**Bioinformatics 529 Homework 4 | Amelia Lauth | 04/06/22**

**Resources used (Including websites, partners in class, etc.):**

Partners in Class: Catherine Barnier, Brad Crone, Crystal Wen, Mahnoor Naseer

Class 17, 18, 19, 22 notebooks, Alan’s solutions, and PPTs.

[**https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html**](https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html)

[**https://numpy.org/devdocs/user/basics.indexing.html**](https://numpy.org/devdocs/user/basics.indexing.html)

[**https://numpy.org/doc/stable/reference/generated/numpy.zeros.html**](https://numpy.org/doc/stable/reference/generated/numpy.zeros.html)

[**https://www.youtube.com/watch?v=ipp-pNRIp4g**](https://www.youtube.com/watch?v=ipp-pNRIp4g)

[**https://www.w3schools.com/python/ref\_math\_sqrt.asp**](https://www.w3schools.com/python/ref_math_sqrt.asp)

[**https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html**](https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html)

[**https://machinelearningknowledge.ai/tutorial-for-dbscan-clustering-in-python-sklearn/**](https://machinelearningknowledge.ai/tutorial-for-dbscan-clustering-in-python-sklearn/)

[**https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/**](https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/)

[**https://becominghuman.ai/dbscan-clustering-algorithm-implementation-from-scratch-python-9950af5eed97**](https://becominghuman.ai/dbscan-clustering-algorithm-implementation-from-scratch-python-9950af5eed97)

[**https://www.educative.io/edpresso/how-to-implement-a-breadth-first-search-in-python**](https://www.educative.io/edpresso/how-to-implement-a-breadth-first-search-in-python)

**Statement of Objective (What was the purpose of this homework assignment):**

The purpose of this homework assignment is to construct and execute, using dynamic programming, the Needleman-Wunsch algorithm to perform global alignments of protein sequences. It is very similar to the Smith-Waterman algorithm which we did in class; the only difference is that Smith-Waterman does local alignment. The second half of the homework explores DBSCAN, which is another clustering method compared to the one we did in class (K-means algorithm). In this case, we are using two arguments (epsilon and min\_points) to identify sets of data points that are near to neighborhoods of high density.

**Procedure (Explain in general terms how you went about implementing the homework assignment):**

As stated above, we completed this homework through dynamic programming, where we break a large problem down into smaller problems, and we combine the solved outputs of the smaller problems to answer the original large problem. For the Needleman-Wunsch algorithm, I followed Alan’s Class 17 code solutions (Smith-Waterman) as a guide, re-watched the class 17 video, and looked over the class 17 PPT again. I also wrote on paper what needed to be added per the pseudocode provided in the homework, and that helped me devise a plan of what code needed to be written and/or what methods I needed to google for guidance. For the SBSCAN, I revisited the class 22 code, class 22 PPT, class 22 video, wrote down pseudocode, and googled specific python methods I could implement in my code.

**Difficulties and Roadblocks (What were the pain points in the implementation of this homework assignment):**

The DBSCAN algorithm was trickier because I struggled to continue the breadth first search for each of the neighbor points to see if they were core points or border points, and to continue that process until no more “children” in that cluster existed. (Note: All my different versions of code are commented and pasted in the cell right below the DBSCAN code.) The algorithm would update the cluster ID and go to the next point that was not visited yet and start over, seeing if that one point was a core point or an outlier. If it was a core point, it would continue the breath first search for its neighbors to see if they were core points or border points. I explained the whole algorithmic concept correctly to Brad, but still had trouble implementing the while loop again with only the neighbor points list. I did end up getting four clusters, not including the outliers which are designated with 0’s. The problem was the clusters were labeled as “2”, “6”, “11”, and “12” instead of “1”, “2”, “3”, “4”. It must be due to the neighbors not being iterated properly with BFS, since they are checked properly for core, border, or outlier point to my knowledge. I did try to implement the BFS again with the neighbor list now, and I got a worse result, with 5-7 clusters and outliers, clusters being labeled wrong. Hoping to discuss this with instructors or Brad after homework is handed in to see exactly what I did wrong.