Project on Machine Learning

Data Analysis and Machine Learning FYS-STK3155/FYS4155

Department of Physics, University of Oslo, Norway

May 2018

Machine learning (ML) approaches to data from Ising model calculations

Introduction. The aim of this project is to use an already developed Monte Carlo program for the one-dimensional and two-dimensional Ising model, in order to produce the spin configurations for a series of energies E_i (10000 in total) for a system of $L = 40$ spins in one dimension and $L = 40 \times 40$ in two dimensions at three different temperatures. In its simplest form the energy of the Ising model is expressed as, without an externally applied magnetic field,

$$E = -J \sum_{<kl>}^{N} s_k s_l$$

with $s_k = \pm 1$. The quantity N represents the total number of spins and J is a coupling constant expressing the strength of the interaction between neighboring spins. The symbol $< kl >$ indicates that we sum over nearest neighbors only. We will assume that we have a ferromagnetic ordering, viz $J > 0$. We will use periodic boundary conditions and the Metropolis algorithm only.

We will use the Ising model to generate our training data and will focus mainly on supervised training. We will follow closely the recent article of Mehta et al, arXiv 1803.08823. This article stands out as an excellent review of machine learning (ML) algorithms applied to typical physics problems. The added benefit is that each figure and model presented in this article is accompanied by its jupyter notebook. This means that we can start using these and compare with our own results. In case you wish to use their data for the Ising model, thir data can be downloaded from the same link which lists to the jupyter notebooks. See also at the end of the project description for more information on how to install various Python packages.

With the abovementioned configurations we will determine, using first various regression methods, the value of the coupling constant for the energy of the one-dimensional Ising model. Thereafter, we will use the two-dimensional data, but now computed at different temperatures, in order to classify the phase of the Ising model. Below the critical temperature, the system will be in so-called ferromagnetic phase. Close to the critical temperature, the final magnetizations starts becoming close to zero while above the critical temperature, the net magnetization is zero. The latter case, that is the two-dimensional Ising model, will be studied using a **random forest** algorithm and deep neural networks.

You should try to program at least one of these methods yourself (choose the one you prefer). Feel free to use the notebooks to benchmark your code. If you wish to write your C++ or Fortran program for say a simple neural network model, please feel free to do so. You can then benchmark your results against the above jupyter notebooks. More information can also be found at the link for the lecture notes of FYS-STK4155.

We recommend that you form groups of 2-3 students and try to collaborate on the notebooks, develop your own software and discuss the final presentations. You can collaborate on all these topics. The final presentation should include an overview of popular machine learning algorithms as introduction and motivation. Thereafter you discuss the explicit Ising model data and how you have implemented the ML algorithms discussed here, discuss their pros and cons and try to develop your own code for at least one of these algorithms. You are encouraged to use the abovementioned notebooks as starting point and guidance. The duration of your presentation should at most be 30 mins. Allow for approximately 15 mins for discussions and questions.

Part a): Producing the data. You can use the Ising model data from the article of Mehta *et al.*, or generate your own data. If you opt for using your own Ising model code, you need to generate 10000 energy configurations with their spin orientations after the system has reached its most likely state. These energies and their corresponding spin orientations represent then your data. We will use a fixed lattice of $L \times L = 40 \times 40$ spins in two dimensions and $L = 40$ spins in one dimension. Make sure the calculations have been equilibrated. For the two-dimensional system, compute the configurations for three values of the temperature, namely $T = 0.75$ (ordered phase), $T = 2.3$ (near the critical point) and $T = 4.0$ (disordered phase). For the one-dimensional system it suffices to compute the various configurations for one temperature only, say $T = 2.0$. These are the data you will use to study different ML algorithms. We generate our data with $J = 1$.

Part b): Estimating the coupling constant of the one-dimensional Ising model. We start with the one-dimensional Ising model and use the data we have generated with $J = 1$. Use linear regression, Lasso and Ridge regression as described section 6 and in Notebook 4 of Mehta *et al.*. Discuss the methods and how they perform in computing the coupling constant J. Give a critical analysis and discuss how to evaluate the *cost function*. You should feel free to write your own code, see also the lecture notes of FYS-STK4155, in particular te material on least square methods. You can use scikit-learn to perform these analyses. See below for instruction on how to install scikit-learn.

Part c): Determine the phase of the two-dimensional Ising model. We switch now to binary classification methods and use logistic regression to define the phases of the Ising model. Use described section 7 and in Notebook 6 of Mehta *et al.*. Discuss the methods and how they perform. Give a critical analysis and discuss how to evaluate the *cost function*. You should feel free to write your own code. Use thereafter the *random forests* algorithm to classify the same phases as done with logistic regression and discuss the pros and cons of these methods. For *random forests* you can use notebook 9 of Mehta *et al.*

You can use scikit-learn to perform these analyses. See below for instruction on how to install scikit-learn.

Part d): Classifying the Ising model phase using neural networks. We end the classification problem of the phases of the Ising model by emplpying feed-forward deep neural networks (see section 9 of Mehta *et al.*). The method is described in notebook 12.

You can use tensorflow to perform these analyses. See below for instruction on how to install tensorflow.

Part e): Summary. You should make a summary of the various methods and their pros and cons. For the final presentation, you should have at least coded one of these methods yourself and discussed the evaluation of the cost function.

Background literature

On Machine Learning we recommend strongly the article of Mehta *et al.* Textbooks on Machine Learning can be found at the Github address of FYS-STK4155, see in particular Marsland's text.

- Mehta et al, arXiv 1803.08823, *A high-bias, low-variance introduction to Machine Learning for physicists*, ArXiv:1803.08823.

If you wish to read more about the Ising model and statistical physics here are three suggestions.

- M. Plischke and B. Bergersen, *Equilibrium Statistical Physics*, World Scientific, see chapters 5 and 6.
- D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, Cambridge, see chapters 2,3 and 4.
- M. E. J. Newman and T. Barkema, *Monte Carlo Methods in Statistical Physics*, Oxford, see chapters 3 and 4.

Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.
- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Software and needed installations

If you have Python installed (we recommend Python3) and you feel pretty familiar with installing different packages, we recommend that you install the following Python packages via **pip** as

1. pip install numpy scipy matplotlib ipython scikit-learn tensorflow sympy pandas pillow

For Python3, replace **pip** with **pip3**.

See below for a discussion of **tensorflow** and **scikit-learn**.

For OSX users we recommend also, after having installed Xcode, to install **brew**. Brew allows for a seamless installation of additional software via for example

1. brew install python3

For Linux users, with its variety of distributions like for example the widely popular Ubuntu distribution you can use **pip** as well and simply install Python as

1. sudo apt-get install python3 (or python for python2.7)

etc etc.

If you don't want to install various Python packages with their dependencies separately, we recommend two widely used distrubutions which set up all relevant dependencies for Python, namely

1. Anaconda Anaconda is an open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system **conda**
2. Enthought canopy is a Python distribution for scientific and analytic computing distribution and analysis environment, available for free and under a commercial license.

Popular software packages written in Python for ML are

- Scikit-learn,
- Tensorflow,
- PyTorch and
- Keras.

These are all freely available at their respective GitHub sites. They encompass communities of developers in the thousands or more. And the number of code developers and contributors keeps increasing.