Bind-Pred - Theoretical base

Introduction

As a brief introduction, our team designed a residue-centered machine learning algorithm that aims to predict binding sites of a given query protein using a training set of homologous proteins of known binding site information. Our program has the ability to perceive residue environment information by considering the physicochemical properties of the atoms and the interactions between each residue of the protein and its structural neighbor residues.

The team decided to make individual predictions for each of the chains of the query protein, instead of making predictions using the whole structure. This is mainly due to the fact that the results of BLAST are local, hence the homologous sequences retrieved are only homologous chains to specific chains of the query protein.

The team opted to prioritize maintaining consistency when calculating the features of the homologous sequences and the query protein. If the homologous sequences were to be protein chains, it might have led to biases if the features of the protein were then to be considered using the whole protein. Thus, our team thought the best approach was to consider the query chain as different chains for which we wanted to make predictions on.

Our protein binding site prediction program is residue centered, for this reason it returns a list of residues that are part of the different binding sites distributed in the protein, which can be visualized from Chimera. Our program also searches for the query protein in the BioLip database to retrieve the real binding sites (in case they are annotated), and then it compares the predicted binding sites and the real ones to provide the metrics of the prediction and a Chimera script to better visualize the accuracy of the prediction.

The theoretical base behind our project has been mostly inspired by the GRaSP program[1,2]. Nevertheless, we tried our best to build the program by our own and give our point of view in every part of the code.

Outline

The main steps followed in order to build Bind-Pred are described as follows:

1. **Obtaining input data:** extracting query protein specified by the user and splitting it into chains.
2. **Homolog search:** performing a BLASTP search using the BioLip database for each chain. The program retrieves 40 homologous proteins (templates) with known binding site information for each chain.
3. **Templates feature extraction:** computing features of the templates (homologous proteins). Generation of the matrix of features used as input for the random forest

machine learning algorithm, and extracting the real binding site residues of the templates from BioLip database information.

4. **Machine Learning Model:** generation of training and test data and building model using Random Forest models using the training data (one per each chain). Model testing using test data not seen by the model and extraction of preliminar metrics.
5. **Query feature extraction:** computing features of the query protein (each chain independently).
6. **Binding site predictions:** Prediction of the binding sites of the query protein. Generation of a chimera script to select and color the binding sites residues.

1. Obtaining input data

The program starts reading the input protein specified by the user in pdb format (more details in *README.md* file) and generating separate chains in *fasta* format in order to perform a homolog search.

2. BLAST

BioLip Database

In order to search for possible homologous proteins to our target protein we decided to do a BLAST using the BioLip database. BioLiP is a semi-manually curated database for biologically relevant ligand–protein interactions. This database is the result of filtering all protein structures present in the PDB database, taking into account the biological relevance of the ligands. This involves a four-step biological feature filtering followed by careful manual verifications.

Each entry in BioLiP contains annotations on:

- Ligand-binding residues.
- Ligand-binding affinity.
- Catalytic sites.
- Enzyme Commission numbers.
- Gene Ontology terms.
- Cross-links to the other databases.

So, the main reason we used this database is to retrieve the protein templates to build a specific training data set for a particular query protein to be studied (see machine learning section), since all proteins contained in this database contain information about their binding sites.

For the purpose of the project, we decided to use the redundant version of BioLip, which contains all the sequences for each protein-ligand interaction. The team followed these steps in order to make use of this database:

1. We first managed to download all the information through the official BioLip website (http://zhanglab.ccmb.med.umich.edu/BioLiP/).

2. Then, making use of the command line, we used the function *makeblastdb* to create our own database using the downloaded BioLip information.

```
makeblastdb -in SBI_Project/BioLip_database/protein.fasta -dbtype prot
    -parse_seqids -out SBI_Project/BioLip_database/biolip_database
```

The output obtained consists in a series of files with different extensions that add up to make the whole database used in the program (check program folder to see database files). Once done, the database was ready to be used in a typical BLAST search.

3. Feature extraction

Our protein binding site prediction program is residue centered. This means that, in order to build our Machine Learning model, we calculated several features for each of the residues that are part of the protein. Moreover, it is considered the physicochemical environment of the residue, by computing the features of the structural neighbor residues and also the interactions that occur between them.

The features we decided to extract in order to build our Machine Learning algorithm can be split into 3 categories: residue-level, atom-level and interaction-level features.

Residue Level

At the residue level, our team calculated the following features: solvent accessibility, hydrophobicity and check if the particular residue is a cysteine.

About solvent accessibility, it is important to be considered when computing binding sites of proteins because it provides information about which parts of the protein structure are exposed to the solvent and therefore available for interaction with other molecules. Protein binding sites are typically composed of amino acid residues that have specific chemical properties that allow them to interact with other molecules, such as small molecules, other proteins, or nucleic acids. These residues must be accessible to the solvent in order to interact with other molecules, thus regions of high solvent accessibility are more likely to be involved in binding, while regions of low solvent accessibility are less likely to be involved in binding[3].

By considering the solvent accessibility of each amino acid residue in a protein, one can identify regions of the protein that are likely to be involved in binding interactions. Therefore, considering solvent accessibility when computing binding sites of proteins can improve the accuracy of the predictions and help identify the most likely binding sites.

Moreover, hydrophobicity is an important factor to consider when computing binding sites of proteins because it can affect the way that proteins interact with other molecules. Hydrophobic amino acids tend to cluster together in the interior of the protein structure, while hydrophilic amino acids tend to be exposed to the solvent[4]. Apart from this, hydrophobic amino acids in the binding site can interact with hydrophobic regions of ligand molecules, while hydrophilic amino acids can interact with hydrophilic regions of the ligand molecule.

Finally, it is important to consider whether a residue is a cysteine when computing binding sites of proteins because cysteine residues can form covalent bonds with other cysteine residues or with small molecules to form disulfide bonds. Disulfide bonds can stabilize protein structures and affect protein function. They also play a critical role in protein-protein interactions, as they can act as bridges between different protein subunits or between different regions of the same protein, which is interesting for binding site detection[5].

Atom Level

At the atom level, our team classified the atoms of each residue according to their physicochemical properties: the atom being a donor, acceptor, hydrophobic, aromatic, positively charged or negatively charged. It is worth saying that each atom can be classified in one or more types. The classification is stored in *atom_types.csv* and it is based on the classification made by *Fassio et al (2019)*[6].

In our program it is computed how many atoms ("count") of each type a residue presents. Moreover, this classification is used when computing the type of interactions between the atoms of different residues.

Interaction Level

The interactions between residues are calculated based on the type of atoms and on the Euclidean distance. These two considerations are needed because the interactions will occur depending on the proximity (distance) of the atoms and the atom physicochemical properties, and they will result in a certain type of interaction or another. Our program considered different types of interactions: aromatic stacking, disulfide bridge, hydrogen bond, hydrophobic, repulsive and attractive interaction.

The atom classification is based on *atom_types.csv,* like we did in the previous section.

The atom types and distances criteria to consider that exists an interaction between two residues are summarized in the *Table 1* extracted from GRASP.

Table 1. Distance criteria (in A) considered to compute interactions

Interaction Type	Atom Types	Distance	
		MIN	MAX
Aromatic stacking	2 aromatic atoms	1.5	3.5
Hydrogen bond	1 acceptor and 1 donor atom	2.0	3.0
Hydrophobic	2 hydrophobic atoms	2.0	3.8
Repulsive	2 atoms with the same charge	2.0	6.0
Salt Bridges	2 atoms with opposite charge	2.0	6.0

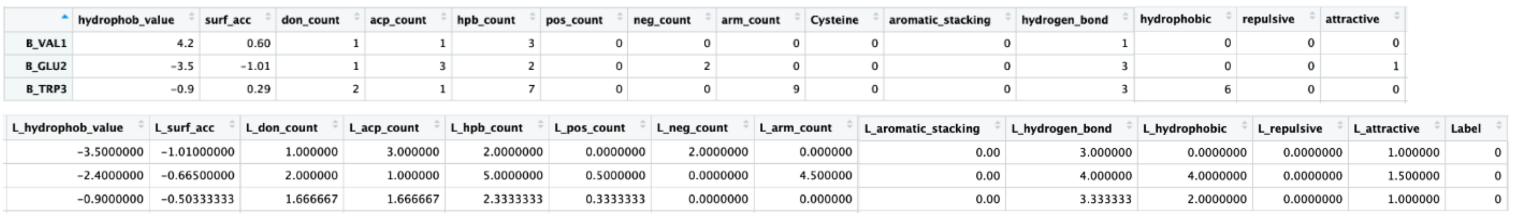
Layers

Our program is characterized by the fact it considers the structural neighbors of each residue, which allows us to perceive the residue environment information. To consider a certain residue a neighbor of a residue our program sets a threshold of maximum 8 A of distance between them[7].

The program will retrieve all the neighbors of each residue of the query protein and will compute the same properties previously mentioned (at residue level and atom level), except for the feature of checking whether the residue is a cysteine. Then it will do the average of the features of all the neighbors of the certain residue.

Once it is all computed, it will result in a dataframe with all the residues of the chain of the protein as records (rows) and 27 features as columns. Here we give an example of the features computed (*Table 2*):

Table 2. Three first rows of a dataframe of features used as input for random forest model and their label.



	hydrophob_value	surf_acc	don_count	acp_count	hpb_count	pos_count	neg_count	arm_count	Cysteine	aromatic_stacking	hydrogen_bond	hydrophobic	repulsive	attractive
B_VAL1	4.2	0.60	1	1	3	0	0	0	0	0	1	0	0	0
B_GLU2	-3.5	-1.01	1	3	2	0	2	0	0	0	3	0	0	1
B_TRP3	-0.9	0.29	2	1	7	0	0	9	0	0	3	6	0	0

L_hydrophob_value	L_surf_acc	L_don_count	L_acp_count	L_hpb_count	L_pos_count	L_neg_count	L_arm_count	L_aromatic_stacking	L_hydrogen_bond	L_hydrophobic	L_repulsive	L_attractive	Label
-3.5000000	-1.01000000	1.000000	3.000000	2.0000000	0.0000000	2.0000000	0.000000	0.00	3.000000	0.0000000	0.0000000	1.000000	0
-2.4000000	-0.66500000	2.000000	1.000000	5.0000000	0.5000000	0.0000000	4.500000	0.00	4.000000	4.0000000	0.0000000	1.500000	0
-0.9000000	-0.50333333	1.666667	1.666667	2.3333333	0.3333333	0.0000000	0.000000	0.00	3.333333	2.0000000	0.0000000	1.000000	0

In *Table 2* the first 14 features are features of the residue being considered, and the next 13 features are the average of the features of the first shell of neighbors of residues. The last column is the label that indicates if the residue is part of the binding site (1) or not (0), as it is supervised learning.

4. Machine Learning Model

The 27 features calculated for each of the residues were used to build a machine learning classifier model to make predictions on potential binding sites of new proteins. In a nutshell, a machine learning classifier aims to retrieve relevant information from the features of samples of known class labels in order to make accurate predictions of the class label of newly unseen samples.

If we apply this rationale to our specific case, our approach focused on calculating the above-mentioned features of a maximum of 40 homologous proteins to the query protein we wanted to predict obtained from BioLip (see previous section), and use this information in the form of a data frame to create our training data set used to train our model. In order to do so, we extracted the known binding site information from each of the residues of the protein homologs, and added this as a new field to the data set in a binary manner (‘1’ for residues being part of binding sites and ‘0’ for residues not being part of binding sites).

It is worth noting that our team noticed two main factors influencing the amount of computational cost and time of the program, the first being making predictions on individual query chains; and secondly the binding site labels are extracted from the BioLip database accessed online. However, the team decided to sacrifice time expense with attempting to get the most accurate predictions possible.

All in all, we expected the model to make accurate predictions on query proteins, since the training data set is being ‘personalized’ each time with homologous proteins in order to build the model and make predictions on the query protein.

Random Forest Classifier

The classifier algorithm chosen for building our machine learning model was a Random Forest[8]. There are a few reasons for this, the first one being the fact that this type of algorithm is good at handling large and complex data sets with high number of features. Since our approach was residue-centered with 27 features being calculated per residue, we thought it made the most sense to use this type of algorithm. Secondly, unlike other classifier methods, Random Forests are especially good at handling multicolinearity, meaning having 2 or more features being correlated with each other. Since we are working with many features, there can be a high chance that some of them might have some level of correlation (for instance, number of hydrophobic atoms and residue hydrophobicity value), and it is well known that colinearity can distort the model into introducing biases in the predictions.

The way a Random Forest works is by making bootstrap samples with substitution from the original data set and building many decision trees with the same length as the original set, but not all features being selected in each sample being taken. Each decision tree keeps splitting the data according to the most relevant features until no more splits are left or allowed. Finally, once all decision trees are built, each of the samples is given a vote to pertain to a certain class according to each tree and ultimately assigned to the class of higher voting. This approach makes it clear how it deals with avoiding problems with colinearity and high complexity of data.

Class imbalance

Also, our team tackled the problem of class imbalance. Protein binding sites represent an overall small percentage of the total residues of the protein compared to the rest of amino acids of the protein. This added a new challenge to the problem as classifiers might tend to predict better the overrepresented class since there is much more information about it in the classifier, and the opposite with the underrepresented one.

So, our team took advantage of a specific attribute ‘class_weight’ from the RandomForestClassifier() function from the *sklearn*[9] module in order to give a weight inversely proportional to the percentage of representation of the class in the training data, in the case the overrepresented class appeared 3 times more than the remaining one (the latter one most probably always being the residues pertaining to binding sites).

5. Query Feature Extraction

Once the training data has been generated, the program proceeds to calculate the features of the query protein. The reason behind this order is to save computer expense, since the program makes sure that, before any information is extracted from the query protein, there are homologous proteins in the BioLip database to create the machine learning model. If there would not be information to build a model and make predictions, it would not make sense to calculate the query protein features.

Again, first the PDB files of each of the chains are obtained and the features are calculated for each of them individually.

6. Binding Site Prediction

Once the feature matrix has been calculated for each of the query protein chains, the machine learning models are applied to the data in order to predict the binding site binary class labels.

The result of this is the creation of a Chimera command file containing the selection and coloring of the predicted binding sites. The user should first start Chimera and load the query protein pdb file. Once the file is loaded, the user should just open the Chimera command file and the predicted residues would be colored automatically.

In fact, the program not only colors the predicted residues but shows 3 different colors corresponding to:

- Residues predicted by the program: these are colored in green.
- Real residues being part of binding sites: these are colored in red.
- Residues that coincide between the prediction and reality: these are colored in blue.

These two last colors will only appear in case the query protein is in the BioLip database and the program is able to extract the real binding sites to do the visual comparison.

Furthermore, if the real binding sites are available the metrics of the prediction are computed: accuracy of the prediction, precision, recall and F1.

Finally, in case the program finds predicted residues that are part of binding sites it will display a list of them and provide the data frame generated to carry out the prediction. In case the program does not find any residues involved in the binding site, it will print a message indicating it.

Bibliography

1. Charles A Santana, Sandro C Izidoro, Raquel C de Melo-Minardi, Jonathan D Tyzack, António J M Ribeiro, Douglas E V Pires, Janet M Thornton, Sabrina de A. Silveira, GRaSP-web: a machine learning strategy to predict binding sites based on residue neighborhood graphs, Nucleic Acids Research, Volume 50, Issue W1, 5 July 2022, Pages W392–W397, https://doi.org/10.1093/nar/gkac323

2. Charles A Santana, Sabrina de A Silveira, João P A Moraes, Sandro C Izidoro, Raquel C de Melo-Minardi, António J M Ribeiro, Jonathan D Tyzack, Neera Borkakoti, Janet M Thornton, GRaSP: a graph-based residue neighborhood strategy to predict binding sites, Bioinformatics, Volume 36, Issue Supplement_2, December 2020, Pages i726–i734, https://doi.org/10.1093/bioinformatics/btaa805

3. Dai, T., Liu, Q., Gao, J. *et al.* A new protein-ligand binding sites prediction method based on the integration of protein sequence conservation information. *BMC Bioinformatics* 12 (Suppl 14), S9 (2011). https://doi.org/10.1186/1471-2105-12-S14-S9

4. Zheng X, Gan L, Wang E, Wang J. Pocket-based drug design: exploring pocket space. AAPS J. 2013 Jan;15(1):228-41. doi: 10.1208/s12248-012-9426-6. Epub 2012 Nov 22. PMID: 23180158; PMCID: PMC3535113.

5. Zhang, Yanmin; Zhang, Danfeng; Tian, Haozhong; Jiao, Yu; Shi, Zhihao; Ran, Ting; Liu, Haichun; Lu, Shuai; Xu, Anyang; Qiao, Xin; Pan, Jing; Yin, Lingfeng; Zhou, Weineng; Lu, Tao; Chen, Yadong (2016). *Identification of Covalent Binding Sites Targeting Cysteines based on Computational Approaches. Molecular Pharmaceutics, (), acs.molpharmaceut.6b00302–.*doi:10.1021/acs.molpharmaceut.6b00302.

6. Fassio,A.V. et al. (2019) nAPOLI: a graph-based strategy to detect and visualize conserved protein–ligand interactions in large-scale. IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 17, no. 4, pp. 1317-1328, 1 July-Aug. 2020, doi: 10.1109/TCBB.2019.2892099.

7. Luttrell, J., Liu, T., Zhang, C. *et al.* Predicting protein residue-residue contacts using random forests and deep networks. *BMC Bioinformatics* 20 (Suppl 2), 100 (2019). https://doi.org/10.1186/s12859-019-2627-6

8. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001.

9. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.