CS 534: Machine Learning
Homework 1

(Due September 28th at 11:59pm)

Submission Instructions:

The homework must be submitted electronically on Canvas. The code has to be done in Python 3.x. The homework should be submitted as a single Python file uploaded to Canvas. Please name your file **hw1.py** to make grading easier. Assume the data is in the same directory as your source code (`./`, do not hardcode path to data.

There is no need to attach the data, and you can assume the data will be in the same directory as the code.

1 P1: (30 pts) Predicting Appliance Energy Usage using Linear Regression

Create method: **LinearRegression()**

Consider the Appliances energy prediction Data set (`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 in a low energy house. The dataset is 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 27 attributes[1] for each 10 minute interval, which are described in detail on the UCL ML repository, Applicances energy prediction dataset.

For this problem, you can use `https://scikit-learn.org/stable/modules/linear_model.html` for linear regression, ridge regression, and lasso regression.

Include comments in your code clearly labeled like **P1A**, etc. answering each of the following questions:

1. Train a standard linear regression model *only on* the training data. What are the RMSE and R^2 on the training set, validation set, and test set?

2. Train a standard linear regression model using training and validation together. What are the RMSE and R^2 on the training set, validation set, and test set? How does this compare to the previous part? And what do the numbers suggest?

3. Train ridge regression and lasso regression *only on* the training data. You will want to consider a range of parameter values (λ) to find the optimal regularization parameter that gives you the lowest RMSE or R^2 on the *validation dataset*. Report the RMSE and R^2 for training, validation, and test for all the different (λ) values you tried.

4. Similar to part (b), train ridge and lasso using both the training and validation set (with your optimal regularization parameter from (c)). What are the RMSE and R^2 on the training set, validation set, and test set? How does this compare to the previous part?

2 P2: Faking Ridge Regression loss. 10pts

Create method: **FakeRidgeRegression()** to be invoked on the training data **train.csv**. Show that the ridge regression estimates can be obtained by ordinary least squares regression on an

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

augmented data set. We augment the centered matrix $\mathbf{X}$ with k additional rows $\sqrt{\lambda}\mathbf{I}$ and augment $\mathbf{y}$ with k zeros. The idea is that by introducing artificial data having response value zero, the fitting procedure is forced to shrink the coefficients towards zero.

Implementation notes: Use ordinary regression with your augmented data matrix. Then use real ridge regression on the original train data: `https://scikit-learn.org/stable/modules/linear_model.html`. Use the same parameter value λ as for your data augmentation. Compare coefficients.

To submit: Include comment labeled P2, listing Fake coefficients learned, and and real cofficients learned.

3 P3: 10pts: Modified Loss with Domain Knowledge

Create method: **LinearRegressionWithKnowledge()**

If we use the regular Ridge Regression, then many of the coefficients will be estimated near zeros, which doesn't conform with the experts' domain knowledge by incorporating an expert's intuition about importance of coefficients. For this problem, you want to incorporate this prior knowledge into your Ridge Regression as much as possible.

You have been thinking for some time, and finally, you came up with an idea. The idea is to modify the penalty term of the loss function to reflect the prior knowledge. A new loss function should be as follows:

```
np.sum(np.square(y - np.dot(X, coef))) + lmbd * np.sum(np.square(coef - coef_prior))
```

Implementation notes: You may use scikitlearn SGD regression but will need to modify the loss function.

To submit: Include comment labeled P3, listing both the Ridge Regression Coffecients (e.g., from part 1) and the coeeficients learned with your custom penalty term.