Pierre TRETON

Compte Rendu :
Module 3 : Clustering

2 : Comparaison des séquences dans la classe *Sequence* :

Pour calculer la distance entre deux séquences, j'ai transcrit la formule fournie en code.

3 : Clustering hiérarchique (agglomératif) dans la classe *ClusterOfSequences* :

- 3.2.3 : Clustering agglomératif

o Méthode *linkage* :

Le calcul du rapprochement entre deux clusters se fait sur une moyenne du rapprochement de chaque élément d'une liste à celui de l'autre liste.

o Méthode *clusterizeAgglomerative* :

La méthode crée d'abord un cluster atomique pour chaque élément présent dans la liste. Cela facilite par la suite la comparaison entre des clusters de grande taille et des cluster qui ne contiennent qu'un seul élément.

Tant que le cluster contient plus de deux sous clusters dans sa liste, on cherche le meilleur rapprochement entre tous les clusters en calculant le linkage. On en récupère les deux clusters qui sont les plus proches, c'est-à-dire ceux qui ont la distance la plus courte, et on les assemble pour former un nouveau cluster qui contient les deux. Les deux clusters sont retirés des clusters initiaux et le cluster nouvellement formé y est ajouté.

- 3.2.2 : Visualisation sous le format Newick

Pour l'écriture de l'arbre phylogénétique au format Newick, j'ai fait une méthode récursive qui parcourt l'arbre à partir de la racine jusqu'à atteindre les feuilles.

A chaque fois que la méthode rencontre des sous-arbres, elle leur applique le même procédé récursivement et les séparent par des virgules entre deux parenthèses.

Enfin, on rend le résultat avec « ; » à la fin pour respecter les conventions d'écriture.

4 : Application au génome de l'hémoglobine dans la classe *Utils* :

- 4.3 : Clustering des séquences protéiques avec la méthode *codon2aa* :

Pour cette partie, j'ai fait une simple boucle qui récupère des morceaux de longueur 3 dans la séquence d'ADN pour les convertir en acides aminés et ensuite les ajouter les uns aux autres afin de former une chaîne d'acides aminés.

Mes résultats durant les premiers tests divergaient énormément des clusters de l'énoncé car dans la comparaison des séquences, j'utilisais la méthode *toString()* de la séquence pour récupérer la suite de nucléotides. L'introduction de la classe SequenceLabeled a changé le fonctionnement de cette méthode, j'ai donc remplacé l'appel à *toString()* par une récupération directe de l'attribut *seq*.

5 : Ajouts :

- 5.1 : Alignement des séquences dans la classe *AlignmentNW* :

J'ai fait cette partie jusqu'aux matrices de substitution. N'ayant pas d'exemple de matrice sous la main, je me suis contenté d'implémenter l'alignement de Needleman et Wunsch.

- 5.2 : Amélioration des dendogrammes :

o Alignement simple des feuilles :

Pour aligner les séquences à droite, il suffit de trouver la taille de la branche la plus longue en nombre de nœuds. Ensuite, lors de la construction de la chaîne au format newick, il suffit de retirer 1 à chaque fois qu'on rentre dans un nouveau cluster, puis lorsqu'on tombe sur une feuille, on lui ajoute 1 + la distance restante avec la notation newick (« (…) : … »). De fait, lorsqu'on atteint la branche la plus longue, on n'ajoute rien car la distance à ajouter est de 0.

o Alignement des feuilles + distance des nœuds :

Pour l'alignement avec distance, le principe est exactement le même mais au lieu de récupérer la branche la plus longue en comptant le nombre de nœuds, on cherche la branche qui a la distance la plus longue, c'est-à-dire la distance successive de du cluster au sous cluster puis du sous cluster au sous cluster atomique. On récupère donc la distance totale du cluster racine à la feuille qui en est le plus éloignée.

Ensuite on construit de même que précédemment la chaîne newick en utilisant comme longueur par défaut la distance du cluster au cluster parent, qu'on retire à la distance restante pour passer aux clusters fils.

Pour vérifier que l'arbre est bien aligné, il suffit de comparer la distance de chaque feuille à la racine, on doit obtenir le même résultat pour toutes les sommes successives de distances entre les nœuds.

- 5.3 : Clustering par approche divisive avec la méthode *clusterizeDivisive* :

Pour mettre en place le clustering par division successives des éléments, ma méthode travaille directement sur les éléments du cluster pour les séparer en deux groupes de proximité jusqu'à des clusters atomiques qui ne peuvent pas être divisés car n'ayant qu'un seul élément.

La méthode, à l'inverse de l'approche agglomérative, cherche les deux éléments qui sont les plus distants. Ensuite, on répartit les autres éléments en fonction de la meilleure proximité pour l'une ou l'autre des deux séquences. On obtient ainsi deux clusters auxquels on applique le clustering par division. On forme ainsi

- 5.4 : Modification du modèle de cluster dans la classe *ClusterOfSequencesBis* :

Pierre TRETON

Pour cette classe, j'ai fait une méthode qui récupère toutes les séquences dans le cluster. Cette méthode récursive parcourt les sous clusters jusqu'à tomber sur des éléments feuilles qui contiennent une séquence. Cette méthode renvoie tous les éléments sous la forme d'une ArrayList.

Comparaison du temps d'exécution :

- ### Clustering agglomératif

Le temps d'exécution de la classe Bis est deux fois plus court. Cela s'explique certainement par le fait qu'à la construction du clusterBis, on n'effectue aucun ajout d'éléments dans la liste, ce qui permet de gagner du temps lorsqu'on forme de nouveaux clusters par le clustering agglomératif.

- ### Clustering divisif

Le temps d'exécution de la classe Bis est très légèrement plus long car on cette méthode de clustering travaille avec la liste des éléments. Ainsi, à chaque division un nouveau parcours de l'arbre est effectué pour trouver les éléments feuilles.

NB : Pour valider mes arbres phylogénétiques, j'ai utilisé le site http://etetoolkit.org/treeview/ qui est moins strict sur la syntaxe newick que le site trex.uqam.ca, également la représentation de l'arbre et des distances varie d'un site à l'autre et le premier site me paraît plus fidèle.