HW Assignment 2 (Due by 5:30pm on Mar 10)

1 Implementation (90 points)

In this exercise, you are asked to implement de gradient descent algorithm and use it to train linear regression models for the Athens houses dataset and an artificial dataset with and without L2 regularization. The input data is available at `https://webpages.uncc.edu/rbunescu/courses/itcs4156/hw02.zip`. Make sure that you organize your code in folders as shown in the table below. Write code only in the Python files indicated in bold.

itcs4156/
  hw02/
    report.pdf
    `code/`
      simple.py
      multiple.py
      polyfit.py
      train_test_line.png
    `data/`
      `simple/`
        train.txt, test.txt
      `multiple/`
        train.txt, test.txt
      `polyfit/`
        train.txt, test.txt, devel.txt

All the required results and plots should be included in an appropriately edited homework report, using proper indentation, section titles, and formatting. If you include formulas, make sure that you use appropriate formatting. The PDF of the report `report.pdf` should be submitted on Canvas, together with the code. While it is important that your code runs correctly, unless we need to verify your results, we will not run your code. The assignment will be graded based on the report.

Pointers to relevant content on the lecture slides:

1. The formulas for standard scaling are shown on slide 29. Do not scale the bias feature!
2. The pseudocode for the gradient descent algorithm is shown on slide 5.
3. The gradient descent update rule for linear regression is shown on slide 24, with the corresponding NumPy code for the gradient computation on slide 33.
4. For ridge regression (linear regression with L2 regularization), the gradient descent update rule is shown on slide 34, whereas the corresponding NumPy code for the gradient computation is shown on slide 35.

Note: For the Athens houses exercises, in order to avoid numerical issues, divide all the house prices by 100.

1. [**Feature Scaling**, 10 points]
   To improve the convergence of gradient descent, it is important that the features are scaled so that they have similar ranges. Implement the scaling to standard normal distribution, using the skeleton code in `scaling.py`.

2. [**Simple Regression**, 20 points]
   Train a simple linear regression model to predict house prices as a function of their floor size, by running gradient descent for 500 epochs with a learning rate of 0.1. Use the dataset from the folder `hw02/data/simple`. Plot $J(\mathbf{w})$ vs. the number of epochs in increments of 10 (i.e. after 0 epochs, 10 epochs, 20 epochs, ...). After training print the parameters and RMSEs and compare with the solution from normal equations. Plot the training using the default blue circles and test examples using lime green triangles. On the same graph also plot the linear approximation.

3. [**Multiple Regression**, 20 points]
   Train a multiple linear regression model to predict house prices as a function of their floor size, number of bedrooms, and year. Run gradient descent for 200 epochs with a learning rate of 0.1. Use the the dataset from the folder `hw02/data/multiple`. Plot $J(\mathbf{w})$ vs. the number of epochs in increments of 10 (i.e. after 0 epochs, 10 epochs, 20 epochs, ...). After training print the parameters and RMSEs and compare with solution from normal equations.

4. [**Polynomial Curve Fitting**, 40 points]
   In this exercise, you are asked to use gradient descent to train a linear regression model, with and without regularization, on the artificial dataset from the folder `hw02/data/polyfit`.

   (a) Select 30 values for $x \in [0, 1]$ uniformly spaced, and generate corresponding t values according to $t(x) = sin(2\pi x) + x(x + 1)/4 + \epsilon$, where $\epsilon = N(0, 0.005)$ is a zero mean Gaussian with variance 0.005. Save and plot all the values. Done in `dataset.txt`.

   (b) Split the 30 samples (x_n, t_n) in three sets: 10 samples for training, 10 samples for validation, and 10 samples for testing. Save and plot the 3 datasets separately. Done in `train.txt, test.txt, devel.txt`.

   (c) Consider a linear regression model with polynomial basis functions, trained with the objective shown below:

$$J(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} \left(h(x_n, \mathbf{w}) - t_n\right)^2 + \frac{\lambda}{2} ||\mathbf{w}||^2$$

   Show the gradient update used for minimizing $J(\mathbf{w})$.

   (d) Train and evaluate the linear regression model in the following scenarios:

      1. Without regularization: Using $M = 5$, run gradient descent with a learning rate that is tuned on the training data using the following consecutive powers of 10: {0.0001, 0.001, 0.01, 0.1, 1, 10}. Plot $J(\mathbf{w})$ vs. epochs and select the largest learning rate that leads to a smooth, decreasing behavior of $J(\mathbf{w})$. If convergence appears to be too slow, you may consider consecutive powers of 2 in the same range for the learning rate. Once the learning rate is selected,

run gradient descent until the cost value appears to have converged (flatten) or performance does not improve on validation data. Depending on the learning rate, you may have to run it for tens or even hundreds of thousands of epochs to converge. You can tune the number of epochs by monitoring the performance on the validation data. Compare the trained parameters and RMSE performance with the solution obtained in the first homework for the same degree M.
Hint: With a learning rate of 0.4 and 5,000,000 epochs, my gradient descent solution matches the RMSE of the solution using the normal equations. If standardization is done before expansion to the design matrix, my gradient descent converges to the same solution with a learning rate of 0.08 after only 5,000 epochs.

2. With regularization: Fixing $M = 9$ and $ln(\lambda) = -10$, repeat the experiments above, this time with regularization. Compare the trained parameters and RMSE performance with the solution obtained in the first homework for the same degree M and λ.
Hint: With a learning rate of 0.2 and 10,000 epochs, my gradient descent solution matches the RMSE of the solution using the normal equations.

- [**Stochastic Gradient Descent (*)**, 20+ points]
Implement SGD and run it for problems 1, 2, and 3 above, using the same hyperparameters (learning rate, number of epochs, M, and λ). Compare the SGD solution to the batch GD solution. Does SGD need fewer or more epochs to arrive at the same parameters as batch GD? If fewer, how many epochs are sufficient? You may also consider experimenting with minibatch SGD.

2 Submission

Electronically submit on Canvas a hw02.zip file that contains the hw02 folder in which your code is in the 3 required files, as well as the `report.pdf`.

On a Linux system, creating the archive can be done using the command:
`> zip -r hw02.zip hw02`.

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.
2. Use adequate comments, both block and in-line to document your code.
3. On the theory assignment, **clear and complete explanations and proofs of your results are as important as getting the right answer**.
4. Make sure your code runs correctly when used in the directory structure shown above. We will not debug your code.