Project Report on Linear Feature Engineering

09/14/2021
Quantitative Foundations

Submitted by – Hong Yang & Gursimran Singh

Instructor - Dr. Linwei Wang

Table 1 :

Features	***Final training error***	***Estimated Test Error***	***Degree (of x)***
1, x, x^2, x^3, ... x^n	*36.1144*	*52.9285*	*4*
1, x, x^1/2, x^1/3, ... x^1/n	*33.1966*	*56.1709*	*7*
sin ((x)^1/3), sin ((x^1/2)^1/3), sin ((x^1/3)^1/3) ...... sin((x^1/n)^1/3)	*33.1489*	*53.1398*	*6*
1,sin (x), sin (x^1/2), sin (x^1/3) ...... sin(x^1/n)	*29.7133*	*55.2416*	*8*

Introduction:

The goal of this project was to estimate dependent variable values from a given test data series by training a model on the data using feature engineering and making the least square error minimal. In this prediction we also need to take care of overfitting problems using cross validation methodology.

Feature selection process:

We started our feature selection process with natural polynomial features **(i.e., 1, x, x^2, x^3, ... x^n)**. Here we expended the data point x polynomial as polynomial function to y. Then we used the inverse polynomial **(1, x, x^1/2, x^1/3, ... x^1/n)** function to y. Then we moved a step forward to see a variation in results and found another improved version in the form of nth root(cube root) polynomial, as shown in **Table 1**. Then we made some more modifications to try trigonometric functions like sine and cosine. We found our best results using sine inverse polynomial function **(1,sin (x), sin (x^1/2), sin (x^1/3) ...... sin(x^1/n))**.
The selection of features was made keeping in mind the variation in data. We also did some other feature functions **(i.e, 1,sin(x), sin(x^2), sin(x^3) ...... sin(x^n)), (1,cos(x), cos(x^2), cos(x^3) ...... cos(x^n)), square root(similar to nth root), etc** , but found the error to be higher than previously mentioned feature functions. Other exciting functions can also be tried to fit the curve and get the behavior, but due to limitations on experimentation and unknown source, and limited knowledge of data, we compare the results of these features only.

Overfitting Models:

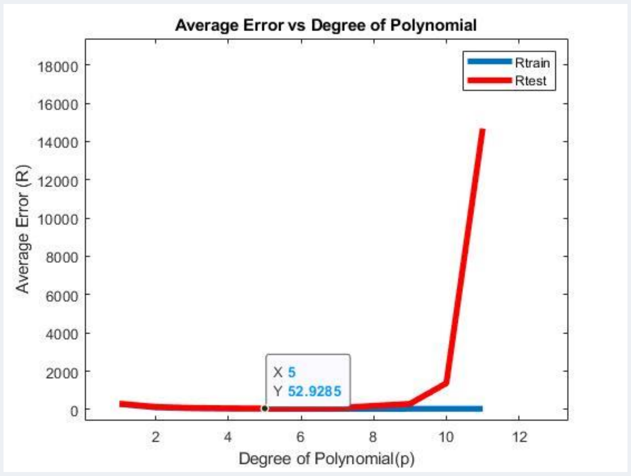
We used the k-fold cross-validation technique to avoid overfitting a model. We divided the training data into k subsets. Each time, one of the *k* subsets is used as the test set, and the

other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once and gets to be in a training set *k-1* times. The variance of the resulting estimate is reduced as *k* is increased. For our case, we used the number of chunks to be 10, as it gives the least mean square error and not taking more computing power, since the major disadvantage of this method is that as you increase k, you need to do more computation.
The cross-validation process helped in choosing the best degree of polynomial out of each model, which in our case is **sine inverse polynomial function**.

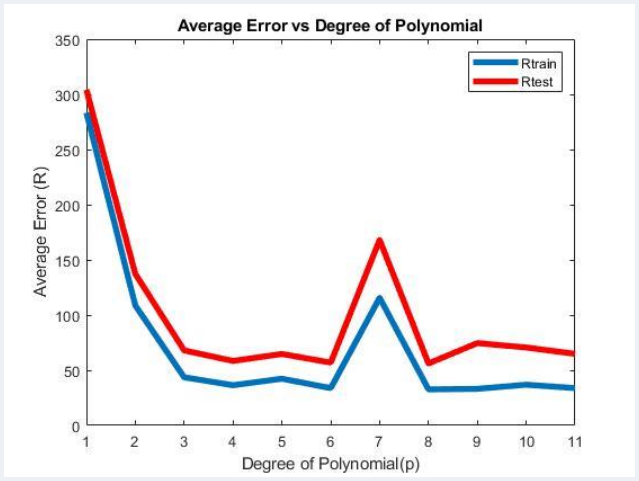
We used the **average test error** generated using **each model** for a particular **degree of polynomial** during **cross-validation** to estimate the **test error** for the given model.

Following are the plots of the four models present in **Table 1** showing training and test errors obtained during cross validation over different degrees of polynomial**.**

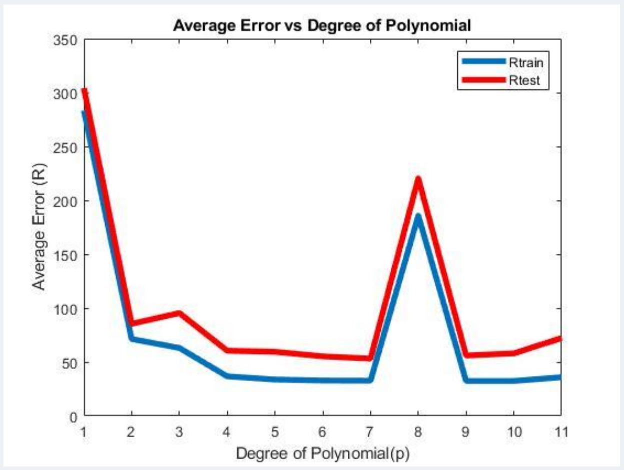
Normal Polynomial



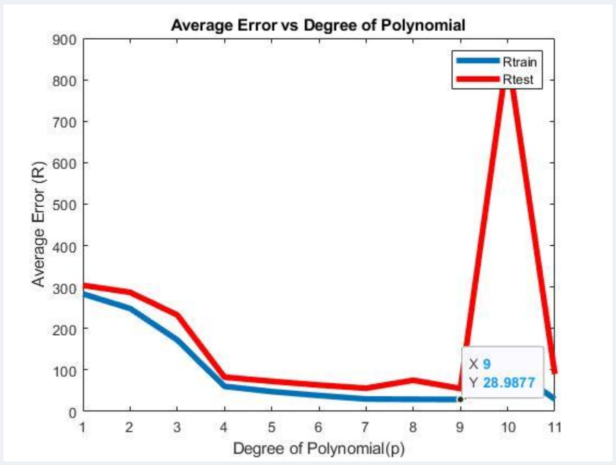
Inverse Polynomial



nth root polynomial



sine inverse polynomial


Average Error vs Degree of Polynomial
Rtrain
Rtest
Average Error (R)
Degree of Polynomial(p)
X 9
Y 28.9877