Institute for Theoretical Physics
University of Cologne

Prof. Dr. S. Trebst
Jan Attig

Computational Many-Body Physics

Exercise Sheet 4

Summer Term 2018

Due date: Monday, **11th June** 2018, 2 pm

Website: www.thp.uni-koeln.de/trebst/Lectures/2018-CompManyBody.shtml

On this week's exercise sheet we want to introduce the concept of **machine learning** and its application to statistical many-body physics. We start by providing you with a relatively simple and straightforward julia code, which constructs a neural network to recognize hand-written numbers – this is the code that was showcased in the lecture on Wednesday.

We will then take this code and apply it in a physics context, by using it to classify configurations of the Ising model, to discriminate its two phases, and to identify the parametric location of the phase transition.

Exercise 12: Digit Recognition (Optional)

In this first (optional) exercise, we provide you with the sample digit recognition code that has been showcased in the lecture on Wednesday. It is able to convert 28×28 pixelated images of hand-written digits to their actual digital counterparts.

Your task is the following: Download the code from the website and run all cells from the notebook. Make sure that the code runs without problems and try to understand which part of the code is doing which task. Explain the main concepts of how machine learning is applied to the problem of digit recognition and document the code by adding markdown cells inbetween the julia code.

Exercise 13: Phase Recognition in the Ising model

In this exercise we want to apply the digit recognition code to the physics of the Ising model. The problem at hand is to identify the phase in which a given spin configuration is solely based on the configuration itself. This problem is indeed closely related to digit recognition as it is the goal in both cases to classify an element (image or Ising spin configuration) into a small amount of categories (digits or phases of the Ising model). You can read more about it in a recent publication, J. Carrasquilla and R. Melko, Nature Physics **13**, 431–434 (2017).

The main task of this exercise is to exchange the training data employed the code of exercise 12. Instead of supplying images of digits and their integer values, we now want to provide Ising spin configurations and assign them a label indicating the phase they belong to. This allows the identification of phases in newly generated configurations and the determination of the transition temperature between those phases as seen later on.

a) To get started, make sure you have a working Monte Carlo code that can simulate the ferromagnetic Ising model on the square lattice using either single-spin flip updates or the Swendsen-Wang cluster updates. You can also find one on the course website.

Next, implement the generation of training and testing data. To get started, you should choose one low and one high temperature point sufficiently far away from the phase transition (for the square lattice, you can choose, e.g., $T = 1.0$ and $T = 4.0$). Make sure that your simulation is properly thermalized and record around 50.000 configurations for each temperature. Then, compile a training dataset and a testing dataset similar to the digit datasets out of your configurations.

b) With a training and a testing set at hand, move on to constructing the neural network. You can base your design on the code provided for exercise 12, i.e. one input layer, one hidden layer and one output layer. The input layer size is fixed by the system size of your lattice, the output layer size is 2 because there are 2 phases to distinguish (similar to 10 distinguishable digits earlier).

Train your neural network and adapt the learning rate as well as batch size and epoch number to your needs (What are good values that you find in your testing?). After training, the neural network should be able to accurately distinguish between configurations of your two temperatures and therefore be able to identify if a configuration resembles the ferromagnetic or the paramagnetic phase.

c) The trained neural network can then be used to find the critical temperature of the phase transition between the ferromagnetic and paramagnetic phase by investigating how the network output changes when feeding the network with configuration for intermediate temperatures. For temperatures within the ferromagnetic phase, configurations look (reasonably) similar to the training data of the ferromagnetic phase and the network will likely give a similar output. But what happens when you further increase the temperature and eventually cross into the temperature regime of the paramagnetic phase?

Visualize this crossover of probabilities by performing a Monte Carlo simulation for the Ising model and plotting the expectation values of the network for the two phases for each temperature. Investigate the crossing point and compare to the phase transition temperature.

d) *(Optional Exercise)* Investigate how you can further improve data quality of the previous transition temperature plot. You might want to try and extend some of the following ideas:

- Try to use the Swendsen Wang algorithm to generate configurations for the lower temperature as it generates global updates in contrast to local single-spin flips.
- Try out to provide not only one, but multiple temperature points as training data for the ferromagnetic and paramagnetic phase.
- Play with the values of the learning rate, batch size, and epoch number and investigate how the learning speed changes.
- ...

Exercise 14: Using Tensor Flow (Optional)

In this last (optional) exercise we want to introduce you to the library **Tensor Flow** – currently one of the most developed open-source program packages for machine learning problems. Tensor flow comes with libraries for many programming languages among which there is also a julia package called **TensorFlow.jl**.

Tensor Flow is a library that assists in building neural nets and other structures in which information *flows*, including not only the structure itself but also learning related routines ready to use. Therefore Tensor Flow can take the role of being the building brick from which you construct and train your neural network inside your code, which leaves you with the task of providing training data and surrounding scripts.

In this task, we want you to revisit the phase or digit recognition code that you used in the previous exercises and translate the neural net within the code into TensorFlow.jl. To do so will allow you to access much more diverse net topologies as well as making your code more versatile.

To get started, you can use one of the many tutorials on Tensor Flow already available. Most of the examples are provided in python, which include

- The official tutorial / getting started website of Tensor Flow
- This Tensor Flow tutorial course which introduces Tensor Flow by discussing several example codes in python, specifically this neural network example which is similar to exercise 12

The sources which are strictly written in julia contain

- The official documentation of TensorFlow.jl, especially the tutorial section which can be used as a valuable help to match the more commonly seen python commands to julia language.

After discussing this exercise in the tutorial, we will also upload a sample julia code for digit recognition to the course website for helping you with future uses of Tensor Flow and julia.