Department of Computational and Data Sciences
DS226: ICAIML
November 3, 2022

Instructions

1. Writing your code

1) Download the zip file for the template code from Moodle and extract it. Inside, you will find four python files -
   (a) `LinearRegression.py`, which includes the parent class
   (b) `NormalEquation.py`, with the code for solving the normal equation
   (c) `LinearRegression1D.py`, with the code for a linear estimator in one variable, and
   (d) `Test.py`, to test the linear regression solvers.
2) To begin with, open `NormalEquation.py` and fill in your code in the `fit` method, in the section marked

   ```
   ### Fill your code here ###
   ```

   In this method, the training data is first prefixed with a column of 1s, to account for the intercept (bias) term. Then the normal equation is solved and assigned to the list `sol`. The first element of `sol` is the intercept, and the remainder of the list are the coefficients. You are expected to use the inverse and dot product functions defined in `numpy` for this code.
3) Next, you'll write the code to calculate the intercept and coefficient for a simple linear regression problem with one variable, defined in `LinearRegression1D.py`.

   ```
   self._coeff = ### Code for calculating the coefficient here ###
   self._intercept = ### Code for calculating the intercept here ###
   ```

   Again, use `numpy` functions to make your code clear and concise.
4) The two estimators implemented above are child classes derived from the parent class `LinearRegression` in `LinearRegression.py`. In it are implemented three methods common to both estimators -
   (a) `predict` - Given a new observations `Xnew`, predict the target using `self._intercept` and `self._coeff` from the model you've fit. Fill the code in

   ```
   self._predicted = ### Fill your code here ###
   ```

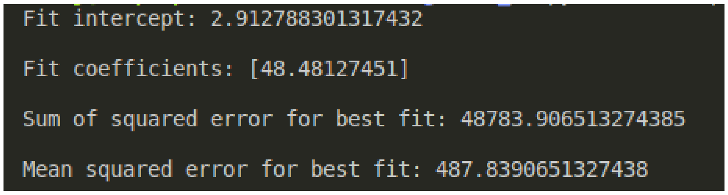
   (b) `error_metrics` - Prints the sum of squared errors and mean of squared errors between the target, y (`self._target`) and the predicted value, $\hat{y}$ (`self._fit`). Fill the code in

   ```
   self._sse = ### Fill in code for sum of squared errors ###
   self._mse = ### Fill in code for mean of squared errors ###
   ```

   (c) `fit_plot` - Scatter plot between true value (`self._target`) and the predicted value (`self._fit`) for the fit. A dotted line should show the true values for reference. Your plot should look like `Fit_Plot.jpg` included in the template folder.

2. Testing your code

You can test your code on any dataset. A sample test has been provided in `Test.py`. The output for this sample test should look like -



```
Fit intercept: 2.912788301317432

Fit coefficients: [48.48127451]

Sum of squared error for best fit: 48783.906513274385

Mean squared error for best fit: 487.8390651327438
```