IFT3295 - TP4

17 novembre 2017

Ce TP est à faire en équipe de deux ou seul si vous le préferez. Vous devez le rendre au plus tard le lundi 5 Décembre avant la démo.

Vous trouverez sur la page web de la démo, un fichier de séquences protéiques *proteines.fa* en format fasta, une liste d'arbres *arbres.nw* et une matrice de mutation *BLOSUM62.txt*. Vous devez remettre votre programme **exclusivement sur remise**. Il n'y a pas de contrainte sur le langage de programmation, tant qu'il n'est pas trop obscur. En cas de doute, me prévenir svp. Il est aussi impératif que votre code soit commenté.

Évolution de gènes

Une étude phylogénétique a permis d'inférer, par parcimonie, un ensemble d'arbres candidats $\mathcal{T}$ qui représentent potentiellement l'évolution d'un ensemble de protéines $\mathcal{P} = \{P_1, P_2, ..., P_k\}$. Tous les arbres de $\mathcal{T}$ ont le même nombre k de feuilles, et à chaque feuille, correspond une protéine distincte. Nous avons l'alignement multiple des séquences des protéines de $\mathcal{P}$ en format fasta, donné par l'ensemble de séquences $\mathcal{A} = \{A_1, A_2, ..., A_k\}$ où (A_i est la séquence de P_i). Nous désirons maintenant comparer ces arbres candidats à celui obtenu par *Neighbor Joining*. Pour ce faire, on vous demande de répondre aux questions suivantes.

Représentation d'arbres phylogénétiques et d'alignement

Vous devez tout d'abord écrire du code pour représenter les arbres phylogénétiques et gérer les alignements de séquences.

Vous devez donc implémenter un programme qui prend en entrée un fichier contenant une liste d'arbres **binaires enracinés** ainsi qu'un fichier

contenant des séquences alignées. Les grandes étapes du projet peuvent être résumées de la façon suivante :

1. Élaborer une structure de données permettant de représenter un arbre binaire enraciné. Votre structure d'arbre doit permettre d'effectuer un parcours en profondeur post-ordre (visiter un noeud seulement si tous ses descendants sont visités) et pouvoir stocker de l'information à chaque noeud. Vous n'avez pas le droit d'utiliser des structures d'arbres pré-existantes (sauf pour tester/valider votre code).
2. Convertir une chaîne newick en une instance de votre structure d'arbre. Vous devez implémenter cette conversion vous-même. Supposez que tous les arbres en entrée sont binaires, enracinés et sous un format newick valide. De l'information supplémentaire sur le format newick est disponible sur le site suivant : `http://evolution.genetics.washington.edu/phylip/newicktree.html`. Vous pouvez également vous référer à la page wikipédia : `http://fr.wikipedia.org/wiki/Newick`.
3. Être capable de calculer la distance RF entre deux instances de votre structure d'arbre. N'oubliez pas que la distance RF se calcule entre des arbres non-enracinés. Ainsi les deux branches incidentes à la racine (**seul noeud interne de degré 2**) représentent en fait une seule bipartition.
4. Lire et représenter un alignement de séquences en format fasta, ainsi qu'associer les feuilles de vos arbres aux séquences correspondantes. Vous devez également être en mesure de filtrer l'alignement afin d'enlever toutes les positions contenant des gaps. Tout le travail se fera sur cet alignement filtré sans gap.

Détermination de l'arbre NJ

Dans cette partie, vous devez reconstruire l'arbre NJ correspondant à l'alignement de séquences, puis le comparer en utilisant la distance RF avec les arbres en entrée afin de déterminer le meilleur candidat. Pour ce faire :

1. Calculer la matrice de distance entre les séquences à partir de l'alignement filtré. Le calcul de cette matrice de distance doit tenir compte de la matrice de poids de mutations (dans ce cas, la matrice de similarité blosum62). La cellule à la rangée r et à la colonne c donne le poids d'une substitution de l'acide aminé r vers l'acide aminé c. Vous

pouvez supposez que ces poids font partie intégrale du code (donc les "hard-codés").

La distance entre deux séquences A_i et A_i, peut être déterminée en fonction de l'équation suivante :

$$p = \sum_{c_k} M(A_i(c_k), A_j(c_k))$$
$$q_i = \sum_{c_k} M\big(A_i(c_k), A_i(c_k)\big)$$
$$q_j = \sum_{c_k} M\big(A_j(c_k), A_j(c_k)\big)$$
$$d(i,j) = 1 - \Big(\frac{p}{\max(q_i, q_j)}\Big)$$

où M désigne la matrice de poids et c_k chaque colonne de l'alignement.

2. Implémentez l'algorithme *Neighbor Joining* et appliquez le sur la matrice de distance précédemment calculée. Vous pouvez représenter l'arbre produit comme un arbre binaire enraciné à l'intérieur de votre code.
3. Calculez et retourner la distance RF entre l'arbre *Neighbor Joining* et chacun des arbres en entrée. Parmi les arbres candidats, lequel possède la plus petite distance RF, par rapport à l'arbre NJ ?

Enracinement de l'arbre NJ

Neighbor Joining retourne un arbre non-enraciné, qu'on souhaite ensuite enraciner. Il est possible d'utiliser la méthode du *mid-point* pour enraciner cet arbre. L'enracinement par le *mid-point* place la racine au milieu de la plus longue branche de l'arbre. Proposez un enracinement par *mid-point* pour votre arbre non-enraciné.