CPSC 483 - Introduction to Machine Learning

Project 2, Fall 2020

due October 12 (Section 02) / October 15 (Section 01)

Last updated Saturday October 10, 6:45 pm PDT

Now that we have derived an analytical solution to find the vector of estimated parameters ***w***, we can work directly with NumPy to implement linear regression as matrix and vector multiplication.

As demonstrated in the notebook Linear regression in vector and matrix format accompanying the textbook, this can also be generalized relatively easily to polynomial regression.

The project may be completed individually or in a group of no more than three (3) people. All students on the team must be enrolled in the same section of the course.

Platforms

The platform requirements for this project are the same as for Project 1.

Libraries

You will need scikit-learn to obtain the data and split it into training and test sets, and NumPy to do matrix and vector multiplication. You may also wish to use pandas DataFrames to examine and work with the data, but this is not a requirement. Use Matplotlib's pyplot framework or pandas to visualize your results.

Note that you may *not* use scikit-learn to do the computation for any answer. You may, however, wish to use the results of `sklearn.linear_model.LinearRegression` to spot-check the answers you receive from your NumPy code.

You may reuse code from the Jupyter notebooks accompanying the textbook and from the documentation for the libraries. All other code and the results of experiments should be your own.

Dataset

The scikit-learn `sklearn.datasets` module includes some small datasets for experimentation. In this project we will use the Boston house prices dataset to try and predict the median value of a home given several features of its neighborhood.

See the section on scikit-learn in Sergiy Kolesnikov's blog article Datasets in Python to see how to load this dataset and examine it using pandas DataFrames.

Reminder: while you will use scikit-learn to obtain and split the dataset, the rest of your code must use NumPy directly.

Experiments

Run the following experiments in a Jupyter notebook, performing each action in a code cell and answering each question in a Markdown cell.

All code should be written in vector and matrix format. Note that this precludes use of scikit-learn's `mean_squared_error()` and some NumPy features such as `np.polyfit()`.

1. Load and examine the Boston dataset's features, target values, and description.

2. Use `sklearn.model_selection.train_test_split()` to split the features and target values into separate training and test sets. Use 80% of the original data as a training set, and 20% for testing.

3. Create a scatterplot of the training set showing the relationship between the feature LSTAT and the target value MEDV. Does the relationship appear to be linear?

4. With LSTAT as $\boldsymbol{X}$ and MEDV as $\boldsymbol{t}$, use `np.linalg.inv()` to compute $\boldsymbol{w}$ for the training set. What is the equation for MEDV as a linear function of LSTAT?

5. Use $\boldsymbol{w}$ to add a line to your scatter plot from experiment *(3)*. How well does the model appear to fit the training set?

6. Use $\boldsymbol{w}$ to find the response for each value of the LSTAT attribute in the test set, then compute the test MSE $\mathcal{L}$ for the model.

7. Now add an x^2 column to LSTAT's x column in the training set, then repeat experiments *(4)*, *(5)*, and *(6)* for MEDV as a quadratic function of LSTAT. Does the quadratic polynomial do a better job of predicting the values in the test set?

8. Repeat experiments *(4)* and *(6)* with all 13 input features as $\boldsymbol{X}$ and using `np.linalg.lstsq()`. (See the Appendix to Linear regression in vector and matrix format for details of why we need to switch away from `np.linalg.inv()`, and the notes

for `np.linalg.solve()` for why we shouldn't use that either.) Does adding additional features improve the performance on the test set compared to using only LSTAT?

9. Now add x^2 columns for all 13 features, and repeat experiment *(8)*. Does adding quadratic features improve the performance on the test set compared to using only linear features?

10. Compute the training MSE for experiments *(8)* and *(9)* and compare it to the test MSE. What explains the difference?

11. Repeat experiments *(9)* and *(10)*, adding x^3 columns in addition to the existing x and x^2 columns for each feature. Does the cubic polynomial do a better job of predicting the values in the training set? Does it do a better job of predicting the values in the test set?

Submission

Submit your Jupyter `.ipynb` notebook file through Canvas before class on the due date. Your notebook should include the usual identifying information found in a `README.TXT` file.

If the assignment is completed by a team, only one submission is required. Be certain to identify the names of all students on your team at the top of the notebook.