CS534 — Implementation Assignment 1 — Due 11:59PM Oct 15th, 2021

General instructions.

1. Please use Python 3 (preferably version 3.6+). You may use packages: Numpy, Pandas, and matplotlib, along with any from the standard library (such as 'math', 'os', or 'random' - for example).

2. You should complete this assignment alone. Please do not share code with other students, or copy program files/structure from any outside sources like Github. Your work should be your own.

3. Submit your report on Canvas and your code on TEACH following this link:
`https://teach.engr.oregonstate.edu/teach.php?type=assignment`.

4. Please follow the submission instructions for file organization (located at the end of the assignment description).

5. Please run your code before submission on the EECS servers (i.e. babylon). You can make your own virtual environment with the packages we've listed in either your user directory or on the scratch directory.

6. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. The report should be clear and concise, with figures and tables clearly labeled with necessary legend and captions. The quality of the report and is worth 10 pts. The report should be a PDF document.

7. In your report, the **results should always be accompanied by discussions** of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

Linear regression
(total points: 90 pts + 10 report pts)

For the first part of the assignment, you will implement linear regression, which learns from a set of N training examples $\{\mathbf{x}_i, y_i\}_{i=1}^N$ an weight vector $\mathbf{w}$ that optimize the following Mean Squared Error (MSE) objective:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 \tag{1}$$

To optimize this objective, you will implement the gradient descent algorithm. Because some features have very large values, for part of the assignment you are asked to normalize the features. This will have an impact on the convergence behavior of gradient descent.

Data. This is the same dataset as used in IA0 (except that the ID feature is removed), which contains historic data on houses sold between May 2014 to May 2015. You need to build a linear regression that can be used to predict the house's price based on a set of features. You are provided with two data files: **train** and **dev** (validation), in csv format. You are provided with a description of the features as well. The last column stores the target y values for each example. You need to learn from the training data and tune with the provided validation data to chose the best model.

Part 0 (5 pts) : Data preprocessing. Excluding the ID, this data contains 19 features that are mixed in types. Follow IA0, you will fist need to split the `date` feature into three separate features `year, month, day`. This gives us a total 21 usable features that are encoded in the numerical form. For this part, we will simply treat these features as numerical. Note that there may be better ways of encoding some of them and you can explore them in PART 4. To prepare your data, perform the following pre-processing steps:

(a) Add to the data a dummy feature with constant value of 1 for all examples. This will allow us to learn the intercept or bias term w_0 for the linear model $y = w_0 + w_1x_1 + ... + w_dx_d$.

(b) Feature `yr_renovated` has value 0 if the house has not been renovated. This creates an inconsistent meaning to the numerical values. Construct a new feature `age_since_renovated` to replace `yr_renovated` as follows. If `yr_renovated` is zero,

$$age\_since\_renovated = year - yr\_built$$

Otherwise

$$age\_since\_renovated = year - yr\_renovated$$

(c) Normalize all features (excluding `waterfront`, which is binary and we will leave it as is) using z-score normalization based on the training data. Do not normalize `price` as it is the target `y`.
To normalize a feature x using z-score normalization, the formula is $z = \frac{x-\mu}{\sigma}$, where μ and σ are the mean and standard deviation of x respectively. The normalized feature z will have zero mean and unit standard deviation. Note that when applying the model learned from normalized data to test/validation data, you should make sure that you normalize the inputs exactly the same way. That is, you must save the training μ and σ for each feature to be used for normalization of the test data.

Part 1 (40 pts). Implement batch gradient descent and explore different learning rates. For this part, you will work with the pre-processed and normalized data, and consider at least the following values for the learning rate: $10, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$.

(a) Train your model for up to 5000 iterations using different learning rates. If it has converged (training MSE no longer changes or changes very little) or diverged (training MSE increase with more iterations) before reaching 5000 iterations, feel free to stop early. Plot the MSE as a function of iterations for each learning rate. You can generate a single plot for each learning rate or put all curves in a single plot, using different colors with legends to indicate which color for what learning rate.

> Question: Which learning rate or learning rates did you observe to be good for this particular dataset? What learning rates (if any) make gradient descent diverge?

(b) For each learning rate that has converged, compute and report the MSE of the validation data.

> Question: Which learning rate leads to the best validation MSE? Between different convergent learning rates, how should we choose one if the validation MSE is nearly identical?

(c) Pick the best converged solution that minimizes the validation MSE, and report the learned weights for each feature.

> Question: learned feature weights are often used to understand the importance of the features. What features are the most important in deciding the house prices according to the learned weights?

Part 2 (20 pts). Training with non-normalized data Use the preprocessed data but skip the normalization. Consider at least the following values for learning rate: $10, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$.

(a) Train your model for up to 5000 iterations using different learning rates. If it has converged or diverged before reaching 5000 iterations, feel free to stop early. Plot the MSE as a function of iterations for each learning rate. You can generate a single plot for each learning rate or put all curves in a single plot (using different colors with legends to indicate which is which).

> Question: What learning rates work for the un-normalized data? Compare between using the normalized and the non-normalized versions of the data. Which one is easier to train and why?

(b) For each learning rate that has converged, compute and report the MSE of the validation data. Pick the best converged solution that minimizes the validation MSE and report the learned weights.

> Question: Compare the weights to those of 1(c), what difference do you observe? What is the explanation for such difference? How does this impact the interpretation of the weights as the measure of feature importance.

Part 3 (10 pts) Redundancy in features. We have seen that `sqrt_living` and `sqrt_living15` are fairly correlated. When features are correlated, they can offer redundant information. For this part, please use the normalized data but drop the feature `sqrt_living15` and train the model with a learning rate of your choice and report the validation MSE and the learned feature weights.

> Question: how does this new model compare to the one in part 1 (c)? What do you observe when comparing the weight for `sqrt_living` in both versions? Consider the situation in general when two features x_1 and x_2 are redundant, what do you expect to happen to the weights (w_1 and w_2) when learning with both features, in comparison with w_1 which is learned with just x_1? Why?

Part 4 (15 pts). Explore feature-engineering and Participate in-class Kaggle competition. We are hosting a in-class competition using this data to explore different feature engineering ideas. Follow this link to access the competition.
`https://www.kaggle.com/t/0c5599f28f344e8eac04d4318f2d48d3`

To get full marks for this part, you will need to come up/experiment with **at least three different ways of modifying the feature set** (similar to 0(c) or part 3) to improve the performance. Your grade will depend on the sensibility and creativity of your explored modifications. As a negative example, simply dropping a few arbitrary features is not considered a sensible modification.

To make a valid submission, use the provided labeled data (you can use both train and dev) to build your model and then apply it to the test instances in the test data available from Kaggle's Data tab. Format your output as a two-column CSV with ID followed by the predicted price.

> For this portion of your report, you must explain and motivate (this is important!) the ideas that you have tried for this part and their corresponding performances in testing MSE reported on Kaggle (please provide your team name so that it can be cross-verified). Discuss the successes and failures of your attempts and the lessons you learned from the exploration.

Competition bonus: The top three performers will receive 2 bonus points.

Submission. Your submission should include the following:
1) Submit your source code with a readme file on TEACH.
2) Your report, which can follow the same structure as this file (see general instruction 6 and 7), divided into parts, with discussions to answer all the questions, and necessary figures/tables to show results;
3) Your report should be in PDF format and should be submitted separately to Canvas.