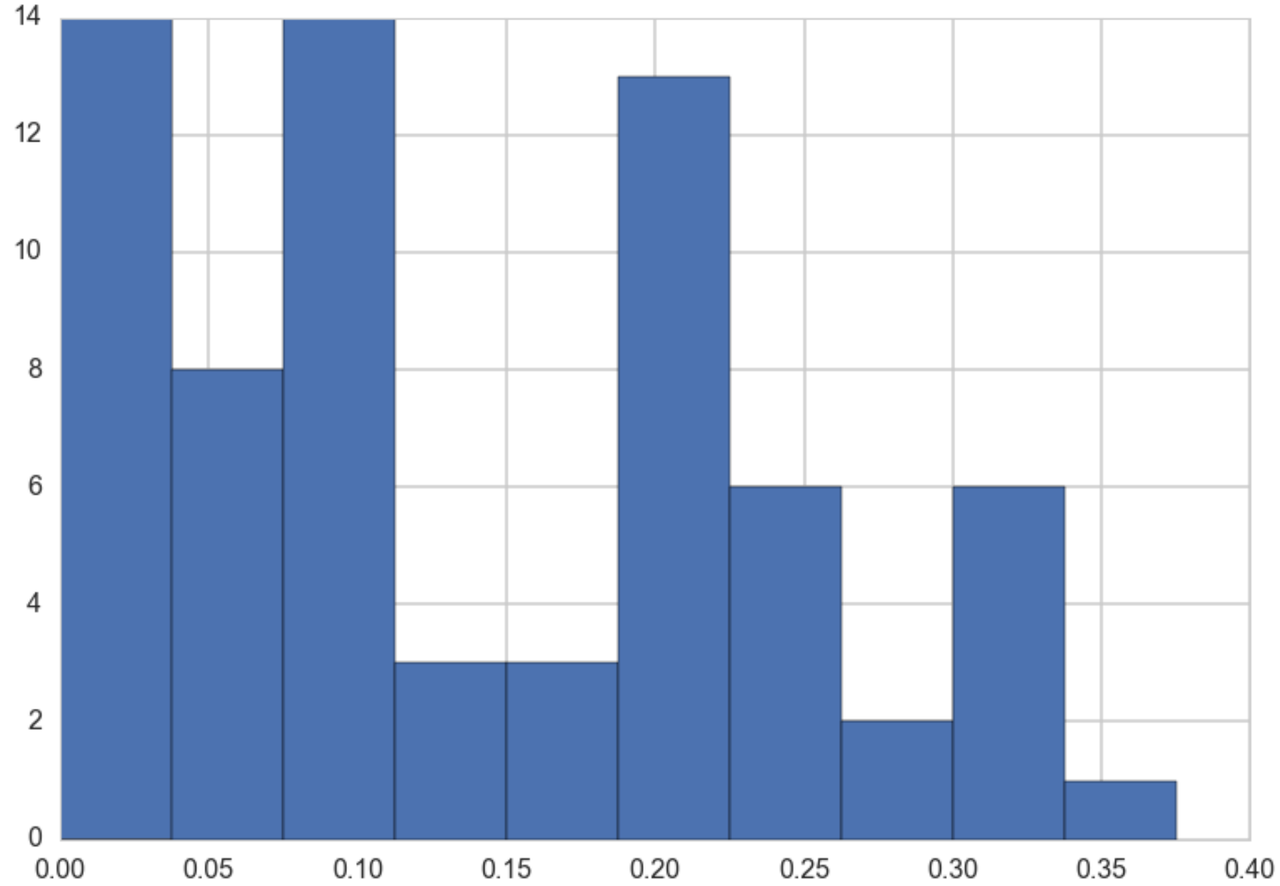
Lecture 15

Metropolis and Gibbs Samplers

Rat Tumors



- tumors in female rats of type "F344" that recieve a particular drug, in 70 different experiments.
- mean and variance of tumor incidence: `0.13600653889043893`, `0.010557640623609196`
- 71st experiment done: 4 out of 14 rats develop tumors. Estimate the risk of tumor in the rats in the 71st experiment

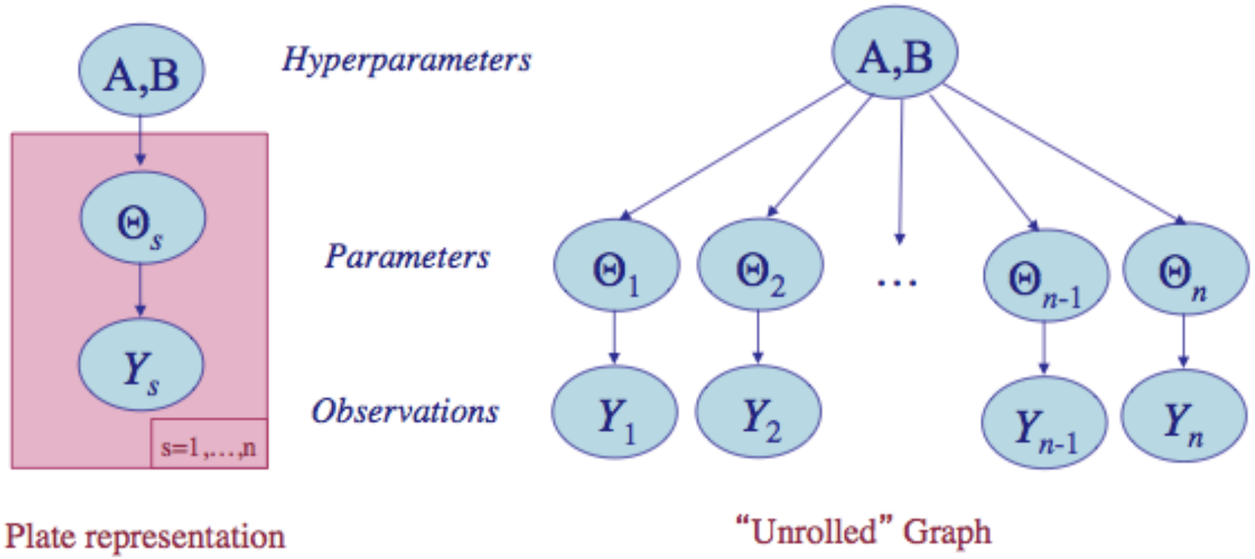
Modeling

$$p(y_i|\theta_i; n_i) = Binom(n_i, y_i, \theta_i)$$

$$p(Y|\Theta; \{n_i\}) = \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Need to choose a prior $p(\Theta)$.

Partial pooling: Hierarchical Model



θ_is drawn from "population distribution" given by a conjugate Beta prior $Beta(\alpha, \beta)$ with **hyperparameters** α and β.

$$\theta_i \sim Beta(\alpha, \beta).$$

$$p(\Theta|\alpha, \beta) = \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta).$$

Priors from data

Where do α and β come from?

Why are we calling them hyperparameters?

So far have assumed α and β known in priors to be weakly informative.

New idea: estimate priors from data. Looks like a cross-validation like setup.

Key Idea: Share statistical strength

- Some **units** (experiments) statistically more robust
- Non-robust experiments have smaller samples or outlier like behavior
- Borrow strength from all the data as a whole through the estimation of the hyperparameters
- **regularized partial pooling model** in which the "lower" parameters (θs) tied together by "upper level" hyperparameters.

First idea: estimate directly from data

Posterior-predictive distribution, as a function of upper level parameters $\eta = (\alpha, \beta)$.

$$p(y^*|D, \eta) = \int d\theta \, p(y^*|\theta) \, p(\theta|D, \eta)$$

A likelihood with parameters η and simply use maximum-likelihood with respect to η to estimate these η using our "data" y^*

Called Empirical Bayes or Type-2 MLE

- MLE with respect to η
- involves an optimization
- unlike cross-validation, θs not-yet estimated on training set.
- indeed we marginalize over θs so can use training set.
- in practice often match moments of predictive or posterior

EB for rats: prior/prior predictive...

Consider the prior expectation and variance:

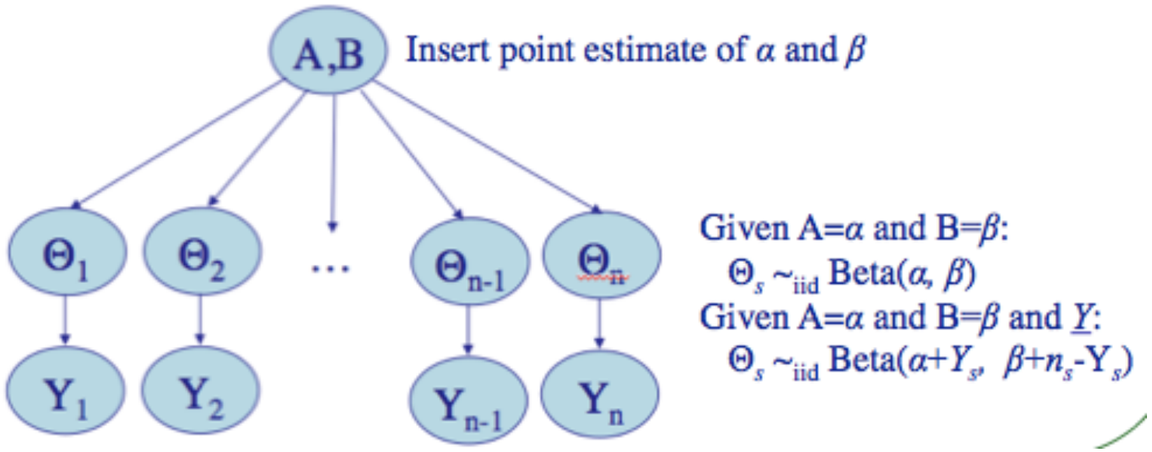
$$\mu = \frac{\alpha}{\alpha+\beta}, V = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Match empirical mean and variance on y_i/n_i

- Need to be careful what "space" you are working in, predictive (y) or not
- Use prior predictive if in a "predictive space":

$$p(y^*) = E_{p(\theta}[p(y^*|\theta)] = \int d\theta p(y^*|\theta)p(\theta).$$

...to posterior/posterior predictive...

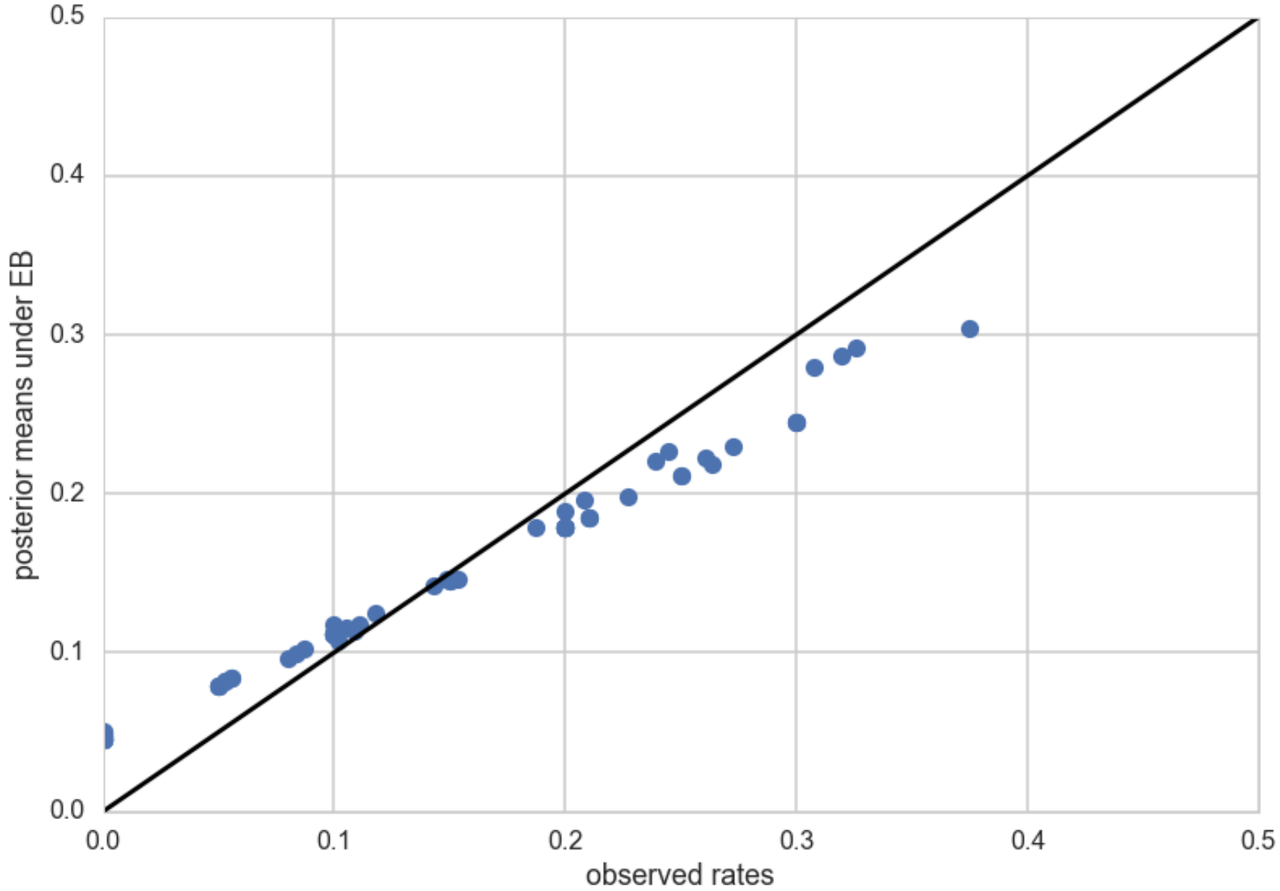


- (α, β) = (`1.3777748392916778`, `8.752435447153129`)
- Conditional posterior distribution for each of the θ_i, given everything else is Beta:.

$$p(\theta_i|y_i, n_i, \alpha, \beta) = Beta(\alpha + y_i, \beta + n_i - y_i)$$

$$\bar{\theta}_{post,i} = \frac{\alpha + y_i}{\alpha + \beta + n_i}$$

Shrinkage in rat (tumors)



Posterior estimates shrink towards full pooling.

Now, for the 71st experiment, we have 4 out of 14 rats having tumors. The posterior estimate for this would be

$$\frac{\alpha + y_{71}}{\alpha + \beta + n_{71}}$$

```
4/14, (4+a_est)/(14+a_est+b_est)
= (0.2857142857142857, 0.22286481449822493)
```

Full Bayesian

- every optimization is a chance to overfit, would like to use integration all the way
- specify a **hyper-prior** $p(\eta)$ ($p(\alpha, \beta)$) on these hyperparameters η (α, β)
- helps us develop a computational strategy of gibbs sampling
- allows estimates of the probabilities of any one unit to borrow strength from all the data as a whole

Fully Bayesian Rat tumors

Joint Posterior:

$$p(\Theta, \alpha, \beta | Y, \{n_i\}) \propto p(\alpha, \beta) \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta) \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Need to move ALL OVER this posterior. BUT, it is not easy to sample from.

Ancestral Sampling

- sample from graph, that is from prior, then conditional prior, then data-distribution
- but not enough to sample from posterior! What we want to do is to to restrict all possible samples we got this way by:
- CONDITIONING on the data, and getting only those samples consistent with this conditioning

MCMC

to the rescue

Basic Idea of Markov Chain Monte Carlo (MCMC)

Move all over p by identifying $E = -log(p)$ with the energy of an imaginary physical system. Thus $p = exp(-E)$.

Move from x_i to x_j via a **proposal** q.

If the new state has lower energy, or higher probability, accept x_j.

If the new state has higher energy, accept x_j, with probability proportional to the ratio $p(x_j)/p(x_i)$ or $exp(-(E_j - E_i))$.

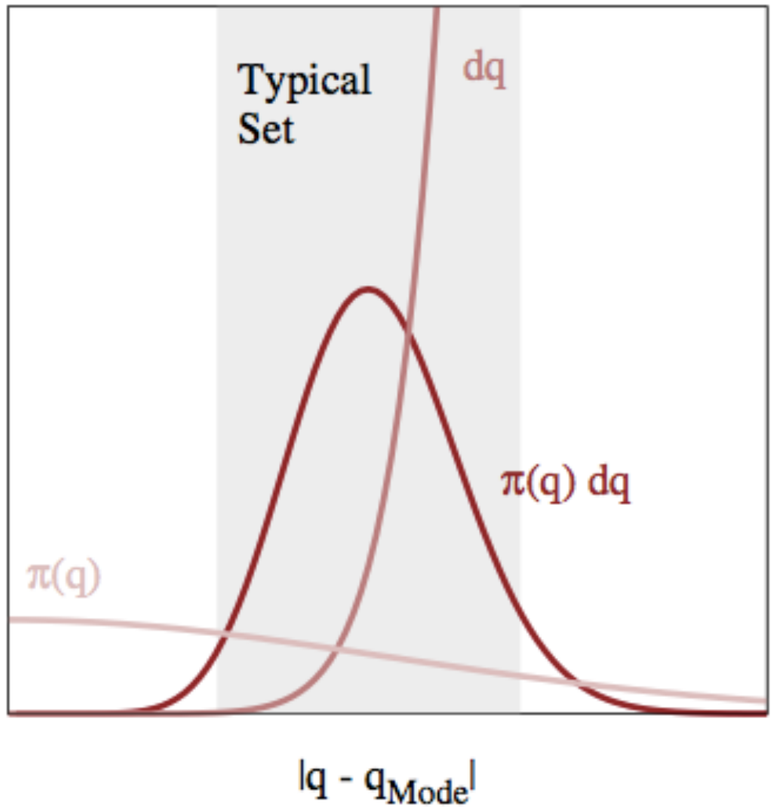
Today

- markov chains and MCMC
- Metropolis Sampler
- continuous pdf sampling
- discrete pmf sampling

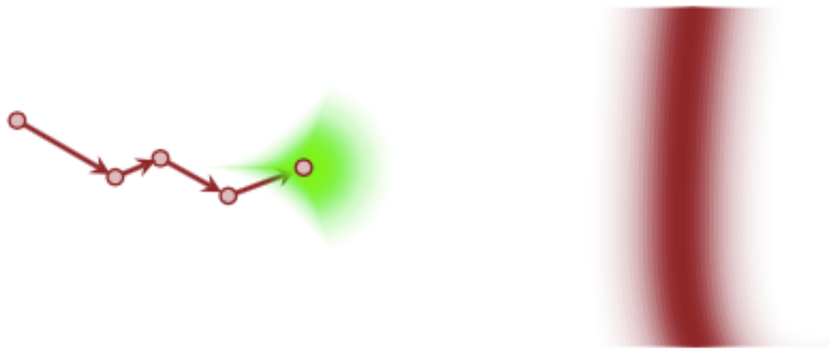
Next Time

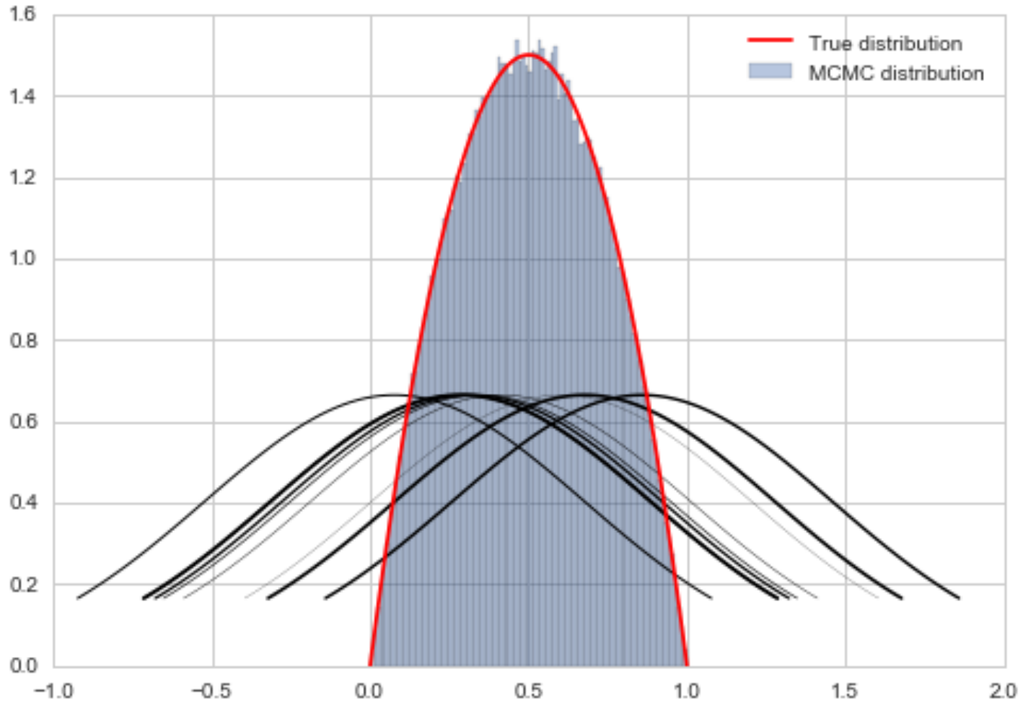
- markov chain theory
- metropolis and metropolis hastings
- gibbs sampling
- diagnostics

The Typical Set



Intuition: a particle approaches typical set





Instead of sampling p we sample q, yielding a new state, and a new proposal distribution from which to sample.

Metropolis

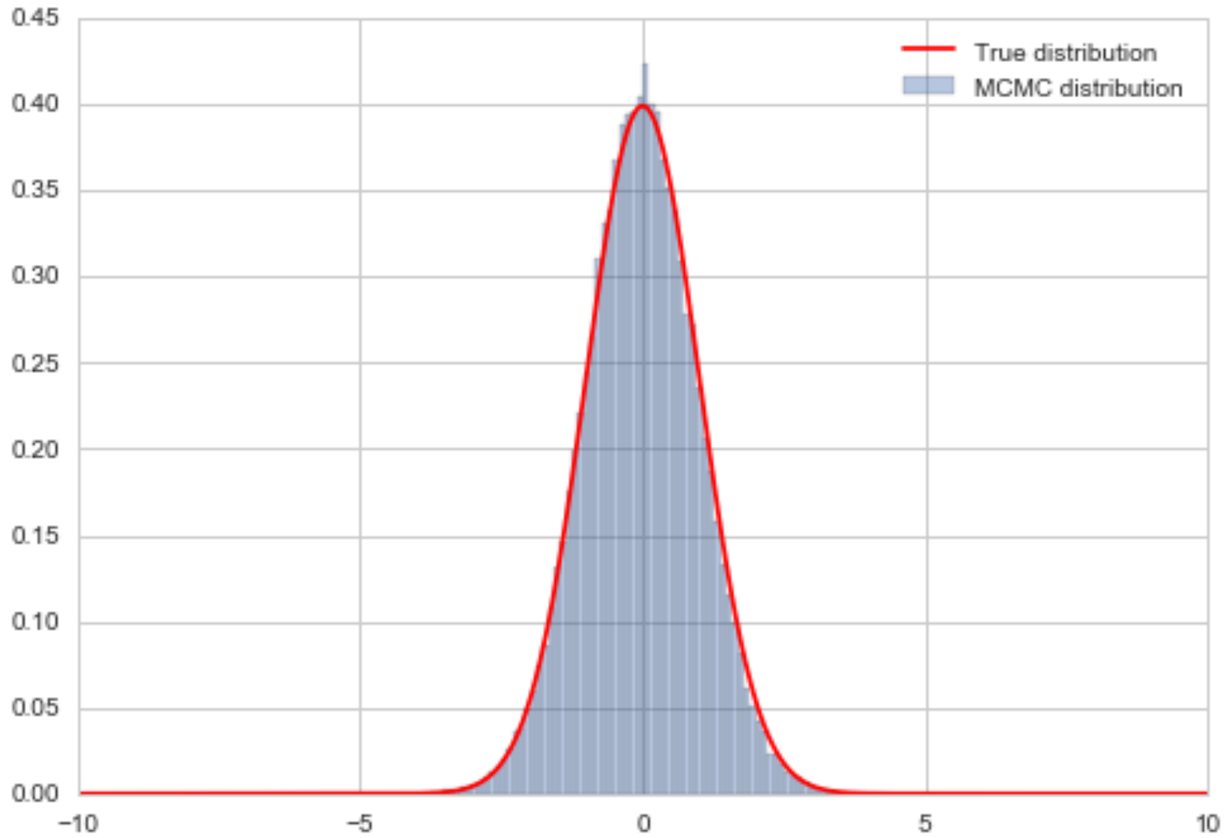
1. use a proposal distribution to propose a step.
2. Then we calculate the pdf at that step, and compare it to the one at the previous step.
3. If the probability increased (energy decreased) we accept. If probability decreased (energy increased) we accept some of the time.
4. Accumulate our samples.

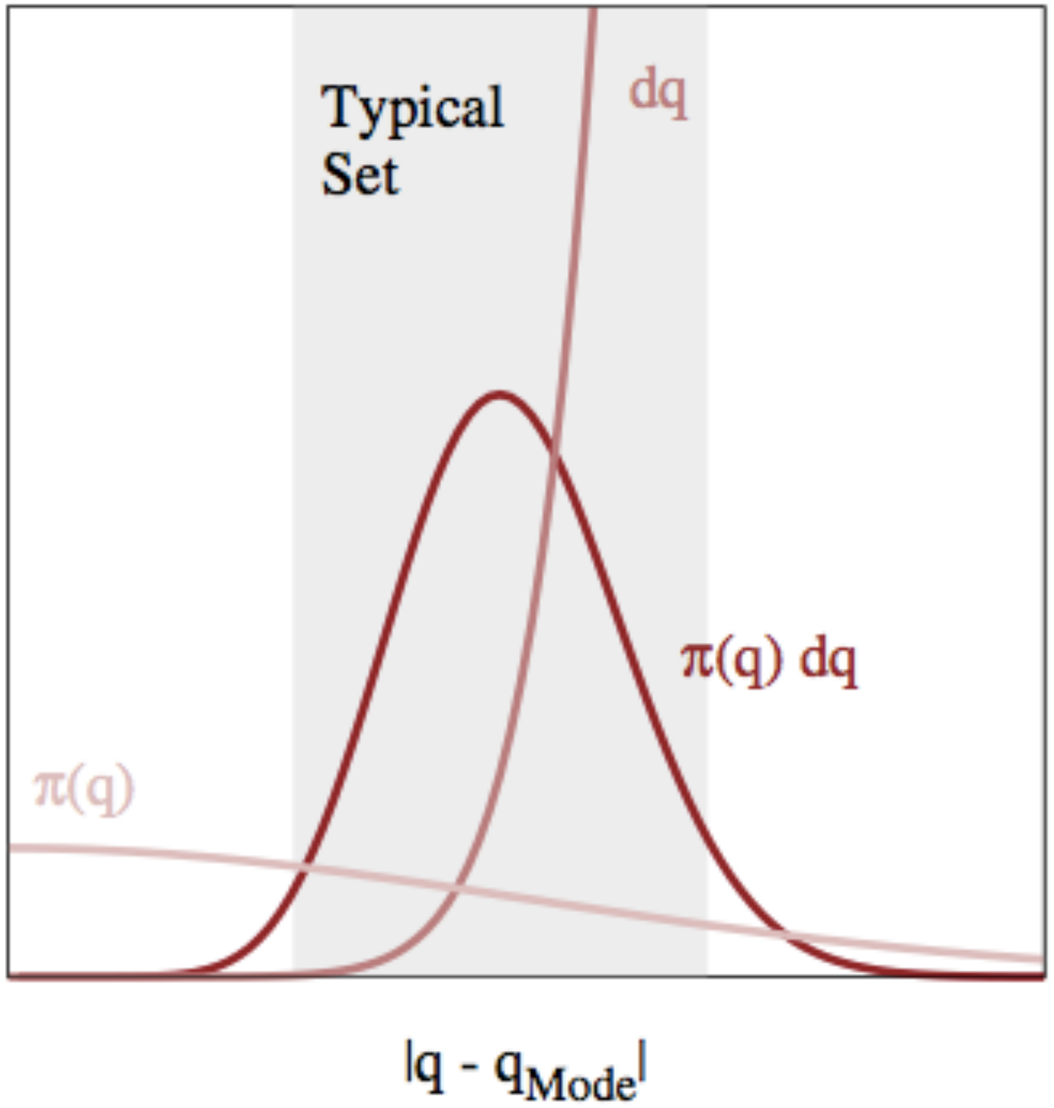
```
def metropolis(p, qdraw, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    for i in range(nsamp):
        x_star = qdraw(x_prev)
        p_star = p(x_star)
        p_prev = p(x_prev)
        pdfratio = p_star/p_prev
        if np.random.uniform() < min(1, pdfratio):
            samples[i] = x_star
            x_prev = x_star
        else:#we always get a sample
            samples[i]= x_prev

    return samples
```

Uniform Proposal to sample the standard gaussian

```
from scipy.stats import uniform
def propmaker(delta):
    rv = uniform(-delta, 2*delta)
    return rv
uni = propmaker(0.5)
def uniprop(xprev):
    return xprev+uni.rvs()
```

Sampling from gaussian with uniform proposal

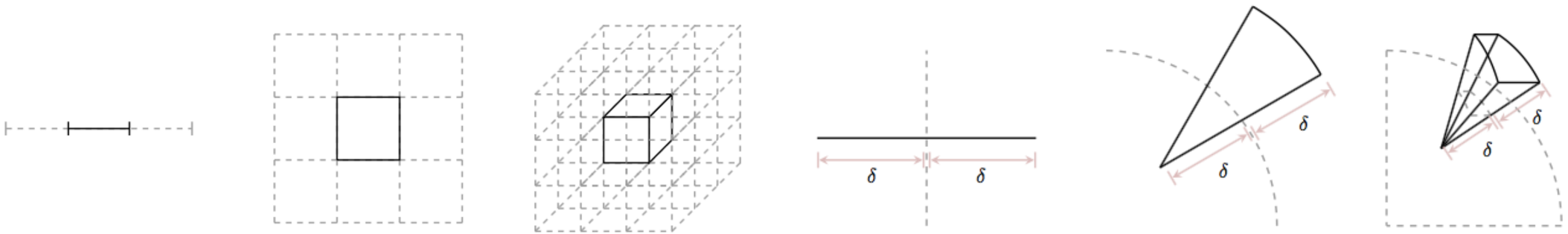




Why do this?

- Why not rejection? wasteful
- more wasteful in higher dimensions
- curse of dimensionality in higher dimensions
- volume around mode gets smaller
- interplay of density and volume

Curse of dimensionality



as dimensionality increases, center is lower volume, outside has more volume

Markov Chain

$$T(x_n|x_{n-1}, x_{n-1}\ldots, x_1) = T(x_n|x_{n-1})$$

- non IID, stochastic process
- but one step memory only
- widely applicable, first order equations

T is a transition probability, so that $\int T(x_n|x_{n-1})dx_{n-1} = 1.$

Examples of Markov Chains

- discretized first order differential equations
- snakes and ladders
- page rank
- certain spammy emails

Read about Markov vs Nekrasov here. Basically Markov proved a law of large humbers for Markov Chains.

Stationarity

$$sT = s \text{ or } \sum_i s_i T_{ij} = s_j \text{ or}$$

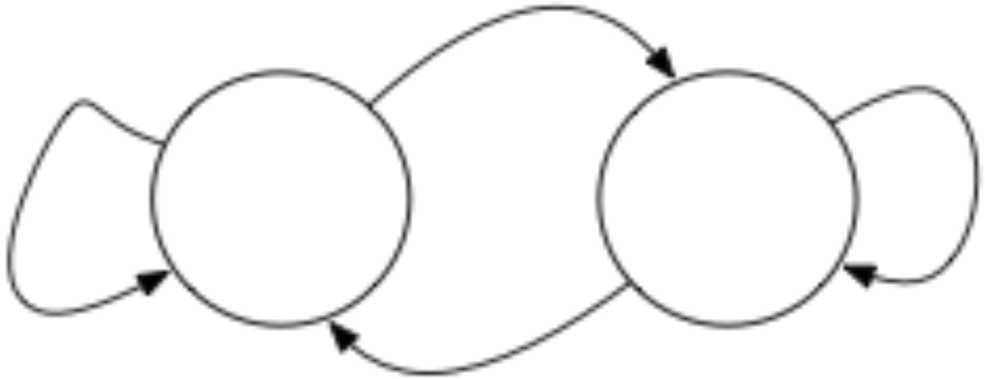
Continuous case: define T so that:

$$\int dx_i \, s(x_i) T(x_{i+1}|x_i) = s(x_{i+1}) \text{ then}$$

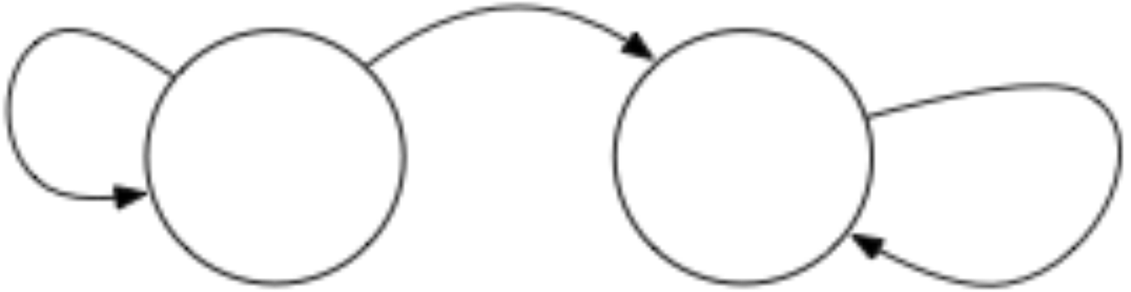
$$\int dx s(x) T(y|x) = \int p(y,x) dx = s(y)$$

Jargon

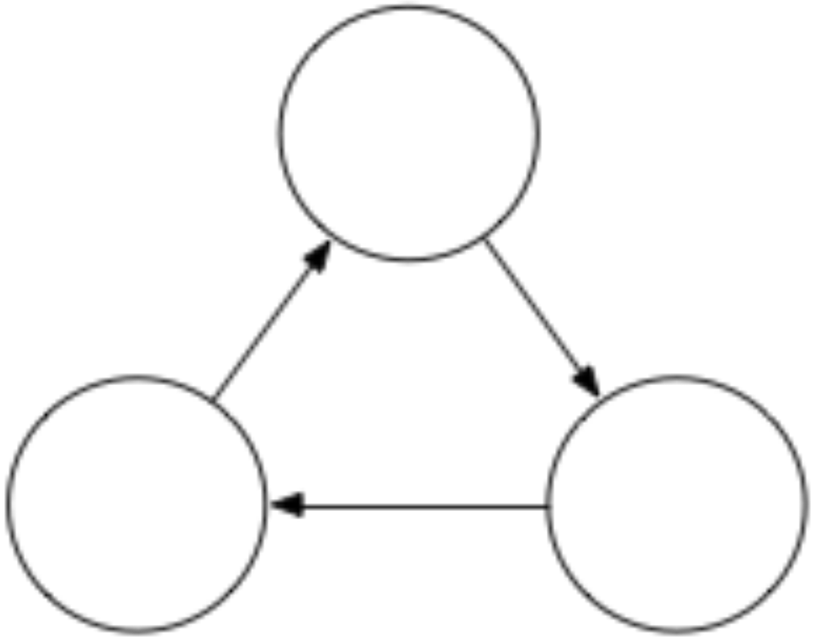
- **Irreducible**: can go from anywhere to everywhere
- **Aperiodic**: no finite loops
- **Recurrent**: visited repeatedly. Harris recurrent if all states are visited infinitely as $t \to \infty$.



Irreducible

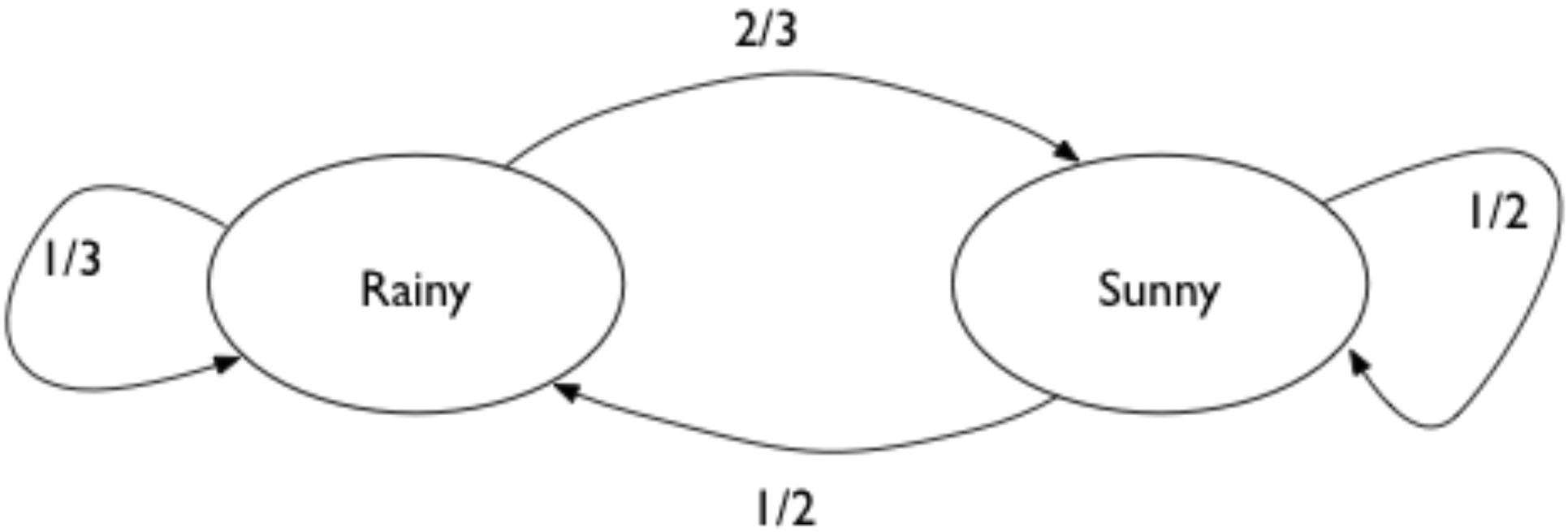


Reducible



Irreducible, but period 3

Rainy Sunny Markov chain



aperiodic and irreducible

Transition matrix, applied again and again

```
array([[ 0.33333333,  0.66666667],
       [ 0.5       ,  0.5       ]])


[[ 0.44444444  0.55555556]
 [ 0.41666667  0.58333333]]
-----------------
[[ 0.42592593  0.57407407]
 [ 0.43055556  0.56944444]]
-----------------
[[ 0.42901235  0.57098765]
 [ 0.42824074  0.57175926]]
-----------------
[[ 0.42849794  0.57150206]
 [ 0.42862654  0.57137346]]
-----------------
[[ 0.42858368  0.57141632]
 [ 0.42856224  0.57143776]]
```

Stationary distribution can be solved for:

Assume that it is $s = [p, 1-p]$

Then: $sT = s$

gives us

$$p \times (1/3) + (1-p) \times 1/2 = p$$

and thus $p = 3/7$

```
np.dot([0.9,0.1], tm_before): array([ 0.42858153,  0.57141847])
```

Stationarity, again

A irreducible (goes everywhere) and aperiodic (no cycles) markov chain will eventually converge to a stationary markov chain. It is the marginal distribution of this chain that we want to sample from, and which we do in metropolis (and for that matter, in simulated annealing).

$$\int dx s(x) T(y|x) = \int p(y,x) dx = s(y)$$

Detailed balance is enough for stationarity

$$s(x)T(y|x) = s(y)T(x|y)$$

If one sums both sides over x

$\int dx s(x)T(y|x) = s(y) \int dx T(x|y)$ which gives us back the stationarity condition from above.

Proposal, redux

- all the positions x in the domain we wish to minimize a function f over ought to be able to communicate: IRREDUCIBLE
- detailed balance: proposal is symmetric
- ensures $\{x_t\}$ generated by metropolis is a stationary markov chain with appropriate target.
- make sure proposal distributions are normalized

Are we done?

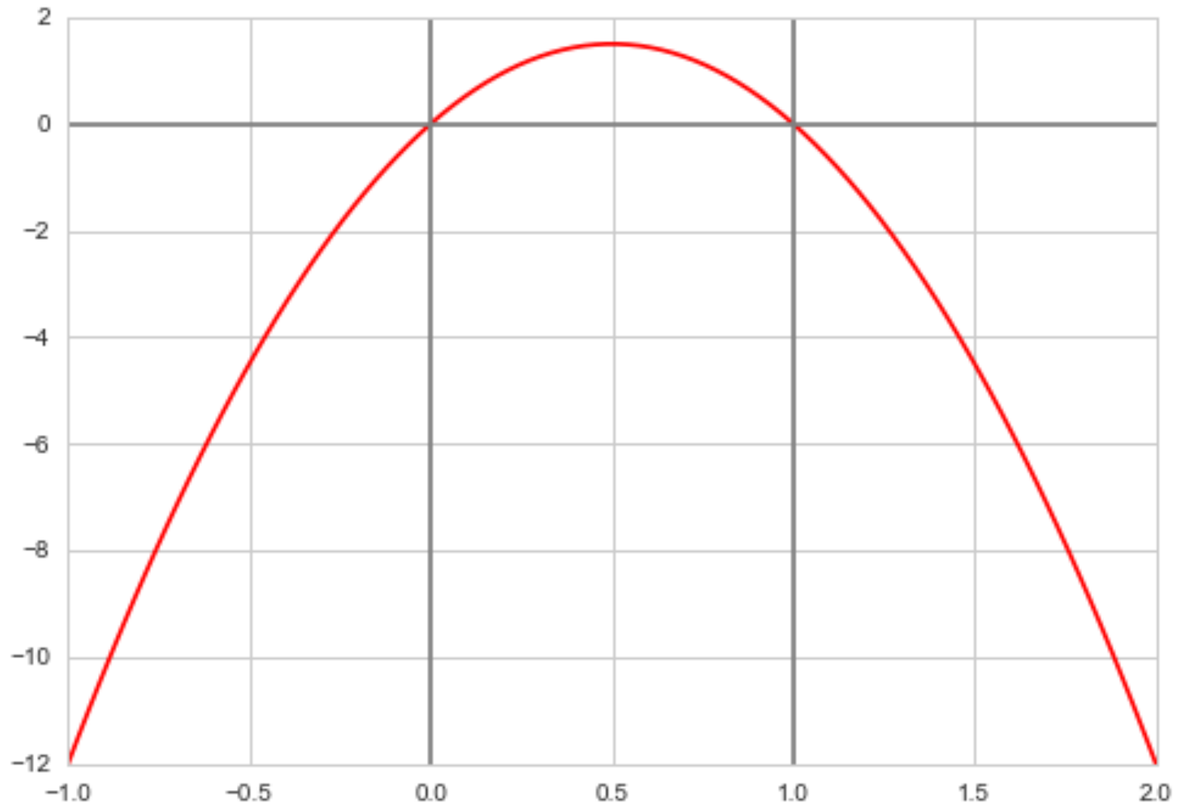
NO. we want to use law of large numbers. But our samples seem to be correlated, not IID.

Need a stronger condition, ergodicity.

And need to consider correlation.

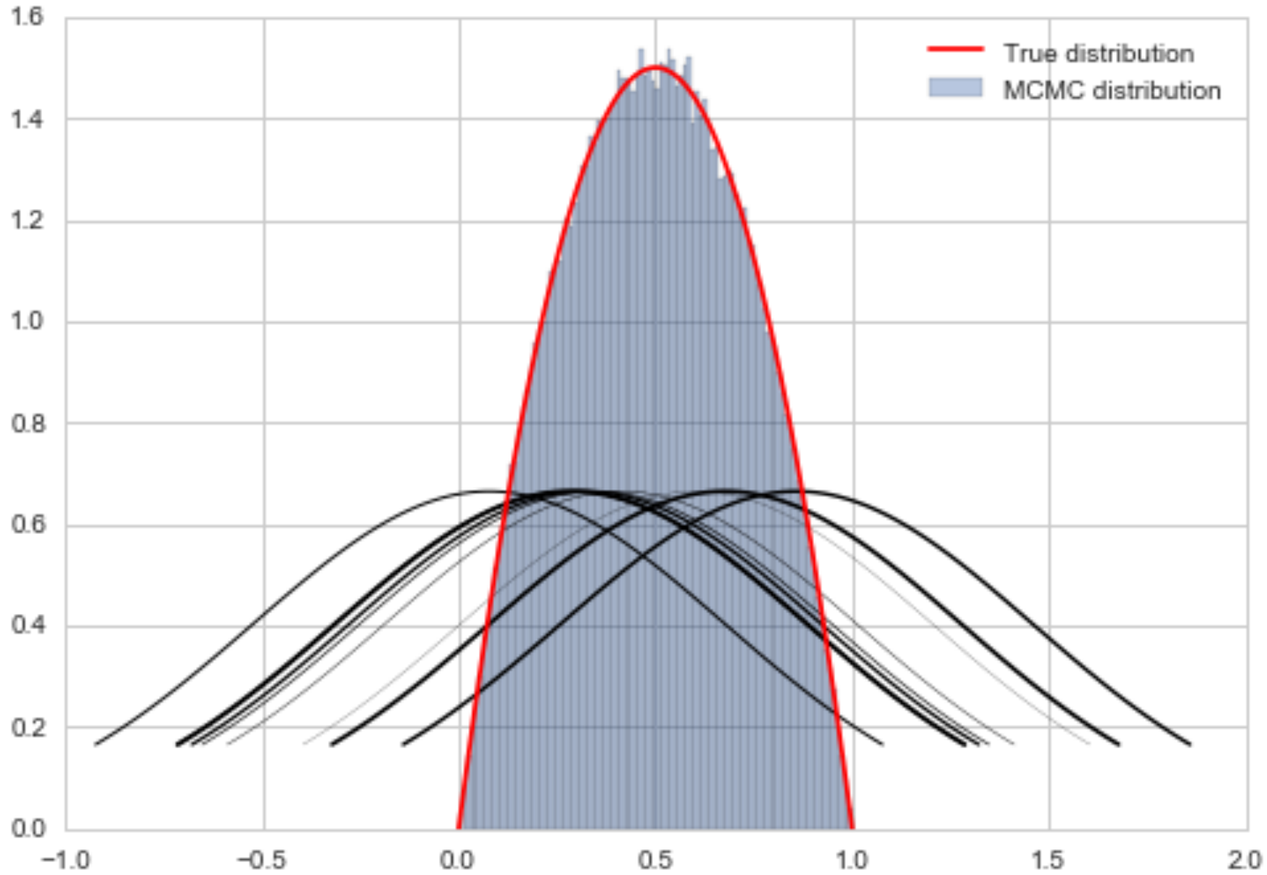
And a generalization to asymmetric proposals: Metropolis Hastings...

Another Example: sampling $6x(1-x)$

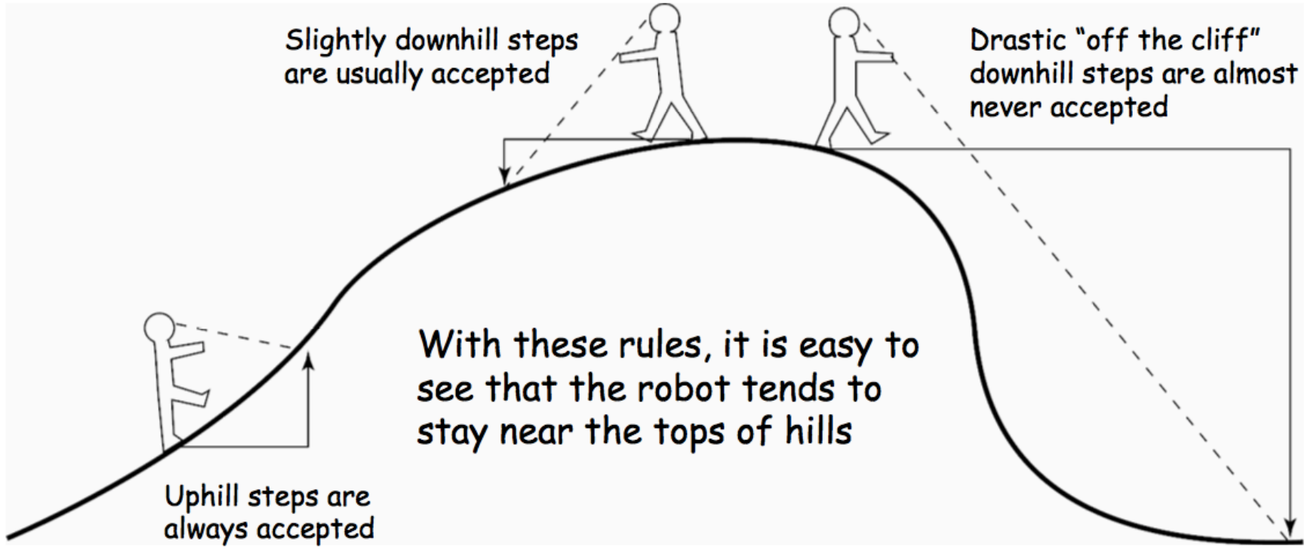


Another Example

```
def metropolis(p, qdraw, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    for i in range(nsamp):
        x_star = qdraw(x_prev)
        p_star = p(x_star)
        p_prev = p(x_prev)
        pdfratio = p_star/p_prev
        if np.random.uniform() < min(1, pdfratio):
            samples[i] = x_star
            x_prev = x_star
        else:#we always get a sample
            samples[i]= x_prev
    return samples

def prop(x):
    return np.random.normal(x, 0.6)

f = lambda x: 6*x*(1-x)
x0=np.random.uniform()
samps = metropolis(f, prop, 1000000, x0)
```

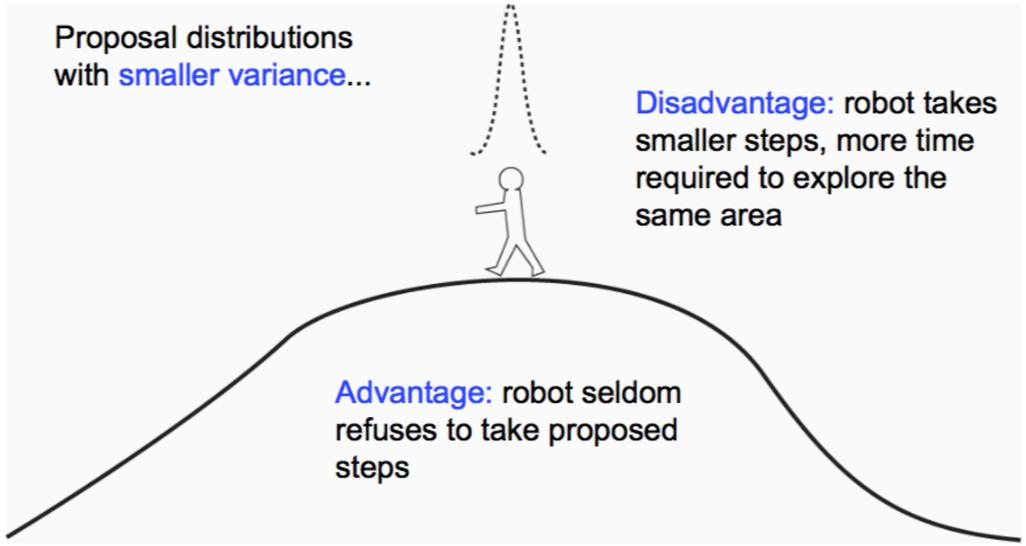


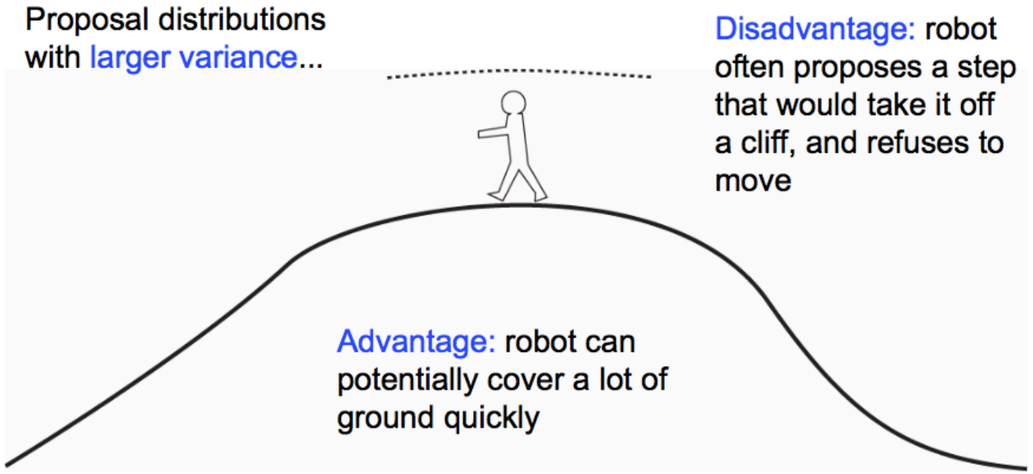
MCMC robot's rules



(from Paul Lewis)

Tuning the width or precision



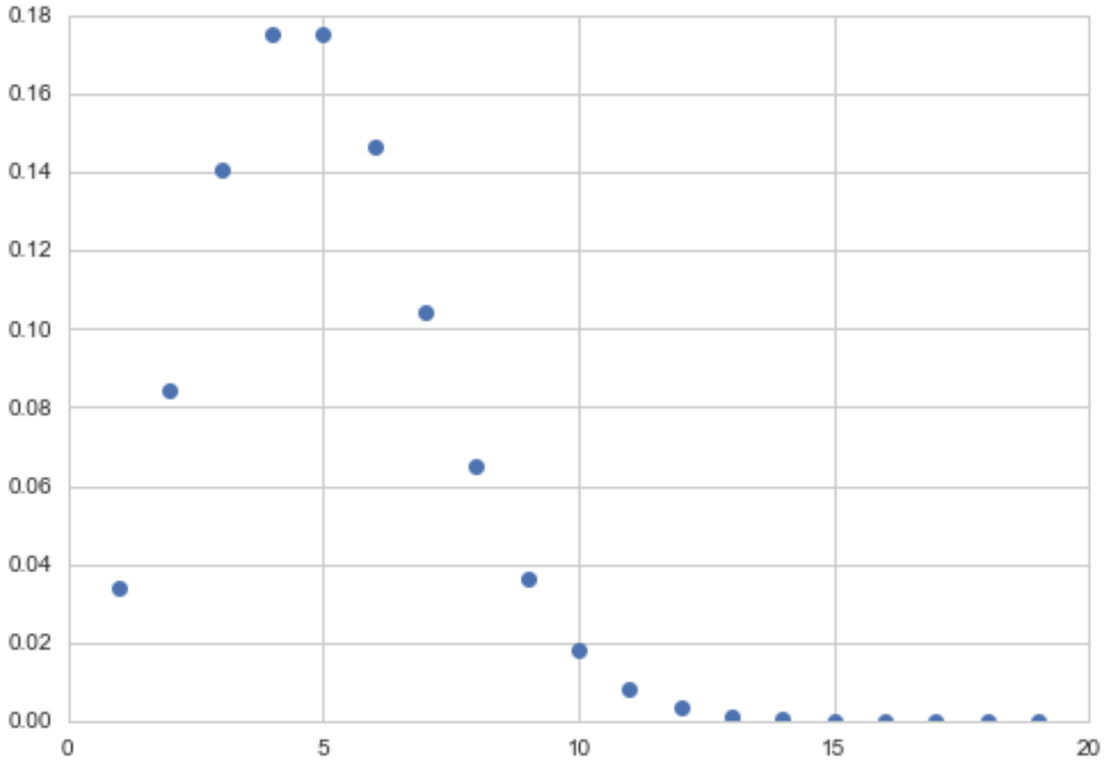


(from Paul Lewis)

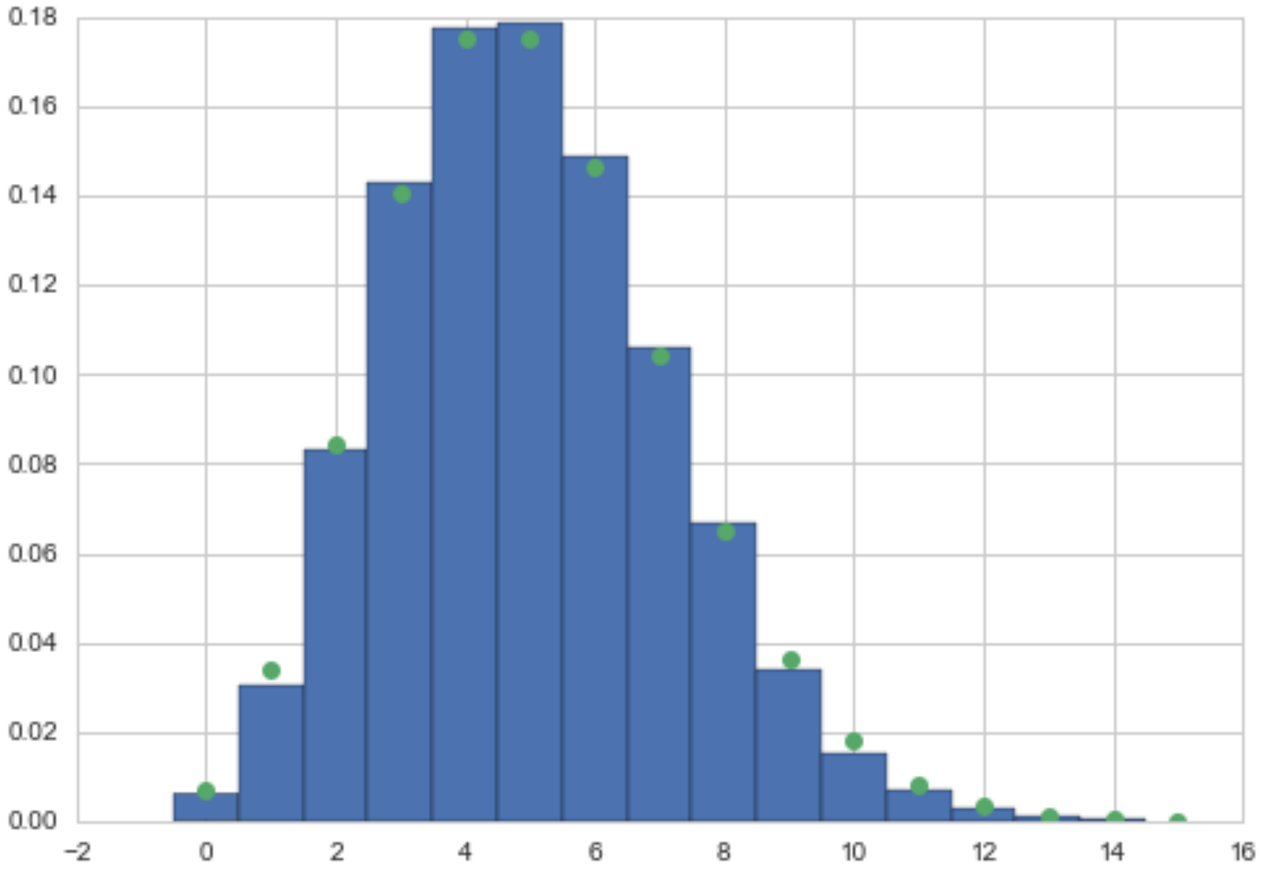
Discrete distribution MCMC

- proposal distribution becomes proposal matrix
- index the discrete outcomes
- can use symmetric or asymmetric proposal as long as rows sum to 1
- make sure matrix is irreducible: ie you can get from any index to any other one.

Example: generate poisson



$$
\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ \dots \end{array}
\begin{array}{c}
\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & \dots \end{array} \\
Q = \begin{bmatrix}
1/2 & 1/2 & 0 & 0 & & \cdots \\
1/2 & 0 & 1/2 & 0 & 0 & \cdots \\
0 & 1/2 & 0 & 1/2 & 0 & \cdots \\
0 & 0 & 1/2 & 0 & 1/2 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots &
\end{bmatrix}
\end{array}
$$



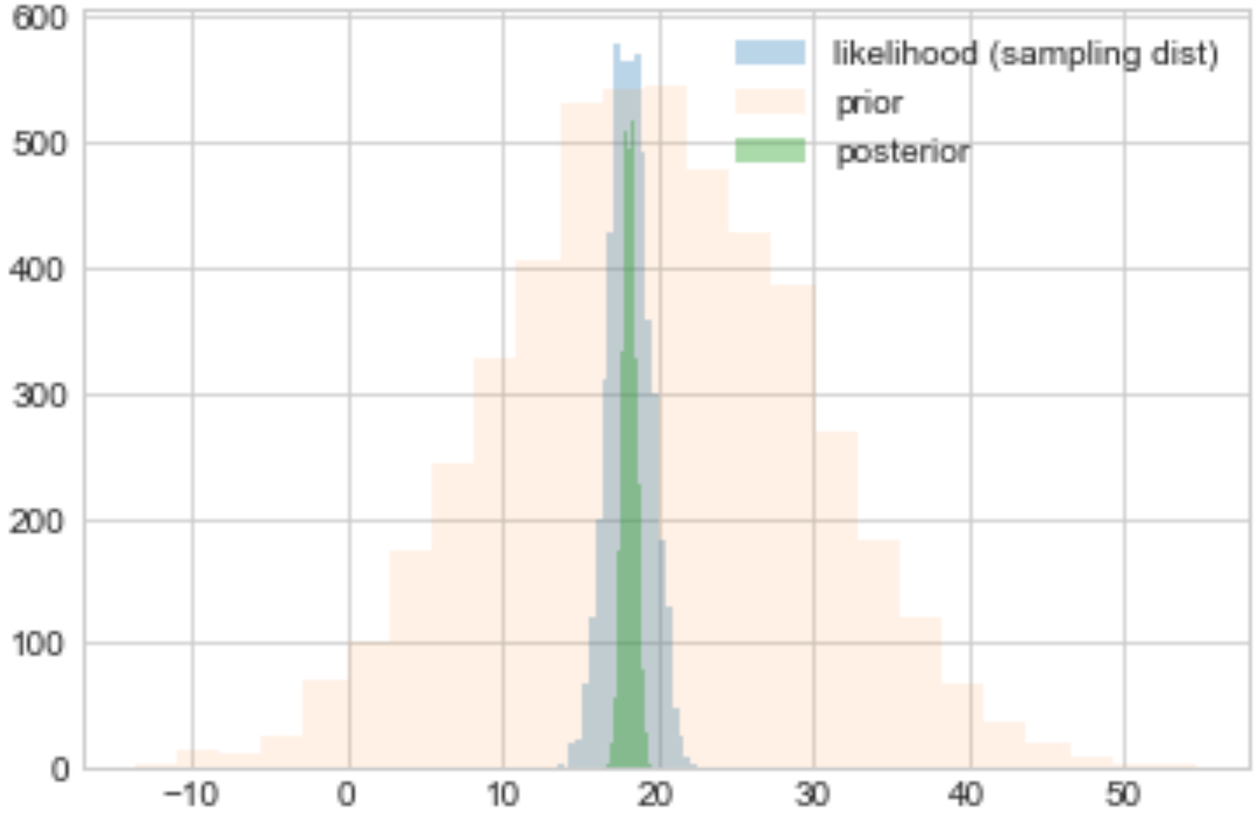
$$p(k) = e^{-\mu}\frac{\mu^k}{k!}.$$

```
def prop_draw(ifrom):
    u = np.random.uniform()
    if ifrom !=0:
        if u < 1/2:
            ito = ifrom -1
        else:
            ito = ifrom + 1
    else:
        if u < 1/2:
            ito=0
        else:
            ito=1
    return ito

def prop_pdf(ito, ifrom):
    if ito == ifrom - 1:
        return 0.5
    elif ito == ifrom + 1:
        return 0.5
    elif ito == ifrom and ito == 0:#needed to make first row sum to 1
        return 0.5
    else:
        return 0
```

Use logs to ensure numerical stability

```
def metropolis(logp, qdraw, stepsize, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    accepted = 0
    for i in range(nsamp):
        x_star = qdraw(x_prev, stepsize)
        logp_star = logp(x_star)
        logp_prev = logp(x_prev)
        logpdfratio = logp_star -logp_prev
        u = np.random.uniform()
        if np.log(u) <= logpdfratio:
            samples[i] = x_star
            x_prev = x_star
            accepted += 1
        else:#we always get a sample
            samples[i]= x_prev

    return samples, accepted
```

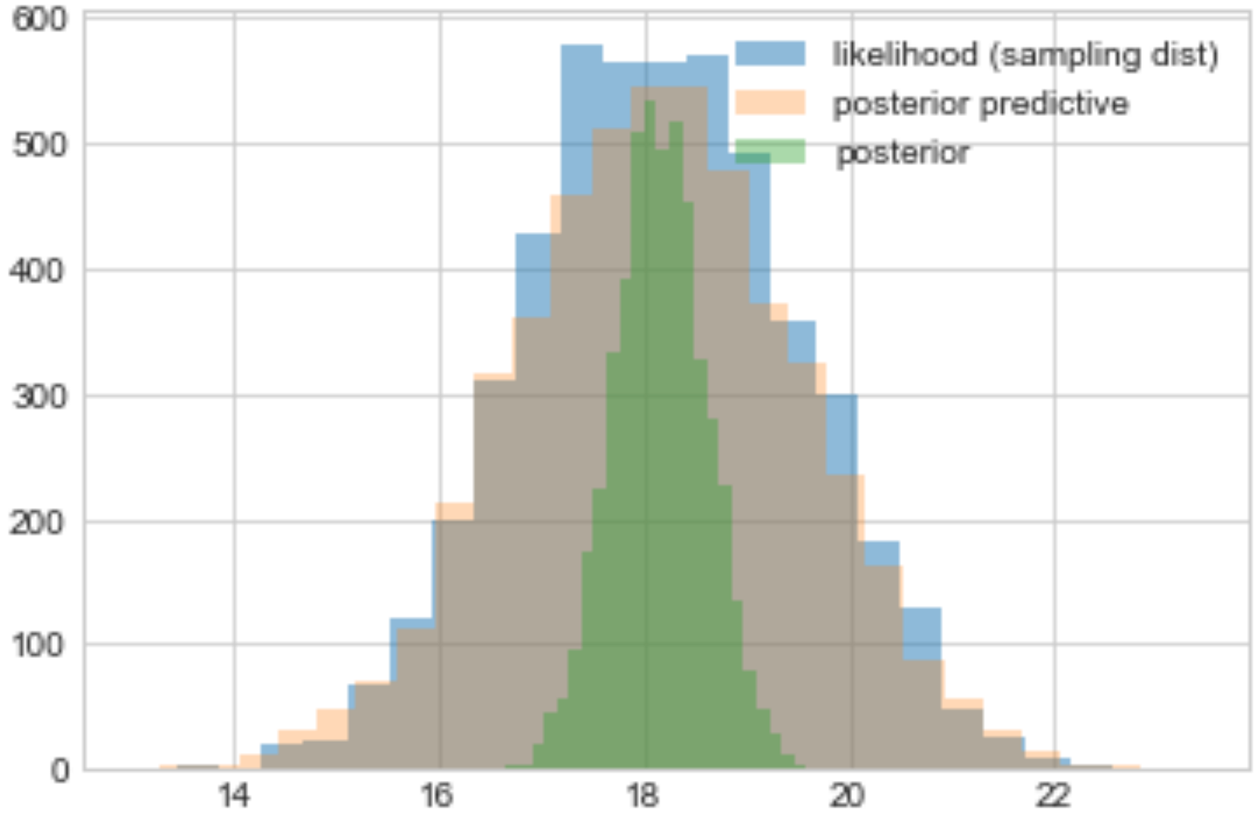
Bayesian Normal-Normal Model




```
logprior = lambda mu: norm.logpdf(mu, loc=mu_prior, scale=std_prior)
loglike = lambda mu: np.sum(norm.logpdf(Y, loc=mu, scale=np.std(Y)))
logpost = lambda mu: loglike(mu) + logprior(mu)
def prop(x, step):
    return np.random.normal(x, step)
x0=np.random.uniform()
nsamps=40000
samps, acc = metropolis(logpost, prop, 1, nsamps, x0)
```

Posterior predictive from sampling

- first draw the thetas from the posterior
- then draw y's from the likelihood
- and histogram the likelihood
- these are draws from joint y, θ



Gibbs Sampling

What did Gibbs do?

He determined the energy states of gases at equilibrium by cycling through all the particles, drawing from each one of them conditionally given the energy levels of the others, taking the time average.

Geman and Geman used this idea to denoise images.

The idea of Gibbs

$$f(x) = \int f(x,y)dy = \int f(x|y)f(y)dy = \int dy f(x|y) \int dx' f(y|x')f(x')$$

Thus: $f(x) = \int h(x,x')f(x')dx'$ integral fixed point equation

where $h(x,x') = \int dy f(x|y)f(y|x')$.

Iterative scheme in which the "transition kernel" $h(x, x')$ is used to create a proposal for metropolis-hastings moves:

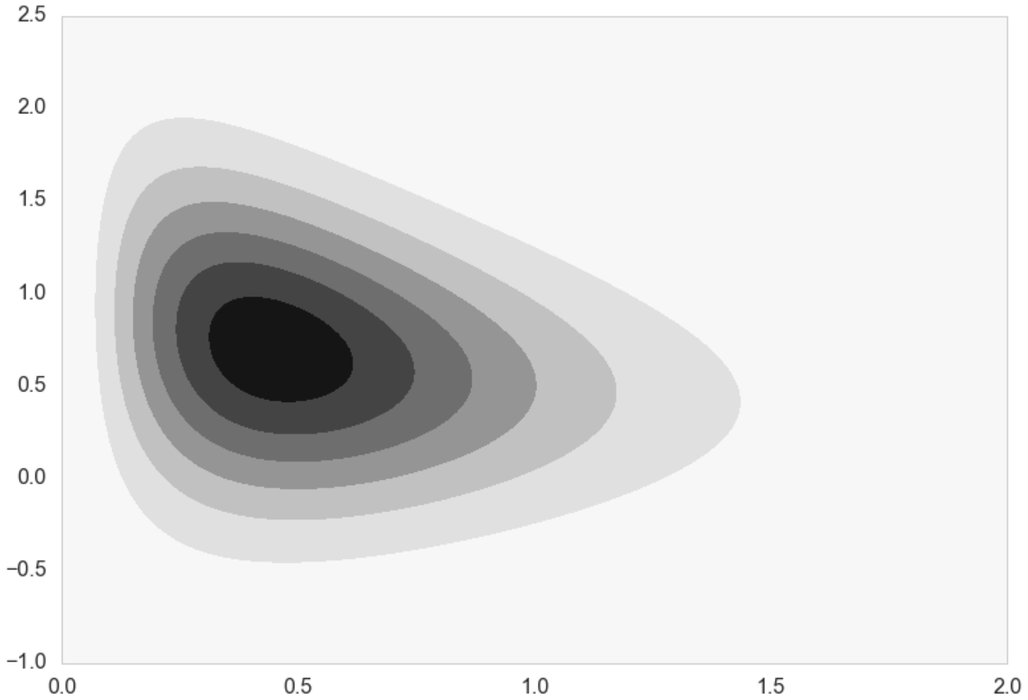
$$f(x_t) = \int h(x_t, x_{t-1}) f(x_{t-1}) dx_{t-1}$$, a Stationary distribution.

$$h(x, x') = \int dy f(x|y) f(y|x')$$.: Sample alternately to get transitions.

Can sample x marginal and $x|y$ so can sample the joint x, y.

Example

Sample from $f(x, y) = x^2 \exp[-xy^2 - y^2 + 2y - 4x]$



Conditionals

$$f(x,y) = x^2 \exp[-xy^2 - y^2 + 2y - 4x] = x^2 \exp[-x(y^2+4)] \exp[-y^2 + 2y]$$

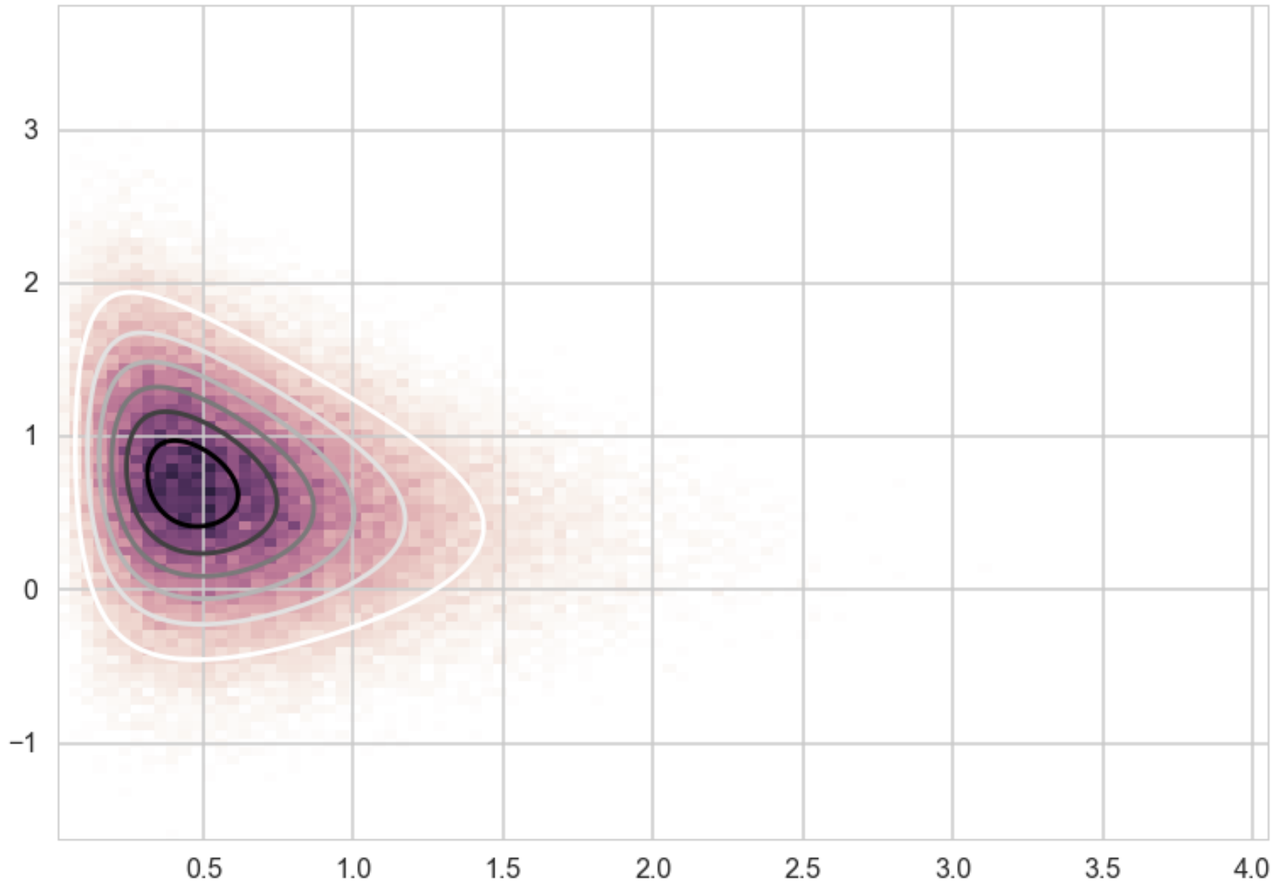
$$= g(y) \text{Gamma}(3, y^2+4) \implies f(x|y) = \text{Gamma}(3, y^2+4)$$

$$f(x,y) = x^2 \exp[-y^2(1+x) + 2y] \exp[-4x]$$

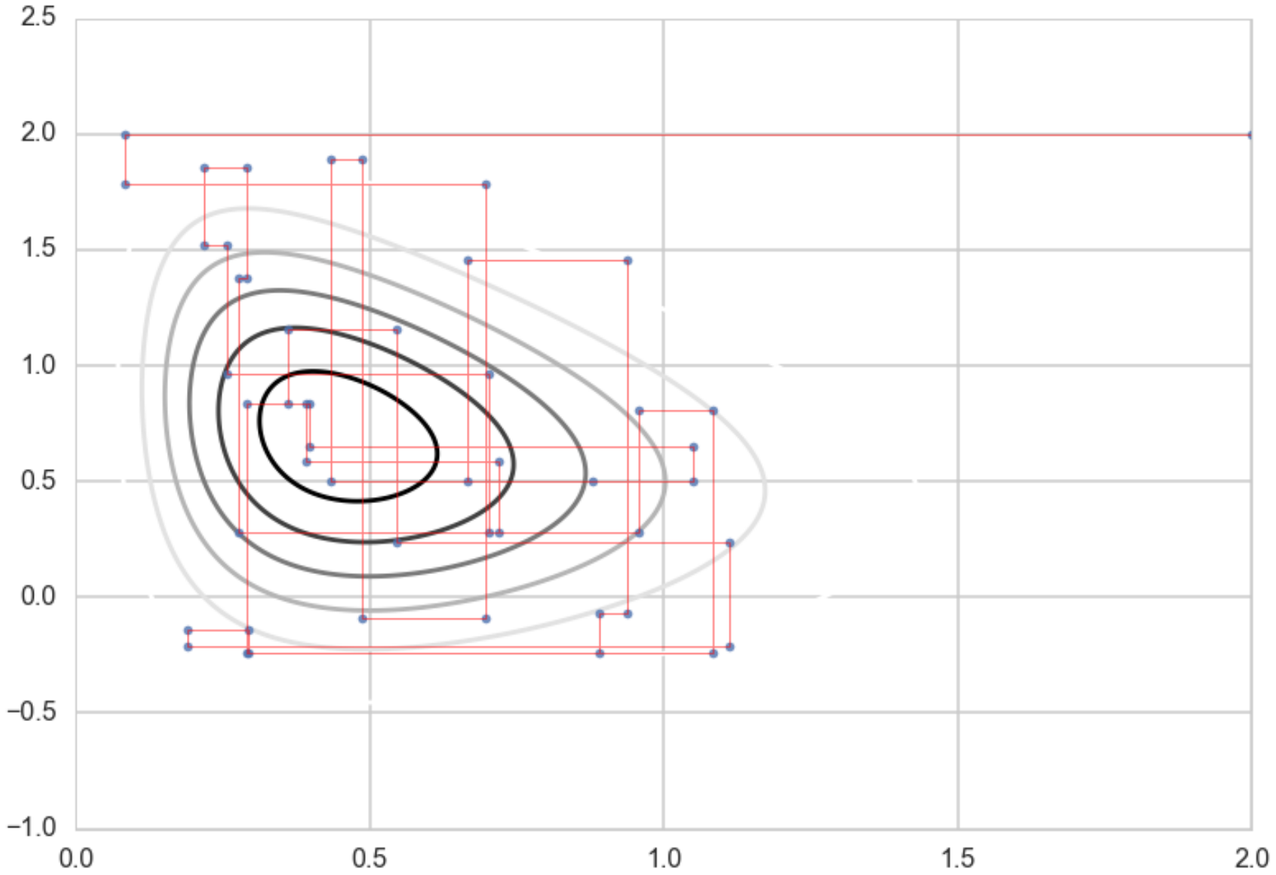
$$\implies f(y|x) = N(\frac{1}{1+x}, \frac{1}{\sqrt{(2(1+x))}})$$

Sampler

```
def xcond(y):
    return gamma.rvs(3, scale=1/(y*y + 4))
def ycond(x):
    return norm.rvs(1/(1+x), scale=1.0/np.sqrt(2*(x+1)))
def gibbs(xgiveny_sample, ygivenx_sample, N, start = [0,0]):
    x=start[0]
    y=start[1]
    samples=np.zeros((N+1, 2))
    samples[0,0]=x
    samples[0,1]=y
    for i in range(1,N,2):
        x=xgiveny_sample(y)
        samples[i,0]=x
        samples[i, 1]=y
        ######################
        y=ygivenx_sample(x)
        samples[i+1,0]=x
        samples[i+1,1]=y
    return samples
out=gibbs(xcond, ycond, 100000)
```



More about gibbs



- easiest is to know how to sample directly from conditionals: **no need for locality**
- moves one component (or one block) at a time
- all is not lost if thats not the case: can use a MH-step once stationarity has been reached
- this makes gibbs a very general idea

Fully Bayesian Rat tumors

Joint Posterior:

$$p(\Theta, \alpha, \beta | Y, \{n_i\}) \propto p(\alpha, \beta) \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta) \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Conditionals:

$$p(\theta_i | y_i, n_i, \alpha, \beta) = Beta(\alpha + y_i, \beta + n_i - y_i)$$

More Conditionals

$$p(\alpha|Y,\Theta,\beta) \propto p(\alpha,\beta)\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}\right)^N \prod_{i=1}^{N}\theta_i^{\alpha}$$

$$P(\beta|Y,\Theta,\alpha) \propto p(\alpha,\beta)\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\beta)}\right)^N \prod_{i=1}^{N}(1-\theta_i)^{\beta}$$

These depend on Y and $\{n\}$ via the θ's

Sampling (sampler done in lab)

- Fix α and β, we have a Gibbs step for all of the θ_is
- For α and β, everything else fixed, use stationary metropolis step, as conditionals are not isolatable to simply sampled distributions
- when we sample for α, we will propose a new value using a normal proposal, while holding all the θs and β constant at the old value. ditto for β.