CS419M (Spring 2020): Assignment 1 [Last Update: 8:20 pm, Feb 6th]

This assignment is due by **11.55 pm on Feb 8, 2020**. Late submissions will be allowed until 11.55 pm on Feb 9, 2020, with a 10% reduction in the overall marks.

Please read the following important instructions before getting started on the assignment.

1. This assignment should be completed individually.
2. This assignment is entirely programming-based. The task is hosted on Kaggle. Click here for further instructions on how to access Kaggle.
3. Your final submission should be a .tgz bundle of a directory organized exactly as described here.

Many faces of linear regression

Predicting fares of taxi rides can be useful for ridesharing companies like Uber and Ola. Given just the pickup and dropoff locations, one could come up with an estimate of the fare. Can one do better with additional features? In this assignment, you will be given a set of datapoints with different features characterizing taxi rides in New York City. You will need to find the best linear regression model that most accurately predicts taxi fares for a set of test rides.

This task is hosted on Kaggle. The data sets `train.csv`, `dev.csv` and `test.csv` are available for download on the Kaggle page. Here is a template script, lr.py. Within `lr.py`, you will find a function `closed_soln(phi, y)` that computes the closed-form solution for an unregularized least squares linear regression model. You will also see placeholders for functions with self-explanatory names (e.g. `compute_RMSE(phi, w, y)`, `generate_output(phi_test, w)`) that need to be filled in. The function `main()` has been expanded to show the sequence of steps we will run to evaluate tasks 1 to 3. Please use **Python 3.5** to run your script and minimize version discrepancies during grading.

Please note that for all subsequent questions that ask for numbers, you will lose points if we cannot reproduce the exact number ($\pm\epsilon$) you report when we run your code.

1. Implement the gradient descent algorithm to estimate the model parameters for an unregularized least squares linear regression model. Add your implementation to the `gradient_descent(phi, y)` function. Compute the weights using gradient descent and find predicted taxi fares for the development set samples in `dev.csv`. Report the root mean squared error you obtain on these samples in **submission/report.pdf**. Set the learning rate in `gradient_descent(phi, y)` to the value we should use during grading. (Your weight vector should start from an all-zeros vector or a random initialization.) **Tip**: You will find it useful to apply some form of feature normalization on your inputs before optimizing your loss function. **[10 points]**

   Report the following details in **submission/report.pdf**:
   - What is the absolute difference between the following two calls, `compute_RMSE(phi, w1, y)` and `compute_RMSE(phi, w2, y)` on `dev.csv`, where `w1` and `w2` were obtained using the closed-form solution and gradient descent, respectively? **[2 points]**
   - What stopping criterion did you use to determine when gradient descent converged? **[1 point]**
   - Instead of batch gradient descent, implement stochastic gradient descent in `sgd(phi, y)`. Let `w1` and `w2` be the respective weight vectors. Report the absolute difference between the two calls

compute_RMSE(phi_train, w1, y) and compute_RMSE(phi_train, w2, y) on dev.csv. (Tune and set the learning rate such that this difference is minimized.) **[5 points]**

2. Within pnorm(phi, y, p), optimize a p-norm regularized least squares objective function (where p is 2 or 4) using either gradient descent or stochastic gradient descent. Your regularization term now becomes $\lambda(||\mathbf{w}||_p)^p$ where p-norm of a d-dimensional $\mathbf{w}$ is defined as $||\mathbf{w}||_p = \left(\sum_{i=1}^{d} |w_i|^p\right)^{\frac{1}{p}}$ and λ is a non-negative value. This term is added to the standard least squares objective to form the p-norm regularized objective. Report root mean squared error on all the development set samples when $p = 2$ and $p = 4$ in **submission/report.pdf**. Set λ in pnorm(phi, y, p) to the value we should use during grading. **[8 points]**

3. Implement two different basis functions that will be applied to your input features with the L2-regularized model and optimized using gradient descent. Add your implementations of the basis functions to get_features_basis1(file_path) and get_features_basis2(file_path). Describe your choice of basis functions in **submission/report.pdf**. Also report the root mean squared error on the development set samples using both basis functions. **[4 points]**

4. Create subsets of the training data containing the first 10000, 15000, 20000, 30000 instances in train.csv. Create a plot where the X-axis is the size of the training set (10000, 15000, 20000, 30000, full) and the Y-axis is the root mean-square error on the dev.csv instances. You can create a new function within lr.py and use existing libraries (like matplotlib) to create this plot. Add it to **submission/report.pdf**. **[3 points]**

5. Of the six features, which are the two least useful ones? Describe how you identified these two features in **submission/report.pdf**. **[2 points]**

6. The objective for this last part is to build the best possible linear regression model using any enhancements you like. Submit the output from generate_output(phi_test, w) on test.csv to Kaggle so that your roll number appears on both the "Public Leaderboard" and "Private Leaderboard". (In this output file, do make sure that you maintain the same order of the samples that appear in test.csv. Check the Kaggle page for a sample submission file.) Top-scoring performers on the "Private Leaderboard" (with a suitable threshold determined after the submission date) will be awarded extra credit points. Describe your innovations in **submission/report.pdf**. Also, submit your code for this part within lr.py. (You can define any new functions for this last part and run this within main().) **[5 points]**

Frequently Answered Questions (FAQs)

1. **How do we use the time and date stamp feature?** Ans: Use standard libraries like pandas to parse the "datetime" type and use the resulting value.
2. **Is it ok to increase the number of arguments of a function (else can we use global variables)?** Ans: It's perfectly fine to use global variables. (Please do not increase the number of required arguments in the predefined functions; you can add optional arguments to the functions.) c)
3. **Can we define our own extra functions?** Ans: Yes, that's allowed as long as calls to these new functions are made from within the functions already defined in our template file.
4. **My laptop hangs when I evaluate on all 20K instances in the dev set. How should I handle this?** Ans: Do your evaluation in batches and accumulate all the predictions before submitting on Kaggle. The other option is to move from your laptop to an environment like Colab.