Computational Physics – Lecture 24

(13 May 2020)

Recap

In the previous lecture, we learnt Monte Carlo methods for integrating functions.

Today's plan

In today's class we want to learn Monte Carlo methods to (a) sample functions and (b) optimise functions.

MCMC

We have learnt two methods of sampling from a general, non-uniform PDF: the transformation method, and the rejection method.

Both methods require a full description of the properly normalised PDF.

There is another method of sampling that does not need this. All it needs is that you be able to compute the ratios of the PDF at two locations.

This method is called Markov Chain Monte Carlo (MCMC). It has transformed the sciences.

MCMC

MCMC is thus a method of sampling random numbers from PDFs.

It has three interesting properties:

- It does not need the PDF to be normalised
- It doesn't require any derivative or integrals of the PDF
- And it is extremely easy to code

Further, MCMC allows the computation of various integrals involving the PDF (such as expectation values) without even knowing the normalisation.

MCMC is therefore particularly desirable for complicated PDFs, such as those in higher dimensions.

MCMC

A sample $\{\theta_k\}_{k=1}^{K}$ from some PDF $p(\theta)$ is a set of k values θ_k drawn from the PDF.

We can define the expectation value of θ as

$$E_{p(\theta)}[\theta] \equiv \int \theta \, p(\theta) \, \mathrm{d}\theta \tag{1}$$

With a good sample $\{\theta_k\}_{k=1}^{K}$ we can accurately compute the expectation value as a sum

$$E_{p(\theta)}[\theta] \approx \frac{1}{K} \sum_{k=1}^{K} \theta_k \tag{2}$$

MCMC

We can even compute the expectation value of any quantity that can be expressed as a function $g(\theta)$ of θ as

$$E_{p(\theta)}[g(\theta)] \equiv \int g(\theta)\, p(\theta)\, \mathrm{d}\theta \tag{3}$$

Again, with a good sample $\{\theta_k\}_{k=1}^{K}$ in hand, we can compute this as a sum

$$E_{p(\theta)}[g(\theta)] \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k) \tag{4}$$

MCMC

Often, however, we have a badly normalised function $f(\theta)$ instead of the normalised PDF $p(\theta)$.

So $f(\theta)$ is just the PDF $p(\theta)$ multiplied by some unknown and hard-to-compute scalar.

The only thing we know is that $f(\theta)$ is non-negative everywhere and has a finite integral Z

$$0 \leq f(\theta) \quad \text{for all } \theta \tag{5}$$

$$Z = \int f(\theta) \, \mathrm{d}\theta \tag{6}$$

(We don't actually know the value of Z, but we do know that it is finite.)

MCMC

When the sample $\{\theta_k\}_{k=1}^{K}$ is of $f(\theta)$, MCMC allows you to compute the expectation value

$$E_{p(\theta)}[g(\theta)] \equiv \frac{\int g(\theta)\, f(\theta)\, \mathrm{d}\theta}{\int f(\theta)\, \mathrm{d}\theta} \tag{7}$$

as

$$E_{p(\theta)}[g(\theta)] \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k) \tag{8}$$

without integrating either the numerator integral or the denominator integral (both of which are generally hard).

MCMC

Two common applications of MCMC in computational physics are (a) statistical physics and (b) probabilistic inference ("fitting a model to data").

In statistical physics, you might know that the probability of a state E is $e^{-E/k_\mathrm{B}T}$, but you might not know the normalisation Z (the partition function).

In probabilistic inference, you typically have PDFs of several theory parameters given experimental data. These PDFs are often poorly understood.

MCMC

Here is how MCMC works.

You need two things before you start:

- Your function $f(\theta)$, which is the function to be sampled, and
- a proposal PDF $q(\theta' \mid \theta)$

MCMC

The algorithm is iterative, so if you have sample θ_k, to generate the next sample θ_{k+1}

- Get a new θ' from the proposal PDF $q(\theta' \,|\, \theta_k)$
- Draw a random number $0 < r < 1$ from the uniform distribution.
- If $f(\theta')/f(\theta_k) > r$ then $\theta_{k+1} = \theta'$; otherwise $\theta_{k+1} = \theta_k$.

Here, the main user-settable knob is the proposal pdf $q(\theta' \,|\, \theta)$.

The PDF q must satisfy a "detailed-balance" condition $q(\theta' \,|\, \theta) = q(\theta \,|\, \theta')$.

Typical choices are Gaussian or Uniform distribution.

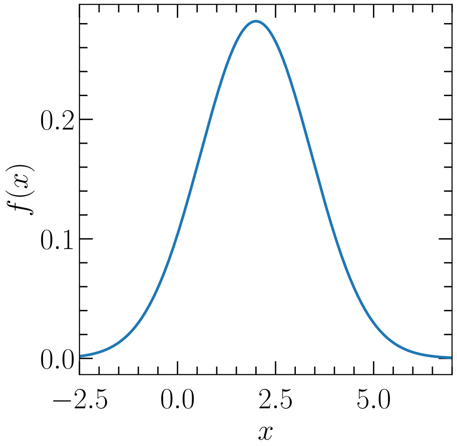
MCMC

Let us use the MCMC method to sample from the following PDF:

$$p(\theta) = \frac{1}{\sqrt{4\pi}}\, e^{-\frac{1}{2}\left(\frac{x-2}{\sqrt{2}}\right)^2}. \tag{9}$$

This is a Gaussian with mean 2 and variance 2.

MCMC



MCMC

Let us choose another Gaussian as our $q(\theta' \mid \theta)$ so that

$$q(\theta' \mid \theta) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}(\theta'-\theta)^2}. \tag{10}$$

This is a Gaussian with mean θ and variance 1.

MCMC

Our code could then look like:

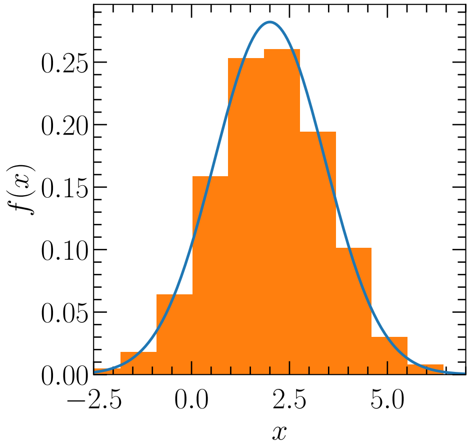
```
from scipy.stats import norm

nsteps = 10000
theta = 0.0

for i in range(nsteps):
   theta_prime = theta + np.random.standard_normal()
   r = np.random.rand()

   if norm.pdf(theta_prime, loc=mu, scale=sigma)/
      norm.pdf(theta, loc=mu, scale=sigma) > r:
        theta = theta_prime
```

MCMC



MCMC

Suppose instead of

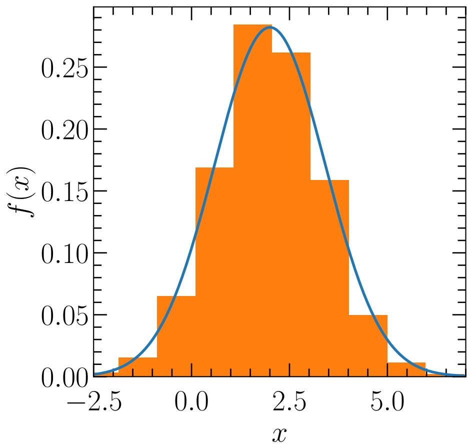
$$p(\theta) = \frac{1}{\sqrt{4\pi}}\, e^{-\frac{1}{2}\left(\frac{x-2}{\sqrt{2}}\right)^2}. \tag{11}$$

we took

$$f(\theta) = e^{-\frac{1}{2}\left(\frac{x-2}{\sqrt{2}}\right)^2}. \tag{12}$$

We still get a good sample!

MCMC

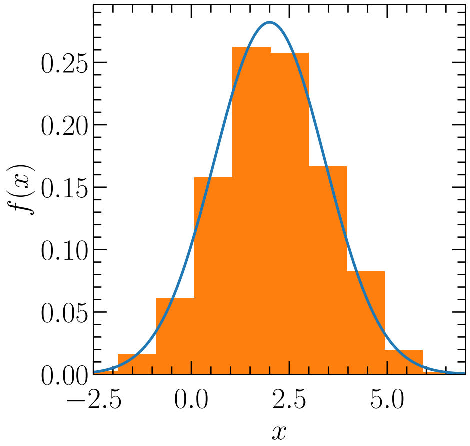


MCMC

We would still get a good sample if we took

$$f(\theta) = 100 \times e^{-\frac{1}{2}\left(\frac{x-2}{\sqrt{2}}\right)^2}. \tag{13}$$

MCMC



MCMC

People have developed many variations of the above MCMC method.

The version that we have just seen is called the “Metropolis algorithm” or the “Metropolis MCMC”.

Some other variants of this method are:

- Metropolis-Hastings algorithm
- Ensemble method
- Gibbs sampling
- Hamiltonian MCMC
- Nested sampling
- Parallel Tempering MCMC
- Reversible Jump MCMC

MCMC

Why did the Metropolis method work?

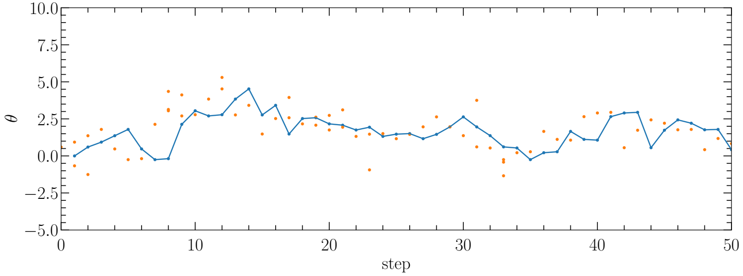
We can derive some inituitive understanding from the last step: "If $f(\theta')/f(\theta_k) > r$ then $\theta_{k+1} = \theta'$; otherwise $\theta_{k+1} = \theta_k$."

So if $f(\theta')/f(\theta_k)$ is large, there is a high chance that we would move to a new point. If this ratio is small, we will not move.

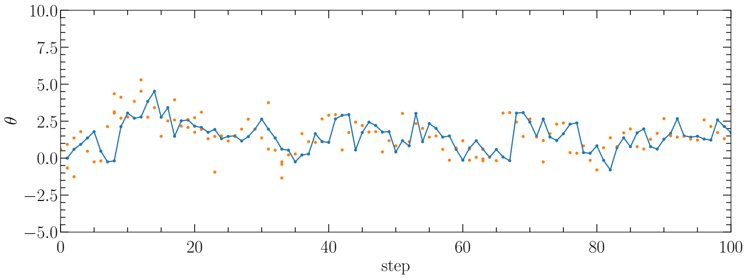
But $f(\theta')/f(\theta_k) = p(\theta')/p(\theta_k)$, so we will tend to stay in regions where $p(\theta)$ is high.

That gives us a sample of $p(\theta)$.

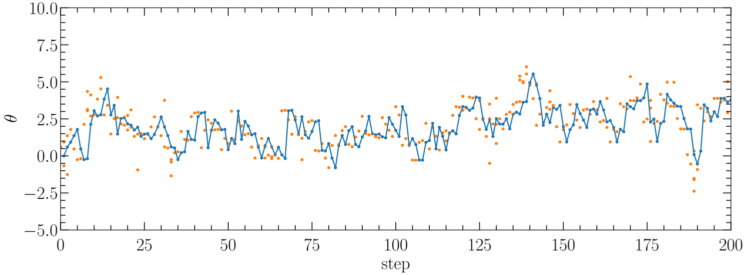
MCMC



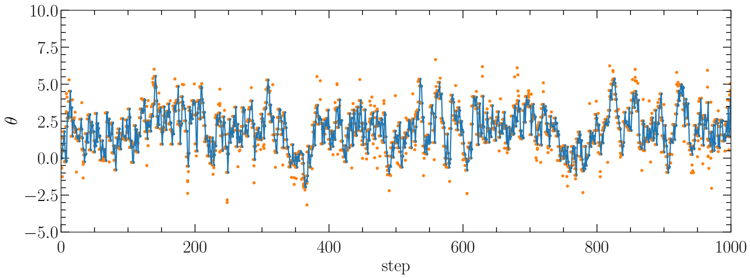
MCMC



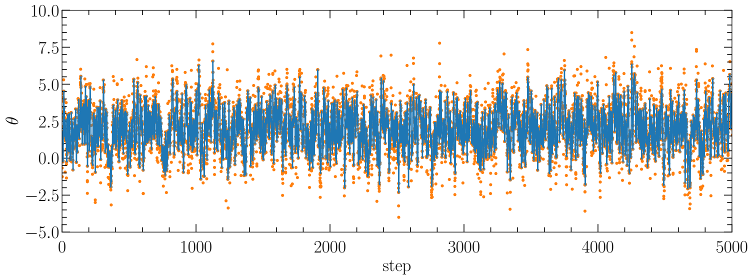
MCMC



MCMC



MCMC



MCMC

Note that although the Metropolis algorithm is guaranteed to converge to the correct PDF, it can take a long time to do so.

Unfortunately, “How long?” is a hard question!

Typically, people do the following heuristic test: you have sampled long enough when you can see that the chain has traversed the high-probability parts many times.

Or, equivalently, you have sampled long enough when the first half of the chain shows very much the same histogram as the second half of the chain, or as any substantial subset.

Or, the auto-correlation of the chain has reduced to zero.

MCMC

Notice how easy it is to generalise our code to higher-dimensions

```
from scipy.stats import norm

ndim = 10
nsteps = 10000
theta = np.zeros(ndim)

for i in range(nsteps):
   theta_prime = theta +
                np.random.standard_normal(ndim)
   r = np.random.rand()

   if norm.pdf(theta_prime, loc=mu, scale=sigma)/
      norm.pdf(theta, loc=mu, scale=sigma) > r:
        theta = theta_prime
```

MCMC

How do we choose the q function?

The heuristic idea here is the if q is very broad then a lot of points will be rejected: unnecessarily long time spent in low probability regions.

If q is very narrow, the chain is "timid". Most points will be accepted: θ will remain in the high probability region.

There is a middle ground, which you can hit by monitoring the acceptance fraction of your points.

Activity

Use the Metropolis algorithm to get a sample from a density that is uniform for $3 < x < 7$ and zero elsewhere.

Simulated Annealing

Another application of random numbers is in optimisation of functions, that is, finding the minimum or maximum of a function.

One such Monte Carlo method is called “Simulated Annealing”.

Simulated Annealing uses the Metropolis Algorithm to find the optimum of a function.

Simulated Annealing

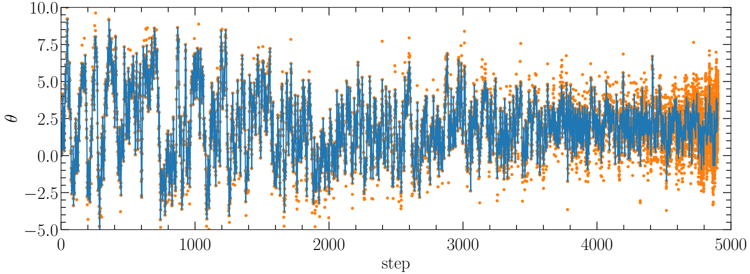
Recall the last step of the Metropolis algorithm: "If $f(\theta')/f(\theta_k) > r$ then $\theta_{k+1} = \theta'$; otherwise $\theta_{k+1} = \theta_k$."

If we change this to: "If $c(k) \cdot f(\theta')/f(\theta_k) > r$ then $\theta_{k+1} = \theta'$; otherwise $\theta_{k+1} = \theta_k$", where $c(k)$ is a decreasing function of k.

Then as the step number k increases, our chain will become increasingly timid: it will get stuck in the region where $f(\theta)$ is maximum.

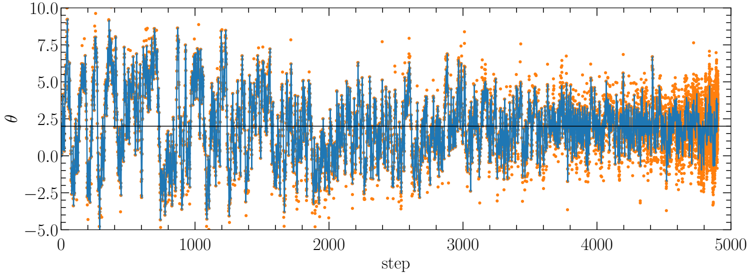
We can use this to find the maximum of $f(\theta)$.

Simulated Annealing



Result for $c(k) = 15 \cdot e^{-k/1500}$.

Simulated Annealing



Result for $c(k) = 15 \cdot e^{-k/1500}$.

Simulated Annealing

The function $c(k)$ is called the “Annealing Schedule”.

Choosing a better Annealing Schedule will take you closer to the optimum and faster.

The Annealing Schedule is a decreasing function of k. It also has to decrease slowly enough. Otherwise, you could get stuck in a local optimum.

Hence the name “annealing”: “heat (metal or glass) and allow it to cool slowly”.

Simulated Annealing

In statistical physics, MCMC and Simulated Annealing can together be used to compute ensemble averages of various physical quantities over the Boltzmann distribution as well as to find ground states of systems.

Beyond statistical physics, these Monte Carlo methods shine in probabilistic inference using Bayesian statistics.