Midterm Report

Yaxuan Hou

118010098

I Multi-collinear & unit length scaling

When the researchers hope to find out whether there is a multi-collinear problem existing in the data set, a traditional way is analyzing the eigensystem of X^TX.

$$X = [x_1, x_2, \dots, x_k]$$

$$x_i = [x_{1,i}, x_{2,i}, \dots, x_{n,i}]^T$$

After performing the eigenvalue decomposition of X^TX and obtaining the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_p$, we rank the eigenvalues and compute the condition number of X^TX as $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$. If the condition number κ exceeds 1000, severe multicollinearity is indicated. Otherwise, it is numerically safe.

About the training data in the project, we perform the method mentioned above. The eigenvalues for this training data is

$$\lambda_1 = 1.9287 * 10^{11} \quad \lambda_2 = 3.7597 * 10^9$$

$$\lambda_3 = 3.8843 * 10^8 \quad \lambda_4 = 7.6721 * 10^7$$

$$\lambda_5 = 3.1075 * 10^7 \quad \lambda_6 = 1.5686 * 10^6$$

$$\lambda_7 = 3.5792 * 10^4 \quad \lambda_8 = 2.1069 * 10^4$$

We can calculate the $\kappa = \frac{\lambda_1}{\lambda_8} = 9.154 * 10^6 > 1000$. Thus, there is multicollinearity happening. Then use the unit length scaling for standardization.

$$w_{ij} = \frac{x_{ij} - \overline{x_j}}{s_{jj}^{\frac{1}{2}}}$$

$$y_i^0 = \frac{y_i - \bar{y}}{SS_T^{\frac{1}{2}}}$$

$$s_{jj}^{\frac{1}{2}} = \sum_{i=1}^{n} \left(x_{ij} - \overline{x_j}\right)^2 \quad SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

After performing the standardization on the original data set, we define X and Y again.

$$X = [w_1, w_2, \dots, w_k]$$

$$w_i = [w_{1,i}, w_{2,i}, \dots, w_{n,i}]^T$$

$$Y = [y_1^0, y_2^0, \dots, y_n^0]^T$$

Before making the model of the data, we try to find out the linear correlation between variables, which may be helpful to build the model.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	1.00000000	-0.92463203	-0.10645271	0.04857081	0.07386800	0.1046559071	0.063183145	-0.0159662471
[2,]	-0.92463203	1.00000000	0.00620147	-0.03754888	-0.06960989	-0.1091949253	-0.075115509	-0.0778299433
[3,]	-0.10645271	0.00620147	1.00000000	-0.36581939	-0.32337753	-0.3012874526	-0.305829267	-0.1186883460
[4,]	0.04857081	-0.03754888	-0.36581939	1.00000000	0.92889788	0.8597052755	0.922645440	0.1912203426
[5,]	0.07386800	-0.06960989	-0.32337753	0.92889788	1.00000000	0.8765994098	0.982673825	-0.0156756681
[6,]	0.10465591	-0.10919493	-0.30128745	0.85970528	0.87659941	1.0000000000	0.902728395	0.0003159772
[7,]	0.06318315	-0.07511551	-0.30582927	0.92264544	0.98267382	0.9027283954	1.000000000	0.0067756024
[8,]	-0.01596625	-0.07782994	-0.11868835	0.19122034	-0.01567567	0.0003159772	0.006775602	1.0000000000

Figure 1: correlations between variables

From figure 1, we get that $(1,2)$, $(4,5)$, $(4,6)$, $(5,6)$, $(4,7)$, $(5,7)$, $(6,7)$ have the correlations.

II Multiple linear regression model

After getting the new data set, we calculate the multiple linear regression model with the aforementioned 8 input variables. Because we already have done the unit length scaling, there is no need to put β_0 as parameters. We only need to calculate β_i while $i = 1,2, \dots 8$.

$$y = -0.9850x_1 - 1.0644x_2 + 0.0665x_3 - 0.1204x_4 + 0.3860x_5 - 0.3214x_6 + 0.1107x_7 + 0.5949x_8$$

After computing the linear regression model, we could get the number of R^2 and R^2_{adj}.

$$R^2 = 0.6427 \quad R^2_{adj} = 0.6424$$

After getting the fitted value, we do the t-test, and we get that the p_values of 8 variables are 0, which means that we reject the H_0, and accept the H_1. All of them are significant.

Apply the trained linear regression model to the test data set.

$$MSE = \frac{1}{n_t} \sum_{i=1}^{n_t=8256} \left(y_{test,i} - \widehat{y_{test,i}}(x_{test,i})\right)^2 = 4.3166 * 10^{-5}$$

Plot the R-studentized residuals versus the fitted value $\hat{y}$.

$$\hat{y} = X * \beta$$

$$e_i = Y - \hat{y}$$

$$t_i = \frac{e_i}{\sqrt{S_{(i)}^2(1 - h_{ii})}}$$

$$S_{(i)}^2 = \frac{(n-p)MS_{Res} - \frac{e_i^2}{1-h_{ii}}}{n-p-1}$$

While the h_{ii} is the ith diagonal of Hessian matrix, $X(X^TX)^{-1}X^T$. Besides, to draw the normal probability plot, we need to sort the value of t and correspond them to the $P_i = \frac{i-0.5}{n}$ $i = 1,2, \dots n$

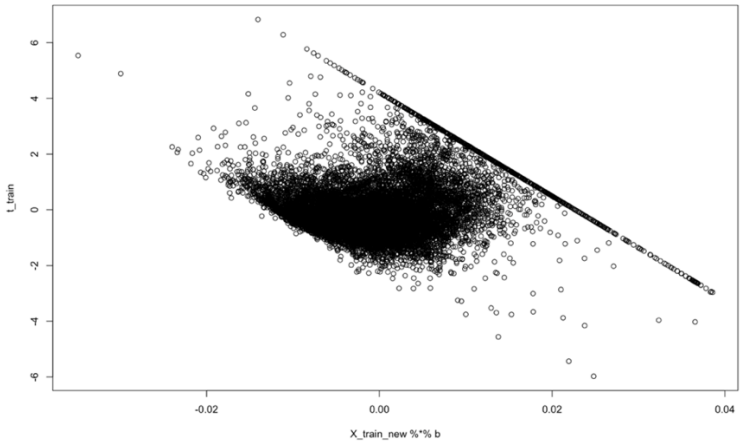

Figure 2: R-studentized residuals versus the fitted values $\hat{y}$

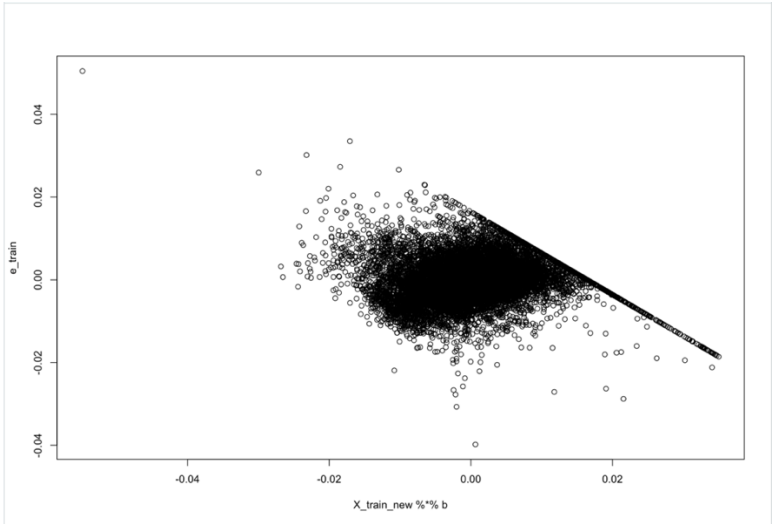

Figure 3: Residuals versus the fitted values $\hat{y}$

From figure 3 and 2, we can clearly see a straight line. That is because if e_i is beyond this line, the value of y may be negative.

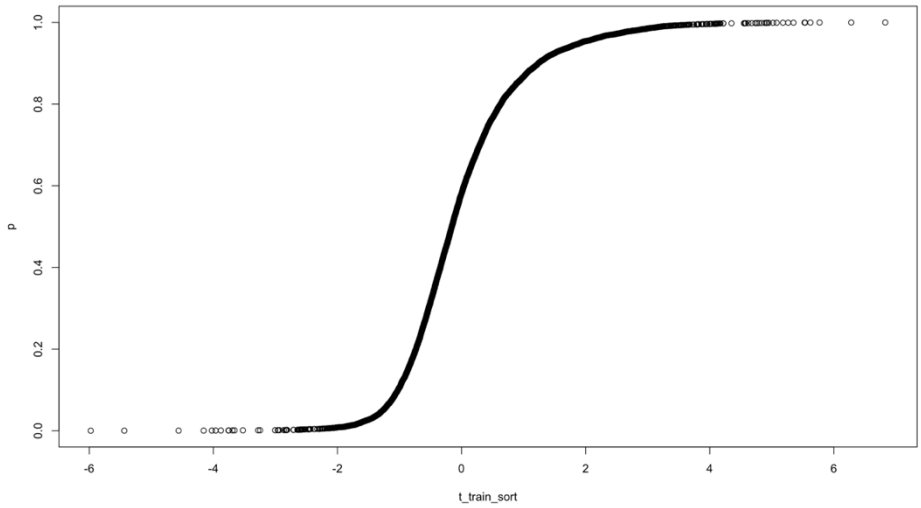

Figure 4: Normal probability plot

From figure 2 and figure 4, we can conclude that actually the random error terms follows the zero mean Gaussian i.i.d assumption. That is because most of t_i are between (-2,2) in figure1 and a straight line between around 0.33 to 0.67 in figure 2.

III Polynomial regression model

Randomly choosing the intersection part, which are $x_1x_2, x_3x_4, x_5x_6, x_7x_8$ four terms.

$$\begin{aligned} y = {} & \beta_{11}x_1 + \beta_{12}x_1^2 + \beta_{21}x_2 + \beta_{22}x_2^2 + \beta_{31}x_3 + \beta_{32}x_3^2 + \beta_{41}x_4 + \beta_{42}x_4^2 + \beta_{51}x_5 \\ & + \beta_{52}x_5^2 + \beta_{61}x_6 + \beta_{62}x_6^2 + \beta_{71}x_7 + \beta_{72}x_7^2 + \beta_{81}x_8 + \beta_{82}x_8^2 \\ & + \beta_1x_1x_2 + \beta_2x_3x_4 + \beta_3x_5x_6 + \beta_4x_7x_8 \end{aligned}$$

After performing the calculation, we get the model is

$$\begin{aligned} y = {} & -1.08x_1 + 28.54x_1^2 - 1.20x_2 + 47.67x_2^2 + 0.09x_3 + 5.07x_3^2 - 0.52x_4 \\ & + 4.08x_4^2 + 0.88x_5 - 9.62x_5^2 - 0.39x_6 + 1.42x_6^2 + 0.09x_7 + 3.00x_7^2 \\ & + 0.84x_8 - 9.80x_8^2 + 74.81x_1x_2 + 2.77x_3x_4 - 0.35x_5x_6 + 5.70x_7x_8 \end{aligned}$$

Also we calculate the R^2, R^2_{adj}, and MSE

$$R^2 = 0.7010 \quad R^2_{adj} = 0.7009$$

$$MSE = \frac{1}{n_t}\sum_{i=1}^{n_t=8256}\left(y_{test,i} - \widehat{y_{test,i}}\left(x_{test,i}\right)\right)^2 = 3.8539 * 10^{-5}$$

Compared with using the linear regression model, using the polynomial regression model has some benefits and disadvantages.

Benefits:

1. In this test, decreasing the number of MSE, increasing the number of R^2 and R^2_{adj}, which means that the polynomial regression model is fitted better than linear regression model.
2. The polynomial regression model provides more optional models.
3. Many functions can be approximated by a polynomial regression model.
4. The measured points can be approximated by increasing the higher-order term of x until satisfactory.

Disadvantages:

1. It is hard to determine the order term of x and the interacting term. There is existing problem of overfitting.

About the improvement of fitting the data, after adding L2 regularization term to the cost function, changing the λ from 0.01 to 100, actually there is no improvement in R^2 and R^2_{adj}. The values of them are similar. All of them are around 0.70. Furthermore, the value of two MSE also similar, and there is no difference. So the L2 regularization does not help.

IV Neural network model (NN model)

The NN model is regarding the x as the input layer and y as the output layer. By using different hidden layers, the data x is transformed to the first hidden layer, and then transformed to the second hidden layer, finally transformed to y. Also, the relationship between different layers is turned to be nonlinear by using the activation function, including sigmoid, tanh, and relu.

$$y = Wx \rightarrow y = AF(Wx)$$

By training the data, we put the x and y inside the model, then get the

transformation from x to y, which is also called NN model.

$$f(x;\theta) = f^{L+1}(\dots f^3(f^2(f^1(x;W_1);W_2);W_3);W_{L+1})$$

In the whole model to find out the best suitable W, we use the idea called gradient descent to get the fitted value, by setting a Δw, we calculate the $\frac{\Delta y}{\Delta w}$. If $\frac{\Delta y}{\Delta w} < 0$, moving right, which increases the w. Otherwise, moving left, which decreases the w. (while Δy means the change in cost function)

Back to this data we need to train. According to Heaton (2008), the optimal size of the hidden layer is usually between the size of the input and the size of the output layers. In that case, I set 1 hidden layer. The initial number of input data is 8. To the first hidden layer, the number is 3, and finally comes to the y. The reason behind this is that I test the number of neutrons from 2 to 7 and finally get the result that the number 3 has the smallest R^2_{adj}.

Table 1: multiple R-square, adjusted R-square, and prediction MSE of NN

Neutrons	2	3	4	5	6	7
R^2	0.6189	0.6281	0.6203	0.6104	0.5914	0.6168
R^2_{adj}	0.6186	0.6279	0.6201	0.6101	0.5911	0.6166

After training the data, we get the model, and also calculate the number of R^2_{adj}, and MSE of test data.

$$R^2_{adj} = 0.6279 \quad MSE = 8.105 * 10^{-5}$$

Compared with using the linear regression model and polynomial regression model, using the NN model has some benefits and disadvantages.

Benefits:

1. Easy to perform, we do not need to know the exact model of the training data set.
2. Using the dataloader, we could do batch training.
3. Setting the parameter lr, we could control the gradient descend.
4. Although this is a sophisticated procedure, the time used is similar to the linear regression model and the polynomial regression model.
5. We can set the number of hidden layers and activation function by ourselves.

Demerits:

1. The result may be worse than the linear regression model and the polynomial regression model, and we could not know what is wrong since we could not know the exact model.

V alternating conditional expectation (ACE) algorithm

After performing the ACE transformation in python, we could get the new data set. Doing the OLS, then get the value of R^2_{adj} and MSE of test data.

$$R^2_{adj} = 0.7197 \quad MSE = 0.28$$

After doing the transformation, the R^2_{adj} is similar to previous ones, even increase a little. However, the MSE is much higher than expected. That is the model is not performing well in the test model.

Besides, we could do the NN and the Polynomial regression analysis on the new dataset. The table below shows the results.

Table 2: fitted quality

Model:	OLS	Polynomial	NN
R^2	0.7199	0.7576	0.7224
R^2_{adj}	0.7197	0.7573	0.7222
$MSE_{predict}$	0.2852	0.2337	0.2639

From these three models, we could conclude that the polynomial model is the most fitted one.

And comparing the residual plots for the ACE model with the Ordinary LS without performing data transformation. We could get that actually the ACE model performs better. Transformed model superior to the original LS model. Because it is more concentrated between (-1,1).

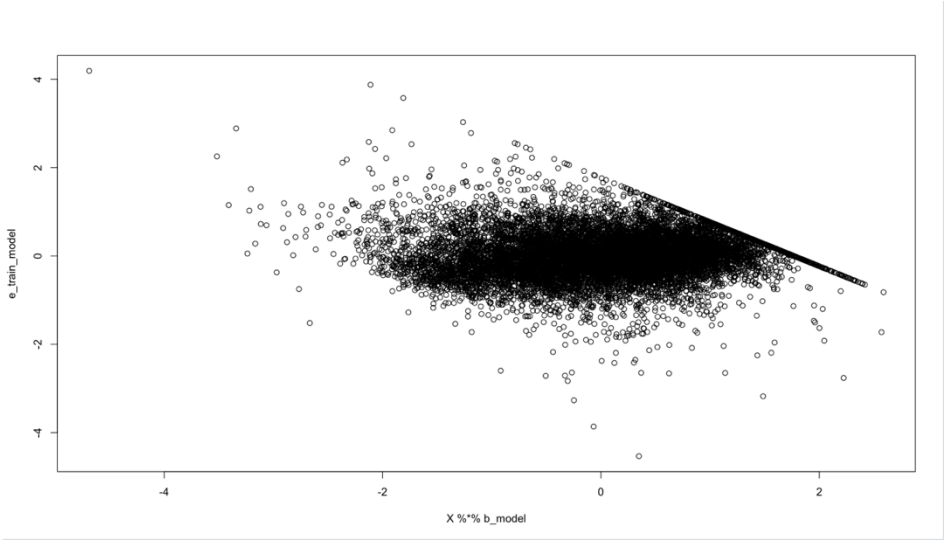

Figure 4: residual plot after transformation

Also, the ACE model has merits and demerits.

Merits:

1. The model is fitted better after transformation with the ordinary LS model, polynomial model and NN model.
2. The ACE package is easy to learn and the package is comprehensive.
3. If using R to do the transformation, the used time is negligible.

Demerits:

1. If we use the Python to get the final value, spend around 1 hour to run all results.
2. Because of the data transformation, it needs numerous time. However, there is no similar function like dataloader to do the batch modeling.
3. Install the ACE package is hard and difficult. Because it is an old package. It is easy to have conflicts with other packages like numpy and pandas.
4. After doing the transformation, the MSE of test data is performing badly.

VI outliers

From the Figure 2, R-studentized residuals versus the fitted values $\hat{y}$, we can know that there are outliers in the data set. Because there are some number higher than 2 or smaller then -2.

By using Cook's Measure, we could find out the influential points, which could be regarded as the outliers. After detected by using Cook's Measure, we can find that there is only one outlier and the fraction of outliers is much smaller compared with the ordinary data set. Moreover, we could use the robust regression analysis to build a new model.

Reference

Heaton, J. (2008) *Introduction to Neural Networks for Java, Heaton Research.*

Retrieved from

https://www.researchgate.net/publication/272508659_Introduction_to_Neural_Networks_for_Java_Heaton_Research