FIAS Frankfurt Institute for Advanced Studies

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

www.goethe-universitaet.de

Systems Engineering Meets Life Sciences: (Sampling: Monte-Carlo, MCMC)

Prepared by:
Prof. Dr. Visvanathan Ramesh

References and Sources: A. Barbu and S. C. Zhu (book draft, lectures), A. Owen (tutorial), ML lectures (A. Singh, CMU), R. Collins (PSU)

Outline of Lectures so far:

- **Recap – Greiffenhagen Thesis / Systems Engineering Methodology**
- **Model-Based Recognition Overview (Mann, 1996, Dissertation)**
- **What is Context ? (Slides based on Derek Hoeim)**
- **Link to Systems Engineering Methodology**
- **Simulation for Cognitive Vision (Subbu Veerasavarappu)**
- **Modeling:**
  - Compositionality (Based on Slides from Borenstein et al, Stuart Geman)
  - Compositional Models (P. Felzenswalb)
  - Pattern Grammars Introduction (Song-Chun Zhu, Mumford)

- Goal of last lecture
  - Sampling as a mechanism for computation of functions of posteriors in high dimensions (vision examples from S.C. Zhu)
    - Introduce Monte Carlo Methods (based on Owen lecture)
    - Introduce SMC (based on Vision examples)

Goal for Today:

- **Recap from last week**
- **Introduction to MCMC**
- **Decide on Project in order to focus next lectures on project related work**

MCMC Intro Slides

Background: choices of modeling & computing paradigms

- Approximate modeling + Exact computing (e.g. Dynamic programming)
- Exact modeling + Local computing (e.g. Gradient descent)
- Exact modeling + Global computing (MCMC)

Approximate model: you simplify the model, such as removing some edges in a graph to make it a tree or a chain, and thus removing certain energy terms.

Local computing: you may only find a local minimum (or maximum) and rely on heuristics to find a “good” one. Unfortunately most of the interesting function, like in deep learning, has astronomic number of local minima !

Gentler Introduction to Sampling: (Source: Robert Collins)

Monte Carlo Integration

Sampling and Expected Values

Inverse Transform Sampling (CDF)

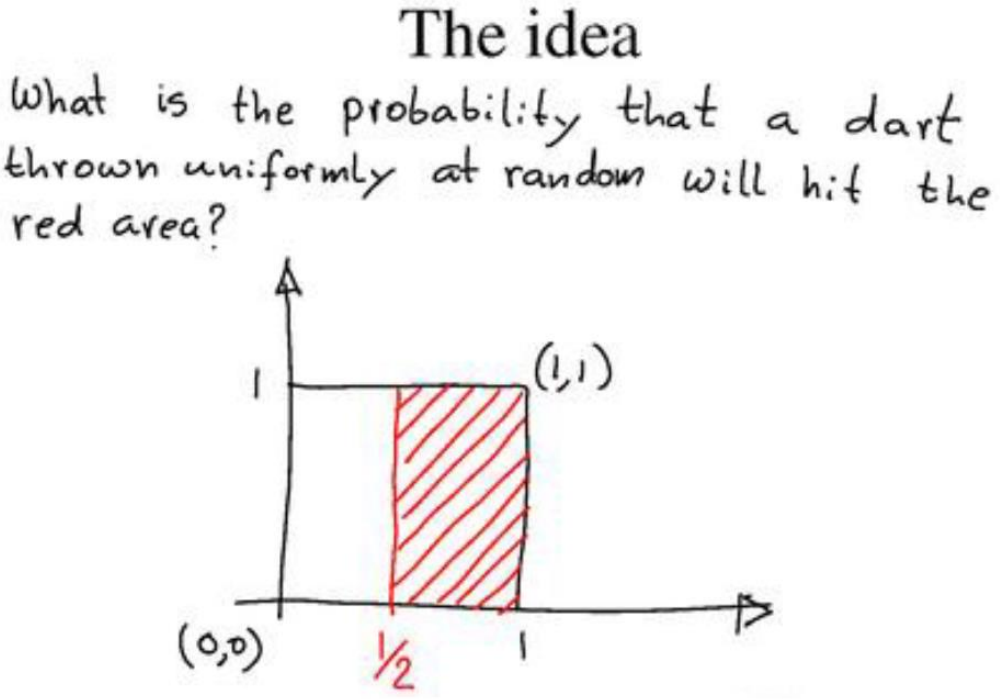
Ancestral Sampling

Rejection Sampling

Importance Sampling

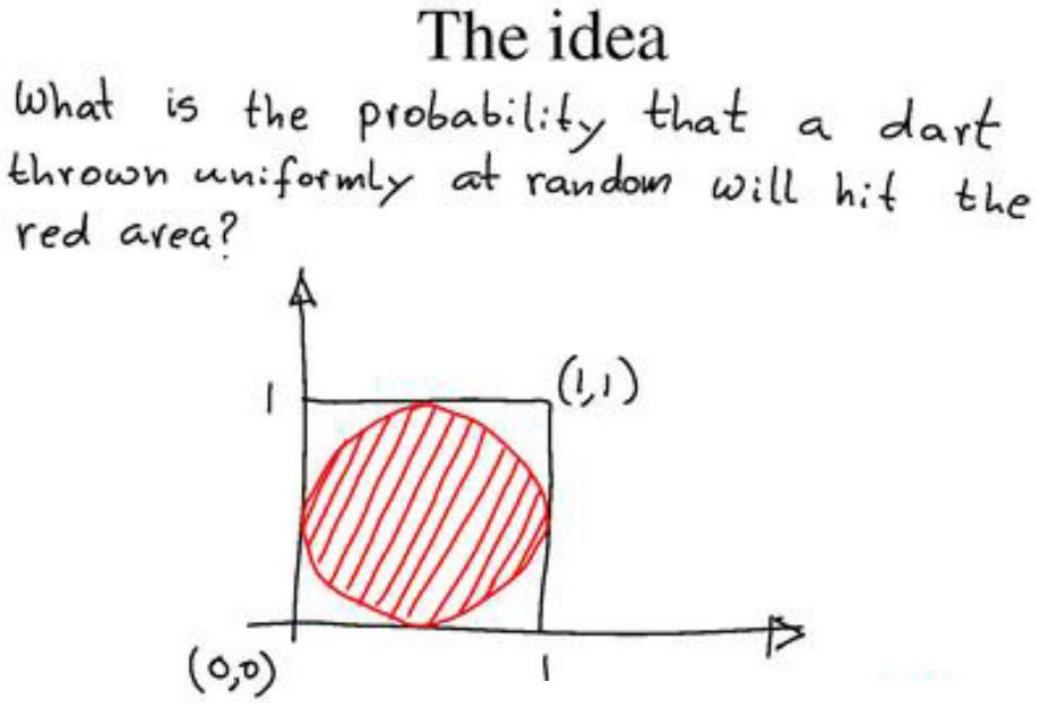
Markov Chain Monte Carlo

Integration and Area



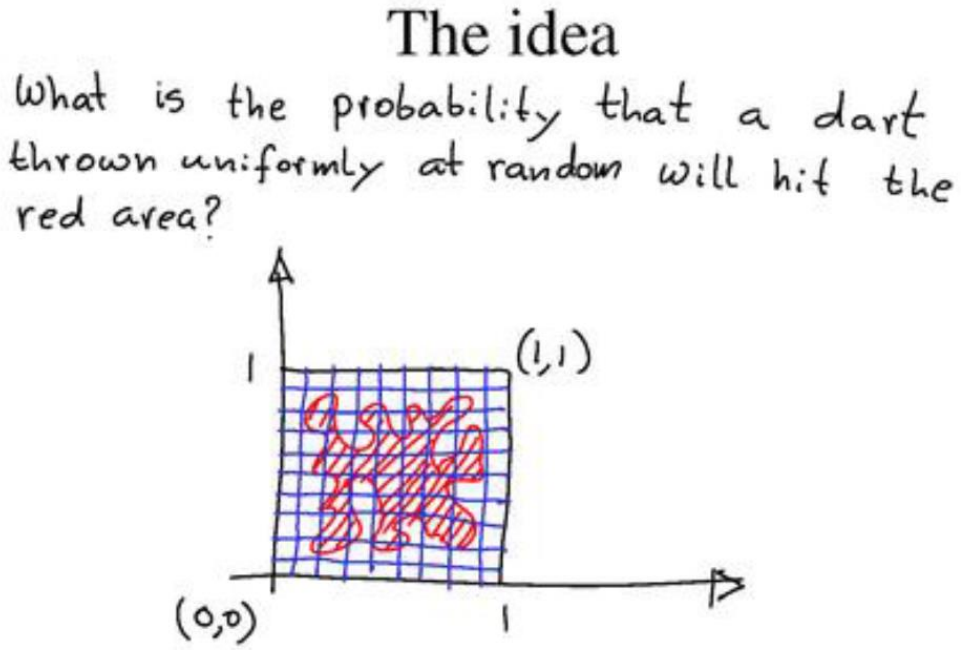
http://videolectures.net/mlss08au_freitas_asm/

Integration and Area



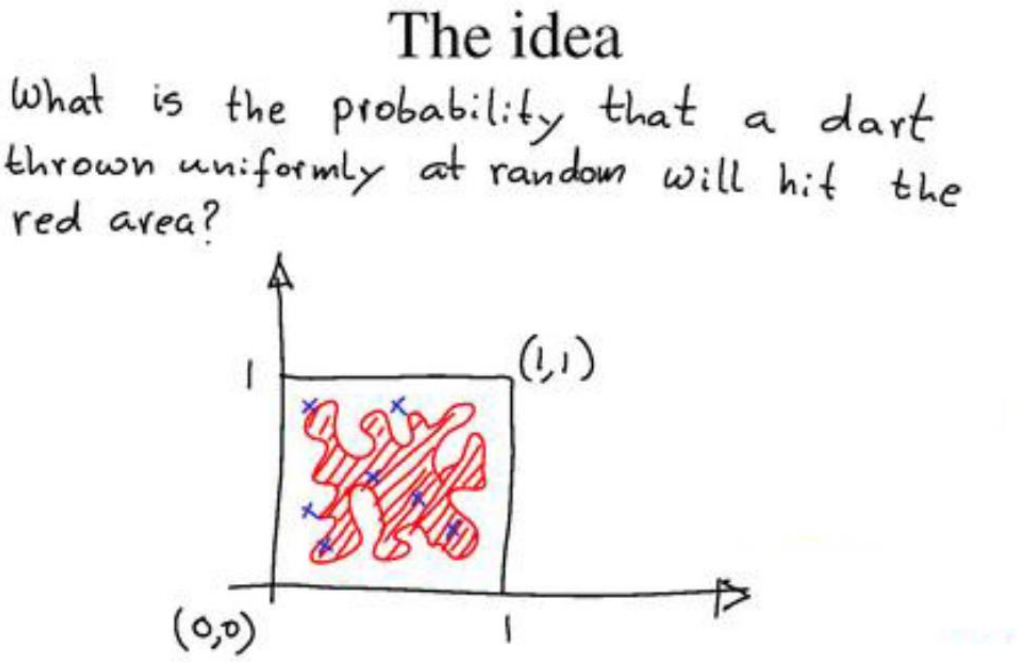
http://videolectures.net/mlss08au_freitas_asm/

Integration and Area



http://videolectures.net/mlss08au_freitas_asm/

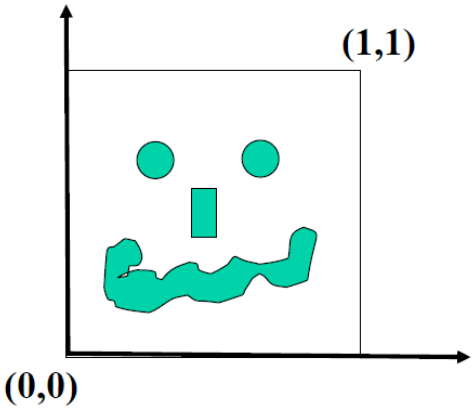
Integration and Area



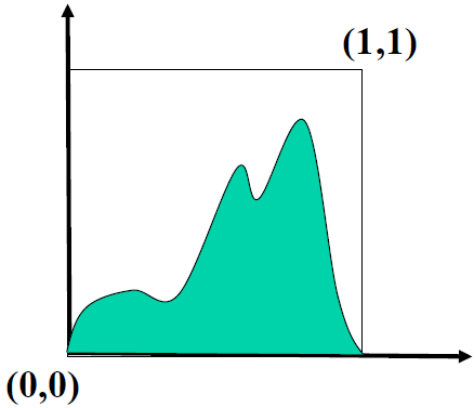
http://videolectures.net/mlss08au_freitas_asm/

Integration and Area

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



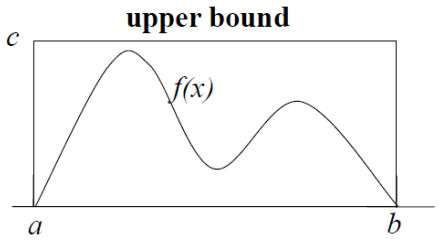
arbitrary region
(even disconnected)



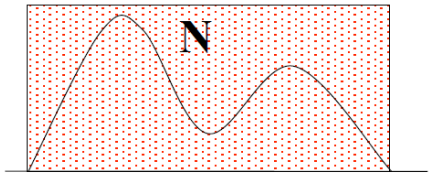
area under curve
aka integration!

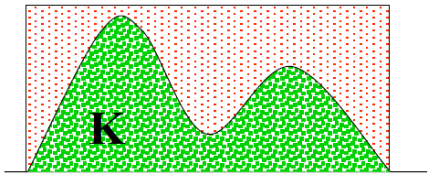
Monte Carlo Integration

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: Normalization 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.

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:

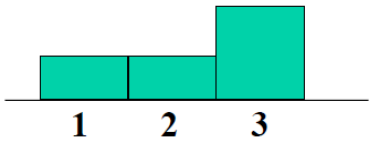
$$\hat{E}(f(x)) = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Expected Values and Sampling

Example:

a discrete pdf

P(x)



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

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

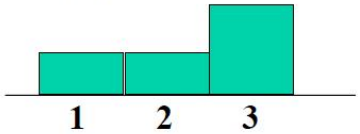
$$= g(1)\frac{1}{4} + g(2)\frac{1}{4} + g(3)\frac{2}{4}$$

Expected Values and Sampling

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)



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

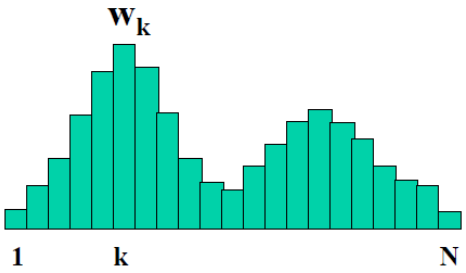
$$= \frac{1}{10}[g(1)+g(3)+g(3)+g(2)+g(2)$$
$$+g(3)+g(1)+g(3)+g(3)+g(1)]$$

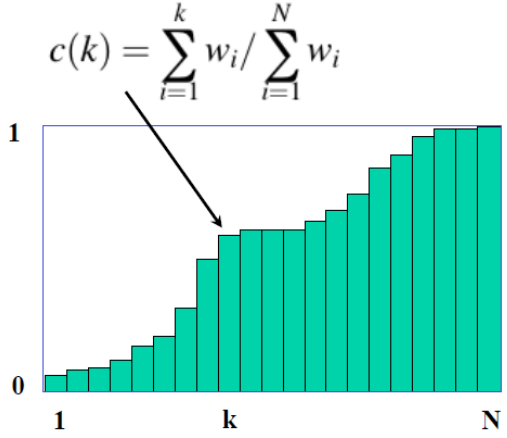
$$= \frac{1}{10}[3g(1)+2g(2)+5g(3)]$$

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

Inverse Transform Sampling

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





cumulative distribution function

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

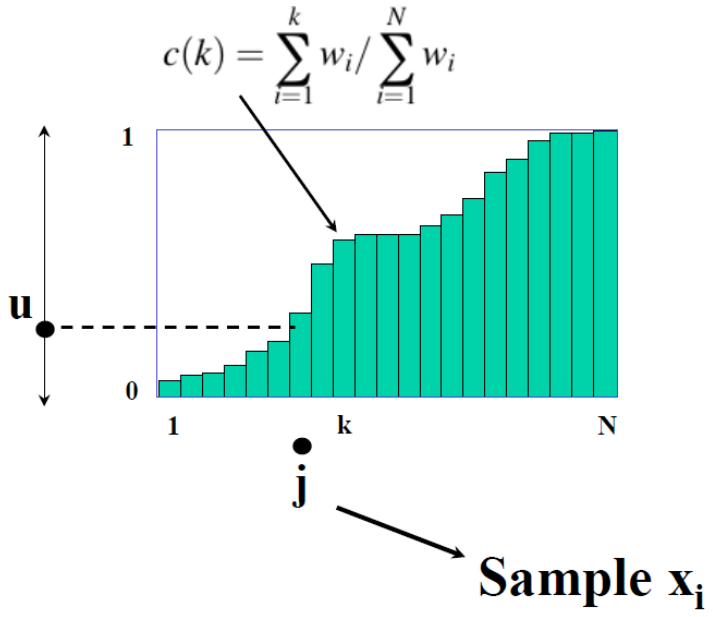
Inverse Transform Sampling

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



Inverse Transform Sampling

Why it works:

cumulative distribution function

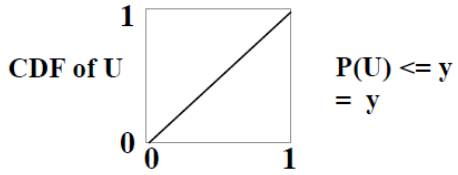
$$F(x) = P(X \leq x)$$

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:

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



Efficient Generation of Many Samples (Arulampalam)

FIAS Frankfurt Institute for Advanced Studies

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

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: $u_1 \sim \mathcal{U}[0, N_s^{-1}]$
- 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

Due to Arulampalam

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...

Efficient Generation of Many Samples

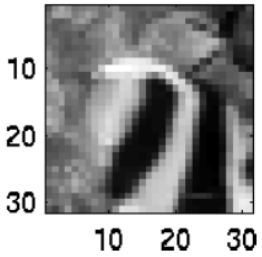
Algorithm 2: Resampling Algorithm

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

- 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: $u_1 \sim \mathcal{U}[0, N_s^{-1}]$
- 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

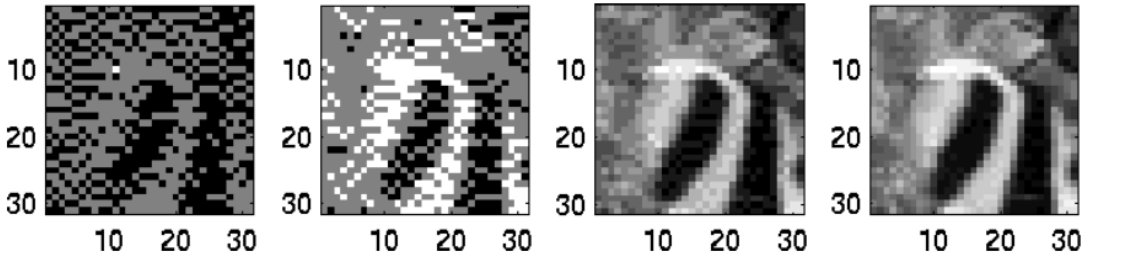
This approach, called "Systematic Resampling" (Kitagawa '96), is known to produce Monte Carlo estimates with minimum variance (more certainty).

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.



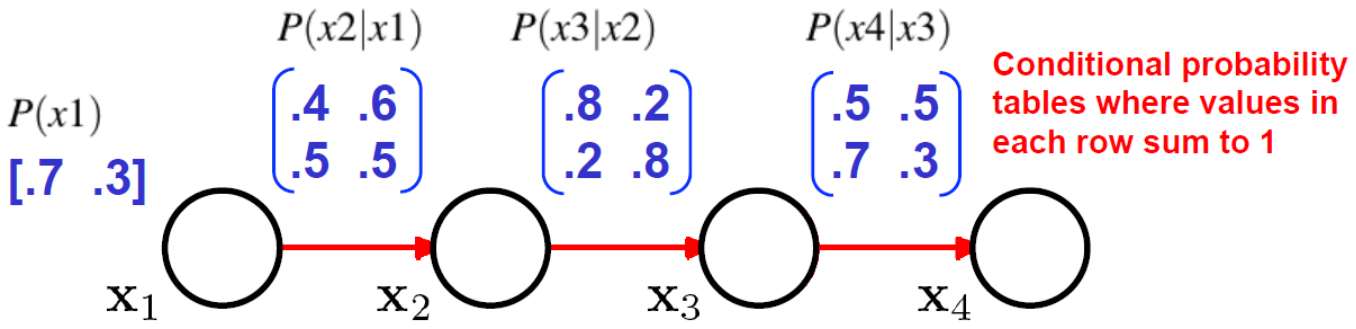
500 samples 1000 samples 5000 samples 10000 samples

Ancestral Sampling (Chris Bishop – PRML book)

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.

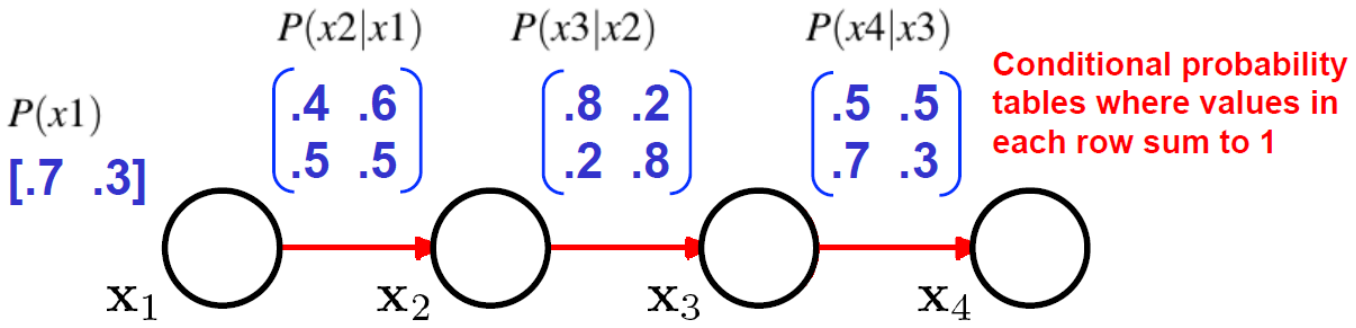
To do this, we start with the lowest-numbered node and draw a sample from the distribution $p(x_1)$, which we call $\widehat{x}_1$. We then work through each of the nodes in order, so that for node n we draw a sample from the conditional distribution $p(x_n|\mathrm{pa}_n)$ in which the parent variables 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 $\{\widehat{x}_{j\neq 2,4}\}$.

Ancestral Sampling



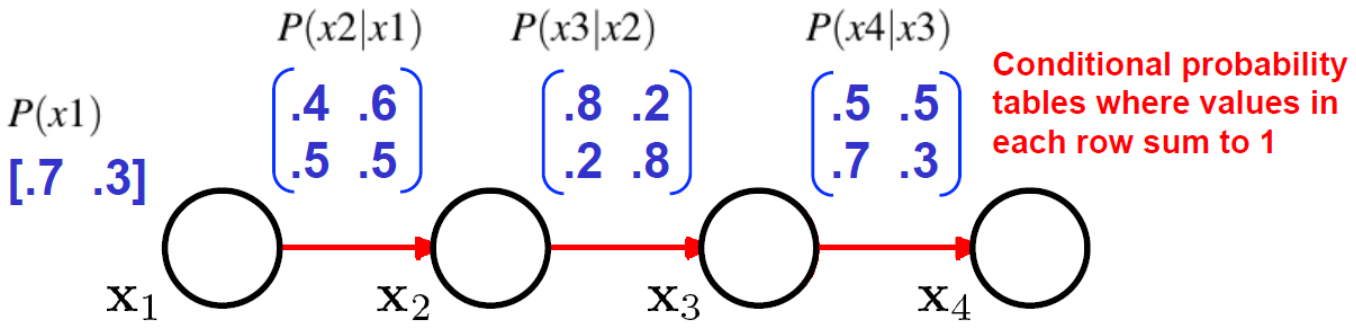
$$P(x1,x2,x3,x4) = P(x1)\ P(x2|x1)\ P(x3|x2)\ P(x4|x3)$$

Ancestral Sampling



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

Ancestral Sampling



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

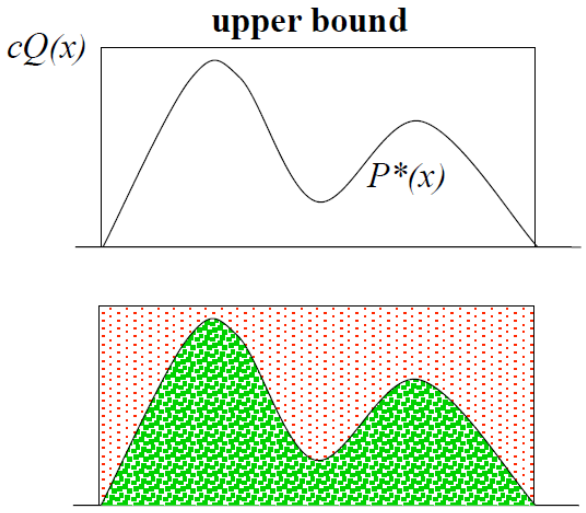
To draw a sample from the joint distribution:

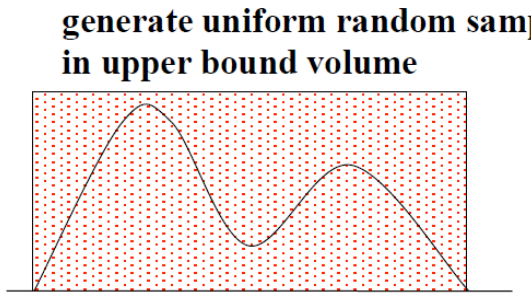
- **Start by sampling from P(x1).**
- **Then sample from P(x2|x1).**
- **Then sample from P(x3|x2).**
- **Finally, sample from P(x4|x3).**
- **{x1,x2,x3,x4} is a sample from the joint distribution.**

Rejection Sampling

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

Example with Q(x) uniform





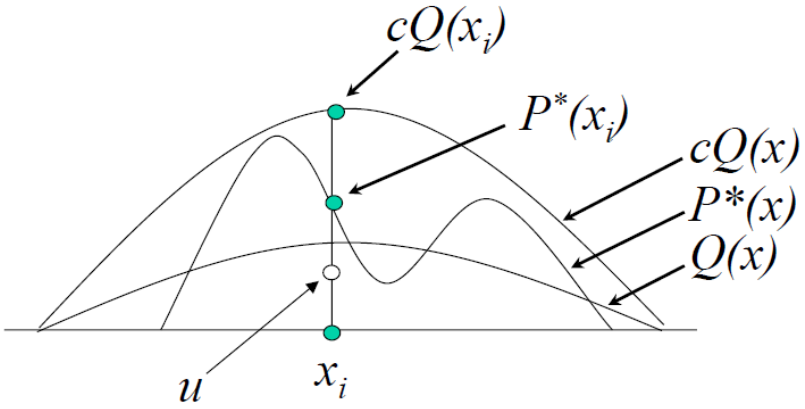
the marginal density of the x coordinates of the points is then proportional to P*(x)

Note the relationship to Monte Carlo integration.

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

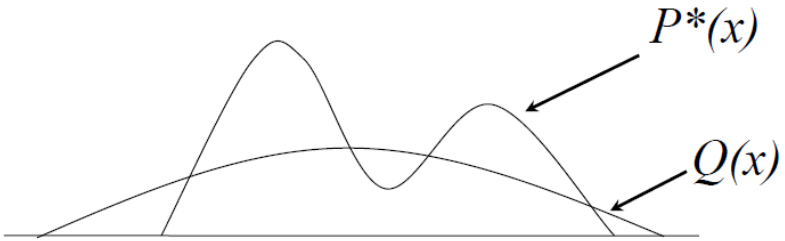
1) Generate x_i from $Q(x)$

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 $(x_i, w_i \mid i=1,\ldots,N)$

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 heavy tails.

Sequential Monte-Carlo

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 with Sampling in High Dim

Sampling in High-dimensional Spaces

Standard methods fail:

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

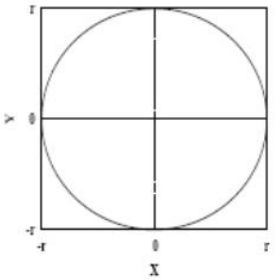
Intuition: In high dimension problems, the "Typical Set" (volume of nonnegligable prob in state space) is a small fraction of the total space.

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

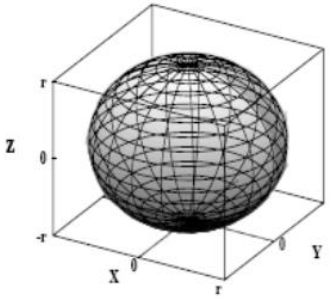
Nature of High Dimensional Spaces

consider ratio of volumes of hypersphere inscribed inside hypercube

2D



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

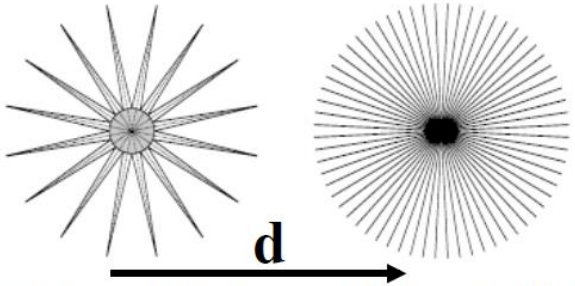


3D

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

Asymptotic behavior:

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



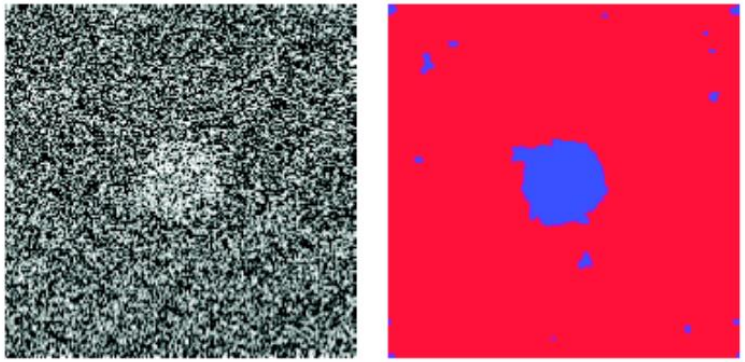
most of volume of the hypercube lies outside of hypersphere as dimension d increases

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

Segmentation Example: High Dim

Segmentation Example

- Binary Segmentation of image



each pixel has two states: on and off

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

Segmentation

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

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

Representation P(Segmentation)

- Histogram ? No !
- Assume pixels independent ?

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

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

- Approximate solution: samples !!!

Metropolis Algorithm

Brilliant Idea!

Nick Metropolis

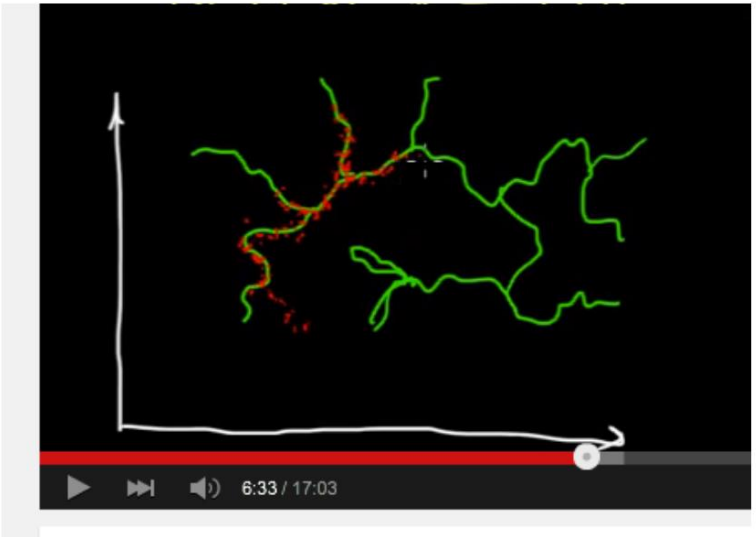
- Published June 1953
- Top 10 algorithm !

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

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

Introduction to MCMC – Video link

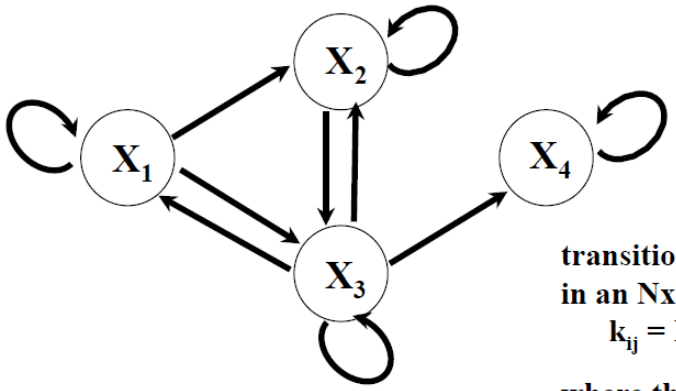
https://www.youtube.com/watch?v=12eZWG0Z5gY



Markov Chains

Markov Chain:

- A sequence of random variables $Y_1, Y_2, Y_3, \ldots$
- Each variable has a distribution over states $(X_1, X_2, X_3, \ldots)$
- Transition probability of going to next state only depends on the current state. e.g. $P(Y_{n+1} = X_j \mid Y_n = X_i)$

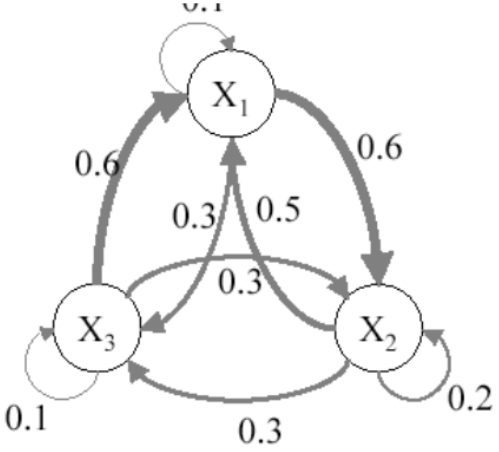


transition probs can be arranged in an NxN table of elements

$$k_{ij} = P(Y_{n+1}=X_j \mid Y_n = X_i)$$

where the rows sum to one

MCMC Sampling: General Idea



Start in some state, and then run the simulation for some number of time steps. After you have run it "long enough" start keeping track of the states you visit.

{... X1 X2 X1 X3 X3 X2 X1 X2 X1 X1 X3 X3 X2 ...}

These are samples from the distribution you want, so you can now compute any expected values with respect to that distribution empirically.

Theory behind MCMC

every state is accessible from every other state.

expected return time to every state is finite

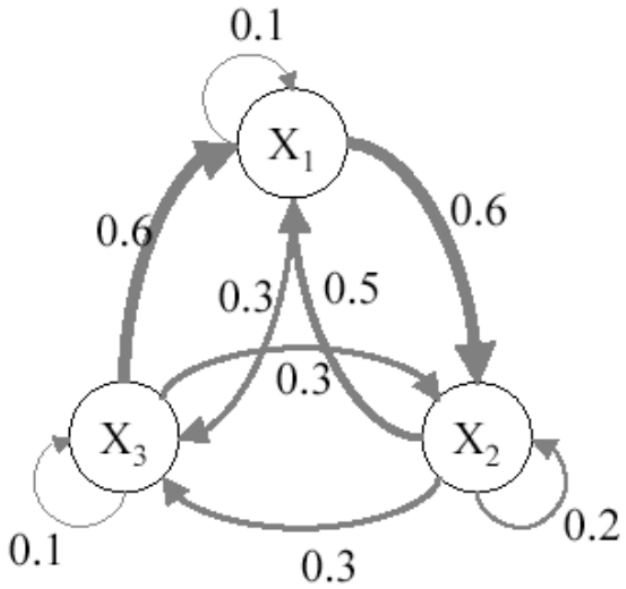
If the Markov chain is positive recurrent, there exists a stationary distribution. If it is positive recurrent and irreducible, there exists a unique stationary distribution. Then, the average of a function f over samples of the Markov chain is equal to the **average with respect to the stationary distribution**

We can compute this empirically as we generate samples.

This is what we want to compute, and is infeasible to compute in any other way.

Example

A simple Markov chain



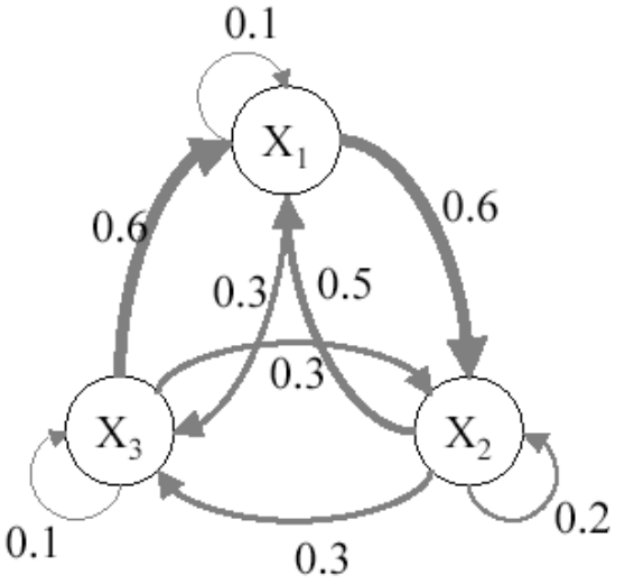
$$K = \begin{bmatrix} 0.1 & 0.5 & 0.6 \\ 0.6 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.1 \end{bmatrix}$$

K= transpose of transition prob table $\{k_{ij}\}$ (cols sum to one. We do this for computational convenience (next slide)

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu

Question

Assume you start in some state, and then run the simulation for a large number of time steps. What percentage of time do you spend at X1, X2 and X3?

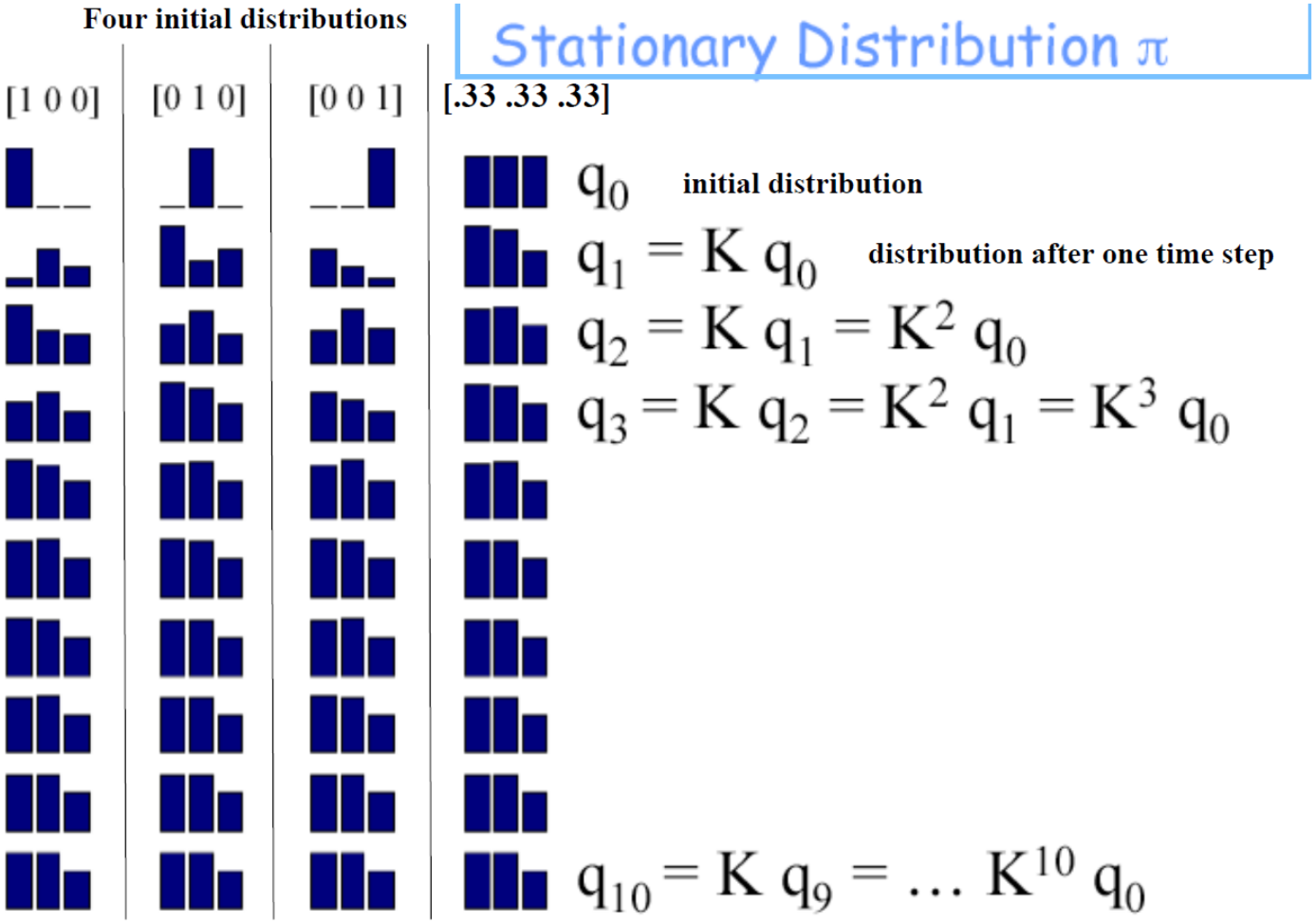


K= [

0.1	0.5	0.6
0.6	0.2	0.3
0.3	0.3	0.1

]

Example:


Four initial distributions
[1 0 0]
[0 1 0]
[0 0 1]
[.33 .33 .33]
Stationary Distribution π
q_0 initial distribution
$q_1 = K\, q_0$ distribution after one time step
$q_2 = K\, q_1 = K^2\, q_0$
$q_3 = K\, q_2 = K^2\, q_1 = K^3\, q_0$
$q_{10} = K\, q_9 = \ldots K^{10}\, q_0$
all eventually end up with same distribution -- this is the stationary distribution!

Eigen-analysis

K =

0.1000	0.5000	0.6000
0.6000	0.2000	0.3000
0.3000	0.3000	0.1000

E =

0.6396	0.7071	-0.2673
0.6396	-0.7071	0.8018
0.4264	0.0000	-0.5345

D =

1.0000	0	0
0	-0.4000	0
0	0	-0.2000

$KE = ED$

in matlab:
[E,D] = eigs(K)

Eigenvalue v_1 always 1

(Perron-Frobenius theorem; K is column stochastic)

Stationary distribution

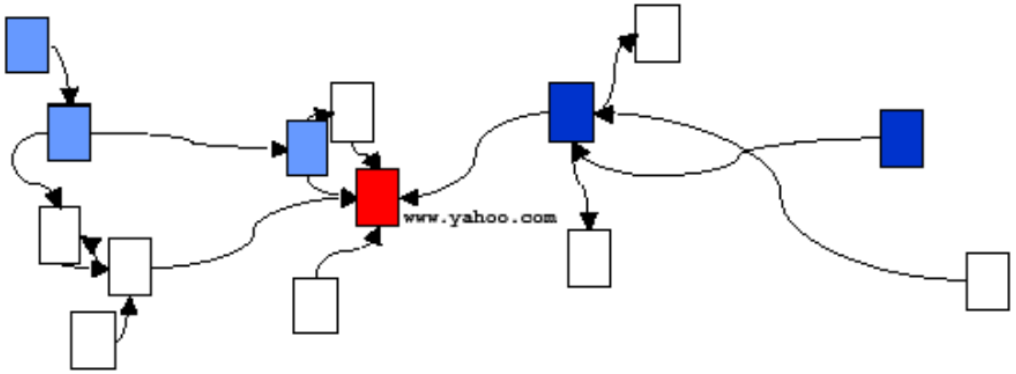
$\pi = e_1/\text{sum}(e_1)$

i.e. $K\pi = \pi$

Note also connection to power method for computing eigenvector associated with largest eigenvalue.

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

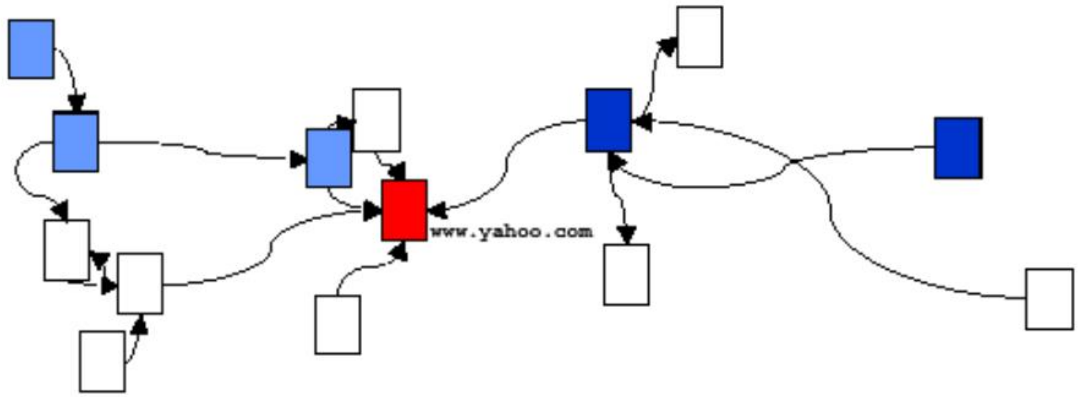
The Web as a Markov Chain



The PageRank of a webpage as used by Google is defined by a Markov chain. It is the probability to be at page i in the stationary distribution on the following Markov chain on all (known) webpages. If N is the number of known webpages, and a page i has k_i links then it has transition probability $(1-q)/k_i + q/N$ for all pages that are linked to and q/N for all pages that are not linked to. The parameter q is taken to be about 0.15.

Google Pagerank

Pagerank == First Eigenvector of the Web Graph !



Computation assumes a 15% "random restart" probability

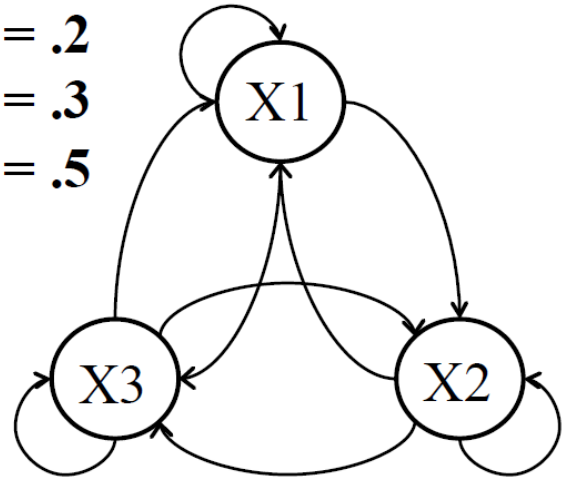
Sergey Brin and Lawrence Page , The anatomy of a large-scale hypertextual {Web} search engine, Computer Networks and ISDN Systems, 1998

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

Another Question

Assume you want to spend a particular percentage of time at X1, X2 and X3. What should the transition probabilities be?

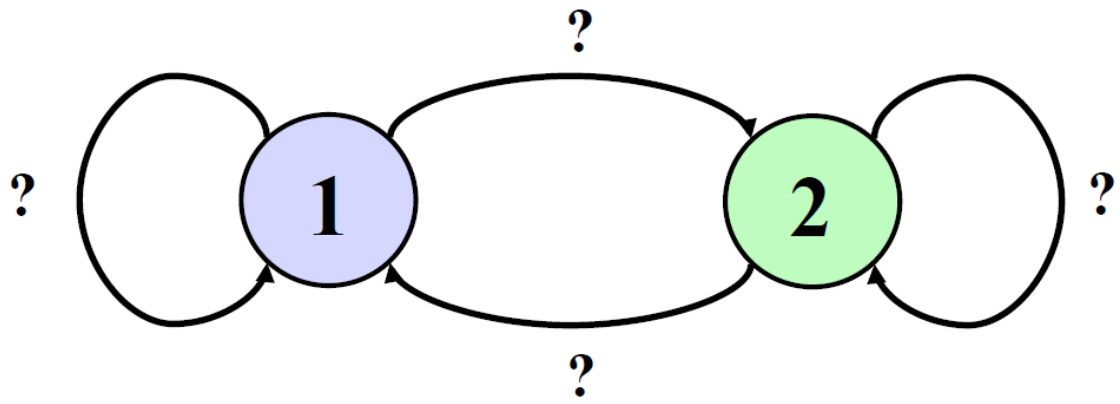
P(x1) = .2
P(x2) = .3
P(x3) = .5



$$K = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Thought Experiment

Consider only two states. What transition probabilities should we use so that we spend roughly equal time in each of the two states? (i.e. 50% of the time we are in state 1 and 50% of the time we are in state 2)



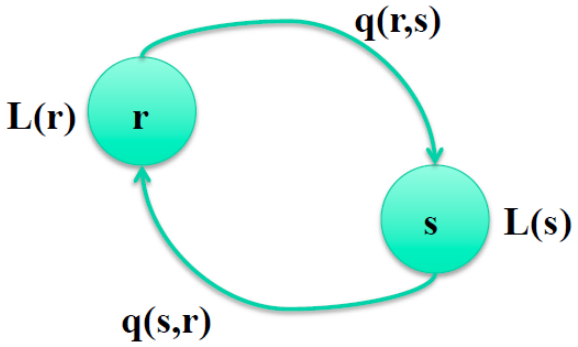
Detailed Balance

- Consider a pair of configuration nodes r,s
- Want to generate them with frequency relative to their likelihoods L(r) and L(s)
- Let q(r,s) be relative frequency of proposing configuration s when the current state is r (and vice versa)

A sufficient condition to generate r,s with the desired frequency is

$$L(r)\, q(r,s) = L(s)\, q(s,r)$$

"detailed balance"



Detailed Balance

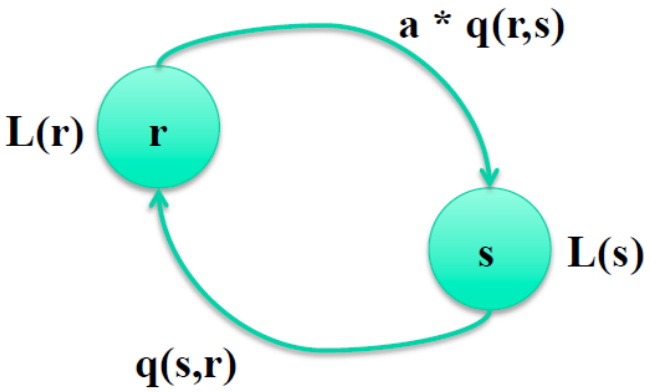
- In practice, you just propose some transition probabilities.
- They typically will NOT satisfy detailed balance (unless you are extremely lucky).
- Instead, you "fix them" by introducing a computational fudge factor

Detailed balance:

$$a^* L(r)\, q(r,s) = L(s)\, q(s,r)$$

Solve for a:

$$a = \frac{L(s)\, q(s,r)}{L(r)\, q(r,s)}$$



Metropolis-Hastings Algorithm

This leads to the following algorithm:

0. Start with $x^{(0)}$, then iterate:
1. propose x' from $q(x^{(t)},x')$
2. calculate ratio

$$a = \frac{\pi(x')q(x',x^{(t)})}{\pi(x^{(t)})q(x^{(t)},x')}$$

3. if a>1 accept $x^{(t+1)}=x'$
   else accept with probability a
   if rejected: $x^{(t+1)}=x^{(t)}$

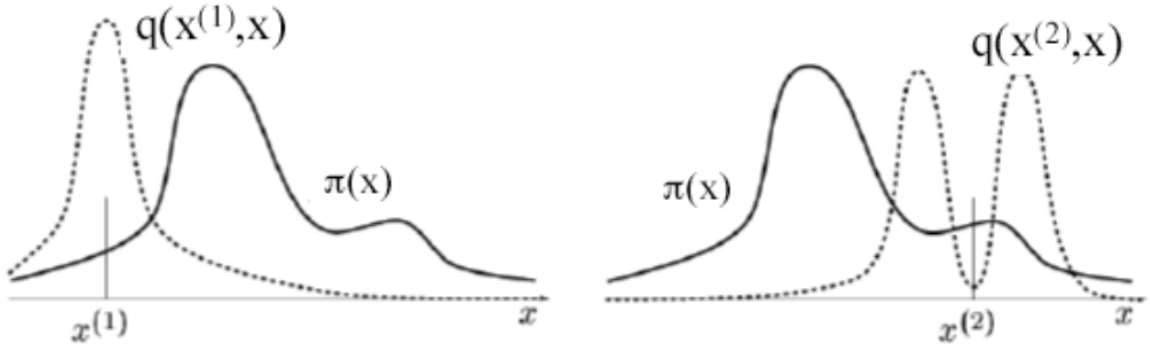
Note: you can just make up transition probability q on-the-fly, using whatever criteria you wish.

diff with rejection sampling: instead of throwing away rejections, you replicate them into next time step.

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu

Proposal Density q(x,x')

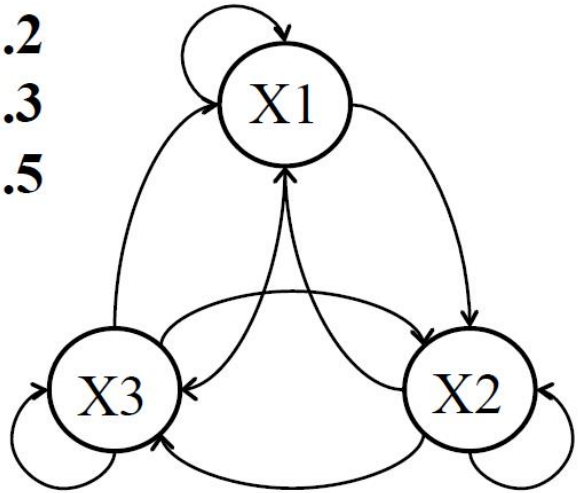
Note: the transition probabilities $q(x^{(t)},x)$ can be arbitrary distributions. They can depend on the current state and change at every time step, if you want.



ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

Metropolis-Hastings Example

P(x1) = .2
P(x2) = .3
P(x3) = .5



Proposal distribution
q(xi, (xi-1)mod3) = .4
q(xi, (xi+1)mod3) = .6

```
% simple metropolis hastings example
% Bob Collins, Penn State University

%desired stationary distrbution
pdist = [.2 .3 .5]

%start state
state = 1;
statelist = [state];

for i=1:10000
    %proposal function (nonsymmetric)
    %go to mod(state-1) with prob .4
    %go to mod(state+1) with prob .6
    tmp = rand(1);
    if (tmp <= .4)
        propstate = mod(state-1-1,3)+1;
        proptrans = .4;
    else
        propstate = mod(state-1+1,3)+1;
        proptrans = .6;
    end

    a =  (pdist(propstate) * (1-proptrans)) / (pdist(state) * proptrans);

    tmp = rand(1);
    if (tmp <= a)
        %accept
        state = propstate;
        %else reject -- leave state the same
    end
    statelist = [statelist state];

end

prob = [length(find(statelist==1))...
        length(find(statelist==2))...
        length(find(statelist==3))];
prob = prob / sum(prob);

bar([prob; pdist]')
prob
```

Variants of MCMC

- there are many variations on this general approach, some derived as special cases of the Metropolis-Hastings algorithm

The Metropolis Algorithm

When q is symmetric, i.e., q(x,x')=q(x',x) : **e.g. Gaussian**

0. Start with $x^{(0)}$, then iterate:
1. propose x' from $q(x^{(t)},x')$
2. calculate ratio

$$a = \frac{\pi(x')}{\pi(x^{(t)})} \quad \frac{q(x',x)}{q(x, x')}$$

cancels

3. if a>1 accept $x^{(t+1)}=x'$
   else accept with probability a
   if rejected: $x^{(t+1)}=x^{(t)}$

Gibbs Sampler

Special case of MH with acceptance ratio always 1 (so you always accept the proposal).

$$q(\mathbf{x}, \mathbf{y}) = \begin{cases} \pi(\mathbf{y}_i|\mathbf{x}_{(i)}) & \mathbf{y}_{(i)} = \mathbf{x}_{(i)},\ i = 1, \ldots, k, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathbf{x}_{(i)} = (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_k),\ i = 1, \ldots, k,\ 1 < k \leqslant p,$

With this proposal, the corresponding acceptance probability is given by

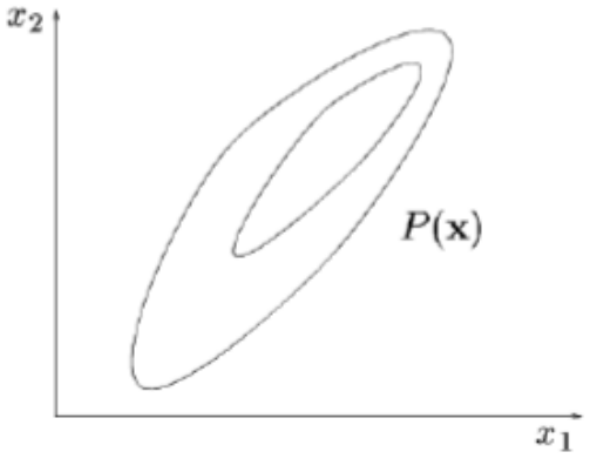
$$\begin{aligned} \alpha(\mathbf{x}, \mathbf{y}) &= \frac{\pi(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{\pi(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})} \\ &= \frac{\pi(\mathbf{y})/\pi(\mathbf{y}_i|\mathbf{x}_{(i)})}{\pi(\mathbf{x})/\pi(\mathbf{x}_i|\mathbf{y}_{(i)})} \\ &= \frac{\pi(\mathbf{y})/\pi(\mathbf{y}_i|\mathbf{y}_{(i)})}{\pi(\mathbf{x})/\pi(\mathbf{x}_i|\mathbf{x}_{(i)})}, \qquad \text{since } \mathbf{y}_{(i)} = \mathbf{x}_{(i)}, \\ &= \frac{\pi(\mathbf{y}_{(i)})}{\pi(\mathbf{x}_{(i)})}, \qquad \text{by definition of conditional probability for } \boldsymbol{\theta} = (\boldsymbol{\theta}_i, \boldsymbol{\theta}_{(i)}), \\ &= 1, \qquad \text{since } \mathbf{y}_{(i)} = \mathbf{x}_{(i)}. \end{aligned}$$

S.Brooks, "Markov Chain Monte Carlo and its Application"

simpler version, using 1D conditional distributions

Gibbs Sampling

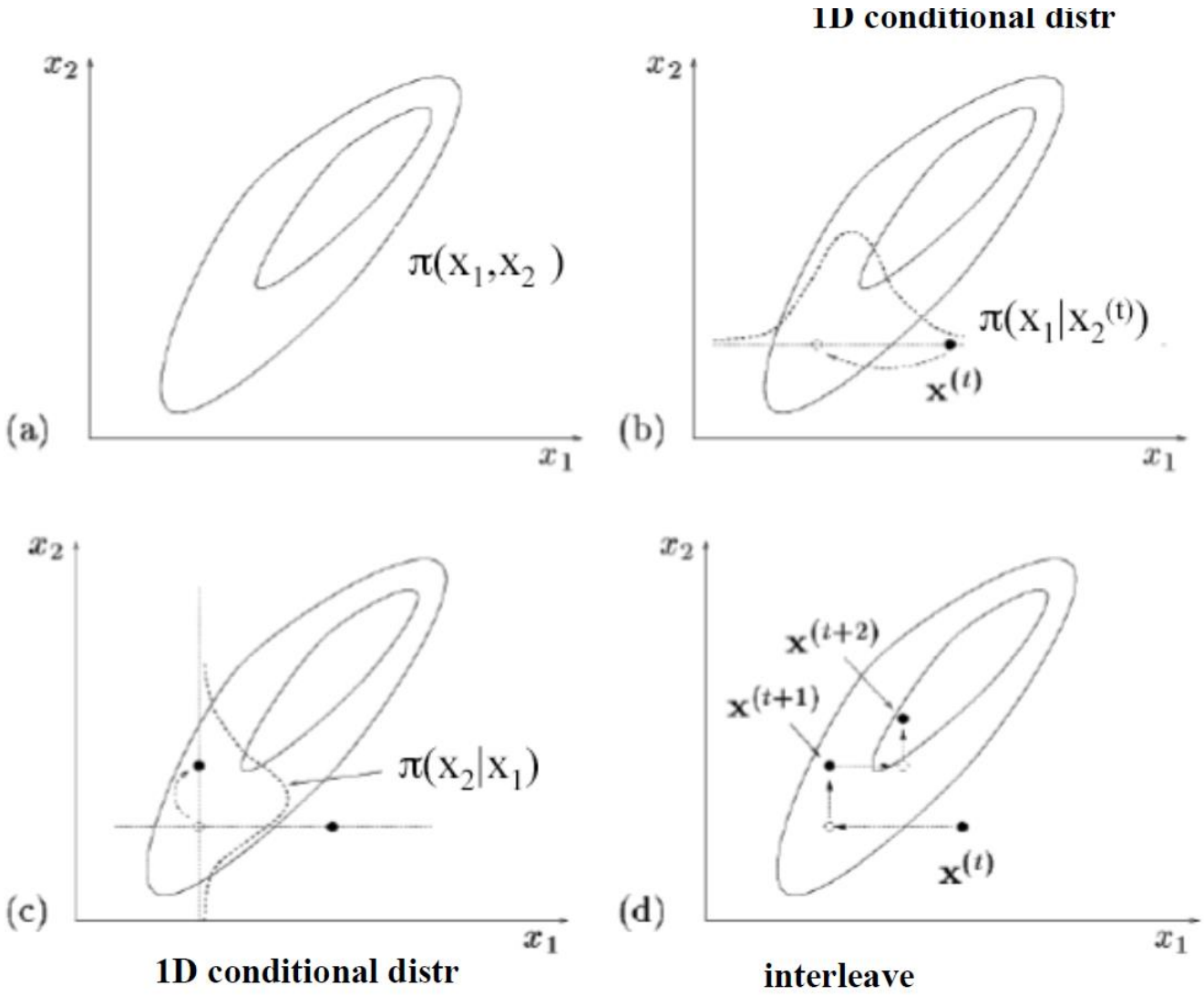
- Example: target $\pi(x_1,x_2)$
- Algorithm:
  - alternate between x_1 and x_2
  - 1. sample from $x_1 \sim P(x_1|x_2)$
  - 2. sample from $x_2 \sim P(x_2|x_1)$
- After a while: samples from target density !



- Sampler equivalent of "Gauss-Seidel" iterations **or line search, or ...**

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu October 2005

Example


1D conditional distr
x_2
x_1
$\pi(x_1,x_2)$
(a)
$\pi(x_1|x_2^{(t)})$
$x^{(t)}$
(b)
$\pi(x_2|x_1)$
(c)
1D conditional distr
$x^{(t+2)}$
$x^{(t+1)}$
$x^{(t)}$
(d)
interleave

Simulated Annealing

- introduce a "temperature" term that makes it more likely to accept proposals early on. This leads to more aggressive exploration of the state space.
- Gradually reduce the temperature, causing the process to spend more time exploring high likelihood states.
- Rather than remember all states visited, keep track of the best state you've seen so far. This is a method that attempts to find the global max (MAP) state.

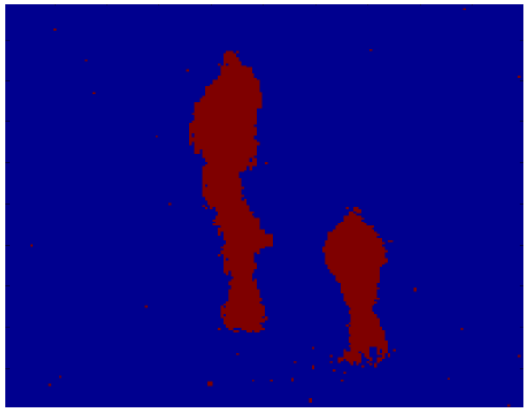
Trans-Dimensional MCMC

- Exploring alternative state spaces of differing dimensions (example, when doing EM, also try to estimate number of clusters along with parameters of each cluster).
- Green's reversible-jump approach (RJMCMC) gives a general template for exploring and comparing states of differing dimension.

People Counting

Problem statement: Given a foreground image, and person-sized bounding box*, find a configuration (number and locations) of bounding boxes that cover a majority of foreground pixels while leaving a majority of background pixels uncovered.

foreground image



person-sized bounding box

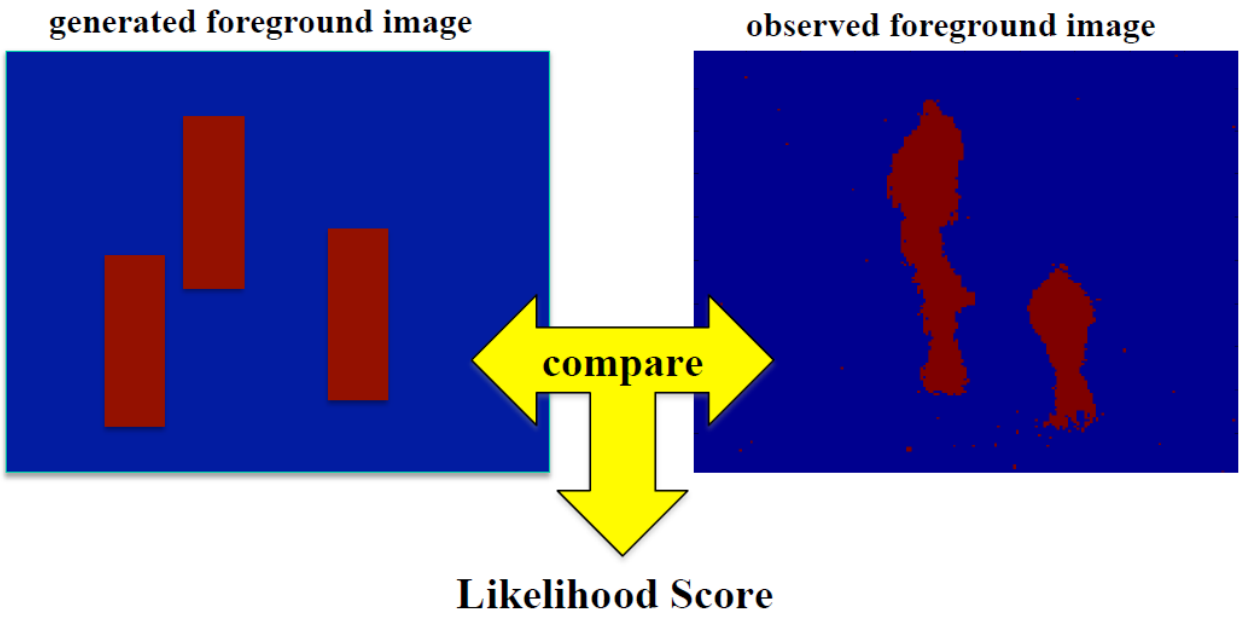
***note: height, width and orientation of the bounding box may depend on image location… we determine these relationships beforehand through a calibration procedure.**

W.Ge and R.Collins, "Marked Point Processes for Crowd Counting," *IEEE Computer Vision and Pattern Recognition (CVPR'09),* Miami, FL, June 2009, pp.2913-2920.

Likelihood Score

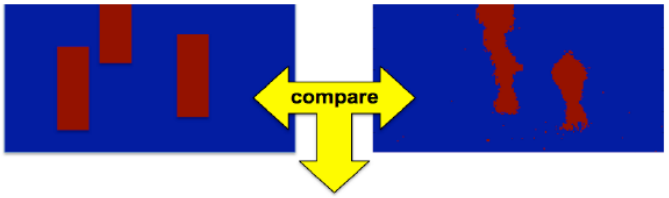
To measure how "good" a proposed configuration is, we generate a foreground image from it and compare with the observed foreground image to get a likelihood score.

config = $\{\{x_1,y_1,w_1,h_1,theta_1\},\{x_2,y_2,w_2,h_2,theta_2\},\{x_3,y_3,w_3,h_3,theta_3\}\}$



Likelihood Score

Likelihood Score



Bernoulli distribution model

$p_{00} = p(y_i = 0 | x_i = 0)$ = prob of observing background given a label of background
$p_{01} = p(y_i = 0 | x_i = 1)$ = prob of observing background given a label of foreground
$p_{10} = p(y_i = 1 | x_i = 0)$ = prob of observing foreground given a label of background
$p_{11} = p(y_i = 1 | x_i = 1)$ = prob of observing foreground given a label of foreground

c_{00} = count of pixels where observation is background and label is background
c_{01} = count of pixels where observation is background and label is foreground
c_{10} = count of pixels where observation is foreground and label is background
c_{11} = count of pixels where observation is foreground and label is foreground

likelihood

$$L(Y|X) = \prod_{i=1}^{N} p(y_i|x_i) = {p_{00}}^{c_{00}} {p_{01}}^{c_{01}} {p_{10}}^{c_{10}} {p_{11}}^{c_{11}}$$

simplify, by assuming $p_{00} = p_{11} = \mu$ and $p_{01} = p_{10} = 1 - \mu$

log likelihood

$$\begin{aligned} logL(Y|X) &= (c_{00} + c_{11}) \log\mu + (c_{01} + c_{10}) \log(1-\mu) \\ &= [N - (c_{01} + c_{10})] \log\mu + (c_{01} + c_{10}) \log(1-\mu) \\ &= N \log\mu - (c_{01} + c_{10}) [\log\mu - \log(1-\mu)] \end{aligned}$$

Number of pixels that disagree!

Searching for the Max

The space of configurations is very large. We can't exhaustively search for the max likelihood configuration. We can't even really uniformly sample the space to a reasonable degree of accuracy.

$$config_k = \{\{x_1,y_1,w_1,h_1,theta_1\},\{x_2,y_2,w_2,h_2,theta_2\},\ldots,\{x_k,y_k,w_k,h_k,theta_k\}\}$$

Let N = number of possible locations for (x_i,y_i) in a k-person configuration.

Size of $config_k = N^k$
And we don't even know how many people there are...

Size of config space = $N^0 + N^1 + N^2 + N^3 + \ldots$

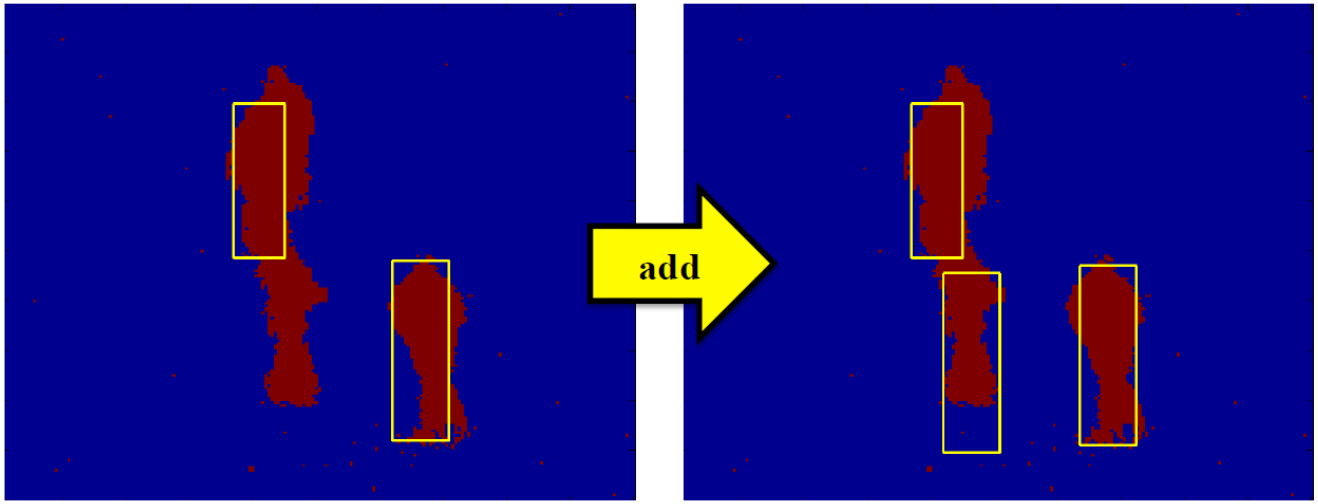
If we also wanted to search for width, height and orientation, this space would be even more huge.

Searching for the Max

- Local Search Approach
  - Given a current configuration, propose a small change to it
  - Compare likelihood of proposed config with likelihood of the current config
  - Decide whether to accept the change

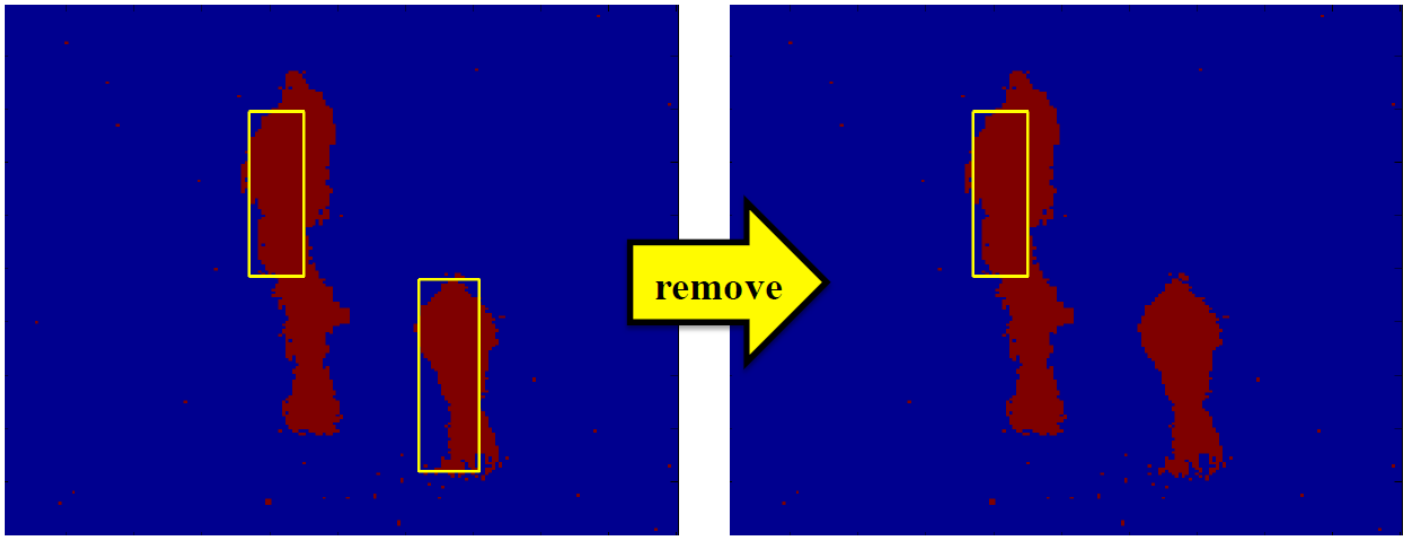
Proposals

- Add a rectangle (birth)



Proposals

- Remove a rectangle (death)

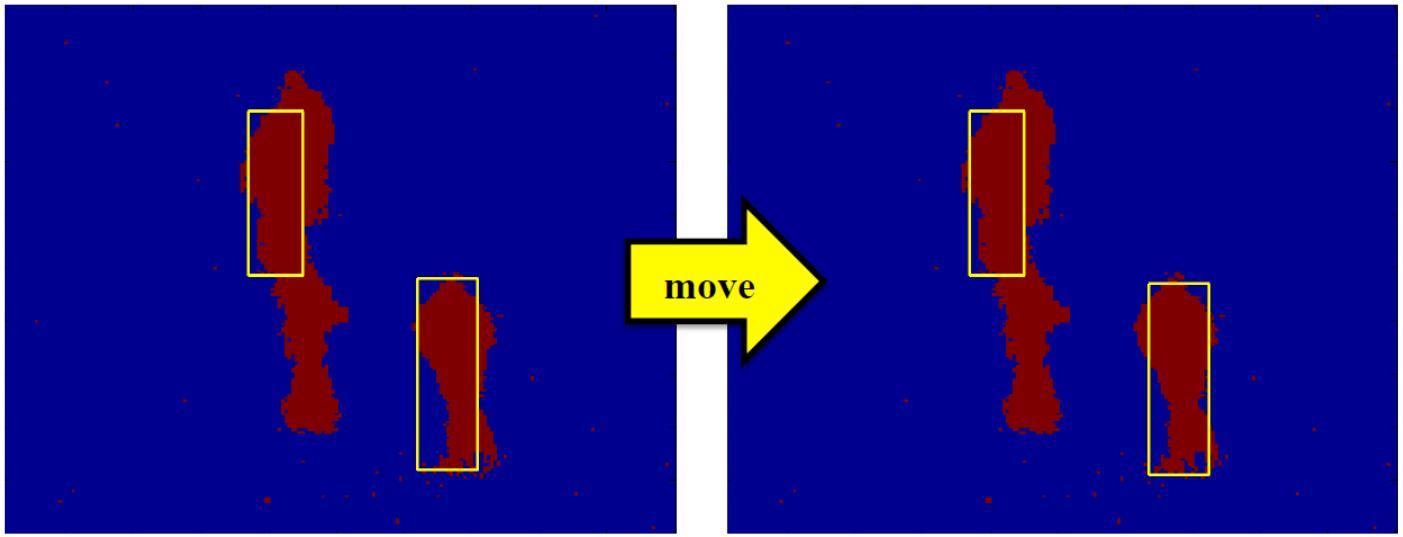


current configuration

proposed configuration

Proposals

- Move a rectangle



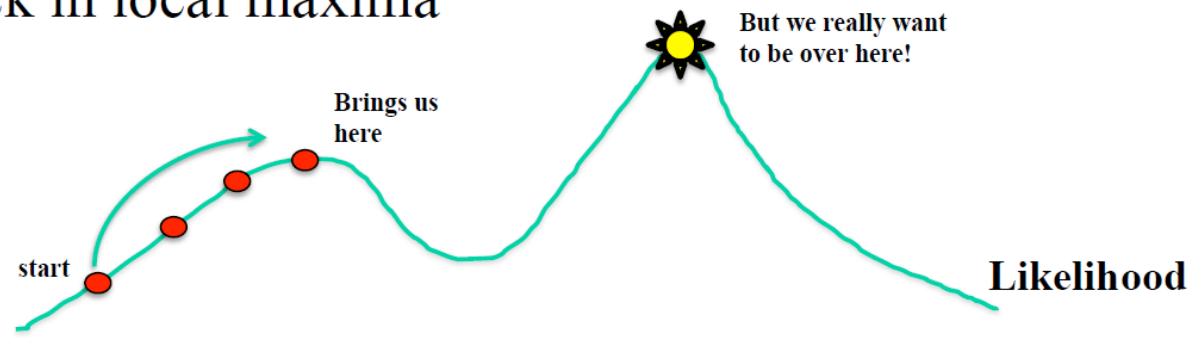
Searching for the Max

- Naïve Acceptance
  - Accept proposed configuration if it has a larger likelihood score, i.e.

$$\textbf{Compute } a = \frac{\textbf{L(proposed)}}{\textbf{L(current)}}$$

Accept if a > 1

  - Problem: leads to hill-climbing behavior that gets stuck in local maxima



MCMC Sampling

- Metropolis Hastings algorithm

  Propose a new configuration

  Compute $a = \frac{L(\text{proposed})}{L(\text{current})} \cdot \frac{q(\text{proposed},\text{current})}{q(\text{current},\text{proposed})}$

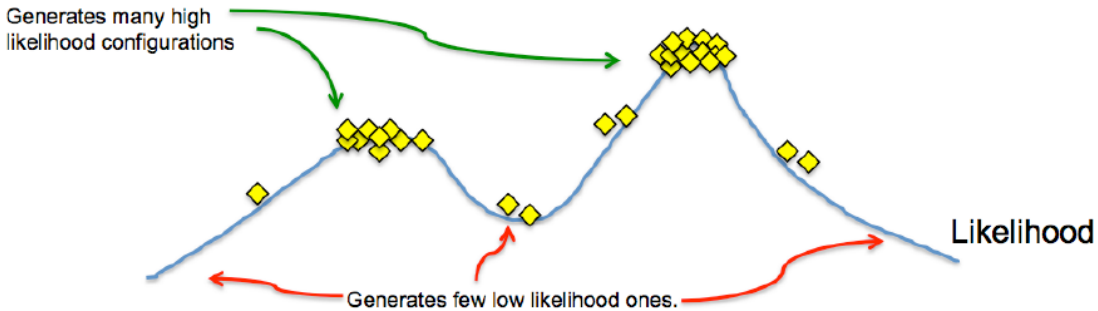
  Accept if a > 1

  Else accept anyways with probability a

Difference from Naïve algorithm

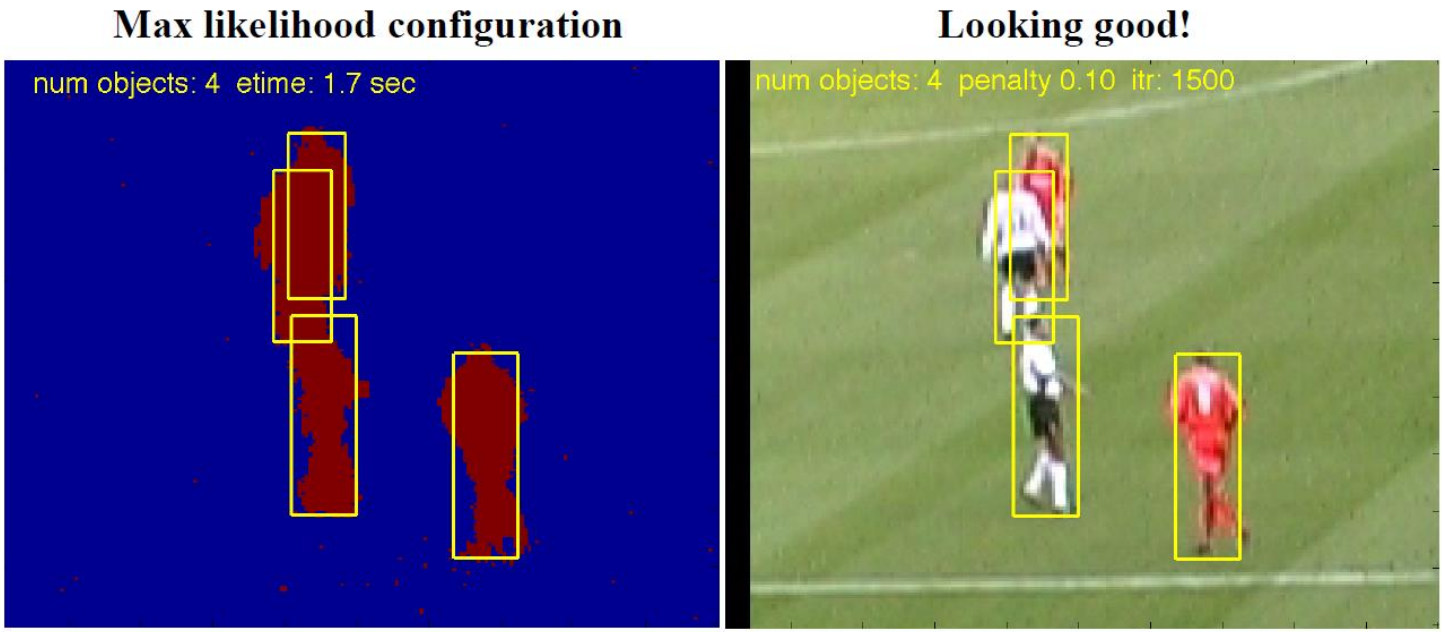
MCMC Approach

- The MCMC approach
  - Generates random configurations from a distribution proportional to the likelihood!

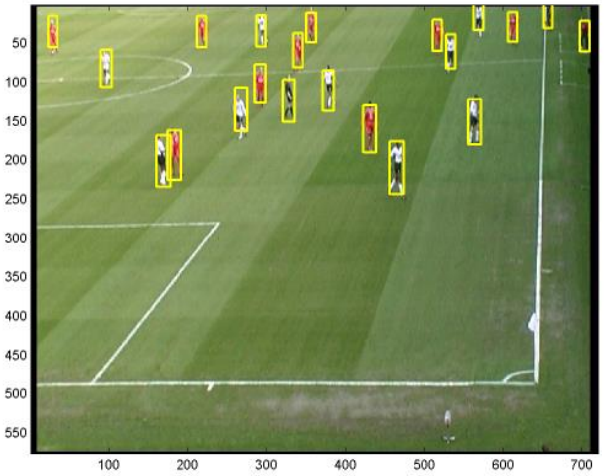


  - **This searches the space of configurations in an efficient way.**
  - Now just remember the generated configuration with the highest likelihood.

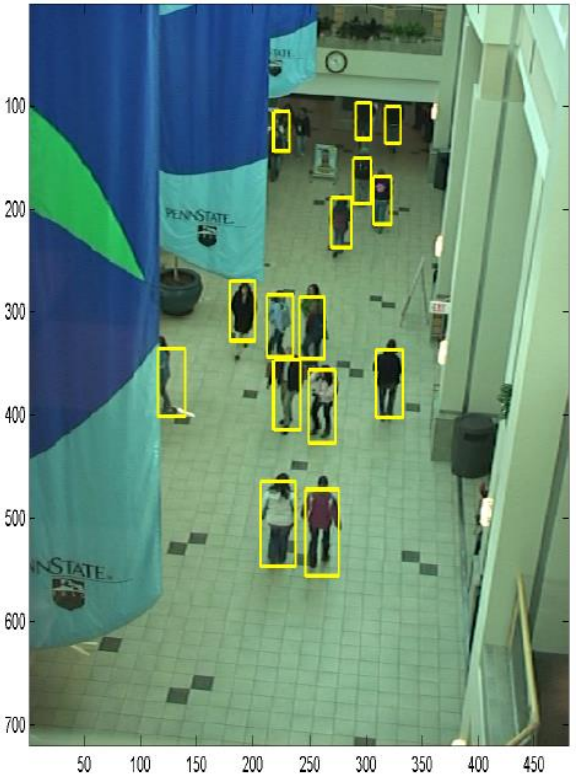
Example Results



Examples


50
100
150
200
250
300
350
400
450
500
550
100
200
300
400
500
600
700


50
100
150
200
250
50
100
150
200
250
300
350


100
200
300
400
500
600
700
50
100
150
200
250
300
350
400
450

Other Examples

CVPR, ICCV Papers

Bayesian Human Segmentation in Crowded Situations *

Tao Zhao Ram Nevatia
University of Southern California
Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089-0273
{taozhao|nevatia}@usc.edu

Fast Crowd Segmentation Using Shape Indexing

Lan Dong*
Dept. of Electrical Engineering
Princeton University

Vasu Parameswaran, Visvanathan Ramesh, Imad Zoghlami
Real-Time Vision and Modeling Department
Siemens Corporate Research

Backup