Report

Prerequisites

Before dealing with line fitting for the dataset given, my program has to be able to receive dataset from train/test files. To address this issue, I imported sys library and used the sys.argv function to receive arguments from command line in order to determine filename. After that, my program will load the datapoint by using the given function from the utilities file. Since every segment has to be dealt with individually, I split the dataset by calculating the total length in the array and dividing it by 20 (20 datapoint each segment) using numpy.array_split function, in order to create an array consisting of segments in the data file before calculating the best line fitting.

Linear Regression

For basic_1.csv file, judging by the scatter plot of the data, linear regression will give the best fit for the dataset and the equation for y predicts will be y = a + bx, where a is the y-intercept, b is the slope coefficient, x is the independent variable and y is the dependent variable. I used the matrix form of least square method to determine the estimates of the parameter a and b. The formula for the matrix form of least square is given by: A = ((X^T *X)^-1)*(X^T)*Y where A = [a, b], X = Input feature value for each instance , Y = Output value of each instance and T = Transpose. Since the equation for linear regression is y = a + bx, the first column of X must be all 1s. In my implementation, I first used numpy.reshape function to reshape the x and y dataset into 20x1 matrix form and used numpy.ones function to create 1s in the shape of 20x1 also. Besides that, I used numpy.column_stack function to combine the 1s and x together to form X. After getting the value of X and Y, I used the formula of least square to obtain the values of a and b. As a result, I obtained 1.681062589025976e-27 for the sum squared error(SSE) where the SSE formula is given by: SSE = (Observe Y - Predicted Y)^2. The SSE value for using the linear regression model is very low which ensure that the model fitted the data well.

Polynomial Regression

Since basic data files have the least noice in the dataset compared to adv and noise files, comparing the sum squared error of the regression model is able to prove which model has the best fit for the data. In basic_3.csv, judging by the scatter plot of the data set, it shows that the dataset can only be fitted by nonlinear

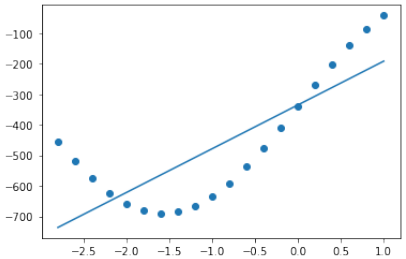
regression. Hence I tried to use polynomial regression to fit the data. By comparing the degree of polynomial regression 2-5, I found out that polynomial regression degree 3 has the best fit for the dataset as it achieved the lowest sum squared error among degrees 2-5. The degree 3 polynomial regression is y = a + b*x + c*(x^2) + d*(x^3) where the coefficient is obtained by using the same method as linear regression but the only difference is that we combined 1s, x, x^2 and x^3 for X.

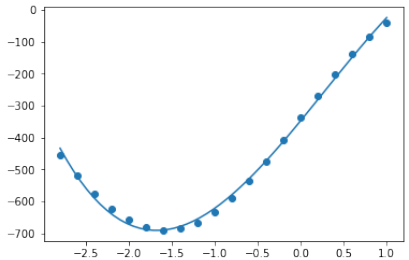
Table below shows the sum squared error for polynomial regression degrees 2-5 on the dataset in basic_3.csv.

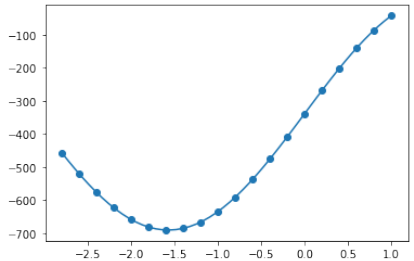
Polynomial Regression (Degree)	Sum squared error
2	15.7433186837487
3	9.89572290437569E-18
4	5.9378533969396E-14
5	2.6947302658801E-09

Unknown Nonlinear Regression

In basic_5.csv, the scatter plot of the dataset appears to be a nonlinear function however using degree 3 polynomial regression to fit the dataset will still produce a very high amount of sum squared error. Which means the degree 3 polynomial regression is not a good fit for the dataset. Judging from the scatter plot of the dataset, it appears to have sinusoidal function form (which is y = a + b*sin(x)) in it. By constructing sinusoidal regression with least square method, I was able to get a low and sensible sum squared error for the result which is 3.053459683057075e-25. By comparing the graph between linear regression, degree 3 polynomial regression and sinusoidal regression, it is obvious that sinusoidal regression is the best model to fit the dataset in basic_5.csv. Besides that the R-squared value also indicates that sinusoidal regression has the value that is nearest to 1 which means the model's predictions fit the data well.







K-fold Cross Validation

After dealing with basic_1 to basic_5 data files, we are able to conclude and identify linear regression, degree 3 polynomial regression and sinusoidal regression are the three regression models in this coursework and by choosing the lowest sum squared error among these three regression models in every segment will give us the best fit for the dataset. However, when dealing with dataset that have a lot of noise, using this method causes overfitting to happen. Which means that we might get the wrong regression model even though we achieved a very low sum squared error. When this occurs, the regression coefficient will represent the noise rather than the genuine relationship in the data set. In order to overcome the issue of overfitting, I decided to include cross validation in my program. Technically, the idea of cross validation is to estimate the performance of a regression model on unseen data by separating the dataset into train and test sets. First, it trains the model with the train dataset and then evaluate the performance of the model using test dataset. In this case, since our dataset is relatively small it is better to use K-fold Cross Validation than Train_Test Split approach. This is because the latter approach is prone to have high bias as we might miss information about the data that is not used to train the model, however K-fold Cross Validation ensures every dataset will have the chance of appearing in train and test sets. In my case, I chose the value of K as 5 because it may waste a lot of computational resources if the value of K is too high and value of 5 will still give a sensible result.

For the implementation of K-fold Cross Validation in my program, I separated them into 4 steps:

1. I shuffle the data randomly and split them into K groups
2. For each unique group, I split the data into train and test sets where 80% of the data goes to the train set while the remaining goes to the test set
3. I train the linear, degree 3 polynomial and sinusoidal regression model using the training set and calculate the sum squared error for each model until every K-fold serves as the test set
4. Then take the average of the sum squared error in each model and compare it

As a result, the model with the best result after K-fold cross validation will be the most sensible and accurate model to fit the dataset. Example of using K-fold cross validation to find the most sensible regression model without having overfitting issue is when dealing with noise_1.csv data file. By just calculating the sum squared

error between linear regression, degree 3 polynomial regression and sinusoidal regression, it showed that degree 3 polynomial regression obtained the lowest sum squared error. However, after using K-fold Cross Validation the linear regression obtained the lowest mean sum squared error. The Table below shows the K-fold Cross Validation result obtained for linear regression, degree 3 polynomial regression and sinusoidal regression.

Regression Model	Mean Sum Squared Error
Linear Regression	3.2950155954657
Degree 3 Polynomial Regression	3.41847916963722
Sinusoidal Regression	1870.97454340873

Conclusion

In this coursework, I have learned to use normal equations to find the coefficient in the regression model to fit the dataset. However, I found out that this approach is not good enough to give the most accurate predictions as most of the dataset contains noise. By using K-fold Cross Validation, even though the regression model will not give the best fit for the dataset given compare to other regression models, it will give a more sensible and accurate prediction. To summarize the functionality of my program, it is able to perform best line fitting for each individual segments with linear, degree 3 polynomial and sinusoidal regression without having the problem of overfitting by using K-fold Cross Validation. After that, the program will provide the output of sum squared error to show how well the fitting is. Besides that, I included an extra feature of generating the scatter plots of the dataset with the plot of the best line fitting when users add an extra parameter '—plot' behind the data file name after compiling the program. This allows users to visually check whether the solution is working well or not.