HW Assignment 1: Implementation (Due by 5:30pm on Feb 24)

1 Implementation (100 points)

In this exercise, you are asked to run an experimental evaluation of linear regression on the Athens houses dataset, and on an artificial dataset with and without L2 regularization. The input data is available at `https://webpages.uncc.edu/rbunescu/courses/itcs4156/hw01.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/
    hw01/
      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.

Note: For exercises 1 and 2, in order to avoid numerical issues, **divide all the house prices by 100**.

1. [**Simple Regression**, 20 points]
   Train a simple linear regression model to predict house prices as a function of their floor size, based on the solution to the system with 2 linear equations discussed in class (slide 14). Use the dataset from the folder `hw01/data/simple`. Python3 skeleton code is provided in `simple.py`. After training print AND REPORT the parameters AS WELL AS the RMSE and the objective function values on the training and test data. Plot the training using the default blue circles and test examples using lime green triangles. On the same graph also plot the linear approximation.

   Hint: You can derive the solution of the system of linear manually, or you can use the `numpy.linalg.solve()` *function as shown in the class examples.*

2. [**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. Use the normal equations discussed in class (slide 44), and evaluate on the dataset from the folder `hw01/data/multiple`. Python3 skeleton code is provided in `multiple.py`. After training report the parameters and as well as the RMSE and the objective function values on the training and test data. Compare the test RMSE with the one from the simple case above. Do not forget to use a bias parameter w_0 and the corresponding constant feature set to 1 in each example.

3. [**Polynomial Curve Fitting**, 40 points]
In this exercise, you are asked to run an experimental evaluation of a linear regression model, with and without regularization. Use the normal equations discussed in class, and evaluate on the dataset from the folder `hw01/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. Generation and saving was done for you 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. Generation and saving was done for you 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 closed form solution (vectorized) for the weights $\mathbf{w}$ that minimize $J(\mathbf{w})$.

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

      1. Without regularization: Use the training data to infer the parameters $\mathbf{w}$ for all values of $M \in [0, 9]$. For each order M, compute the RMSE separately for the training and test data, and plot all the values on the same graph, as shown in class (slide 36). *Hint: To find the parameters, you can solve the system of linear equations for polynomial curve fitting (slide 22), or the vectorized normal equations (slide 44).*
      2. With regularization: Fixing $M = 9$, use the training data to infer the parameters $\mathbf{w}$, one parameter vector for each value of $\ln \lambda \in [-50, 0]$ in steps of 5. For each parameter vector (lambda value), compute the RMSE separately for the training and validation data, and plot all the values on the same graph, as shown in class. Select the regularization parameter that leads to the parameter vector that obtains the lowest RMSE on the validation data, and use it to evaluate the model on the test data. Report and compare the test RMSE with the one obtained without regularization. *Hint: For each value of $\lambda \in \{e^{-50}, e^{-45}, ..., e^{-5}, e^{0}\}$, finding the parameters $\mathbf{w}$ is done by using the vectorized normal equations for ridge regression (slide 45).*

   (e) [20 Bonus points] Train and evaluate the linear regression model above using `sklearn`. For ridge regression, add the validation data to the training data and

use the `RidgeCV` function to tune the hyper-parameter λ (use 10 folds). Compare with the results from (d) above.

2 Submission

Electronically submit on Canvas a hw01.zip file that contains the hw01 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 hw01.zip hw01`.

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.