Ben Habermeyer Final Project for CIS192

Deep Learning for predicting kinase inhibitor binding

Project consists of 2 files:

1. Project.ipynb
2. Final\_Project\_CIS\_192.ipynb

The premise of my project was to compete in an open source bioinformatics competition to develop a regression model for predicting Kd (dissociation constant) values for a given pair of small molecule drug and kinase protein (competition details at <https://www.synapse.org/#!Synapse:syn15667962/wiki/583305>). I am interested in bioinformatics and wanted to get a start here. The data set given was a large open source collection of protein and molecule interactions (Drug Target Commons). After conducting a literature review, I decided the most important features for this project would be 1) the molecule’s sequence-specific molecular signature (atoms and bonding), and 2) the protein’s sequence of amino acids surrounding kinase domains, or if the protein was not a kinase, then its largest functional domain or sequence of first amino acids, if it had no functional domains.

In the first notebook, I read in the dataset (2GB!) and remove columns which are unnecessary to me for collecting the protein / compound data. Then I remove any columns missing identification information, as well as those whose metric for evaluating binding is not Kd (some other concentration-dependent factor). I then take lists of compound ID’s and used NCBI’s ID exchange service to get a sequence of SMILES (chemical signatures) for each compound. I used UNIPROT to collect information about each protein ID, then parsed its domains and sequence to get the region around the kinase as stated previously.

In the second notebook, I remove duplicate entries by taking the average Kd values of multiple trials, then created a KinaseBinding class to wrap my ML algorithms. The class takes as input lists of protein amino acid sequences, SMILES sequences, and Kd values for the paired data. Functions in the class can convert Kd values to pKd (-log(Kd)), and convert each sequence of proteins and SMILES to one-hot-encoded Numpy arrays describing the sequence-specific data. Since most kinases had between 250-300 amino acids and consisted of 21 amino acids, I make a kernel of size 300x21 to represent each protein and compound. I figured this was the best way to represent sequence-specific data, and the important regions of the proteins.

An example of a protein would be AAGKT represented as a 300x21 matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | G | K | T | … | W |
| 0 | 1 | 0 | 0 | 0 |  | 0 |
| 1 | 1 | 0 | 0 | 0 |  | 0 |
| 2 | 0 | 1 | 0 | 0 |  | 0 |
| 3 | 0 | 0 | 1 | 0 |  | 0 |
| 4 | 0 | 0 | 0 | 1 |  | 0 |
| … |  |  |  |  |  | 0 |
| 299 | 0 | 0 | 0 | 0 |  | 0 |

Likewise for a compound with SMILES HC=CNH would be represented as a 300x21 matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | C | H | N | = | … | @ |
| 0 | 0 | 1 | 0 | 0 |  | 0 |
| 1 | 1 | 0 | 0 | 0 |  | 0 |
| 2 | 0 | 0 | 0 | 1 |  | 0 |
| 3 | 1 | 0 | 0 | 0 |  | 0 |
| 4 | 0 | 0 | 1 | 0 |  | 0 |
| 5 | 0 | 1 | 0 | 0 |  | 0 |
| … |  |  |  |  |  | 0 |
| 299 | 0 | 0 | 0 | 0 |  | 0 |

I wrote functions to perform each of these encodings, as well as split the data into test and training data. I wrote a decorator to time each function, and a \_\_sizeof\_\_ magic method to get the input size in bytes. Then I trained 6 different CNN models, experimenting with different architectures. I focused on how to best implement pairwise learning, incorporating both datasets into the training. I tried concatenation, summation, and convolution before concatenation. Concatenation performed better than summation. The best model came when I incorporated more pooling and convolution layers, and the best accuracy on test data came when I added more dense layers with dropout. I will submit my finalized model with its predictions to the competition. My final RMSE is pretty good (<1.2 in pKd values), and each of the outputs are shown in the second ipynb file.