Econ 3389 Big Data: Homework 3

Due *Wednesday*, March 16th (Beginning of class)

1 Programming Practice

Load boston data Before beginning these exercises, load the public `boston` data set using the following code.

```
from sklearn.datasets import load_boston
boston = load_boston()
boston_data = pd.DataFrame(data =
                np.hstack([boston.target[:,np.newaxis], boston.data]),
                columns = ["price"] + list(boston["feature_names"]))
DIS = boston_data.ix[:, "DIS"].values[:,np.newaxis]
NOX = boston_data.ix[:, "NOX"].values[:,np.newaxis]
```

Polynomial Regression Extending what we learned in Week 5, our objective in this question will be to fit and evaluate the model.

$$NOX_i = \beta_0 + \beta_1 DIS_i^1 + \beta_2 DIS_i^2 + \cdots + \beta_r DIS_i^r + \epsilon_i$$

1. Create a function `create_polynomial_matrix(X, degree)`, where `X` is a column array and `degree` is an integer. This function must return a new array where each r^{th} column equals to Z^r elementwise, for r smaller than or equal to `degree` . That is,

$$\begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} \mapsto \begin{bmatrix} Z_1^0 & Z_1^1 & Z_1^2 & \cdots & Z_1^r \\ Z_2^0 & Z_2^1 & Z_2^2 & \cdots & Z_2^r \\ Z_3^0 & Z_3^1 & Z_3^2 & \cdots & Z_3^r \end{bmatrix}$$

2. Recall[1] that the formula for the OLS coefficients, when written in matrix notation, is given by[2]:

$$\widehat{\beta} = (X'X)^{-1}X'Y$$

   where X is a matrix of regressors, possibly containing a column of ones for the intercept, and Y is a column vector containing the dependent variable.

   Write a function `ols_coefficients(X, Y)` that takes conformable arrays `X` and `y`, and returns the $\widehat{\beta}$ coefficients, as computed by the formula above.

3. Write a function `ols_predict(X, betahat)` that takes conformable arrays `X` and `betahat`, and returns the predicted values of the dependent variable.

4. Write a function `ols_mse(y, yhat)` that computes the mean squared difference (not the R^2!) between predicted and actual values.

5. Now we shall put it all together. Let `X` be the polynomial matrix in `DIS`, and let `Y` be a column array containing `NOX`. For `r` in $\{0, \cdots, 8\}$, do the following.

   - Use `create_polynomial_matrix` to create a matrix `X` containing the polynomial arrays in the variable `DIS`.
   - Use `scikit_learn.cross_validation.train_test_split` to divide `X` and `Y` into a training and a test test. The training set should be contain 70% of the observations.
   - Use `ols_coefficients` to estimate the OLS coefficients associated with the polynomial matrix X (you should get an r-by-1 array).
   - Use `ols_predict` to predict values in the train and test set.
   - Use `ols_mse` to find out test and training set MSE, and print out the values of both.

[1] Or learn!

[2] The notation A' means the *transpose* of the matrix A.

Extra (Not for credit) Repeat all the steps in 5, but instead of using `ols_coefficients`, write the function `ridge_coefficients(X, Y, p)`. This function is identical to its OLS counterpart, except it computes the coefficients using the formula:

$$\widehat{\beta} = (X'X + pI)^{-1}X'Y$$

where p is a positive real number, and I is an identify matrix of appropriate dimensions.

Think What will happen to the magnitude of the coefficients as you increase or decrease p? How does the MSE compare with OLS? How do your results relate to the Gauss-Markov assumptions?[3]

2 Theory

1. Textbook exercise 2.4.3
2. Textbook exercise 2.4.5
3. Textbook exercise 3.7.4 (Feel free to test your intuition by modifying your code in the Programming Practice part!)

[3] In just a few weeks we will spend a lot a time studying this tiny change in the OLS formula. Stay tuned!