ALIGNEMENT GLOBAL DE CHAINES DE SÉQUENCES

(algorithme de Needleman-Wunsch)

25 octobre 2020

.

1 Principe

L'algorithme de **Needleman-Wunsch** est un algorithme qui effectue un alignement global maximal de deux chaînes de caractères. Il est couramment utilisé en **bio-informatique** pour aligner des séquences de protéines ou de nucléotides. L'algorithme de Needleman-Wunsch est un exemple de **programmation dynamique**, tout comme l'algorithme de **Wagner et Fischer** pour le calcul de la distance de *Levenshtein* auquel il est apparenté. Il garantit de trouver l'alignement de score maximal. Ce fut la première application de la **programmation dynamique** pour la comparaison de séquences biologiques.

Les scores pour les caractères alignés sont spécifiés par une matrice de similarité. Ici, $\mathrm{S(i,j)}$ est la similarité des caractères i et j. Elle utilise une 'pénalité de trou', appelée ici d.

Exemple : on d = -5 , deux sequences AGACTAGTTAC et CGA—GACGT et

$$S = \begin{pmatrix} - & A & G & C & T \\ A & 10 & -1 & -3 & -4 \\ G & -1 & 7 & -5 & -3 \\ C & -3 & -5 & 9 & 0 \\ T & -4 & -3 & 0 & 8 \end{pmatrix}$$

Alors on a le score suivant :
S(A,C)+S(G,G)+S(A,A)+3×d+S(G,G)+S(T,A)+S(T,C)+S(A,G)+S(C,T)
=-3+7+10-3×5+7+-4+0+-1+0=1

2 Algorithme

Pour déterminer l'alignement de score maximal, un tableau bidimensionnel, ou matrice est utilisé. Cette matrice est appelée matrice F, et ses éléments aux positions (i, j) sont notés F_{ij}. Il y a une colonne pour chaque caractère de la séquence A, et une ligne pour chaque caractère de la séquence B.

Au fur et à mesure de la progression de l'algorithme, F_{ij} se verra affecter le score optimal pour l'alignement des i premiers caractères de A avec les j premiers caractères de B.

Une fois que la matrice F est calculée, on voit que l'élément (i, j) correspond au score maximum pour n'importe quel alignement. Pour déterminer quel alignement fournit ce score, il faut partir de cet élément (i, j), et effectuer le 'chemin inverse' vers l'élément (1,1), en regardant à chaque étape à partir de quel voisin on est partis. S'il s'agissait de l'élément diagonal, alors A(i) et B(i) sont alignés. S'il s'agissait de l'élément (i-1,j), alors A(i) est aligné avec un trou, et s'il s'agissait de l'élément (i, j-1), alors B(j) est aligné avec un trou.

```
Function Needleman-Wunsch(S, A, B){

        /* On vas construire la matrice F et trouver le scrore maximal */
        for i=0 to length(A)-1
                F(i, 0) = d*i
        enfor
        for j=0 to length(B)-1
                F(0,j) = d*j
        endfor
        for i=1 to length(A)-1
                for j = 1 to length(B)-1
                        Choice1 = F(i-1,j-1) + S(A(i), B(j))
                        Choice2 = F(i-1, j) + d
                        Choice3 = F(i, j-1) + d
                        F(i, j) = max(Choice1, Choice2, Choice3)
                endfor
        endfor

        /*En ce moment F(i, j) contient le score maximal
```

```
        determinons maintenant les elements de A et B fournissant ce score */

        AlignmentA = ""
        AlignmentB = ""
        i = length(A) - 1
        j = length(B) - 1
        while (i > 0 AND j > 0)
                Score = F(i, j)
                ScoreDiag = F(i - 1, j - 1)
                ScoreUp = F(i, j - 1)
                ScoreLeft = F(i - 1, j)
                if (Score == ScoreDiag + S(A(i), B(j)))
                        AlignmentA = A(i) + AlignmentA
                        AlignmentB = B(j) + AlignmentB
                        i = i - 1
                        j = j - 1
                else if (Score == ScoreLeft + d)
                        AlignmentA = A(i) + AlignmentA
                        AlignmentB = "-" + AlignmentB
                        i = i - 1

                else (Score == ScoreUp + d)
                        AlignmentA = "-" + AlignmentA
                        AlignmentB = B(j) + AlignmentB
                        j = j - 1
                endif
        endwhile

        while (i > 0)

            AlignmentA = A(i) + AlignmentA
            AlignmentB = "-" + AlignmentB
            i = i - 1
        endwhile

        while (j > 0)

                AlignmentA = "-" + AlignmentA
                AlignmentB = B(j) + AlignmentB
                j = j - 1

        endwhile


}
```

3 Complexité

Si on aligne des séquences de taille n et m, le temps d'exécution de l'algorithme est $\bigcirc(nm)$, et l'espace mémoire utilisé est $\bigcirc(nm)$ (cependant, il existe une version modifiée de l'algorithme, qui utilise un espace mémoire en $\bigcirc(n + m)$, mais a un temps d'exécution plus long. Cette modification est en fait une technique générale en programmation dynamique ; elle fut introduite dans l'algorithme d'Hirschberg)