Physics through Computational Thinking

The Monte Carlo method

Auditya Sharma and Ambar Jain
Dept. of Physics, IISER Bhopal

Outline

In this module we look at

1. The one dimensional Ising model.
2. The Metropolis algorithm.

The Ising Model

```
Clear["Global`*"]
```

Consider the one-dimensional Ising model whose Hamiltonian is given by

$$H = -J \sum_{i=1}^{N} S_i S_{i+1} \tag{1}$$

where periodic boundary conditions are assumed, and the spins S_i can take values ± 1. For simplicity, let us take this to be a ferromagnetic system, which means that J is positive. We assume that the system is in equilibrium at a temperature T. Implicitly, the assumption is that the system is being treated in what is called the canonical ensemble. The statistical mechanical problem involves understanding how various system quantities behave as a function of the temperature T. It turns out that if we are able to compute the partition function:

$$Z = \mathrm{Tr}\left(e^{-\beta H}\right), \tag{2}$$

all the equilibrium quantities of interest can then be computed from there. In general if we are interested in the average value of any operator O, the way to compute its equilibrium average is simply given by

$$< O > = \frac{\mathrm{Tr}\left(O\, e^{-\beta H}\right)}{\mathrm{Tr}\left(e^{-\beta H}\right)}. \tag{3}$$

In general, obtaining an analytical closed form expression for the partition function is a very hard problem, and in practically impossible. The one-dimensional Ising model is an exception, where in fact, the exact partition function can be computed exactly. It is beyond the scope of the present discussion to show how this is done. However, we will just simply state one key result which follows from this computation. The average energy per spin as a function of the inverse temperature is simply given by

$$\frac{E}{N} = -J \tanh(\beta J). \tag{4}$$

The Metropolis Algorithm

Since the computation of averages is really a matter of computing sums like in the expression

$$< O > = \frac{\mathrm{Tr}\left(O\, e^{-\beta H}\right)}{\mathrm{Tr}\left(e^{-\beta H}\right)}, \tag{5}$$

we can expect that a Monte Carlo method that carries out an importance sampling, may be able to effectively evaluate it. Indeed, this turns out to be the case. Although there are many algorithms which can carry this out, we present here just the prescription of the most famous algorithm: the Metropolis algorithm.

The algorithm can be outlined in the following steps:

- Construct an array of spins, each of which can take the values ± 1.
- Start with a random initial configuration of spins.
- Sequentially scan through the all the spins, and attempt to flip the spin. If such a spin flip results in a decrease in energy, flip the spin with probability one. If such a spin flip results in no change in energy, flip the spin with probability $\frac{1}{2}$. If on the other hand, the spin flip would cause an increase in energy by an amount ΔE, then the flip is carried out with probability

$$P(\text{flip}) = \frac{e^{-\beta \Delta \mathrm{E}}}{1 + e^{-\beta \Delta \mathrm{E}}}. \tag{6}$$

- After a large number of iterations, the system equilibrates to the temperature T, after which more iterations are carried out to measure quantities of interest. An average of the quantity of interest over these equilibrated configurations, converges to the statistical mechanical average.

A detailed understanding of the theory behind this and other related algorithms is way out of the scope of the current course. Moreover, even a professional implementation of these algorithms for various spin systems itself is a refined art form, that may take even years to master. Our goal here will be to demonstrate a simple implementation of this algorithm, and one quick check that it indeed works.

Monte Carlo simuation of the 1D Ising ferromagnet

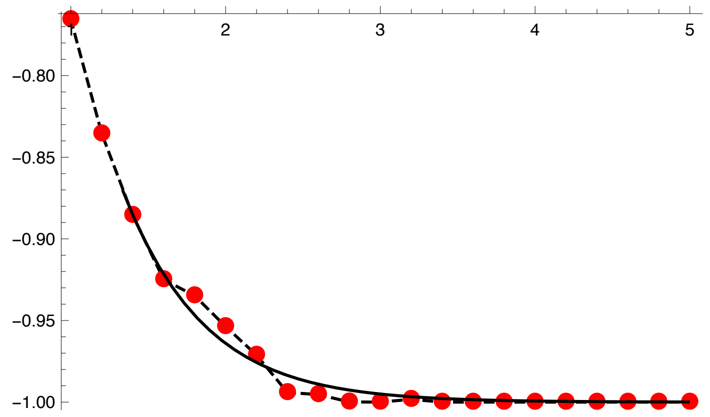
```
decide[x_, beta_] := If[x < 0, -1, If[x == 0, RandomChoice[{-1, 1}], If[RandomReal[{0, 1}] < Exp[-beta x]/(1 + Exp[-beta x]), -1, 1]]];

monteCarlo[nspins_, beta_, num_] := Module[{spins, hp, deltaE, mag, kleft, kright},
  spins = Table[RandomChoice[{-1, 1}], {nspins}];
  hp = Table[(spins[[Mod[k - 1, nspins] + 1]] + spins[[Mod[k + 1, nspins] + 1]])/(-2), {k, 0, nspins - 1}];
  Table[Table[
     deltaE = -4 hp[[k]] spins[[k]];
     spins[[k]] = spins[[k]] decide[deltaE, beta];
     kleft  = Mod[k - 2, nspins] + 1;
     kright  = Mod[k, nspins] + 1;
     hp[[kleft]] = (spins[[Mod[kleft - 2, nspins] + 1]] + spins[[Mod[kleft, nspins] + 1]])/(-2);
     hp[[kright]] = (spins[[Mod[kright - 2, nspins] + 1]] + spins[[Mod[kright, nspins] + 1]])/(-2);
     , {k, 1, nspins}];, {2^(num-1)}];
  AvgEn = Mean[Table[Table[
      deltaE = -4 hp[[k]] spins[[k]];
      spins[[k]] = spins[[k]] decide[deltaE, beta];
      kleft  = Mod[k - 2, nspins] + 1;
      kright  = Mod[k, nspins] + 1;
      hp[[kleft]] = (spins[[Mod[kleft - 2, nspins] + 1]] + spins[[Mod[kleft, nspins] + 1]])/(-2);
      hp[[kright]] = (spins[[Mod[kright - 2, nspins] + 1]] + spins[[Mod[kright, nspins] + 1]])/(-2);
      , {k, 1, nspins}]; (spins.hp)/nspins, {2^(num-1)}]];
  Return[AvgEn] // N
 ]
```

```
en = Table[{beta, monteCarlo[64, beta, 13]}, {beta, 1, 5, 0.2}]
```

```
{{1., -0.765686}, {1.2, -0.835648}, {1.4, -0.885574}, {1.6, -0.924957}, {1.8, -0.934814}, {2., -0.953705},
 {2.2, -0.971237}, {2.4, -0.994125}, {2.6, -0.995361}, {2.8, -0.999893}, {3., -0.999985}, {3.2, -0.998276},
 {3.4, -1.}, {3.6, -1.}, {3.8, -1.}, {4., -1.}, {4.2, -1.}, {4.4, -1.}, {4.6, -1.}, {4.8, -1.}, {5., -1.}}
```

```
Show[ListPlot[en, Joined → True, PlotStyle -> Dashed, PlotMarkers → Style["●", 14, Red]], Plot[-Tanh[x], {x, 1, 5}]]
```


2
3
4
5
-0.80
-0.85
-0.90
-0.95
-1.00