CS-GY 6923 Mini-Project

Basic Info

Students have the option to replace the final homework assignment of the course with a class project. The project grade will be averaged into to the student's written homework or lab grade -- which ever one gives the highest final score.

- Work in groups of **one, two, or three**. I find groups with more than one student tend to work best.
- **Deliverables**: Either a 3-4 page written report summarizing the work (with code submitted as an appendix) or a Python Workbook with all code and documentation. If you present your results as a workbook, it should be commented and structured similar to the "demos" we post for the class.
- **Deadline**: May 9th (last day of Spring classes).

Types of Projects

You can choose between two different types of projects.

1. Concept Based Project (aka create a new demo for the class!)

For this sort of project, the goal is to take an empirical deep-dive into a specific concept learned into the class, or a related topic we did not cover in depth. The idea would be to produce a demo that serves more or less the same purpose as those we use in the class -- to teach that concept in a hands on way.

Example topics:

- Explore the connection between l1 regularization and feature selection. Can you find natural datasets where l1 regularization does or does not encourage sparsity? How do the features selected when usign l1 regularization compared to e.g. those selected with stepwise regression?
- Empirically evaluate the PAC-learning bounds discussed in Lecture 7. For simple finite hypothesis classes, do they accurately predict generalization performance? You can check

by randomly generating train/test data.
- Explore the "double decent" phenomena for simple models like polynomial regression. Can you find simple data sets where the phenomena appears? On a related topic, can you construct simple neural nets + data sets where you see or don't see the phenomena?
- We saw multiple algorithms for piecewise regression: either use explicit feature transformation and brute-force search for the beginning/end of each piece, or use a neural network with Relu activations trained with gradient descent. Which gives better performance? In terms of accuracy or training time.
- Design a lab to explore the differences between batch, stochastic, and full gradient descent. Can you see the trade-off between convergence rate and the cost of each iteration.
- Find a simple problem where feature learning can be used for transfer learning.
- If you have an idea for the project that you are unsure about, please post on Ed or set up a time to chat with me or a TA.

2. Dataset Based Project

For this sort of project, the goal is to explore a data set you are interested in using tools learned in the class. In short: a) Find or collect a data set. b) Ask a question (or two) about the data set which can possibly be answered with machine learning. c) Apply tools and techniques learned in the class to answering that question. d) In your report, discuss what worked, why you think that's the case, and what might be tried next.

Any data set is allowed. You can use publically available data or data you collect yourself (e.g. via web-scraping). However, you should not reproduce an analysis that has already been done! Even if the question you ask has been asked before, you should take a new approach to solving it.

There are two options for the project

1. Find or collect a data set.
2. Ask a question (or two) about the data set which can possibly be answered with machine learning.
3. Apply tools and techniques learned in the class to answering that question.

Tips for this sort of project:

- There are *tons* of free data sets available online. This KDnuggets page is a wonderful starting point with lots of links and inspiration.
- In you are interested in *social network data* take a look at the Stanford SNAP project.
- Lots of interesting data can be scraped from the internet -- e.g. financial data, text data,

images, etc. If you go in the direction of collecting your own data, *make sure you have enough time to invest*. We are grading you on what you do with the data, not how you collect it.

- Choose a data set and that interests you. It will be more fun that way. Center your project around a question or two that you think would be interesting to answer using data. Talk to me or a TA if your team is struggling!
- **Look at the data.** Look at the data in table form, plot different features against each other, run PCA and plot, etc. Spend time getting to know your data set before running it through any ML algorithms. You will notice lots of things: missing data, outliers, features which are clearly useless, errors in how the data was imported/processed, etc. Re-examine data after doing any preprocessing, feature extraction, cleaning, etc.
- **Start simple.** Try the simplest methods we know before moving onto more complex approaches. Simple linear regression, logistic regression, naive bayes, or k-nearest neighbors for supervised problems, PCA for dimensionality reduction, etc. These methods will give you baseline results to improve on.
- **Start small.** Large datasets are slow to process! When writing and debugging code, test on a small subset of your data. This goes for everything from preprocessing, to initial model implementation. Don't run on the full data set until you are sure your code is working and bug free.
- **Sanity Check.** Is the error you achieved for a regression problem better than what would have been acheived by predicting every value to be the mean of the training data? Is your classification error better than what you would get from predicting 0 every time?