Project 3 on Machine Learning, deadline December 15 (midnight), 2021

Data Analysis and Machine Learning FYS-STK3155/FYS4155

Department of Physics, University of Oslo, Norway

Nov 15, 2022

Paths for project 3

Defining the data sets to analyze yourself

For project 3, you can propose own data sets that relate to your research interests or just use existing data sets from say

1. Kaggle
2. The University of California at Irvine (UCI) with its machine learning repository.
3. Or other sources.

As an example on applications of the methods we have discussed in this course, see the article from economy that deals with machine learning on bankruptcy data from Spain at URL":"https://link.springer.com/article/10.1007/s10614-020-10078-2". The data are most likely not accessible within short time, but it is a nice demonstration on how to use the various machine learning methods we have discussed during the semester.

The approach to the analysis of these new data sets should follow to a large extent what you did in projects 1 and 2. That is:

1. Whether you end up with a regression or a classification problem, you should employ at least two of the methods we have discussed among **linear regression (including Ridge and Lasso)**, **Logistic Regression**, **Neural Networks**, **Convolution Neural Networks**, **Recurrent Neural Networks**, **Adversarial Neural Networks**, **Support Vector Machines** and **Decision Trees, Random Forests, Bagging and Boosting**. You could for example explore all of the approaches from decision trees, via bagging and voting classifiers, to random forests, boosting and finally

XGboost. If you wish to venture into **convolutional neural networks** or **recurrent neural networks**, or extensions of neural networkds, feel free to do so. You can also study unsupervised methods, although we in this course have mainly paid attention to supervised learning. We will only touch upon unsupervised methods like **Principal Component Analysis** and **k-means Clustering**.

For Boosting, feel also free to write your own codes.

1. For project 3, you should feel free to use your own codes from projects 1 and 2, eventually write your own for SVMs and/or Decision trees/random forests/bagging/boosting' or use the available functionality of **Scikit-Learn**, **Tensorflow**, etc.
2. The estimates you used and tested in projects 1 and 2 should also be included, that is the $R2$-score, **MSE**, confusion matrix, accuracy score, information gain, ROC and Cumulative gains curves and other, cross-validation and/or bootstrap if these are relevant.
3. Similarly, feel free to explore various activations functions in deep learning and various approachs to stochastic gradient descent approaches.
4. If possible, you should link the data sets with exisiting research and analyses thereof. Scientific articles which have used Machine Learning algorithms to analyze the data are highly welcome. Perhaps you can improve previous analyses and even publish a new article?
5. A critical assessment of the methods with ditto perspectives and recommendations is also something you need to include.

All in all, the report should follow the same pattern as the two previous ones, with abstract, introduction, methods, code, results, conclusions etc..

We propose also an alternative to the above. This is a project on using machine learning methods (neural networks mainly) to the solution of ordinary differential equations and partial differential equations, with a final twist on how to diagonalize a symmetric matrix with neural networks.

This is a field with a large interest recently, spanning from studies of turbulence in fluid mechanics and meteorology to the solution of quantum mechanical systems. As reading background you can use the slides from week 42 and/or the textbook by Yadav et al.

Note: Project 3 has an additional exercise which can give you an additional score of 20 (twenty) points. These are added to the total score from all projects. See below for the additional exercise.

The basic structure of your project

Here follows a set up on how to structure your report and analyze the data you have opted for.

Part a). The first part deals with structuring and reading the data, much along the same lines as done in projects 1 and 2. Explain how the data are produced and place them in a proper context.

Part b). You need to include at least two central algorithms, or as an alternative explore methods from decisions tree to bagging, random forests and boosting. Explain the basics of the methods you have chosen to work with. This would be your theory part.

Part c). Then describe your algorithm and its implementation and tests you have performed.

Part d). Then presents your results and findings, link with existing literature and more.

Part e). Finally, here you should present a critical assessment of the methods you have studied and link your results with the existing literature.

Solving partial differential equations with neural networks

For this variant of project 3, we will assume that you have some background in the solution of partial differential equations using finite difference schemes. We will study the solution of the diffusion equation in one dimension using a standard explicit scheme and neural networks to solve the same equations.

For the explicit scheme, you can study for example chapter 10 of the lecture notes in Computational Physics or alternative sources. For the solution of ordinary and partial differential equations using neural networks, the lectures by Kristine Baluka Hein and included in the lectures of week 42 at this course are highly recommended.

For the machine learning part you can use your own code from project 2 or the functionality of for example **Tensorflow/Keras**..

Part a), setting up the problem. The physical problem can be that of the temperature gradient in a rod of length $L = 1$ at $x = 0$ and $x = 1$. We are looking at a one-dimensional problem

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, t > 0, x \in [0, L]$$

or

$$u_{xx} = u_t,$$

with initial conditions, i.e., the conditions at $t = 0$,

$$u(x, 0) = \sin(\pi x) \qquad 0 < x < L,$$

with $L = 1$ the length of the x-region of interest. The boundary conditions are

$$u(0,t) = 0 \qquad t \geq 0,$$

and

$$u(L,t) = 0 \qquad t \geq 0.$$

The function $u(x,t)$ can be the temperature gradient of a rod. As time increases, the velocity approaches a linear variation with x.

We will limit ourselves to the so-called explicit forward Euler algorithm with discretized versions of time given by a forward formula and a centered difference in space resulting in

$$u_t \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t) - u(x_i,t_j)}{\Delta t}$$

and

$$u_{xx} \approx \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2},$$

or

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j) - 2u(x_i,t_j) + u(x_i-\Delta x,t_j)}{\Delta x^2}.$$

Write down the algorithm and the equations you need to implement. Find also the analytical solution to the problem.

Part b). Implement the explicit scheme algorithm and perform tests of the solution for $\Delta x = 1/10$, $\Delta x = 1/100$ using Δt as dictated by the stability limit of the explicit scheme. The stability criterion for the explicit scheme requires that $\Delta t/\Delta x^2 \leq 1/2$.

Study the solutions at two time points t_1 and t_2 where $u(x,t_1)$ is smooth but still significantly curved and $u(x,t_2)$ is almost linear, close to the stationary state.

Part c) Neural networks. Study now the lecture notes on solving ODEs and PDEs with neural network and use either your own code from project 2 or the functionality of tensorflow/keras to solve the same equation as in part b). Discuss your results and compare them with the standard explicit scheme. Include also the analytical solution and compare with that.

Part d) Solving eigenvalue problems. Follow the discussion in the work of Yi *et al.* in the article from Computers and Mathematics with Applications 47, 1155 (2004), and use your differential equation solver with neural networks, set up a simple square, real and symmetric 6×6 matrix and find the eigenvalues. Compare with the solution from numerical diagonalization with standard eigenvalue solvers from linear algebra.

Part e). Finally, present a critical assessment of the methods you have studied and discuss the potential for the solving differential equations and eigenvalue problems with machine learning methods.

Additonal (optional) exercise, adding 20 more points to final score

This exercise can be done independently of the other tasks. Here you can also choose the data set you want to use. Furthermore, you can use your codes from projects 1 and 2 as well as the codes here or simply use libraries like **Scikit-Learn**, **Tensorflow** or similar.

Your task is to perform an analysis of the bias-variance tradeoff using at least three of the main sets of algorithms we have discussed in this course. We will limit ourselves to a regression problem (fitting). However, feel free to venture into a classification problem.

You are free to choose between bootstrap for resampling (recommended) or cross-validation in order to get the best possible estimates. The methods you could study are Linear Regression (OLS, Ridge and Lasso), deep learning (feed forward neural networks and/or recurrent neural networks), Ensemble methods (decision trees, bagging, random forests and boosting) and support vector machines.

Study the bias-variance tradeoff for at least three of these sets of algorithms for a data set of your choice as function of the complexity of your model. Comment and discuss the results. Discuss the pros and cons of the various methods. Are there some methods which provide both low variance and low bias?

Hint: when you use different methods, pay attention to how you represent (and understand) the complexity of the model. For example, when using decision trees you may represent the complexity of your model by the depth of the tree.

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 Canvas to hand in your projects, log in at https://www.uio.no/english/services/it/education/canvas/ with your normal UiO username and password.

- Upload **only** the report file or the link to your GitHub/GitLab or similar typo of repos! For the source code file(s) you have developed please provide us with your link to your GitHub/GitLab or similar 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 GitHub/GitLab or similar 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.

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.