Project 2 on Machine Learning, deadline November 5

Data Analysis and Machine Learning FYS-STK3155/FYS4155

Department of Physics, University of Oslo, Norway

Oct 8, 2018

1. add about ising model
2. link to where we have the data
3. explain how to fit the model
4. link to mehta's article

Classification and Regression, from linear and logistic regression to neural networks

The main aim of this project is to study both classification and regression problems, starting with the regression algortihms studied in project 1. We will include logistic regresion for classification problems and write our own multilayer perceptron code for studying both regression and classification problems. The codes developed in project 1, including bootstrap and/or cross-validation as well as the computation of the mean-squared error and the R2 score function can also be utilized (and included in logistic regression and the neural network codes) in the present analysis.

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 on 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, their 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 a so-called ferromagnetic phase. Close to the critical temperature, the final magnetization becomes smaller and smaller in absolute value while above the critical temperature, the net magnetization is zero. This classification case, that is the two-dimensional Ising model, will be studied using logistic regression and deep neural networks. The aim is to develop your own logistic regression code for the classification of the phases (this is a binary model) and your multilayer perceptron code for the classification and regression case.

Feel free to use the notebooks to benchmark your code. If you wish to write your own C++ or Fortran program for say a simple neural network model and a logistic regression model, please feel free to do so. You can then benchmark your results against the above jupyter notebooks.

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.

Part d): Classifying the Ising model phase using neural networks. We end the classification problem of the phases of the Ising model by employing the algorithm for so-called 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.

Background literature

1. The textbook of Trevor Hastie, Robert Tibshirani, Jerome H. Friedman, The Elements of Statistical Learning, Springer, chapters 3 and 7 are the most relevant ones for the analysis here.

2. 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.

1. M. Plischke and B. Bergersen, *Equilibrium Statistical Physics*, World Scientific, see chapters 5 and 6.

2. D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, Cambridge, see chapters 2,3 and 4.

3. 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.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Use Devilry to hand in your projects, log in at `http://devilry.ifi.uio.no` with your normal UiO username and password and choose either 'fysstk3155' or 'fysstk4155'. There you can load up the files within the deadline.
- Upload **only** the report file! For the source code file(s) you have developed please provide us with your link to your github domain. The report file should include all of your discussions and a list of the codes you have developed. Do not include library files which are available at the course homepage, unless you have made specific changes to them.
- In your git repository, please include a folder which contains selected results. These can be in the form of output from your code for a selected set of runs and input parameters.
- In this and all later projects, you should include tests (for example unit tests) of your code(s).

- Comments from us on your projects, approval or not, corrections to be made etc can be found under your Devilry domain and are only visible to you and the teachers of the course.

Finally, we encourage you to collaborate. Optimal working groups consist of 2-3 students. You can then hand in a common report.

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.