Title: Q.E.D's solution to IDG-DREAM Drug-Kinase Binding Prediction Challenge

Authors

Fangping Wan[1, 4] (Username: Stig, Email: wfp15@mails.tsinghua.edu.cn), Shuya Li[1, 4] (Username: Shuya Li, Email: lishuya17@mails.tsinghua.edu.cn), Yunan Luo [2](Username: Yunan Luo, Email: yunan@illinois.com), Hailin Hu[3, 4] (Username: Corgi, Email: huhl16@mails.tsinghua.edu.cn), Jian peng[2] (Email: jianpeng@illinois.edu), Jianyang Zeng[1, 4] (Email: zengjy321@tsinghua.edu.cn)

1: Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. 2: Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA. 3: School of Medicine, Tsinghua University, Beijing, China. 4: Silexon Co., LTD, Beijing, China.

1. Summary

This project contains both submission 2 (objectID 9686285) and submission 1 (objectID 9686189) methods from Q.E.D team, for both sub-challenges of IDG-DREAM Drug-Kinase Binding Prediction Challenge round2.

2 .Methods - submission 2 (objectID 9686285)

2.1 Data pre-processing

Generally, we used the compound-protein affinity data from Drug Target Commons (DTC) (Tang J, Ravikumar B, Alam Z, et al. DrugTargetCommons:a community effort to build a consensus knowledgebase for drug-target interactions. Cell Chemical Biology, 2018, 25(2):224-229.e2.) (downloaded from https://www.synapse.org/#!Synapse:syn17017461).

2.1.1 Used compounds:

Among the compounds curated in DTC, we only considered compounds that (1) have CHEMBL ID or (2) have PKD affinity. Compounds that have CHEMBL ID have structure information encoded by InChI or SMILES (stored in DTC). Compounds that have PKD affinity are encoded as InChIkey in DTC. We used PubChem Identifier Exchange Service (https://pubchem.ncbi.nlm.nih.gov/idexchange/idexchange.cgi) to map InChIkey to corresponding SMILES (stored as "DTC_pkd_inchikey_to_smiles" in the data folder in docker).

2.1.2 Used proteins:

Among the proteins curated in DTC, we only considered proteins that (1) have Uniprot ID and were labeled as kinase in Uniprot or (2) came from round_2_template.csv (downloaded from https://www.synapse.org/#!Synapse:syn16809885).

2.1.3 Used affinities:

Among the binding affinity pairs curated in DTC, we only considered the pairs that satisfied the following conditions: (1) compounds that came from 2.1.1; (2) proteins that came from 2.1.2; (3) the binding affinity measurement type was

Ki or KI or Kd or KD or EC50 or PKD; (4) the binding affinity measurement relation was "=" (i.e., equal); (5) if the binding affinity measurement type was Ki or KI or Kd or KD or EC50, the standard unit should be "NM"; (5) In DTC dataset, each row represents a binding affinity, the "target id" column (i.e., the fifth column) should contain only one Uniprot ID, otherwise we did not use this row.

We used the binding affinity pairs that satisfied the above conditions. Then, for the pairs that had the type Ki or KI or Kd or KD or EC50, we converted the binding affinity x using the transformation: $-\log_{10}(x / 10^9)$. For pairs that had the type PKD, we did not do such transformation. Some compound-protein pairs can have multiple binding affinities. In such cases, the median of the binding affinities was used. Note that, the median operation was applied after the $-\log_{10}(x / 10^9)$ transformation.

After the above pre-processing, the total number of compounds we used in training process is 13,608; the total number of proteins we used in training process is 527; and the total number of binding affinities we used in training process is 60,462.

2.2 Prediction method

Under the problem setting that requires fine-grained discrimination between similar compounds or targets, we found the explicit introduction of similarity metrics as model input generally outperformed other more complex model architectures that attempt to organize representative features from scratch. In particular, we defined a comprehensive set of compound structure similarity and protein sequence similarity metrics as the input of our model. Then, we leveraged CGKronRLS method (Pahikkala T. Fast gradient computation for learning with tensor product kernels and sparse training labels[C]//Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). Springer, Berlin, Heidelberg, 2014: 123-132.) (implemented in Pahikkala T, Airola A. RLScore: regularized least-squares learners[J]. The Journal of Machine Learning Research, 2016, 17(1): 7803-7807. https://github.com/aatapa/RLScore) as the regression model to predict the binding affinity.

2.2.1 Features of compounds

We computed the following compound similarity matrices as compound features (computed by RDKit: https://github.com/rdkit/rdkit):

1: Tanimoto similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=True.

2: Tanimoto similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=False.

3: Tanimoto similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=True.

4: Tanimoto similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=False.

5: Dice similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=True.

6: Dice similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=False.

7: Dice similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=True.

8: Dice similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=False.

2.2.2 Features of proteins

We computed the protein similarity matrix as protein features (computed by https://github.com/mengyao/Complete-Striped-Smith-Waterman-Library). Specifically, protein similarity is defined as the normalized Striped-Smith-Waterman similarity. Let sw(s1, s2) be the alignment score of Striped-Smith-Waterman algorithm between protein sequences s1 and s2. The protein similarity between s1 and s2 can be defined as sw(s1, s2) / sqrt(sw(s1, s1) * sw(s2, s2)).

2.2.3 Regression model

We used CGKronRLS as the regression model. It took the compound and protein similarity matrices as input and output the binding affinity.

2.2.4 Model ensemble

Instead of using single model, we used the ensemble of multiple CGKronRLS (with different iterations, regularization parameters and input features) models. We ensembled 440 CGKronRLS models with the following setting: protein feature ∈ {the protein similarity matrix from 2.2.2} x compound feature ∈ {eight compound similarity matrices from 2.2.1} x regularization parameter of CGKronRLS ∈ {0.1, 0.5, 1.0, 1.5, 2.0} x iteration of CGKronRLS ∈ {400, 410, 420, 430, 440 ,450, 460, 470, 480, 490, 500}, where x means cartesian product.

After training the 440 CGKronRLS models, we averaged the predictions among them to produce the final prediction.

3 .Methods - submission 1 (objectID 9686189)

3.1 Data pre-processing

Generally, we used the compound-protein affinity data from Drug Target Commons (DTC) (downloaded from https://www.synapse.org/#!Synapse:syn17017461).

3.1.1 Used compounds:

Among the compounds curated in DTC, we only considered compounds that (1) has CHEMBL ID or (2) has PKD affinity. Compounds that have CHEMBL ID have structure information encoded by InChI or SMILES (stored in DTC). Compounds that have PKD affinity are encoded as InChIkey in DTC. We used PubChem Identifier Exchange Service (https://pubchem.ncbi.nlm.nih.gov/idexchange/idexchange.cgi) to convert InChIkey into corresponding SMILES (stored as "DTC_pkd_inchikey_to_smiles" in data folder in docker).

3.1.2 Used proteins:

Among the proteins curated in DTC, we only considered proteins that (1) has Uniprot ID and were labelled as kinase in Uniprot or (2) came from round_2_template.csv (downloaded from https://www.synapse.org/#!Synapse:syn16809885).

3.1.3 Used affinities:

Among the binding affinity pairs curated in DTC, we only considered the pairs that satisfied the following conditions: (1) compounds that came from 2.1.1; (2) proteins that came from 2.1.2; (3) the binding affinity measurement type was Ki or KI or Kd or KD or PKD; (4) the binding affinity measurement relation was "=" (i.e., equal); (5) if the binding affinity measurement type was Ki or KI or Kd or KD, the standard unit should be "NM"; (5) In DTC dataset, each row represents a binding affinity, the "target id" column (i.e., the fifth column) should contain only one Uniprot ID, otherwise we did not use this row.

We used the binding affinity pairs that satisfied the above conditions. Then, for the pairs that had the type Ki or KI or Kd or KD, we converted the binding affinity x by the following transformation: -log~ 10~ (x / 10^9). For pairs that had the type PKD, we did not do such transformation. Some compound-protein pairs can have multiple binding affinities. In such cases, the median of the binding affinities was used. Note that, the median operation was conducted after the -log~ 10~ (x / 10^9) transformation.

After the above pre-processing, the total number of compounds we used in training process is 11396, the total number of proteins we used in training process is 501 and the total number of binding affinities we used in training process is

58023.

3.2 Prediction method

Generally, we used compound structure similarity and protein sequence similarity as features and CGKronRLS (implemented in https://github.com/aatapa/RLScore) as the regression model to predict the binding affinity.

3.2.1 Features of compounds

We computed the following compound similarity matrices as compound features (computed by RDKit: https://github.com/rdkit/rdkit):

1: Tanimoto similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=True.

2: Tanimoto similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=False.

3: Tanimoto similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=True.

4: Tanimoto similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=False.

5: Dice similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=True.

6: Dice similarity of morgan fingerprint with arguments radius=2, nBits=1024, useChirality=False.

7: Dice similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=True.

8: Dice similarity of morgan fingerprint with arguments radius=3, nBits=1024, useChirality=False.

3.2.2 Features of proteins

2.2.2.1 Protein sequence similarity

We first computed the following protein sequence similarities:

(1) We first computed the protein sequence similarity matrix by Striped-Smith-Waterman algorithm (computed by https://github.com/mengyao/Complete-Striped-Smith-Waterman-Library). Specifically, protein similarity is defined as the normalized Striped-Smith-Waterman similarity. Let sw(s1, s2) be the alignment score of Striped-Smith-Waterman algorithm between protein sequences s1 and s2. The protein similarity between s1 and s2 is: sw(s1, s2) / sqrt(sw(s1, s1) * sw(s2, s2)).

(2) We then computed the eight protein sequence similarity matrices by Generic string kernel (Giguere S, Marchand M, Laviolette F, et al. Learning a peptide-protein binding affinity predictor with kernel ridge regression[J]. BMC bioinformatics, 2013, 14(1): 82.) (computed by function gs_gram_matrix from https://github.com/search?q=generic+string+kernel) with the following hyperparameters: sigma_position = 1.0, sigma_amino_acid = 1.0, substring_length = 1; sigma_position = 1.0, sigma_amino_acid = 1.0, substring_length = 2; sigma_position = 1.0, sigma_amino_acid = 1.0, substring_length = 3; sigma_position = 1.0, sigma_amino_acid = 1.0, substring_length = 4; sigma_position = 2.0, sigma_amino_acid = 2.0, substring_length = 1; sigma_position = 2.0, sigma_amino_acid = 2.0, substring_length = 2; sigma_position = 2.0, sigma_amino_acid = 2.0, substring_length = 3; sigma_position = 2.0, sigma_amino_acid = 2.0, substring_length = 4. In total, we had nine protein sequence similarity matrices (one based on Striped-Smith-Waterman algorithm and eight based on Generic string kernel).

3.2.2.2 Other protein similarities

We further calculated the following protein similarity matrices:

(1) Protein-protein interaction similarity. We downloaded protein-protein interaction network from STRING database (Szklarczyk D, Franceschini A, Wyder S, et al. STRING v10: protein-protein interaction networks, integrated over the tree of life[J]. Nucleic acids research, 2014, 43(D1): D447-D452.) ("9606.psicquic-mitab_2.5.v10.5.txt" in data folder in docker). We calculated the protein-protein interaction similarity matrix by Jaccard similarity.

(2) Protein-pathway association similarity. We downloaded protein-pathway association network from The Comparative Toxicogenomics Database (CTD) database (Mattingly C J, Colby G T, Forrest J N, et al. The Comparative Toxicogenomics Database (CTD)[J]. Environmental health perspectives, 2003, 111(6): 793-795.) ("CTD_genes_pathways.tsv" in data folder in docker). We calculated the protein-pathway association similarity matrix by Jaccard similarity.

(3) Protein-go association similarity. We used pygosemsim (https://github.com/mojaie/pygosemsim) to calculate Protein-go association similarity matrix. Specifically, we used "go-basic" to build go term graph and "goa_human", as go term annotation. Best-Match Average (BMA) was used to calculate the similarity between two proteins.

(4) Protein-protein structure similarity. We used PYMOL (DeLano W L. PyMOL[J]. 2002.) to calculate the structure similarity matrix. To be specific, first, each protein was mapped to its corresponding PDB entries (if any). If there were more than one PDB entries for one protein, we used the one with the longest sequence coverage. Then, the structure files (.pdb) were downloaded for RCSB PDB (http://www.rcsb.org). After that, we used the "super" function provided PYMOL to calculate the pairwise sequence-independent structure-based alignment (https://pymolwiki.org/index.php/Super). Finally, the similarity score between a pair of proteins was defined as *-RMSD*. We found it difficult to install PYMOL in linux system. We computed this similarity matrix in Windows system and the result ("structural_sim_mat_rmsd.npy") was provided as in data folder in docker. A mapping list ("kinase_to_PDB_mapping.txt") between Uniprot ID and PDB ID of proteins we used are also provided in the data folder in docker. In total, we obtained four protein similarity matrices in this subsection.

3.2.2.3 Similarity network fusion.

We used similarity network fusion (Wang B, Mezlini A M, Demir F, et al. Similarity network fusion for aggregating data types on a genomic scale[J]. Nature methods, 2014, 11(3): 333.) to fuse the similarity networks from sections 2.2.2.1 and 2.2.2.2. Specifically, we fused {nine protein similarity matrices from 2.2.2.1} x {four similarity matrices from 2.2.2.2}, where x means cartesian product, to create 36 protein similarity matrices as input features for the regression model (see section 2.2.3). Similarity network fusion was implemented by SNFpy(https://github.com/rmarkello/snfpy). Since we already had the similarity matrices, we directly used snf.make_affinity() function with default hyperparameters to fuse networks.

3.2.3 Regression model

We used CGKronRLS as the regression model. It took the compound and protein similarity matrices as input and output the binding affinity.

3.2.4 Model ensemble

Instead of using single model, we used the ensemble of multiple CGKronRLS (with different iterations, regularization parameters and input features) models. Specifically, we ensemble 5184 CGKronRLS models with the following setting: protein feature ∈ {36 protein similarity matrices from 2.2.2.3} x compound feature ∈ {8 compound similarity matrices from 2.2.1} x regularization parameter of CGKronRLS ∈ {0.1} x iteration of CGKronRLS ∈ {200, 210, 220, 230, 240, 250} + protein feature ∈ {36 protein similarity matrices from 2.2.2.3} x compound feature ∈ {8 compound similarity matrices from 2.2.1} x regularization parameter of CGKronRLS ∈ {0.5, 1.0} x iteration of CGKronRLS ∈ {450, 460, 470, 480, 490, 500} , where x means cartesian product and + means union.

After getting the 5184 CGKronRLS models, we averaged the predictions among them to produce the final prediction.

4. Testing environment

Submission 2 (objectID 9686285) model was trained and tested on a server with the following configuration: System version: Ubuntu 16.04.2 LTS; Cores: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, 32 in total; Memory: 264024700 kB.

Submission 1 (objectID 9686189) model was trained on several severs, and the results were then ensembled. The severs have the following configurations: 1) System version: Ubuntu 16.04.2 LTS; Cores: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, 32 in total; Memory: 264024700 kB. 2) System version: Ubuntu 16.04 LTS; Cores: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz, 48 in total; Memory: 264025868 kB. 3) System version: Ubuntu 14.04.5 LTS; Cores: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, 32 in total; Memory: 264032840 kB.

#5. Running the final model To run our docker image and get only the final output file for our submission 2 (objectID 9686285), run the following command:

```
$ docker run -it --rm -v ${PWD}/input:/input -v ${PWD}/output:/output
docker.synapse.org/syn18519352/qed-sub2:9686285
```

To run our docker image and get all the output files of intermediate processes for our submission 2 (objectID 9686285), run the following command:

```
$ docker run -it --rm -v ${PWD}/input:/input -v ${PWD}/output:/output -v
${PWD}/data:/data -v ${PWD}/SW_based_prediction:/SW_based_prediction
docker.synapse.org/syn18519352/qed-sub2:9686285
```

To run our docker image and get only the final output file for our submission 1 (objectID 9686189), run the following command:

```
$ docker run -it --rm -v ${PWD}/input:/input -v ${PWD}/output:/output
docker.synapse.org/syn18519352/qed-sub1:9686189
```

To run our docker image and get all the output files of intermediate processes for our submission 1 (objectID 9686189), run the following command:

```
$ docker run -it --rm -v ${PWD}/input:/input -v ${PWD}/output:/output -v
${PWD}/data:/data -v ${PWD}/SNF_based_prediction:/SNF_based_prediction
docker.synapse.org/syn18519352/qed-sub1:9686189
```

Note that the running the submission 1 (objectID 9686189) docker container may take more than one week. Thus we recommend using our submission 2 solution (objectID 9686285), which ran much faster and achieved better performance than submission 1 (objectID 9686189).

6. Author contribution

F.W., S.L., Y.L., H.H., J.P., and J.Z. conceived the method. F.W. and S.L. conducted the data pre-processing. F.W., S.L., Y.L. and H.H. calculated the features. F.W. and S.L. wrote and ran the regression model. S.L. prepared the docker with the help of F.W. F.W. wrote the writeup with support from all authors.