CW1: An Unknown Signal - Report

Abdizamed Ali

April 19, 2021

1 Methods

1.1 Regression

Least squares regression is used to create our model, which finds the fit of a given segment from the train data. The following equation for regression will give us the weights for a our fitted line where, $\hat{w} = (X^T X)^{-1} X^T Y$ Here X is different for polynomial and the unknown function (sin). X is shown below for both sin and polynomial.

$$\boldsymbol{X} = \begin{bmatrix} 1 & sin(x_1) \\ 1 & sin(x_2) \\ ... & ... \\ 1 & sin(x_n) \end{bmatrix} \tag{1}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & ... & {x_1}^n \\ 1 & x_2 & ... & {x_2}^n \\ ... & ... & ... & ... \\ 1 & x_n & ... & {x_n}^n \end{bmatrix} \tag{2}$$

$\hat{w}$ can easily be expressed in python as np.linalg.inv(X.T.dot(X)).dot(X.T).dot(Y)

1.2 Sum Squared Error

We use the sum squared error measure the similarity between our fitted line and our actual data, we call this the sum squared error(SSE throughout the report). The formula for the sum squared error is: $\sum_i (y_i - \hat{y}_i)^2$. This can easily be expressed in python using numpy as np.sum((y - $\hat{y}$) ** 2)

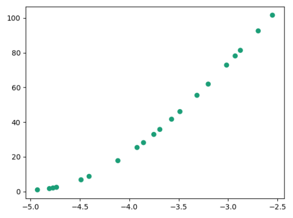
1.3 K-Fold Cross-Validation

As the given segments of our data sets are relatively small, we must prevent overfitting the data set. A common solution is cross-validation, this will allow any outliers in our small dataset to not have a large impact on our model. We decided to use k-fold cross validation which, uses the following approach to evaluate a mode:

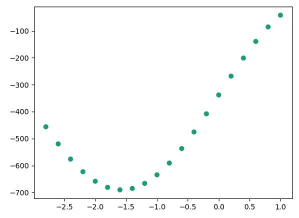
1. Randomly divide a dataset into k groups, or "folds", of roughly equal size keeping the pairs the same.
2. Choose one of the folds to be the holdout set. Fit the model on the remaining k-1 folds. Calculate the test mean squared error on the observations in the fold that was held out.
3. Repeat this process k times, using a different set each time as the holdout set
4. Calculate the overall test mean squared error to be the average of the k test mean squared errors.

2 Justification

As the coursework brief stated,*that the order of the polynomial (and the form of the unknown function) will not change between different files/segments.* Therefore, we decided to use basic_3.csv and basic_5.csv as the basis for the deciding the order polynomial and unknown functions. The reason why we chose basic_3.csv and basic_5.csv as the basis for the polynomial and unknown function was due to the fact that they do not have any noise, as can be seen in Figure 1. Furthermore, when looking ar Figure 1 it is easier to determine which of the graphs is polynomial and we came to the conclusion that Figure 1a was polynomial and Figure 1b must be the unknown.



(a) *Basic_3.csv*

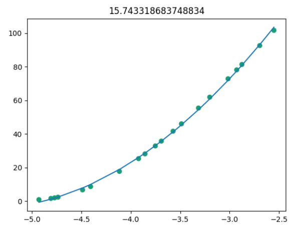


(b) *Basic_5.csv*

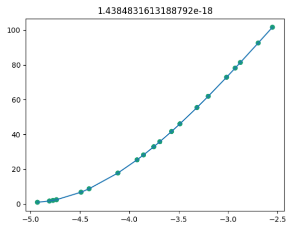
Figure 1: basic_3.csv and basic_5.csv plotted

2.1 Polynomial

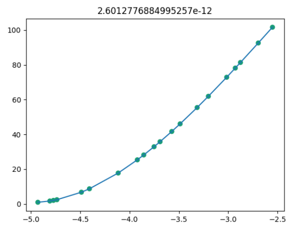
To find the degree of the polynomial Figure 1a a brute force approach was used. We produced a best fit line for the Figure 1a starting from a degree of two Figure 2a and taking note of the SSE (as you can see on the title of all the graphs in Figure 2 all the way up to a degree of seven Figure 2f. As you can see from Figures 2b 2c 2d they look as if the line is fitted perfectly but Figure 2b has the lowest SSE and as the degree increase the SSE seems to increase after a degree of 3. Therefore since Figure 2b has the lowest SSE and higher degrees only seem to increase the SSE, as can be seen in the titles Figure 2, we can safely assume the polynomial degree is 3.



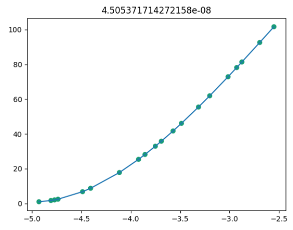
(a) $Degree = 2$



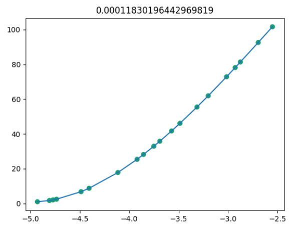
(b) $Degree = 3$



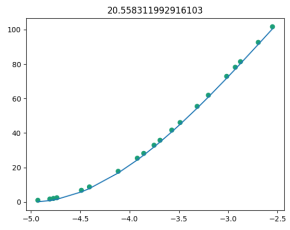
(c) $Degree = 4$



(d) $Degree = 5$



(e) $Degree = 6$

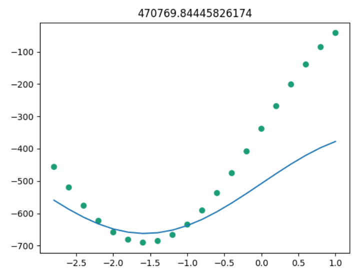


(f) $Degree = 7$

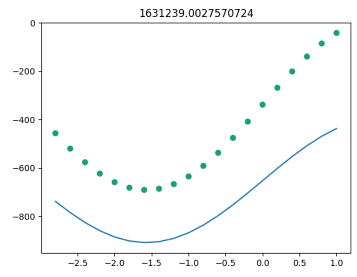
Figure 2: Different polynomial degree for the basic_3 graph

2.2 Unknown

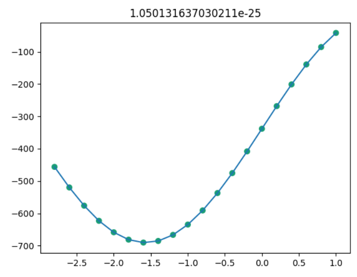
To find the unknown function we used a similar approach to the one used for the polynomial degree but rather than changing the degree we loop over different trigonometric functions which we fitted to our data points recording the SSE as you can see in Figure 3. This clearly showed that the unknown function was a sine function to a degree of accuracy according to the SSE.



(a) *Cos*



(b) *Exp*

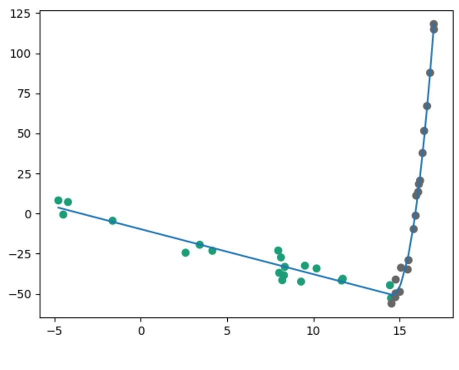


(c) *Sin*

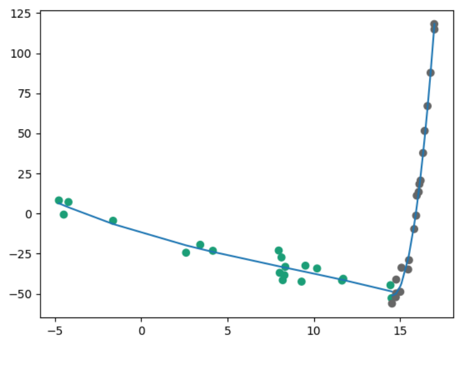
Figure 3: Different Trigonometric functions fitted for the basic_5 graph

3 Selection

To select between the three different possibilities (linear, polynomial, unknown) for a given data segment we first perform k-fold cross validation as explained in section 1.3 (note we used a K-value of 10 for the model as this produced the wanted results against the train data) on said segment for each function type . If the data is shuffled randomly, we repeat this process several times in a loop and take the mode to get an accurate result. we then preform regression as explained in Section 1.1 for the function type with the lowest cross-validation score and add that to the total error. This method does not only successfully select between the different functions but also accounts for overfitting as shown in Figure 4. For example, using the noise_2.csv data, the model without cross validation fails to fit the linear line with noise Figure 4b compared to my model which cross validates as you can see in Figure 4a



(a) Noise_2 cross-validated



(b) Noise_2 not cross-validated

Figure 4: noise_2.csv fitted with k-fold cross-validation and model using only lowest SSE

3.1 Results

The table shown below are the results from my final model on all the training data. We believe that the model has identified all the function types correctly allowing for the most accurate reconstruction error.

Files	**Reconstruction Error**	**Function Type**
adv_1.csv	199.7256013125421	Polynomial, Linear, Polynomial
adv_2.csv	3.685132050447601	Sine, Linear, Sin
adv_3.csv	1019.4370683679183	Linear, Polynomial, Sine, Linear, Sin, Polynomial
basic_1.csv	1.6881623371729653e-28	Linear
basic_2.csv	6.214646211331131e-27	Linear, Linear
basic_3.csv	1.4384517213810976e-18	Polynomial
basic_4.csv	1.3851152817518837e-12	Linear, Polynomial
basic_5.csv	1.050131637030211e-25	Sin
noise_1.csv	12.207460140137117	Linear
noise_2.csv	849.5527462320404	Linear, Polynomial
noise_3.csv	482.9090507852757	Linear, Polynomial, Sin

4 Conclusion

This report shows the use of least squares to select between 3 different function types of model, linear, cubic and sine, and then successfully select and fit the data to the function type. We can easily extend this model to fit a wider range of function types while also accounting for overfitting