Robert Collins
CSE586, PSU

Markov-Chain Monte Carlo

CSE586 Computer Vision II

Penn State Univ

Recall: 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

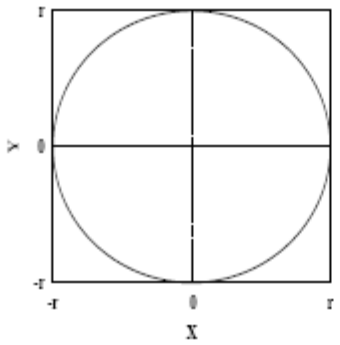
Intuition: In high dimension problems, the "Typical Set" (volume of nonnegligable prob in state space) is a small fraction of the total space.

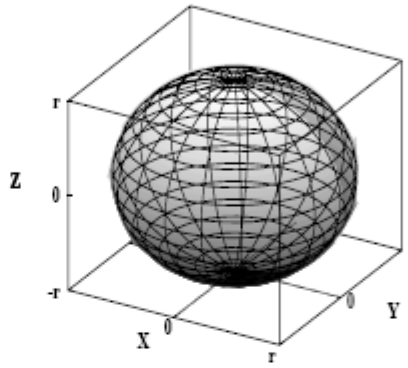
Robert Collins
CSE586, PSU

High-Dimensional Spaces

consider ratio of volumes of hypersphere inscribed inside hypercube

2D





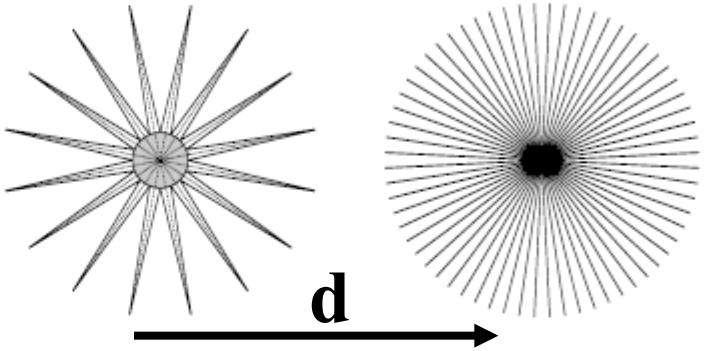
3D

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

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

Asymptotic behavior:

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

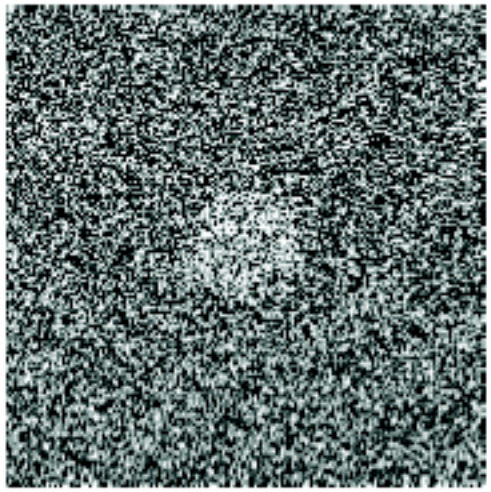


most of volume of the hypercube lies outside of hypersphere as dimension d increases

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

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 ?

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

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

- Approximate solution: samples !!!

Brilliant Idea!

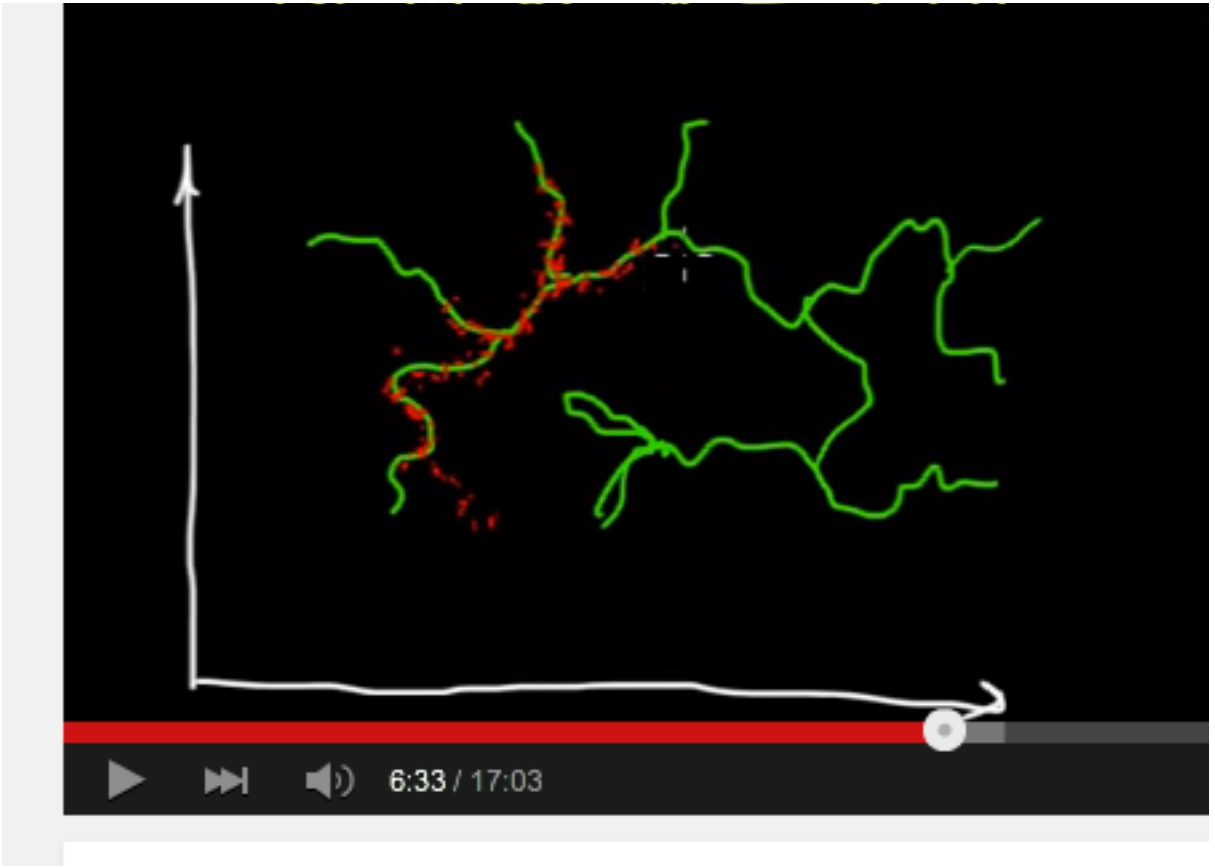

Nick Metropolis

- Published June 1953
- Top 10 algorithm !

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

Robert Collins
CSE586, PSU

Good Intuitive Intro



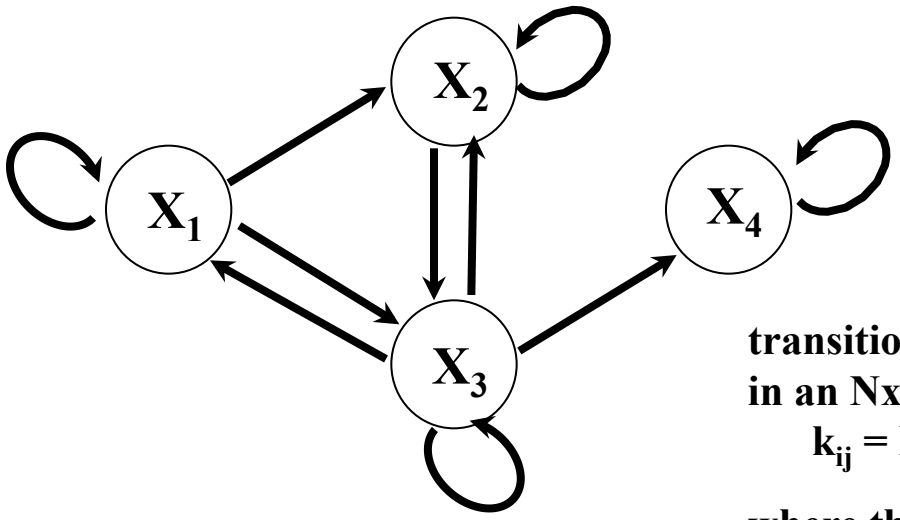
https://www.youtube.com/watch?v=12eZWG0Z5gY

Robert Collins
CSE586, PSU

Recall: 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. $P(Y_{n+1} = X_j \mid Y_n = X_i)$



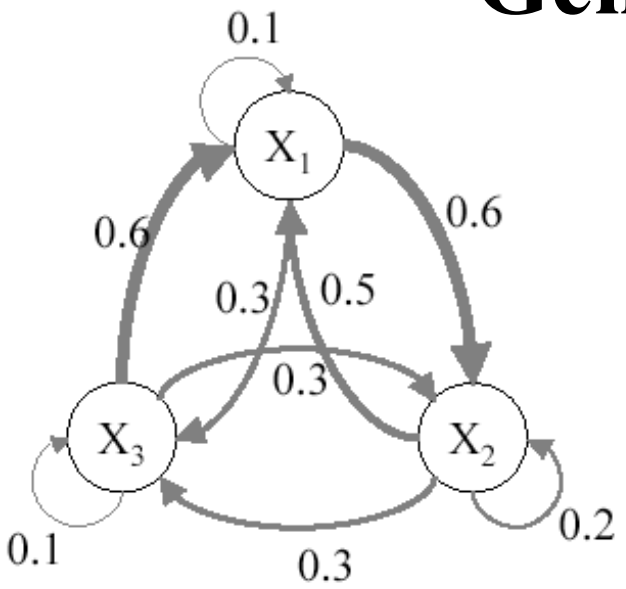
transition probs can be arranged in an NxN table of elements

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

where the rows sum to one

Robert Collins
CSE586, PSU

General Idea: MCMC Sampling



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.

Robert Collins
CSE586, PSU

The Theory Behind MCMC Sampling

every state is accessible from every other state.

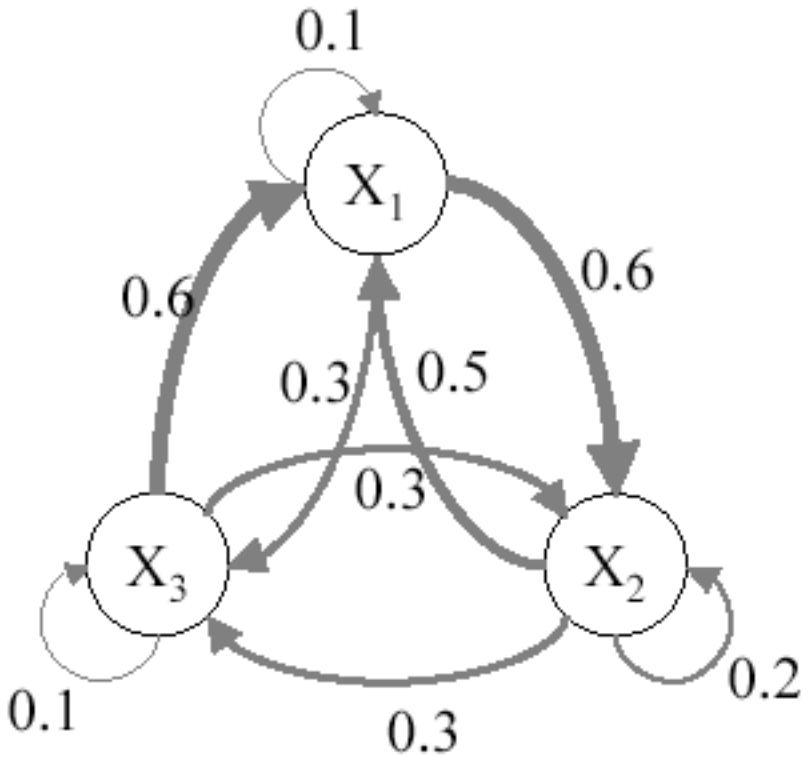
expected return time to every state is finite

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.

A simple Markov chain



K= [

0.1	0.5	0.6
0.6	0.2	0.3
0.3	0.3	0.1

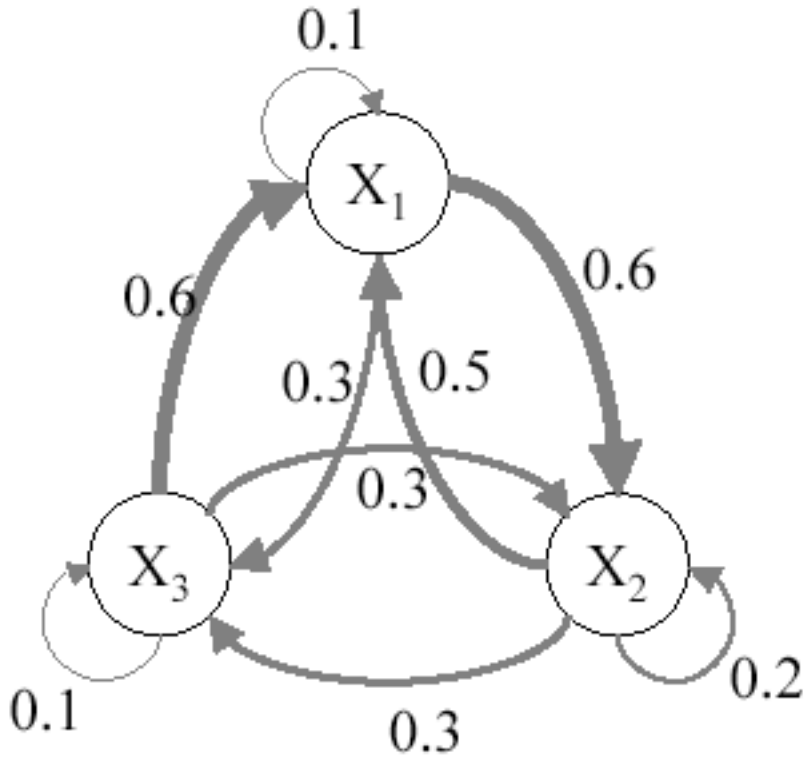
]

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

Robert Collins
CSE586, PSU

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= [

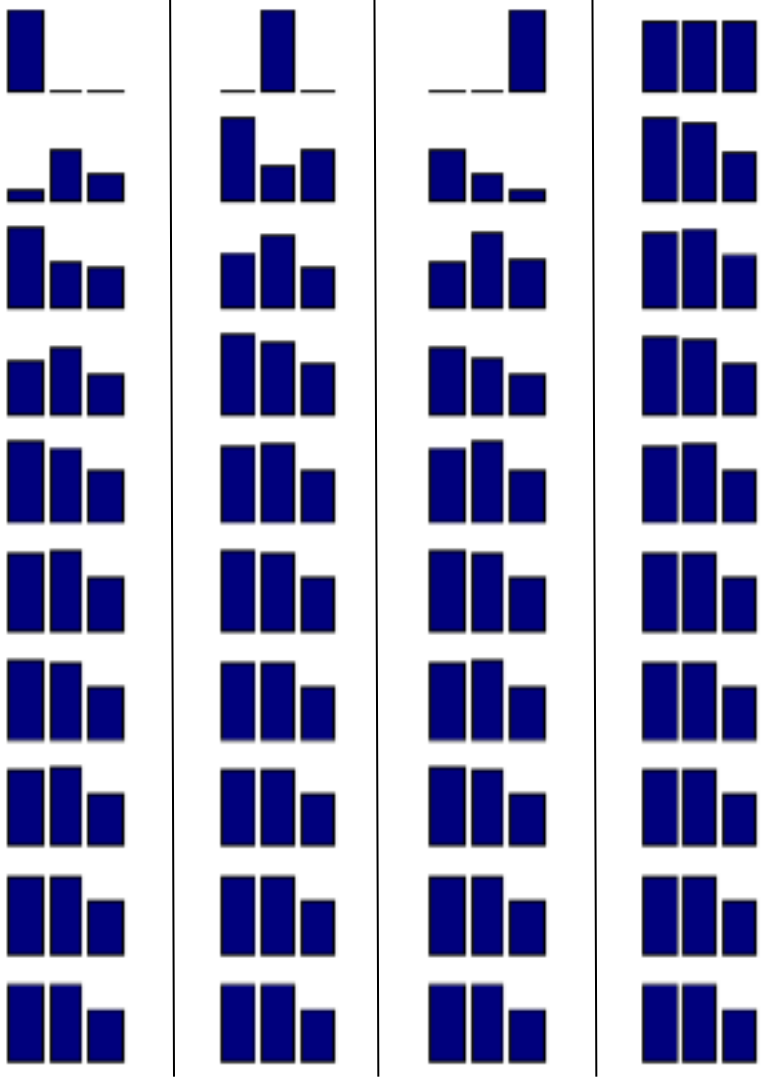
0.1	0.5	0.6
0.6	0.2	0.3
0.3	0.3	0.1

]

Stationary Distribution π

Four initial distributions

[1 0 0] [0 1 0] [0 0 1] **[.33 .33 .33]**



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)

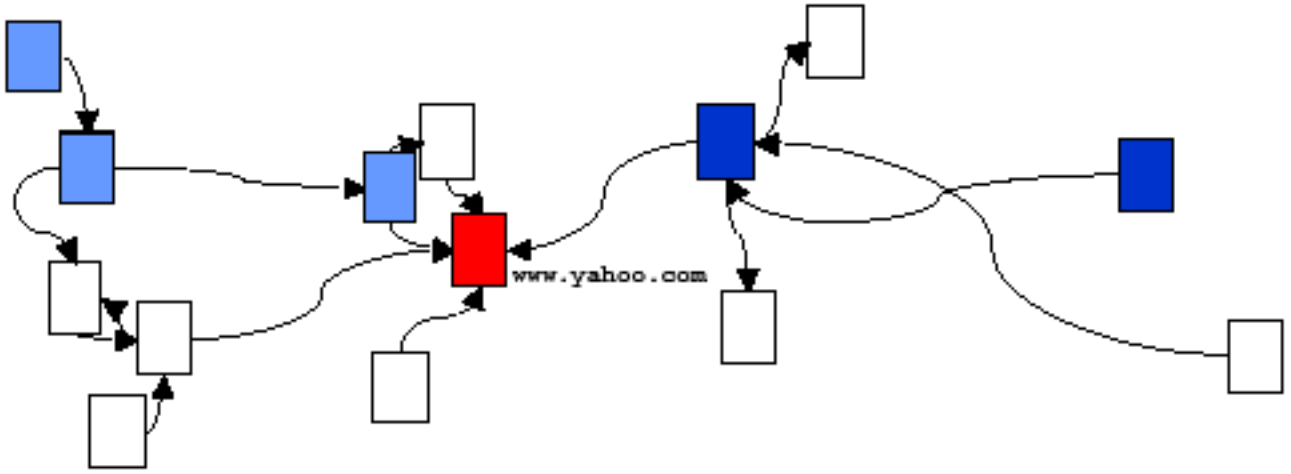
Stationary distribution

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

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

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

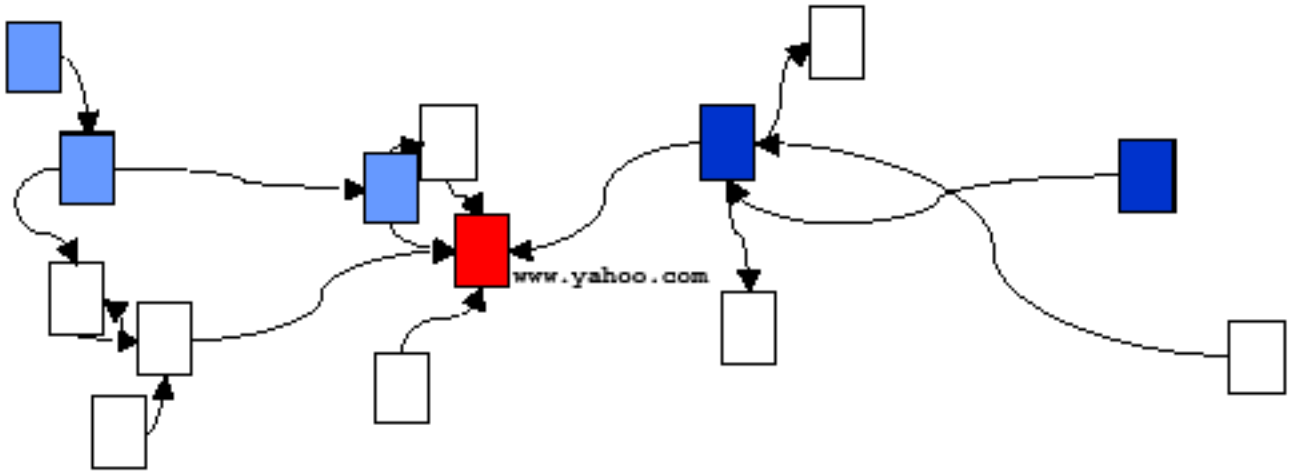
The Web as a Markov Chain



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 *ki* links then it has transition probability (1-*q*) /*ki* + *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

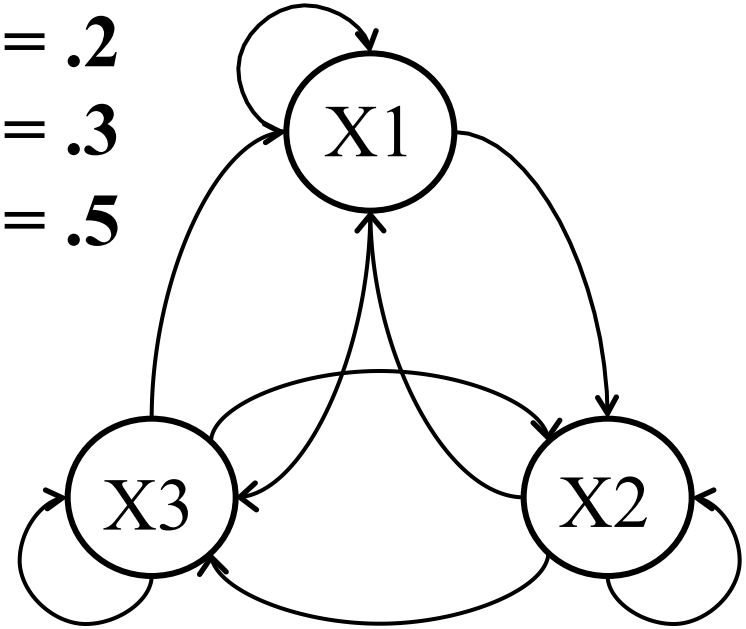
Sergey Brin and Lawrence Page , The anatomy of a large-scale hypertextual {Web} search engine, Computer Networks and ISDN Systems, 1998

Robert Collins
CSE586, PSU

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

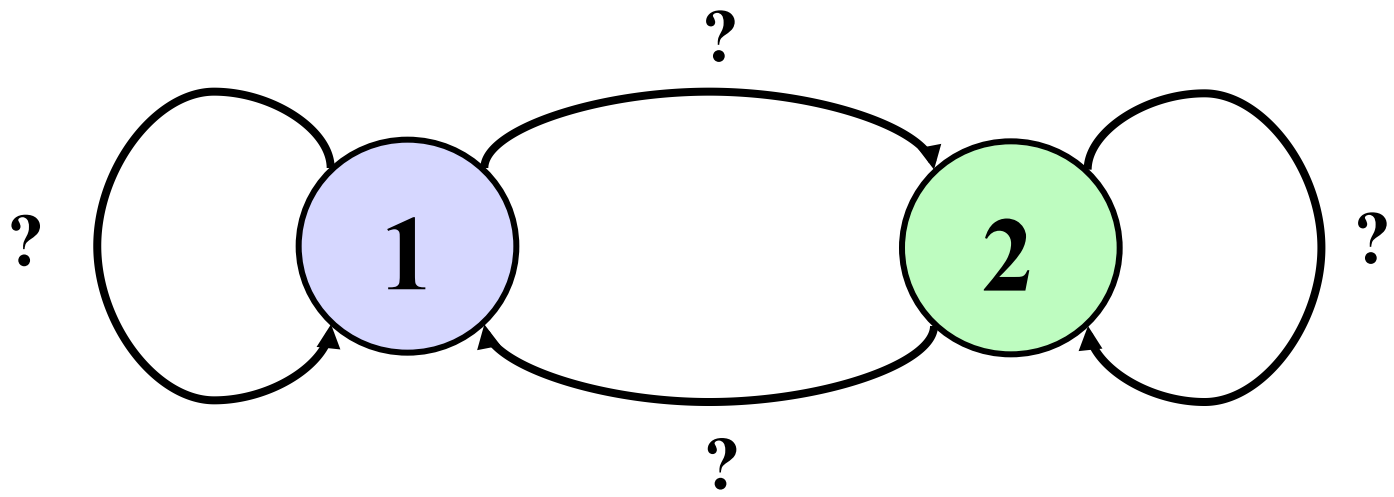


$$K = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

Robert Collins
CSE586, PSU

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)



Robert Collins
CSE586, PSU

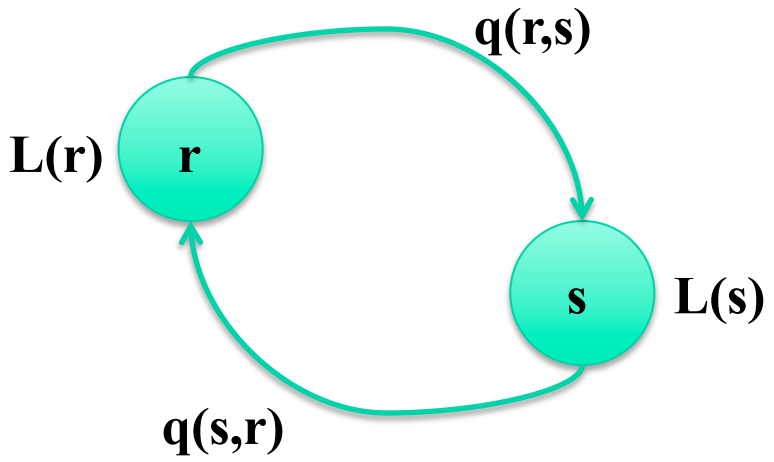
Detailed Balance

- 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"



Robert Collins
CSE586, PSU

Detailed Balance

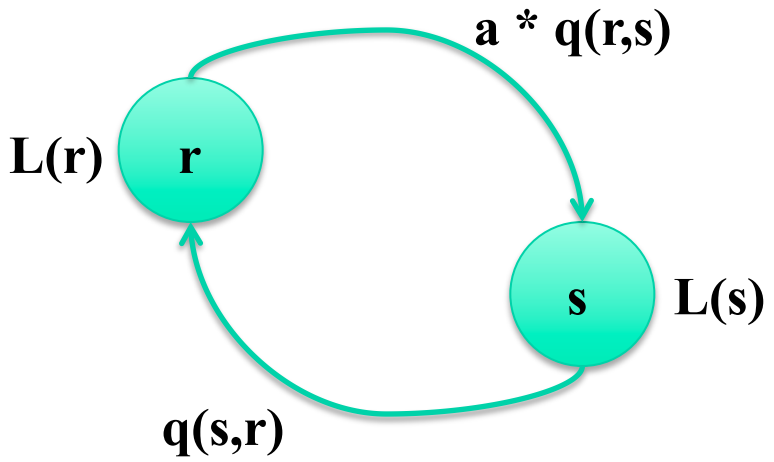
- In practice, you just propose some transition probabilities.
- They typically will NOT satisfy detailed balance (unless you are extremely lucky).
- Instead, you "fix them" by introducing a computational fudge factor

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

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

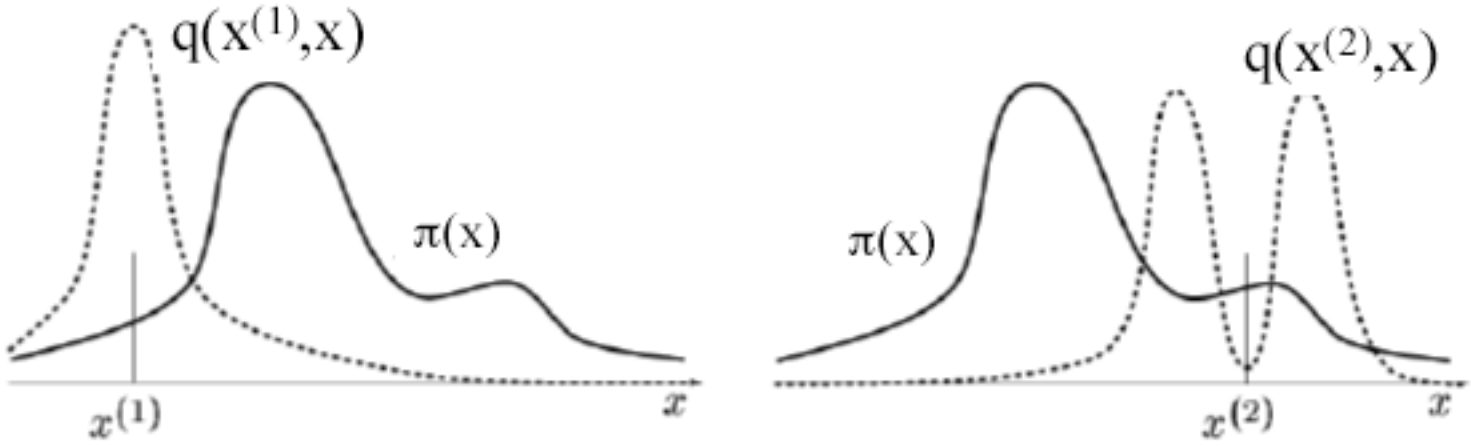
3. if a>1 accept $x^{(t+1)}=x'$
   else accept with probability a
   if rejected: $x^{(t+1)}=x^{(t)}$

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.

Proposal Density q(x,x')

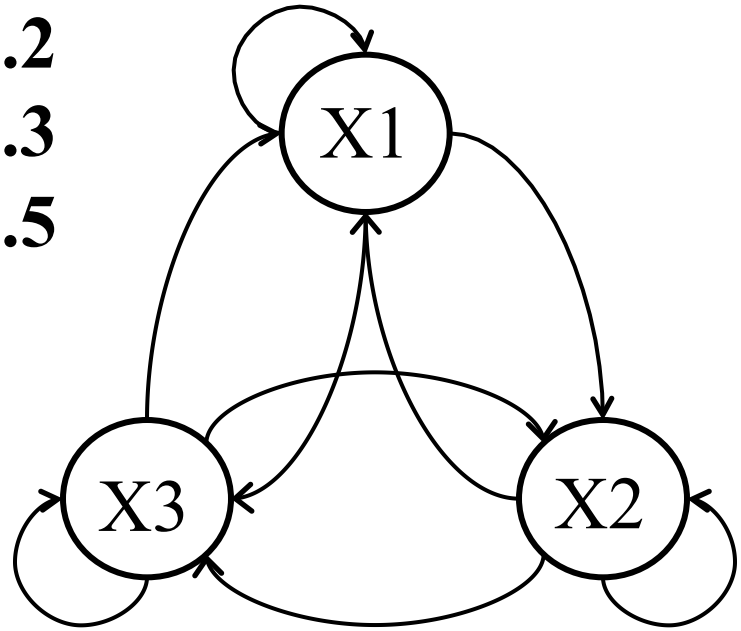
Note: the transition probabilities $q(x^{(t)},x)$ can be arbitrary distributions. They can depend on the current state and change at every time step, if you want.



Robert Collins
CSE586, PSU

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

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

cancels

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

Robert Collins
CSE586, PSU

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

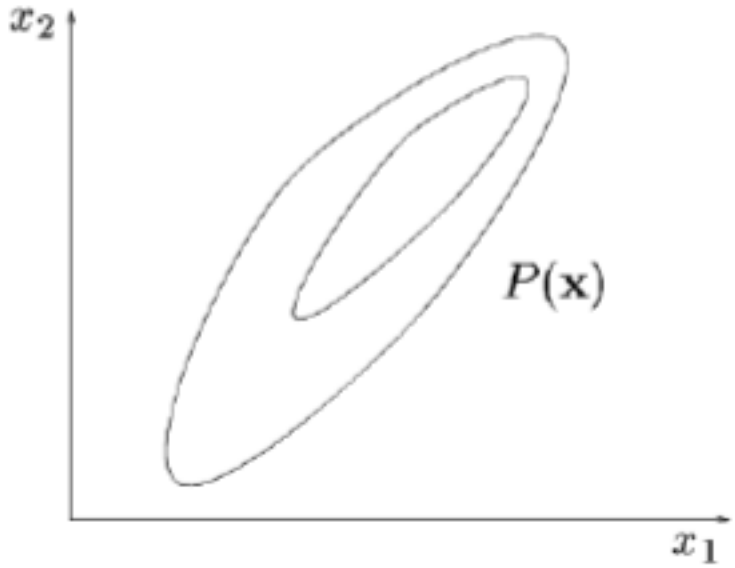
$$\begin{aligned} \alpha(\mathbf{x}, \mathbf{y}) &= \frac{\pi(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{\pi(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})} \\ &= \frac{\pi(\mathbf{y})/\pi(\mathbf{y}_i|\mathbf{x}_{(i)})}{\pi(\mathbf{x})/\pi(\mathbf{x}_i|\mathbf{y}_{(i)})} \\ &= \frac{\pi(\mathbf{y})/\pi(\mathbf{y}_i|\mathbf{y}_{(i)})}{\pi(\mathbf{x})/\pi(\mathbf{x}_i|\ma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

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

simpler version, using 1D conditional distributions

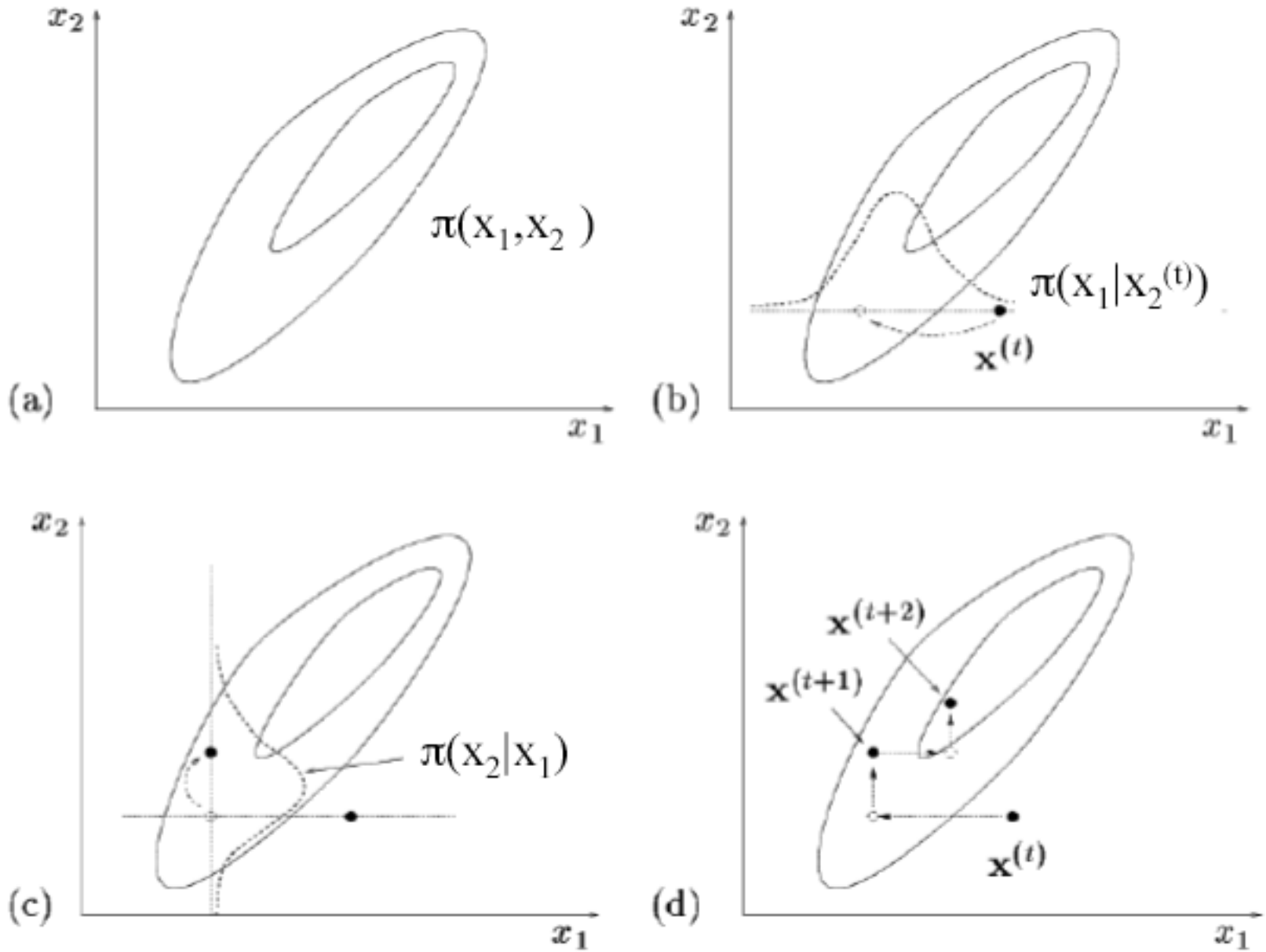
Gibbs Sampling

- 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 ...**

Robert Collins
CSl



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.

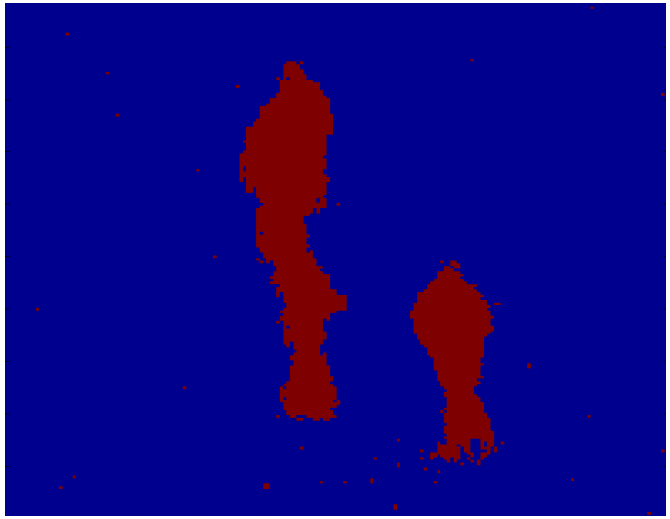
Trans-dimensional MCMC

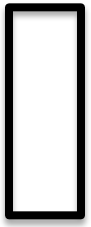
- Exploring alternative state spaces of differing dimensions (example, when doing EM, also try to estimate number of clusters along with parameters of each cluster).
- Green's reversible-jump approach (RJMCMC) gives a general template for exploring and comparing states of differing dimension.

Example: 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.**

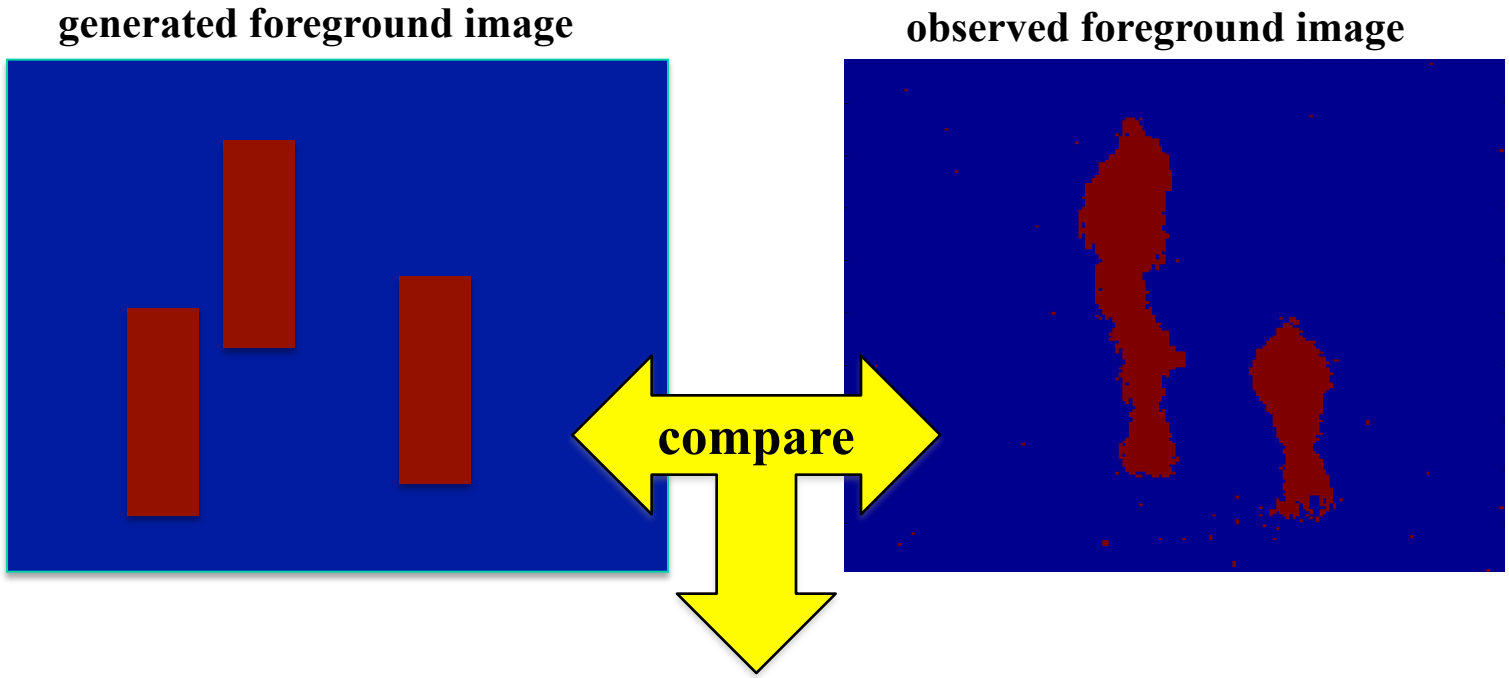
W.Ge and R.Collins, "Marked Point Processes for Crowd Counting," *IEEE Computer Vision and Pattern Recognition (CVPR'09),* Miami, FL, June 2009, pp.2913-2920.

Robert Collins
CSE586, PSU

Likelihood Score

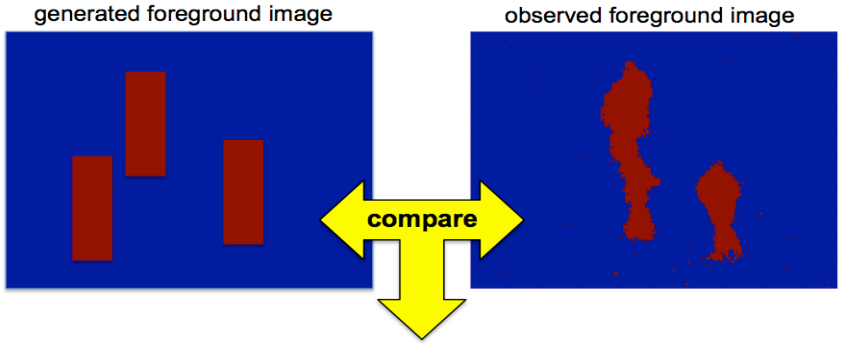
To measure how "good" a proposed configuration is, we generate a foreground image from it and compare with the observed foreground image to get a likelihood score.

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,$theta_3$}}



Robert Collins
CSE586, PSU

Likelihood Score



Bernoulli distribution model

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

$$
\begin{aligned}
c_{00} &= \text{count of pixels where observation is background and label is background} \\
c_{01} &= \text{count of pixels where observation is background and label is foreground} \\
c_{10} &= \text{count of pixels where observation is foreground and label is background} \\
c_{11} &= \text{count of pixels where observation is foreground and label is foreground}
\end{aligned}
$$

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

$$p_{00} = p_{11} = \mu \quad \text{and} \quad p_{01} = p_{10} = 1 - \mu$$

log likelihood

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

Number of pixels that disagree!

Robert Collins
CSE586, PSU

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.

$$\text{config}_k = \{\{x_1,y_1,w_1,h_1,\text{theta}_1\},\{x_2,y_2,w_2,h_2,\text{theta}_2\},\ldots,\{x_k,y_k,w_k,h_k,\text{theta}_k\}\}$$

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

Size of $\text{config}_k = N^k$
And we don't even know how many people there are...

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

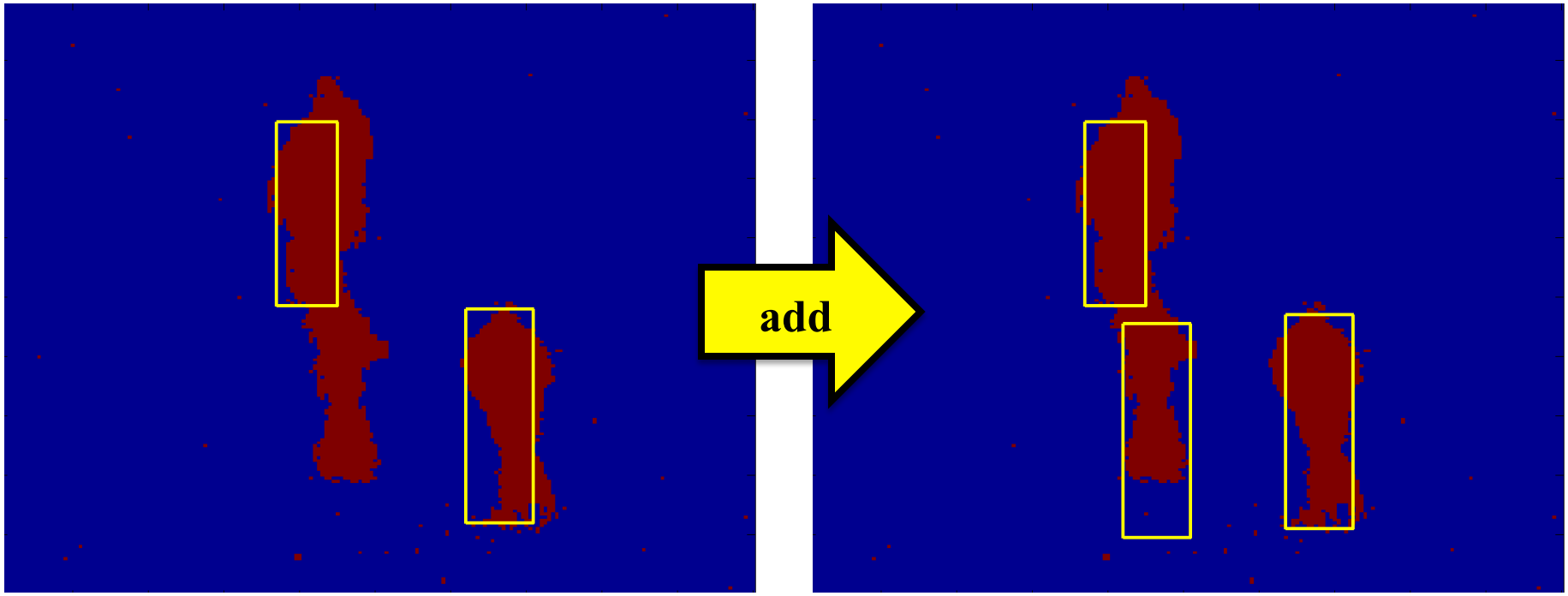
If we also wanted to search for width, height and orientation, this space would be even more huge.

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)



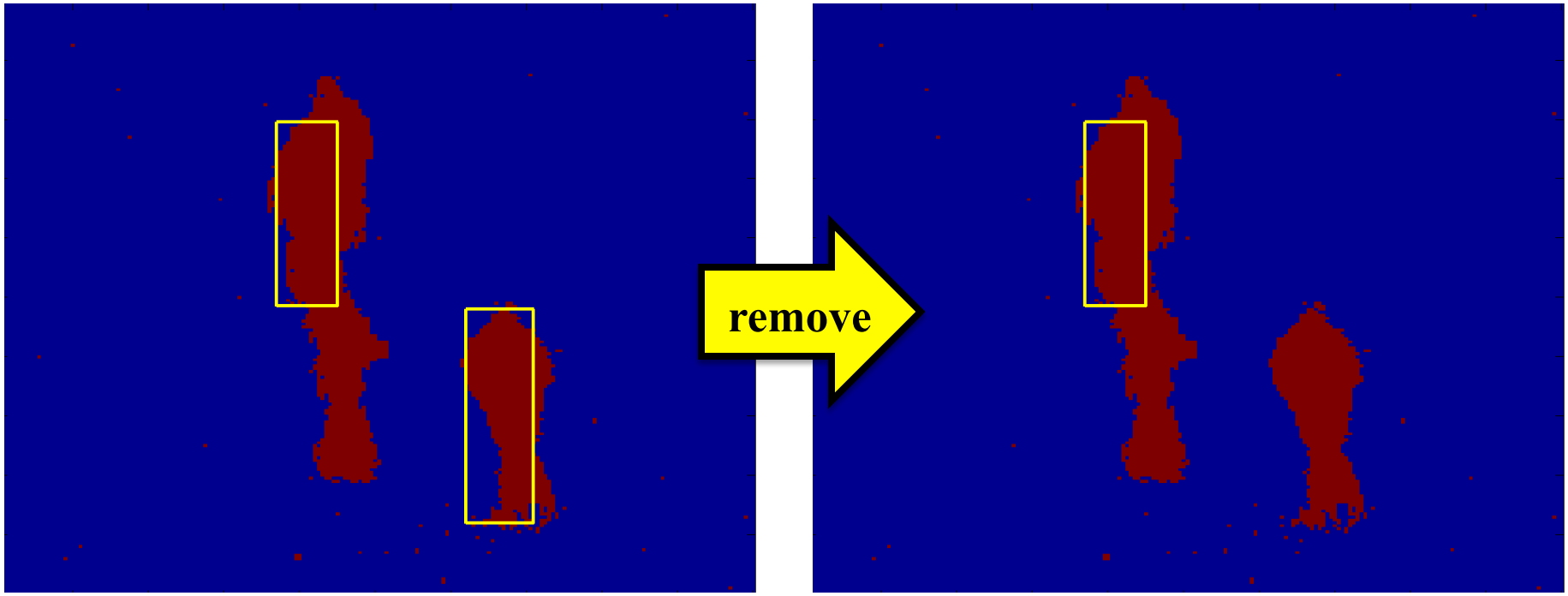
current configuration

proposed configuration

Robert Collins
CSE586, PSU

Proposals

- Remove a rectangle (death)

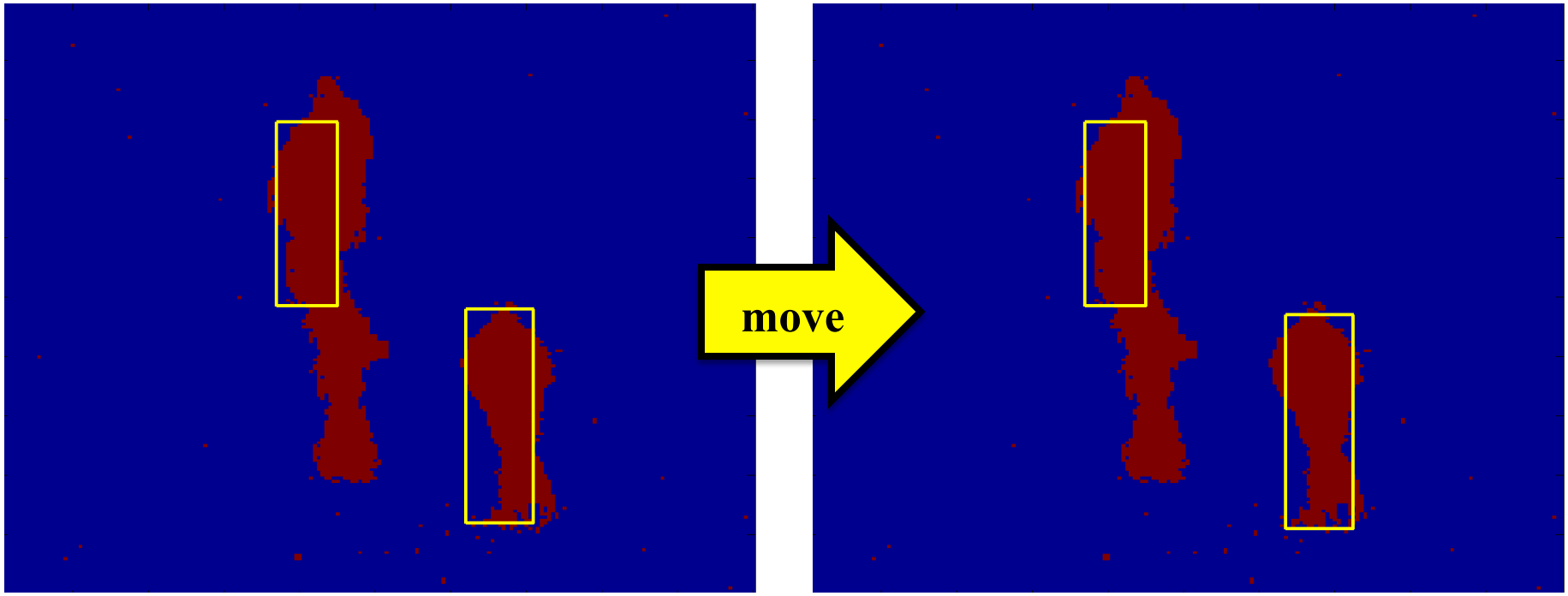


current configuration

proposed configuration

Proposals

- Move a rectangle



current configuration

proposed configuration

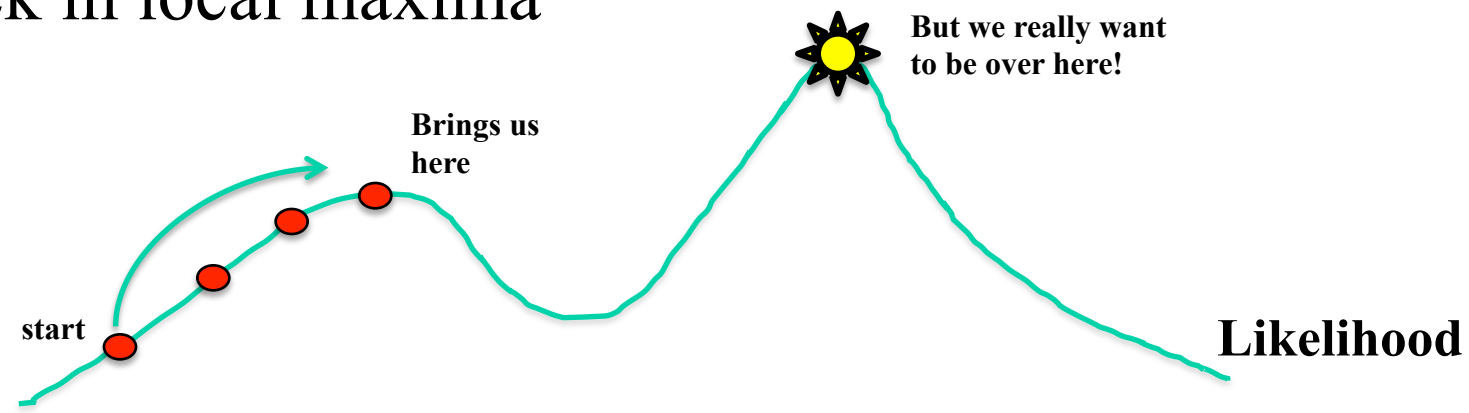
Searching for the Max

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

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

Accept if a > 1

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



Robert Collins
CSE586, PSU

MCMC Sampling

- Metropolis Hastings algorithm

  Propose a new configuration

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

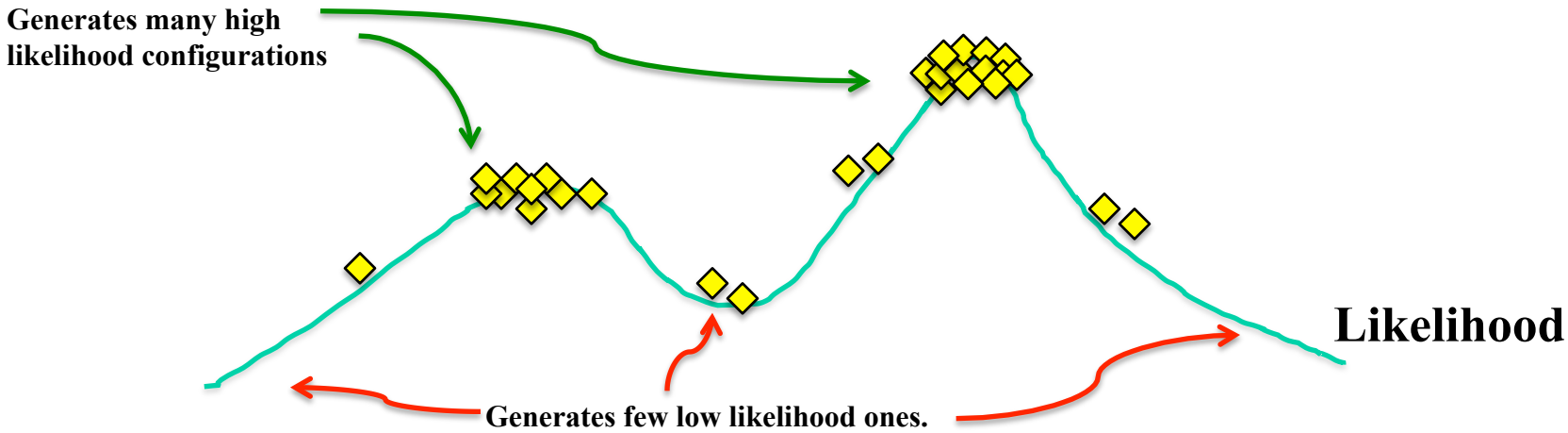
  Accept if a > 1

  Else accept anyways with probability a

Difference from Naïve algorithm

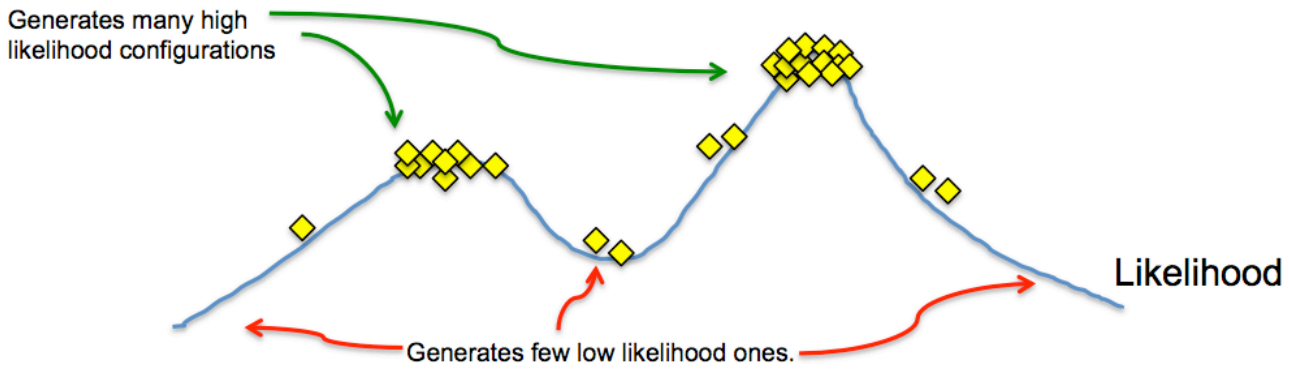
Searching for the Max

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



Searching for the Max

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

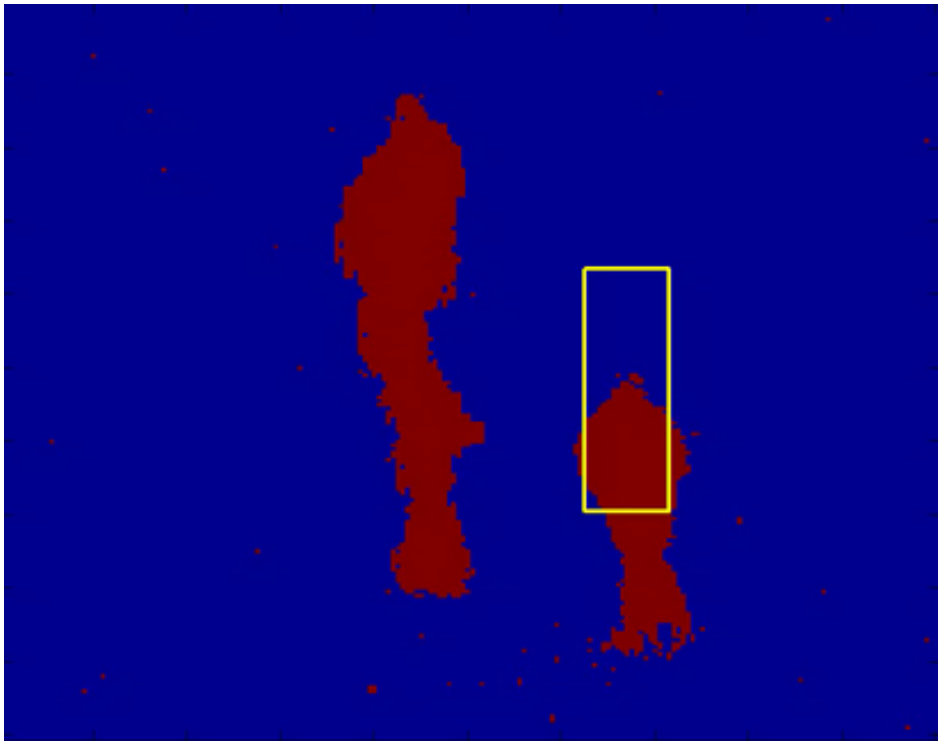


  - **This searches the space of configurations in an efficient way.**
  - Now just remember the generated configuration with the highest likelihood.

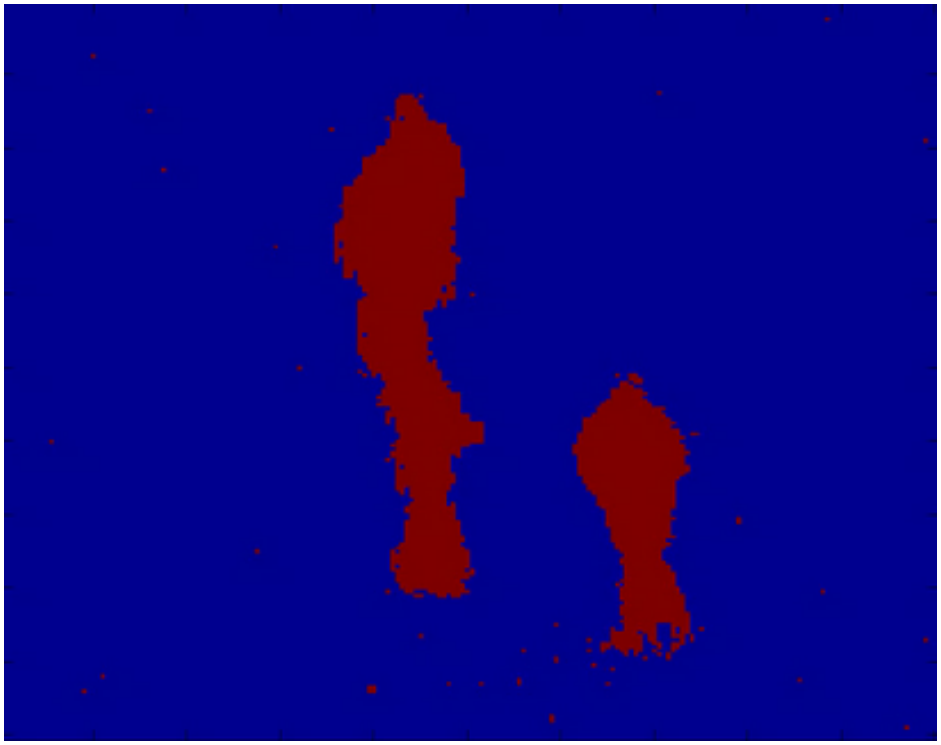
Robert Collins
CSE586, PSU

MCMC in Action

Sequence of proposed configurations



Sequence of accepted configurations

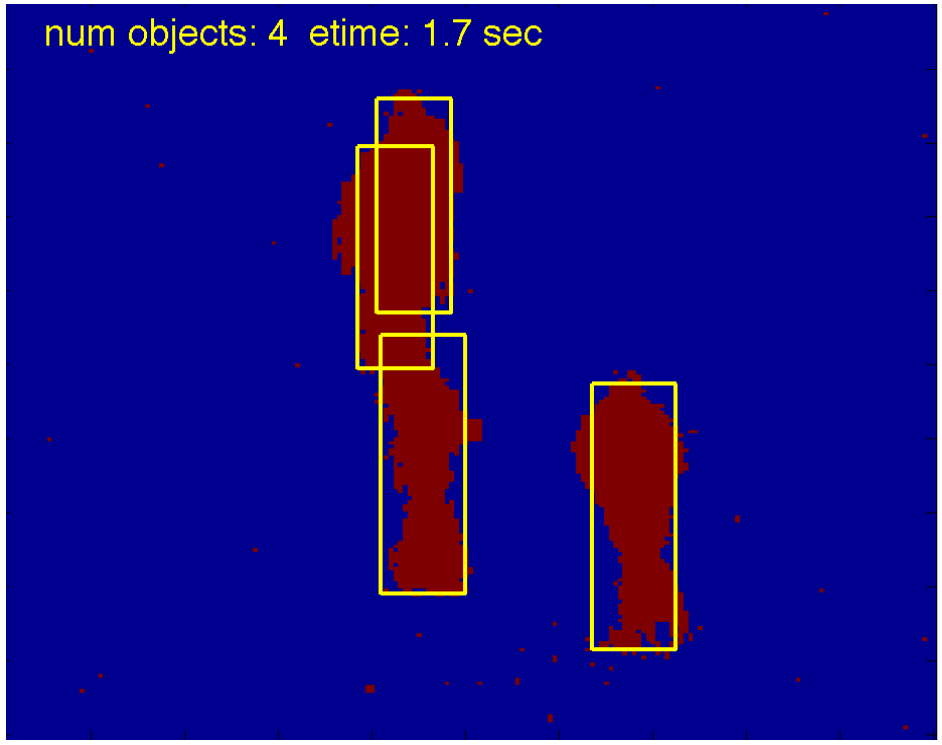


movies

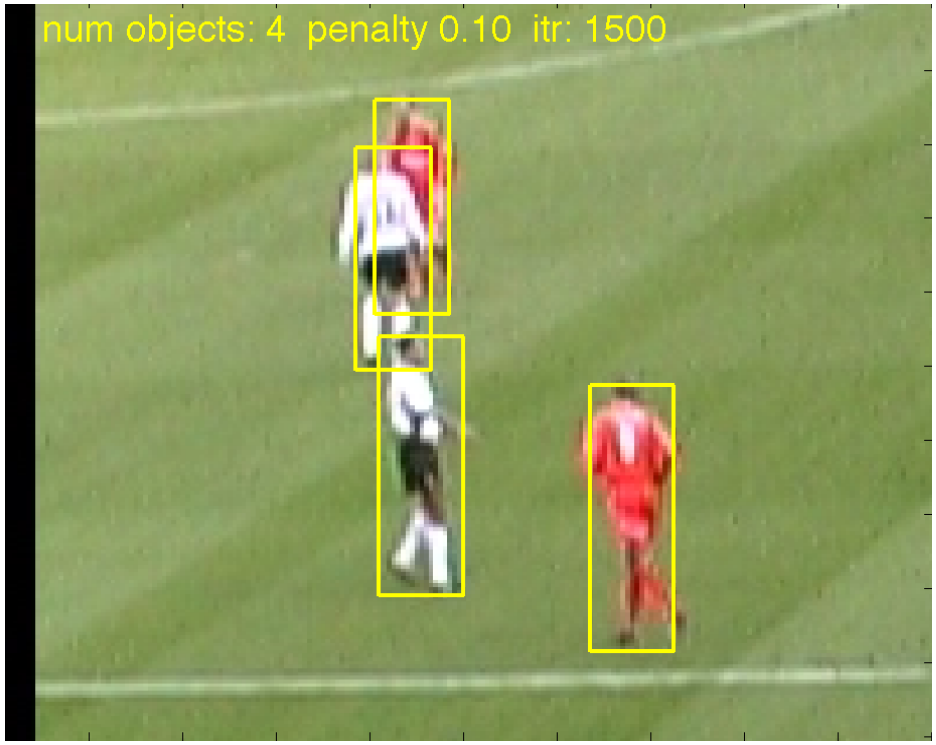
Robert Collins
CSE586, PSU

MCMC in Action

Max likelihood configuration



Looking good!



Robert Collins
CSE586, PSU

Examples