Clémence Lauden

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

Introduction

L'alignement de protéines est une étape nécessaire pour déterminer certaines propriétés chez les protéines, notamment pour déterminer leur degré d'homologie mais également pour analyser leur similarité structurale et donc identifier des sites actifs.
Il existe différentes méthodes d'alignements de protéines. Généralement les alignements sont réalisés à partir des séquences d'acides aminés des protéines. Cependant, il est possible d'en réaliser à partir d'un embedding comportant de nombreux paramètres (notamment biophysique) pour chaque acide aminé d'une protéine.
Les alignements réalisés à partir d'embedding pourraient donc être plus précis pour la réalisation d'alignement (Bepler et al., 2021).

Le but de ce projet a été de créer un programme produisant un alignement de protéines à partir d'embedding, selon 3 méthodes d'alignement différentes (locale, global et semi-global).

Matériels et méthodes

Ce programme réalise des alignements pour 3 méthodes d'alignements différentes : local ou Smith-Waterman (Smith et al., 1981), global ou Needleman-Wunsch (Needleman et al., 1970) et semi-global. Ces méthodes fonctionnent sur le même principe en deux étapes : d'abord la production d'une matrice de score et ensuite un alignement par backtracking sur cette matrice.
La matrice de score diffère légèrement selon si l'alignement est global ou local (la matrice pour le semi-global est la même que pour le global). Dans les deux cas, la matrice comporte en abscisse et en ordonnée les séquences d'acide aminé des protéines. Cependant, elle comporte une ligne et une colonne de plus au début ne correspondant à aucun acide aminé.
Dans le programme, un éléments vide est ajouté aux séquences de protéines avant le lancement des fonctions de construction des matrices pour gérer ce contexte.
Le remplissage de la matrice se fait ensuite selon la méthode voulue, global ou local avec quelques différences.
Pour un alignement local, les pénalités de gap ne sont pas comptées sur la première ligne et la première colonne. De plus, une valeur de la matrice ne peut pas être négative. Dans un alignement global, les pénalités de gap sont comptées sur toutes les lignes et colonnes et les valeurs de la matrice peuvent être négatives.

Ensuite, l'alignement est déterminé par backtracking selon la méthode sélectionnée. La différence entre les méthodes pour le backtracking est au niveau du début et de la fin de la procédure. Dans le cas d'un alignement global, le backtracking commence en bas à droite de la matrice et fini en haut à gauche. Pour un semi-global, il commence dans la colonne de droite, là où la valeur est la plus importante et se termine quand la première ligne est atteinte. Pour un alignement local, le début se fait sur la valeur maximale de la matrice et se termine dès qu'une valeur de 0 est atteinte.

Le programme a été développé en python 3.10 et utilise le module numpy 1.23.
Il est disponible sur https://github.com/clelauden/pc_almnt_embedding.git

Résultat

Pour adapter ces méthodes à l'utilisation d'embedding, la valeur de similarité utilisée pour produire la matrice de score est le dot product entre les 2 matrices contenant les embedding pour les 2 protéines.
La fonction add_col_row rajoute une ligne et une colonne à la matrice de dot product pour lui donner la même taille que la matrice d'alignement.

La valeur des gaps est fixée à 0.

Comparons le résultat de l'alignement des protéines 1BIF et 1SHKA avec une méthode locale (Smith-Waterman) produite à partir de ce programme avec un alignement de ces protéine avec Blastp (obtenue sur le site du NCBI) utilisant également un alignement local.

Ces alignements sont assez différents. L'alignement réalisé par Blastp aligne la protéine 1SHKA sur 1BIF, ainsi il n'y a pas de gap dans cette protéine. Le sens des séquences est également inversé par rapport au programme. De plus, l'alignement du programme est plus long (163) que celui de Blastp (83), et les gap ne correspondent pas.

a)

```
1BIF - IRRMFDETAEDSDRNV-DPSGLKVQVINAAIVEPDVCISEVFFTKYGN-EGFNFIMARRERTTNTADFVAVHGGEESLFKRVDNLL--AALACQKRIKLGEENNDPLFFEFSKYTKVMDRR-QGVNFERTPVGIFNLYRTLKKSIYTKGRAPLGVMVILTPC
1SHKA - APLRMTQMLECVIAAPPPQTADVVYHAVDQYLAERERLVAEMEEAIQLLRLALEEAPAFLYVVTGHARMFQRNQELLVMGGGTAV-VV--PTAVAQLAESERR-FGPWGEAAVVDAVTMGSSTHQMFIDTDVFEYGLARALERGVTTKGCGRAGVMFIPETM
```

b)

Range 1: 5 to 82 Graphics

Score	Expect	Method	Identities	Positives	Gaps
25.0 bits(53)	4e-04	Compositional matrix adjust.	23/84(27%)	36/84(42%)	6/84(7%)

```
Query  5   IVMVGLPARGKTYISKKLTRYLNFIGVPTREFNVGQYRRDMVKTYKSFEFFLPDNEEGLK  64
           I MVG    GKT + ++L R L +  V T  F   Q+   M       +    +   G +
Sbjct  5   IFMVGARGCGKTTVGRELARALGYEFVDTDIFM--QHTSGMTVA----DVVAAEGWPGFR  58

Query  65  IRKQCALAALNDVRKFLSEEGGHV  88
            R+  AL A+    + ++  GG V
Sbjct  59  RRESEALQAVATPNRVVATGGGMV  82
```

Figure 1. Alignement des protéines 1BIF et 1SHKA avec une méthode locale, a) par ce programme (pour plus de lisibilité les noms des séquences ont été rajoutés devant les séquences bien qu'ils ne soient pas présent dans le fichier de sorti originel), b) par Blastp. La partie alignée par Blastp est encadrée dans a)

L’alignement réalisé par le programme semble peu cohérent. En effet, deux problèmes ont été identifiés. D’une part, au moment du backtracking pour la méthode locale, il ne prend pas en compte le cas où il y a plusieurs valeurs maximums dans la matrice. D’autre part, pendant le backtracking, au moment de choisir si la séquence est alignée ou s’il y a un gap, le programme favorise le choix de l’alignement, ce qui conduit à une forte sous-estimation du nombre de gap dans l’alignement.

Conclusion

Le programme pourrait être amélioré en réglant les problèmes identifiés et en ajoutant des options pour rendre le programme plus simple d’utilisation et plus polyvalent.
Concernant l’ergonomie, la saisie des fichiers d’entrée pourrait être simplifiée, par exemple en permettant à l’utilisateur de sélectionner les fichiers dans une liste au lieu d’entrer leur nom. De plus, les fichiers d’entrée pourraient être réunis dans un même dossier.
L’alignement pourrait être amélioré en ajoutant la possibilité d’utiliser une pénalité de gap affine plutôt que fixe.

L’utilisation d’embedding pour les alignements de protéines ne semble pas optimal dans tous les cas. En effet, des résultats récents suggèrent que si les alignements par embedding fonctionnent particulièrement bien dans le cas d’identification de domaine ayant des relations anciennes et pour les protéines à un seul domaine, le cas des protéines multidomaines peuvent poser problème avec les techniques actuelles (Schütze et al., 2022).

Bibliographie

Bepler T, Berger B. Learning the protein language: Evolution, structure, and function. Cell Syst. 2021 Jun 16;12(6):654-669.e3. doi: 10.1016/j.cels.2021.05.017. PMID: 34139171; PMCID: PMC8238390

Smith, Temple F. & Waterman, Michael S. (1981). "Identification of Common Molecular Subsequences" (PDF). Journal of Molecular Biology. 147 (1): 195–197. CiteSeerX 10.1.1.63.2897. doi:10.1016/0022-2836(81)90087-5. PMID 7265238

Needleman, Saul B. & Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". Journal of Molecular Biology. 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325.

Konstantin Schütze, Michael Heinzinger, Martin Steinegger, Burkhard Rost, Nearest neighbor search on embeddings rapidly identifies distant protein relations, bioRxiv. Preprint. 2022.09.04.506527; doi: https://doi.org/10.1101/2022.09.04.506527

Annexe

Exemple de fonctionnement du programme

Initialisation

En entrée, le programme demande de donner le nom des 4 fichiers à partir desquels est réalisé l'alignement puis le type d'alignement. Pour cet exemple, nous alignerons la protéine 1BIF et 1SHKA avec une méthode locale (Smith-Waterman).

Module

Pour fonctionner, le programme a besoin du module Numpy. Ce module est installé dans l'environnement fourni sur Git-hub, l'utilisateur n'a donc pas à s'occuper de son installation.

Mise en forme des données

Tout d'abord, le programme va lire les fichiers mis en input et mettre en forme les données.
Les données d'embedding vont être mis sous forme d'array. Pour les fichiers fasta de séquence, seule la deuxième ligne est lue (celle contenant la séquence). La séquence est ensuite transformée en liste où chaque acide aminé est un élément de la liste. Enfin, un élément vide est ajouté devant les listes, ces dernières sont nommées prot1 et prot2.

Calcul du dot product

Le programme commence par inverser l'array contenant le deuxième embedding pour permettre le calcul de la matrice, puis le dot product est calculé et est nommé dotp.

Production de l'alignement

La production de l'alignement se fait par une succession de 5 fonctions faisant appel les unes aux autres. La structure du programme est détaillée dans la figure 2.
La fonction *result_file* est la première à être lancée. Elle fonctionne avec prot1, prot2 et dotp en entrée. Elle appelle la fonction Alignement qui appelle l'une des 2 fonctions de production de matrice de score selon la méthode d'alignement sélectionnée en entrée (*matrice_sw* pour un alignement local et *matrice_nw* pour global et semi-global). Les deux fonctions qui produisent les matrices appellent la fonction *add_col_row*. Après constitution des matrices, la fonction Alignement appelle l'une des trois fonctions de backtracking selon la méthode d'alignement choisie (*backtrack_sw* pour la méthode local, *backtrack_nw* pour global et *backtrack_sg* pour semi-global).

Production de la matrice de score

Etant donné que la méthode choisie est locale, la fonction *matrice_sw* est lancée par la fonction Alignement. Cette fonction utilise prot1, prot2 et dotp ainsi que le gap (ici fixé à 0) en paramètres d'entrée. Elle commence par appeler la fonction *add_col_row* qui va ajouter une colonne et une ligne vide au début de la matrice de dot product.
La fonction va ensuite créer une matrice remplie de 0 de la taille des séquences protéiques. Cette matrice va maintenant être remplie selon l'algorithme de Smith-Waterman. (Smith et al., 1981).

Production de l'alignement par backtracking

Une fois que la fonction *matrice_sw* a produit la matrice de score (nommée result), la fonction Alignement appelle la fonction *backtrack_sw* avec les paramètres prot1, prot2 et result.
Pour un alignement local, le backtracking commence au point où la valeur de la matrice est maximum. Ainsi la fonction commence par trouver et sélectionner les coordonnées pour lesquelles cette condition est remplie.
La fonction créé ensuite deux chaînes de caractères vides où seront notées les séquences alignées pendant le backtracking. Le backtracking est ensuite réalisé selon l'algorithme de Smith-Waterman (Smith et al., 1981).

Alignement et Création de fichier de sortie

Une fois le backtracking réalisé, la fonction Alignement va renvoyer les alignements à la fonction *result_file*.
Etant donné que dans les 3 cas de backtracking, l'alignement se fait de la fin de la matrice jusqu'au début, la fonction va commencer par inverser les alignements pour que les séquences soient dans le même sens que celles d'entrée dans un soucis de lisibilité.
Ensuite la fonction va écrire ces deux séquences d'alignement dans un fichier texte appelé alignement.txt (Fig1. a) dans le dossier principal.

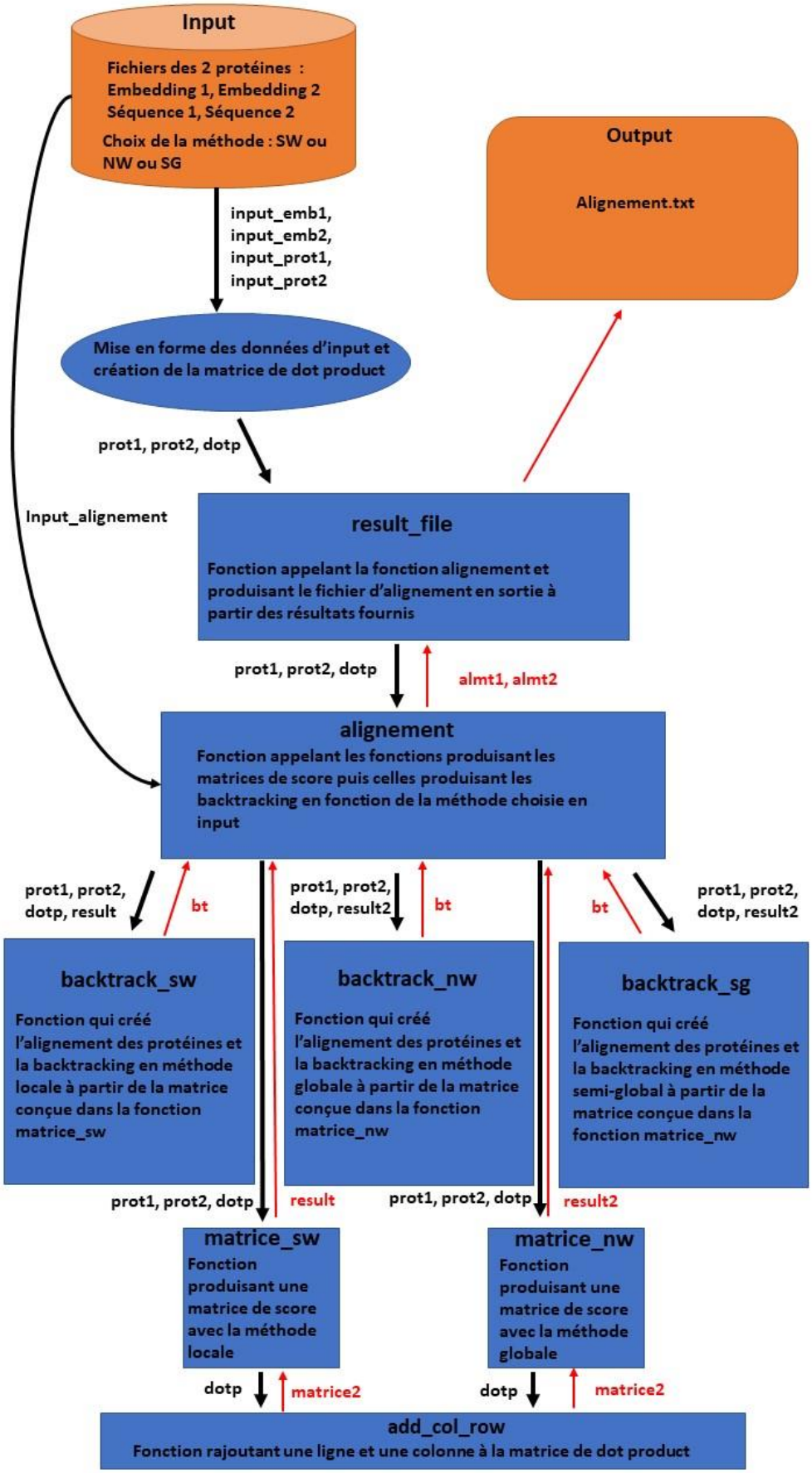

Figure 2. Schéma du fonctionnement du programme, les fonctions sont représentées par des rectangles bleu, les commandes hors fonctions par un ovale bleu, les fichiers d'entrée et de sortie par un cylindre orange, les paramètres d'entrées par des flèches noires et les sorties par des flèches rouges.