Protein ligand binding prediction by neural networks and gradient boosted trees ensemble

Mehmet Tan TOBB University of Economics and Technology Dept. of Computer Engineering Ankara, Turkey

Abstract

This document explains the basic methodology followed for our submission in the IDG-DREAM Drug-Kinase Binding Prediction Challenge. A total of 394 kinase binding predictions were required in this challenge. Binding data only from Drug Target Commons database is exploited. However, three other databases were used for the representation of chemicals and proteins. Our submission has achieved a final RMSE score of 1.113 in round 2.

Introduction

Protein ligand binding score prediction is a fundamental issue in drug discovery. It has a great potential to decrease the large cost of the drug discovery process. In this DREAM challenge, the binding scores of the protein-chemical pairs were to be predicted by exploiting available knowledge in binding databases.

The first database to be used is the Drug Target Commons (DTC) database [1]. It has a large number of protein ligand binding data in terms of different metrics such as the IC50, Ki and Kd. DTC is the only database we used for binding data. In addition to this database, we used the EBI-ChEMBL [2] for protein and chemical representations. We used the basic representations of SMILES and amino acid sequences for chemicals and proteins respectively.

The following sections describes the methodology and conclusions.

Methods

Data sets

The challenge requires the prediction in terms of pKd. Therefore, as a first step, we extracted the rows corresponding to the Kd values from the DTC database. The other types of binding data did not improve the results for our model so we did not include them in our dataset. This amounts to 55678 protein ligand binding data samples.

For chemicals, we used the well-known Extended Connectivity FingerPrints (ECFP) as the descriptors. We downloaded the SMILES strings for each of the compounds from the EBI-ChEMBL database and used the rdkit ibrary [3] to produce 512 length binary bit vectors to represent the compounds. Similarly, for representing kinases, we downloaded the amino acid sequences from EBI-ChEMBL in fasta format. From those sequences, we used PyBioMed library [4] to construct descriptors of proteins. From this library, we used the composition, transition and distribution features (calculateCTD method from CTD module) and dipeptide composition features (CalculateDipeptideComposition method from the AAComposition module) as the feature set. We simply concatenated these two feature vectors to build a descriptor for proteins.

As a second preprocessing step, we removed those features whose variance are below a threshold. For this, we used the VarianceThreshold module of the scikit-learn library [5] with a threshold of 0.1 for chemicals and proteins separately. After this, we obtained descriptors vectors of length 185 and 206 for chemicals and proteins respectively. Finally, we standardised the feature vectors separately to zero mean and unit standard deviation. Therefore, at the end, our final descriptor for a compound-protein pair is a vector of length 391, a simple concatenation of the above mentioned vectors.

Models

The machine learning models that we use to model the protein ligand binding data is neural networks and gradient boosted decision tree models. These two models are the best performing models from a number of other models that we executed.

The neural network model we use is a network of two hidden layers of size 1500 and 400 respectively. In addition to that an input layer of 391 and output layer of a single neuron exist in the network. The activation functions of hidden layers are sigmoid functions and the output neuron has a linear activation function. We exploited the Keras library [6] to construct and train this network. We trained the network with a batch size of 128 for 400 epochs by using ADAM method for the optimization.

For gradient boosted trees (GBT), we used the official implementation [7] of the efficient LightGBM algorithm [8]. This algorithm usually works faster than other GBT methods by exploiting the size of gradients for the samples in the dataset and eliminating those that have a small sized gradient. We used all the default values for the parameters except that the number of estimators is set to 3960 which is the value we found by parameter optimization.

After training these two models, the final predictions are produced by a linear ensemble of these. The ensemble is a simple weighted model that computes $$(w_1 * o_{nn} + w_2 * o_{gbt})$$ where $$(o_{nn})$$ and $$(o_{gbt})$$ are the predictions of neural network and GBT model for a given test sample, respectively. Based on a simple parameter optimization, we observed that an equal weighting gives the best results, therefore we set $$(w_1 = w_2 = 0.5)$$.

Conclusion

This submission has used one type of binding data, the Kd values. In the future, we plan to extend the work here to also exploit the other types of binding data as well. Also the performance can be improved by extending the ensemble with other models.

Author Contribution Statement

This is a single author submission.

Running the docker image

Please run the following command to run the image associated:

```
docker run -it --rm -v ${PWD}/io:/input -v ${PWD}/io:/output demo
```

where 'demo' represents the name of the image and 'io' is a folder in the host machine for the input (the template.csv file in the challenge).

References

[1]Tang et al. Drug Target Commons: A Community Effort to Build a Consensus Knowledge Base for Drug-Target Interactions. Cell Chem Biol. 2018 Feb 15;25(2):224-229.e2

[2] Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E, Davies M, Dedman N, Karlsson A, Magariños MP, Overington JP, Papadatos G, Smit I, Leach AR. (2017) 'The ChEMBL database in 2017.' Nucleic Acids Res., 45(D1) D945-D954.

[3] RDKit: Open-Source Cheminformatics Software. (https://www.rdkit.org)

[4] PyBioMed Molecular descriptors library. (https://github.com/gadsbyfly/PyBioMed)

[5] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[6] Chollet, François. Keras. (2015) (http://keras.io)

[7] LightGBM Library. (https://github.com/Microsoft/LightGBM)

[8] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in Neural Information Processing Systems. 2017.