FRE 9733 Big Data in Finance Week 6 Homework

Mengheng Xue

April 18, 2019

1 MLP model

1.1 Gradient Decent vs L-BFGS

I use the default settings and compare the performances on *df_dirty2* using *gd* and *l-bfgs*, then obtain that

	rho	distEntropy
Gradient Decent	0.1436	0.0923
L-BFGS	0.1510	0.0633

From the table above, we could see that since rho of gradient decent is smaller than L-BFGS, gradient decent performed better in this case study.

1.1.1 In-sample dataset performance of GD vs L-BFGS

I use hidden layer structure $[2, 2, 2]$ and test the performance of MLP and test it on the in-sample dataset, the results are shown as follows:

	rho	distEntropy
GD	0.2099	0.1189
L-BFGS	0.1989	0.0938

We could see that both GD and L-BFGS are doing poorly of fitting the model and GD is worse. Therefore, instead of having the overfitting problem, MLP with GD or L-BFGS fit the model badly. Also, it is unexpected that both models perform much better on out-sample dirty dataset than in-sample dataset. It shows that even they didn't fit the model very well, but they could have robust performance on dirty dataset.

1.2 Hidden Layers

1.2.1 No. of Hidden Layers

First, we try different number of hidden layers. We use the default settings and use *gd* as *setSolver*, we use the same size of neurons for each layer (=4) and we obtain that

no. of hidden layers	rho	distEntropy
1	0.1436	0.0923
2	0.1466	0.1127
3	0.1418	0.1149
4	0.1420	0.1032

We could see that when neutral network is deeper, the performance is getting better since more hidden layers could help to model the deep complex relationships between inputs and outputs. Based on my experiments, when no. of hidden layers is 3, it achieves the minimum rho (in blue). However, it doesn't mean the deeper of neutral network, the better performance we will get, since too many hidden layers may cause the overfitting problem.

1.2.2 No. of Neurons in Hidden Layers

Then I try to change number of neurons in hidden layers. I choose 3 hidden layers. The results are as follows:

no. of neurons in each hidden layer	rho	distEntropy
[2, 2, 2]	0.1426	0.1201
[4, 4, 4]	0.1418	0.1149
[6, 6, 6]	0.1414	0.1034
[8, 8, 8]	0.1419	0.1048

We can see that when we expand neurons in each hidden layer, the more neurons we apply, the better the performance of our model. This result is expected for the the same reason as increasing number of hidden layers. The more complex neutral network will have more powerful prediction ability. However, it doesn't mean we could increase neuron numbers infinitely, which will have the overfitting problem as well. That's why when 3 layers all have 6 neurons, we will achieve the best performance (in blue).

1.3 Remove Hidden Layers

Then I remove all hidden layers in MPL Model and remove $setRegParam()$ and $setElasticNetParam()$ in LR model, and compare the performances of MLP and LR, we obtain that

	rho	distEntropy
MLP with no hidden layer	0.1802	0.1687
simple LR	0.2213	0.0273

It is not what I expected, since I thought neutral network with no hidden layer is the same as logistic regression. MLP did better than LR. My guess is Spark's MLlib may have different default parameters for LR and MLP modules which make their performances different. MLP is not monotonic in the parameters, indicating that it has more than just the (regressors) $\times$ (states) parameters, even with no hidden layers.

1.4 MLP vs LR

1.4.1 Performance Comparsion

Comparing my current best MLP (hidden layers: $[6, 6, 6]$, solver: gd) and LR (RegParam=0.002, ElasticNetParam=0.008) model, I obtain the performance as follows:

	rho	distEntropy
best MLP	0.1414	0.1034
best LR	0.1379	0.0790

We could see that LR is better than MLP. The reason is that although MLP is a stronger model, it could learn the hidden relationship between inputs and outputs, thus reduce the bias of prediction, it may suffer from the overfitting problem. Therefore, on the dirty dataset, the prediction of MLP will have high variance based on variance-bias tradeoff.

1.4.2 Time Comparison

The sample size I am using to fit both models are 100k data. The amount of time needed for LR and MLP model fitting are shown as follows.

	time
best MLP	
best LR	

Due to the Windows system issue, println() function couldn't work properly on Spark, but we could expect that MLP model would take much longer time than LR model.

1.4.3 Conclusion

Based on the above discussion, we could see that LR model has better prediction performance, also the model fitting time is much less than MLP. Therefore, we could say that in our case study, LR is a better model than MLP.