GEMSEC Neuropeptide Docs

January 28, 2019

Chris Pecunies, Aaron, Savvy Gupta

Led by: Jacob Rodriguez

PI: Mehmet Sarikaya

Abstract

The purpose of this documentation is to serve as a reference for the motivation, background, methods, and results of the GEMSEC project to discover the method of signal transduction in human neuropeptides, to ultimately discover candidate peptides with binding affinity to graphene, as well as to illuminate their mechanism of binding.

Introduction

The overarching aim of this project is to uncover the mechanism of signal transduction in neuropeptides which are experimentally determined to bind to graphene. To determine candidate peptides to be experimented on, we have utilized three principal methods: cross-sequence entropy, PAM30 sequence similarity, and RRM[1] relative to two known binders, *GrBP5*[2] and *M6*[3]

Background

While there is extensive literature surrounding the subject of peptide-solid binding and its most important determinants, we have focused on two primary methods: sequence domain functionality clustering and the *Resonant Recognition Model* proposed by Irena Cosic [4].

Methods

RRM

The *Resonant Recognition Model*, or RRM, is a method proposed by Irena Cosic for the analysis of peptides and proteins. The model assumes electron-ion interaction potential (EIIP, hereafter) along the backbone of amino acid chains to be the most significant deteriminant in a peptide or protein's biochemical phenomenological features. For each amino acid, a unique scalar EIIP value can be determined, which allows a amino acid sequence to be converted to a scalar sequence of EIIP values in its most simple form.

[1]dfsdfsfs
[2]fdsfs
[3]dfsdfsf
[4]dfsdfs

Cross entropy

We determined the cross entropy for both the pure amino acid sequences as well as the functional domain numerical sequences, which can be found in the appendix (and which will be generated in the main program).

PAM30 Sequence similarity

The PAM (point accepted mutation) matrices, initially developed by Margaret Dayhoff[5] are commonly used in measuring the similarity of two peptide or protein sequences. The PAM30 matrix, specifically, is a sequence alignment matrix that allows 30 point accepted per 100 amino acids. The PAM30 matrix is a "shallow" sequence alignment matrix, in that it is more appropriate in determining alignment for more "similar" sequences.

Results

Candidate peptides

The five similarity metrics generated for each neuropeptide served as a feature set which could be weighted and used to train a regression model, but without experimental data, the relative weights for each feature were chosen by [METHOD].

After forwarding the candidate peptides for experimentation, we utilized statistical clustering and signal processing methods to predict determinant sequence domains in graphene binding which could then be used with the experimental data once acquired.

Experimental results

After generating the list of [NUMBER] candidate peptides and receiving binding affinity metrics, we were then able to perform several statistical methods to ascertain the most influential factors in peptide graphene binding.

First, we trained a simple linear regression model, using the gathered experimental results as training data, and [...]

Signal transduction

From results gathered from model training, statistical clustering, and experimental cross-validation, we were able to identify several sequence patterns and corresponding functions which may prove significant. [...]

Discussion

Computational results

[5] dsfsdfs

Our program, using all aforementioned computational methods, generated two "similarity tables" for 934,235 [@TODO: revise this] neuropeptides: one for GrBP5, and one for the wild-type peptide M6. This generated a table of the schema (example data – replace with top 10-20 candidate peptides):

Table 1: *Output schema of .csv similarity tables*

seq	numseq	rrmsn	rrmcorr	pam30	numentr	aaentr
IMVSTED	1132335	73	88	23	45	55
GTTYUEI	3224121	11	5	6	14	13

Where the columns correspond as follows:

- **seq**: Original amino acid sequence of the neuropeptide
- **numseq**: The original amino acid sequence converted to numbers corresponding to their biological function as determined by the biological function key-value table[6]
- **rrmsn**: The signal-to-noise ratio of the cross-signal of the EIIP frequency spectum calculated for a given neuropeptide and the sequence of comparison, normalized in the domain [0, 100].
- **rrmcor**: The Pearson correlation coefficient between the EIIP frequency spectrum calculated for a given neuropeptide and the sequence of comparison, mapped to the domain [0, 100]
- **pam30**: The PAM30 similarity score (formula described in methods section); a measure of inverse distance using the PAM30 matrix, mapped to [0, 100]
- **numentr**: The cross-entropy of the numerical sequence represencted by `numseq`
- **aaentr**: The cross-entropy of the amino acid sequence

From this data, we determined that [...]

Experimental results

[to be completed after experimentaiton]

Clustering and signal analysis

After gathering the experimental results, we were then able to apply supervised statistical learning techniques to the full set of neuropeptides, as well as signal processing techniques. We first [...]

[6]See table 1 in appendix

How to run

Prerequisites: A computer running Windows/Mac OSX/Linux with Python (3+) installed, and the Python libraries pandas, NumPy, and scikit-learn. For visualizations, the library matplotlib should be installed. If these libraries are not already installed, instructions for their installation will be listed below.

1. Download the .zip containing all .py files and sample data set [**!FIX**] and extract to preferred location.
2. Open a terminal and navigate to the directory containing the extracted files
    1. On Windows, press the Windows key and type "cmd", and the press enter. `dir` lists all files and folders in the working directory, while `cd dirName` changes the working directory to the specified folder (in this case, `dirname`). To move up the directory hierarchy, type `cd ../`
    2. On Mac OSX, enter spotlight search with Command + Spacebar and type "Terminal", then hit enter. Instructions are the same as for windows, but replace `dir` with `ls`.
    3. On Linux, a terminal is likely easily accessible. Commands are the same as for Mac OSX.
3. When in the directory containing the extracted files, type the following command to generate the "similarity tables" for the example dataset: `python main.py example_data.csv`
    - This can be run with any .csv sequence file, but it must follow the same schema as the example_data.csv table provided, and the sequences must all be of the same length relative to each other as well as to the known binder(s) (non length-matching input sequences will be ignored in output)
    - *(To implement)* The script accepts as a second argument a sequence (or list of sequences), each of which will generate its own similarity table for the input .csv.
        - If no argument is given (as above), it will generate two .csv similarity tables, one for GrBP5 and one for M6. To specify only one .csv output similarity table (for GrBP5), run `python main.py example_data.csv grbp5`.
        - This can be done for an arbitrary number of different sequences, for example: `python main.py example_data.csv IVTSSY UVGEASTT EEVTUSGMII` will output three .csv tables for peptides in `example_data.csv` of lengths corresponding to each sequence specified by the user.
        - Finally, the second argument can itself be a .csv of sequences,

following the same schema as the first input .csv. In this way, for a .csv entered as a second argument with 10 rows of sequences will generate 10 separate .csv tables.

4. Similarity tables and visualizations will be generated in a folder titled "output" located in the same directory as the `main.py` file.

Appendix

Table 1: AA

Function	AA	Num	EIIP
Aromatic	F	0	
Aromatic	Y	0	
Aromatic	W	0	
Hydrophobic	A	1	
Hydrophobic	V	1	
Hydrophobic	I	1	
Hydrophobic	L	1	
Hydrophobic	M	1	
Polar	S	2	
Polar	T	2	
Polar	N	2	
Polar	Q	2	
Proline	P	3	
Glycine	G	4	
Charge (-)	D	5	
Charge (-)	E	5	
Charge (+)	K	6	
Charge (+)	H	6	
Charge (+)	R	6	
Excluded	C	7	

Bibliography