Hierarchical clustering of protein's contact maps

To run the code the following libraries are required:

- Argparse
- Numpy
- Scikit learn
- Scipy
- Yellowbrick

This python script returns the hierarchical clustering of the contact maps given in input by giving in output:

- A representative contact map for each cluster created
- A dendrogram showing the clusters and where the tree is cut
- A file txt containing the count of distinct edges in all snapshots
- A file txt containing the label of each snapshot
- A file txt containing the distance matrix used in order to obtain the clustering
- A file containing the most important contacts in each cluster

The code must be ran from the Script folder and to obtain the outputs the command is:

```
python3 main.py contact_maps_paths.txt
```

Where *contact_maps_paths.txt* is the file containing the path to the edge files (snapshots to clusterize).

Evaluation of the clustering

To evaluate the obtained clustering we compared it to the hierarchical clustering obtained using the pairwise RMSD which is computed using *TM-Score.cpp*.

In order to compute the pairwise RMSD for a new MD the command is:

```
python3 main.py contact_maps_paths.txt -RMSD_path path_to_RMSD_file -path_to_pdb
path_to_pdb_folder
```

It is also possible to change the path of the output directory using *-out_dir*.

To use the *-path_to_pdb* argument is necessary to compile *TMscore.cpp* using the linux command:

```
g++ -static -O3 -ffast-math -lm -o TMscore TMscore.cpp
```

The '-static' flag should be removed on Mac OS, which does not support building static executables.

When using *-path_to_pdb* the program will write a *RMSD.txt* file in *out_dir/contact_maps_paths* which is the file to pass as argument when using *-RMSD_path*.

-*RMSD_path* makes so the program computes the clustering based on the RMSD and compares it with the contact map clustering using *adjusted_rand_score* from scikit learn

Adjusted RandIndex

The Rand Index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings, the raw RI score is then “adjusted for chance” into the ARI score using the following scheme:

$$ARI = \frac{(RI - Expected_{RI})}{(max(RI) - Expected_{RI})}$$

Note that this measure is severe since is usually used to check the obtained clustering against the ground truth.

Clustering Algorithm

Data

The data contained in the edge files is transformed into multiple matrices and vectors which represent the contact maps of each snapshot with a bit of tweaking, in fact these matrices contains 0 when there is no contact between residue i and residue j (matrix[i][j]) meanwhile when there is a contact the value is the result of this sum:

$$matrix[i][j] = \sum bonds\_energy + 0.2|i - j|$$

This way of computing the contact matrix allows us to give more importance to contacts with more energy and also give more importance to contacts between residues which are fare from each other.

In order to proceed with the clustering we use a vector version of the matrix described above which is obtained by lining up each row of the matrix, this is done because clustering methods from scikit requires in input a matrix of shape (n_snapshot, n_features).

PCA

Before computing the clustering we use Principal component analysis in order to decrease the dimensionality of the data with the objective of simplify the task.

The PCA method searches for new variables that can be seen as linear combination for the initial data which explain the maximum possible variance and are also incorrelated between each other, scikit allows us to to specify the amount of variance to capture (we set 90%) and the method then chooses the right amount of components.

Note that before applying PCA the data must be centered and scaled, we do so using scikit StandardScaler.

Metric

The metric that has been chosen is the **cosine distance** since it is commonly used in high dimensional spaces and seemed to perform better than other like euclidean, jaccard, other boolean metrics like yule and other correlation metrics like pearson.

The cosine distance between two vectors u and v is defined as $1 - cosine\_similarity$ where:

$$cosine\_similarity = \frac{u \cdot v}{||u||_2 ||v||_2}$$

$|| * ||_2$ is the 2-norm of its argument `*`, and $u \cdot v$ is the dot product of `u` and `v`.

Elbow and Silhouette

To select the best number of cluster we use the **elbow** method which uses the point of maximum curvature of a score function which depends on the number of cluster. The score function we chose is the **Silhouette score** which is defined as:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

where $b(i)$ is the smallest mean distance of i to all points in any other cluster meanwhile $a(i)$ is the mean distance between i and all other data points in the same cluster, the final score which the elbow method uses is the mean silhouette of the clustering.

Outputs

The **Representative contact map** is the visualization of the contacts which are present on at least half of the snapshots in the considered cluster, this should help visualizing the most important contacts for each cluster.

The **Important contacts** are computed comparing the Representative contact maps of each cluster, in fact the list of important cluster contains for each cluster the contacts that are present only in its own representative contact map.

The **Dendrogram** shows the clusters and how they are built, each cluster has a different color and the cut is also visible.