IDG-DREAM Drug-Kinase Binding Prediction Challenge: Group KinaseHunter

Hansaim Lim[1] and Lei Xie[1,2]
[1]Ph.D. program in Biochemistry, Graduate Center, The City University of New York
[2]Department of Computer Science, Hunter College, The City University of New York

Data set preparation

1. Chemical-kinase binding affinity data collection

To build a large-scale chemical-kinase binding affinity data set, we integrated multiple public databases and published kinome assays. The databases we used are ChEMBL (ver. 24), BindingDB, and LINCS-HMS KinomeScan database. ChEMBL database contains chemical-protein binding affinities for mutant proteins. We collected those activities for mutants from ChEMBL. BindingDB contains chemical-protein affinity data sets from multiple sources, including PubChem, PDSPKi, U.S. patents, and the curated data set by BindingDB team. LINCS-HMS KinomeScan database contains chemical-kinase binding affinities for various chemicals and target kinases. Four published kinome assay data sets were included from 1) Christmann-Franck et al. JCIM, 2016, 2) Drewry et al. PLoS One, 2017, 3) Klaeger et al. Science, 2017, and 4) Sorgenfrei et al. ChemMedChem, 2017. To merge redundant activity measurements (e.g. multiple activity records for a chemical-kinase pair), we converted chemicals into InChIKey and kinases into UniProt ID. We converted all activities into log-scale since the target problem is to predict binding affinity in pKd. We excluded activity measurements in other metrics than Ki, pKi, Kd, and pKd. If multiple activity records found for a chemical-kinase pair, we averaged them. For feature calculation processes, we collected SMILES strings for each chemical and primary amino acid sequences for each kinase. In case the data source does not provide chemical SMILES or InChIKey, we used the PUGRest service from PubChem to convert the chemical identifiers. We used UniProt database to convert gene names into UniProt IDs. For mutant proteins, we prepared mutated protein sequences by replacing, inserting, or deleting sequence parts as appeared in protein identifiers.

2. Splitting data into train, dev, and test sets

We used Leave-Chemical-Set-Out (LCSO) strategy to simulate new drug discovery process and reduce overfitting. To evaluate model fitting and generalized performance, we split the chemical-kinase samples into train, dev, and test sets. We used train set to train models, dev set to evaluate how well the model is trained and optimize hyperparameters, and test set to evaluate the final performance of our model. Before the final prediction for the challenge, we trained the model again with the dev and test sets.

We split the chemicals into two groups: 42708 training chemical set and 581 dev-test chemical set. We kept the structural similarity for any chemical-chemical pairs across two sets lower than 0.8, ensuring that the dev or test data represent new chemical molecules that are not found in training data. The chemical-chemical similarity scores were measured by Tanimoto coefficient (Jaccard similarity) between the two ECFP4 (1024 bits)

representations of chemical molecules. The chemical-kinase affinity samples for dev-test chemicals are split into dev and test sets in approximately 8:2 ratio. The data statistics for train/dev/test sets are in Table 1.

Table 1	Train	Dev	Test
#samples	400170	33701	8618
#unique chemicals	42708	520	343
#unique proteins	482	460	447

Method

1. Model architecture

Our model uses graph-based molecular representations for chemicals and proteins. Our target problem is to predict binding affinity in pKd, given the input chemical and kinase. We denote each sample of chemical-kinase affinity data as $y_i = (c_i, k_i)$, where y_i is the known pKd value, c_i is the chemical, and k_i is the kinase in the i^{th} sample.

Our model processes chemical molecules using graph convolutional neural network (Neural Fingerprint) proposed by Duvenaud et al. (NIPS, 2015), which produces chemical molecular fingerprint of a user-defined length. Briefly, the Neural Fingerprint applies weight filters to atoms and bonds ordered by their degrees in each molecule. We set the length of chemical fingerprint to 128. We denote the chemical fingerprint operation as $f_{c_i} = \mathcal{G}_c(c_i)$, where f_{c_i} is the feature vector representation of c_i ($f_{c_i} \in \mathbb{R}^{128}$), and $\mathcal{G}_c$ is the Neural Fingerprint operation. We used RDKit python package to preprocess chemical molecules for fingerprint operation.

Kinases are processed into two different representations: kinase domain feature vectors, and kinase active site feature vectors. We denote the features of kinase domain sequence and kinase binding site sequence in the i^{th} sample as k_i^d and k_i^b, respectively, where both k_i^d and k_i^b are sequence of 27 values for each amino acid (20 for PSSM and 7 for amino acid physical properties). For each kinase, we downloaded kinase domain sequences from UniProt, and we calculated the position-specific scoring matrix (PSSM) against UniRef50 database (nonredundant sequence database clustered at 50% sequence identity) using PSIBLAST standalone package with 3 iterations for each kinase domain sequence. In the kinase domain feature representation, we used the PSSM values for the whole domain as k^d.

In the kinase active site feature representation, we first aligned the kinase domain sequences using CLUSTALW web server with default option. Then, we identified the active site residues of the serine/threonine-protein kinase pim-1 (P11309) from its 3D structure. From the 3D structure, we also collected neighboring triplets, three amino acid residues that are closer than 6.0 Å with each other. Then, we identified the binding site residues and structural neighbors of each kinase from the multiple sequence alignment. The corresponding PSSM values for the binding site residues were used as the input, k^b, in contrast to the kinase domain representation. For both representations, we also used various types of physical properties of amino acids: average hydrophobicity (Cid et al, 1992. PMID: 1518784), van der Waals volume (Fauchere et al. 1988. PMID:3209351), polarity

(Grantham et al. 1974. PMID:4843792), net charge (Klein et al., 1984. PMID:6547351), average volume in buried state (Chothia, 1975. PMID:1118010), accessible surface area in tripeptide, and accessible surface area in folded protein (Chothia, 1976. PMID:994183). The PSSM and amino acid properties were normalized by z-scaling, i.e. $z = \frac{x-\mu}{\sigma}$, where x is the feature value, and μ, σ are the mean and standard deviation of the features of same type.

The kinase feature vectors were obtained by applying graph convolutional network with attention mechanism. Many recent graph neural network methods use message passing scheme with attention mechanism as an aggregator of the node weights. Our method uses a cardinality preserved attention network (CPAN), a novel graph convolutional method developed by us (manuscript under review). CPAN takes graph representation of kinases as input, and it produces protein feature of length 154. Each kinase is a graph, where each amino acid is a node with its feature (k_i^d or k_i^b). We denote the kinase feature calculation as $f_{k_i} = \mathcal{G}_k\left(k_i^d\right)\ or\ \mathcal{G}_k(k_i^b)$, where f_{k_i} is the protein feature output from CPAN ($f_{k_i} \in \mathbb{R}^{154}$), and $\mathcal{G}_k$ is the graph convolutional operation by CPAN.

2. Attentive pooling and feature transformation

With the feature representation of chemicals and kinases, we applied attentive pooling strategy similar to (dos Santos et al, 2016, arXiv) to weigh the contributions of each feature and rescale the chemical and kinase feature vectors.

$$f'_{c_i} = f_{c_i} \odot softmax\left(f_{k_i} \cdot \Theta\right)$$
$$f'_{k_i} = f_{k_i} \odot softmax(f_{c_i} \cdot \Theta^T)$$

In the above equations, f'_{c_i}, f'_{k_i} represent the rescaled feature vectors of chemical and kinase in the i^{th} sample, Θ represents the attentive weight matrix ($\theta \in \mathbb{R}^{128\times154}$), $\odot$ represents element-wise product (Hadamard product), and $\cdot$ represents dot product of matrices.

The rescaled feature vectors were fed into a feature transformation layer to make final prediction. We denote the feature transformations as $f''_{c_i} = \mathcal{T}_{\mathrm{c}}\left(f'_{c_i}\right)$ and $f''_{k_i} = \mathcal{T}_{\mathrm{k}}\left(f'_{k_i}\right)$, where f''_{c_i} and f''_{k_i} represent transformed chemical and kinase features, respectively. The transformation layer contains four layers of fully-connected 64 neurons activated by ReLU after batch normalization. The fifth layer contains fully-connected 64 neurons with batch normalization but without ReLU activation. The predicted pKd activity value was calculated by the dot product of the transformed feature vectors. $\hat{y}_i = f''_{c_i} \cdot f''_{k_i}$, where $\hat{y}$ denotes the predicted pKd value.

3. Training procedure

We used Adam optimizer with cosine annealing for learning rate adjustment. The initial learning rate was set to 1e-4, and batch size was set to 128. We also applied balanced regression strategy to ensure the diversity of training samples. We split the training samples into 16 bins by the pKd values, where the first bin contains samples with pKd ≤ 4.0, and the last bin contains samples with pKd > 10. The other bins were evenly spaced by an increment of 0.4. Then, we chose 8 samples from each bin to form a minibatch of 128 samples. The model was trained for 100 epochs by minimizing MSE (i.e. $\Sigma(y - \hat{y})^2$) with early stopping strategy based on the Pearson's correlation coefficient measured on the dev set. From the dev set, we randomly picked 128 samples and measured the Pearson's

correlation coefficient. We repeated it 10 times and averaged for each epoch to measure the dev performance. Our final model was chosen by the training epoch where the dev performance was highest.

Below are the docker commands to run our model for Round 2 prediction.

```
$docker run docker.synapse.org/syn17037396/graphdti-v1 > round2-1.csv
#Round 2. First submission. (objID 9686292)

$docker run docker.synapse.org/syn17037396/graphdti-v2 > round2-2.csv
#Round 2. Second submission. (objID 9686304)
```