The Clever Machine

Topics in Computational Neuroscience & Machine Learning

Blog Archives

MCMC: The Metropolis Sampler

OCT 5
Posted by **dusti nstansbury**

As discussed in an earlier post (https://theclevermachine.wordpress.com/2012/09/24/a-brief-introduction-to-markov-chains/), we can use a Markov chain to sample from some ***target probability distribution*** $p(x)$ from which drawing samples directly is difficult. To do so, it is necessary to design a transition operator for the Markov chain which makes the chain's stationary distribution match the target distribution. The Metropolis sampling algorithm (and the more general Metropolis-Hastings sampling algorithm) uses simple heuristics to implement such a transition operator.

Metropolis Sampling

Starting from some random initial state $x^{(0)} \sim \pi^{(0)}$, the algorithm first draws a possible sample x^* from a ***proposal distribution*** $q(x|x^{(t-1)})$. Much like a conventional transition operator for a Markov chain, the proposal distribution depends only on the previous state in the chain. However, the transition operator for the Metropolis algorithm has an additional step that assesses whether or not the target distribution has a sufficiently large density near the proposed state to warrant accepting the proposed state as a sample and setting it to the next state in the chain. If the density of $p(x)$ is low near the proposed state, then it is likely (but not guaranteed) that it will be rejected. The criterion for accepting or rejecting a proposed state are defined by the following heuristics:

1. If $p(x^*) \geq p(x^{(t-1)})$, the proposed state is kept x^* as a sample and is set as the next state in the chain (i.e. move the chain's state to a location where $p(x)$ has equal or greater density).
2. If $p(x^*) < p(x^{(t-1)})$–indicating that $p(x)$ has low density near x^*–then the proposed state may still

be accepted, but only randomly, and with a probability $\frac{p(x^*)}{p(x^{(t-1)})}$

These heuristics can be instantiated by calculating the ***acceptance probability*** for the proposed state.

$\alpha = \min\left(1, \frac{p(x^*)}{p(x^{(t-1)})}\right)$

Having the acceptance probability in hand, the transition operator for the metropolis algorithm works like this: if a random uniform number u is less than or equal to α, then the state x^* is accepted (as in (1) above), if not, it is rejected and another state is proposed (as in (2) above). In order to collect M samples using Metropolis sampling we run the following algorithm:

1. set t = 0
2. generate an initial state $x^{(0)}$ from a prior distribution $\pi^{(0)}$ over initial states
3. repeat until $t = M$

set $t = t + 1$

generate a proposal state x^* from $q(x|x^{(t-1)})$

calculate the acceptance probability $\alpha = \min\left(1, \frac{p(x^*)}{p(x^{(t-1)})}\right)$

draw a random number u from $\mathrm{Unif}(0, 1)$

if $u \leq \alpha$, accept the proposal and set $x^{(t)} = x^*$

else set $x^{(t)} = x^{(t-1)}$

Example: Using the Metropolis algorithm to sample from an unknown distribution

Say that we have some mysterious function

$p(x) = (1 + x^2)^{-1}$

from which we would like to draw samples. To do so using Metropolis sampling we need to define two things: (1) the prior distribution $\pi^{(0)}$ over the initial state of the Markov chain, and (2) the proposal distribution $q(x|x^{(t-1)})$. For this example we define:

$\pi^{(0)} \sim \mathcal{N}(0, 1)$

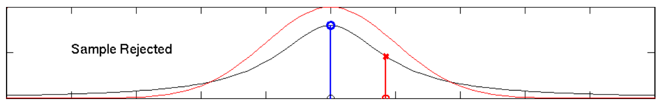
$q(x|x^{(t-1)}) \sim \mathcal{N}(x^{(t-1)}, 1)$,

both of which are simply a Normal distribution, one centered at zero, the other centered at previous state of the chain. The following chunk of MATLAB code runs the Metropolis sampler with this proposal distribution and prior.

```
% METROPOLIS SAMPLING EXAMPLE
randn('seed',12345);

% DEFINE THE TARGET DISTRIBUTION
p = inline('(1 + x.^2).^-1','x')

% SOME CONSTANTS
nSamples = 5000;
burnIn = 500;
nDisplay = 30;
sigma = 1;
minn = -20; maxx = 20;
xx = 3*minn:.1:3*maxx;
target = p(xx);
pauseDur = .8;

% INITIALZE SAMPLER
x = zeros(1 ,nSamples);
x(1) = randn;
t = 1;

% RUN SAMPLER
while t < nSamples
    t = t+1;

    % SAMPLE FROM PROPOSAL
    xStar = normrnd(x(t-1) ,sigma);
    proposal = normpdf(xx,x(t-1),sigma);

    % CALCULATE THE ACCEPTANCE PROBABILITY
    alpha = min([1, p(xStar)/p(x(t-1))]);

    % ACCEPT OR REJECT?
    u = rand;
    if u < alpha
        x(t) = xStar;
        str = 'Accepted';
    else
        x(t) = x(t-1);
        str = 'Rejected';
    end

    % DISPLAY SAMPLING DYNAMICS
    if t < nDisplay + 1
        figure(1);
        subplot(211);
        cla
        plot(xx,target,'k');
        hold on;
        plot(xx,proposal,'r');
        line([x(t-1),x(t-1)],[0 p(x(t-1))],'color','b','linewidth',
        scatter(xStar,0,'ro','Linewidth',2)
        line([xStar,xStar],[0 p(xStar)],'color','r','Linewidth',2)
        plot(x(1:t),zeros(1,t),'ko')
        legend({'Target','Proposal','p(x^{(t-1)})','x^*','p(x^*)','

        switch str
            case 'Rejected'
```



```
                scatter(xStar,p(xStar),'rx','Linewidth',3)
            case 'Accepted'
                scatter(xStar,p(xStar),'rs','Linewidth',3)
        end
            scatter(x(t-1),p(x(t-1)),'bo','Linewidth',3)
        title(sprintf('Sample % d %s',t,str))
        xlim([minn,maxx])
        subplot(212);
        hist(x(1:t),50); colormap hot;
        xlim([minn,maxx])
        title(['Sample ',str]);
        drawnow
        pause(pauseDur);
    end
end

% DISPLAY MARKOV CHAIN
figure(1); clf
subplot(211);
stairs(x(1:t),1:t, 'k');
hold on;
hb = plot([-10 10],[burnIn burnIn],'b--')
ylabel('t'); xlabel('samples, x');
set(gca , 'YDir', 'reverse');
ylim([0 t])
axis tight;
xlim([-10 10]);
title('Markov Chain Path');
legend(hb,'Burnin');

% DISPLAY SAMPLES
subplot(212);
nBins = 200;
sampleBins = linspace(minn,maxx,nBins);
counts = hist(x(burnIn:end), sampleBins);
bar(sampleBins, counts/sum(counts), 'k');
xlabel('samples, x' ); ylabel( 'p(x)' );
title('Samples');

% OVERLAY ANALYTIC DENSITY OF STUDENT T
nu = 1;
y = tpdf(sampleBins,nu)
hold on;
plot(sampleBins, y/sum(y) , 'r-', 'LineWidth', 2);
legend('Samples',sprintf('Theoretic\nStudent''s t'))
axis tight
xlim([-10 10]);
```



(https://theclevermachine.files.wordpress.com/2012/10/metropolis2.gif)
Using the Metropolis algorithm to sample from a continuous distribution (black)

In the figure above, we visualize the first 50 iterations of the Metropolis sampler.The black curve represents the target distribution $p(x)$. The red curve that is bouncing about the x-axis is the proposal distribution $q(x|x^{(t-1)})$ (if the figure is not animated, just click on it). The vertical blue line (about which the bouncing proposal distribution is centered) represents the quantity $p(x^{(t-1)})$, and the vertical red line represents the quantity $p(x^*)$, for a proposal state x^* sampled according to the red curve. At every iteration, if the vertical red line is longer than the blue line, then the sample x^* is accepted, and the proposal distribution becomes centered about the newly accepted sample. If the blue line is longer, the sample is randomly rejected or accepted.

But why randomly keep "bad" proposal samples? It turns out that doing this allows the Markov chain to every-so-often visit states of low probability under the target distribution. This is a desirable property if we want the chain to adequately sample the entire target distribution, including any tails.

An attractive property of the Metropolis algorithm is that the target distribution $p(x)$ does not have to be a properly normalized probability distribution. This is due to the fact that the acceptance probability is based on the ratio of two values of the target distribution. I'll show you what I mean. If $p(x)$ is an unnormalized distribution and

$$p^*(x) = \frac{p(x)}{Z}$$

is a properly normalized probability distribution with normalizing constant Z, then

$$p(x) = Zp^*(x)$$

and a ratio like that used in calculating the acceptance probability α is

$$\frac{p(a)}{p(b)} = \frac{Zp^*(a)}{Zp^*(b)} = \frac{p^*(a)}{p^*(b)}$$

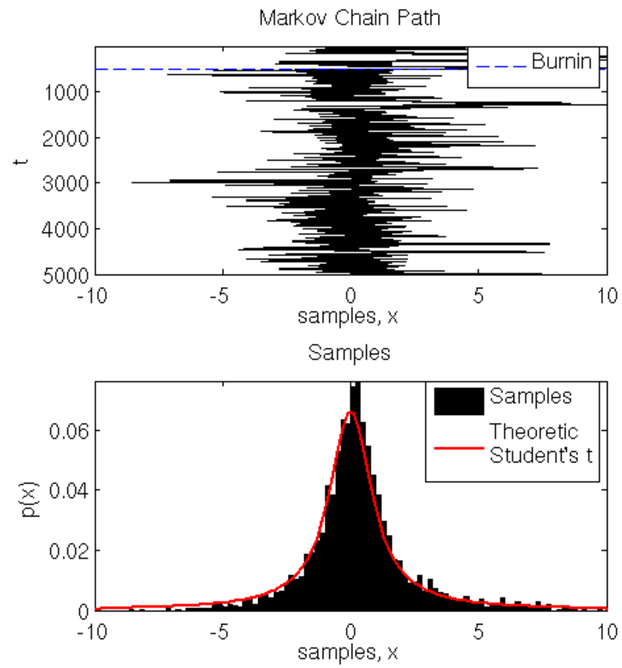
The normalizing constants Z cancel! This attractive property is quite useful in the context of Bayesian methods, where determining the normalizing constant for a distribution may be impractical to calculate directly. This property is demonstrated in current example. It turns out that the "mystery" distribution that we sampled from using the Metropolis algorithm is an unnormalized form of the Student's-t distribution with one degree of freedom. Comparing $p(x)$ to the definition of the definition Student's-t

$$Student(x,\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}} = \frac{(1+x^2)^{-1}}{Z} = \frac{p(x)}{Z}$$

we see that $p(x)$ is a Student's-t distribution with degrees of freedom $\nu = 1$, but missing the normalizing constant

$$Z = \left(\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\right)^{-1}$$

Below is additional output from the code above showing that the samples from Metropolis sampler draws samples that follow a *normalized* Student's-t distribution, even though $p(x)$ is not normalized.



(https://theclevermachine.files.wordpress.com/2012/10/metropolisstudentst2.png)
Metropolis samples from an unnormalized t-distribution follow the normalized distribution

The upper plot shows the progression of the Markov chain's progression from state $x^{(0)}$ (top) to state $x^{(5000)}$ (bottom). The burn in period for this chain was chosen to be 500 transitions, and is indicated by the dashed blue line (for more on burnin see this previous post (https://theclevermachine.wordpress.com/2012/09/24/a-brief-introduction-to-markov-chains/)).

The bottom plot shows samples from the Markov chain in black (with burn in samples removed). The theoretical curve for the Student's-t with one degree of freedom is overlayed in red. We see that the states kept by the Metropolis sampler transition operator sample from values that follow the Student's-t, even though the function $p(x)$ used in the transition operator was not a properly normalized probability distribution.

Reversibility of the transition operator

It turns out that there is a theoretical constraint on the Markov chain the transition operator in order for it settle into a stationary distribution (i.e. a target distribution we care about). The constraint states that the probability of the transition $x^{(t)} \to x^{(t+1)}$ must be equal to the probability of the reverse transition $x^{(t+1)} \to x^{(t)}$. This reversibility property is often referred to as ***detailed balance***. Using the Metropolis algorithm transition operator, reversibility is assured if the proposal distribution

$q(x|x^{(t-1)})$ is symmetric. Such symmetric proposal distributions are the Normal, Cauchy, Student's-t, and Uniform distributions.

However, using a symmetric proposal distribution may not be reasonable to adequately or efficiently sample all possible target distributions. For instance if a target distribution is bounded on the positive numbers $0 < x \leq \infty$, we would like to use a proposal distribution that has the same support, and will thus be assymetric. This is where the ***Metropolis-Hastings*** sampling algorithm comes in. We will discuss in a later post how the Metropolis-Hastings sampler uses a simple change to the calculation of the acceptance probability which allows us to use non-symmetric proposal distributions.

Posted in Algorithms, Sampling Methods, Statistics

Tags: Acceptance Probability, Detailed Balance, Markov Chain Monte Carlo, MCMC, Metropolis sampling, Metropolis-Hastings Sampling, Proposal Distribution, Reversibility, Target Distribution

3 Comments

Rejection Sampling

SEP 10
Posted by **dusti nstansbury**

Suppose that we want to sample from a distribution $f(x)$ that is difficult or impossible to sample from directly, but instead have a simpler distribution $q(x)$ from which sampling is easy. The idea behind Rejection sampling (aka Acceptance-rejection sampling) is to sample from $q(x)$ and apply some rejection/acceptance criterion such that the samples that are accepted are distributed according to $f(x)$.

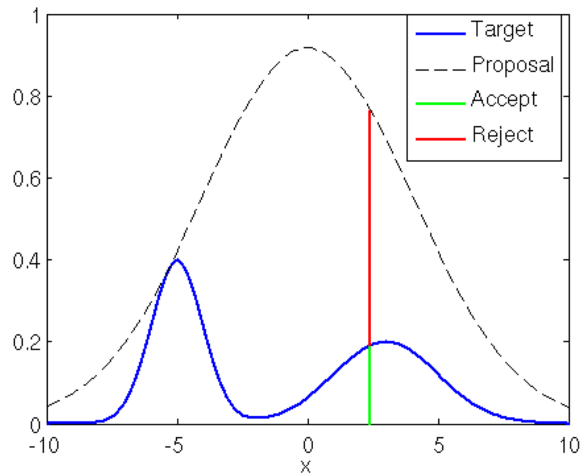
Envelope distribution and rejection criterion

In order to be able to reject samples from $q(x)$ such that they are sampled from $f(x)$, $q(x)$ must "cover" or envelop the distribution $f(x)$. This is generally done by choosing a constant $c > 1$ such that $cq(x) > f(x)$ for all x. For this reason $cq(x)$ is often called the *envelope distribution*. A common criterion for accepting samples from $x \sim q(x)$ is based on the ratio of the target distribution to that of the envelope distribution. The samples are accepted if

$\frac{f(x)}{cq(x)} > u$

where $u \sim Unif(0,1)$, and rejected otherwise. If the ratio is close to one, then $f(x)$ must have a large amount of probability mass around x and that sample should be more likely accepted. If the ratio is small, then it means that $f(x)$ has low probability mass around x and we should be less likely to accept the sample. This criterion is demonstrated in the chunk of MATLAB code and the resulting figure below:


```
rand('seed',12345);
x = -10:.1:10;
```

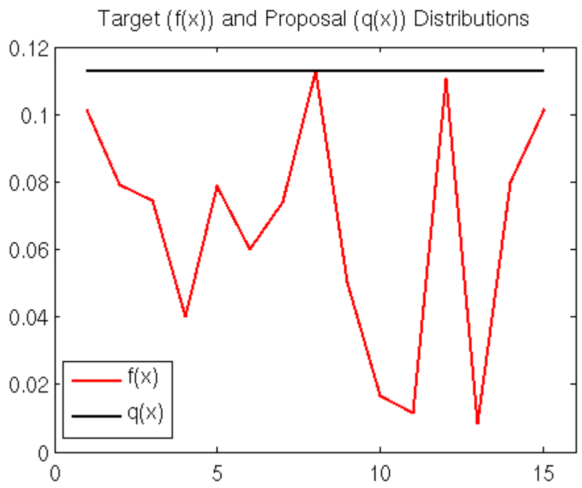


(https://theclevermachine.files.wordpress.com/2012/09/rejectionsamplingcriterion.png)
Rejection Sampling with a Normal proposal distribution

Here a zero-mean Normal distribution is used as the proposal distribution. This distribution is scaled by a factor $c = 9.2$, determined from $f(x)$ and $q(x)$ to ensure that the proposal distribution covers $f(x)$. We then sample from $q(x)$, and compare the proportion of $cq(x)$ occupied by $f(x)$. If we compare this proportion to a random number sampled from $Unif(0,1)$ (i.e. the criterion outlined above), then we would accept this sample with probability proportional to the length of the green line segment and reject the sample with probability proportional to the length of the red line segment.

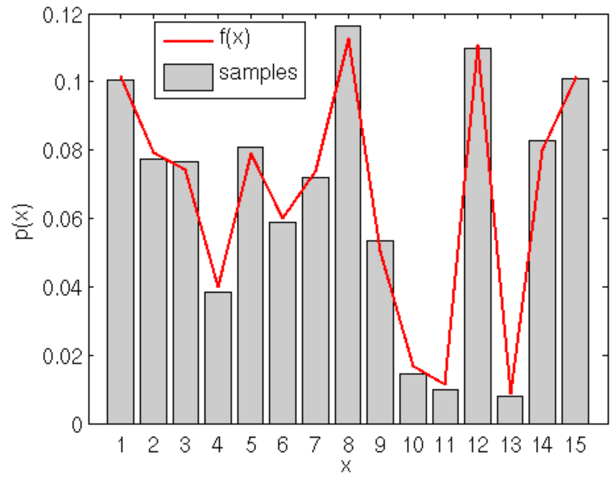
Rejection sampling of a random discrete distribution

This next example shows how rejection sampling can be used to sample from any arbitrary distribution, continuous or not, and with or without an analytic probability density function.



(https://theclevermachine.files.wordpress.com/2012/09/rejectionsamplingtargetproposal.png)
Random Discrete Target Distribution and Proposal that Bounds It.

The figure above shows a random ***discrete*** probability density function $f(x)$ generated on the interval (0,15). We will use rejection sampling as described above to sample from $f(x)$. Our proposal/envelope distribution is the uniform discrete distribution on the same interval (i.e. any of the integers from 1-15 are equally probable) multiplied by a constant c that is determined such that the maximum value of $f(x)$ lies under (or equal to) $cq(x)$.



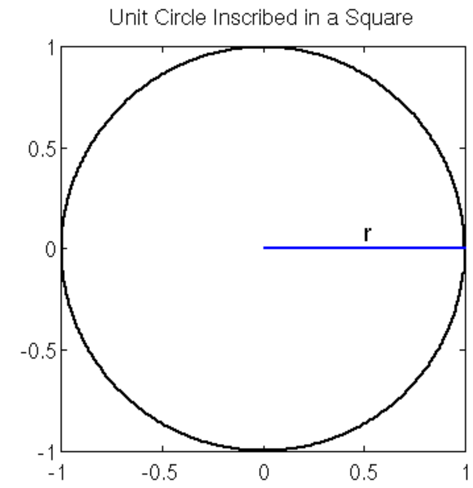
(https://theclevermachine.files.wordpress.com/2012/09/rejectionsamplingdiscrete.png)
Rejection Samples For Discrete Distribution on interval [1 15]

Plotted above is the target distribution (in red) along with the discrete samples obtained using the rejection sampling. The MATLAB code used to sample from the target distribution and display the plot above is here:

```
rand('seed',12345)
randn('seed',12345)

fLength = 15;
% CREATE A RANDOM DISTRIBUTION ON THE INTERVAL [1 fLength]
f = rand(1,fLength); f = f/sum(f);

figure; h = plot(f,'r','Linewidth',2);
hold on;
l = plot([1 fLength],[max(f) max(f)],'k','Linewidth',2);

legend([h,l],{'f(x)','q(x)'},'Location','Southwest');
xlim([0 fLength + 1])
xlabel('x');
ylabel('p(x)');
title('Target (f(x)) and Proposal (q(x)) Distributions');

% OUR PROPOSAL IS THE DISCRETE UNIFORM ON THE INTERVAL [1 fLength]
% SO OUR CONSTANT IS
c = max(f/(1/fLength));

nSamples = 10000;
i = 1;
while i < nSamples
   proposal = unidrnd(fLength);
   q = c*1/fLength; % ENVELOPE DISTRIBUTION
   if rand < f(proposal)/q
      samps(i) = proposal;
      i = i + 1;
```

Rejection sampling from the unit circle to estimate π

Though the ratio-based acceptance-rejection criterion introduced above is a common choice for drawing samples from complex distributions, it is not the only criterion we could use. For instance we could use a different set of criteria to generate some geometrically-bounded distribution. If we wanted to generate points uniformly within the unit circle (i.e. a circle centered at $(y, x) = 0$ and with radius $r = 1$), we could do so by sampling Cartesian spatial coordinates x and y uniformly from the interval (-1,1)–which samples form a square centered at (0,0)–and reject those points that lie outside of the radius $r = \sqrt{x^2 + y^2} = 1$



(https://theclevermachine.files.wordpress.com/2012/09/unitcircleinsquare2.png)
Unit Circle Inscribed in Square

Something clever that we can do with such a set of samples is to approximate the value π: Because a square that inscribes the unit circle has area:
$A_{square} = (2r)^2 = 4r^2$

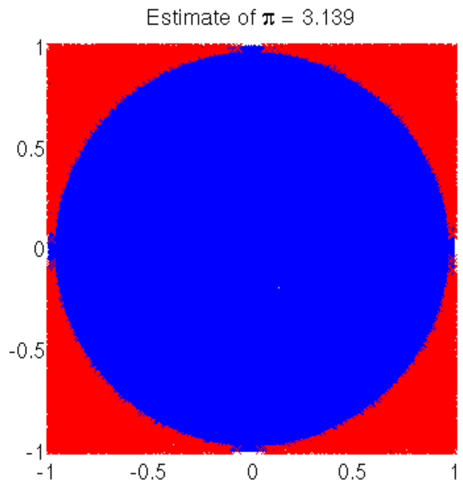
and the unit circle has the area:

$A_{circle} = \pi r^2$

We can use the ratio of their areas to approximate π:

$\pi = 4\frac{A_{circle}}{A_{square}}$

The figure below shows the rejection sampling process and the resulting estimate of π from the samples. One-hundred thousand 2D points are sampled uniformly from the interval (-1,1). Those points that lie within the unit circle are plotted as blue dots. Those points that lie outside of the unit circle are plotted as red x's. If we take four times the ratio of the area in blue to the entire area, we get a very close approximation to 3.14 for π.



(https://theclevermachine.files.wordpress.com/2012/09/rejectionsamplingpi.png)
Rejection Criterion

The MATLAB code used to generate the example figures is below:


```
% DISPLAY A CIRCLE INSCRIBED IN A SQUARE

figure;
a = 0:.01:2*pi;
x = cos(a); y = sin(a);
hold on
plot(x,y,'k','Linewidth',2)

t = text(0.5, 0.05,'r');
l = line([0 1],[0 0],'Linewidth',2);
axis equal
box on
xlim([-1 1])
ylim([-1 1])
title('Unit Circle Inscribed in a Square')

pause;
rand('seed',12345)
randn('seed',12345)
delete(l); delete(t);

% DRAW SAMPLES FROM PROPOSAL DISTRIBUTION
samples = 2*rand(2,100000) - 1;

% REJECTION
reject = sum(samples.^2) > 1;

% DISPLAY REJECTION CRITERION
scatter(samples(1,~reject),samples(2,~reject),'b.')
scatter(samples(1,reject),samples(2,reject),'rx')
hold off
```

Wrapping Up

Rejection sampling is a simple way to generate samples from complex distributions. However, Rejection sampling also has a number of weaknesses:

- Finding a proposal distribution that can cover the support of the target distribution is a non-trivial task.
- Additionally, as the dimensionality of the target distribution increases, the proportion of points that are rejected also increases. This curse of dimensionality makes rejection sampling an inefficient technique for sampling multi-dimensional distributions, as the majority of the points proposed are not accepted as valid samples.
- Some of these problems are solved by changing the form of the proposal distribution to “hug” the target distribution as we gain knowledge of the target from observing accepted samples. Such a process is called *Adaptive Rejection Sampling*, which will be covered in another post.

Posted in Density Estimation, Sampling Methods, Statistics

Tags: Curse of Dimensionality, Envelope Distribution, Proposal Distribution, Rejection Sampling, Sampling Methods, Target Distribution

4 Comments