CSC 448/692 Programming Project – Due 2:50 p.m., 2 December

A good way to show not only your programming skills, but also your understanding and interest in a particular topic is to create a Program Portfolio. We will not a have final in this class, but we will have a project instead, and your project will be to create a Machine Learning Portfolio. A portfolio is not just a collection of work samples and sample code. It is an opportunity for you to showcase:

- Who you are
- Your work
- Your understanding of the topic
- Where you like to go next
- And maybe even what YOU really like to work with

A significant part of your portfolio will of course be your code. You will have to at a minimum show code for all algorithms that we will cover. Some things will be mandatory parts, others will be up to you to decide on what to implement or not.

I will not force you to write in a particular language, nor to write all your code in the same language. It does make sense to actually stay in a single language as it will significantly ease your presentation, and you have already begun your work in Python. You do not need to use the same language for all your code, you may switch language for different parts of the book, and even for different subsections.
The force assumption though is that the reader is not used to compile or run code, so you will need to carefully document not only all necessary steps to run the code (compile, inputs etc), but also how use your code (input limitations, assumptions etc).

Examples is not only a great way to test your code, but also an excellent way to describe your code for the user. Make use of and provide multiple examples for each code module you write.

As the purpose is to show your skill, your code of course needs to be well documented and working correctly. By well documented I mean that you should be able to go back in a year or two, read your documentation, look at the code and be able to understand what the code is doing, how it can be modified or even corrected.

You will also need to display that you understand the topic. This means that in your documentation you need to provide enough theoretical background, such that not only does the reader understands the algorithm, and how it is implemented, but there should be enough information in the text you have written that you would not need to find the (or a) book. This is the tough part, as you need to write enough to understand the concept, using your own words so you do not risk any copy infringement.

How can you do this? One way is to create a webpage, present your writing on the page and allow the user to run the code on the page, or download it and run on a local machine. Another way is to create a PDF document with all your write-ups and make the code downloadable archive. The way you do this is up to you, but it is important that your intended target finds your portfolio accessible and readable. Remember that a picture is 1000 words, but it is not enough by itself, make sure to enhance it with your own words.

Classical Machine Learning:

Mandatory modules:

- **Perceptron:** Create a perceptron class for binary labeling. It should be generic enough to take training set of any size. At least four functions, `__init__()`, `fit()`, `net_input()`, `predict()`.
- **Linear Regression:** Create a linear regression class for *d=1* dimension data. It should take training set where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}$ and calculate **w**.
  Graduate Students: Create a linear regression class for *d-dimension* data. It should take training set where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^d$ and calculate **w**.
- **Other regression models:** Pick one of:
  - **Linear Regression for Polynomial Regression Tasks**
  - **Logistic Regression**
- **Hypothesis classes of low VC-dimension:** Implement two of:
  - **Threshold Functions**
  - **Intervals**
  - **Axis Aligned Rectangles**
  - **Decision Stumps**
- **Support Vector Machine**
  - Hard SVM
- **K- Nearest Neighbor**
  - 1-NN

Visualization:

Mandatory modules:

- **Scatter plots:** Use `mathplotlib`, to at least display a scatter plot of the data in two dimensions.
- **Decision boundary:** Use `mathplotlib` and `numpy`, to draw a contour plot of the decision boundary between two classes.
- **Regression line:** Use `mathplotlib`, to at least visualize the data and the best-fit regression line.

Other things we have talked about

- **ADABoost**
- **SGD**
- **Soft SVM**
- **k-NN**

These and the things you didn't pick from the mandatory list are things you could use to personalize your project with. It is perfectly fine for example to substitute Soft SVM instead of Hard SVM or k-NN instead of 1-NN.

Assignment Submission:

Your project is due and should be accessible no later than 2:50 p.m. on the day of the finals.

Grading:

Your project will be graded on three things:

- Your code, functionality correctness 40%

• Your narrative, algorithm description etc	40%
• Your code documentation	20%