Spis rzeczy

3 Przeszukiwanie baz danych **35**
3.1 Heurystyczne algorytmy 35
3.1.1 FASTA 35
3.1.2 BLAST 37
3.2 Macierze substytucyjne 38
3.2.1 PAM 38
3.2.2 BLOSUM 40

3 Przeszukiwanie baz danych

Jest dużo baz dnaych specjalizujących się w różnych aspektach związanych z sekwencjami naturalnie pojawiającymi się w biologii. Dla sekwencji DNA główne bazy danych to: GenBank (USA), EMBL (Europa), DDBJ (Japonia). Natomiast dla białek główną bazą danych jest Swiss-Prot. Przeszukiwanie baz danych jest jedną z głównych metod pracy współczesnego biologa. Poniżej omówimy kilka zagadnień związanych z praktycznymi aspektami przeszukiwania baz.

3.1 Heurystyczne algorytmy

Omówimy dwa najbardziej popularne heurystyczne algorytmy używane do obliczania przybliżonej wartości lokalnego uliniowienia. Algorytmy te stanowią standardowe narzędzie do przeszukiwania baz danych w procesie wyszukiwania podobieństw pomiędzy sekwencjami.

3.1.1 FASTA

Jest to heurystyczny algorytm (Lipman, Pearson, 1985) służący do przybliżonego obliczania lokalnego uliniowienia danego wzorca Q względem tekstowej sekwencji T wziętej z bazy danych. Zwykle stosuje się go do kolejnych sekwencji z bazy danych. Na początku użytkownik wybiera liczbowy parametr, zwany *ktup*. Standardowo sugerowane wartości dla *ktup* to 6 dla sekwencji DNA oraz 2 dla białek. Przyjmijmy, że k jest wartością parametru *ktup*. Przez k-słowo będziemy rozumieć dowolne słowo długości k.

Niech $n = |Q|$ oraz $m = |T|$. Działanie algorytmu można przedstawić w następujących czterech krokach.

1. Dla $1 \leq i \leq n$, $1 \leq j \leq m$ algorytm znajduje pary (i, j), takie że k-słowo zaczynające się w Q w pozycji i jest identyczne z k-słowem zaczynającym się w T w pozycji j. Każda taka para (i, j) nazywa się *gorącym miejscem*. Operację tę można wykonać efektywnie sporządzając na początku tablicę haszującą dla Q, lub (rzadziej) dla wszystkich słów T z bazy danych.

2. Każde gorące miejsce (i, j) można traktować jako odcinek długości k leżący na przekątnej o numerze[1] $(i - j)$ w tablicy V (otrzymanej metodą dynamicznego programowania — V oczywiście nie mamy). Algorytm przypisuje pewne wartości dodatnie gorącym miejscom oraz wartości ujemne przerwom pomiędzy takimi miejscami (im dłuższa przerwa, tym mniejsza wartość). Dla każdej przekątnej zawierającej gorące miejsce, algorytm wybiera fragment pomiędzy gorącymi miejscami o maksymalnej wartości. W ten sposób zostaje wybranych 10 przekątnych (i zawartych w nich fragmentów) o maksymalnej wartości. Dla każdego z tych fragmentów algorytm znajduje część takiego fragmentu (podsłowo) o maksymalnej wartości uliniowienia bez spacji (do obliczania tej wartości stosuje się tablicę PAM lub BLOSUM) Taką część fragmentu nazwiemy *poduliniowieniem*. Niech *init1* będzie najlepszym poduliniowieniem.

3. Wybrane są poduliniowienia, których wartość przekracza pewną z góry ustaloną granicę. Z tych dobrych poduliniowań próbuje się ułożyć uliniowienie o maksymalnej wartości. W tym celu buduje się następujący graf poduliniowień. Wierzchołkami są poduliniowienia. Każdemu wierzchołkowi jest przypisana liczba będąca wartością tego poduliniowienia. Jeśli u i v są poduliniowieniami, takimi że u zaczyna się w pozycji (i, j) i kończy w pozycji $(i+d, j+d)$, a v zaczyna się w pozycji (i', j'), to tworzymy krawędź od u do v, gdy $i' > i + d$ oraz $j' > j + d$, tzn gdy wiersz (kolumna) w którym zaczyna się v jest poniżej wiersza (na prawo od kolumny), w którym kończy się u. Krawędzi tej przypisujemy pewną wagę zależącą od liczby spacji jakie trzeba wprowadzić we fragmencie lokalnego uliniowienia, w którym poduliniowienie v występuje po poduliniowieniu u. Im wększa liczba spacji tym waga takiej krawędzi jest mniejsza. Następnie algorytm znajduje drogę o maksymalnej wartości

[1] Główna przekątna ma numer 0, przekątne o numerach dodatnich leżą nad główną przekątną, a o numerach ujemnych pod główną przekątną.

w wyżej opisanym grafie. Taka droga wyznacza lokalne uliniowienie pomiędzy dwoma słowami. To nie musi być optymalne lokalne uliniowienie pomiędzy Q oraz T. Oznaczmy to uliniowienie przez *initn*.

4. Algorytm wraca do poduliniowienia *init1* z kroku 2 i znajduje najlepsze lokalne uliniowienie wokół przekątnej zawierającej *init1* w pasie $[-8, 8]$ (dla białek) oraz w pasie $[-16, 16]$ (dla DNA). Niech *opt* będzie takim uliniowieniem.

W ten spsób Q jest porównywane z kolejnymi słowami T z bazy danych. Biorąc pod uwagę *opt* lub *initn* wyznacza się małą liczbę słów T, najbardziej obiecujących z punktu widzenia uliniowienia z Q. Dla każdego z nich wykonuje się pełny algorytm Smitha-Watermana obliczający optymalne uliniowienia.

3.1.2 BLAST

Algorytm BLAST podaje jako wynik całe spektrum rozwiązań (uliniowień) wraz z oszacowaniem statystycznej ważności znalezionego rozwiązania (czyli prawdopodobieństwa tego, że znaleziona wartość, lub wartość od niej większa mogła się pojawić przypadkiem (z losowej sekwencji).

BLAST porównuje wzorzec Q z każdą sekwencją z bazy danych, starając się zidentyfikować te sekwencje T, dla których MSP (*maximal segment pair*, czyli para podsłów równej długości maksymalizująca wartość uliniowienia bez spacji[2]) jest większe od pewnej z góry ustalonej wartości C. W ten sposób wybiera się pewne słowa T "podejrzane" o pewne podobieństwo z Q.

Jak się szuka takich T, dla których MSP jest większe od C? Ustala się długość w oraz wartość graniczną t. Następnie BLAST znajduje wszystkie w-podsłowa w T, dla których istnieje w-podsłowo w Q o wartości uliniowienia (bez spacji) większej od t. Każde takie miejsce jest rozszerzane w celu znalezienia wartości uliniowienia większej od C. Jeśli w trakcie rozszerzania wartość uliniowienia (która może rosnąć lub maleć z każdym krokiem rozszerzenia) spadnie poniżej pewnej wartości progowej, to poszukiwania dla takiego miejsca są przerywane.

Dobór wartości C, w oraz t ma kluczowe znaczenie dla jakości znajdowanych wyników. Na przykład, dla porównywania białek w jest przyjmowane pomiędzy 3 a 5, natomiast dla DNA jest zwykle równe około 12.

[2]Przy użyciu pewnej macierzy substytucyjnej.

3.2 Macierze substytucyjne

Zacznijmy od paru ogólnych uwag. Załóżmy, że dla liter $a, b \in \Sigma$, $q_{a,b}$ jest prawdopodobieństwem tego, że aminokwasy a oraz b pochodzą od wspólnego aminokwasu (przodka) w drodze punktowej mutacji. Mamy dane dwa słowa $x = x_1 \dots x_n$ oraz $y = y_1 \dots y_n$. Chcemy obliczyć jakie jest prawdopodobieństwo tego że x oraz y pochodzą od wspólnego przodka w wyniku punktowych zmian. Dla dowolnej litery a, niech p_a oznacza prawdopodobieństwo wystąpienia litery a w słowie. Wówczas prawdopodobieństwo pojawienia się słów x oraz y to

$$\prod_{i=1}^{n} p_{x_i} \cdot \prod_{i=1}^{n} p_{y_i}.$$

Natomiast prawdopodobieństwo tego, że x i y pochodzą od wspólnego przodka jest równe $\prod_{i=1}^{n} q_{x_i,y_i}$. Iloraz tych dwóch wartości

$$\prod_{i=1}^{n} \frac{q_{x_i,y_i}}{p_{x_i} \cdot p_{y_i}}$$

nazywa się *odds ratio*. Im większa jest ta wartość tym bardziej dane dwa słowa x, y nie są całkowicie losowe (niezależne), ale pochodzą od wspólnego przodka. Ponieważ dużo wygodniej jest pracować z addytywną miarą podobieństwa, to przechodzimy do logarytmu:

$$s(x, y) = \sum_{i=1}^{n} \log(\frac{q_{x_i,y_i}}{p_{x_i} \cdot p_{y_i}}),$$

liczba

$$s(a, b) = \log(\frac{q_{x_i,y_i}}{p_{x_i} \cdot p_{y_i}})$$

jest karą/nagrodą za zamianę a na b.

3.2.1 PAM

Macierze PAM (*percent accepted mutations*) zostały zaproponawane przez M. Dayhoff i jej współpracowników około 1978r. Są to tzw. macierze substytucyjne dla aminokwasów i jako takie reprezentują funkcję podobieństwa. PAM również używa się jako jednostki miary opisującej ewolucyjną rozbieżność pomiędzy dwoma ciągami aminokwasów. Powiemy, że dwa słowa S_1 i S_2 różnią się o jedną jednoskę PAM, jeśli S_2 można otrzymać z S_1 w ciągu

akceptowalnych punktowych mutacji, tak że średnia liczba akceptowalnych mutacji na 100 aminokwasów wynosi 1. Akceptowalna punktowa mutacja to taka, która albo nie zmieniła funkcji białka, lub była korzystna dla organizmu (co najmniej nie spowodowała śmierci organizmu). Zwróćmy uwagę, różnica 1PAM pomiędzy S_1 i S_2 nie oznacza, że słowa te różnią się pomiędzy sobą o 1%. Liczba mutacji na pewnej pozycji w słowach może być większa od jedynki, a nawet w wyniku tych mutacji ta pozycja nie musi się zmienić.

Jako podstawę do budowy macierzy PAM wzięto pewną rodzinę bardzo podobnych białek (każde dwa nie różniły się o więcej niż 15%) i ręcznie sporządzono globalne uliniowienia dla tej rodziny. Następnie stworzono tablicę A, taką że dla aminokwasów a, b, $A(a, b)$ jest liczbą wystąpień ulinowienia pary (a, b) w wyżej wymienionych uliniowieniach. Wówczas prawdopodobieństwo mutacji z a na b jest równe

$$q_{a,b} = \frac{A(a,b)}{\sum_c A(a,c)}.$$

Niech p_a będzie prawdopodobieństwem wystąpienia litery a w w/w białkach. Wówczas wartość oczekiwana (średnia) liczby zamian w losowej parze słów długości m wynosi

$$m \cdot \sum_{a,b} p_a \cdot p_b \cdot q_{a,b}.$$

Ponieważ macierz 1PAM to taka, dla której powyższa wartość oczekiwana wynosi 1 dla $m = 100$, to musimy tak przeskalować $q_{a,b}$ na $q'_{a,b}$ aby zachodziło

$$\sum_{a,b} p_a \cdot p_b \cdot q'_{a,b} = 0.01.$$

Wystarczy stosownie dobrać stałą d i wziąć:

$$q'_{a,b} = \begin{cases} d \cdot q_{a,b} & \text{dla } a \neq b, \\ d \cdot q_{a,b} + (1-d) & \text{dla } a = b. \end{cases}$$

W ten sposób otrzymujemy 1PAM macierz. Oznaczmy ją przez $S(1)$. Tak więc (w przybliżeniu) $S(1)(a, b)$ jest prawdopodobieństwem zamiany a na b w jednej umownej jednostce czasu. Macierz $S(1)$ przedstawia pewien łańcuch Markowa. Prawdopodobieństwo zamiany a na b w n jednostkach czasu to $S(1)^n(a, b)$, gdzie $S(1)^n$ jest n-krotnym iloczynem macierzy $S(1)$ przez siebie. Macierze $S(1)^n$ nazywają się nPAM macierzami. Wartości do prawdziwej macierzy nPAM są brane z $S(1)^n$ przez stosowanie logarytmu, skalowanie i zaokrąglanie. Bardzo popularne są 250PAM macierze.

3.2.2 BLOSUM

Macierze substytucyjne BLOSUM (blocks substitution matrix) zostały zaproponawane przez S. oraz J. Heinkoff w 1991r. jako efekt krytyki tego, że macierze nPAM dla $n >> 1$ nie oddają dobrze wpływu czasu na zmiany sekwencji kodujących białka. Macierze BLOSUM zostały oparte na białkowej bazie danych BLOCKS w następujący sposób. Niech $0 \leq L \leq 100$. Opiszemy sposób budowania macierzy BLOSUML. Białka z bazy danych są dzielone na grupy. Do jednej grupy są zaliczane dwa białka jeśli można przejść od jednego białka do drugiego, używając białek pośrednich wziętych z bazy danych, takich że każde dwa kolejne białka mają skład identyczny w co najmniej L%. Białka w bazie danych BLOCKS są uliniowione w tzw. blokach. Tworzymy macierz A. Wybierzmy dwie grupy g, g'. Dla aminokwasów a, b, liczba $A^{g,g'}(a,b)$ jest częstością z jaką aminokwas a pochodzący z grupy g jest uliniowiony z aminokwasem b pochodzącym z grupy g' (liczba ta jest podzielona przez $n \cdot n'$, gdzie n jest liczebnością grupy g, a n' jest liczebnością grupy g'). $A(a,b)$ otrzymuje się przez sumowanie $A^{g,g'}(a,b)$ po wszystkich parach grup g, g'. Mając A możemy obliczyć prawdopodobieństwo wystąpienia aminokwasu a:

$$p_a = \frac{\sum_b A(a,b)}{\sum_{c,d} A(c,d)}$$

oraz prawdopodobieństwo zamiany a na b:

$$q_{a,b} = \frac{A(a,b)}{\sum_{c,d} A(c,d)}.$$

Wówczas

$$s(a,b) = \log(\frac{q_{a,b}}{p_a \cdot p_b}).$$

Najczęściej używane L to $L = 50$ oraz $L = 62$.