Liniowe dopasowanie dwóch sekwencji

Wiele informacji o działaniu genów i enzymów dostarcza wykrywanie „podobnych” sekwencji

- fragmenty podobne są zwykle skorelowane ewolucyjnie, mają wspólny rodowód,
- pełnią zbliżone funkcje w organizmie,
- białka o podobnej strukturze pierwszorzędowej przyjmują zbliżone konformacje przestrzenne,
- krótsze odcinki o znanej sekwencji z długiej nici DNA można łączyć wspólnymi końcami w dłuższą mapę

itd. ...

Odległość edycyjna

Rozważamy słowa nad alfabetem Σ. Dodajemy nowy symbol spacji $\Sigma' = \Sigma \cup \{-\}$. Dla słów $u, w \in \Sigma^*$ ich **liniowym dopasowaniem** nazywamy parę słów $u^*, w^* \in (\Sigma')^*$ spełniających warunki:

- usunięcie spacji z u^* i w^* daje w wyniku odpowiednio u i w,
- $|u^*| = |w^*|$,
- $\forall_{i \le |u^*|}\ u^*[i] \neq '-' \vee w^*[i] \neq '-'$.

Przykład. u=ATAAGC, w= AAAAACG
Dopasowaniem mogą być np.

u^*= –ATAAGC–
w^*= AAAAA–CG

Liniowe dopasowanie przedstawia sposób przekształcenia jednego słowa w drugie przy zastosowaniu elementarnych operacji edytorskich: wstawienie–usunięcie litery (tzw. **indel**) oraz zamiana znaków.

Problem: jak znaleźć „najprostszą” drogę przekształcenia ciągów?

Definiujemy **metrykę** (miarę „odległości” między literami) $d: \Sigma^2 \to R^+$ ($d(a,b)$ dla $a,b \in \Sigma$ – koszt zamiany litery a na b).

Warunki metryki (przypomnienie):

- $d(a,b)=0 \Leftrightarrow a=b$,
- $d(a,b)= d(b,a)$ (symetria),
- $d(a,b)+ d(b,c) \ge d(a,c)$ (warunek trójkąta).

Następnie przedłużamy d na Σ'^2 określając karę za indel:

$$\forall_{a \in \Sigma}\ d(a,'-')= d('-',a)>0$$

Możemy teraz określić **koszt dopasowania** u^*, w^*:

$$\mathrm{d}(u^*,w^*)=\Sigma_{i=1,\ldots,|u^*|}\ d(u^*[i],w^*[i])$$

Przykład. Przyjmijmy funkcje kosztu $d(a,a)=0$ dla $a \in \Sigma$ i 1 w innych przypadkach. Rozważmy dopasowanie
u^*= –ATAAGC–
w^*= AAAAA–CG
mamy $\mathrm{d}(u^*,w^*)=4$.

<u>Odległość edycyjną</u> pomiędzy słowami u,w zdefiniujemy jako koszt ich najtańszego dopasowania

$$đ(u,w)=\min_{(u^*,w^*)\in Dopasowania(u,w)} đ(u^*,w^*)$$

– metryka đ tym razem na Σ^*

Problem: jak znaleźć optymalne (tj. najtańsze) dopasowanie?

Spróbujmy policzyć dopasowania.
$f(i,j)$ – liczba możliwych dopasowań słowa u (i–literowego) z w (j–literowym). Możliwości:

- ostatnie litery u,w są dopasowane,
- ostatnia litera u dopasowana ze spacją,
- ostatnia litera w dopasowana ze spacją.

Stąd rekursja

$$f(i,j)= f(i–1,j–1)+ f(i,j–1)+ f(i–1,j)$$

Ale nawet równanie $g(i,j)=g(i,j–1)+ g(i–1,j)$ ma rozwiązanie $g(i,j)=\binom{i+j}{i}=\frac{(i+j)!}{i!\,j!}$, czyli $g(i,i)= \Theta(4^i i^{–½})$.

Wniosek. $f(i,i)=\Omega(4^i i^{–½})$.
Liczba dopasowań rośnie wykładniczo z długością słów.

Mimo to problem optymalnego dopasowania jest wielomianowy.

Algorytm. Dane są $u,w\in\Sigma^*$, $|u|=n$, $|w|=m$.
Programowanie dynamiczne. Tworzymy tablicę liczb:
$D(i,j)$=koszt najtańszego dopasowania przedrostków $u[1..i]$ $w[1..j]$.

1. Brzeg tabeli:
$D(0,0)=0$, $D(0,j)=\Sigma_{k=1,\ldots,j} d(‘–‘,w[k])$, $D(i,0)=\Sigma_{k=1,\ldots,i} d(u[k], ‘–‘)$

2. Środek tabeli ($i,j>0$):
$D(i,j)=\min \{D(i–1,j–1)+d(u[i],w[j]), D(i,j–1)+ d(‘–‘,w[j]),$
$D(i–1,j)+ d(u[i],‘–‘)\}$.

3. Odległość słów u,w odczytujemy z tabeli $đ(u,w)=D(n,m)$.

Złożoność czasowa i pamięciowa $O(nm)$.

Znana jest implementacja rekurencyjna zużywająca tylko $O(n+m)$ pamięci.

Jak odczytać najtańsze dopasowanie?

- w polach tabeli ustawiamy wskaźniki:
  - na brzegu $D(0,i)\rightarrow D(0,i–1)$ i $D(i,0)\rightarrow D(i–1,0)$ dla $i>0$
  - w środku $D(i,j)\rightarrow$ te z $\{D(i–1,j–1), D(i,j–1), D(i–1,j)\}$, które realizują minimum w punkcie 2.
- odczytujemy dopasowanie „od końca” idąc po wskaźnikach wzdłuż dowolnej drogi z komórki $D(n,m)$ do $D(0,0)$.

Przykład. *u*=’writers’, *w*=’vintnter’, funkcja *d* jak poprzednio.

D(*i*, *j*)			*w*	*r*	*i*	*t*	*e*	*r*	*s*
		0	1	2	3	4	5	6	7
	0	0	← 1	← 2	←3	← 4	← 5	← 6	← 7
v	1	↑ 1	↖ 1	↖ ← 2	↖←3	↖ ← 4	↖ ← 5	↖ ← 6	↖ ← 7
i	2	↑ 2	↖ ↑ 2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
n	3	↑ 3	↖ ↑ 3	↖ ↑ 3	↖↑3	↖ 3	↖ ← 4	↖ ← 5	↖ ← 6
t	4	↑ 4	↖ ↑ 4	↖ ↑ 4	↖↑4	↖ 3	↖ ← 4	↖ ← 5	↖ ← 6
n	5	↑ 5	↖ ↑ 5	↖ ↑ 5	↖↑5	↑ 4	↖ 4	↖ ← 5	↖ ← 6
e	6	↑ 6	↖ ↑ 6	↖ ↑ 6	↖↑6	↑ 5	↖ 4	↖ ← 5	↖ ← 6
r	7	↑ 7	↖ ↑ 7	↖ 6	↖←↑7	↑ 6	↑ 5	↖ 4	← 5

Jedno z optymalnych dopasowań:

```
writ-ers
vintner-
```

„Grafowy” punkt widzenia:

- pary (0,0) ...(*n*,*m*) tworzą wierzchołki digrafu
- krawędzie postaci
  - (*i*,*j*)→ (*i*–1,*j*–1), waga *d*(*u*[*i*],*w*[*j*])
  - (*i*,*j*)→ (*i*,*j*–1), waga *d*(‘–‘,*w*[*j*])
  - (*i*,*j*)→ (*i*–1,*j*), waga *d*(*u*[*i*],‘–‘)
- dopasowanie: dowolna ścieżka z (*n*,*m*) do (0,0).
- najtańsze dopasowanie: najkrótsza taka ścieżka.

Funkcja podobieństwa sekwencji

Słowa można dopasować tym lepiej im mniejsza odległość edycyjna między nimi. Z podobieństwem słów jest odwrotnie: im większe tym lepsze dopasowane jest możliwe.

Jak poprzednio najpierw definiujemy funkcję na pojedynczych literach $s:(\Sigma')^2 \to R$ (gdzie znów $\Sigma'=\Sigma\cup\{-\}$). Z reguły przyjmuje się, że:

- $s(a,b)<s(a,a)>0$ dla $a\in\Sigma$,
- $s(a,b)= s(b,a)$ (symetria),
- $s(a,'-')= s('-',a)<0$.

Definiujemy **<u>podobieństwo dopasowania</u>** u^*,w^* słów $u,w\in\Sigma^*$:

$$\S(u^*,w^*)=\Sigma_{i=1,\ldots,|u^*|}\, s(u^*[i],w^*[i])$$

<u>Optymalnym dopasowaniem</u> słów *u*,*w* jest to, które maksymalizuje podobieństwo

$$\S(u,w)=\max_{(u^*,w^*)\in Dopasowania(u,w)} \S(u^*,w^*)$$

Jak znaleźć optymalne dopasowanie?

Algorytm. (podobny do poprzedniego).
Dane $u,w\in\Sigma^*$, $|u|=n$, $|w|=m$.

Tablica liczb:
S(*i*,*j*)=maksymalne podobieństwo dopasowania przedrostków *u*[1..*i*],*w*[1..*j*].

1. Brzeg tabeli:
$S(0,0)=0$, $S(0,j)=\Sigma_{k=1,\ldots,j}\, s('-',w[k])$, $S(i,0)=\Sigma_{k=1,\ldots,i}\, s(u[k],'-')$

2. Środek tabeli ($i,j>0$):
$S(i,j)=\max\{ S(i-1,j-1)+s(u[i],w[j]),\ S(i,j-1)+ s('-',w[j]),$
$S(i-1,j)+ s(u[i],'-')\}$.

3. Ostatecznie z definicji $\S(u,w)=S(n,m)$.

Złożoności (jak poprzednio) wynoszą $O(nm)$.

Grafowo:

Digraf acykliczny (podobny do poprzedniego):

- pary (0,0) ...(n,m) – wierzchołki
- krawędzie:
  - (i,j)→ (i–1,j–1), waga $s(u[i],w[j])$
  - (i,j)→ (i,j–1), waga $s(\text{'–'},w[j])$
  - (i,j)→ (i–1,j), waga $s(u[i],\text{'–'})$

Optymalne dopasowanie: najdłuższa ścieżka z (n,m) do (0,0).

Algorytm wyznaczania najdłuższej ścieżki w digrafie acyklicznym (przypomnienie):
Obliczamy długości najdłuższych ścieżek prowadzących od z do innych wierzchołków.

1. Ponumeruj wierzchołki kolejnymi liczbami naturalnymi wg porządku topologicznego (istnienie łuku $v_i \to v_j$ implikuje $i<j$).
2. Wierzchołkowi z nadaj etykietę $l(z)=0$.
3. Wierzchołkom v o kolejnych numerach przypisuj etykiety według zasady:
   $l(v)=\max\{l(u)+w(e)$: łuk e prowadzi z u do $v\}$
4. Etykiety określają szukane długości ścieżek.

Przykład.
Problem **Najdłuższego Wspólnego Podciągu (NWP)**.
Dany jest skończony zbiór słów $R\subseteq\Sigma^+$. Szukamy najdłuższego możliwego słowa $w=w_1...w_k\in\Sigma^*$, takiego że

$$\forall_{v\in R}\ v=x_0w_1x_1w_2...w_kx_k,\ \ x_i\in\Sigma^*.$$

NWP jest NP–trudny (pozostaje takim nawet dla alfabetu 2–literowego, redukcja z pokrycia wierzchołkowego), ale staje się wielomianowy dla dwóch słów ($|R|=2$).

W przypadku dwóch słów sprowadzamy NWP do optymalnego dopasowania przyjmując funkcję podobieństwa $s(a,a)=1$ i $s(a,b)=s(a,\text{'–'})=s(\text{'–'},a)=0$ dla $a\neq b\in\Sigma$. Liczba prawidłowo dopasowanych symboli decyduje o wartości podobieństwa.

Odległość edycyjna a podobieństwo

Niech d będzie metryką na alfabecie Σ, za pomocą której określamy odległość edycyjną d w Σ^*. Definiujemy funkcję podobieństwa

- $s(a,b)=c-d(a,b)$ dla $a,b\in\Sigma$
- $s(a,\text{'–'})=s(\text{'–'},a)=c/2-d(a,\text{'–'})$

gdzie c – stała.
Rozważamy słowa $u,w\in\Sigma^*$ ($|u|=n$, $|w|=m$) i pewne ich dopasowanie $u^*,w^*\in(\Sigma')^*$. Zauważmy, że:

$$n+m=2|\{i\le|u^*|:\ \text{'–'}\notin\{u^*[i],w^*[i]\}\}|+$$
$$+|\{i\le|u^*|:\ \text{'–'}\in\{u^*[i],w^*[i]\}\}|$$

Stąd

$$s(u^*,w^*)+d(u^*,w^*)=\Sigma_{i=1,...,|u^*|}\,(s(u^*[i],w^*[i])+d(u^*[i],w^*[i]))=$$
$$=\Sigma_{i=1,...,|u^*|,\text{'–'}\notin\{u^*[i],w^*[i]\}}\,c+\Sigma_{i=1,...,|u^*|,\text{'–'}\in\{u^*[i],w^*[i]\}}\,c/2=c(n+m)/2$$

Równość jest prawdziwa dla każdego dopasowania, zatem minimalizacja odległości edycyjnej odpowiada maksymalizacji podobieństwa.

$$s(u,w)+d(u,w)=c(n+m)/2$$

Przykład. Poprzednio używaliśmy prostej metryki $d(a,a)=0$ dla $a\in\Sigma$ i 1 w innych sytuacjach. Przyjmując stałą $c=1$ i mnożąc wynik przez 2 uzyskujmy równoważną metryce d funkcję podobieństwa:

- $s(a,a)=2$, $s(a,b)=0$ dla $a\neq b\in\Sigma$,
- $s(a,'-')=s('-',a)=-1$.

Funkcja kary za przerwy w dopasowaniu

Lokalne insercje/delecje mają często inne przyczyny, niż punktowe podmiany pojedynczych liter – mogą dotyczyć dłuższych fragmentów sekwencji:

- polimeraza DNA może „poślizgnąć się" na nici gubiąc kilka nukleotydów,
- „nierówna" rekombinacja,
- efekt bardziej rozległych rearanżacji.

Proponuje się, uwzględniać długość odcinków leżących obok siebie indeli przy naliczaniu „kary" obniżającej podobieństwo sekwencji. Oddzielnie uwzględniamy składnik związany z zamienionymi literami, osobno zaś przerwy w dopasowaniu.

Przykład.

```
C-AGCCCTA--C
CCTG---TACCC
```

Dopasowanie zawiera trzy przerwy o długościach 1, 3 i 2.

Niech dana będzie funkcja kary $p:N\to R$ (zwykle malejąca). Dla dopasowania u^*,w^* słów u,w redefiniujemy podobieństwo:

$$ş(u^*,w^*)=\Sigma_{i=1,\ldots,|u^*|,'-'\notin\{u^*[i],w^*[i]\}} s(u^*[i],w^*[i])+$$
$$+\Sigma_{br\in \text{Przerwy}(u^*,w^*)} p(|br|)$$
$$ş(u,w)=\max_{(u^*,w^*)\in Dopasowania(u,w)} ş(u^*,w^*)$$

Problem. Jak znaleźć optymalne dopasowanie i wartość $ş(u,w)$ podobieństwa słów?

Rozważamy tablice:

- $A(i,j)$ = maksymalne dopasowanie przedrostków $u[1..i]$ $w[1..j]$, przy warunku: $w[j]$ połączono z '–'.
- $B(i,j)$ = maksymalne dopasowanie przedrostków $u[1..i]$ $w[1..j]$, przy warunku: $u[i]$ połączono z '–'.
- $C(i,j)$ = maksymalne dopasowanie przedrostków $u[1..i]$ $w[1..j]$, przy warunku: $u[i]$ połączono z $w[j]$.
- $S(i,j)$ = maksymalne dopasowanie $u[1..i]$ z $w[1..j]$.

Algorytm (programowanie dynamiczne). Dane $u,w\in\Sigma^+$, $|u|=n$, $|w|=m$.

1. Wypełniamy tabele (wartości początkowe: $-\infty$)
$S(0,0)=0$, $S(i,0)=S(0,i)=B(i,0)=A(0,i)=p(i)$ dla $i>0$ oraz

$$A(i,j)=\max_{k\in\{0,j-1\}}\{\max\{B(i,k),C(i,k)\}+p(j-k)\}$$
$$B(i,j)=\max_{k\in\{0,i-1\}}\{\max\{A(k,j),C(k,j)\}+p(i-k)\}$$
$$C(i,j)=S(i-1,j-1)+s(u[i],w[j])$$
$$S(i,j)=\max\{A(i,j),B(i,j),C(i,j)\}$$

– dla $i,j>0$

2. Odczytujemy podobieństwo słów $ş(u,w)=S(n,m)$

Złożoność czasowa $O(nm\max\{n,m\})$ i pamięciowa jak poprzednio: $O(nm)$.

Niekiedy rozważa się liniową funkcję kary:

$$p(x)=-\alpha-\beta x$$

gdzie $\alpha+\beta$ – koszt pojedynczego indelu, β – koszt przedłużenia przerwy o jedną pozycję ($\alpha,\beta>0$).

Dla takiego modelu istnieje algorytm oparty na programowaniu dynamicznym o złożoności czasowej i pamięciowej $O(nm)$.