Alexander Powers

Predicting Protein Flexibility for Better Drug Docking

Introduction

It is of critical interest for drug development to predict the drug "binding pose", the three-dimensional interaction between a drug molecule and a protein molecule. Using the binding pose information, we can design more effective drugs with less side effects. We can also use pose predictors to virtually screen large databases of potential drugs. As put by Gschwend et.al. "Fueled by advances in molecular structure determination, tools for structure-based drug design are proliferating rapidly. Lead discovery through searching of ligand databases with molecular docking techniques represents an attractive alternative to high-throughout random screening." However, *pose prediction*, also known as *ligand docking*, is still a very challenging computational problem as both the drug molecule and protein molecule are flexible and together possess a nearly intractable number of degrees of freedom. We need to figure out if and how these two flexible molecules fit together accurately and efficiently.

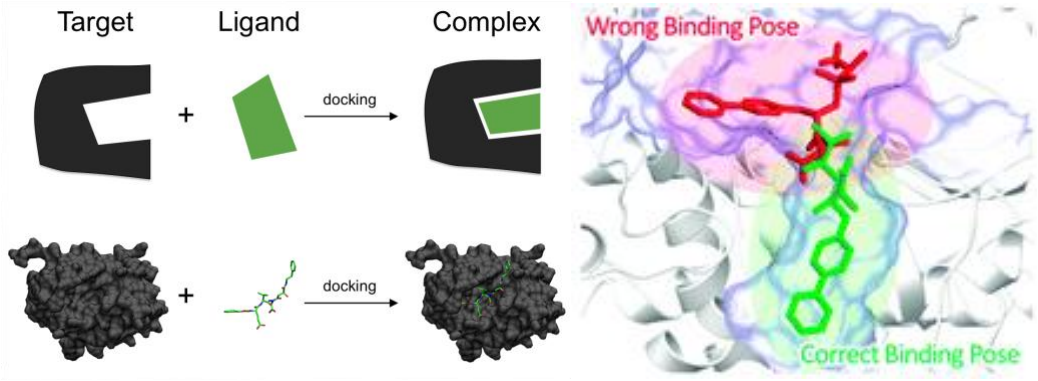


Figure 1. (Left) Basics of molecular docking (Wikipedia) (Right) Docking involves generating possible poses and determining which ones are more likely correct (Shota et.al.)

A naïve but still quite common approach to ligand docking is called "rigid receptor" docking. Here the protein atoms are held fixed, and many possible poses of the flexible ligand (or drug molecule) are sampled. A function is then used to score each of these poses, and determine which are most likely. Based on common benchmarks, the ranking of poses is only rarely correct (30-45% of the time top pose is correct) and at least one correct pose is only generated about 60-70% of the time. Thus, there is real room left for improvement in ligand docking both in scoring and sampling. One key reason for failure is flexibility of the protein binding site. The proteins are capable of moving to adapt to each particular drug molecule. The protein structure used for docking however, may have been determined with a different drug molecule initially or no drug molecule at all. Thus, the protein is not *poised* to fit with even the correct geometry of the ligand (Figure 2).

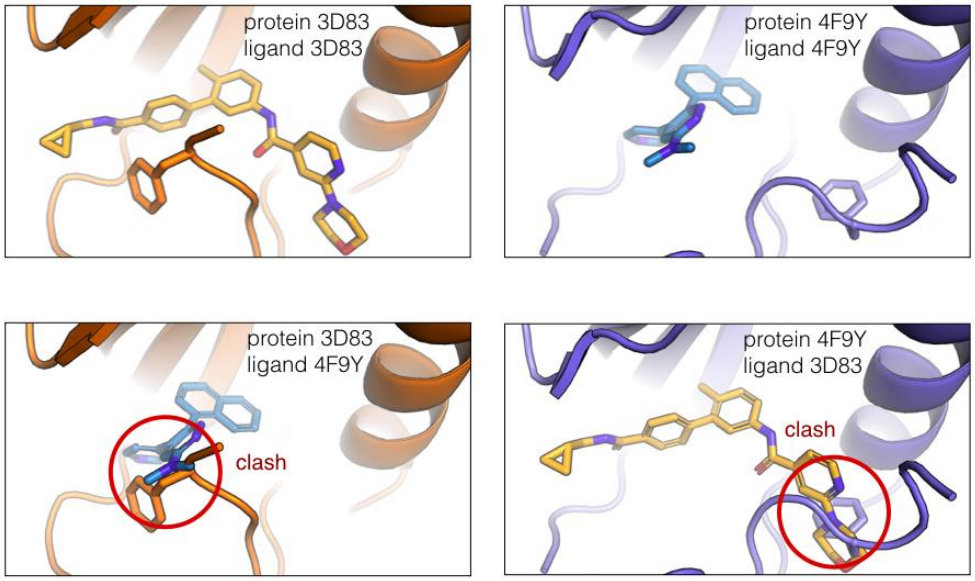


Figure 2. Why flexibility of proteins is an issue. On top, we are showing two different structures of the same protein (a kinase) bound to two different drug molecules (orange and blue). If we try to put blue ligand into orange structure it doesn't fit right (lower left)! Likewise, the orange drug doesn't fit into the blue structure (lower right). Note that only part of the protein structure is moving (the ring on the loop), the rest is more rigid. Can we predict ahead of time that the loop is flexible?

I am interested in the question of whether we can predict the *location* of protein flexibility. The protein can be divided up into a chain of different amino acids or residues. Specifically, can we accurately predict whether a protein residue is going to "move" when binding one ligand vs. another? This is an essential first step for any docking algorithm that incorporates flexibility. Some programs just make all residues potentially flexible, which is highly inefficient. We could narrow search problem significantly with accurate flexibility predictions. Predicting flexibility could help with structure-based drug design. We could identify parts of binding pocket that are rigid and part that could move (hidden pockets?) to accommodate drug alterations.

We will employ a graph machine learning approach to predicting flexibility based on residues level features and their spatial arrangement in relation to each other. We will rely on datasets of structures where we can determine which residues moved and which didn't (like Fig. 2). In particular, the residues or even atoms of a protein can be represented as a graph where edges are distances in 3D space. I hypothesize that this spatial information should help in prediction of flexibility. Rigid residues should be surrounded by other rigid residues, and flexible residues near other flexible residues. Thus, the surrounding nodes in our molecular graph should provide enhanced predictive power compared with only predicting node flexibility using isolated node features. We will compare the results of predicting node label just on isolated node features to those of a graph neural network that uses both node features and spatial relationship to other nodes.

Related Work

Previous work has mostly focused on the use of algorithms, including graph approaches, for scoring of ligand protein complexes. Graph approaches have also been used to predict more basic chemical properties from molecular graphs. I haven't found anyone who used graph neural networks (or any other machine learning) to specifically predict flexibility of protein structures.

Jaechang et. al. tries to predict the 3D structure of a drug+protein complex using a graph neural network. They use the list of ligand poses generated by a commercial sampling algorithm, then try to score these poses to find the best one. First, two graphs are constructed from distances and bonds between pairs of atoms. One graph has a node for every atom and an edge where a pair of atoms have a covalent bond. The second graph constructs additional edges between the drug and protein atoms if the distance is below a cutoff (5 angstroms). They also use a graph attention mechanism, though it's not clear from the paper whether this actually improves the results, because it isn't compared to a non-attention mechanism approach. This paper showed only a 5-7% improvement in the scoring accuracy for protein+drug poses, leaving a lot of room for improvement.

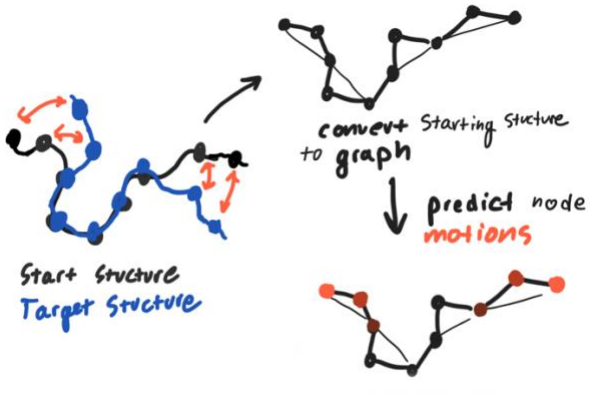
Feinberg et. al. propose a more general approach using graph neural networks to predict various molecular properties on protein+drug complexes. This goes beyond just trying to predict the 3D structure of a protein+drug complex, to predicting quantum properties (energy), toxicity, and solubility. This paper's approach also uses an adjacency matrix based on both covalent bond and non-covalent interactions between ligand and proteins. Feinberg et. al. also comment on training and dataset preparation. Apparently, due to lack of data, many projects search hyperparameters directly on the test set. Fienberg et. al. uses a validation set to do hyperparameter search across 9 parameters including number of convolution layers and gather widths. One problem from this paper is that many different types of predictions are made, but the same model architecture is generally used. It would be useful to know whether different model architectures are more applicable to different types of molecular problems; for example, if one is considering a drug+protein vs. the drug alone.

Schutt et. al. uses a "deep tensor neural network" to predict molecular properties. The input to this model is a distance matrix across all pair of atoms as well as a vector of charges, the length of the number of atoms. One thing that's done differently here is that the distances are expanded in a Gaussian basis, which converts each distance into a vector of weights. This may help the learning process. Each atom is given a feature vector; during a calculation, these features are updated by pairwise interactions with each neighbor atom. The feature vector for the atoms is then summed to predict a final molecular property, like energy.

Model/Algorithm/Method

Exact Problem Statement

We are using pairs of (3D) molecular structures of *same protein* bound to *different ligands.* The first starting structure is transformed into a graph where each node is an amino acid and edges are based on the 3d distances. We want to predict which amino acids will move (in 3D space) between the starting protein structure and the second protein structure. The movement is calculated as root mean squared displacement (RMSD) in angstroms, and we are trying to do binary classification on the nodes as having RMSD > 2 angstroms.



Raw Data and Processing

We are using a dataset of molecular structures where we have the same protein bound to different drugs/ligands (Table 1). These are obtained from public database (PDB). This set of targets and structures were prepared through a combination of manual curation and adaptation of the PDBbind refined set. We did not consider structures with duplicate ligands, mutant proteins, or no ligands at all. All complexes were prepared in the presence of the ligand using the Schrodinger software. All waters were removed, ligand states were assigned using Epik at pH 7.0 +/- 2.0, hydrogen bonds were optimized, and all atoms were minimized with non-hydrogen atoms constrained to an RMSD of 0.3 Å.

We then divided the raw data into pairs of (3D) molecular structures of *same protein* bound to *different ligands.* We have then calculated motion between these pairs of structures. Specifically, we aligned the structures, and then calculated how much each amino acid had moved.

Table 1: Structural and binding data. Protein family, gene name, Uniprot ID, ChEMBL target ID, number of ligands

PROTEIN FAMILY	PROTEIN	UNIPROT	CHEMBL	# LIGANDS
GPCR	5-HT_{2B}	P41595	CHEMBL1833	5
	β_1AR	P07700	CHEMBL213	11

	β_2AR	P07550	CHEMBL210	7
	mGluR5	P41594	CHEMBL2564	4
	Smo	Q99835	CHEMBL5971	4
TRANSPORTER	DAT	Q7K4Y6	CHEMBL238	8
	SERT	P31645	CHEMBL228	4
	GLUT1	P11166	CHEMBL2535	2
NUCLEAR RECEPTOR	ER	P03372	CHEMBL206	20
	GR	P04150	CHEMBL2034	16
	MR	P08235	CHEMBL1994	12
	AR	P10275	CHEMBL1871	19
	VDR	P11473	CHEMBL1977	20
PROTEASE	F2	P00734	CHEMBL3856166	20
	F10	P00742	CHEMBL244	20
	uPA	P00749	CHEMBL3286	20
	CT	P00760	CHEMBL3769	20
	BACE1	P56817	CHEMBL4822	20
PHOSPHORYLASE	PYGM	P00489	CHEMBL352	20
PHOSPHATASE	PTPN1	P18031	CHEMBL335	20
TRANSCRIPTION FACTOR	BRD4	O60885	CHEMBL1163125	16
CHAPERONE	HSP90-α	P07900	CHEMBL3880	20
PHOSPHO-DIESTERASE	PDE10A	Q9Y233	CHEMBL4409	20
RECEPTOR	σ_1	Q99720	CHEMBL287	4
ION CHANNEL	TrpV1	O35433	CHEMBL5102	2
ELATASE	ELANE	P08246	CHEMBL248	8
REDUCTASE	DHFR	P00374	CHEMBL202	20
KINASE	Cdk2	P24941	CHEMBL301	20

Transforming to Graph and Calculating Node Features and Labels

I've written python scripts to calculate these features and the graph connectivity. Six features were calculated for each node/amino acid. We only used amino acids that were in the close vicinity of the drug/ligand molecule. The node level features included:

- The molecular weight and volume of the amino acid
- B-factors (these are part of experimental data from a structure determined using X-ray crystallography, B-factors relate to how diffuse the signal is at that atom and possibly how flexible it is).
- The solvent accessibility (amino acids at the surface of protein are likely to be more flexible than those deep in the core)
- Information about the ligand size and how different it is from the target ligand (these are the same for all nodes in a given graph).

To create the weighted graph, I computed distance between each pair of amino acids (using centroid of atoms of each amino acid). If the distance was below a certain threshold, I added an

edge between those nodes with the weight being the distance. I chose to not include edges between all nodes to cut down on memory and training time.

For the prediction task, I simplified 'flexibility' into a binary classification problem. For each amino acid (each node in graph) in the binding site, I calculated the rmsd between starting protein structure and target protein structure. E.g. did that amino acid move when protein was bound to starting ligand vs. target ligand. The rmsd is the root mean square deviation of the position of each atom. The pair of structures need to be appropriately aligned in 3D space to calculate this and I used a standard method to for this. We then used 2 angstroms as a cutoff to classify whether the amino acid "moves a lot" or "moves a little". This might make prediction task easier. We don't care as much if some moves 10 angstroms vs. 15 angstroms; we just care that it moved significantly.

GNN Model

As the graphs aren't very large (usually 20-30 nodes), I chose to use a very simple preliminary architecture based on some of the examples in pyTorch Geometric. I was more interested in seeing if this would work, and then possibly expanding with a more complex architecture. The GNN has two layers, each with a graph convolutional operator shown below where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$. Here X is the activations of the previous layer and X' is the activations after the operator is applied. Θ is the learnable weight matrix. This layer is a fast approximate convolution that is a first-order approximation of localized spectral filters on graphs (Kipf and Welling, 2017).

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta},$$

The first layer operator takes 6 node features and transforms to 16 output features. We then have a relu activation. The second layer takes 16 features and transforms to 1 feature. A sigmoidal activation is then applied to get a scalar between 0 and 1. For the prediction, this is then rounded to 0 or 1.

For training I used 60% of data for training and 40% for testing. I used different proteins in the testing data compared with the training data also, not just different graphs. I tested both L1 and MSE loss.

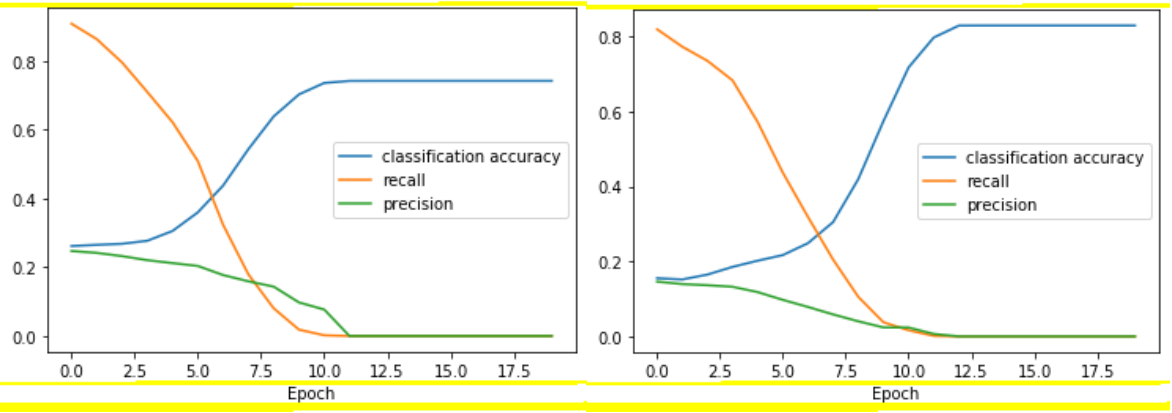
As a control, I'm using some baseline models that operate on nodes independently: decision trees and linear regression in sklearn.

Results and findings

Preliminary results on GNN

I was able to successfully create and train the model on a small dataset (1000 graphs, 600 used for training, 400 for test). Unfortunately, preliminary results were not good. Although the binary classification accuracy went up quickly, the recall and precision both went to 0. Essentially the network was just predicting all the classes as 0, which still achieves an accuracy around 80%. This demonstrates an imbalance in the data and that the loss function isn't working so well.

Train **Test**



Progress: Exploration of data

We wanted to know about how prevalent motion of the protein was at the ligand binding site using our pairs of structures bound to different ligands. We calculated the deviation of the residues in the binding site for every pair of structures (Figure 3). The graph below plots the average number of residues with a deviation greater than 2 anstroms for each protein. On average 12.4% of the residues in a protein's binding site will be flexible during docking. This indicates that there is a *pretty big class imbalance* between flexible and not flexible (most residues are going to be not very flexible).

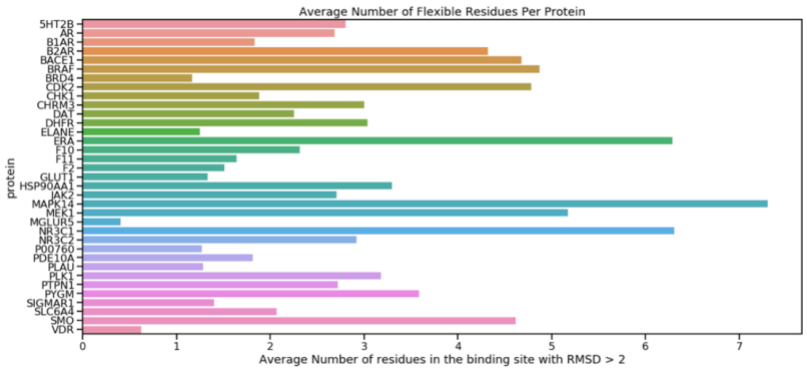


Figure 3.

Baseline Models

- Using only node level features we made some baseline predictions.
- We also investigated the importance of the node level features.
- To correct for the class imbalance between flexible and nonflexible residue cases, we up sampled the positive examples for training data (about 5x to yield class balance of 1:2). We could play around a bit more with this and other approaches for unbalanced data. For test set, retained natural balance of data.
- We did a 70:30 split for training and testing.
- We did classification for an easily interpretable baseline, classifying a residue as flexible if it moved by more than 2 angstroms between the initial and target structure.

Performance Statistics

	Accuracy	Recall	Precision
Linear Regression	0.625	0.610	0.187
Polynomial Regression	0.670	0.571	0.191
Decision Tree	0.722	0.524	0.195

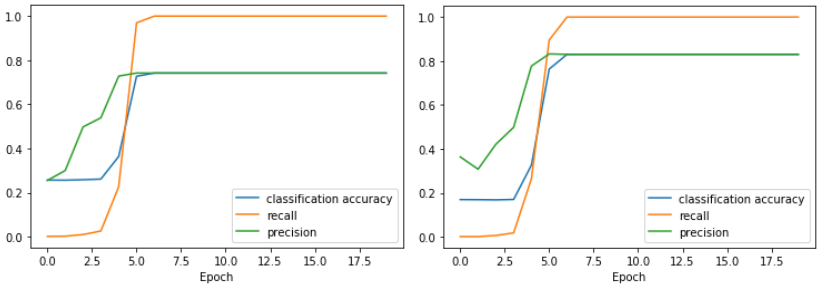
Feature Vector Importance

Linear Regression		***Decision Tree***	
Ligand similarity ratio	0.966	Normalized b-factor	0.199
Normalized b-factor	0.473	Solvent accessibility	0.118
Secondary structure	0.359	General Average RMSD of rotamers	0.100
Next residue's b-factor	0.311	Raw b-factor	0.091
Specific Average RMSD of rotamers	0.230	Next residue's b-factor	0.065

Changing class weighting for GNN

We needed a way to correct for the imbalance in the data for the graph neural networks. I showed for simpler classifiers on only node level features, we could resample the positive examples in the data set to achieve better recall and precision at the expense of some accuracy. In fact for decision trees, we could get the recall to 52% and the accuracy to 72%, which isn't bad start.

To improve the graph neural networks, we can't just resample positive nodes as this would disrupt the network structure. I think to fix this in the future, I instead outputed probability for 2 classes, then use the CrossEntropyLoss with a higher weight for the positive class. I tried 2x weight for positive examples with promising results (see below figures). The results had very high recall > 99% and a classification accuracy ~80%. This was slightly better than decision trees perhaps demonstrating effective use of spatial information. However, I'm a bit concerned that in fact the recall is too good here so I may have made a mistake in how I partitioned the data into train and test (I may have included same proteins in both if ordering of proteins was not preserved by PyTorch Data module). I will be further investigating this in future work.



Train **Test**

References

1. Gschwend, D. A., Good, A. C. & Kuntz, I. D. Molecular docking towards drug discovery. *J. Mol. Recognit.* 175–86 (1996). doi:10.1002/(SICI)1099-1352(199603)9:2<175::AID-JMR260>3.0.CO;2-D

2. Shota Uehara, Kazuhiro J. Fujimoto, Shigenori Tanaka. Protein–ligand docking using fitness learning-based artificial bee colony with proximity stimuli. *Phys. Chem. Chem. Phys.*, 2015,17, 16412-16417

3. Lim Jaechang, Seongok Ryu et. al. Predicting drug-target interaction using 3D structure-embedded graph representations from graph neural networks. KAIST, South Korea. 2019. https://arxiv.org/pdf/1904.08144.pdf

4. Evan N. Feinberg, Debnil Sur et. al. PotentialNet for Molecular Property Prediction. Stanford, USA. 2018. https://arxiv.org/pdf/1803.04465.pdf

5. Schütt et. al. Quantum-chemical insights from deep tensor neural networks. Nature Communications. Technische Universität Berlin, Germany. 2017. https://www.ncbi.nlm.nih.gov/pubmed/28067221