Conception d'un programme d'alignement d'embedding par programmation dynamique

Introduction

Les alignements de séquences protéiques sont couramment utilisés en bio-informatique afin de détecter des similitudes entre deux ou plusieurs séquences protéiques ou fragments de séquences. Dans les cas des alignements par paire où deux séquences sont alignées l'une par rapport à l'autre, trois approches peuvent être abordées : l'alignement global qui permet la détection de similitude sur la séquence entière, l'alignement local permettant la détection de similitude sur un fragment de séquence et l'alignement semi-global qui permet l'identification de similitude pour des séquences chevauchantes. Des méthodes ont été mises en place afin de réaliser ces alignements comme la méthode de Needleman et Wunsch[1] pour les alignements globaux et la méthode de Smith et Waterman[2] pour les alignements locaux.

Lors des alignements de séquences protéiques cités ci-dessus, seule la structure primaire de la protéine est considérée. Cependant, les protéines possèdent des propriétés physico-chimiques, des repliements et des fonctions. De plus, au cours de l'évolution, les structures tridimensionnelles des protéines sont plus conservées que leur séquence primaire. Les alignements de séquences protéiques seraient donc plus précis s'ils prenaient en compte ces caractéristiques.

Une méthode pouvant améliorer les résultats des alignements est l'utilisation d'embeddings. Lorsqu'une protéine est encodée en embedding, chaque position de la séquence correspond à un vecteur de plusieurs dimensions. Ces dimensions représentent différentes caractéristiques physico-chimiques et incluent les repliements et fonctions de la protéine.

Ainsi, le but de ce travail est de réaliser un programme permettant d'aligner des séquences protéiques représentées par un embedding grâce à la programmation dynamique.

Matériel et méthodes

Séquences sous la forme d'embeddings et fasta

La méthode T5 ProtTrans a permis d'encoder 1033 fichiers d'embeddings correspondant à 1033 structures protéiques où chaque position d'acide aminé est représentée par un vecteur de 1024 valeurs.

Les séquences de structures protéiques au format FASTA sont utilisées pour réaliser les alignements.

TMscore

Les TMscores ont été mesurés pour chaque paire de structures protéiques. Ce score permet d'identifier des ressemblances significatives entre protéines et a été utilisé pour extraire deux paires de séquences de références avec des TMscores différents :

- Deux séquences ayant un TMscore élevé : arch_histone_1a7w par rapport à histone_1aoic avec un TMscore de 0.85442.

  arch_histone_1a7w correspond à l'histone HMFB de *Methanothermus fervidus*[3] et histone_1aoic correspond à la chaîne C d'un complexe de particule de noyau de nucléosome de *Xenopus laevis*[4]. Ces deux protéines sont majoritairement composées en hélice alpha et ont comme fonction de se lier à l'ADN.

- Deux séquences ayant un TMscore « moyen » : 6PF2K_1bif par rapport à 5_3_exonuclease_1bgxt avec un TMscore 0.30667.

6PF2K_1bif[5] correspond à une enzyme qui réalise la synthèse et la dégradation du fructose 2,6-bisphosphate et elle est présente chez le rat. 5_3_exonuclease_1bgxt[6] à une exonucléase de l'ARN polymérase Taq présente chez la souris et la grenouille. Ces deux protéines sont majoritairement structurées en hélice alpha.

Les séquences sélectionnées serviront à mesurer l'efficacité des algorithmes dans la recherche de ressemblance structurales depuis les embeddings de séquences.

Calcul du produit scalaire

Une matrice bidimensionnelle est créée où une ligne correspond à chaque vecteur de l'embedding 1 et une colonne correspond à chaque vecteur de l'embedding 2. Le programme remplit la matrice en calculant le produit scalaire à chaque position. Plus le résultat sera positif, plus les deux vecteurs seront corrélés positivement et inversement. Cette matrice est utilisée comme matrice de score pour les alignements suivants.

Alignement global d'embedding par la méthode Needleman et Wunsch (ND)

A l'aide de la matrice de score, une nouvelle matrice bidimensionnelle est créée et est appelée matrice transformée. Une ligne correspond à chaque position de la séquence protéique 1 et une colonne correspond à chaque position de la séquence protéique 2. L'algorithme complète la matrice en affectant à la position la valeur maximum entre les trois positions (trois chemins possibles) qui l'entourent. La première valeur est initialisée dans ce projet à 0.

Pour une matrice M où i correspond au numéro de ligne et j le numéro de colonne, la position M(i,j) prendra la valeur maximale entre :

- M(i-1, j-1) + score de la matrice du produit scalaire à la position (i,j)
- M(i, j-1) + pénalité de gap
- M(i-1, j) + pénalité de gap

Une fois la matrice remplie, le chemin optimal est celui qui part du point à l'extrémité en bas à droite de la matrice et qui mène au point (0,0)

Alignement semi-global d'embedding basé sur la méthode Needleman et Wunsch

Cette méthode est similaire à l'alignement de Needleman et Wunsh. La matrice transformée est complétée de la même manière à l'exception qu'il n'y a pas de pénalité de gap en début et en fin d'alignement. Dans le cadre de ce projet, la première colonne est initialisée à zéro, pour une pénalité de gap fixe. Le traçage du chemin optimal part de la valeur maximum dans la dernière colonne et se termine dans la première colonne (n'importe quelle position possible).

Alignement local d'embedding par la méthode Smith et Waterman

Comme précédemment, à l'aide de la matrice de score, une nouvelle matrice bidimensionnelle est créée et est appelée matrice transformée. L'algorithme complète la matrice en affectant à la position la valeur maximum entre les trois positions (trois chemins possibles) qui l'entourent. La première valeur est initialisée à 0, et tant qu'une valeur maximale n'a pas un score supérieur à zéro, celle-ci prends la valeur de 0. La valeur à la position M(i,j) sera la valeur maximale entre:

- M(i-1, j-1) + score de la matrice du produit scalaire à la position (i,j)
- M(i, j-1) + pénalité de gap
- M(i-1, j) + pénalité de gap
- 0

Ainsi, le chemin optimal est celui qui part de la valeur maximale de la matrice et qui se termine lorsqu'il rencontre un zéro.

Pénalités de gap fixe et affine

Lorsque la pénalité de gap est fixe, elle prend la valeur de 0. Sinon, la pénalité de gap affine se déroule comme ceci : -1 pour une ouverture de gap et 0 pour une extension de gap.

Résultats

Les résultats des alignements des deux paires de séquences sélectionnées sont représentés en figure 1.

Figure 1 : (A) Résultat des alignements des séquences arch_histone_1a7w (bleu) par rapport à histone_1aoic (orange) pour chaque méthode (5' vers 3'). (B) Résultat de la fin des alignements de 6PF2K_1bif par rapport à 5_3_exonuclease_1bgxt pour chaque méthode (5' vers 3').

Les résultats montrant la différence entre pénalité de gap fixe et affine sont représentés en figure 2.

Figure 2 : Résultat des alignements des séquences arch_histone_1a7w (-1-) par rapport à histone_1aoic (-2-) pour deux alignements globaux avec pénalité de gap fixe et affine.

Discussion

Les résultats d'alignements pour une paire de séquence par les différents algorithmes sont semblables. La figure 1A met en évidence une similarité de région pour les séquences arch_histone_1a7w par rapport à histone_1aoic, ce qui est concordant car ce sont toutes les deux des histones majoritairement structurées en hélice alpha avec un TMscore élevé. Au contraire, les résultats d'alignement des séquences 6PF2K_1bif par rapport à 5_3_exonuclease_1bgxt ne mettent pas en évidence de régions similaires dans les séquences, ce qui est en accord avec le TMscore. Si ces mêmes alignements avaient été réalisés à partir des séquences FASTA, le pourcentage de similarité aurait été faible (proportionnellement au nombre de gap et à la valeur des pénalités de gap). Cependant, les séquences arch_histone_1a7w et histone_1aoic présentent des fonctions et structures proches, ce qui met en avant l'intérêt d'utiliser des embeddings pour des alignements.

La différence entre le programme d'alignement global avec pénalité de gap fixe ou affine est montrée en figure 1B. Ces résultats ne semblent pas tout à fait satisfaisants car peu de différences sont observées. Il serait intéressant de mettre des pénalités de gap plus négatives pour observer leur impact sur les alignements. De plus, il faudrait laisser le choix à l'utilisateur de mettre les pénalités qu'il souhaite, que ce soit pour une pénalité de gap affine ou fixe.

Conclusion

Un programme d'alignement de séquence sous la forme d'embeddings à été réalisé. Ce programme réalise des alignements globaux, locaux et semi-globaux avec des pénalités de gap fixes ou affines. Une piste d'amélioration serait la possibilité de laisser à l'utilisateur le choix des valeurs des pénalités de gap. Aligner depuis des embeddings permet de prendre en compte la structure, fonction et séquence de la protéine ce qui améliore la précision des alignements et donc de la détection de séquence homologue.

Bibliographie

1. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453 (1970).
2. Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195–197 (1981).
3. Decanniere, K., Babu, A. M., Sandman, K., Reeve, J. N. & Heinemann, U. Crystal structures of recombinant histones HMfA and HMfB from the hyperthermophilic archaeon Methanothermus fervidus. *J. Mol. Biol.* **303**, 35–47 (2000).
4. Luger, K., Mäder, A. W., Richmond, R. K., Sargent, D. F. & Richmond, T. J. Crystal structure of the nucleosome core particle at 2.8 A resolution. *Nature* **389**, 251–260 (1997).
5. Hasemann, C. A., Istvan, E. S., Uyeda, K. & Deisenhofer, J. The crystal structure of the bifunctional enzyme 6-phosphofructo-2-kinase/fructose-2,6-bisphosphatase reveals distinct domain homologies. *Struct. Lond. Engl. 1993* **4**, 1017–1029 (1996).
6. Murali, R., Sharkey, D. J., Daiss, J. L. & Murthy, H. M. Crystal structure of Taq DNA polymerase in complex with an inhibitory Fab: the Fab is directed against an intermediate in the helix-coil dynamics of the enzyme. *Proc. Natl. Acad. Sci. U. S. A.* **95**, 12562–12567 (1998).

Annexe

Arborescence de travail

L'organisation des répertoires et fichiers utilisés pour ce projet est représentée en figure 3.

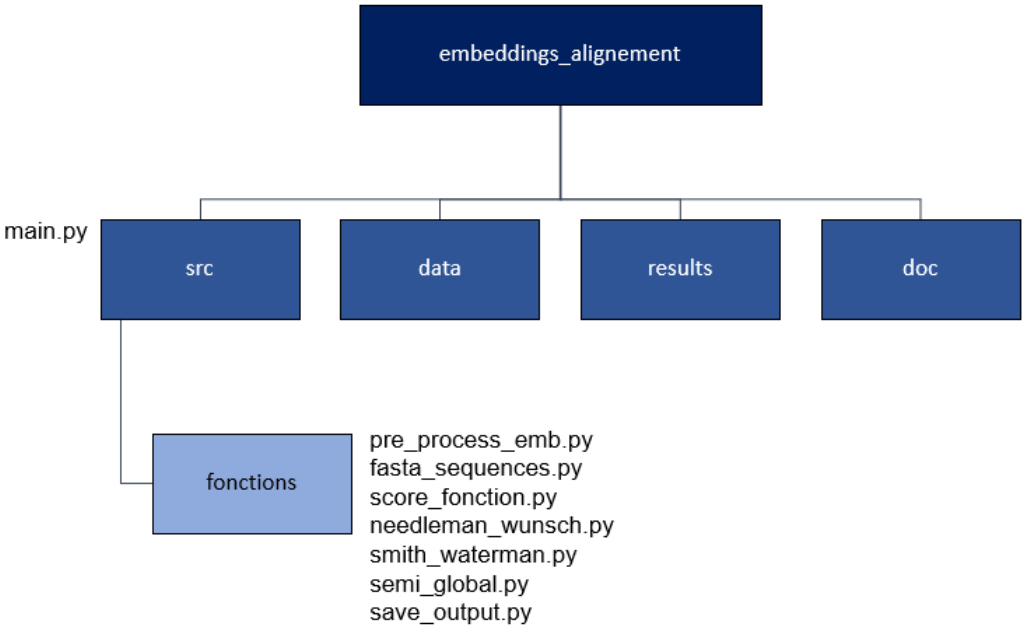

Figure 3 : Arborescence du projet. Dans les rectangles bleus, le répertoires. En noir, le nom des fichiers.

Fonctionnement du programme

Le programme prend comme entrée une commande bash avec les séquences à aligner en format *.t5emb* et *.fasta*. Il faut indiquer le type d'alignement (global, local ou semi-global) et rajouter un paramètre « -g affine » si les pénalités de gap doivent être de -1 pour une ouverture de gap et de 0 pour une extension de gap.

Dans le fichier principal nommé « main.py », différentes fonctions sont appelées par rapport aux arguments de l'input. Voici une courte description de chaque scripts :

- les fichiers fasta sont lu et mis sous forme de liste dans le script fasta_sequences.py
- les embeddings sont lu et mis sous forme de liste dans le script pre_process_emb.py
- la matrice de score (matrice de produit scalaire) est calculée dans le script score_fonction.py
- la matrice transformée et l'alignement global avec pénalité de gap fixe et affine sont réalisés dans le script needleman_wunsch.py
- la matrice transformée et l'alignement local avec pénalité de gap fixe et affine sont réalisés dans le script smith_waterman.py
- la matrice transformée et l'alignement semi-global avec pénalité de gap fixe et affine sont réalisés dans le script semi_global.py
- les résultats sont sauvegardés et mis dans un fichier *.txt* grâce au script save_output.py

La sortie du programme est un fichier *.txt* nommé ainsi :

" nomEmbedding1__nomEmbedding2_method__(affine_gap)_alignement.txt"

Le fichier est composé d'une séquence par ligne, alignées avec la deuxième. Dans le cas de plusieurs alignements optimaux, les paires de séquences alignées sont séparées par deux retours à la ligne.

Exemple d'utilisation du programme

Les séquences 6PF2K_1bif par rapport à 5_3_exonuclease_1bgxt sont utilisées comme exemple d'alignement.

Vous pouvez vous référer au README présent sur github (https://github.com/AdrianaLecourieux/embeddings_alignment/blob/main/README.md) pour toutes les étapes. Une fois que vous êtes dans le bon répertoire :

- Pour réaliser un alignement global avec pénalité de gap fixe :

```
python main.py -emb1 ../data/embeddings/6PF2K_1bif.t5emb -emb2 ../data/embeddings/5_3_exonuclease_1bgxt.t5emb -f1 ../data/fasta_sequences/6PF2K_1BIF.fasta -f2 ../data/fasta_sequences/5_3_EXONUCLEASE_1BGXT.fasta -m global
```

- Pour réaliser un alignement global avec pénalité de gap affine :

```
python main.py -emb1 ../data/embeddings/6PF2K_1bif.t5emb -emb2 ../data/embeddings/5_3_exonuclease_1bgxt.t5emb -f1 ../data/fasta_sequences/6PF2K_1BIF.fasta -f2 ../data/fasta_sequences/5_3_EXONUCLEASE_1BGXT.fasta -m global -g affine
```

- Pour réaliser un alignement local avec pénalité de gap fixe :

```
python main.py -emb1 ../data/embeddings/6PF2K_1bif.t5emb -emb2 ../data/embeddings/5_3_exonuclease_1bgxt.t5emb -f1 ../data/fasta_sequences/6PF2K_1BIF.fasta -f2 ../data/fasta_sequences/5_3_EXONUCLEASE_1BGXT.fasta -m local
```

- Pour réaliser un alignement local avec pénalité de gap affine :

```
python main.py -emb1 ../data/embeddings/6PF2K_1bif.t5emb -emb2 ../data/embeddings/5_3_exonuclease_1bgxt.t5emb -f1 ../data/fasta_sequences/6PF2K_1BIF.fasta -f2 ../data/fasta_sequences/5_3_EXONUCLEASE_1BGXT.fasta -m local -g affine
```

- Pour réaliser un alignement semi-global avec pénalité de gap fixe :

```
python main.py -emb1 ../data/embeddings/6PF2K_1bif.t5emb -emb2 ../data/embeddings/5_3_exonuclease_1bgxt.t5emb -f1 ../data/fasta_sequences/6PF2K_1BIF.fasta -f2 ../data/fasta_sequences/5_3_EXONUCLEASE_1BGXT.fasta -m semi_global
```

- Pour réaliser un alignement semi-global avec pénalité de gap affine :

```
python main.py -emb1 ../data/embeddings/6PF2K_1bif.t5emb -emb2 ../data/embeddings/5_3_exonuclease_1bgxt.t5emb -f1 ../data/fasta_sequences/6PF2K_1BIF.fasta -f2 ../data/fasta_sequences/5_3_EXONUCLEASE_1BGXT.fasta -m semi_global -g affine
```

Difficultés rencontrées

Ma première (et majeure) difficulté rencontrée est liée au manque de temps que nous avons eu pour réaliser ce projet. Il s'agissait surtout pour ma part des cours qui concernait la programmation qui sont tombés durant la première semaine que nous avions pour commencer le projet. Je pense qu'il serait favorable d'avoir ces cours là avant, pour ne pas perdre de temps (ce qui aurait peut-être permis de coder en OOP).

Ma seconde difficulté a été de réaliser les différents algorithmes d'alignements ainsi que les pénalités de gap fixe et affine.