# DMIS_DK Submission

Sungjoon Park[1], Minji Jeon[1], Sunkyu Kim[1], Junhyun Lee[1], Seongjun Yun[1], Bumsoo Kim[1], Buru Chang[1], and Jaewoo Kang[1,2,*] 1.Department of Computer Science and Engineering, Korea University, Seoul, Republic of Korea 2.Interdisciplinary Graduate Program in Bioinformatics, Korea University, Seoul, Republic of Korea

[*] corresponding author

# Introduction

It is important to obtain binding affinity between drugs and kinases in drug discovery process. However, measuring binding affinity is cost-intensive. To address this, we developed a machine learning model to predict binding affinity between drugs and kinases. The IDG-DREAM Drug-Kinase Binding Prediction Challenge provided the UniProt IDs of proteins in the Drug Target Commons dataset and we could get sequence information of the proteins using the UniProt IDs. In the binding affinity prediction task, 3D structural information of proteins is known to be more informative than sequence information of proteins. However, predicting of structures of proteins is a difficult problem. We, therefore, propose a ligand-based prediction model that focuses on the structures of drugs, rather than using information such as sequences of proteins or structures of proteins.

# Methods

## Data

In this challenge, we used Drug Target Commons (DtcDrugTargetInteractions.csv) [1] and BindingDB data (BindingDB_All_2019m2.tsv.zip) [2] for training binding affinity prediction models. Among diverse measurements, kd, ki and IC50 were used for the training. All binding affinity values were transformed by -log10(x/1e9). We chose median values for the duplicate samples (i.e., same compound-protein pair). We submitted two prediction models: Random Forest (RF) and Ensemble of multi-task Graph Convolutional Networks (GCN). For the RF model, 128,181 samples (compound-protein pair) having 199 proteins and 57,399 compounds were used for the training. For the multi-task GCN model, 953,521 samples (compound-protein pair) having 1,474 proteins and 474,875 compounds were used for the training.

## Models

### Random Forest

We trained a Random Forest model [3] for each protein. 2048-dimensioned Extended Connectivity Fingerprint (ECFP) [4] is used as input. ECFP is one of the drug structure representation methods that represents the presence of substructures in a molecule as a binary vector. The output of the model is the pKd, pKi and IC50 values of the dataset. We divided the dataset into 80:20 and used each dataset as a training set and a validation set. Hyper-parameters were selected based on the performance of the validation set. There is no model for S4 samples because we trained the models only for the proteins in the training set. For each S4 sample in Round 2, we measured the similarity between 199 proteins and the protein of the sample based on protein sequences, and selected the top 3 similar proteins. The predicted pKd value of the sample is the average of the predicted values from the top 3 protein models.

### Multi-task GCN Ensemble

We designed 4 multi-task GCN architectures. The multi-task GCN model takes a SMILES string as input and predict binding affinities for 1,474 proteins. In the 1,474 proteins, 199 out of 207 round 2 proteins were included. SMILES strings were converted to molecular graphs using RDKit python library [5]. We designed a 78 dimensional feature vector to represent a node (here, atom) in a molecular graph. Description of the feature vector is shown in Table 1. For the submission, we averaged the predictions of the last K epochs. Then, we averaged all the 12 multi-task GCN models (4 different architecture with 3 different weight initialization) averaged predictions. We selected the hyper-parameters of the multi-task GCN models based on the performance of the validation set. We implemented the GCN models using PyTorch Geometric (PyG) library. Procedure for predicting S4 samples in Round2 data was the same as the random forest model.

### Multi-task GCN architecture 1

GAT layer + GCN layer + Pooling layer + 1 dense layer + 1 output layer GAT layer: Graph convolution layer using graph attention networks proposed in “Graph Attention Networks” [6]. Multi-head vectors were concatenated (# of head: 10, input dim: 78 , output dim: 780). GCN layer: Graph convolution layer proposed in "Semi-Supervised Classification with Graph Convolutional Networks" [7] (input dim: 780 , output dim: 780). Pooling layer: Concatenation of average pooling and max pooling across all node feature vectors (input dim: 780 , output dim: 1,560). Dense layer: Fully connected layer with dropout (dropout rate : 0.5, input dim: 1,560, output dim: 1,500). Output layer: Fully connected layer (input dim: 1,500 , output dim: 1,474).

### Multi-task GCN architecture 2 /3

4 GCN layer & Pooling layer (after each GCN layer) + 4 GCN layer(after pooling) + 1 dense layer + 1 output layer GCN layer: Graph convolution layer proposed in "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks" [8] (input dim: 78 , output dim: 128). Pooling layer: Hierarchical graph pooling layer proposed in “Self-Attention Graph Pooling” [9] (pooling ratio=0.25) GCN layer (after pooling): Graph convolution layer proposed in "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks". (input dim: 128 , output dim: 128). Dense layer: Fully connected layer with dropout (dropout rate : 0.5, input dim: 512, output dim: 512). Output layer: Fully connected layer (input dim: 512 , output dim: 1,474).

### Multi-task GCN architecture 4

4 GAT layer & Pooling layer (after each GCN layer) + 4 GCN layer(after pooling) + 1 dense layer + 1 output layer GAT layer: Graph convolution layer using graph attention networks proposed in “Graph Attention Networks”. Multi-head vectors were concatenated (# of head: 2 & 4 , input dim: 78 , output dim: 128). Pooling layer: Hierarchical graph pooling layer proposed in “Self-Attention Graph Pooling” (pooling ratio=0.25) GCN layer (after pooling): Graph convolution layer proposed in "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks". (input dim: 128 , output dim: 128). Dense layer: Fully connected layer with dropout (dropout rate : 0.5, input dim: 512, output dim: 512). Output layer: Fully connected layer (input dim: 512 , output dim: 1,474).

| Atom feature type | RDkit function | Encoding type | Dimension |
|---|---|---|---|
| Atom symbol | atom.GetSymbol() | One hot encoding | 44 |
| Degree | atom.GetDegree() | One hot encoding | 11 |
| Total number of Hs | atom.GetTotalNumHs() | One hot encoding | 11 |
| Implicit valence | atom.GetImplicitValence() | One hot encoding | 11 |
| Is aromatic | atom.GetIsAromatic() | Bool (0 or 1) | 1 |
| Total | | | 78 |

Table 1. Description of the atom feature

# Results

The results of round2 leaderboard

| Model | objectID | RMSE | Spearman | AUC |
|---|---|---|---|---|
| Random Forest | 9686312 | 1.002 | 0.484 | 0.774 |
| Multi-task GCN Ensemble | 9686330 | 0.949 | 0.485 | 0.771 |

# References

[1] Tang, J., Ravikumar, B., Alam, Z., Rebane, A., Vähä-Koskela, M., Peddinti, G., ... & Gautam, P. (2018). Drug Target Commons: a community effort to build a consensus knowledge base for drug-target interactions. Cell chemical biology, 25(2), 224-229.

[2] Gilson, M. K., Liu, T., Baitaluk, M., Nicola, G., Hwang, L., & Chong, J. (2015). BindingDB in 2015: a public database for medicinal chemistry, computational chemistry and systems pharmacology. Nucleic acids research, 44(D1), D1045-D1053.

[3] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[4] Rogers, D., & Hahn, M. (2010). Extended-connectivity fingerprints. Journal of chemical information and modeling, 50(5), 742-754.

[5] RDKit: Open-source cheminformatics; http://www.rdkit.org

[6] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[7] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[8] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2018). Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. arXiv preprint arXiv:1810.02244.

[9] Lee, J., Lee, I., & Kang, J. (2019). Self-Attention Graph Pooling. arXiv preprint arXiv:1904.08082.