BMI 203 Winter 2021
Homework Assignment
#2

Due Wednesday 02/17/2021 by 11:59pm PST

In this assignment, you will evaluate results from a high-throughput virtual screen against the SARS-CoV2 Spike/ Human ACE2 interface. There are two parts to this assignment and Part 2 requires completion of Part 1. We recommend reading through both Part 1 and Part 2 before beginning this assignment.

Project skeleton: https://github.com/ucsf-bmi-203-2021/Project2

- **You must fork / clone this repo for this assignment**

Part 1: Implementation

When presented with a dataset, we would first like to determine any defining similarities or differences. The most straightforward way to do so is to search for "clusters" of similar entries.

The data we are considering come from Smith and Smith, 2020. In this study, they generated 6 Spike-Ace2 interface poses using MD simulations. They then docked ~10k small molecules against each protein conformation. Provided for you is the top (#1) pose for each ligand docked against one Spike-ACE2 interface conformation, as well as the corresponding SMILES string, AutoDock Vina score, and the "On" bits in the Extended Connectivity Fingerprint for that compound.

You will explore two different clustering approaches - a partitioning algorithm, and a hierarchical algorithm.

Your implementations should be classes meeting the following specifications:

- Class initialization requires parameters that describe the clustering procedure
- Class has a *cluster* method that accepts as input a set of ligands to cluster and returns a set of clusters
- All other implementation details are up to you.
- Class API is documented in a python docstring. **Please add your API to your README.md in your github repo or use a tool such as Sphinx. This will serve as "documentation" for your TAs to review your implementation.**

With this in mind, please implement the following:

1. A Ligand class to store relevant information for each ligand.
2. A partitioning algorithm to cluster the set of small molecules
3. A hierarchical algorithm to cluster the set of small molecules
4. A function to measure the quality of a set of clusters.
5. A function to compare the similarity of one clustering to another clustering.

To confirm your implementations are functioning correctly, provide unit tests that check any relevant functions you've written.

Part 2: Please answer the following questions

1. Explain the distance metric you utilized to calculate the similarity/dissimilarity between small molecules.
2. Use a dimensionality reduction algorithm (PCA, t-SNE, UMAP, etc) to generate a 2D visualization of the small molecule dataset. Each point should represent a single molecule. (Note: you may have to plot a subset of your data, depending on which dimensionality reduction algorithm you choose.)
3. Cluster the small molecules using your implementation of a partitioning clustering algorithm. Visualize this clustering by coloring clusters on the 2D visualization generated in question 2.
4. Explain your choice of partitioning clustering algorithm. Is it sensitive to initialization conditions? How do you select the number of clusters?
5. Cluster the small molecules using your implementation of a hierarchical clustering algorithm. Visualize this clustering in the same way as question 3.
6. Explain your choice of hierarchical clustering algorithm. Is it sensitive to initialization conditions? How do you select the number of clusters?
7. Evaluate the quality of both clusterings using your implementation of a clustering quality metric. Explain your choice of quality metric. Which clustering performed 'best' according to your metric?

8. Compare the two clusterings using your implementation of clustering similarity. How similar are the two clusterings using this function?

9. For the “best” clustering, as determined by your quality metric, visualize the distribution of Autodock Vina scores in each cluster. Do members of the same cluster have similar docking scores? Why or why not?

10. Select the top scoring molecule from each cluster. This is your list of cluster heads. Visualize the top 5 by score in PyMOL and pick your favorite. Are they structurally diverse?

Automated Testing with Github Workflows

The project skeleton is currently set up with a “workflow” that triggers when a commit is pushed to the main branch. This runs pytest with your python testing files located in the `/test/` folder.

For this reason, you must implement your project code in a **forked / cloned version of the project skeleton repo**. Code not built on this skeleton will not be accepted.

If this is your first time using github workflows, you must enable them under the “**Actions**” tab in your forked repo. If you do not enable them, commits pushed to main will not automatically activate testing.

To verify that your unit tests have passed, your testing badge, located in the README.md, must point to the correct URL. You can alter the URL for this badge as follows:

https://github.com/{your-username}/{your-repo-name}/workflows/HW2/badge.svg?event=push

To submit this assignment,

- Use Google Classroom to submit a link to your **public** Github repository that:
    - Contains a single pdf or ipython notebook labeled **Firstname_Lastname_BMI203_HW2.(pdf/ipynb)**, including your answers to the above questions in prose and associated plots.
    - Contains a README.Md that explains the layout of your repo (what code is where, how to use your implementation)
    - Contains all code relevant for this assignment
    - Is commented explaining the role of each functional code block
    - Has passed the implemented unit tests (as indicated by the github badge)
    - ***Note!*** we will only consider commits before the submission deadline.