Iterative Knowledge-Based Protein-Ligand Scoring Function: Implementation and Improvements

Thanh Lai

I. Backgrounds, goals, and significance

Protein-ligand binding is a process that occurs in all biological scope—from endogenous ligand-receptor binding for signal transduction, to selective drugging of a protein target. Therefore, it is of utmost importance to understand the thermodynamics and kinetics that govern such process, as doing so has implications for fields such as drug discovery. One can gauge the stability of a protein-ligand complex by considering only the thermodynamics of the binding, however this consideration has multiple levels of theory, from empirically-derived scoring functions to *ab inito* calculations. In the early stages of a computational drug discovery program, scoring functions are favored as a quick method of filtering thousands of protein-ligand (ligand = drug candidate) complexes. However, as scoring functions trade accuracy for computing speed, it is not far-fetched to say that the trajectory of a drug discovery program greatly hinges on identifying strong drug candidates from a pool of several thousands. Therefore, there is a need to improve on scoring functions as it has the ability to shape the trajectory of these drug discovery programs.

There are four classifications of scoring functions: physics-based, empirical-based, descriptor-based, and knowledge-based.[1] Physics-based scoring functions generally score protein-ligand complexes by the summation of the following energy terms: van der Waals, electrostatic, hydrogen bonding, and the free energy of de-solvation. This scoring type is limited by the approximations in some of the energy terms and neglects or poorly describes the entropic contribution. Descriptor-based scoring functions are machine learning models that are trained to predict either a binary (real vs. decoy) or a regression (free energy) result. While descriptor-based scoring functions are known to be accurate, they require massive training data and is considered as a "black box" method. Empirical-based scoring functions implements intuitive reward and penalty scores based on interaction patterns (hydrophobicity, hydrogen bonding, etc.). As the method is based on the summation of terms describing specific interaction patterns, it is difficult to capture all forms of protein-ligand interactions. Lastly, knowledge-based scoring is a statistical approach that creates pairwise interaction potentials by counting the number density of pairwise atoms relative to a reference. A large limitation of this approach is obtaining the reference state.[2]

The iterative knowledge-based protein ligand scoring function, ITScore,[3,4] avoids the calculation of the reference state by iteratively improving the trial protein-ligand pairwise atomic potentials until native complexes are discriminated from decoy complexes. The primary goal of this project is to replicate the algorithm on python. The open-source python module Biopython[5] will be used for most of the algorithm implementation. Biopython's ease of use will ensure the success of the project, as it makes it easy to iterate through atoms in a PDB file. The replicated code base will be trained and tested on the same PDB set that is used by Huang et. Al.[3] If time permits, the model will use a larger training set with a stricter convergence criterion, and the code will be parallelized. These changes will improve the model since the model is very dependent on the quality of the training PDBs. Parallelizing the code is mainly for fun/bonus but could also be useful if the model ever needs to be retrained on a better/larger PDB set in the future.

The success of this project will have several implications. Firstly, replicating the code base means the code for the algorithm is now open source and can be readily modified in the future. Improving the model by the ideas mentioned above could be very useful for computational drug discovery programs. As mentioned previously, these programs screen thousands of potential drug candidates through docking and scoring. An improved scoring function, even if only marginal, could be the difference between finding a truly effective drug molecule or discarding it under the pretense of inactivity.

II. Datasets

The initial training and testing dataset will be from the list of protein-ligand crystal structures curated by Huang et. Al (formatted as PDB files from the RCSB database[6]). The selected PDBs fulfills the following criteria: the resolution is greater than 2.5 Å; the protein-ligand interaction is noncovalent and the ligand is not DNA, RNA, or peptide (for information on scoring functions for these kinds of systems, please see future publications by Huang et. Al.); the ligands must contain between 5 to 66 heavy atoms; the crystallization must be at normal pH conditions (6.5 to 7.5); the structure must contain no metal ions other than Na+ and K+ near the binding site; and finally, complex must be absent of steric clashes (<1.75 Å for all protein-ligand atoms). The curated dataset contains a total of 786 complexes. As of February 14th 2021, 677 out of 786 of these complexes are still present in the RCSB database and will be used for this study. The project will further evaluate the scoring function on a test set of 77 complexes curated by Muegge and Martin from the Brookhaven Protein Data Bank.[7] If time permits, the extensive PDBbind[8] dataset will be used to train the scoring function. The dataset is a comprehensive collection (>5000) of high-quality protein-ligand crystal structures formatted as PDB files. These complexes are associated with binding affinity data (K_d, K_i, or IC_{50}) and is often used as training data for various models such as regression machine learning. Therefore, the reliability and quality of this dataset sufficient to train the scoring function.

Briefly, the PDB data format mainly contains four key information: comments (REMARK, HETNAM, etc.), protein atoms (ATOM), ligand/cofactor/solvent atoms (HETATM), and connectivity information (CONNECT). The structure is described in a hierarchy with four levels: model, chain, residue, atom. There is usually only one model in a file, but NMR studies could produce multiple models owing to the slight conformational changes of the structure. A chain describes a molecule or a grouping of atoms. For example, a common file would consist of the following chains: protein (can be broken up into further chains if multimeric), bound peptides, bound cofactors, bound ligand, surrounding ions, solvent. The next level in the hierarchy is the residue description, in which atoms are grouped into amino acids, water, or user-defined names (these names are often defined at the beginning of the file and usually describe ligand atoms). The final level describes a singular atom. Each atom has an (X,Y,Z) coordinate, its element, the residue label it is associated with, the atom number, the residue number, and an identifier that is unique to the residue it belongs in.

III. Computational methods/approaches

The main idea of ITScore is as follows:[3] from a set of protein-ligand complexes, calculate the experimentally observed pair distribution function, $g^{obs}_{ij}(r)$, for each ligand atom protein atom pair:

$$g_{ij}^{obs}(r) = \frac{\rho_{ij}^{obs}(r)}{\rho_{ij,bulk}^{obs}} \quad (1)$$

$g^{obs}_{ij}(r)$ is the ratio between the number density of a (i, j) ligand-protein atom pair at a radial shell of r ± dr (figure A1) and the total number density of (i, j) ligand-protein atom pair within a sphere of radius R. For this procedure dr is 0.1 Å and R is 10 Å. All complexes in the training dataset are used for the calculation of $g^{obs}_{ij}(r)$. Biopython's NeighborSearch class is used to count the number densities. A dictionary is used for the $g^{obs}_{ij}(r)$ function, where the key is r, and the value is the corresponding pair distribution. The set of $g^{obs}_{ij}(r)$ functions is stored by another dictionary, where the key is a (ligand atom i, protein atom j) tuple, and the value is the corresponding $g^{obs}_{ij}(r)$ dictionary. The dictionary data structure is chosen for its fast O(1) look up time complexity.

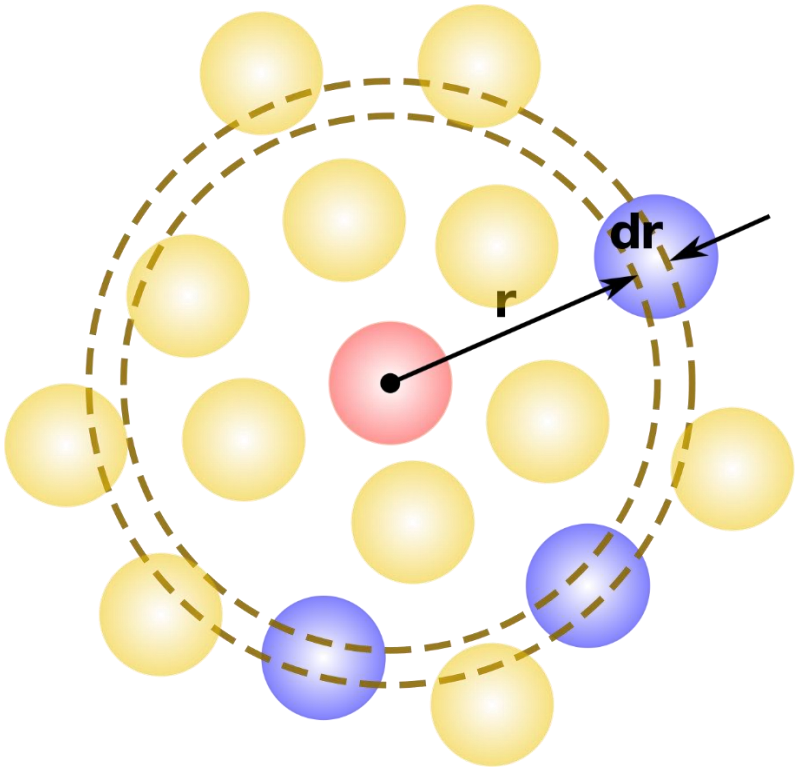

Figure A1. The number of the ligand atom (red) and protein atom (yellow) pairs are counted for each r ± dr radial shell (image taken from Wikipedia)[9].

To begin the iterative procedure, an initial pairwise interaction potential for all atom pairs must be created. Only atom pairs whose occurrences are greater than 500 within the sphere of radius R is considered. The "potential of mean force function" (note: not truly a PMF but is still referred as so by literature) for each ligand protein atom pair is calculated. The initial pairwise potentials are calculated by weighing the PMF function with the respective VDW radii from AMBER forcefields. Huang et. Al. did not specify the forcefield, but this project uses the parameters from ff14SB[10] (which "inherit" from AMBER's parm10.dat[11]). Once again, a dictionary is used to store the initial pairwise potentials.

Next, through a docking program (this project will employ AutoDock Vina[12]), a set (or "ensemble") of 200 ligand-protein poses for each complex are generated. These "decoy" complexes are used to iteratively refine the initial pairwise potentials. Boltzmann-weighted pair distribution functions for each ligand protein atom pairs are calculated for all poses (essentially similar functional form to the pair distribution function described beforehand except scaled by a Boltzmann factor). At the beginning, the Boltzmann factor uses the initial potential to calculate the energy, but in future iterations it uses the trial/updated potentials. During the iterative refinement step, the current trial interaction potentials at the *nth* iteration are used to determine the best ligand-protein complexes from the training set (decoys + natives). If the success rate fails to meet a threshold, that is the current potentials describe the decoy complexes as being more stable than the native complexes, the Boltzmann-weighted distribution function is recalculated, the trial potentials are updated based on the distribution function, and the updated trial potentials are used to predict the best complexes from the training set. This procedure repeats until the success rate meets a convergence criterion. The procedure is outlined in figure B2.

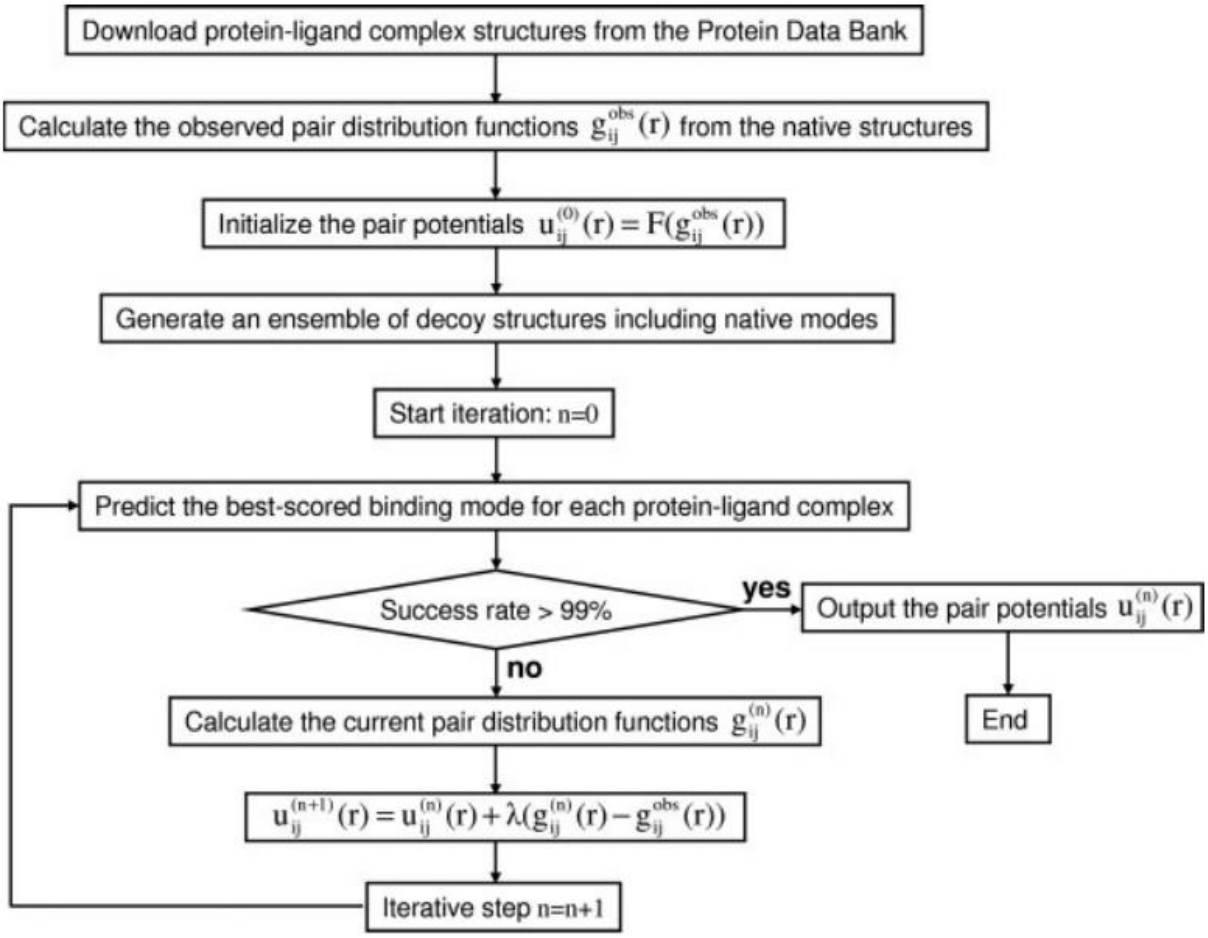

Figure B2. Flowchart of the entire procedure.[3]

The potentials at each iteration (contained within a dictionary) are serialized by the Pickle module since it offers an easy, fast, and efficient manner of data storage. These potentials could easily be read back into a python program for further iterative refinement by invoking Pickle's load method. Additionally, Pickle files has easy transportability, which means they can be sent from a high-performance computing server (where the potentials are created and refined) to a user's PC (where the potentials are analyzed and used in docking programs).

If time permits, there are several improvements to the training procedure. More decoy complexes can be added using the "Blur" method by the Merz group at the MSU Chemistry department.[13] Starting from bound ligand pose, this method can generate an ensemble of alternative ligand poses in the binding pocket by small perturbations in the starting position. For example, a ligand's bonds can be rotated in 15-degree increments, its center of mass can be translated slightly, and it can be rotated. While ligands in the decoy complexes generated by a docking program tend to have different poses than the native complexes (RMSD > 1.0), it is expected that these "blurred" decoys will be very similar to the poses of native complexes. Therefore, the training process becomes more rigorous as the potentials are iteratively improved to discriminate between complexes with very similar ligand poses. The convergence criterion will become stricter.

IV. Evaluation Plan

There are two methods of evaluating the results: firstly, Huang et. Al. provided the graph of several pairwise potentials. The refined pairwise potentials from this project's implementation can be visually compared to these graphs. Secondly, the minima of the refined pairwise potentials should be qualitatively reasonable; for example, C2X-C2X pair interactions should have a shorter equilibrium point and a deeper relatively to other hydrophobic interactions due to their favorable aromatic carbons.[14] Finally, the pairwise potentials can be quantitatively assessed by computing the energy scores for protein-ligand complexes with known binding energy scores. ITScore defines the energy score of a protein ligand complex to be the summation of all protein-ligand atom pair potential energies found by the refined pairwise potentials. While the potentials are not expected to yield accurate binding energies, the energies should be correlated with the experimental binding energies. It is reported that ITScore's has a correlation of 0.81 for a test set of 77 complexes.[3]

V. Potential Challenges and Alternative Approaches

Due to the limited time (~2.5 months), generating 200 docked complexes for all 677 PDB structure may not be feasible. However, it is worth mentioning that the main goal of this project is to implement the code base for ITScore. Therefore, as long as the python code can replicate the iterative procedure, there is no rush to train the potentials to the same degree as Huang et. Al. The potentials will be initially trained on a decoy set generated from 5 docked complexes for each PDB structure to show proof of concept. "Rougher" potentials will result from this weaker training process, and the evaluation process will shift to a qualitative basis accordingly (e.g., are the shapes of the potentials similar to the classic Lennard-Jones potentials? Do the wells make physical/chemical sense?). The potentials will be trained on the full 200 * 677 decoy complexes, and then trained on the decoys produced by the "Blur" methodology as well as the PDBbind complexes.

Another concern is the "Blur" methodology. Since the method generate an ensemble of bound ligands by very small perturbations in the ligand's native pose, there is a chance that the decoy and native ligand poses produces the same pairwise number densities and thus cannot be discriminated from. To resolve this, only decoy poses that has an RMSD higher than a certain threshold can be used for the training process. Therefore, the "Blur" methodology will be used to search for ligand poses that are different enough to be discriminated from the native poses, but not radically different like some of the poses generated by docking. In short, the combination of "Blur" and traditional docking by AutoDock Vina should produce a set of decoys that cover a large spectrum of ligand poses.

The final—and most important—concern is the labeling of certain atoms in the PDB structures. Huang et. Al. uses the SYBYL convention (Tripos inc.), which has identifiers for common atom "motifs". For example, CF3 are sp^3 hybridized carbons bonded only to carbon or hydrogen, and O31 are oxygens bonded to exactly one non-hydrogen atom. However, the ligands in PDB files often do not follow SYBYL convention. For example, if acetic acid were to be a ligand in the PDB file, its carbons would simply be identified as "C1, C2, C3" instead of accounting for its hybridizations, etc., like SYBYL. There is a file format called "mol2" that follows SYBYL but it is not supported by Biopython. The lack of SYBYL identifiers in PDB files results in "nonsense" counting of atom pairs. Preliminary testing with Biopython on the 677 PDB files generated over 2000 atom pairs whose occurrences are greater than 500 in the sphere of radius R = 10 angstroms. Meanwhile, Huang et. Al. identified only 26 atom pairs by using the SYBYL convention. The vast majority of the 2000 atom pairs generated by Biopython are arbitrary groupings such as ("CA", "C2") which describes the very specific interaction between the first alpha carbon of a residue and the second carbon of a ligand. This pair holds zero chemical information, and its initial potential curve looks to be a messy combination of different ("C", "C") pair potentials. A potential solution of obtaining the SYBYL identifiers is to convert all PDB files into their corresponding mol2 files. Then, for each PDB

and mol2 file pair, change the identifiers of each atom in the PDB file with the corresponding SYBYL identifier in the mol2 file.

VI. Specific Milestones

1. The code up to the iterative portion of the procedure has been implemented already (2/14/21)
2. Five decoy complexes will be generated for each PDB file by a simple and automatable docking program such as AutoDock using rigid docking, organize AMBER VDW parameters in an accessible Python format and associate it with the atom types in the PDB files (week of 2/22/21)
3. The "skeletal" code for the iterative procedure will be written. The skeletal code will be a nested loop to show that the program can access files in different folders, contains conditional statements, etc. without specific implementations such as energy score calculation (week of 2/22/21 to 3/1/21)
4. Write the function to calculate the pairwise distribution function at the *nth* iteration (week of 3/1/21)
5. Write the function to the calculate the energy score, the function to update the trial potentials (week of 3/1/21 to 3/8/21)
6. Write the convergence function (week of 3/8/21)
7. Add in the functions to the "skeletal" code and check if it works for a "small scale" training procedure (week of 3/8/21)
8. Convert the atom identifiers of all PDB files to SYBYL identifiers (week of 3/15/21)
9. With the correct atom identifiers, train the potentials on the full dataset and hope decent-looking potentials are created (week of 3/15/21 to 3/22/21)
10. Obtain the in-house "Blur" code written by the Merz group and learn how to use it. If not possible, then replicate the "Blur" code using the built-in translation and rotation functionality of Biopython (week of 3/22/21)
11. Train the potentials on the decoys generated by the "Blur" code (3/29/21)
12. The rest of the time in April will be used accordingly, either to catch up on the schedule, do larger training steps to improve the potentials, or parallelize the code.

VII. References

(1) Liu, J.; Wang, R. Classification of Current Scoring Functions. *J. Chem. Inf. Model.* **2015**, *55* (3), 475–482. https://doi.org/10.1021/ci500731a.

(2) Thomas, P. D.; Dill, K. A. Statistical Potentials Extracted from Protein Structures: How Accurate Are They? *J. Mol. Biol.* **1996**, *257* (2), 457–469. https://doi.org/10.1006/jmbi.1996.0175.

(3) Huang, S.-Y.; Zou, X. An Iterative Knowledge-Based Scoring Function to Predict Protein–Ligand Interactions: I. Derivation of Interaction Potentials. *J. Comput. Chem.* **2006**, *27* (15), 1866–1875. https://doi.org/10.1002/jcc.20504.

(4) Huang, S.-Y.; Zou, X. An Iterative Knowledge-Based Scoring Function to Predict Protein–Ligand Interactions: II. Validation of the Scoring Function. *J. Comput. Chem.* **2006**, *27* (15), 1876–1882. https://doi.org/10.1002/jcc.20505.

(5) Cock, P. J. A.; Antao, T.; Chang, J. T.; Chapman, B. A.; Cox, C. J.; Dalke, A.; Friedberg, I.; Hamelryck, T.; Kauff, F.; Wilczynski, B.; De Hoon, M. J. L. Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics. *Bioinformatics* **2009**, *25* (11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163.

(6) Berman, H.; Henrick, K.; Nakamura, H. Announcing the Worldwide Protein Data Bank. *Nat. Struct. Mol. Biol.* **2003**, *10* (12), 980–980. https://doi.org/10.1038/nsb1203-980.

(7) Muegge, I.; Martin, Y. C. A General and Fast Scoring Function for Protein−Ligand Interactions: A Simplified Potential Approach. *J. Med. Chem.* **1999**, *42* (5), 791–804. https://doi.org/10.1021/jm980536j.

(8) Wang, R.; Fang, X.; Lu, Y.; Wang, S. The PDBbind Database: Collection of Binding Affinities for Protein−Ligand Complexes with Known Three-Dimensional Structures. *J. Med. Chem.* **2004**, *47* (12), 2977–2980. https://doi.org/10.1021/jm030580l.

(9) Radial Distribution Function https://en.wikipedia.org/wiki/Radial_distribution_function.

(10) Maier, J. A.; Martinez, C.; Kasavajhala, K.; Wickstrom, L.; Hauser, K. E.; Simmerling, C. Ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from Ff99SB. *J. Chem. Theory Comput.* **2015**, *11* (8), 3696–3713. https://doi.org/10.1021/acs.jctc.5b00255.

(11) D.A. Case, D.S. Cerutti, T.E. Cheatham, III, T.A. Darden, R.E. Duke, T.J. Giese, H. Gohlke, A.W. Goetz, D.; Greene, N. Homeyer, S. Izadi, A. Kovalenko, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. L.; D. Mermelstein, K.M. Merz, G. Monard, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D.R. Roe, A.; Roitberg, C. Sagui, C.L. Simmerling, W.M. Botello-Smith, J. Swails, R.C. Walker, J. Wang, R.M. Wolf, X.; Wu, L. Xiao, D. M. Y. and P. A. K. Amber17. 2017.

(12) Trott, O.; Olson, A. J. AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading. *J. Comput. Chem.* **2009**, NA-NA. https://doi.org/10.1002/jcc.21334.

(13) Ucisik, M. N.; Zheng, Z.; Faver, J. C.; Merz, K. M. Bringing Clarity to the Prediction of Protein-Ligand Binding Free Energies via “Blurring.” *J. Chem. Theory Comput.* **2014**, *10* (3), 1314–1325. https://doi.org/10.1021/ct400995c.

(14) Burley, S.; Petsko, G. Aromatic-Aromatic Interaction: A Mechanism of Protein Structure Stabilization. *Science (80-.).* **1985**, *229* (4708), 23–28. https://doi.org/10.1126/science.3892686.