DreamChallenge Report - DruginaseLearning Team

Christos Fotis[1], Panagiotis Terzopoulos[1], Konstantinos Ntagiantas[1] and Leonidas G.Alexopoulos[1,2]

1. BioSys Lab, National Technical University of Athens, Athens, Greece.

2. ProtATonce Ltd, Athens, Greece.

Introduction

Our efforts focused on building a deep end-to-end model similar to Ozturk et al. (Ozkirimli Hakime Ozturk and Arzucan Ozgur. Deepdta: Deep drug-target binding affinity prediction.arXiv:1801.10193, 2018.), that takes as input the SMILES representation of a compound and the amino acid sequence of a protein and outputs the KD value of the compound-protein pair. On this front, we investigated several deep architectures in order to represent the SMILES-sequence pairs in a latent space that best captures the nature of the binding affinity prediction. Furthermore, a considerable amount of our effort focused on augmenting the initial DTC dataset, with more compound-protein pairs having available KD values in the literature.

Methods

Data and augmentation

The initial dataset from DTC was augmented using various compound-kinase binding datasets that are publicly available in the web and in the literature. Overall, compound-kinase pairs with KD values from DTC, BindingDB, KKB, PKIS, HMS LINCS and Davis et al. were combined to create the final dataset for training and validation. The final dataset consisted of over 105K unique drug-protein interactions labeled with the Kd affinity metric. A detailed report of this work can be found on https://github.com/bsl-ntua.

Models

We experimented with different end-to-end architectures that utilize different methods for the latent representation of the SMILES and a deep CNN for the latent representation of the amino acid sequences.

1. The first architecture used a 3 layer deep graph convolutional network similar to (Jorge Aguilera-Iparraguirre Rafael Gomez-Bombarelli Timothy Hirzel AlanAspuru-Guzik David Duvenaudy, Dougal Maclauriny and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. NeuralInformation Processing Systems (NIPS), 2015.), to extract application specific neural fingerprints from the compound structures. These fingerprints were then concatenated with the output of a 3 layer deep CNN that encodes the amino acid sequences of the proteins. The combined feature vector was fed through 2 fully connected layers for the final KD prediction. Batch normalization layers and relu activations were used throughout the network except for the final prediction layer. In order to reduce overfitting, dropout and L2 regularization was used between the fully connected layers.
2. Regarding the second architecture, a deep LSTM autoencoder was first trained on the SMILES sequences of all the compounds in the training, validation and test sets. The autoencoder used as input the one-hot representation of the SMILES and was tasked to predict the next letter in the sequence. The output of the trained encoder can serve as a compressed latent representation of the SMILES space. The idea behind training an autoencoder first, is that the encoder learns to represent all the available SMILES (training and test) and the final model should

perform better than a fully end-to-end architecture that has never seen the structures of the test set. Thus, the final model used as input the output of the encoder along with the one-hot encoded amino acid sequences. The sequences were encoded again using a 3 layer deep CNN and concatenated with the output of the encoder to build the final feature vector. This feature vector was then passed through 2 fully connected layers for the final KD prediction. Batch normalization layers and relu activations were used throughout the network except the final prediction layer. In order to reduce overfitting, dropout and L2 regularization was used between the fully connected layers.

Training and Evaluation

The final augmented dataset consisted of 105431 unique pkd values between 12041 compounds and 1690 kinases, with more than 70000 pairs having a pkd value close to 5 (KD=10000μM). In order to reduce the bias of the trained model towards inactive compounds we decided to filter the interactions resulting in a final 3:1 ratio between inactive (*pkd*<7) and active pairs (*pkd*>=7). For model evaluation and parameter tuning a competition specific 5-fold cross validation scheme was employed. More specifically, we identified 5 sets of compounds, with similarity profiles with the training set, almost identical to the similarity profiles of the test set. During each step of the cross-validation all interactions that included the compounds of the validation set were used for model evaluation and the rest for model training. The data augmentation pipeline was implemented using R while the models were built in python using keras with tensorflow as back end. Training was performed on a NVIDIA GPU GTX-1080Ti.

Results and discussion

The best predictions for the test set came out of the second architecture we implemented which included the encoder. Having in mind how difficult it is to really generalize to new compound scaffolds never previously seen during training (Izhar Wallach and Abraham Heifets. Most ligand-based classification bench-marks reward memorization rather than generalization. J. Chem. Inf.Model., 58:916–932, 2018.), an encoder that has been trained to represent the combined train and test-set distribution is expected to boost performance when its encoded feature vector is fed for further training.
As a disclaimer we should say that the predictions we submitted in the final round of the competition are not the true ones our model predicts. This is because of a code error we discovered, unfortunately, after the submission deadline, which implicitly changes the dictionary according to which the SMILES are encoded to 1hot arrays every time one loads the model. This means that the autoenoder was trained with a different 1hot encoding than the one used for the test compounds predictions. We strongly believe that if it wasn't for this error, our latest submission would have scored substantially better.

Docker instructions

The command to run the docker is:

```
$ docker run -it --rm -v ${PWD}:/input -v ${PWD}:/output
docker.synapse.org/syn18525357/druginase-model:9686322
```

As per the instructions on the contest's website, the script included in the docker container reads a provided file with the name "input.csv" that is similar to the round 2 template file, as well as the .h5 files necessary to load our pre-trained keras deep-learning model to make the predictions. All of these files are put into the /input directory of the docker container. Running the docker locally with the provided command, requires the input file and the two .h5 files in the working directory, and will result in the creation of the "predictions.csv" file, that is stored in the /output directory of the docker container. In the instructions it is stated that the provided file should be named "input.csv" or "template.csv", and we decided to go with the former. In order for the container to run succesfully, an active Internet connection is required on the host machine, as the script queries the website of UniProt, in order to retrieve the protein sequences of the respective Uniprot_IDs of the kinases provided in the "input.csv" file. In a system with a moderately stable Internet connection, this may take up to 15 minutes. If the querying is successful, a message will be printed with the number of proteins queried equal to the number of interactions in the "input.csv" file. It should be noted that to run the container locally, not only the "input.csv" file is necessary, but also the two .h5 files that contain the trained model, namely "docker_model22_4.h5", and "test_encoder21.h5".

Authors contribution statement

C. Fotis and K. Ntagiantas conceived a significant part of the presented idea, did most of the data augmentation work in R and helped with fitting the model and interpreting the results. P. Terzopoulos wrote most of the python code and custom functions needed to feed the data through the keras pipeline and trained the models. L.G. Alexopoulos supervised the project.