Project Description

This project is to estimate a PM2.5 value from a sequence of observation data from an air quality check station in Taiwan. The data has 18 features, including PM2.5 itself, and the data is a series of data with continuing 9 hours observation. Consequently, the input data is a 18x9 matrix, in which each row represents a feature. The output is a single value.

For better understanding this homework, in this file, I will show 3 ways to try accomplishing the linear regression goal.

All training data and test data can be found at:

https://www.kaggle.com/c/ml2020spring-hw1/overview

Data Preprocessing

Before heading to build linear regression structures and train data, the first step is to preprocess the data. The original data is a .csv file, with 240 days air quality data; in each day, data is provided by each hour in 24 hours. The data preprocessing procedure is concluded as follows:

1. Put the data by continuous hour sequence, by horizontal stack data from the next day to the one from the current day;

2. Continuously extract data from a continuous 9-hour sequence as the input. The input data size is 18x9, with all 18 features. The target will be the data in the 10th hour, following the input data from the original .csv file;

3. In the original data file, at the precipitation feature, 'NR' (no rain) should be transferred to a float number 0;

4. Calculate mean values and standard deviations for each feature; and normalize the data for each feature by representative mean value and standard deviation;

5. Export the training sets, and the total number of training sets is 24x240-9 = 5751.

Based on different training schemes, the format of the output/target is different. This will be described in detail in the following sections.

Trial Method 1

Since the input data is a 18x9 matrix and the target value is only a single float type value, the first trail method is to expand the 18x9 matrix to a list/vector (1-d matrix) with 18*9 = 162 elements. The weight **W** is also a vector with 162 elements, and the bias b is just a float number. The model is then shown as the following function:

$$y_i = \mathbf{W}^{\mathrm{T}}\mathbf{X}_i + b$$

In which:

W: weight vector, with 162 elements;

X: input data with features, as a 162-element vector;

b: bias, a float value;

y: output value;

i: *i*th training data set.

When generating training data sets at preprocessing, the preprocessed target is the 10th hour PM2.5 value, with the previous 9 hours data as input. Loss function is the squared error. Loss functions and gradients for **X** and *b* are shown as the follows:

$$L = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial \mathbf{W}} = 2\sum_{i=1}^{n} (y_i - \hat{y}_i)\mathbf{X}_i$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial b} = 2\sum_{i=1}^{n} (y_i - \hat{y}_i)$$

In which:

$\hat{y}_i$: target value for the *i*th training data set;

L: loss function.

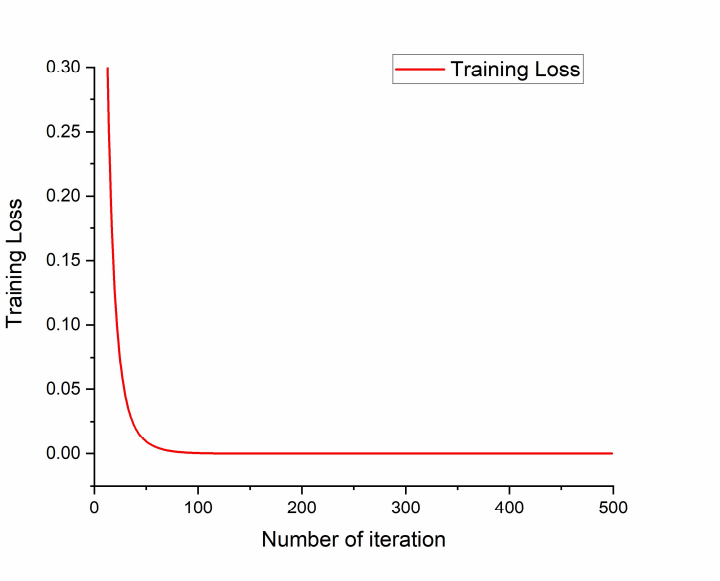
And the gradient decent algorithm is:

$$\mathbf{W} = \mathbf{W} - \eta\frac{\partial L}{\partial \mathbf{W}}$$

$$b = b - \eta\frac{\partial L}{\partial b}$$

The learning rate η is chosen as 0.0005. Initial values for each element in **W** and b are all 0. For simplicity, 15 data sets from the original training data file are used as the training data, and 5 data sets from the original training data file are used as the validation sets. Both training sets and validation sets are chosen randomly.

The relationship between total training loss and the iteration time is shown in the following figure:



After enough iterations (around 200 iterations), the total training loss is around $6\text{x}10^{-7}$, which is considered an acceptable training loss.

Use **W** and *b* results from 200 iterations, the mean squared error for the validation set is 0.548. The result from Sklearn is calculated as a reference, whose mean squared error for the validation data is 0.606.

Trial Method 2

Another way to consider the problem is to hold the structure of the input data with 18x9 matrix instead of expanding it like in Trial Method 1. The model is described as the following function:

$$y_i = \mathbf{W}_1^{\mathrm{T}}\mathbf{X}_i\mathbf{W}_2 + b$$

In the model function, $\mathbf{W}_1$ is a 18x1 vector, and $\mathbf{W}_2$ is a 9x1 vector. The same loss function (squared error) from the first method is used. The gradient for $\mathbf{W}_1$ and $\mathbf{W}_2$ are shown as follows:

$$\frac{\partial L}{\partial \mathbf{W_1}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial \mathbf{W_1}} = 2\sum_{i=1}^{n}(y_i - \hat{y_i})\mathbf{X}_i\mathbf{W_2}$$

$$\frac{\partial L}{\partial \mathbf{W_2}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial \mathbf{W_2}} = 2\sum_{i=1}^{n}(y_i - \hat{y_i})\mathbf{X}_i^T\mathbf{W_1}$$

For this method, the choice of initial values for weights is essential: $\mathbf{W}_1$ and $\mathbf{W}_2$ cannot be all zero matrices, as what is chosen in Trail Method 1. Consequently, each element is chosen randomly by numpy from a standard normal distribution.

The results for this method are not shown, but the code is attached in the GitHub file. The efficiency of this method is lower than the trail method 1. However, this can be considered as a prototype of a neural network, which is comprised with several 'layers' of weight and bias.

Trial Method 3

A sample method is given by Prof. Hung-Yi Lee. The link is attached:
https://colab.research.google.com/drive/131sSqmrmWXfjFZ3jWSELl8cm0Ox5ah3C

The input data is a 18x9 matrix; for the output, the sample answer predicts all 18 features, instead of 1 feature required (PM2.5).