Drug Property Prediction

Background

Molecular property prediction is one of the major cheminformatics tasks. In light of the development of neural machinery, trained property predictors can screen potential compounds with antiviral activity against various virus[1].

At the end of 2019, a new type of coronavirus named COVID-19 was found and it resulted in an ongoing pandemic. According to the data from the Johns Hopkins Coronavirus Resource Center, the outbreak of COVID-19 has infected over 10 million individuals and caused over 500 thousand death worldwide . Trained predictors provide critical information and help scientists focus on high possible compounds, saving invaluable time and human lives.

Review of Existing Methods

Traditional models of drug property prediction requires molecular descriptors as inputs, whose selection impacts the performance of models to a large extent. Deep-learning methods and models can avoid time-consuming feature engineering and learn the features automatically from the raw representation or drugs. Therefore, most of recent researches on drug property prediction are based on deep-learning methods. At present, there are two main technical routes for the study of drug property prediction with deep-learning methods, one of which is to learn the features from simplified molecular input line entry specification (SMILES) expressions using sequence to sequence learning methods, and the other is to learn the features from molecular graphs using graph convolutional network (GCN).

The first route involves models which can learn the features from SMILES expressions, thus models based on recurrent neural network (RNN), word2vec and seq2seq are widely adopted and studied. Zheng et al [2] combine bidirectional long short-term memory (BiLSTM) unit and self-attention mechanism to automatically locate the features and learn the representation vectors from SMILES expressions and achieve high accuracy in predicting drug properties. Jaeger et al [3] propose mol2vec method, which is based on word2vec and regard drug molecules as sequences generated by chemical substructures. Representation vectors learned by mol2vec is superior to manually designed molecular descriptors in solubility and toxicity prediction tasks.

The second route begins with the study of neural fingerprint (NFP) by Duvenaud et al [4]. Duvenaud used GCN to learn the representation vector of drug molecules with translation invariance. However, the end-to-end learning of NFP has the disadvantage of high complexity and time-consuming, which limits its practical application.

Algorithm Analysis and Code Summary

In this task, we adopt SABiLSTM model with mol2vec method and random forest (RF) model to predict activity of molecules from SMILES expressions. In this section, we'll first analysis our algorithms and summarize key codes.

Algorithm Analysis

The SABiLSTM model is based on [2] and takes advantages of [3]. It uses a pre-trained mol2vec model [3] to convert input SMILES expressions to feature tensors. Each character of the SMILES expression will be embedded into a high dimension vector by the mol2vec model and the representation vectors will be stacked up and passed to the BiLSTM units. Different self attentions will be randomly initialized. Then output of BiLSTM units will be scored and multiplied by itself according to attentions' weights and passed throw a fully connected layer to score the activity of molecules.

In the SABiLSTM model, the representation of molecules is not a feature vector, but a feature matrix who has same number of rows as the length of SMILES expression, in order to utilize the structural and spatial information of the SMILES expression. The self-attention mechanism is introduced into this model by scoring the output of BiLSTM units and do a matrix multiplication operation between the scores and the BiLSTM output. The self-mechanism can be interpreted as focusing on a certain part of molecule's impact on the whole molecule's activity, which is consistent with our chemical knowledge. Moreover, the fully connected layer will do a weighted sum over the output of self-attention layer, which is accords with our impression that the impact of different part of molecule on its property can be sum up. Above all, we believe that our SABiLSTM model has reasonable chemical implications and it does work well according to various criteria. The training of SABiLSTM model is not fast enough and requires GPU device, result from its complex structure and large amount of variables, but the complexity of model can be limited by decreasing the number of multiple attentions and the dimension of attention and then the training process can be accelerated. The structure of SABiLSTM model is shown below(Figure 1).

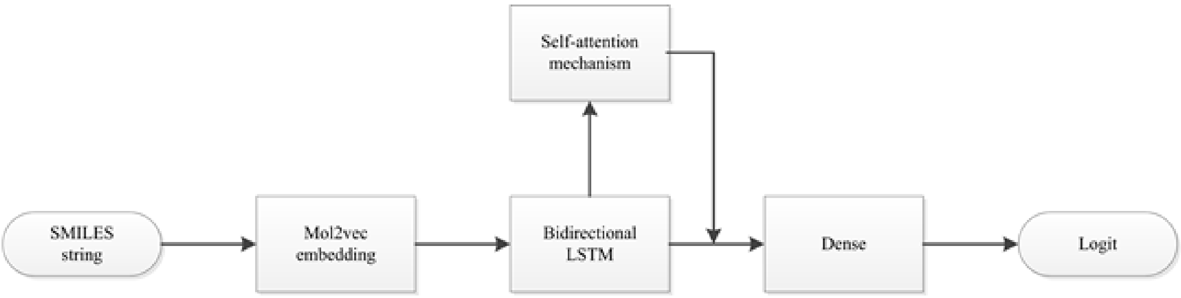

Figure 1. Structure of SABiLSTM Model

The RF model is much simpler than SABiLSTM model. It accepts feature vectors as inputs and relies on decision trees to score molecules.

In this model, decision trees will take part of training embeddings as inputs and use out of bag samples to score the selected features and select the best feature combination. The training process of RF model almost completes in an instant, but its performance fluctuates to a great extent. This is because the model over-fits on the training set and performs badly on the testing set. To reduce model's overfitting, we use *max_depth* and *oob_score* option to pre-prune and select valuable features. The structure of SABiLSTM model is shown below(Figure 2).

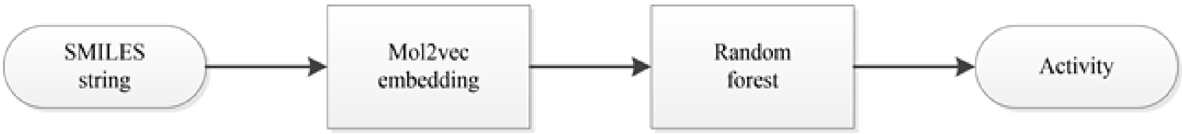

Figure 2. Structure of Random Forest Model

Code Summary

The key code structure is listed below. To run our code, one needs to install the GPU version of tensorflow2 and other python packages include sklearn, mol2vec, RDKit and gensim. If you can't perfectly install mol2vec, just put it under the handout directory and make a minor change in handout/data.py according to comment.

PRML final tasking

- __init__.py
- mol2vec
- source.py
- 10foldtest.py
- handout/
  - model.py
  - data.py
  - model_300dim.pkl
  - __init__.py

Mol2vec is just a external library which we use in this project.

source.py provides user with various command line options which allow user to train, evaluate and cross validate different models. For example, one can enter "python source.py —model SABiLSTM —train True" in command line to train a SABiLSTM model. Below are some available command line options provided by source.py.

10foldtest.py provides user a simple way to train and evaluate the models we build on 10-fold dataset . Users can simply run this file by entering "python 10foldtest.py" in command line and get the results of RF model by default on 10foldtest. Users can also modify this file by calling *train_lstm* and *train_ensemble* instead of *train_RF* by default. It might take hours to train SABiLSTM model. Before calling *train_ensemble* function, user must call *train_lstm* first.

handout/data.py includes functions that are used to handle SMILES expressions. *embedData2Vector* embed SMILES strings in a 300-dimension vector using the pre-trained mol2vec model loaded from handout/model_300dim.pkl or other specified pre-trained model path. *embedData2Tensor* embed SMILES strings in a n-characters*300-dimension feature tensor using the pre-trained mol2vec model. The difference between these two functions is that *embedData2Tensor* stack up the word vector generated by mol2vec model while *embedData2Vector* sum them up. g*enVectorFor10Folds* calls *embedData2Vector* to embed SMILES strings used for 10 folds cross validation and g*enTensorFor10Folds* has the same function. *generateBalancedData* makes the embedding result balanced according to the expected rate of valid samples, which can accelerate the training process of SABiLSTM model.

handout/model.py includes our SABiLSTM and RF model. *SABiLSTMModel* consists of a *LSTM* layer, a custom *selfAttention* layer and a *Dense* layer. The *selfAttention* layer has multiple attentions and enables tensorflow gradient tape to trace the gradients and apply them to trainable variables of the SABiLSTM model. To train the SABiLSTM model, one needs to call *trainSABiLSTMModel* and the function will train and return a model. The training requires a GPU device. Other functions begin with the word *train* train the corresponding model and functions begins with the word *evaluate* evaluate the trained model and save the ROC-AUC and PRC-AUC of model as pictures.

Results and Discussion

In this part, we analyze our models‘ performance in E.coli dataset and 10-fold dataset. Meanwhile, ensemble methods aiming at optimizing our models' performance are discussed. We use area under the receiver operating characteristic curve (ROC-AUC) and area under the precision recall curve (PRC-AUC) as metrics.

An **RO curve** is a graph showing the performance of a classification model at all classification thresholds. It applies True Positive Rate(TPR) as y axis and False Positive Rate(FPR) as x axis. Definitions of TPR and FPR are as following:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Compared to ROC curve, **PR curve** focuses on the minority class, avoiding excessively optimistic views of the performance of imbalanced classification models. The x-axis indicates recall rate and y-axis indicates precision rate. Definitions of recall rate and precision rate are as following:

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

E.coli **dataset** contains a total of 2336 molecules, 121 of which are tagged positive(about 5.2%). The 10-fold dataset randomly chooses a certain size of molecules as train(1695, 80%), validation(202, 10%) and test(203, 10%) dataset and repeats 10 times. The number of total active molecules is only 48(2.3%), much lower than E.coli. The shrinking size and increased imbalance of dataset lead to a more challenging task compared with E.coli dataset.

SABiLSTM Model

First, we build SABiLSTM model and continuously optimize our model on E.coli dataset until it achieves solid performance. ROC-AUC reaches 0.933 and PRC-AUC reaches 0.488.

The RO curve and PR curve are presented in Figure 3.

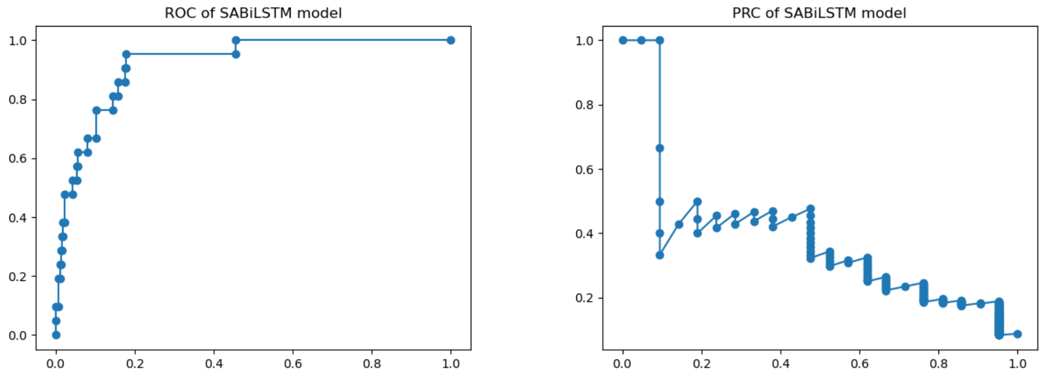

Figure 3. ROC and PRC of SABiLSTM model on E.coli dataset

Then we test the model on both E.coli and 10-fold dataset. Results are as follows:

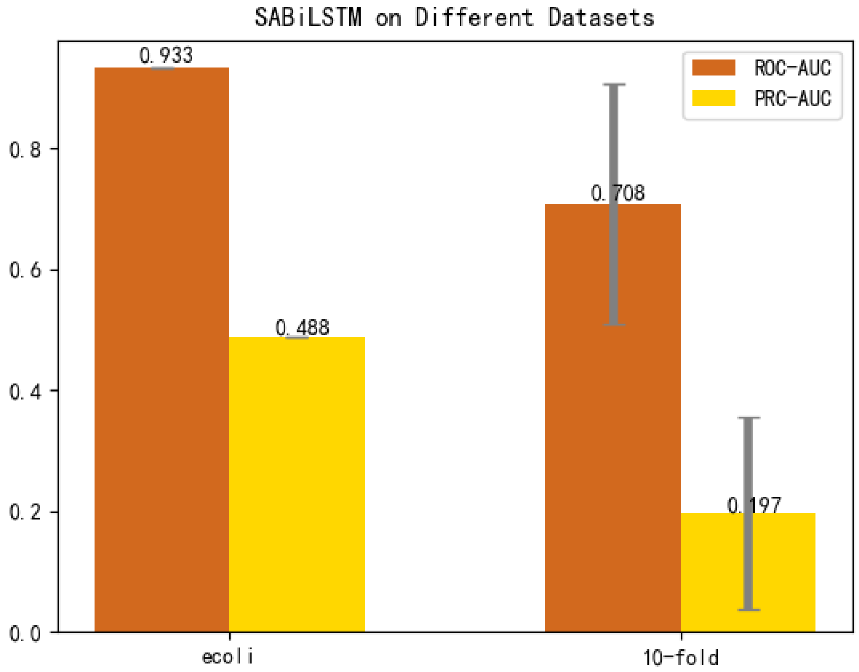

Figure 4. ROC-AUC and PRC-AUC performance of SABiLSTM model on E.coli and 10-fold dataset

SABiLSTM performs much worse on 10-fold dataset, meanwhile, the standard errors are nonnegligible. To clarify what contributes to the poor performance, we analyze results on each fold(Table 1).

Fold	ROC-AUC	PRC-AUC
0	0.496	0.127
1	0.639	0.453
2	0.756	0.516
3	0.375	0.015
4	0.891	0.387
5	0.859	0.381
6	0.585	0.068
7	0.401	0.004
8	0.967	0.216
9	0.906	0.402

Table 1 SABiLSTM model on 10-fold Dataset

According to the table above, it seems SABiLSTM model faces great challenges in some folds, for example, fold 0, 3, 6 and 7. We suppose these folds are relatively harder due to 2 possible reasons: 1. Active molecules in training set share little common features with those in test set in these folds ; 2. Our model are incapable of handling with certain molecules in test set of these folds.

Other Models

To justify our supposes, we introduce other classifiers including Random Forest(**RF**) and simple Logistic Regression(**LR**) classifier. Performances of these models on 10-fold dataset are collected as below(parameters of models have been optimized).

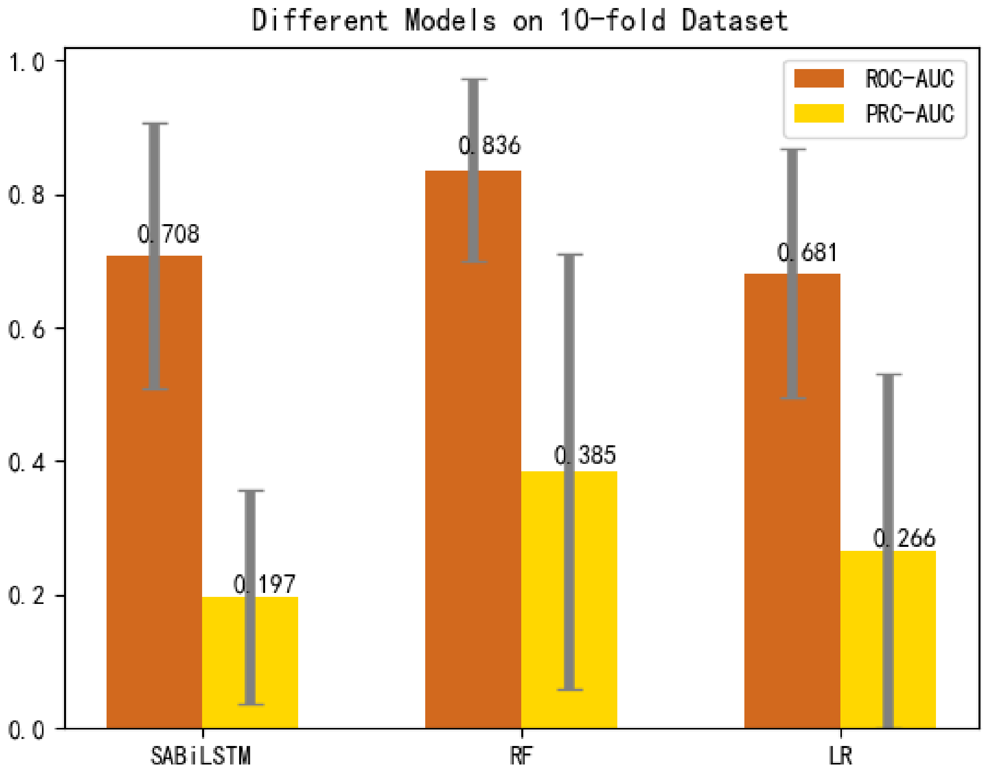

Figure 5. performance of three models on 10-fold dataset

The performance of SABiLSTM model is similar with LR, while RF outperforms other models. All three models find challenging on certain folds: folds that both RF and LR perform terribly on are 3, 6, 7 and 9 while SABiLSTM does not perform well on fold 0, 3, 6 and 7, indicating that some folds are harder than the others for all models.

However, differences between simple classifiers and our SABiLSTM model emerge via further study. We find that while RF outperforms SABiLSTM, RF neglects more active molecules than other models. SABiLSTM, nevertheless, has the lowest false negative count(Table 2).

Models	Total False Positive Count	Total False Negative Count
SABiLSTM	252	23
RF	0	31
LR	80	27

Table 2 total false positive and false negative counts of three models on 10-fold dataset

Ensemble

Considering the distinction between three models mentioned above and the fact that none of the models can reach satisfactory performance, we adopt ensemble method to achieve better performance. One simple solution we come up with is taking the average of possibility predicted by three models as the final possibility. We adopt this strategy and attain the following results(Figure 6).

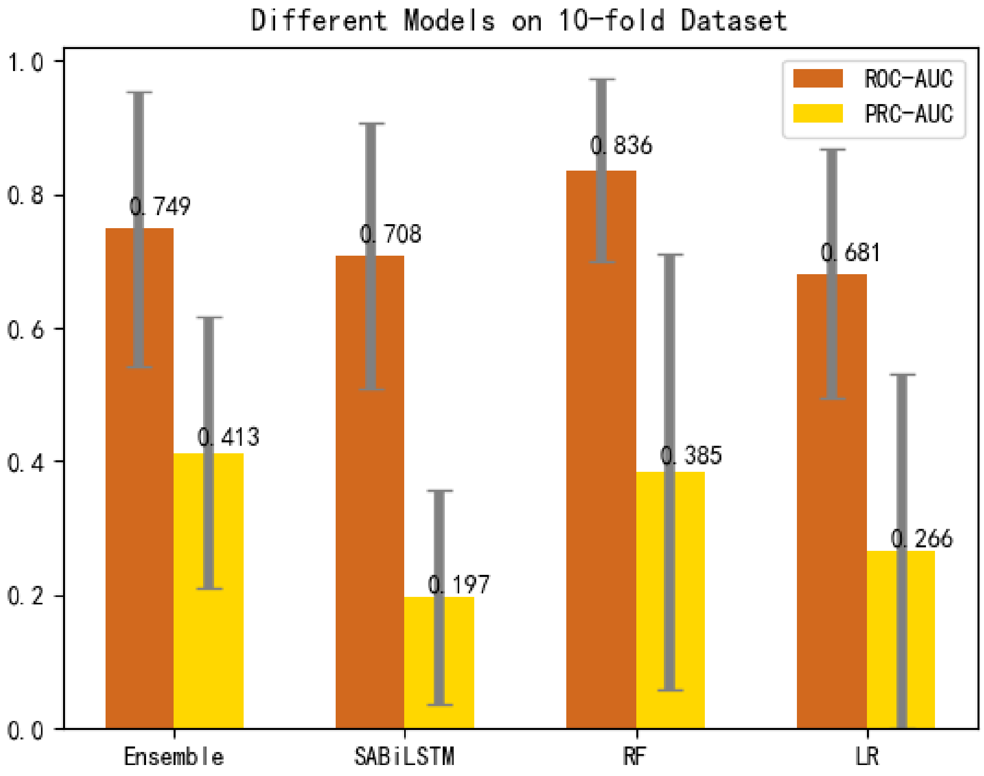

Figure 6. Ensemble model and single models

PRC-AUC of ensemble model ranks top among all models, however, the ROC-AUC of ensemble model is much worse than RF model. FP and FN counts are as following:

Models	Total False Positive Count	Total False Negative Count
RF	0	31
Ensemble	32	26

Table 3 total false positive and false negative counts of RF model and Ensemble model on 10-fold dataset

Table 3 suggests that when other models are introduced, although the model is able to predict more active molecules, it suffers from distraction and makes more mistakes in total which damage its performance at the same time. To optimize the ensemble strategy, we come up with two methods. One is endowing RF model more weight to avoid excessive distraction(**Ensemble1**). The other is only focusing on negative molecules RF model predicted and enhancing the possibility of molecules which LR model or SABiLSTM model give positive predictions, thus, these negative molecules may flip to positive molecules(**Ensemble2**). Comparison of different ensemble strategies is shown in Figure 7.

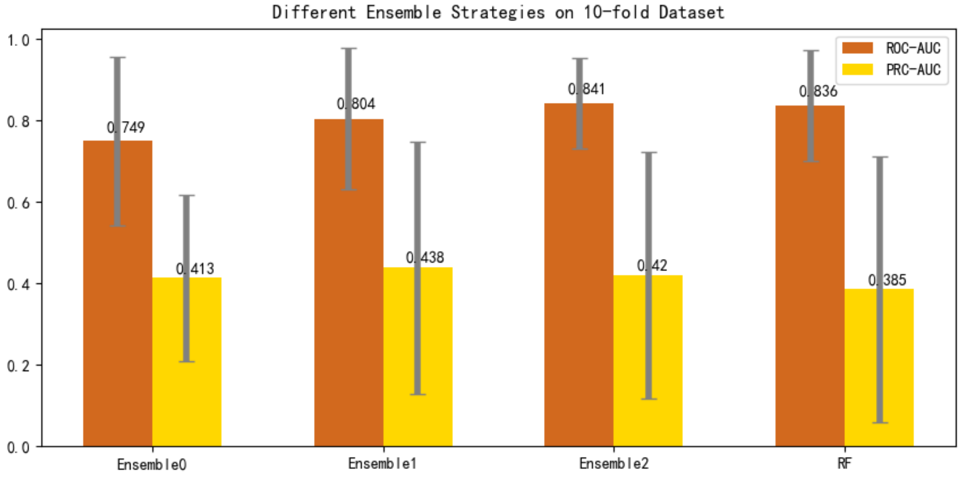

Figure 7. Ensemble0 for original ensemble strategy; Enemble1 and Ensemble2 are two optimizing strategies mentioned above.

All these ensemble models have higher PRC-AUC than RF model while Enemble1 and Ensemble2 are better than Ensemble0 in both ROC-AUC and PRC-AUC. Ensemble1 shows the highest PRC-AUC and Ensemble2 outperforms the RF model in both ROC-AUC and PRC-AUC. Solid performance of Ensemble2 may because it protects accuracy of RF model on predicting negative molecules from disturbance and avoids overmuch caution when predicting positive molecules.

References

[1] Yang, Kevin, et al. "Analyzing learned molecular representations for property prediction." *Journal of chemical information and modeling* 59.8 (2019): 3370-3388.

[2] Zheng Shuangjia, Yan Xin, Yang Yuedong, et al. Identifying structure-property relationships through SMILES syntax analysis with self-attention mechanism [J]. Journal of Chemical Information and Modelling, 2019, 59 (2): 914-923.

[3] Jaeger Sabrina, Fulle Simone, Turk Samo. Mol2vec: Unsupervised machine learning approach with chemical intuition [J]. Journal of Chemical Information and Modelling, 2018, 58(1): 27-35.

[4] Duvenaud D K, Maclaurin D, Aguilera-Iparraguirre J, et al. Convolutional networks on graphs for learning molecular fingerprints [C]. Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge, USA:ACM, 2015:2224-2232.