DESKGEN TASK REPORT

JEROME JAVELLE

1. Overview of the Project

In order to establish a link correlation between guide sequences and their activities at different times, with or without the presence of the drug, we suggest the following approach :
— cluster the data
— look for patterns in the guide sequences inside each cluster
The program outputs the clusters and the id of the data points included in it as well as some statistics about the frequencies of the subsequences involved in each cluster.

2. Features and Implementation Choices

2.1. **Language and Framework.** In the purpose of having a reusable and maintainable code, we decide to implement our project using `C++` mainly. Although the development time is longer, the execution speed reachable with this language is several order of magnitudes faster than scripting languages or VM interpreted languages. This choice is also motivated by the many executions we would run in order to test several parameters for the machine learning algorithms used. We also use a python script to process the input data file once (no need for huge optimization since the task is of linear complexity and is executed only once).

2.2. **Clustering.** First, we must select a training set from the whole data set. We use a python script `selectData.py` to select a training set which presents noticeable bias by ignoring data points with similar guide counts in the different environments. Then, after reading the articles provided, we decide to use a variant of SVMs for unsupervised clustering called "Support Vector Clustering". We base our program on the algorithm found in the paper "Support Vector Clustering" by Ben-Hur, Horn, Sieglemann and Vapnik. The clustering algorithm goes as follows :

— Create the kernel matrix $Q = (k(x_i, x_j))_{i,j}$ where $k = e^{-q||x_i - x_j||_2^2}$ is the Gaussian kernel function of parameter q.
— Solve the optimization problem suggested in the SVC paper based on the Wolfe dual form $\sum_i k(x_i, x_i)\alpha_i - \sum_{i,j} k(x_i, x_j)\alpha_i\alpha_j$ using `dlib`.
— Compute the radius R of the smallest enclosing sphere in the kernel space.
— Build graph of data points where 2 points are connected whenever the whole segment between both points are inside the sphere (20 sampled points).
— Find connected components of the graph that will constitute the clusters using `boost`.

2.3. **Cluster Analysis.** To perform quicker DNA sequence analysis, we encode the 20 amino acids sequences as `uint64_t` ($T = 00$, $A = 01$, $G = 10$, $C = 11$). After clustering the data, we analyze the sequences by counting the frequency of the singletons, pairs and triples of amino acids. We can then detect potential bias between the clusters and associate the different activities to the presence of these specific subsequences.

3. Complexity Analysis

The complexity of the graph creation from the training set is quadratic (processing pairs of vertices), whereas the cluster assignment of the data points is linear. Therefore, we have a reasonable balance when the size of the training set scales with the square root of the size of the initial data set. This is why we tried to consider training sets of length between 100 and 300.

4. Results

First, we can visualize different projections of the data space in raw form or after computing activities. The corresponding plots can be found in the directories `plot/raw_counts/` and `plot/activities/`. From these projections we clearly see the interest of considering the activities rather than the raw data.

We note that there seems to be a huge cluster but smaller ones could also exist. This will confirm the fact that our algorithm often finds 2 to 5 clusters on average with one cluster being much bigger that the others.

The clustering process seems to give meaningful results since we managed to find clusters of data points with a particularly high activity after 14 days in presence of the drug. We can see this by using the file `data/training_data2.tsv` as a training set and `data/data_10000.tsv` as a data set for example.

The sequence analysis test shows some light bias in the frequency of the subsequences, but this step may need more development time to get rid of the noise and perform more advanced logistic regression. However we can note a higher frequency of the subsequence `FFF` in the 3rd cluster which corresponds to a high activity after 14 days in presence of the drug. For example, the previously mentioned files give rise to 3 clusters (labeled '0', '1' and '2'), and the analysis of the frequencies of the triples of amino acids show a point which detaches significantly from the rest when we compare the clusters 0 and 1 together, then 1 and 2 (see files `plot/bias2_1_3.png` and `plot/bias2_2_3.png`). We may deduce from this that the presence or absence of a triple could have an impact on the guide activities in the presence of the drug.

In order to increase the significance of this analysis, we can consider absolute positioning of the subsequences as well.

5. Future Improvements

Next steps in this project would be :
— analyze the impact on the parameters of the clustering algorithm (gaussian kernel, soft margin)
— change the selection process for the training set
— consider other functions of the raw guide counts (other than activities)
— improve the performance of the cluster assignment step

— add more features to the statistical analysis of sequences
— change the way we chose the training set
— last but not least : use LookUp Tables for the Gaussian kernel computation would drastically reduce the execution time

In relation to this project, it may be useful to maintain internally a "clustering engine" with metrics to check how good the clustering is. We may even consider several clustering algorithms and have them compete with each other for 2 purposes :

— take the result which gives the best metrics (might depend on the initial data set)
— give more credit to a discovered pattern if it has been found by several fundamentally different algorithms

I would be curious to try clustering using "augmented" Minimum Spanning Tree because it would not be affected by the shape of the clusters, and it would allow to change the "granularity" of the clustering without having to run the algorithm again. Moreover, the complexity does not grow exponentially with the dimension of the data space but stays linear. Another advantage is the metrics that come with the algorithm. I would be glad to discuss about this possibility with you to find out if it could fit the type of data you process.