Intro to Sampling Methods

CSE586 Computer Vision II
Penn State Univ

Topics to be Covered

Monte Carlo Integration

Sampling and Expected Values

Inverse Transform Sampling (CDF)

Ancestral Sampling

Rejection Sampling

Importance Sampling

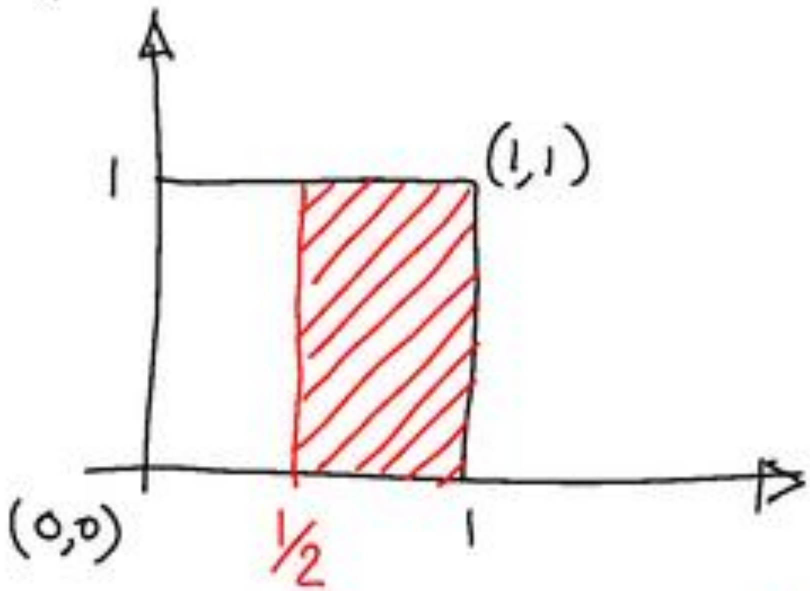
Markov Chain Monte Carlo

Robert Collins
CSE586, PSU

Integration and Area

The idea

What is the probability that a dart thrown uniformly at random will hit the red area?



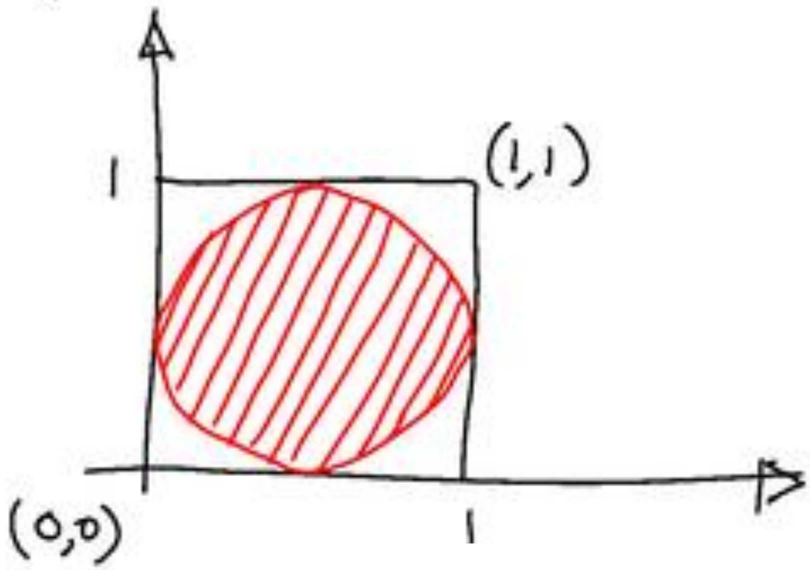
http://videolectures.net/mlss08au_freitas_asm/

Robert Collins
CSE586, PSU

Integration and Area

The idea

What is the probability that a dart thrown uniformly at random will hit the red area?



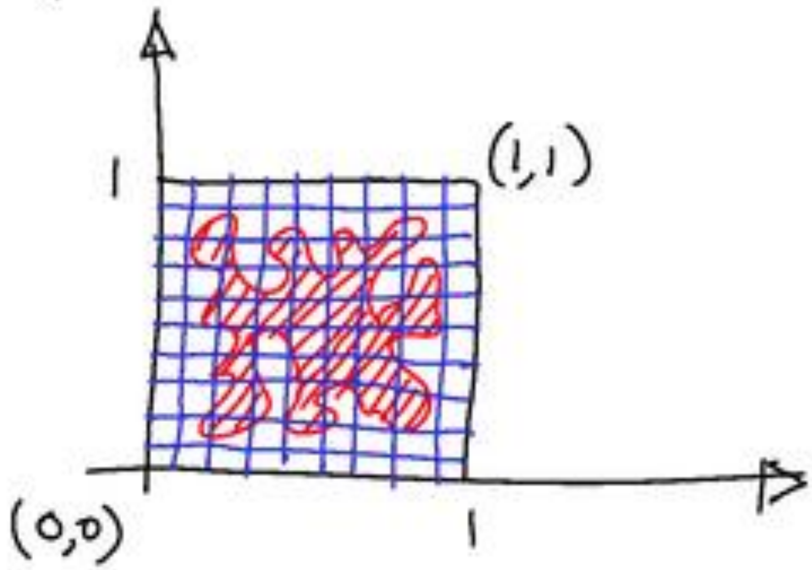
http://videolectures.net/mlss08au_freitas_asm/

Robert Collins
CSE586, PSU

Integration and Area

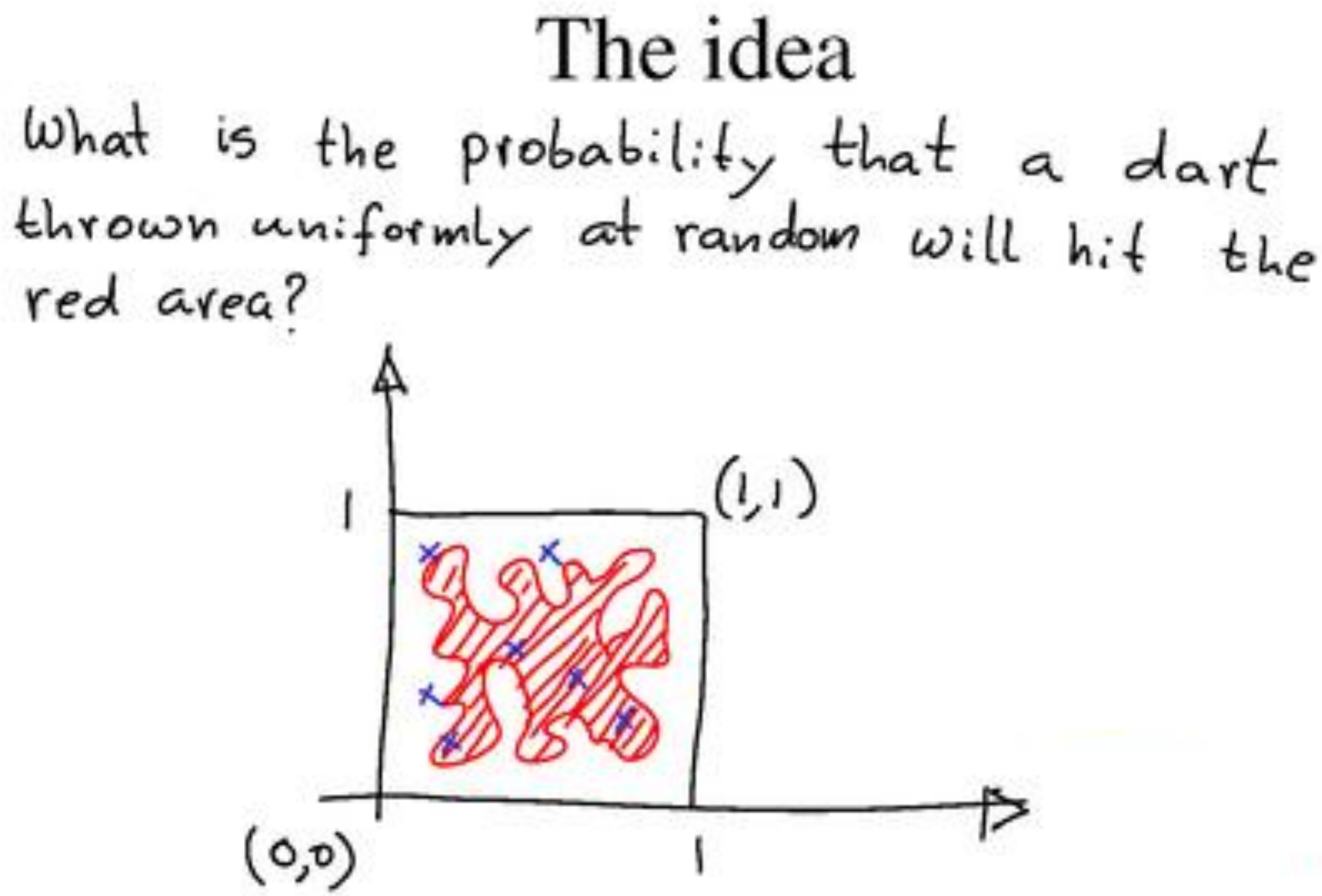
The idea

What is the probability that a dart thrown uniformly at random will hit the red area?



http://videolectures.net/mlss08au_freitas_asm/

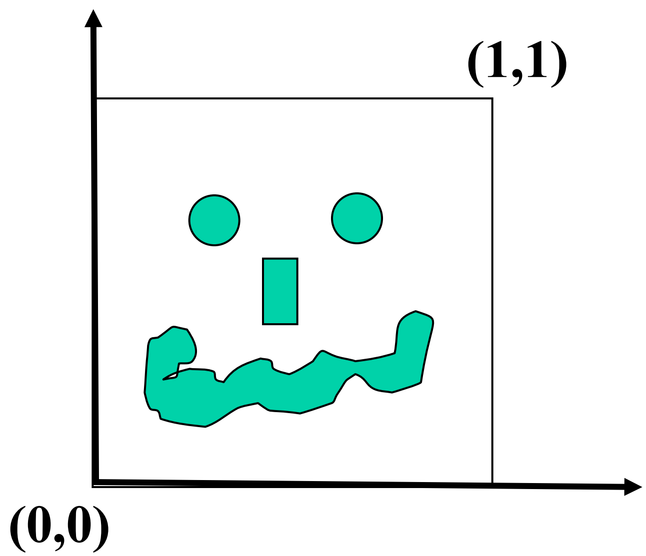
Integration and Area



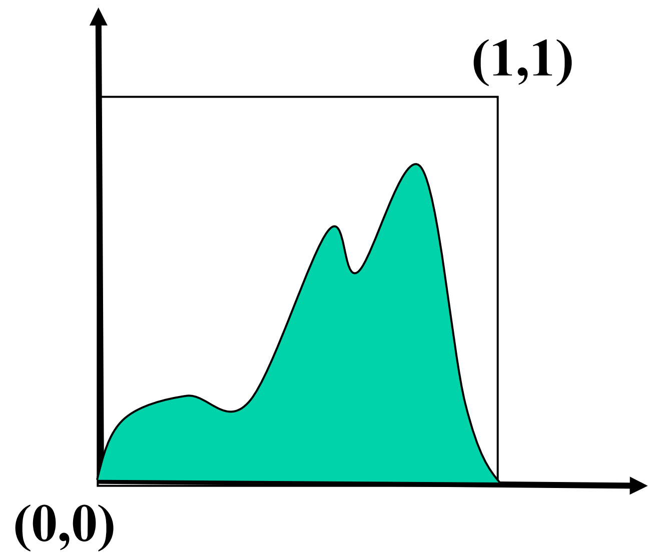
http://videolectures.net/mlss08au_freitas_asm/

Integration and Area

- As we use more samples, our answer should get more and more accurate
- Doesn't matter what the shape looks like



arbitrary region
(even disconnected)

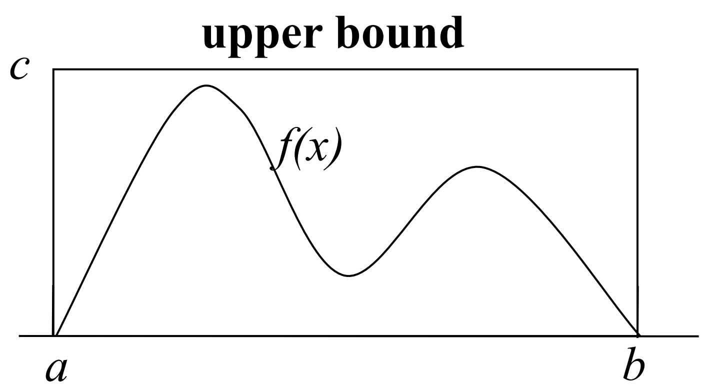


area under curve
aka integration!

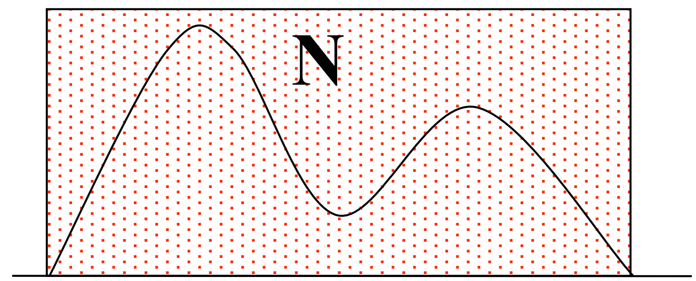
Robert Collins
CSE586, PSU

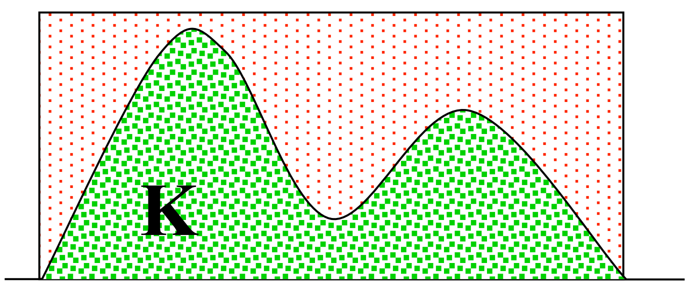
Monte Carlo Integration

Goal: compute definite integral of function f(x) from a to b



Generate N uniform random samples in upper bound volume





count the K samples that fall below the f(x) curve

$$\text{Answer} = \frac{K}{N} * \text{Area of upper bound volume}$$

$$= \frac{K}{N} * (b-a)(c-0)$$

Motivation: Normalizing Constant

Sampling-based integration is useful for computing the normalizing constant that turns an arbitrary non-negative function f(x) into a probability density function p(x).

$$\mathbf{Z} = \int f(x)\, dx$$

Compute this via sampling (Monte Carlo Integration). Then:

$$P(x) = \frac{\mathbf{1}}{\mathbf{Z}} f(x)$$

Note: for complicated, multidimensional functions, this is the ONLY way we can compute this normalizing constant.

Motivation : Expected Values

If we can generate random samples $\mathbf{x}_i$ from a given distribution P(x), then we can estimate expected values of functions under this distribution by summation, rather than integration.

That is, we can approximate:

$$E(f(x)) = \int f(x)P(x)dx$$

by first generating N i.i.d. samples from P(x) and then forming the empirical estimate:

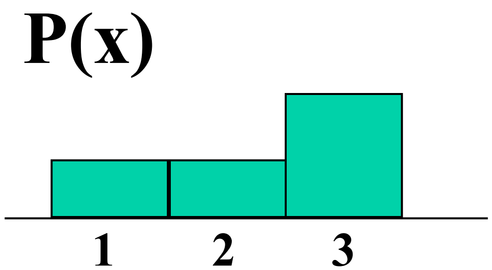
$$\hat{E}(f(x)) = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Robert Collins
CSE586, PSU

Expected Values and Sampling

Example:

a discrete pdf



P(1) = 1/4
P(2) = 1/4
P(3) = 2/4

$$E_P(g(x)) = \sum_{i=1}^{3} g(i)P(i)$$

$$= g(1)\frac{1}{4} + g(2)\frac{1}{4} + g(3)\frac{2}{4}$$

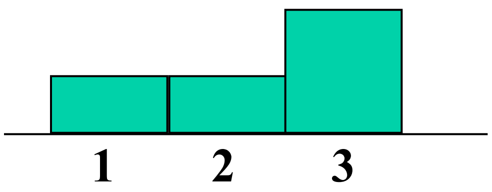
Robert Collins
CSE586, PSU

Expected Values and Sampling (cont)

generate 10 samples from P(x)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	3	3	2	2	3	1	3	3	1

P(x)



$$\hat{E}_P(g(x)) = \frac{1}{10}\sum_{i=1}^{10} g(x_i)$$

$$= \frac{1}{10}[g(1)+g(3)+g(3)+g(2)+g(2)$$

$$+g(3)+g(1)+g(3)+g(3)+g(1)]$$

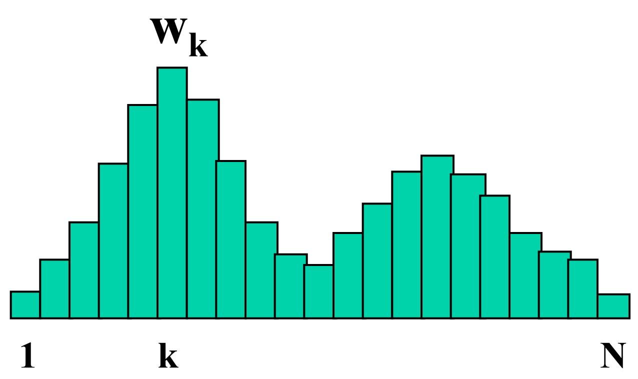
$$= \frac{1}{10}[3g(1)+2g(2)+5g(3)]$$

$$= \frac{3}{10}g(1)+\frac{2}{10}g(2)+\frac{5}{10}g(3) \sim g(1)\frac{1}{4}+g(2)\frac{1}{4}+g(3)\frac{2}{4}$$

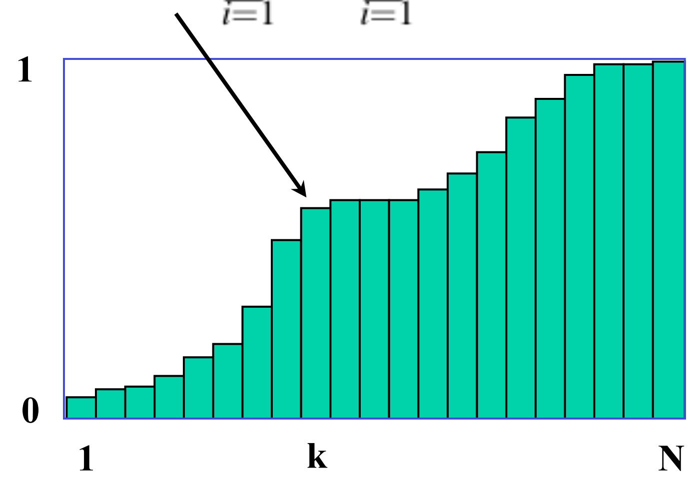
Robert Collins
CSE586, PSU

Inverse Transform Sampling

It is easy to sample from a discrete 1D distribution, using the cumulative distribution function.



$$c(k) = \sum_{i=1}^{k} w_i \Big/ \sum_{i=1}^{N} w_i$$



cumulative distribution function

$$F(x) = P(X \leq x)$$

Inverse Transform Sampling

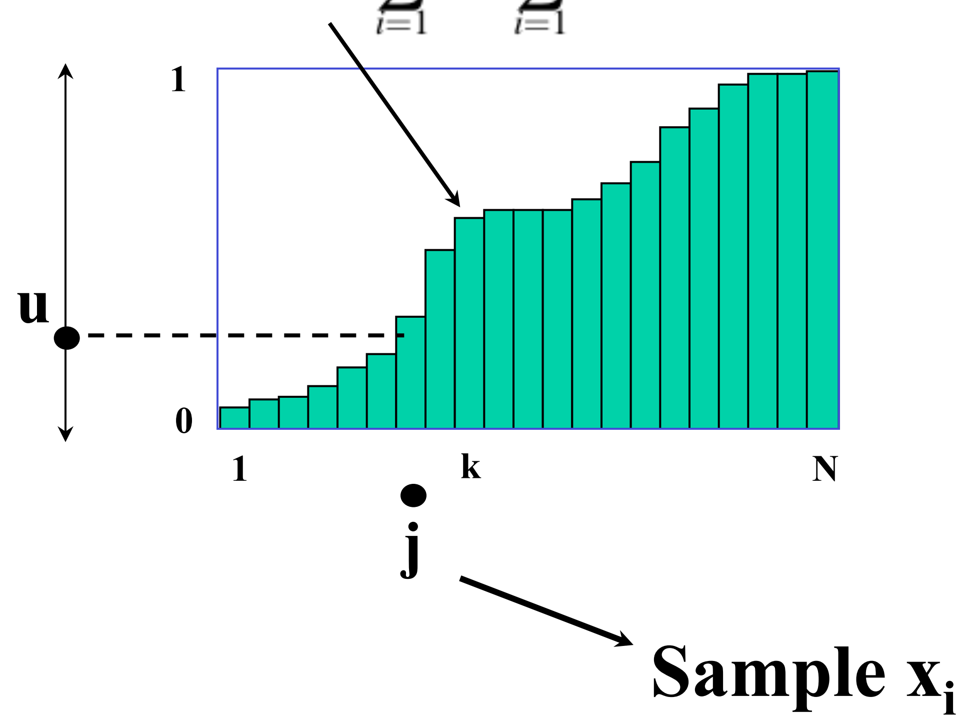
It is easy to sample from a discrete 1D distribution, using the cumulative distribution function.

1) **Generate uniform u in the range [0,1]**

2) **Visualize a horizontal line intersecting bars**

3) **If index of intersected bar is j, output new sample x_i=j**

$$c(k) = \sum_{i=1}^{k} w_i / \sum_{i=1}^{N} w_i$$



Robert Collins
CSE586, PSU

Inverse Transform Sampling

Why it works:

cumulative distribution function

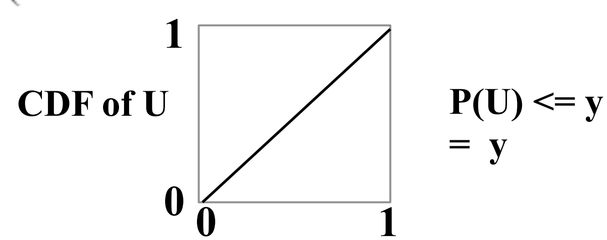
$$F(x) = P(X \leq x)$$

inverse cumulative distribution function

$$F^{-1}(t) = \min\{x : F(x) = t, 0 < t < 1\}$$

Claim: if U is a uniform random variable on (0,1) then X=$F^{-1}(U)$ has distribution function F.
Proof:

$$\begin{aligned}
&P(F^{-1}(U) \leq x) \\
&\quad = \quad P(\min\{x : F(x) = U\} \leq x) && \text{(def of } F^{-1}\text{)} \\
&\quad = \quad P(U \leq F(x)) && \text{(applied } F \text{ to both sides)} \\
&\quad = \quad F(x) && \text{(def of distribution function of } U\text{)}
\end{aligned}$$



Efficient Generating Many Samples
(naive approach is NlogN, but we can do better)

Algorithm 2: Resampling Algorithm

$[\{\mathbf{x}_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}]$ = RESAMPLE $[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$

- Initialize the CDF: $c_1 = 0$
- FOR $i = 2 : N_s$
  - Construct CDF: $c_i = c_{i-1} + w_k^i$
- END FOR
- Start at the bottom of the CDF: $i = 1$
- Draw a starting point: $u_1 \sim \mathbb{U}[0, N_s^{-1}]$
- FOR $j = 1 : N_s$
  - Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  - WHILE $u_j > c_i$
  - * $i = i + 1$
  - END WHILE
  - Assign sample: $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$
  - Assign weight: $w_k^j = N_s^{-1}$
  - Assign parent: $i^j = i$
- END FOR

Basic idea: choose one initial small random number; deterministically sample the rest by "crawling" up the cdf function. This is O(N).

odd property: you generate the "random" numbers in sorted order...

Due to Arulampalam

Efficient Generating Many Samples
(naive approach is NlogN, but we can do better)

Algorithm 2: Resampling Algorithm

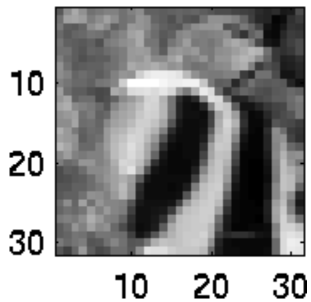
$[\{\mathbf{x}_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}] = \text{RESAMPLE}\ [\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$

- Initialize the CDF: $c_1 = 0$
- FOR $i = 2{:}\ N_s$
  - Construct CDF: $c_i = c_{i-1} + w_k^i$
- END FOR
- Start at the bottom of the CDF: $i = 1$
- Draw a starting point: $u_1 \sim \mathbb{U}[0, N_s^{-1}]$
- FOR $j = 1{:}\ N_s$
  - Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  - WHILE $u_j > c_i$
    * $i = i + 1$
  - END WHILE
  - Assign sample: $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$
  - Assign weight: $w_k^j = N_s^{-1}$
  - Assign parent: $i^j = i$
- END FOR

This approach, called "Systematic Resampling" (Kitagawa '96), is known to produce Monte Carlo estimates with minimum variance (more certainty).

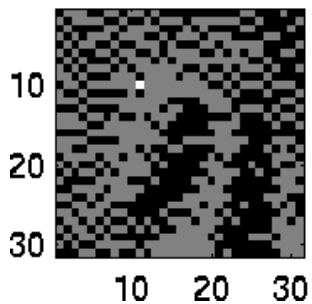
Due to Arulampalam

Example: Sampling from a Weight Image

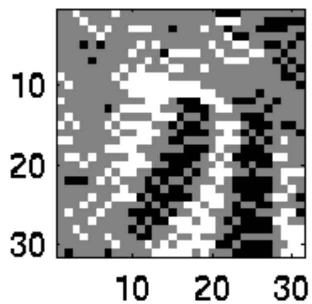


“Likelihood image” to sample from.

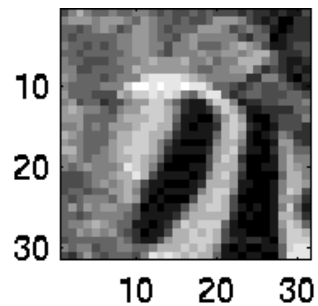
Concatenate values into 1D vector and normalize to form prob mass function . Do systematic resampling. Accumulate histogram of sample values generated and map counts back into the corresponding pixel locations.



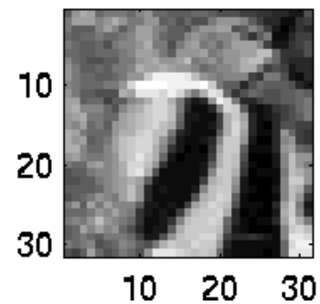
500 samples



1000 samples



5000 samples



10000 samples

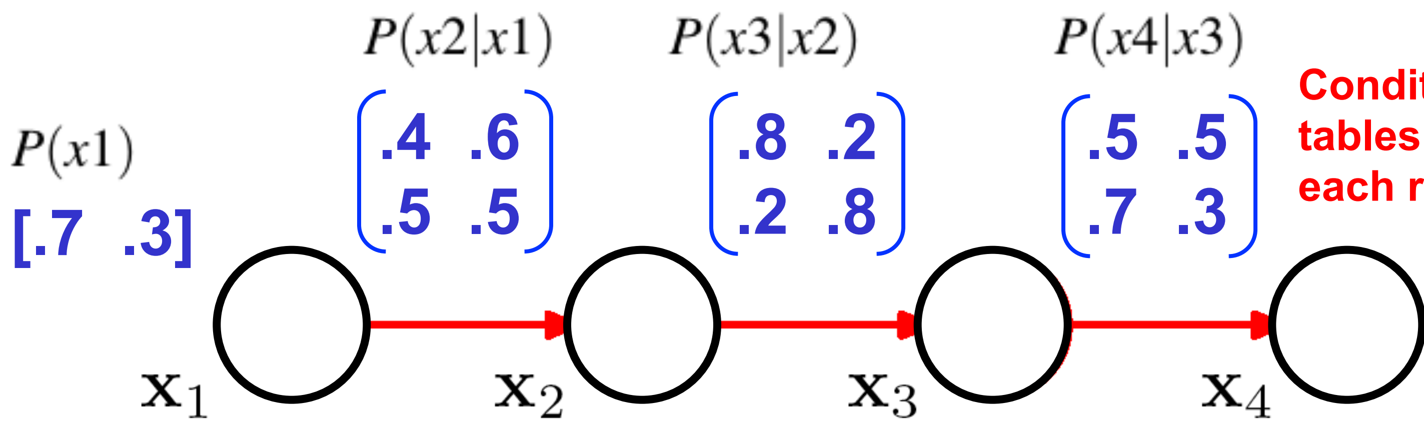
Example: Ancestral Sampling

There are many situations in which we wish to draw samples from a given probability distribution. Although we shall devote the whole of Chapter 11 to a detailed discussion of sampling methods, it is instructive to outline here one technique, called *ancestral sampling*, which is particularly relevant to graphical models. Consider a joint distribution $p(x_1, \ldots, x_K)$ over K variables that factorizes according to (8.5) corresponding to a directed acyclic graph. We shall suppose that the variables have been ordered such that there are no links from any node to any lower numbered node, in other words each node has a higher number than any of its parents. Our goal is to draw a sample $\widehat{x}_1, \ldots, \widehat{x}_K$ from the joint distribution.

To do this, we start with the lowest-numbered node and draw a sample from the distribution $p(x_1)$, which we call $\widehat{x}_1$. We then work through each of the nodes in order, so that for node n we draw a sample from the conditional distribution $p(x_n|\mathrm{pa}_n)$ in which the parent variables have been set to their sampled values. Note that at each stage, these parent values will always be available because they correspond to lower-numbered nodes that have already been sampled. Techniques for sampling from specific distributions will be discussed in detail in Chapter 11. Once we have sampled from the final variable x_K, we will have achieved our objective of obtaining a sample from the joint distribution. To obtain a sample from some marginal distribution corresponding to a subset of the variables, we simply take the sampled values for the required nodes and ignore the sampled values for the remaining nodes. For example, to draw a sample from the distribution $p(x_2, x_4)$, we simply sample from the full joint distribution and then retain the values $\widehat{x}_2, \widehat{x}_4$ and discard the remaining values $\{\widehat{x}_{j \neq 2,4}\}$.

Chris Bishop, *PRML.*

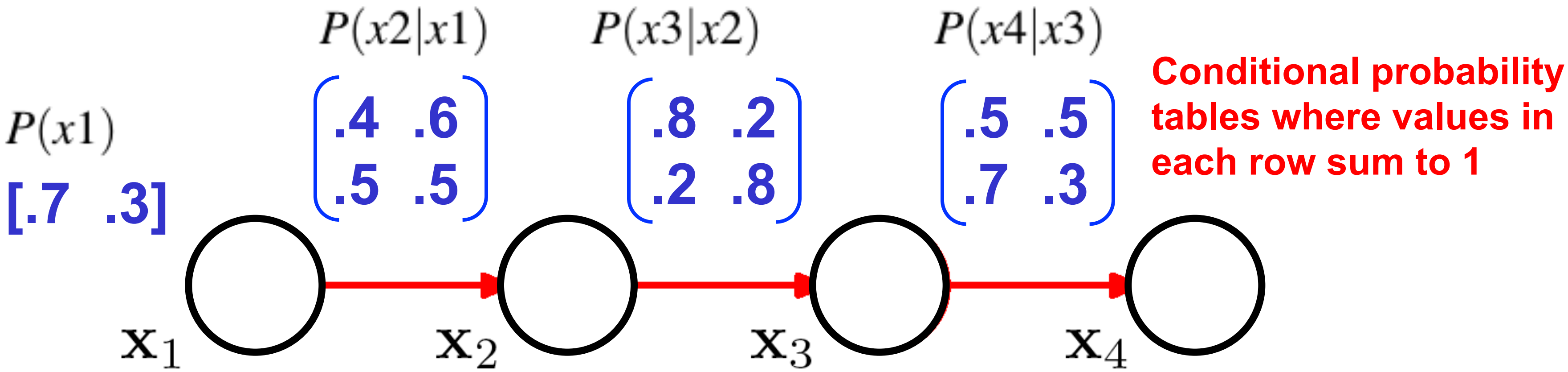
Ancestral Sampling



Conditional probability tables where values in each row sum to 1

$$P(x1,x2,x3,x4) = P(x1)\ P(x2|x1)\ P(x3|x2)\ P(x4|x3)$$

Ancestral Sampling



$$P(x1,x2,x3,x4) = P(x1)\ P(x2|x1)\ P(x3|x2)\ P(x4|x3)$$

To draw a sample from the joint distribution:

- **Start by sampling from P(x1).**
- **Then sample from P(x2|x1).**
- **Then sample from P(x3|x2).**
- **Finally, sample from P(x4|x3).**
- **{x1,x2,x3,x4} is a sample from the joint distribution.**

Ancestral Sampling

```
p1 = [.7 .3];    %marginal on p1 (root node)
p12 = [.4 .6; .5 .5];  %conditional probabilities p(xn|xn-1)
p23 = [.8 .2; .2 .8];  %rows must sum to one!
p34 = [.5 .5; .7 .3];

clear foo
for i=1:10000
    %x1
    x1 = sampleFrom(p1);
    %x2
    x2 = sampleFrom(p12(x1,:));
    %x3
    x3 = sampleFrom(p23(x2,:));
    %x4
    x4 = sampleFrom(p34(x3,:));
    %compute prob
    prob = p1(x1)*p12(x1,x2)*p23(x2,x3)*p34(x3,x4);
    foo(i,:) = [x1 x2 x3 x4 prob];
end
```

Matlab Demo

Robert Collins
CSE586, PSU

Ground Truth (to compare)

Joint Probability, represented in a truth table

x1	x2	x3	x4	P(x1,x2,x3,x4)
1.0000	1.0000	1.0000	1.0000	0.1120
1.0000	1.0000	1.0000	2.0000	0.1120
1.0000	1.0000	2.0000	1.0000	0.0392
1.0000	1.0000	2.0000	2.0000	0.0168
1.0000	2.0000	1.0000	1.0000	0.0420
1.0000	2.0000	1.0000	2.0000	0.0420
1.0000	2.0000	2.0000	1.0000	0.2352 ← MAP
1.0000	2.0000	2.0000	2.0000	0.1008
2.0000	1.0000	1.0000	1.0000	0.0600
2.0000	1.0000	1.0000	2.0000	0.0600
2.0000	1.0000	2.0000	1.0000	0.0210
2.0000	1.0000	2.0000	2.0000	0.0090
2.0000	2.0000	1.0000	1.0000	0.0150
2.0000	2.0000	1.0000	2.0000	0.0150
2.0000	2.0000	2.0000	1.0000	0.0840
2.0000	2.0000	2.0000	2.0000	0.0360

```
marginal x1:    0.700000    0.300000
marginal x2:    0.430000    0.570000
marginal x3:    0.458000    0.542000
marginal x4:    0.608400    0.391600
```

Marginals

A Brief Overview of Sampling

Inverse Transform Sampling (CDF)

Rejection Sampling

Importance Sampling

For these two, we can sample from an unnormalized distribution function.

That is, to sample from distribution P, we only need to know a function P*, where P = P* / c , for some normalization constant c.

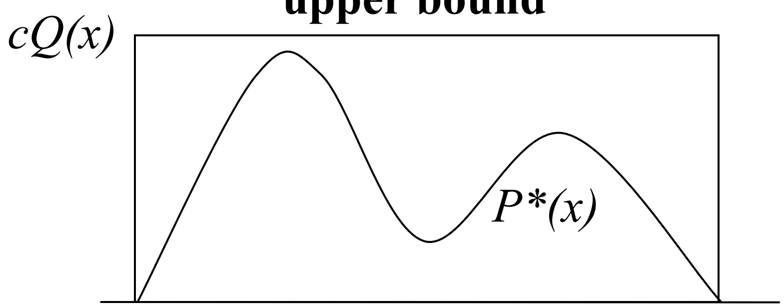
Robert Collins
CSE586, PSU

Rejection Sampling

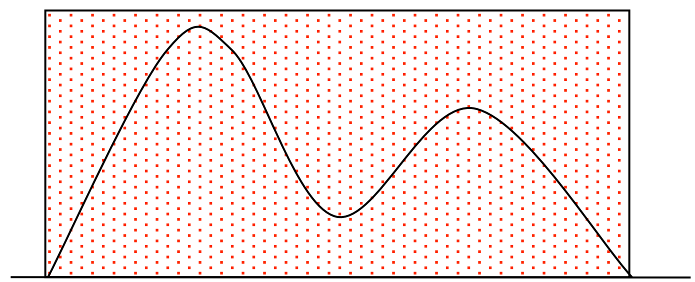
Need a proposal density Q(x) [e.g. uniform or Gaussian], and a constant c such that c(Qx) is an <u>upper bound</u> for P*(x)

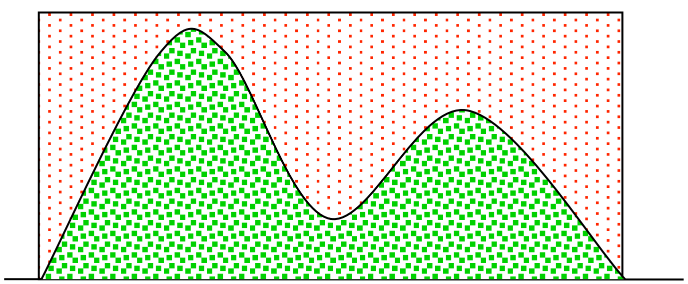
Example with Q(x) uniform

upper bound



generate uniform random samples in upper bound volume





accept samples that fall below the P*(x) curve

the marginal density of the x coordinates of the points is then proportional to P*(x)

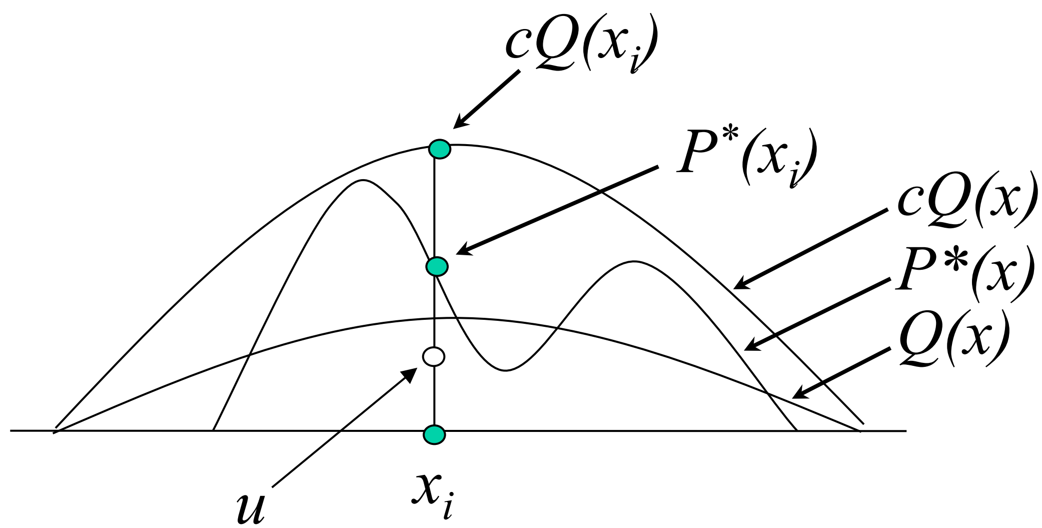
Note the relationship to Monte Carlo integration.

Robert Collins
CSE586, PSU

Rejection Sampling

More generally:

1) generate sample x_i from a proposal density Q(x)
2) generate sample u from uniform [0,cQ(x_i)]
3) if u <= P*(x_i) accept x_i; else reject



accept!

Importance "Sampling"

Not for generating samples. It is a method to estimate the expected value of a function $f(x_i)$ directly

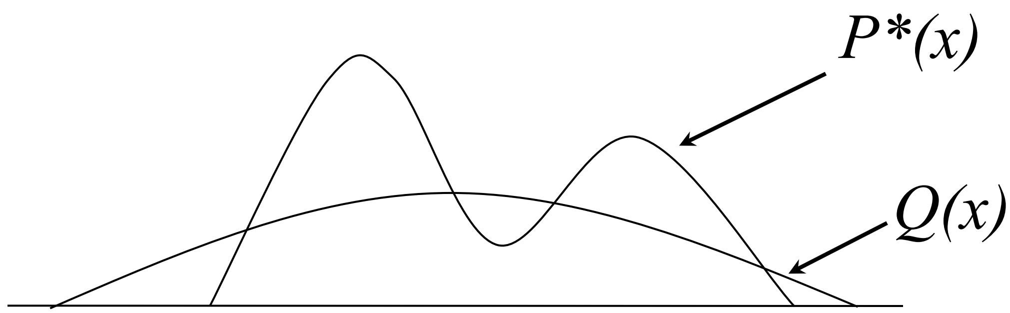
1) Generate x_i from Q(x)

2) an empirical estimate of $E_Q(f(x))$, the expected value of f(x) under distribution Q(x), is then

$$\hat{E}_Q(f(x)) = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

3) However, we want $E_P(f(x))$, which is the expected value of f(x) under distribution $P(x) = P^*(x)/Z$

Importance Sampling



When we generate from Q(x), values of x where Q(x) is greater than P*(x) are overrepresented, and values where Q(x) is less than P*(x) are underrepresented.

To mitigate this effect, introduce a weighting term

$$w_i = \frac{P^*(x_i)}{Q(x_i)}$$

Importance Sampling

New procedure to estimate $E_P(f(x))$:

1) Generate N samples x_i from Q(x)

2) form importance weights

$$w_i = \frac{P^*(x_i)}{Q(x_i)}$$

3) compute empirical estimate of $E_P(f(x))$, the expected value of f(x) under distribution P(x), as

$$\hat{E}_P(f(x)) = \frac{\sum w_i f(x_i)}{\sum w_i}$$

Resampling

Note: We thus have a set of weighted samples (x_i, w_i | i=1,…,N)

If we really need random samples from P, we can generate them by resampling such that the likelihood of choosing value x_i is proportional to its weight w_i

This would now involve now sampling from a discrete distribution of N possible values (the N values of x_i)

Therefore, regardless of the dimensionality of vector x, we are resampling from a 1D distribution (we are essentially sampling from the indices 1...N, in proportion to the importance weights w_i). So we can using the inverse transform sampling method we discussed earlier.

Note on Proposal Functions

Computational efficiency is best if the proposal distribution looks a lot like the desired distribution (area between curves is small).

These methods can fail badly when the proposal distribution has 0 density in a region where the desired distribution has non-negligeable density.

For this last reason, it is said that the proposal distribution should have <u>heavy tails.</u>

Sequential Monte Carlo Methods

Sequential Importance Sampling (SIS) and the closely related algorithm Sampling Importance Sampling (SIR) are known by various names in the literature:

- bootstrap filtering
- particle filtering
- Condensation algorithm
- survival of the fittest

General idea: Importance sampling on time series data, with samples and weights updated as each new data term is observed. Well-suited for simulating Markov chains and HMMs!

Problem

Sampling in High-dimensional Spaces

Standard methods fail:

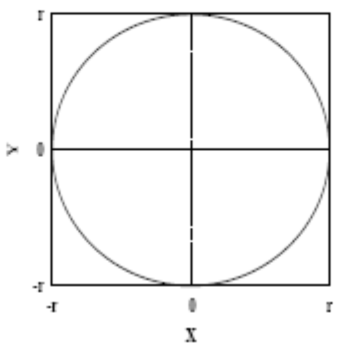
- Rejection Sampling
  - Rejection rate increase with N -> 100%
- Importance Sampling
  - Same problem: vast majority weights -> 0

Intuition: In high dimension problems, the "Typical Set" (volume of nonnegligable prob in state space) is a small fraction of the total space.

High-Dimensional Spaces

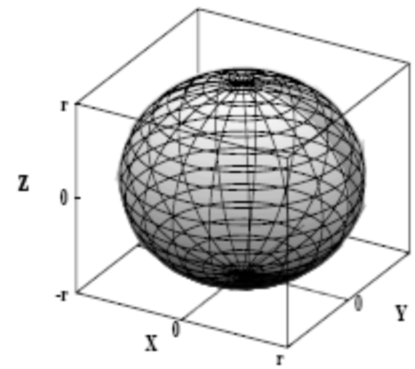
consider ratio of volumes of hypersphere inscribed inside hypercube

2D



$$\frac{\mathbb{V}(S_2(r))}{\mathbb{V}(H_2(2r))} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \approx 75\%$$

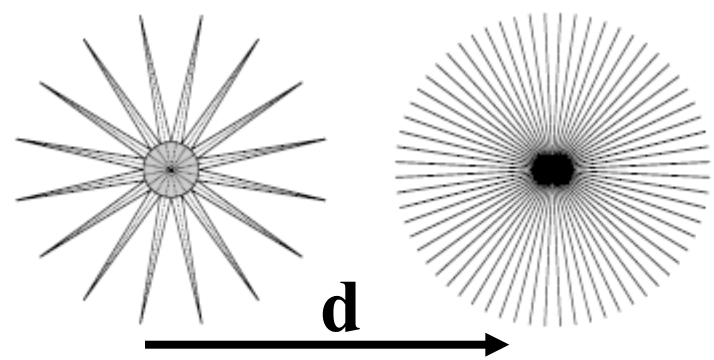
3D



$$\frac{\mathbb{V}(S_3(r))}{\mathbb{V}(H_3(2r))} = \frac{\frac{4}{3}\pi r^3}{8r^3} = \frac{\pi}{6} \approx 50\%$$

Asymptotic behavior:

$$\boxed{\lim_{d\to\infty} \frac{\mathbb{V}(S_d(r))}{\mathbb{V}(H_d(2r))} = \lim_{d\to\infty} \frac{\pi^{d/2}}{2^d\Gamma(\frac{d}{2}+1)} \to 0}$$

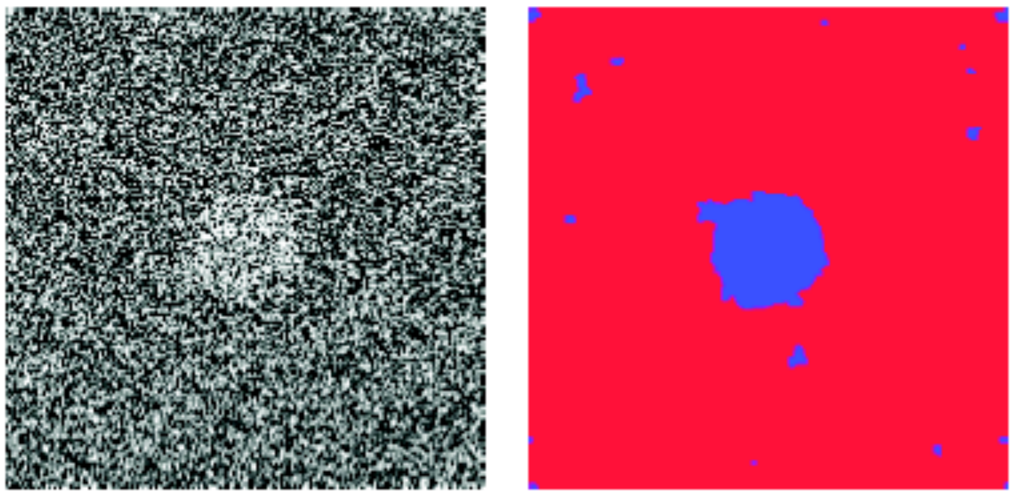


most of volume of the hypercube lies outside of hypersphere as dimension d increases

http://www.cs.rpi.edu/~zaki/Courses/dmcourse/Fall09/notes/highdim.pdf

High Dimensional Spaces
Segmentation Example

- Binary Segmentation of image



each pixel has two states: on and off

Probability of a Segmentation

- Very high-dimensional
- 256*256 pixels = 65536 pixels
- Dimension of state space N = 65536 !!!!

- # binary segmentations = finite , **but...**
- $2^{65536} = 2*10^{19728} >> 10^{79}$ = atoms in universe

Representation P(Segmentation)

- Histogram ? No !
- Assume pixels independent ?

$$P(x_1x_2x_2\ldots)=P(x_1)P(x_2)P(x_3)\ldots$$

ignores neighborhood structure of pixel lattice and empirical evidence that images are "smooth"

- Approximate solution: samples !!!

Brilliant Idea!

Nick Metropolis

- Published June 1953
- Top 10 algorithm !

- Set up a Markov chain
- Run the chain until stationary
- All subsequent samples are from stationary distribution