Final Report

Shuijian Zhang

12/7/2018

Final report

In the software developing, I can divide my work to two parts. First, I build and apply the MLP called multiple layer regression, also same as NN, to train and predict the RPE (training intensity and variety) for current U.S football player based on their muscle soreness, fatigue, sleep, duration, acute load average, chronic load average, TSB and so on. Recently I have two datasets f3 and f4, in the first part, I built a MLP function and test f3 for k^2 and MSE which is test score for test sets and the error distributed, and the next part is joining the f3 and f4 and try to run with MLP.

What's the question I met so far is the memory question. Since the project dataset is huge, which means after I join two tables together, I got a 1351512 rows × 99 columns matrix therefore it can't be run successfully in my virtual environment. Since I also did a feature selection which a method that selects multiple combinations of features and tries to predict the target variable using MLP, my kernel always dead after a long time waiting. But finally, through the Supercomputer in our department, this question has been solved. I just pasted the output graph below this paper.

Let talk about more details in this project. Before I shows with my work, I just simply talk with the data structure in this project. The structure of this program is such that the functions to be used are defined first and called later into the program. First, I define the utility functions

that are called as required through the program. These are followed by regression functions namely, Linear Regression (not my part) and MLP Regressor. Second. The main function is the RegressionPerTarget or Perform Regression Function (Not Great Names) which basically calls the above regression functions and does additional processing like splitting the dataset. Then, the combinatorial feature selection function choses nCk combinations of features for a range of 1 to k features. This gives us the best set of features that will maximize the R^2 or MSE Scores that are used to gauge the performance of regression functions. Five, debugging and join datasets and do linear and multiple layer regression again. R^2 and Mean Square Error (MSE) are two scoring function to find the difference between the actual and predicted values of the test set. For R^2, 1.0 is a perfect fit and for MSE values closer to 0 describe minimum difference.

The first case I did is create a proper training and testing dataset for MLP. Before I do this, I try to see multiple combinations of features and try to predict the target variable using MLP, so I create a feature selection function. In this function, I dropped the target value first and then create a combination function as nCk = n! / (k! * (n-k)!) Where n is the number of features and k is the length of combination. It used for Creating a data frame with the subset of features. After that, I sending the subsets adding the target variable to the Perform Regression Function to find the best set of features.

Next step is preparing for applying MLP. I will pass it into my main function which called “RegressionPerTarget” or Perform Regression Function (Not Great Names) which basically calls the above regression functions and does additional processing like splitting the dataset. “RegressionPerTarget” takes the following parameters:

target : The feature that will be treated as a target variable or variable that has to be predicted

dframe : The dataset after dropped off the useless features

mode = 1: For Debugging, Where the values of all the features except for the input variable are set to 1. The input variable is compared with the target variable 0 or any other number: No Debugging. Features used as is input variable: For Debugging, Where the input variable is a feature that is compared with target variable.

classifier: Name of the classifier to be used. As of now there are only 2 classifiers (here is "linear regression", the next I will talk about later.

The most important part I did is drop the missing value and converted a part of data set to One Hot Encoded Vectors. Handling the missing value is done by set them to mean value because it'll less influence with the output prediction. Converting to One Hot Encoded Vectors is much more complicated, first I take a data frame as an input and splits it into categorical and numerical features. The numerical features are returned as is whereas categorical features are converted to One Hot Encoded Vectors. Through by NumPy dummy function, it's easy to implement it as a final input dataset.

Now, we already have input dataset, I split it as training sets and test sets by using train_test_split function and compute for MLP of the NN network for 30,30,30 in hidden layer which means each level has 30 nodes, learning rate is 0.001 at the beginning and pick the 'adam' solver. Multi-Layer Perceptron Regression is from SKlearn. Takes the train and test dataset and returns the model, scoring functions (r^2 and mse), and predictions.

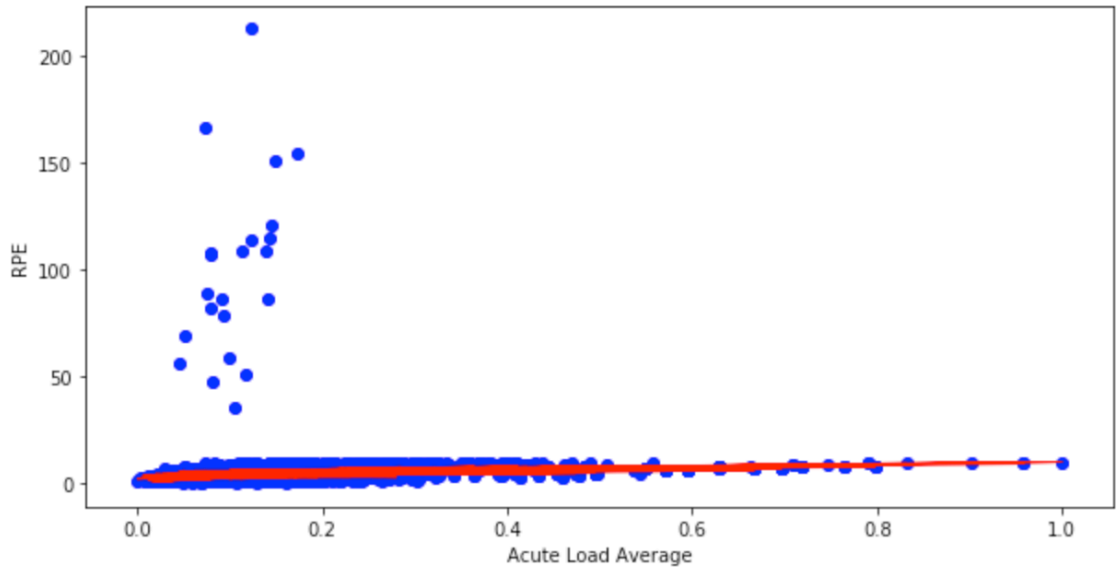
For the output of the MLP for f3 is a dataset which list all possible combinations of features since I applied the function feature selection and their r^2, MSE value. Here is the diagram for

The output after sorting them, I got the best r^2 score almost reach to 0.8 (0 is the worst and 1 is the best in r^2 model).

	Feature Subsets	R^2 Values	MSE Values
99	Duration,Acute Load Average,TSB,Fatigue	0.786664	7.043574
102	Duration,Acute Load Average,Last 14d minimum,Fatigue	0.782924	7.167049
38	Duration,Acute Load Average,Last 14d minimum	0.780924	7.233083
36	Duration,Acute Load Average,Chronic Load Average	0.778836	7.302021
40	Duration,Acute Load Average,Fatigue	0.776871	7.366903
164	Duration,Acute Load Average,Chronic Load Average,TSB,Fatigue	0.776256	7.387213
162	Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum	0.775884	7.399485
42	Duration,Chronic Load Average,TSB	0.773958	7.463085
107	Duration,Chronic Load Average,TSB,Last 14d minimum	0.773801	7.468254
219	Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Fatigue	0.773713	7.471178

Here is the graph for Observing the Actual vs Predicted Values of one input variable and Regression line of single feature over the target variable:

	Acute Load Average	Actual	Predicted
9953	0.069304	0.0	3.642907
13558	0.129349	0.0	3.659794
13560	0.160494	0.0	4.274944
23555	0.069585	0.0	4.465898
23568	0.059203	0.0	3.328897
22344	0.107744	0.0	4.723297
16924	0.049944	0.0	3.014451
13267	0.164422	1.0	4.992137
8010	0.087262	1.0	3.369618
21555	0.019080	1.0	3.463666
...	...	...	...
3734	0.079686	108.0	5.289717
3732	0.138047	109.0	5.839703
3773	0.112233	109.0	5.387435
3728	0.123737	114.0	5.240690
3752	0.143659	115.0	5.671715
3759	0.144220	121.0	6.512820
3754	0.149832	151.0	5.439199
3749	0.173681	155.0	5.951608
3761	0.073232	167.0	5.128489
3764	0.122334	213.0	5.887354



Several Comments over here:

1.When "Weekly Load" was selected as the target feature to predict we got a perfect fit and the predicted values were very close to the actual values. The difference is visualized in Observing the Actual vs Predicted Values.

2.Though, as seen in Analyzing the Regression Function the model overfits. This could be because of various reasons one of them being the test set having low variance and similar values to the training set. Another reason could be a very good linear relationship between the target variable and the other features. This occurs when the target is calculated with a linear function over other features. We could use regularization as a technique to improve overfitting but then the MSE scores would go up as the model learns to generalize. This means it would make incorrect test predictions. We could test that on new a new training set or drop features to remove features with a linear relationship.

3.Feature Elimination: Through recursive feature elimination it was found that only the numerical feature was selected for regression. Though that provides the same fit as the baseline regression

4.. Feature EElimination: The following features were excluded as a test "Overall load","Acute Load Total","Acute Load Average" and the R^2 was reduced to 0.91 whereas the MSE increased to 127008

5. Correction: The above model does not seem to overfit, the graphs above display the accuracy (in terms of MSE) of the regression with the actual data. The low MSE values are the best indicator of the regression model.

So far, I've interpreted what I finished the first part one working in this project, but all above are based on data frame f3. So, the next part is joining them together and apply MLP again. It looks very simply to implement, however, due to the horrible data storing and key for joining is not unique in two tables, it'll create lots of redundant if forcing to join two datasets, and usually waste lots of time on duplicate computation. Although it joined successfully, I must drop some missing values to shortage our dataset as much as possible. In above, I applied one kind of way to handle missing value is setting them to mean value in that particular row, but here I need to create another. What I come up with here is drop the row in f4 when the missing value is beyond 10% of the total after joining them together due to when I reviewed the dataset f4, I found it keeps number of columns larger than f3 and their value is almost null values. So through by dropping value and by project leader said just left the rows where the column name is Session in Period Name, I got a optimize dataset has 1351512 rows × 99 columns.

The next step here is similar as above, separate the features to different kinds of combinations, converted part of dataset to one Hot Encoded Vectors and split it up to training. The best r^2 in the output is approach to 0.8 which is better performance as I expected. Here is diagram for final output before sorted, since the value after sorted is not been copied from supercomputer..

	Feature Subsets	R^2 Values	MSE Values
0	Key	-0.006332	2.223995
1	Session Type	-0.000829	2.211832
2	Duration	0.412768	1.297782
3	Acute Load Average	0.119586	1.945716
4	Chronic Load Average	-0.437262	3.176350
5	TSB	0.139387	1.901954
6	Last 14d minimum	-0.001707	2.213772
7	Date	0.014542	2.177862
8	Training Status	-0.008239	2.228209
9	Key,Session Type	-0.017755	2.249239
...	...	...	...
501	Key,Session Type,Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Date	0.345572	1.446286
502	Key,Session Type,Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Training Status	0.222554	1.718156
503	Key,Session Type,Duration,Acute Load Average,Chronic Load Average,TSB,Date,Training Status	0.135264	1.911066
504	Key,Session Type,Duration,Acute Load Average,Chronic Load Average,Last 14d minimum,Date,Training Status	0.069106	2.057276
505	Key,Session Type,Duration,Acute Load Average,TSB,Last 14d minimum,Date,Training Status	-0.085304	2.398522
506	Key,Session Type,Duration,Chronic Load Average,TSB,Last 14d minimum,Date,Training Status	-0.267535	2.801253
507	Key,Session Type,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Date,Training Status	-0.043436	2.305994
508	Key,Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Date,Training Status	0.064136	2.068259
509	Session Type,Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Date,Training Status	-0.076853	2.379845
510	Key,Session Type,Duration,Acute Load Average,Chronic Load Average,TSB,Last 14d minimum,Date,Training Status	-0.036624	2.290938

Since I actually don't have enough time to optimize the code, I assume it could be some others way to do as:

1. joining datasets: when I join two datasets, I selected a composed key which is not unique in both datasets, so it always creates lots of duplicate what I not expected. Although this composed key is selected by the project leader, if I could find an unique key in datasets, it should be saving the time joining them.

2. Why the MLP for the dataset after joined together can't not be implemented in my virtual environment is due to two reasons. First, numbers of features existed in dataset. I know that the speed for MLP is based on how many rows and columns in datasets rather than only focus on numbers of rows like linear regression. So, for a dataset with 99 columns and 1351512 rows, I assume it spent a long time in repeat training on different training set in particular features. If there is another way to optimize my feature selection function like trying to choose larger r^2 features in feature selection part, the speed should be different. Second, Probably the part handing the missing data when the missing value is beyond 10% is still not most optimize way to handle missing value. If there exists some other way can shortage the numbers of row, the speed also could be different.

As a conclusion, I think the most important thing for a large dataset is optimizing the speed. Anyway, through this project, I understand the actual right ways to use between linear regression and multiply layer regression. Their main different between them is one is computing based on numbers of rows while another focus on both. Linear regression is always a line to classify or cover different output sets while MLP might be not. Since it's a NN network and can be understand by each time when the input value passing to the hidden layer, the output might be no more a linear function. So, this is also why MLP can covered points outside the line.

Steps:

1.build up some utility functions as "normalizeCols" which is a function handle missing value Joinning table and "plot graph" function

2.build up function “RegressionPerTarget” involved all of utility functions. The most important part I did is drop the missing value and converted a part of data set to One Hot Encoded Vectors.
3.build up feature selection for MLP, see multiple combinations of features and tries to predict the target variable using MLP. It applies combination function as nCk = n! / (k! * (n-k)!) Where n is the number of features and k is the length of combination. It used for Creating a data frame with the subset of features.
4.MLP for f3, call fearure selection function first, and it will call “RegressionPerTarget”. The output is a dataset that should be sorted for seek for largest r^2, feature combination and mse
5. plot single input out as Observing the Actual vs Predicted Values and **Ana**yzing the Regression Function.
6. handle missing value by utility function in j4 before Join f3, f4 together by compose key (Key,Session_type).
7.Join f3, f4 together and apply for feature selection, the output is a dataset that should be sorted for seek for largest r^2, feature combination and mse.