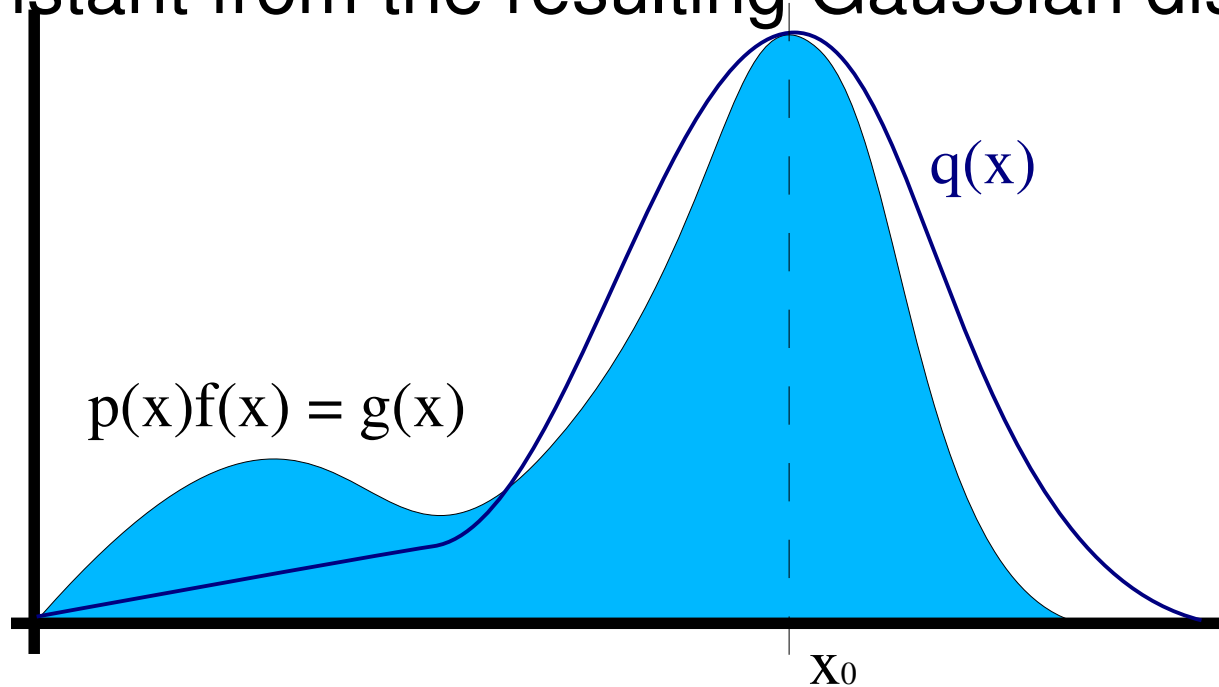
- Pros:
 - Designed to work in high dimensional spaces
 - Terribly simple to implement
 - Automatable

- Cons:
 - Hard to judge convergence, can require many many samples to get an independent one (often worse than MH)
 - Only applicable when conditional distributions are 'nice'
 - (Though there are ways around this)



Laplace (Saddlepoint) Approximation

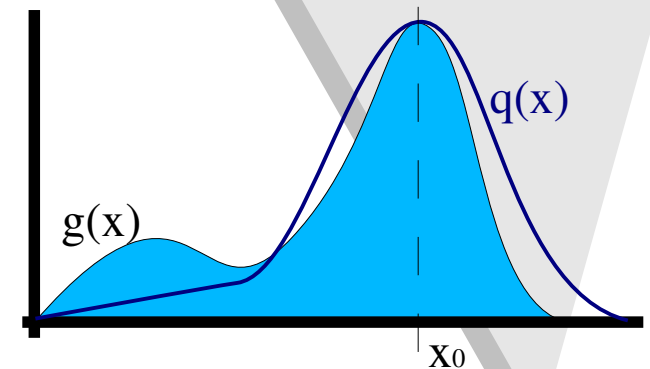
- Idea: approximate the expectation by a quadratic (Taylor expansion) and use the normalizing constant from the resulting Gaussian distribution



$$F = \int_{\mathcal{X}} dx p(x) f(x) \approx g(x_0) \sqrt{\frac{2\pi}{c}} \quad , \quad c = - \left[\frac{\partial^2}{\partial x^2} \ln g(x) \right]_{x=x_0}$$

Laplace Approximation

- Find a mode x_0 of the high-dimensional distribution p
- Approximate $\ln p(x)$ by a Taylor expansion around this mode:



$$F \approx g(x_0) \sqrt{2\pi/c}$$

$$c = -\left[\partial^2 \ln g(x) / \partial x^2\right]_{x=x_0}$$

- $$\ln g(\bar{x}) \approx \ln g(\bar{x}_0) - \frac{1}{2} (\bar{x} - \bar{x}_0)^T \mathbf{A} (\bar{x} - \bar{x}_0)$$
- Compute the matrix \mathbf{A} of second derivatives

$$A_{ij} = -\left[\frac{\partial^2}{\partial x_i \partial x_j} \ln g(\bar{x}) \right]_{\bar{x}=\bar{x}_0}$$

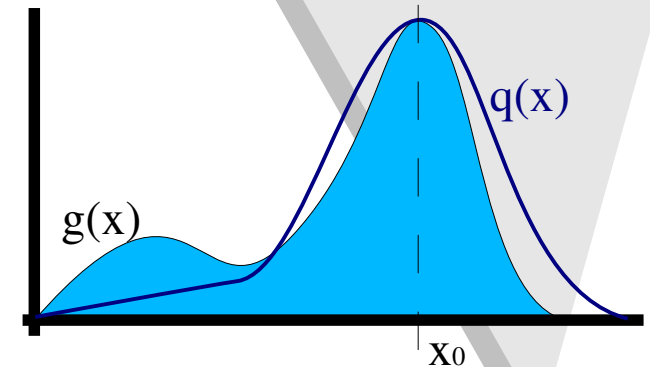
- The exponential form is a Gaussian distribution; use the Gaussian normalizing constant:

$$F = \int_{\mathbb{R}^D} dx g(x) \approx g(\bar{x}_0) \sqrt{\frac{(2\pi)^D}{\det \mathbf{A}}}$$

Laplace Approximation

- Pros:
 - Deterministic
 - Efficient if **A** is of a suitable form (i.e., diagonal or block-diagonal)
 - Can apply transformations to make quadratic approximation more reasonable

- Cons:
 - Poor fit for multimodal distributions
 - Often, $\det \mathbf{A}$ cannot be found efficiently



$$F \approx g(x_0) \sqrt{2\pi/c}$$

$$c = -\left[\partial^2 \ln g(x) / \partial x^2\right]_{x=x_0}$$