CS 534: Homework #1

Submission Instructions: The homework is due on Sept 19th at 11:59 PM ET on Gradescope. A part of your homework will be automatically graded by a Python autograder. The autograder will support Python 3.10. Additional packages and their versions can be found in the `requirements.txt`. Please be aware that the use of other packages and/or versions outside of those in the file may cause your homework to fail some test cases due to incompatible method calls or the inability to import the module. We have split homework 1 into 2 parts on Gradescope, the autograded portion and the written answer portion. If *either of the two parts is late, then your homework is late.*

1. **Upload PDF to HW1-Written Assignment**: Create a single high-quality PDF with your solutions to the non-coding problems. The solutions must be *typed* (e.g., Word, Google Docs, or LaTeX) and each problem appropriately tagged on Gradescope. If we must search through your entire PDF for each problem, you may lose points! Note that you *must submit the code used to generate your results in the Code part of the assignment, otherwise you may not get any points for the results.*

2. **Submit code to the HW1-Code Assignment**: Your submitted code must contain the following files: 'q2.py', 'elastic.py', 'README.txt'. You must submit *ALL files you used to generate the results* but the autograder will only copy these files when running the test cases so make sure they are self-contained (i.e., capable of running standalone). Make sure you always upload *ALL* of these files when you (re)submit. The `README.txt` file *must contain a signed honor statement* that contains the following words:

   ```
   /* THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT
   CONSULTING CODE WRITTEN BY OTHER STUDENTS
   OR LARGE LANGUAGE MODELS LIKE CHATGPT.
   Your_Name_Here */

   I collaborated with the following classmates for this homework:
   <names of classmates>
   ```

1. **(Written) Ridge Regression** (10 pts): Show that the ridge regression estimates can be obtained by ordinary least squares regression on an augmented data set. We augment the centered matrix X with k additional rows with the value $\sqrt{\lambda}\mathbf{I}$ and augment y with k zeros. The idea is that by introducing artificial data having response value zero, the fitting procedure is forced to shrink the coefficients toward zero.

2. **Predicting Appliance Energy Usage using Linear Regression** ($2 + 3 + 3 + 3 + 2 + 3 + 3 + 2 + 4 + 2 + 10 + 3 = 40$ pts)

   Consider the Appliances energy prediction dataset (`energydata.zip`), which contains measurements of temperature and humidity sensors from a wireless network, weather from a nearby airport station, and the recorded energy use of lighting fixtures to predict the energy consumption of appliances (`Appliances` attribute) in a low energy house. The data has been split into three subsets: training data from measurements up to 3/20/16 5:30, validation data from measurements between 3/20/16 5:30 and 5/7/16 4:30, and test data from measurements after 5/7/16 4:30. There are 26 attributes[1] for each 10-minute interval, which is described

[1]We have removed the last two random variables as they aren't relevant for this class.

in detail on the UCL ML repository, Applicances energy prediction dataset. Your goal is to predict the Appliances For this problem, you will use `scikit-learn` for linear regression, ridge regression, and lasso regression. All the specified functions should be in the file 'q2.py'. The functions in 'q2.py' will be tested against a different training, validation, and test set, so it should work for a variety of datasets and assume that the data has been appropriately pre-processed (i.e., do not do any standardization or scaling or anything to the data prior to training the model). Any additional work such as loading the file, plotting, and required analysis with the data (e.g., parts 2e, 2h, 2j, etc.) should be done in a separate file and submitted with the Code.

(a) (Written) How did you preprocess the data? Explain your reasoning for using this pre-processing.

(b) (Code) Write a Python function `preprocess_data(trainx, valx, testx)` that does what you specified in 2a above. If you do any feature extraction, you should do it outside of this function. Your function should return the preprocessed trainx, valx, and testx.

(c) (Code) Write a Python function `eval_linear1(trainx, trainy, valx, valy, testx, testy)` that takes in a training set, validation set, and test set, (in the form of a numpy arrays), respectively, and trains a standard linear regression model *only on* the training data and reports the RMSE and R^2 on the training set, validation set, and test set. Your function must return a dictionary with 6 keys, 'train-rmse', 'train-r2', 'val-rmse', 'val-r2', 'test-rmse', 'test-r2' and the associated values are the numeric values (e.g., {'train-rmse': 10.2, 'train-r2': 0.3, 'val-rmse': 7.2, 'val-r2': 0.2, 'test-rmse': 12.1, 'test-r2': 0.4}).

(d) (Code) Write a Python function `eval_linear2(trainx, trainy, valx, valy, testx, testy)` that takes in a training set, validation set, and test set, respectively, and trains a standard linear regression model *using* the training and validation data together and reports the RMSE and R^2 on the training set, validation set, and test set. Your function should follow the same output format specified above.

(e) (Written) Report (using a table) the RMSE and R^2 between 2c and 2d on the energydata. How do the performances compare and what do the numbers suggest?

(f) (Code) Write a Python function `eval_ridge1(trainx, trainy, valx, valy, testx, testy, alpha)` that takes the regularization parameter, `alpha`, and trains a ridge regression model *only on* the training data. Your function should follow the same output format specified in (a) and (b).

(g) (Code) Write a Python function `eval_lasso1(trainx, trainy, valx, valy, testx, testy, alpha)` that takes the regularization parameter, `alpha`, and trains a lasso regression model *only on* the training data. Your function should follow the same output format specified in (a), (b), and (d).

(h) (Written) Report (using a table) the RMSE and R^2 for training, validation, and test for all the different (λ) values you tried. What would be the optimal parameter you would select based on the *validation* data performance?

(i) (Code) Similar to part 2d, write the Python functions, `eval_ridge2(trainx, trainy, valx, valy, testx, testy, alpha)` and `eval_lasso2(trainx, trainy, valx, valy, testx, testy, alpha)` that train ridge and lasso using the training and validation set.

(j) (Written) Use the optimal regularization parameter from 2h and report the RMSE and R^2 on the training set, validation set, and test set for the functions you wrote on 2i? How does this compare to the results from 2h? What do the numbers suggest?

(k) (Written) Generate the coefficient path plots (regularization value vs. coefficient value) for both ridge and lasso. Also, note (line or point or star) where the optimal regularization parameters from 2h are on their respective plots. Make sure that your plots encompass all the expected behavior (coefficients should shrink towards 0).

(l) (Written) What are 3 observations you can draw from looking at the coefficient path plots, and the metrics? This should be different from your observations from 2e, 2h, and 2j.

3. **(4 + 5 + 10 + 2 + 4 + 10 + 5 + 5 + 5 = 50 pts) Predicting Appliance Energy Usage using SGD**

Consider the Appliances energy prediction Data set from the previous problem. A template file, elastic.py, defines a class ElasticNet that takes in the regularization parameters, el (λ),[2] alpha (α), eta (η) or the learning rate, the batch size (batch $\in [1, N]$), and epoch or the maximum number of epochs as parameters when creating the object (i.e., `elastic = new ElasticNet(el, alpha, eta, batch, epoch)`). You will implement ElasticNet using stochastic gradient descent to train your model. The functions in 'elastic.py' will be tested against a different training, validation, and test set, so it should work for a variety of datasets and assume that the data has been appropriately pre-processed (i.e., do not do any standardization or scaling or anything to the data prior to training the model). For this problem, you *ARE NOT* allowed to use any existing toolbox/implementation (e.g., `scikit-learn`). Similar to problem 2, any additional work outside of (Code) should be done in a separate file and submitted with the Code for full credit.

(a) (Code) Implement the loss objective helper function in `elastic.py`. As a reminder, the optimization problem is:

$$\min f_o(\mathbf{x}) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda\left(\alpha||\boldsymbol{\beta}||_2^2 + (1-\alpha)||\boldsymbol{\beta}||_1\right), 0 \le \alpha \le 1 \tag{1}$$

In other words, given the coefficients, data and the regularization parameters, your function will calculate the loss $f_o(\mathbf{x})$ as shown in Eq (1).

(b) (Code) Implement the gradient helper function in `elastic.py`. You may find it helpful to derive the update for a single training sample and to consider proximal gradient descent for the $||\boldsymbol{\beta}||_1$ portion of the objective function. As a reminder, given step size η and regularization parameter λ such that $f(\mathbf{x}) = g(\mathbf{x}) + \lambda||\mathbf{x}||_1$, the proximal update is:

$$prox(x_i) = \begin{cases} x_i - \lambda\eta & \text{if } x_i > \lambda\eta \\ 0 & \text{if } -\lambda\eta \le x_i \le \lambda\eta \\ x_i + \lambda\eta & \text{if } x_i < -\lambda\eta \end{cases}$$

(c) (Code) Implement the Python function `train(self, x, y)` for your class that trains an elastic net regression model using stochastic gradient descent. Your function should return a dictionary where the key denotes the epoch number and the value of the loss associated with that epoch.

[2] lambda is not used since it is a Python function and can cause confusion.

(d) (Code) Implement the Python function `coef(self)` for your class that returns the learned coefficients as a numpy array.

(e) (Code) Implement the Python function `predict(self, x)` that predicts the label for each training sample in x. If x is a numpy $m \times d$ array, then y is a numpy 1-d array of size $m \times 1$.

(f) (Written) For the optimal regularization parameters from ridge (λ_{ridge}) and lasso (λ_{lasso}) from 2h, and $\alpha = \frac{1}{2}$, what are good learning rates for the dataset? Justify the selection by trying various learning rates and illustrating the objective value ($f_o(\mathbf{x})$) on a graph for a range of epochs (one epoch = one pass through the training data)[3]. For the chosen learning rate you identified, what are the RMSE and R^2 for the elastic net model trained on the entire training set on the training, validation, and test sets?

(g) (Written) Using the learning rate from the previous part, train elastic net (using only training data) for different values of α (it should encompass the entire range and include $\alpha = 0, 1$). Report the RMSE and R^2 for the models on training, validation, and test set.

(h) (Written) Based on the results from (c) and 2(a) and 2(c), what conclusions can you draw in terms of RMSE and R^2? Which model is the best? Also, discuss the differences between the SGD-variants of Ridge and LASSO and the standard implementations (Problem 2).

(i) (Written) What are the final coefficients that yield the best elastic net model on the test data? Compare these with the final coefficients for the best-performing model on the validation dataset. Are there noticeable differences? If so, discuss the differences with respect to the impact on the performance.

[3]You do not need to use the entire training set. SGD, in theory, is not sensitive to the dataset. Thus, you can subsample a reasonable percentage of data to tune the learning rate.