This is the full code description of protein function prediction based on sequence feature clustering.

1. Download the data

First of all, from the official website of UniProt database (http://www.uniprot.org/) Download Homo sapiens(human, [9606]) protein sequence (version 2020.08.03), the specific download method is as follows, first, select the manually reviewed and annotated protein, reviewed:yes. Then select entry name, gene names, sequence, protein names, and gene ontology from the options list (go), click to download the information contained in the protein sequence, select to download all 20375 sequences, save the format as FASTA file, we save it as **origin data** file.

After downloading the protein sequence data, we continue to download the go number of Homo sapiens protein sequence on UniProt, and select the same reviewed:yes in the options list, select gene ontology IDs and download all 20375 sequence data. We save them as **GO-BP-CC-MF.csv**.

2. We are in GitHub(https://github.com/pufengdu/UltraPse) download the latest version of Ultrapse, the folder is **Windows**. In the command line interface, we use the help document of Ultrapse to select parameters. Its parameter selection is described as follows.

The tdfs parameter input is classic-pseaac.lua This is the standard pseudo amino acid composition transformation task. For -t, 2 is selected here, -l is 2-15, step size is 1, a total of 14 parameter values. -w is 0.05-0.80, step size is 0.05, a total of 16 parameter values, where w is the ω parameter of pseudo amino acid composition, that is, the proportion of sequence order utility, -f is selected as SVM, that is, libsvm format. Using Ultrapse's own command parameter -v to remove untransformed protein sequences, 20375 protein sequences are transformed according to the corresponding parameters. The program is **step1**, and the results are saved as **ultrapse-file-origin**.

3. We shuffled these protein sequences with shuffle. After shuffling, we cut out the sample set (18302 proteins) and test set (2034 proteins) which about 10% of total. These proteins are used to make the final prediction data, code is **step2**, and the results are saved as **training-ultrapse-file**(18302 proteins) and **test-ultrapse-file**(2034 proteins). We saved the sample set and training set according to the SP number of each protein. They are **train-protein-list** and **test-protein-list** files respectively. In order to facilitate the data processing, we saved the vector results under each parameter according to the protein classification., we saved each set of parameter vector results as **train-protein-**

to-vector and **test-protein-to-vector**. Because these two files are too large, we did not upload them. Readers can generate the results according to **step5**.

4. In order to store the clustering results and facilitate the enrichment in the next step, we label the SP number of the cluster proteins. We use spectral clustering and other related methods in the python scientific computing library sklearn to implement spectral clustering. Numpy(1.16.4) and sklearn(0.22.1) libraries of Python3.6.9. The function has two main parameters, numbers of clusters, which refers to the number of clusters of data output structure Neighborhood refers to the number of adjacent nodes reserved in the process of building a graph. We choose 7 parameter values of 10-40 and step size of 5. Considering the time complexity of full connection algorithm, we choose k-nearest neighbor method when constructing spectral clustering graph, in which the number of adjacent points of parameter k is fixed to 150, and we choose NCut cutting method. The execution code is **step3**. the results are saved as **spectral-cluster-file**, Because this file are too large, we did not upload it. Readers can generate the result according to **step3.** The **step3** spectral clustering code due to the speed of the experiment,we opened a number of processes to run the code at the same time, so it was used **parameter.py** generate **parameter.txt** to save the parameter combination, which was used to guide the code to conduct spectral clustering of which parameter combinations when running code in parallel**.**

5. We choose to use the enrichgo(gene, keytype, orgdb, ont, pvalue, cutoff, readable) function in the cluster profiler Library of R language to realize enrichment operation. The version of R is 3.6.0 (2019-04-26). The specific code is **step4**, and the results are saved as **bp-go-p-result**，**mf-go-p-result**，**cc-go-p-result** respectively. Because this file are too large, we did not upload it. Readers can generate the result according to **step4.**

6. We used the center point of each cluster as the representative vector value of the class, that is, the value of the representative vector of each cluster. For each GO enrichment result, we extracted all the enriched GO terms and their corresponding P values as their tag values. The format is as follows:

"cluster_name" : "2-0.05-10-150-2"

"vector" : [ ]

"data" :

{ "go_id": "GO:0048018", "p_value": "4.30412994772681e-19"}

Among them, cluster_name refers to the parameter values of the clustering result,

vector is the center value under the clustering result, that is, the representative value of the clustering result, and data refers to all the enriched GO terms and their corresponding p values, go_id refers to the ID value of the corresponding GO term in the database, p_value is the p value of the enrichment result. So far, our GO enrichment matching library has been established. Because this file are too large, we did not upload it. Readers can generate the result according to **step6**, and the results are saved as **bp-data, mf-data, cc-data** respectively. We compared the real GO of the predicted protein with the GO library, and remove the GO that is not in the GO library. The execution code is **step8**, and the result is **real-protein-go-ranklist**. Because there will be very small values in enrichment and distance calculation, it will be displayed as 0 when it is implemented. According to the programming tools, we uniformly mark the point with the minimum value of 0 as 10^{-308}.

7. The vector value of each protein to be predicted under each set of pseudo amino acid composition parameters, the center point and corresponding p value of each corresponding parameter in GO library were calculated according to our score model, and then the score table of the protein to be predicted was constructed by arranging the score from high to low. The execution code is **step9**, and the saved results are **protein-predict-bp-list-all, protein-predict-mf-list-all, protein-predict-cc-list-all**. Because these three files are too large, we did not upload them. Readers can generate the results according to **step9**.

8. Our goal is to make the protein in the data set to be predicted pass through our prediction algorithm, and make its go item appear high score in our database. By observing the proportion of the real GO terms of the protein to be predicted, which appears in the top 100(<=10%) of our scoring table, we can evaluate whether our algorithm is effective. The number of GO library tables in MF, CC and BP are 1195, 777 and 6418 respectively.

9. All the results were saved in the **evaluation** folder.

Code of **bp-step 10.py** was used to count the percentage. This percentage is the ratio of number of GO terms of per protein that appeared in the top 100 of the prediction table to its all number of GO terms(Did not include GO terms that was not in our library table). **cc-step 10.py** and **mf-step 10.py** were also used in the same way. These results are **bp-1-2034-100.txt**, **cc-1-2034-100.txt, mf-1-2034-100.txt.**

Code of **bp-count.py** was used to calculate the proportion of the whole result.

In terms of molecular function, we found that the data of the top 100 proteins were

as follows:

Among the 846 proteins to be predicted, there are no GO terms related to molecular function, so we did not include 846 proteins, and each number was reserved to two decimal places. Among the remaining proteins to be predicted, 30.47%(362/1188) of the proteins have all GO terms rose in the ranking list, and 50% GO terms of the 47.56%(565/1188) of the predicted proteins have been ranked up.

In terms of cell composition, we found that the data of the top 100 proteins were as follows:

Among the 951 proteins to be predicted, there were no GO terms related to cell composition, so we did not include 951 proteins. Among the remaining proteins to be predicted, 35.92%(389/1083) of the proteins have all GO terms rose in the ranking list, and 50% GO terms of the 52.35%(567/1083) of the predicted proteins have been ranked up.

In terms of biological processes, we found that the data statistics of the top 100 proteins were as follows:

Among the 551 proteins to be predicted, there are no GO terms related to biological process, so we did not include 551 proteins. Among the remaining proteins to be predicted, 6.81%(101/1483) of the proteins have all GO terms rose in the ranking list, and 50% GO terms of the 14.90%(221/1483) of the predicted proteins have been ranked up.