4 Learning The Critical Temperature

In this exercise, you will be working with snapshots of dipole moments from a Monte Carlo simulation of the Ising model. The Ising model is the simplest theoretical model for magnets. Magnetic dipoles in a material can in general point to any direction. If these directions are random from one dipole to the next, their net magnetic field is zero and the material is said to be a paramagnet. In certain materials, interactions between the dipole moments (or spins) favors their alignment with each other, and at low temperatures, where these interactions become dominant, the material can spontaneously magnetize. That means the spins suddenly align with each other below a critical temperature, giving rise to a macroscopic magnetic field. A bar magnet at room temperature is an example of a magnetic material below the critical temperature. If you heat up a bar magnet, at some point it loses its magnetic behavior.

The Ising model captures the spontaneous magnetization phenomena by considering interacting spins on nodes of a square lattice. In this model, each spin can take only two directions, up or down, which we show mathematically by $+1$ and -1. The interactions are between nearest neighbors only and can be tuned to be ferromagnetic (favoring alignment of neighboring spins) or anti-ferrimagnetic (favoring anti-alignment of neighboring spins). You will be training an artificial neural network to detect the ferromagnetic phase transition in the square lattice Ising model as the temperature is lowered. The neural network does not know about the physics and makes this detection based only on spin configurations we provide at different temperatures.

The data you will be working with are stored in `Ising_Configs.dat`. They are for a 20×20 Ising system. The first 400 digits in each row of the file represent the orientation of the spins; $+1$ is spin up and -1 is spin down. The last column is the temperature at which the configuration has been generated. There are a total of 10,000 data points; 100 ordered temperatures and 100 configurations per temperature.

Start with the previous exercise for learning to multiply two numbers. Modify the code so that your ANN receives 400 binary inputs (± 1 spins) and gives a binary answer that is supposed to indicate **whether the configuration belongs to a temperature below T_c or above T_c** (take T_c to be 2.3). You may choose to have a single output neuron. In that case, `softmax` may not be a good choice for the activation function of the output neuron and you might want to consider `sigmoid` instead. How ever you design your output layer, there has to be a meaningful rule for determining the cost.

Among the things you might have to modify in the code are:

- The data size, and input to your network,
- The design of your network (start very simple –one hidden layer with only a few neurons),

- How you define "labels",
- How your cost function is defined, and
- How you transform the network output to a binary (hot/cold) output.

Note that since the configurations in the file are ordered based on their temperature, you have to **randomize them at the very beginning** right after the data are read from file and before the temperature is extracted. You may use something like `np.random.shuffle(data)`.

Since you have 400 input neurons, and therefore, much more weight and bias variables than in the original code, if the optimization turned out to be a lot slower, start with a small number of training configurations (e.g., < 1000) and slowly increase it if everything worked.

After the training is completed, plot the **average ANN output at each temperature** for the test subset vs temperature. You have to keep track of how many configurations at each temperature you have in the test set.

Has your ANN learned to distinguish configurations at $T > T_c$ from those at $T < T_c$? How can you tell? Ideally, your plot would look something like the "$L = 20$" curve in **Fig. 1a** of "Machine learning phases of matter" by Juan Carrasquilla and Roger G. Melko, Nat. Phys. 13, 431 434 (2017) at `https://www.nature.com/articles/nphys4035`

Other exercises:

- Limit your training to more extreme temperatures; those that are not too close to T_c and check whether your network can still locate T_c accurately.
- Try adding a "convolutional" layer with 1-2 "kernels" before the hidden layer. How does that affect the training progression?
- Investigate how the network is able to tell high-temperature and low-temperature configurations apart; for a simple fully-connected network, study the weights and biases before and after the training. For a convolutional neural network, visualize the kernels after the training.