Computer-Aided Molecular Binding Affinity Prediction

ESTR3108 Project Report

LIU, Zhanhao
1155092201

1 Introduction

Molecular Docking is a computational technique that aims to predict whether and how a particular small molecule will stably bind to a target protein. It is an important component of many drug discovery projects when the structure of the protein is available.[1]

As an alternative to modeling assumptions in scoring functions, non-parametric machine learning can be used to implicitly capture binding effects that are hard to model explicitly. So in this project, we want to explore some machine learning methods(Multiple Linear Regression, Random Forest, Deep Neural Network...) on different tools(R, Tensorflow) introduced in the class and use these machine learning methods to tackle the prediction problem.

In the project, we are adopting the widely-used quality criteria of PDBbind database(v2013 Benchmark).[2]

2 Performance Metrics

The prediction performance of the binding affinity was quantified through root mean square error RMSE, standard deviation SD in linear correlation, Pearson correlation coefficient Rp, and Spearman correlation coefficient Rs between the measured and predicted binding affinities of the test set complexes:

RMSE Root Mean Square Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)} - p^{(n)}\right)^2}$$

SD Standard Deviation in linear correlation

$$SD = \sqrt{\frac{1}{N-2}\sum_{n=1}^{N}\left(y^{(n)} - \overline{p}^{(n)}\right)^2}$$

Rp Pearson correlation coefficient

$$R_p = \frac{N\sum_{n=1}^{N} p^n y^n - \sum_{n=1}^{N} p^n \sum_{n=1}^{N} y^n}{\sqrt{\left(N\sum_{n=1}^{n}\left(p^n\right)^2 - \left(\sum_{n=1}^{N} p^n\right)^2\right)\left(N\sum_{n=1}^{N}\left(y^n\right)^2 - \left(\sum_{n=1}^{N} y^n\right)^2\right)}}$$

Rs Spearman correlation coefficient

$$R_p = \frac{N\sum_{n=1}^{N} p_r^n y_r^n - \sum_{n=1}^{N} p_r^n \sum_{n=1}^{N} y_r^n}{\sqrt{\left(N\sum_{n=1}^{n}(p_r^n)^2 - \left(\sum_{n=1}^{N} p_r^n\right)^2\right)\left(N\sum_{n=1}^{N}(y_r^n)^2 - \left(\sum_{n=1}^{N} y_r^n\right)^2\right)}}$$

where N is the number of complexes in the set, a and b are the intercept and coefficient of the linear correlation between $\{p^n\}_{n=1}^N$ and $\{y^n\}_{n=1}^N$ on the test set, whereas $\{p_r^n\}_{n=1}^N$ and $\{y_r^n\}_{n=1}^N$ are the rankings of $\{p^n\}_{n=1}^N$ and $\{y^n\}_{n=1}^N$, respectively.

Therefore, lower values of RMSE and SD and higher values of Rp and Rs indicate better prediction performance.[3]

3 Data partition

- PDBbind v2013 Benchmark
- Refined set (N=2959)
- 5-fold cross validation, Round-robin scheduling
- 1 fold for testing and 4 folds for training

Table 1: Round-robin scheduling

Partition	N	Lowest pKd	Highest pKd
1	592	2	11.74
2	592	2	11.80
3	592	2	11.85
4	592	2	11.92
5	591	2.05	11.72

Throughout the project, I used the 5-fold cross validation for prediction and take the average of the performance metrics as result.

4 Regression Approaches

4.1 Multiple Linear Regression

Multiple linear regression is nearly the simplest regression model in machine learning, it models the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. So in my project, I first implemented an MLR model in both R and Tensorflow to use the four features in Cyscore as explanatory variables and the measured binding affinity as the response variable for prediction. Here is my result:

#	N	RMSE	SD	Rp	Rs
1	592	1.66	1.66	0.560	0.555
2	592	1.62	1.62	0.589	0.600
3	592	1.69	1.70	0.531	0.529
4	592	1.68	1.68	0.542	0.557
5	591	1.65	1.65	0.559	0.553
avg		1.66	1.66	0.556	0.559

Table 2: MLR::Cyscore

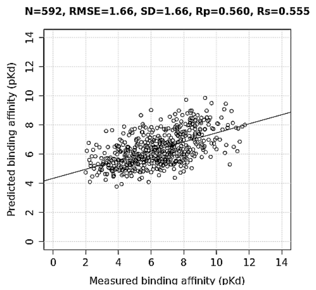


Figure 1: MLR::Cyscore(Sample 1)

My result is consistent with the reference result given by Dr. Hongjian Li, which means that I am on the right track.

4.2 Random Forest

Random Forests is competitive with the best-known machine learning methods, it is an ensemble learning method for both classification and regression. It has high accuracy and stability for classification and regression. Since my supervisor Dr. Hongjian Li's previous work on this project was mainly using the Random Forests regression method for predicting the binding affinity, I first use the tool RF-Score[4], which is designed by him to build a Random Forest model to have a try and help me get familiar with this project.

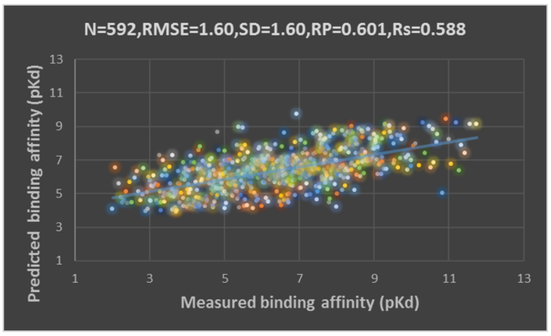
After getting a consistent result of Dr. Hongjian Li, I used R to build a random forest model and got the following results using the Cyscore features and Cyscore + Vina features:

Table 3: RF::Cyscore

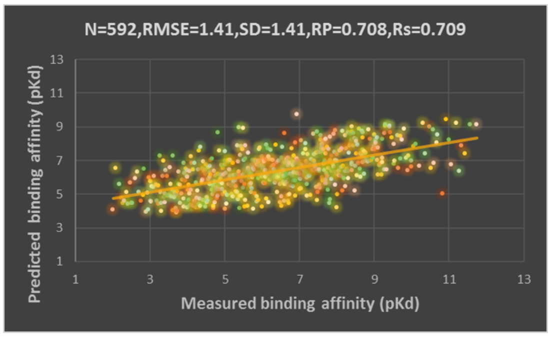
#	N	RMSE	SD	Rp	Rs
1	592	1.60	1.60	0.601	0.588
2	592	1.51	1.51	0.657	0.641
3	592	1.66	1.66	0.561	0.545
4	592	1.63	1.63	0.580	0.576
5	591	1.57	1.57	0.615	0.586
avg		1.59	1.59	0.603	0.587

Table 4: RF::CyscoreVina

#	N	RMSE	SD	Rp	Rs
1	592	1.41	1.41	0.708	0.709
2	592	1.38	1.37	0.730	0.725
3	592	1.49	1.49	0.668	0.665
4	592	1.51	1.51	0.657	0.661
5	591	1.42	1.42	0.701	0.692
avg		1.44	1.44	0.693	0.690



(a) RF::Cyscore(sample 1)



(b) RF::CyscoreVina(sample 1)

Figure 2: Comparison between RF::Cyscore and RF::CyscoreVina

We can see that Random Forest gives a better prediction performance than Multiple Linear Regression, and using more features as explanatory variables can also improve the prediction accuracy.

Tune the RF model

mtry The mtry parameter is the number of data in each decision tree used for training in the random forest. It is one of the most important parameters in a random forest model, so I try to tune my RF model by finding the optimal mtry value. I started with the default value of mtry, searched for the optimal value with respect to Out-of-Bag(OOB) error[1] estimation of mtry for my RF model.

[1]Random forests technique involves sampling of the input data with replacement (bootstrap sampling). In this sampling, about one-third of the data is not used for training and can be used to testing.These are called the out of bag samples. Error estimated on these out of bag samples is the out of bag error.

Here is my result for finding the optimal mtry parameter:

mtry	OOB Error	mtry	OOB Error
1	2.157958	6	2.131094
2	2.132698	7	2.140175
3	2.131236	8	2.128880
4	2.119785	9	2.119842
5	2.127408	10	2.135400

Table 5: Tune mtry
(CyscoreVina, improve=1e-5, ntree=500)

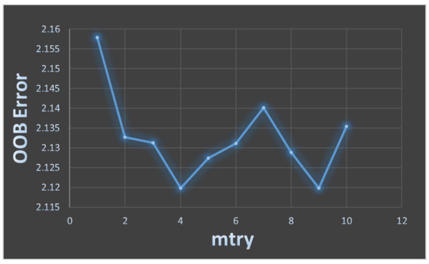


Figure 3: MLR::Cyscore(Sample 1)

We can easily find that when mtry equals 4, the OOB Error is minimized. So I fix other parameter and using different mtry values to repeat the prediction process using the CyscoreVina features and get the following results:

mtry	#	RMSE	SD	Rp	Rs	OOB Error
2	592	1.41	1.41	0.708	0.707	2.132698
4	592	1.41	1.41	0.708	0.708	2.119785
6	592	1.41	1.41	0.708	0.708	2.131094
8	592	1.41	1.41	0.708	0.706	2.12888
10	592	1.41	1.41	0.707	0.705	2.1354
avg		1.41	1.41	0.708	0.707	2.129571

Table 6: RF models using different mtry values (ntree=500)

When mtry equals 4, the prediction performance is the best.

ntree Another important parameter in the Random Forest model is ntree. The ntree parameter is the number of decision trees in the random forest. To tune my RF model by finding the optimal ntree value, I fix the mtry value as 4, searched for the optimal value of ntree with respect to performance metrics.

Here is my result for finding the optimal ntree parameter:

ntree	#	RMSE	SD	Rp	Rs	OOB Error
30	592	1.45	1.45	0.69	0.691	2.337980
60	592	1.42	1.42	0.702	0.702	2.227404
125	592	1.42	1.42	0.703	0.703	2.205851
250	592	1.41	1.41	0.708	0.709	2.135712
500	592	1.41	1.41	0.708	0.708	2.123670
avg		1.42	1.42	0.702	0.703	2.206123

Table 7: RF models using different ntree values (mtry=4)

It can be seen that when ntree is small(30-60), increasing it can increase the RF model's performance. But when ntree is big enough(60-500), increasing it cannot increase the RF model's performance very much.

4.3 Support Vector Machine

Support Vector machines(SVM) is a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis. It has the advantages of global minima and well generalization ability. So I built an SVM model based on the package 'e1071' of R and get the following prediction result using the Cyscore features:

#	N	RMSE	SD	Rp	Rs
1	592	1.65	1.64	0.570	0.559
2	592	1.56	1.56	0.626	0.612
3	592	1.69	1.68	0.544	0.529
4	592	1.64	1.64	0.572	0.567
5	591	1.61	1.61	0.590	0.577
avg		1.63	1.63	0.580	0.569

Table 8: SVM::Cyscore

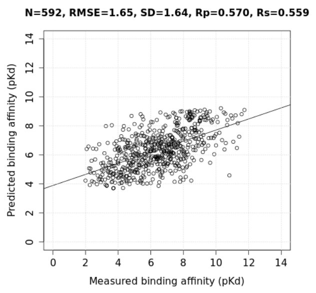


Figure 4: SVM::Cyscore(sample 1)

It can be seen that the result given by SVM is slightly better than the MLR's. However, it still cannot beat Random Forest in all the performance metrics.

4.4 Deep Neural Network

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers of units between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships. Since this machine learning is so widely used nowadays and it is one of the most significant parts of this course, I built a DNN model and tried many methods to tune it for the prediction of this project. Here are some of my results using different number of hidden layers and nodes in each layer(all the training steps are 2000):

#	N	RMSE	SD	Rp	Rs
1	592	1.64	1.64	0.570	0.561
2	592	1.59	1.59	0.605	0.607
3	592	1.68	1.68	0.542	0.538
4	592	1.66	1.65	0.564	0.565
5	591	1.63	1.64	0.571	0.561
avg		1.64	1.64	0.570	0.566

Table 9: RF::Cyscore
hidden layers: [10 20 10]

#	N	RMSE	SD	Rp	Rs
1	592	1.65	1.65	0.567	0.560
2	592	1.59	1.59	0.609	0.611
3	592	1.68	1.68	0.541	0.537
4	592	1.66	1.66	0.557	0.561
5	591	1.63	1.63	0.574	0.563
avg		1.64	1.64	0.570	0.566

Table 10: RF::Cyscore
hidden layers: [20 40 20]

#	N	RMSE	SD	Rp	Rs
1	592	1.64	1.64	0.570	0.563
2	592	1.59	1.59	0.608	0.606
3	592	1.68	1.68	0.546	0.544
4	592	1.65	1.65	0.564	0.566
5	591	1.64	1.64	0.569	0.559
avg		1.64	1.64	0.571	0.568

Table 11: RF::Cyscore
hidden layers: [10 20 10 20]

#	N	RMSE	SD	Rp	Rs
1	592	1.64	1.64	0.571	0.563
2	592	1.59	1.59	0.604	0.607
3	592	1.68	1.68	0.545	0.538
4	592	1.66	1.65	0.563	0.563
5	591	1.63	1.63	0.573	0.562
avg		1.64	1.64	0.571	0.567

Table 12: RF::Cyscore
hidden layers: [10 20 10 20 10]

It can be seen that the result given by DNN is slightly better than MLR's, but it is also slightly worse than SVM's. Also, changing the number of hidden layers and nodes in each layer cannot improve the prediction performance very much.

5 Classification Approaches

Since we learn about both Classification and Regression method in the class, I want to change this regression problem into a classification problem to see if it can give a high prediction accuracy.

5.1 2-type Classification

I first divide all the samples into two types according to their binding affinity[2]:

- Type 1: $Pbindaff <= 6.5$ (Weak binding)
- Type 2: $Pbindaff > 6.5$ (Strong binding)

I then use a 4 layer DNN Classifier with 10, 20, 10, 20 units in each layer respectively, to classify all the samples and get the following accuracy(training steps = 2000):

$$Accuracy = 0.682432$$

5.2 3-type Classification

Using the similar method to divide all the samples into 3 types:

- Type 1: $Pbindaff <= 5.2$ (Weak binding)
- Type 2: $5.2 < Pbindaff <= 8.5$ (Mild binding)
- Type 3: $Pbindaff > 8.5$ (Strong binding)

I then use a 4 layer DNN Classifier with 10, 20, 10, 20 units in each layer respectively, to classify all the samples and get the following accuracy(training steps = 2000):

$$Accuracy = 0.616554$$

5.3 4-type Classification

Using the similar method to divide all the samples into 3 types:

- Type 1: $Pbindaff <= 4.4$ (Weak binding)
- Type 2: $4.4 < Pbindaff <= 6.9$ (Mild binding)
- Type 3: $6.9 < Pbindaff <= 9.3$ (Slightly strong binding)
- Type 4: $Pbindaff > 9.3$ (Strong binding)

I then use a 4 layer DNN Classifier with 10, 20, 10, 20 units in each layer respectively, to classify all the samples and get the following accuracy(training steps = 2000):

$$Accuracy = 0.508446$$

To conclude, I think the accuracies of these three DNN Classification model is not so bad, but this may not be very useful because the purpose of this project is to accurately predict the binding affinity of the complex.

[2]6.5 is the average of the max binding affinity(11.72) and the minimum binding affinity(2) among all the samples

6 Results and Conclusion

1. In the Random Forest model:
   - The mtry value of smallest OOB Error is optimal.
   - When ntree value is small, increasing it can increase the RF model's performance. But when ntree is big enough, increasing it cannot increase the RF model's performance very much.

2. Here is the comparison between different machine learning methods performance metrics(average):

#	RMSE	SD	Rp	Rs
MLR::Cyscore	1.66	1.66	0.556	0.559
RF::Cyscore	1.59	1.59	0.602	0.586
SVM::Cyscore	1.63	1.63	0.580	0.569
DNN::Cyscore ([10 20 10])	1.64	1.64	0.570	0.566
DNN::Cyscore ([20 40 20])	1.64	1.64	0.570	0.566
DNN::Cyscore ([10 20 10 20])	1.64	1.64	0.571	0.568
DNN::Cyscore ([10 20 10 20 10])	1.64	1.64	0.571	0.567

Table 13: Comparision of the performance metrics between different machine learning methods

   It is not hard to find out that Random Forest regression model gives the best prediction performance within all these machine learning methods, and Multiple Linear Regression's performance is the worst.

3. Classification model can give a preferable result of prediction accuracy, but this may not be very helpful in this problem since the purpose of this project is to accurately predict the binding affinity of the protein-ligand complex.

7 Further Prospects

- Using the high-performance computer to train the DNN Regression model for more layers and more nodes in each layer for more training steps may improve the prediction performance.
- Using other machine learning classification models, such as Random Forest and Naive Bayes, may improve the prediction accuracy of classifying the samples with different binding affinity.
- The data Dr. Hongjian Li used in his past research was derived by the Cyscore v1.1.4, and now the author of Cyscore publishes the Cyscore v2.0.0 and improves the hydrophobic free energy calculation. If we can derive a new set of features of the PDBbind v2015 refined set, we may have a chance to improve the prediction performance.

References

[1] P. Ballester and J. Mitchell, "A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking", Bioinformatics, vol. 26, no. 9, pp. 1169-1175, 2010.

[2] PDBbind database, "PDBbind database", pdbbind.org.cn, 2016. [Online]. Available: http://www.pdbbind.org.cn. [Accessed: 29- Nov- 2016].

[3] H. Li, K. Leung, M. Wong and P. Ballester, "Low-Quality Structural and Interaction Data Improves Binding Affinity Prediction via Random Forest", Molecules, vol. 20, no. 6, pp. 10947-10962, 2015.

[4] H. Li, "RF-Score", GitHub, 2016. [Online]. Available: https://github.com/HongjianLi/RF-Score. [Accessed: 30- Nov- 2016].