Structural and Data-Driven Approaches to Drug Discovery

Fergus Boyles and Tom Hadfield, Doctoral Training Centre, University of Oxford, 30NOV2021

Fragment Elaboration

An alternative framework for discovering new drugs to 'High Throughput Screening' is to take a fragment-based approach. Fragment-based drug discovery (FBDD) involves screening a library of extremely small compounds against a target and identifying which ones bind. Fragments generally bind very weakly to the protein and so it's usually necessary to add additional structure to the bound fragments to allow further inter-molecular interactions (thereby improving the binding affinity). You'll learn much more about FBDD later in the week, but in this practical we're going to explore how deep-learning methods can be used to propose novel elaborations, and how we can incorporate structural information into generative models.

Requirements

This session will involve generating elaborations to a fragment using two different generative models. The first model is called GGM (Graph Generative Model)[1] and all the code required to generate molecules can be found in the `DTC_virtual_screening/GGM` directory. To run this model, you'll need to create a new conda environment, which can be done by cd'ing into the above directory and running `conda env create -f ggm_env.yml`.

For the second part of this session, you'll be using a model called STRIFE [2], for which the installation is slightly more painful, as it's based on a few different pieces of academic software. To download STRIFE, go to the github repo `https://github.com/tomhadfield95/STRIFE.git` and clone it onto your

computer. Then you'll need to follow the installation instructions in the github README (the main page), except for installing the CSD Python API, which has already been done. It's completely fine to have difficulties installing STRIFE, so if you do please don't hesitate to ask a demonstrator for help!

Part 1: Generating elaborations using GGM

We'll get started by generating some molecules using the GGM. First, cd into the GGM_scripts directory - it contains a previously trained model called `saved_GGM_model.pt` and a python script for sampling new elaborations (`sample.py`).

For the sake of familiarity, we're going to be working with the same protein target (PDB: 1Q8T) as this morning. We've created a 'fake' fragment by chopping the Pyridine off the crystal ligand, and want to suggest elaborations which replace it (so this exercise is really more akin to an R-group optimisation than a true fragment elaboration). The SMILES of the fragment for elaboration is saved in the file 1q8t_fragSmiles_ggm.txt.

Either by running sample.py, or running the bash script sample_1q8t_GGM.sh, generate 250 elaborations using the GGM.

The output should now be saved in a txt file (in the location specified by the argument `output_filename`). It lists all of the elaborations suggested by the GGM in SMILES format. We want to dock these molecules using GOLD so we can see how they interact with the protein:

Convert the generated molecules into SDF format using the `smiles_to_sdf.py` script and then dock the molecules in the SDF file using the `dock_gold.py` script. As we did this morning, rank them by the docking score assigned to them by GOLD.

Examine the molecules produced by the GGM algorithm. Do the suggested elaborations facilitate additional interactions (Hydrogen Bonds, Pi-stacking etc.) with the protein?

Part 2: Generating elaborations using STRIFE

First, follow the instructions in the STRIFE repo to install it on your computer. STRIFE extracts information from Fragment Hotspot Maps [3], which describe regions of the

protein binding pocket likely to make a disproportionate contribution to binding affinity, and uses this information to determine what types of elaborations to make. ***View the FHM for c-AMP dependent protein kinase by running the following commands in the terminal:***

```
cd <Location of STRIFE repo>
conda env create -f PyMol_environment.yml #Creates environment to
view FHMs
conda activate pymol_env
cd example/hotspotsOut
python pymol_file.py &
```

This should start a PyMol session containing a saved FHM. The Red regions denote likely 'Acceptor Hotspots', where a ligand Hydrogen Bond Acceptor is likely to be able to bind to a protein Hydrogen Bond Donor, whilst the Blue regions denote similar 'Donor Hotspots' and the Yellow regions 'Apolar Hotspots'. Load the fragment saved in `DTC_virtual_screening/1q8t_fragment.sdf` into the PyMol session and see if there are any hotspots in the vicinity of the fragment - if there are then STRIFE will attempt to make elaborations such that matching pharmacophores are placed within those hotspots.

CD back into the STRIFE repo and run the STRIFE algorithm:

```
python STRIFE.py -f example/1q8t_frag.sdf
                 -s example/1q8t_frag_smiles.smi
                 -p example/1q8t_protein.pdb
                 -z example/hotspotsOut/out.zip
                 -o example/STRIFE_1q8t
```

STRIFE generally takes a while (~20-30 mins) to run, so feel free to get on with something else (perhaps skimming some of the papers linked below or making a cup of tea) while it's running. When it's finished, the generated molecules should be saved in the file `example/STRIFE_1q8t/pharmsElabsTestDocked.sdf`. ***As before, identify the highest ranked molecules and inspect some of them in PyMol - is there any difference between the molecules proposed by STRIFE and GGM?***

References

[1] - Lim, J., Hwang, S.Y., Moon, S., Kim, S. and Kim, W.Y., 2020. Scaffold-based molecular design with a graph generative model. *Chemical science*, *11*(4), pp.1153-1164

[2] - Hadfield, T.E., Imrie, F., Merritt, A., Birchall, K. and Deane, C.M., 2021. Incorporating Target-Specific Pharmacophoric Information Into Deep Generative Models For Fragment Elaboration. *bioRxiv*.

[3] – Radoux, C.J., Olsson, T.S., Pitt, W.R., Groom, C.R. and Blundell, T.L., 2016. Identifying interactions that determine fragment binding at protein hotspots. *Journal of medicinal chemistry*, *59*(9), pp.4314-4325.