Synthèse projet de session IFT784

Réalisé par : Rami Abdellah

Sujet : Classification des séquences RNA

Dans ce projet j'avais pour objectif de créer un modèle informatique qui va essayer de classifier une séquence RNA à une classe parmi un set de familles choisies. Ainsi, le modèle va seulement fonctionner sur un nombre limité de familles de séquences RNA parmi ceux connues jusqu'à maintenant.

Pour faciliter la tâche je n'ai travaillé que sur 10 différentes familles RNA que j'ai choisi arbitrairement à partir de la base de données RFAM [1] qui est une collection de familles RNA. Les 10 familles choisies sont les suivantes :

- **RF01047**
- **RF01051**
- **RF01055**
- **RF01057**
- **RF01065**
- **RF01067**
- **RF01068**
- **RF01070**
- **RF01072**
- **RF01099**

J'ai donc commencé par télécharger pour chaque famille un fichier sous format Stockholm[1] contenant des séquences RNA appartenant à cette famille. Par la suite en utilisant le logiciel HMMER [2], qui est un outil d'analyse de bioséquence à l'aide de modèles de Markov cachés, j'ai généré pour chaque famille un fichier contenant son profile HMM[2], ce dernier est un modèle probabiliste qui encapsule les changements évolutifs qui se sont produits dans l'ensemble des séquences se trouvant dans le fichier sous format Stockholm de cette famille.

Par la suite, grâce à ces fichiers de profils HMM, j'ai pu générer avec le même outil HMMER de nouvelles séquences avec alignement, j'ai généré 1000 nouvelles séquences pour chacune des 10 familles.

La prochaine étape était d'extraire des motifs par rapport à chaque famille, qui vont la représenter et qui vont servir à entraîner le modèle de classification. J'ai décidé d'extraire pour chaque famille 10 motifs. Ces motifs vont être des chaînes de nucléotides qui se répètent le plus dans les séquences générées de la famille, et ceci en prenant en compte l'alignement des séquences, c'est-à-dire qu'un motif ne va être considérées qu'il est présent dans deux séquences que s'il est présent au même endroit dans les deux.

Pour choisir ces 10 motifs, pour une famille donnée, j'ai extrait toutes les combinaisons possibles de chaînes de nucléotides qui ont une longueur entre 4 et 20, et qui sont présents

[1] Stockholm format : Un format de fichier contenant l'alignement de séquences multiples [5]
[2] HMM : hidden Markov models

dans l'une des chaînes générées de cette famille. Puis grâce au profile HMM de cette famille, j'ai calculé pour chaque motif la probabilité qu'il soit généré avec une suite contiguë de nœuds du modèle de Markov tout en prenant en compte la probabilité de transition d'un nœud à l'autre avec un match[3]. Après calcul, j'ai prix pour chaque famille les 10 motifs qui donne la plus grande probabilité.

Par la suite j'ai combiné les motifs des 10 familles, donnant donc 100 motifs, ces derniers vont représenter les features du modèle qu'on va créer. En utilisant cette liste de features, j'ai généré une matrice où chaque colonne représente l'un des motifs, plus une 101éme colonne qui va contenir l'information sur la famille de la séquence. Puis j'ai fait le tour sur l'ensemble des séquences générées de toutes les familles, et pour chaque séquence j'ajoute à la matrice une ligne où chacune des colonnes correspondantes à un motif contient 0 ou 1 selon si ce motif est présent dans cette séquence ou pas, puis la dernière colonne contient la famille de cette séquence.

Cette matrice va représenter les données qu'on va utiliser pour entraîner et tester notre modèle. Elle contient 10 000 instances, dont on va choisir aléatoirement 70% pour l'entraînement et le reste pour le test. Pour créer le modèle, J'ai utilisé l'algorithme de la régression logistique, avec un maximum de 1000 itérations. Pour l'implémentation de l'algorithme j'ai utilisé la fonction "*LogisticRegression*" de la bibliothèque Scikit-learn [3].

La justesse obtenue avec ce modèle calculée sur les données de test était de : **90.59 %**

En guise d'analyse des résultats, j'ai calculé la moyenne du nombre de fautes de classifications faites par rapport à chaque famille, et ce sur trois essais qui diffère sur la division aléatoire des données en ceux d'entrainement et de test. Puis j'ai essayé de comparer ces 10 valeurs avec la longueur moyennes des séquences générées de chaque famille, ainsi que le niveau de conservation[4] de chaque famille extrait à partir du fichier Stockholm de ces derniers.

Nos attentes sont que plus la longueur moyenne des chaînes est petite plus il sera facile de classifier correctement la séquence et donc plus on aura moins d'erreurs, ainsi, la longueur moyenne des séquences doit être corrélée positivement au nombre d'erreurs de classification. Puis, on s'attend aussi à ce que plus les séquences d'une famille sont conservées, plus y'aura peut d'erreur, donc le niveau de conservation doit être corrélé négativement au nombre d'erreurs par famille. Et donc la variable de la longueur moyenne des séquences divisé par le niveau de conservation de chaque famille doit être corrélé positivement au nombre d'erreurs de classification dans famille.

En appliquant ces calculs pour voir si nos résultats reflètent notre analyse, j'ai trouvé grâce à la fonction " corrcoef " de la bibliothèque Numpy [4], qui calcul les coefficients de corrélation de Pearson, un taux de corrélation de **0.61** ce qui signifie que notre analyse est plus ou moins correcte.

[3] Le match est l'un des trois états d'un nœud du modèle de Markov parmi *Match, Insert* et *Delete*
[4] Une mesure du niveau de ressemblance entre les séquences de la famille

Références

[1] «Rfam,» [En ligne]. disponible: https://rfam.xfam.org/. [Accès le 12 6 2020].

[2] «biosequence analysis using profile hidden Markov models,» HMMER, [En ligne]. disponible: http://hmmer.org/. [Accès le 12 6 2020].

[3] «scikit learn,» [En ligne]. disponible: https://scikit-learn.org/. [Accès le 10 06 2020].

[4] «Numpy - numerical computing with Python,» [En ligne]. disponible: https://numpy.org/. [Accès le 24 05 2020].

[5] wikipedia, «Stockholm format,» [En ligne]. disponible: https://en.wikipedia.org/wiki/Stockholm_format. [Accès le 10 08 2020].