| **Machine Learning** | **F2020** |
|---|---|
| Problem Set 2 | |
| *Handed Out: Sep* $28^{th}$, 2020 | *Due: Oct* $9^{th}$, 2020 |

In this assignment you will use gradient descent to perform polynomial regression on three synthetic datasets. In the "data" folder, you will synthetic-1.csv, synthetic-2.csv, and synthetic-3.csv. These files contain two columns of real-valued data that you will use to perform your regression analysis. In this case, your input (X) values are in column 1, and your output (Y) values are in column 2.

You will submit a writeup in Word or PDF that summarizes your results and all code as a zip file. Submit the writeup (with attached source code) to the Canvas submission locker before 11:59pm on the due date.

# Polynomial Regression with Gradient Descent (50 points)

Write a method that uses gradient descent to perform polynomial regression on each of the three synthetic datasets provided for you. For this part of the assignment, we'll look at how well we can predict each dataset by explicitly testing a 1st order polynomial, a 2nd order polynomial, a 4th order polynomial, and for an extreme comparison a 7th order polynomial. Recall that for polynomial regression, we are using the following hypothesis:

$$h_\theta(x) = \theta_0 + \sum_{i=1}^{n} \theta_i x^i \tag{1}$$

In the above equation, n is the *order* of the polynomial. Thus, the hypothesis for a 4th order polynomial would be:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 \tag{2}$$

For each fitted polynomial model, report the mean squared error that you obtained on each dataset as well as the weight values produced by your code.

Details:

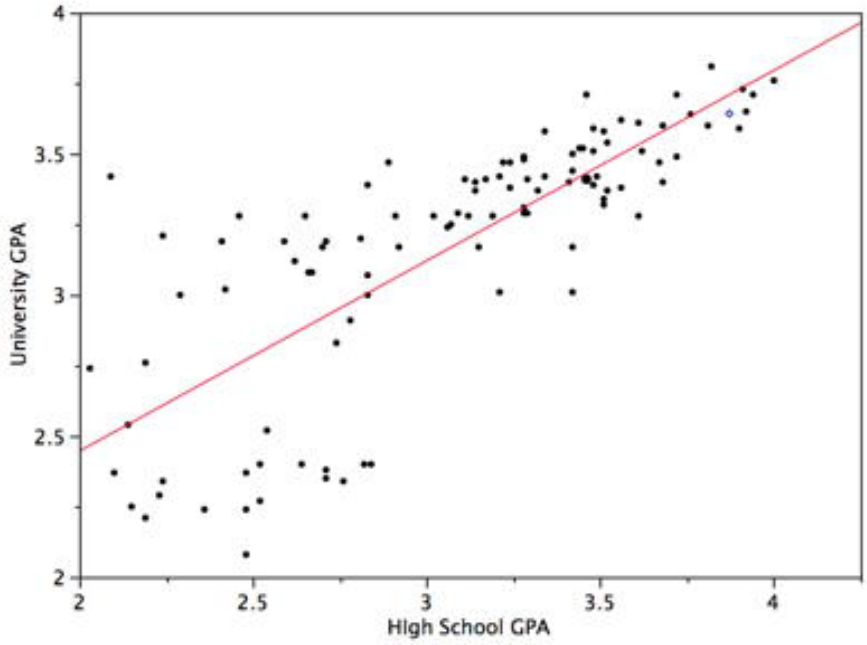
- You may not use a high-level function for implementing polynomial regression or gradient descent. Anything else should be fine, however. If you are ever in doubt, just ask me and I'll let you know if it's okay to use it or not.
- All code should be written in Python 3.
- For this assignment, we are using mean squared error as a loss function. You may not use a high level function to automatically calculate mean squared error; however, if you happen to have some code lying around from a previous assignment that calculates it, I would not be opposed to you making use of it.

- It is up to you to choose an alpha value that works for you and to decide when you want to update the weights. You can choose to do full batch gradient descent, or stochastic gradient descent, or anything in between. Just outline what you ended up using in your report.
- Explain any implementation choices you had to make in the final report.
- Include the mean squared error obtained by each model on each dataset in the report. Also include the weight values associated with each model in the report.

# Plot your regression lines (30 points)

Write a function that creates a visualization of your regression lines for each model on each dataset. This figure should include the datapoints used for regression as well as the regression line itself.

Here is an example of how this might look:




Details:

- You can choose to plot the lines for each of your models on the same figure per dataset if you'd like. Just be sure to differentiate the lines (and include a legend) so that it is posisble to determine which line corresponds to which model.
- As before, you can use whatever means you see fit to make the visualizations. Matplotlib is nice, but you could use any other tool that you feel comfortable with. All I'm concerned with is the visualizations themselves.

Your writeup must include visualizations for each of your regression models on each of the synthetic datasets (meaning at most, 12 plots). As with the last time we did plotting, make sure you title each figure with the dataset name.

# Presentation (20 points)

Your report must be complete and clear. A few key points to remember:

- Complete: the report does not need to be long, but should include everything that was requested.
- Clear: your grammar should be correct, your graphics should be clearly labeled and easy to read.
- Concise: I sometimes print out reports to ease grading, don't make figures larger than they need to be. Graphics and text should be large enough to get the point across, but not much larger.
- Credit (partial): if you are not able to get something working, or unable to generate a particular figure, explain why in your report. If you don't explain, I can't give partial credit.

# Bonus: Polynomial Regression with Regularization (10 Points, required for graduate students)

While a 7th order polynomial has the potential to accurately model complex patterns, it is also incredibly likely to overfit our data. I mean, we are only looking at 100 points after all. Let's see if we can do something about that. Extend your gradient descent algorithm to make use of L2 norm regularization. Then, use this method to perform polynomial regression with a 7th order polynomial on each synthetic dataset. Calculate the mean squared error for these models on each dataset and plot your resulting model (as you did in Part 2). Compare these with the unregularized 7th order polynomial model created in Part 1. Did regularization make a difference? Why do you think that it did or didn't?

Details:

- As always, don't use any high level functions to add L2 regularization to your code.
- As in Part 2, feel free to construct the plots using whatever means you are most comfortable with.
- Explain any implementation choices you had to make in the final report.
- Include BOTH the mean squared error and regression line plot for each synthetic dataset in the final report.