Lecture 10
Annealing to Metropolis with a bit on Markov

Last time: Simulated Annealing

Minimize f by identifying with the energy of an imaginary physical system undergoing an annealing process.

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

If the new state has lower energy, accept x_j.

If the new state has higher energy, accept with probability

$$A = \exp\left(-\Delta f / kT\right)$$

Today

- from annealing to Metropolis
- markov chains and MCMC
- Metropolis

Next Time

- markov chain theory
- metropolis and metropolis hastings
- discrete proposals

Physical Annealing

A system is first heated to a melting state and then cooled down slowly.

- when solid is heated, its molecules start moving randomly, and its energy increases
- if subsequent process of cooling is slow, the energy decreases slowly, with some random increases governed by the Boltzmann distribution
- if cooling slow and deep enough, system will eventually settle

Simulated Annealing

Minimize f by identifying with the energy of an imaginary physical system undergoing an annealing process.

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

If the new state has lower energy, accept x_j.

If the new state has higher energy, accept with probability

$$A = \exp\left(-\Delta f / kT\right)$$

- stochastic acceptance of higher energy states, allows our process to escape local minima.
- When T is high, the acceptance of these uphill moves is higher, and local minima are discouraged.
- As T is lowered, more concentrated search near current local minimum, since only few uphill moves will be allowed.
- Thus, if we get our temperature decrease schedule right, we can hope that we will converge to a global minimum.

If the lowering of the temperature is sufficiently slow, the system reaches "thermal equilibrium" at each temperature. Then Boltzmann's distribution applies:

$$p(X = i) = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{kT}\right)$$

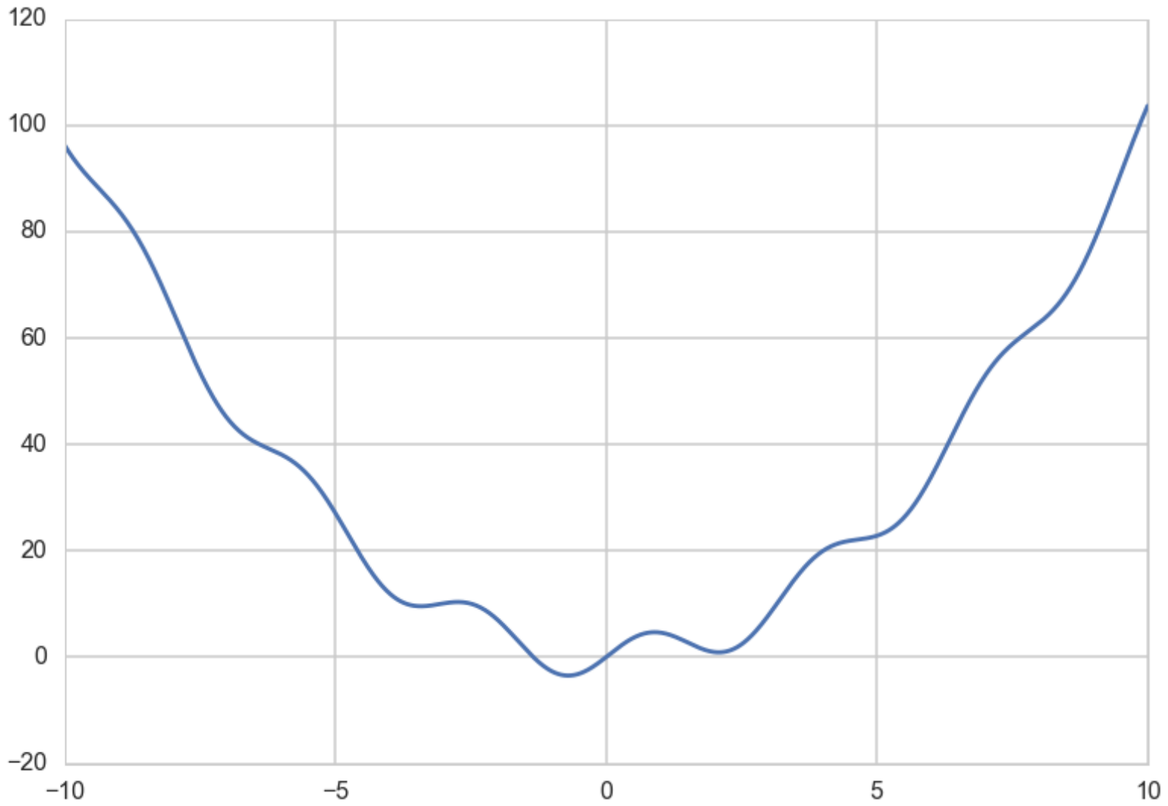
where

$$Z(T) = \sum_j \exp\left(\frac{-E_j}{kT}\right)$$

Proposal

- it proposes a new position x from a **neighborhood** $\mathcal{N}$ at which to evaluate the function.
- all the positions x in the domain we wish to minimize a function f over ought to be able to communicate.
- detailed balance: proposal is symmetric
- ensures $\{x_t\}$ generated by simulated annealing is a stationary markov chain with target boltzmann distribution: equilibrium

Example: $x^2 + 4sin(2x)$



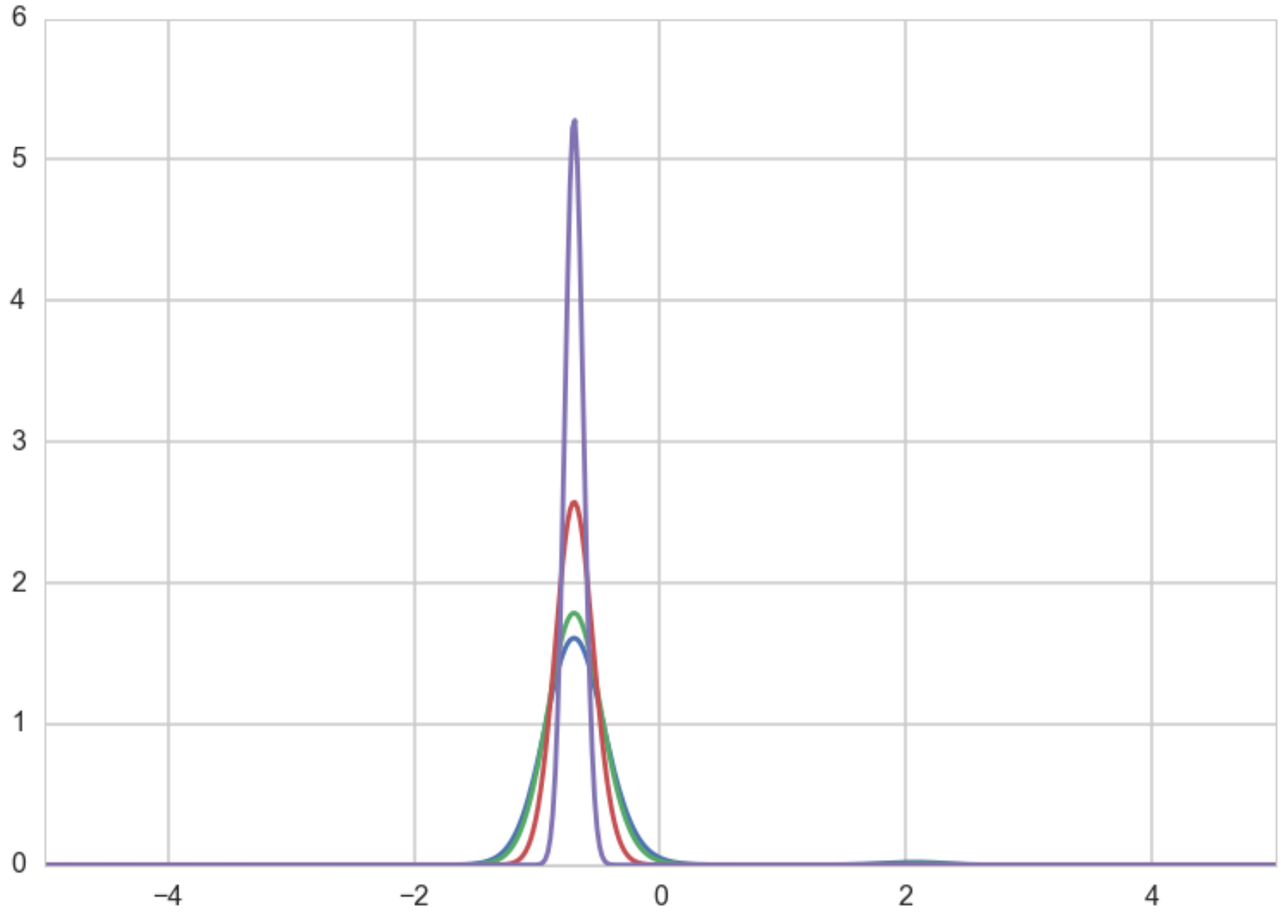
If you identify

$$p_T(x) = e^{-f(x)/T} \text{ and } p(x) = e^{-f(x)}$$

Then:

$$P_T(x) = P(x)^{1/T}$$

- you get a peakier distribution as $T to 0$ around the global minimum: distribution $\rightarrow$ optimum!
- the globality and the exponentiation ensures that this peak is favored over the rest in f



Normalized Boltzmann distribution

- M global minima in set $\mathcal{M}$
- function minimum value f_{min}:

$$p(x_i) = \frac{e^{-(f(x_i)-f_{min})/T}}{M + \sum_{j \notin \mathcal{M}} e^{-(f(x_i)-f_{min})/T}}$$

As $T \to 0$ from above, this becomes $1/M$ if $x_i \in \mathcal{M}$ and 0 otherwise.

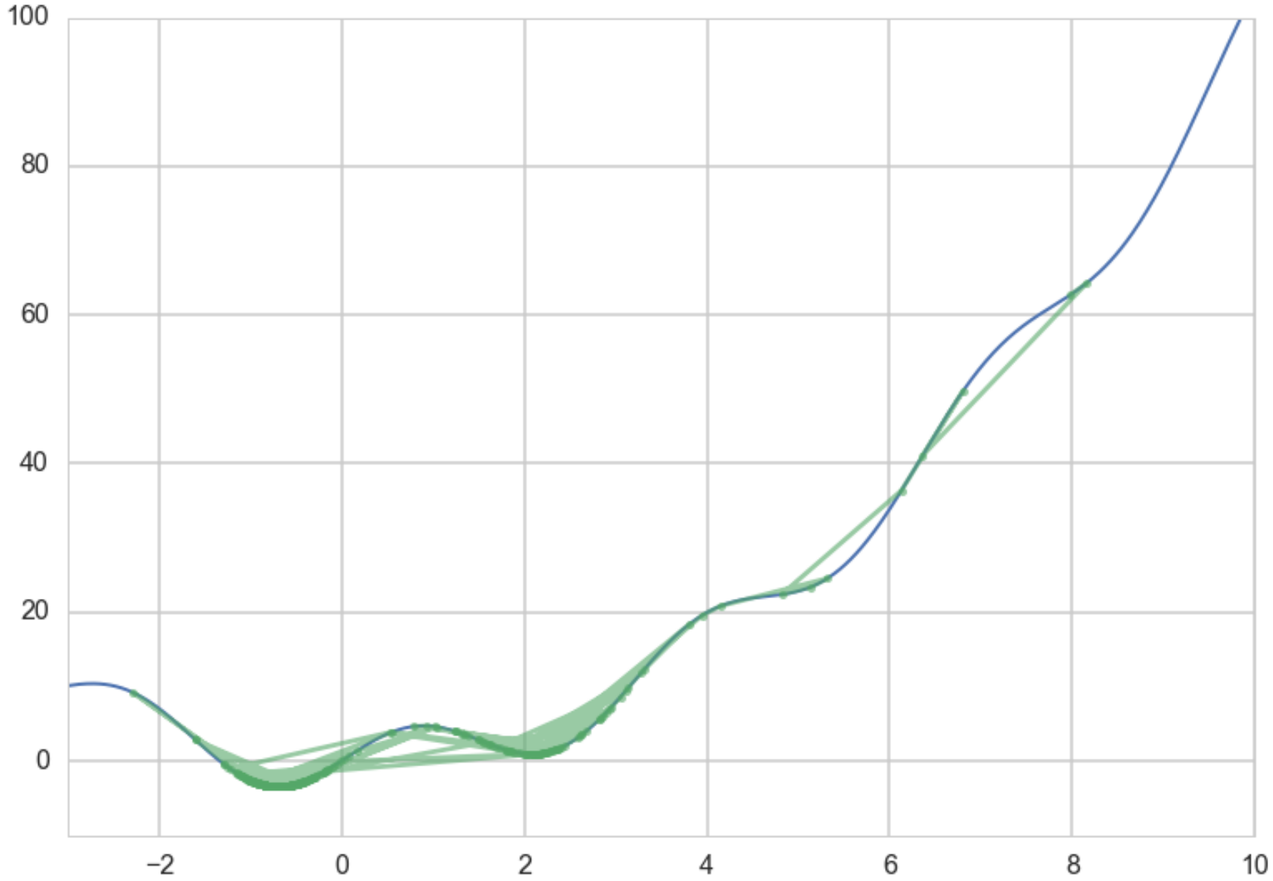
The Simulated Annealing Algorithm

1. Initialize $x_i, T, L(T)$ where L = iterations at a particular temperature.

2. Perform L transitions:
   (a) propose x_j (b) If x_j is accepted (according to probability $P = e^{(-\Delta E/T)}$), set $x_{i+1} = x_j$, else set $x_{i+1} = x_i$

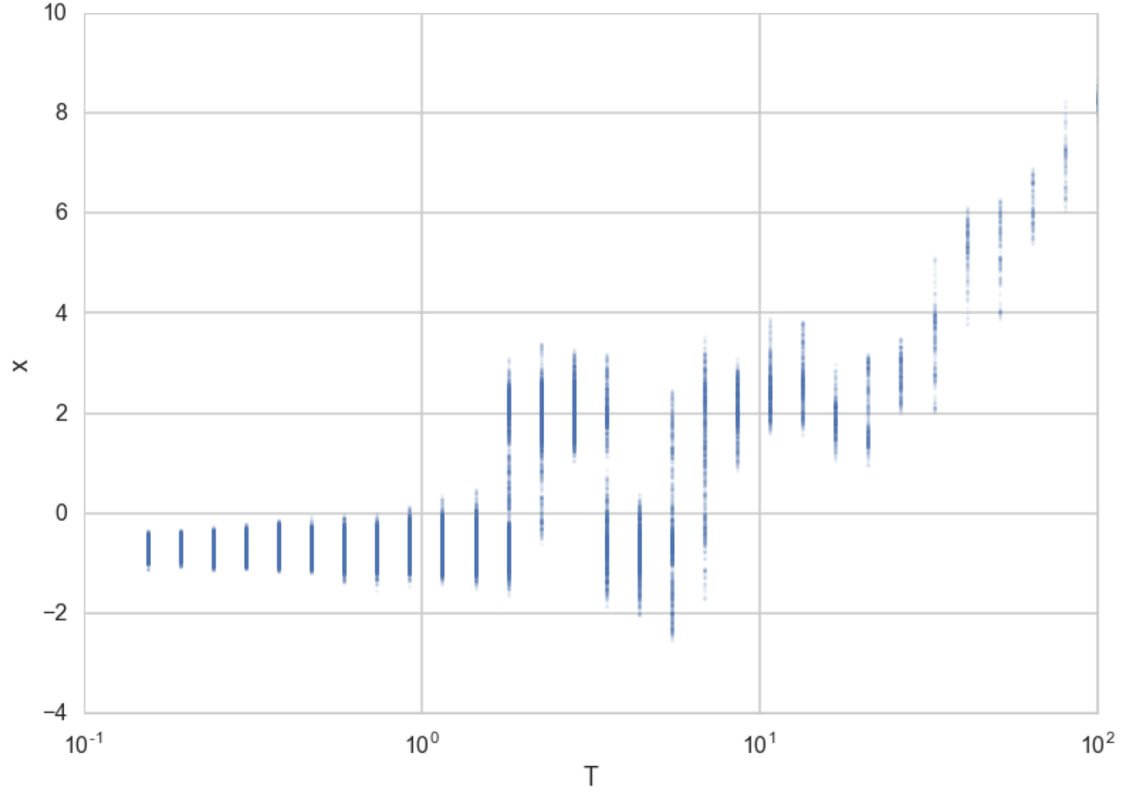
3. Update T and L, go to 2

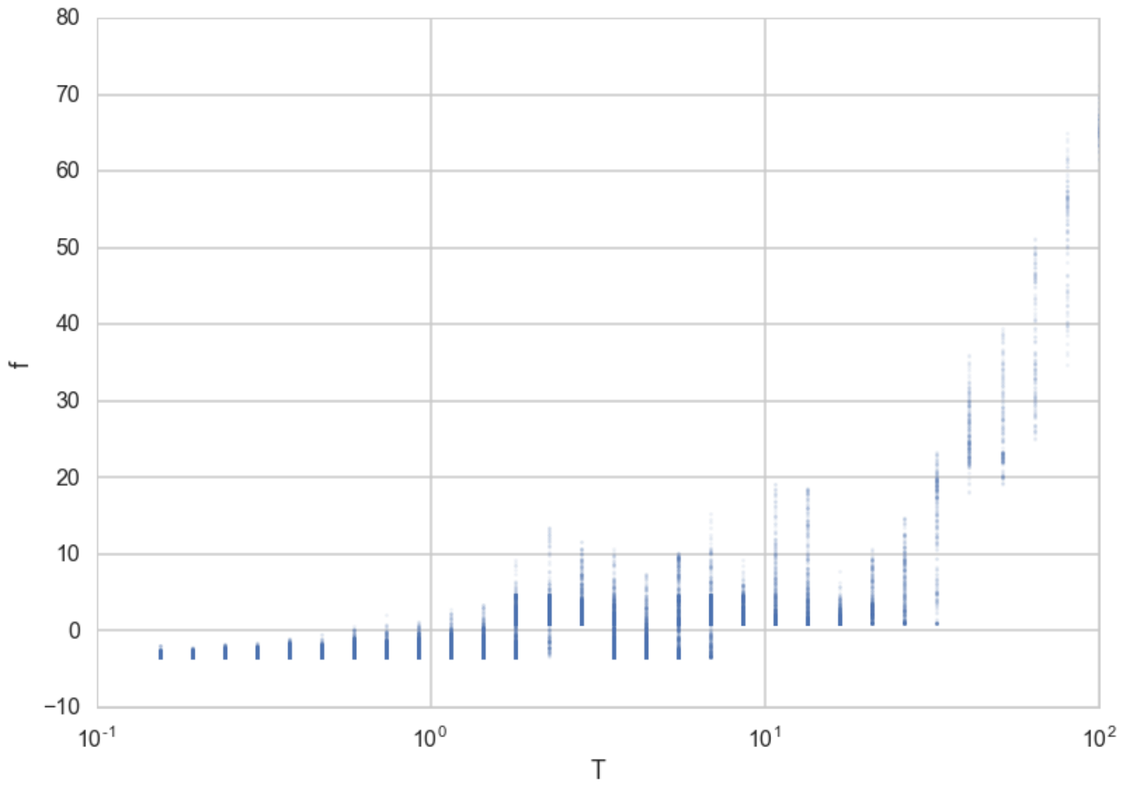
```python
def sa(energyfunc, initials, epochs, tempfunc, iterfunc, proposalfunc):
    accumulator=[]
    best_solution = old_solution = initials['solution']
    T=initials['T']
    length=initials['length']
    best_energy = old_energy = energyfunc(old_solution)
    accepted=0
    total=0
    for index in range(epochs):
        print("Epoch", index)
        if index > 0:
            T = tempfunc(T)
            length=iterfunc(length)
        print("Temperature", T, "Length", length)
        for it in range(length):
            total+=1
            new_solution = proposalfunc(old_solution)
            new_energy = energyfunc(new_solution)
            # Use a min here as you could get a "probability" > 1
            alpha = min(1, np.exp((old_energy - new_energy)/T))
            if ((new_energy < old_energy) or (np.random.uniform() < alpha)):
                # Accept proposed solution
                accepted+=1
                accumulator.append((T, new_solution, new_energy))
                if new_energy < best_energy:
                    # Replace previous best with this one
                    best_energy = new_energy
                    best_solution = new_solution
                    best_index=total
                    best_temp=T
                old_energy = new_energy
                old_solution = new_solution
            else:
                # Keep the old stuff
                accumulator.append((T, old_solution, old_energy))

    best_meta=dict(index=best_index, temp=best_temp)
    print("frac accepted", accepted/total, "total iterations", total, 'bmeta', best_meta)
    return best_meta, best_solution, best_energy, accumulator
```

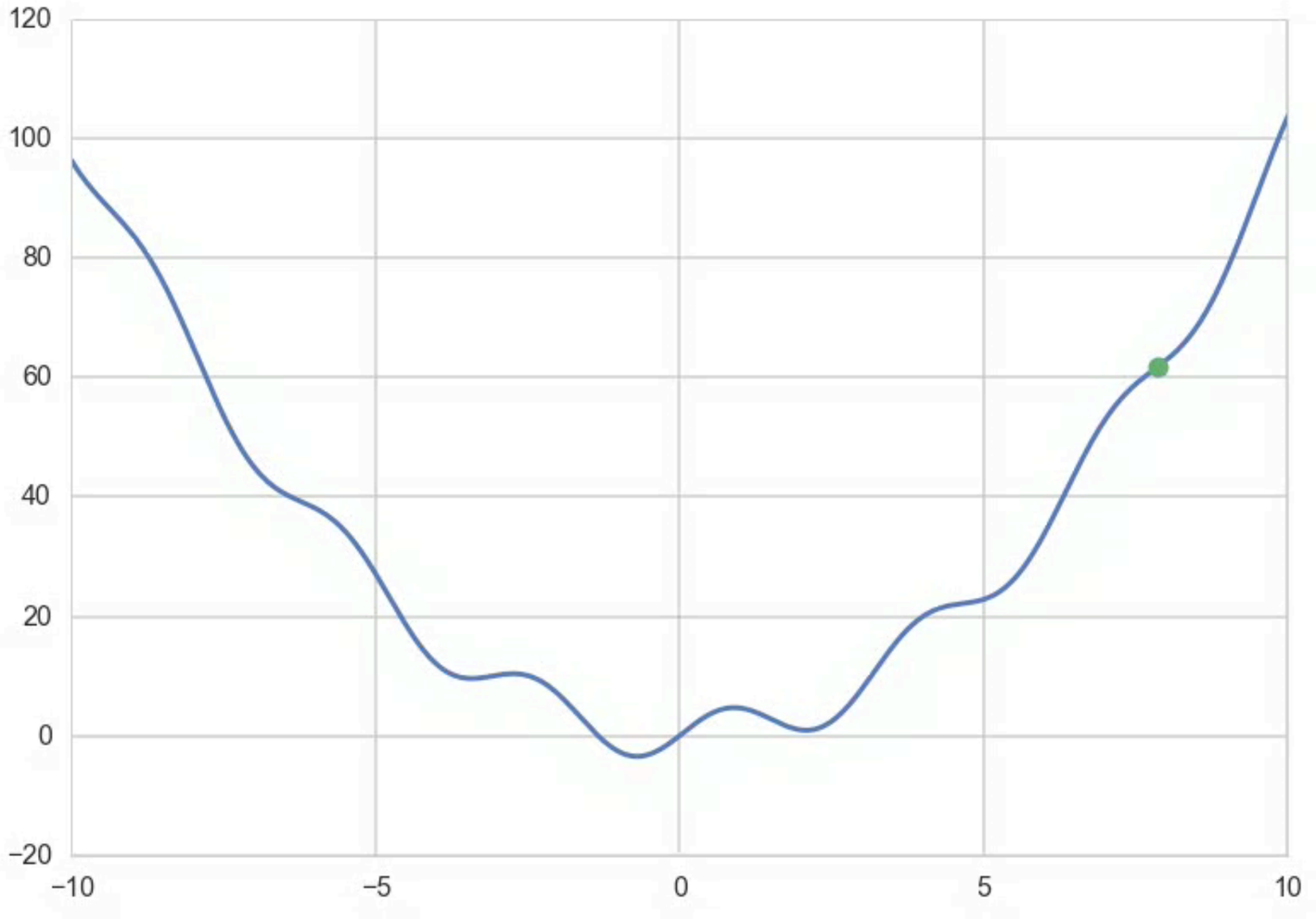


```
tf = lambda t: 0.8*t #temperature function
itf = lambda length: math.ceil(1.2*length) #iteration function
inits=dict(solution=8, length=100, T=100)
bmeta, bs, be, out = sa(f, inits, 30, tf, itf, pf)
```

```
Epoch 0
Temperature 100 Length 100
Epoch 1
Temperature 80.0 Length 120
Epoch 2
Temperature 64.0 Length 144
Epoch 3
Temperature 51.2 Length 173
Epoch 4
Temperature 40.96000000000001 Length 208
Epoch 5
Temperature 32.76800000000001 Length 250
Epoch 6
Temperature 26.21440000000001 Length 300
Epoch 7
Temperature 20.97152000000001 Length 360
...
Epoch 27
Temperature 0.24178516392292618 Length 13863
Epoch 28
Temperature 0.19342813113834095 Length 16636
Epoch 29
Temperature 0.15474250491067276 Length 19964
frac accepted 0.7921531132581857 total iterations 119232 bmeta {'index': 112695, 'temp': 0.15474250491067276}
```


10
8
6
4
2
0
−2
−4
x
10^{-1}
10^{0}
10^{1}
10^{2}
T


80
70
60
50
40
30
20
10
0
−10
f
10^{-1}
10^{0}
10^{1}
10^{2}
T


120
100
80
60
40
20
0
−20
−10
−5
0
5
10

Practical choices

- Start T_0 large to accept all transitions.
- Thermostat

  1. Linear: Temperature decreases as $T_{k+1} = \alpha T_k$.
  2. Exponential: Temperature decreases as 0.95^{k}
  3. Logarithmic: Temperature decreases as $1/\log(\mathrm{k})$

- Reannealing interval, or epoch length is the number of points to accept before reannealing (change the temperature). Typical starting value is 100, increase it as $L_{k+1} = \beta L_k$ where $\beta > 1$.
- Larger decreases in temperature require correspondingly longer epoch lengths to re-equilibriate
- Running long epochs at larger temperatures is not very useful. Decrease temperature rapidly at first.

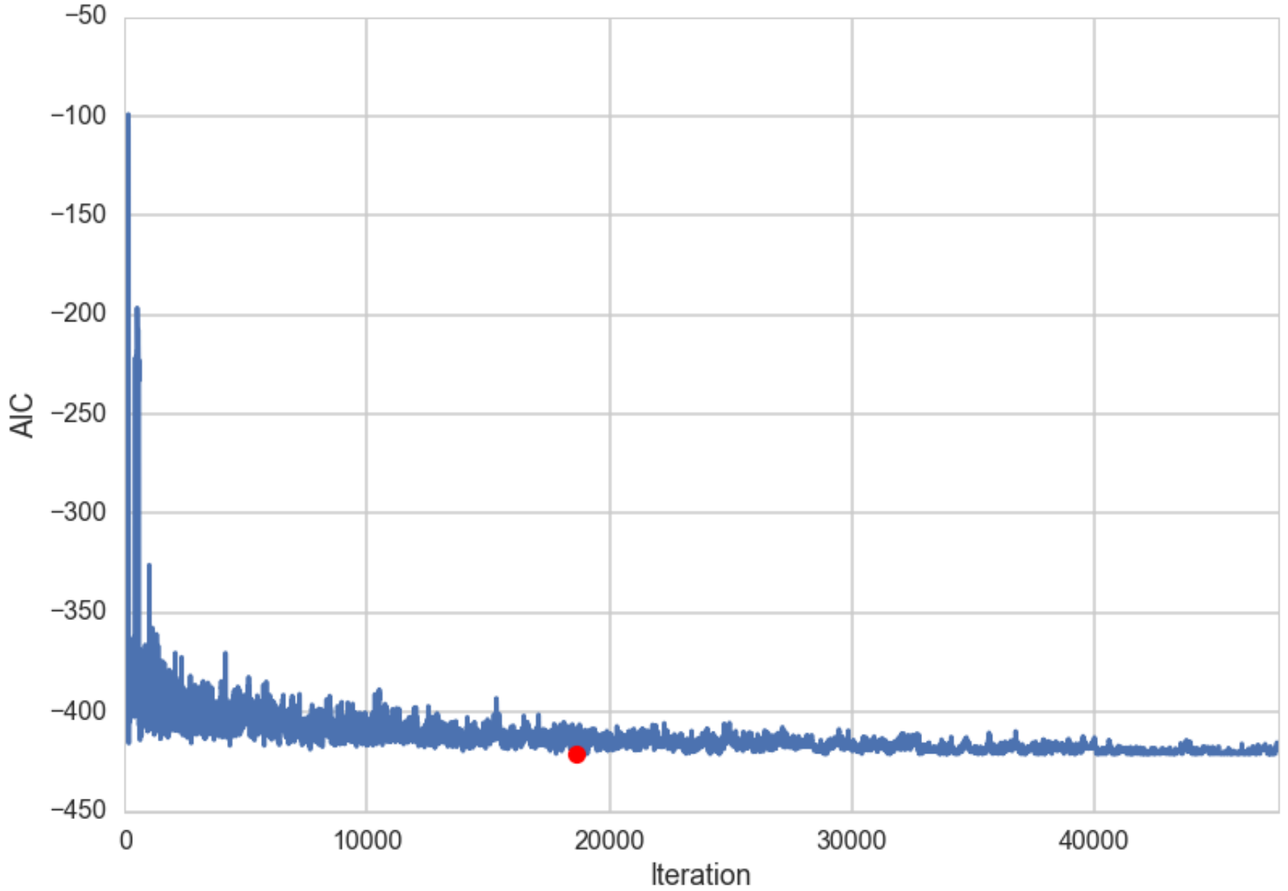
Simulated Annealing for baseball

```
bbinits=dict(solution=np.random.binomial(1, 0.5, ncols).astype(bool),
                  length=100, T=100)
def efunc(solution):
    solution_vars = predictors[predictors.columns[solution]]
    g = LinearRegression().fit(X=solution_vars, y=logsalary)
    return aic(g, solution_vars, logsalary)
def pfunc(solution):
    flip = np.random.randint(0, ncols)
    solution_new = solution.copy()
    solution_new[flip] = not solution_new[flip]
    return solution_new
```

```
tf2 = lambda temp: 0.8*temp
itf2 = lambda length: math.ceil(1.2*length)
bb_bmeta, bb_bs, bb_be, bb_out = sa(efunc, bbinits, 25, tf2, itf2, pfunc)
```

```
Epoch 0
Temperature 100 Length 100
Epoch 1
Temperature 80.0 Length 120
Epoch 2
Temperature 64.0 Length 144
Epoch 3
Temperature 51.2 Length 173
Epoch 4
Temperature 40.96000000000001 Length 208
Epoch 5
Temperature 32.76800000000001 Length 250
...
Epoch 21
Temperature 0.9223372036854786 Length 4641
Epoch 22
Temperature 0.7378697629483829 Length 5570
Epoch 23
Temperature 0.5902958103587064 Length 6684
Epoch 24
Temperature 0.4722366482869651 Length 8021
frac accepted 0.37204513458427013 total iterations 47591 bmeta {'index': 18619, 'temp': 1.4411518807585602}
```

```
Best AIC: -420.9472114371548
Best solution: (array([ 1,  2,  5,  7,  9, 12, 13, 14, 15, 23, 24, 25]),)
Discovered at iteration 18618
```



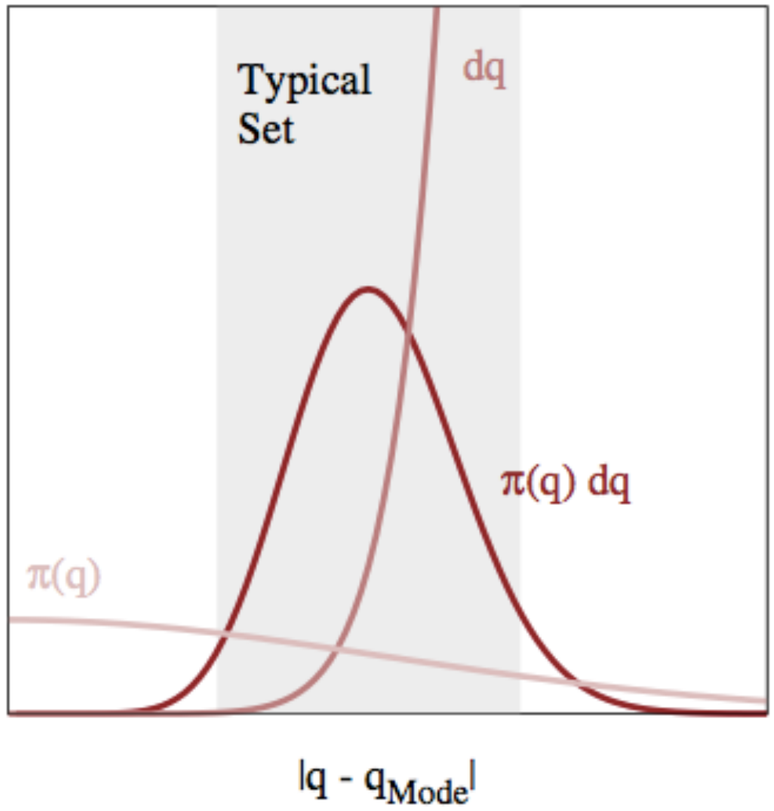
MCMC

Markov Chain Monte Carlo

Sampling a Distribution

- Turn the question on its head.
- Suppose we wanted to sample from a distribution $p(x)$ (corresponding to a minimization of energy $-log(p(x))$).
- keep our symmetric proposal (reversibility!). Need irreducibility to sample from full distribution
- set T=1, and use our simulated annealing method

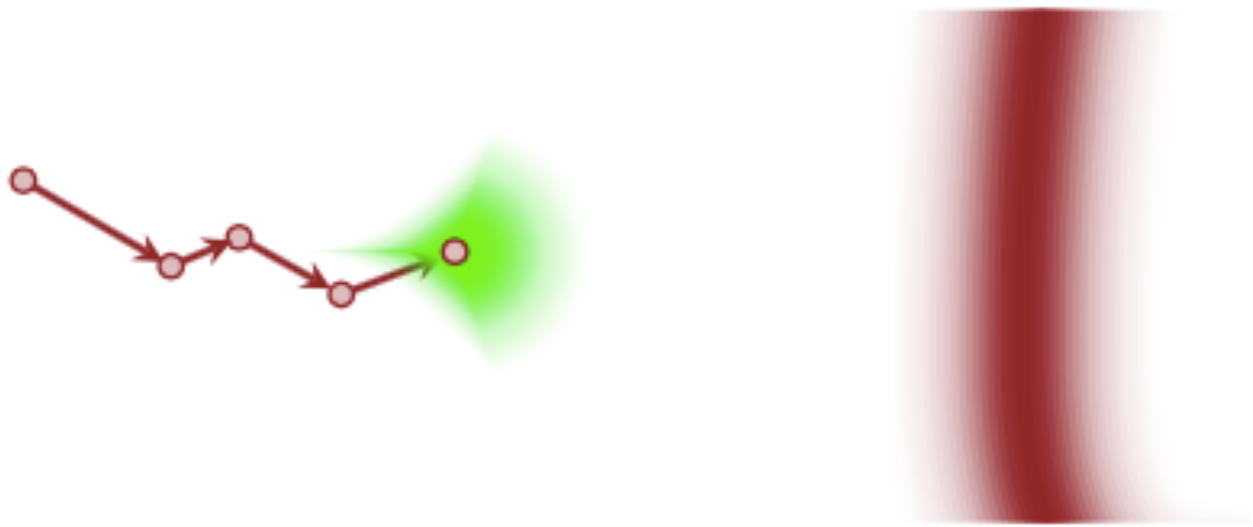
The Typical Set



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.

Intuition: approaches typical set



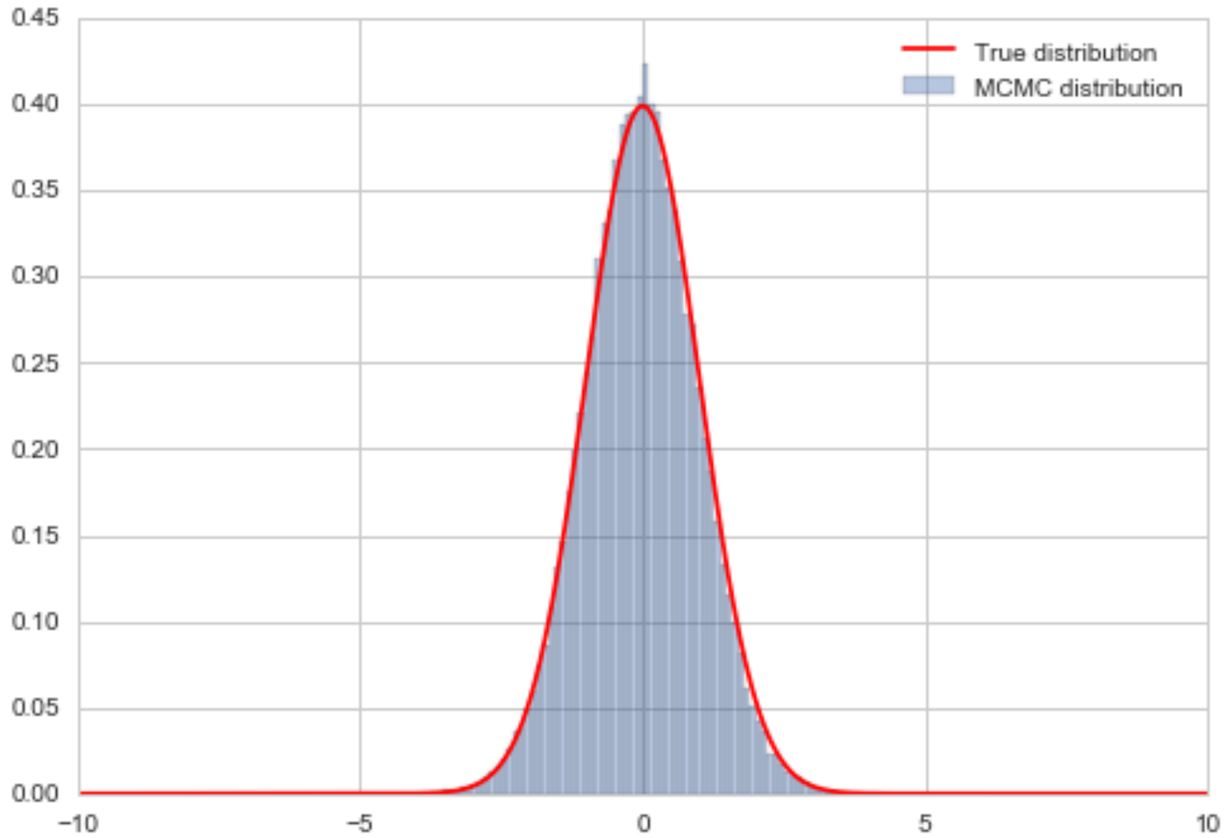
Instead of sampling p we sample q, yielding a new state, and a new proposal distribution from which to sample.

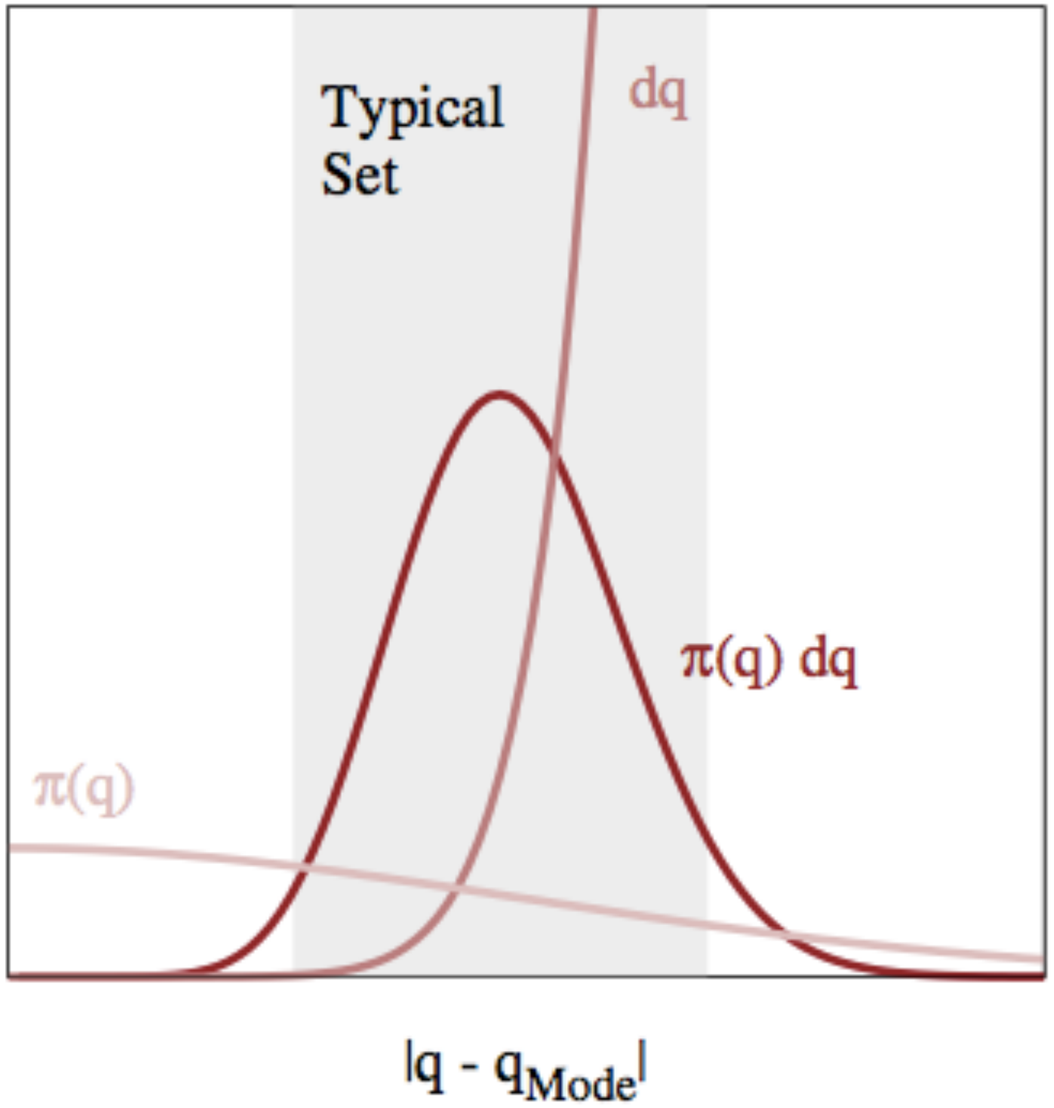
```
def metropolis(p, qdraw, nsamp, xinit):
    samples=np.empty(nsamp)
    x_prev = xinit
    for i in range(nsamp):
        x_star = qdraw(x_prev)
        p_star = p(x_star)
        p_prev = p(x_prev)
        pdfratio = p_star/p_prev
        if np.random.uniform() < min(1, pdfratio):
            samples[i]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

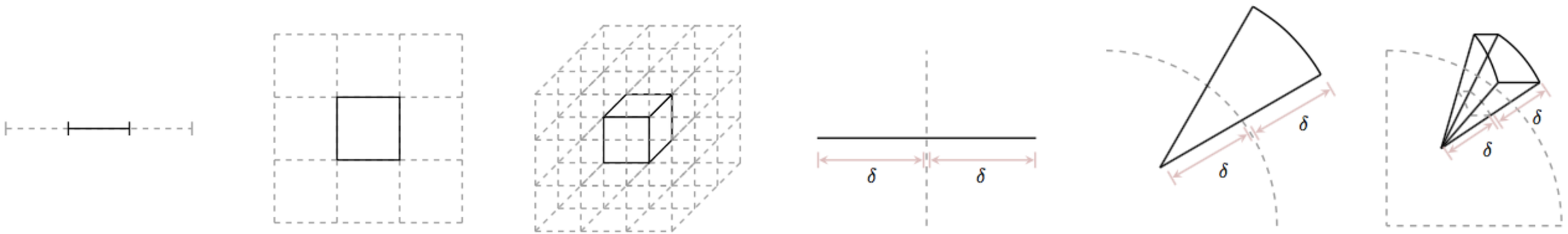




Why do this?

- 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

Stationarity

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

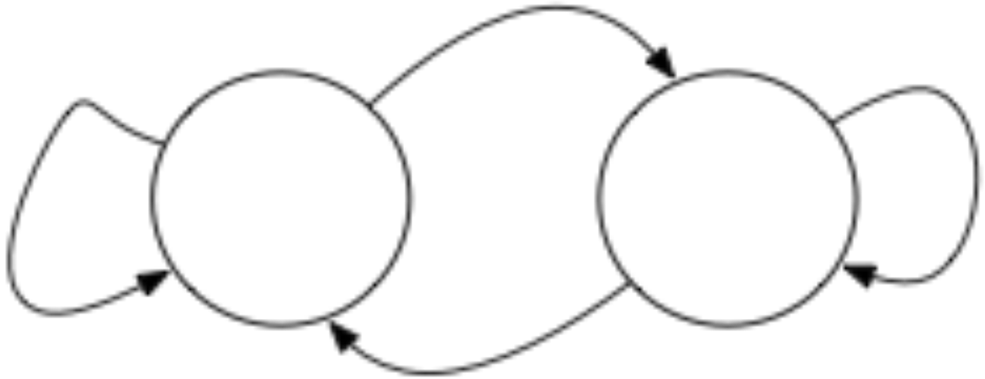
Continuous case: define T so that:

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

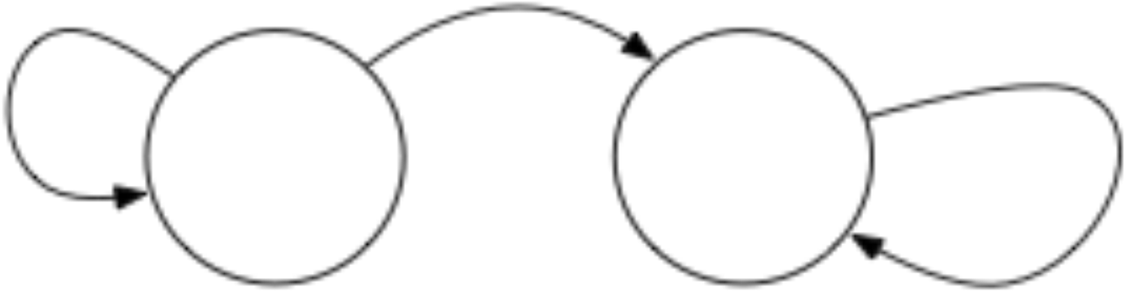
$$\int dx s(x) T(y|x) = \int p(y,x) dx = s(y)$$

Jargon

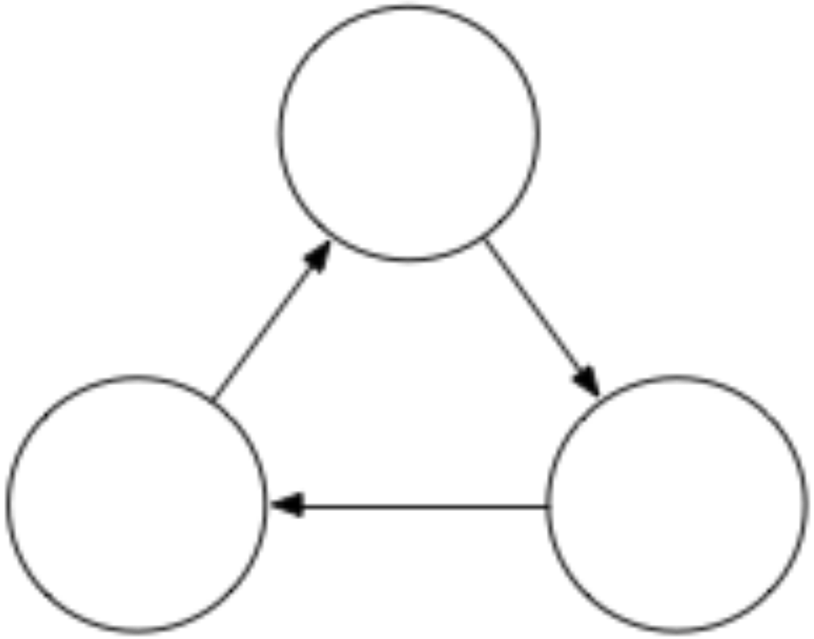
- **Irreducible**: can go from anywhere to everywhere
- **Aperiodic**: no finite loops
- **Recurrent**: visited repeatedly. Harris recurrent if all states are visited infinitely as $t \to \infty$.



Irreducible

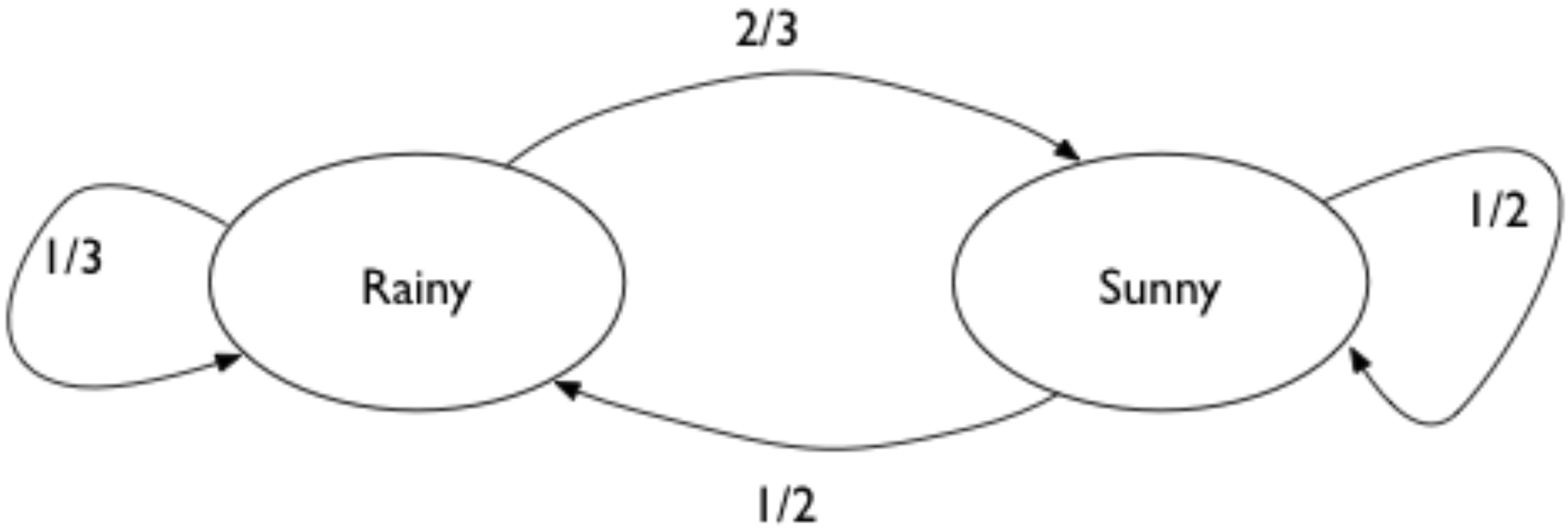


Reducible



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 simulated annealing is a stationary markov chain with target boltzmann distribution: equilibrium
- ensures $\{x_t\}$ generated by metropolis is a stationary markov chain with appropriate target.

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.

But first, a generalization to asymmetric proposals: Metropolis Hastings...