Estudi de mètriques de distància en estructures discretes

Rafael Castillo López
Gabriel Prat Masramon
rafaelc@lsi.upc.es
gprat@lsi.upc.es

1 Introducció

Pretenem fer un estudi sobre les mètriques de distància en estructures discretes, habitualment cadenes de caràcters però extensibles a d'altres estructures discretes, i més concretament ens centrarem en les basades en la distància d'edició i en les seves extensions heurístiques. Començarem parlant de la distància de Levenshtein, que és la primera que empra aquesta idea de la distància d'edició per comparar estructures discretes, i després parlarem de les variants existents que exploten el mateix concepte però otorguen diferent ponderació a les diferents operacions d'edició, anomenades per aquest fet variants ponderades de la distància d'edició (Needleman-Wunch [1] i Smith-Waterman [2]); per acabar amb l'estudi parlarem de mètodes heurístics emprats per tal d'estalviar càlculs (parlarem principalment de dos mètodes : el FASTA [3],[4] i del BLAST [5],[6]) i tancarem l'estudi amb les aplicacions de les mètriques de distància analitzades.

2 Distancia de Levenshtein

És la distància entesa com el nombre més petit d'operacions simples d'edició que calen per tal de transformar una estructura discreta en una altra. Donades dues estructures discretes (per exemple cadenes de caràcters) s y t, entendrem com operacions simples d'edició :
Copiar un caràcter de s a t
Esborrar un caràcter a s (cost 1)
Insertar a un caràcter a t (cost 1)
Substituir un caràcter per un altre (cost 1)

Per tal de calcular-la seguirem els següents passos:

1. Construim una matriu amb files numerades de 0..m i columnes numerades de 0..n.
2. Examinem cada caràcter de s.
3. Examinem cada caràcter de t.
4. Si s[i] és igual a t[j], el cost és 0. Si s[i] no és igual a t[j], el cost és 1.
5. La cel·la d[i,j] de la matriu serà igual al mínim de :
    a. La cel·la inmediatament superior més 1: d[i-1,j] + 1.
    b. La cel·la inmediatament a l'esquerra més 1: d[i,j-1] + 1.
    c. La cel·la a la diagonal superior a l'esquerra més el cost: d[i-1,j-1] + cost.
6. Un cop acabat el procés tenim el resultat a la cel·la d[n,m]

Aquesta seria la versió algorísmica. En una notació més compacta podem definir cada cel·la de la següent manera :

D(i,j) = "puntuació" del **millor** alineament de *s1..si* a *t1..tj*

$$\min \begin{cases} D(i-1,j-1) + d(si,tj) & //subst/copiar \\ D(i-1,j)+1 & //insertar \\ D(i,j-1)+1 & //esborrar \end{cases}$$

on d(si,tj)=1 si si≠sj i 0 si si=sj (després veurem variacions d'aquesta distància que entre d'altres coses tenen en compte que la distància entre els elements de l'estructura discreta pot variar en funció dels elements).

A continuació veurem un exemple del seu funcionament per acabar d'aclarir com es faria el seu càlcul.

Pas 1 a l'esquerra i del 2 al 5 per i=1 a la dreta.

		G	U	M	B	O				G	U	M	B	O
	0	1	2	3	4	5			0	1	2	3	4	5
G	1							G	1	0				
A	2							A	2	1				
M	3							M	3	2				
B	4							B	4	3				
O	5							O	5	4				
L	6							L	6	5				

Podem veure com es va agafant com a valor de la cel·la el mínim de la cel·la immeditament superior més 1, la cel·la immediatament a l'esquerra més 1 i la cel·la diagonal superior esquerra més el cost, essent el cos 0 si són iguals els simbols i 1 si són diferents

Passos del 2 al 5 per i=2 a l'esquerra i per i=3 a la dreta.

		G	U	M	B	O				G	U	M	B	O
	0	1	2	3	4	5			0	1	2	3	4	5
G	1	0	1					G	1	0	1	2		
A	2	1	1					A	2	1	1	2		
M	3	2	2					M	3	2	2	1		
B	4	3	3					B	4	3	3	2		
O	5	4	4					O	5	4	4	3		
L	6	5	5					L	6	5	5	4		

Seguim calculant el valor de les cel·les emprant el mateix criteri d'abans.

Passos del 2 al 5 per i=4 a l'esquerra i per i=5 a la dreta.

		G	U	M	B	O				G	U	M	B	O	
	0	1	2	3	4	5			0	1	2	3	4	5	
G	1	0	1	2	3			G	1	0	1	2	3	4	
A	2	1	1	2	3			A	2	1	1	2	3	4	
M	3	2	2	1	2			M	3	2	2	1	2	3	
B	4	3	3	2	1			B	4	3	3	2	1	2	
O	5	4	4	3	2			O	5	4	4	3	2	1	
L	6	5	5	4	3			L	6	5	5	4	3	**2**	

Un cop acabat el procés tenim el valor de la distància d'edició o distància de Levenshtein a la cel·la inferior dreta de la matriu. La distància calculada és 2, i intuitivament ja podiem veure com per passar de GUMBO a GAMBOL calient dues operacions d'edició : la substitució de la U per la A i la inserció d'una L final.

3 Variants

Com a extensió de la distància de Levenshtein, podem trobar-ne vàries que es calculen de similar forma i on tan sols varien els pesos otorgats a les operacions d'edició. Entre aquestes tenim per exemple la distància de Needleman-Wunch, on com ja avançàvem abans es té en compte la distància entre els elements de les estructures discretes que estem analitzant.

L'algorisme de Needleman-Wunch maximitza un índex de similitud per tal d'assolir el que anomena 'emparellament màxim', entenent com a tal al nombre més gran de residus d'una seqüència que poden ser emparellats amb una altra permetent possibles supressions. D'aquesta forma troba el millor alineament **global** entre dues seqüències (o estructures discretes).

Per tal de fer-ho emprarem també una matriu i seguirem els següents passos :

1. Assignem valors de similitud : com abans ja avançavem aquests valors poden ser tan simples con +1 si son iguals i 0 altrament o bé més elaborats en el cas com afinitats (en el cas que als elements de les seqüències els hi sigui aplicable aquest concepte, com podria ser el cas de que fossin elements químics per exemple), o freqüències.
2. Per cada cel·la, mirem tots els possibles "camins" cap al principi de la seqüència (permetent insercions i supressions) i donem a la cel·la el valor de la màxima puntuació. Per tal de fer-ho :
    a. Per cada cel·la volem saber la màxima puntuació possible per un alineament que s'acabi en aquest punt
    b. Busquem a la subfila i a la subcolumna la puntuació més alta
    c. Afegim aquesta puntuació a la puntuació de la cel·la actual
    d. Seguim amb aquest procés linia a linia per la matriu
    e. Podem emprar el concepte de penalització per la introducció de forats en l'alineament
3. L'emparellament màxim **sempre** estarà a la fila o a la columna exterior de la matriu construida
4. Per construir l'alineament de les estructures discretes que porti a aquest emparellament, es parteix de la cel·la amb el màxim emparellament i es torna enrera per la matriu.

A continuació veurem un exemple del seu funcionament per acabar d'aclarir com es faria el seu càlcul.

	M	P	R	C	L	C	Q	R	J	N	C	B
P		1										
B												1
R			1					1				
C				1		1					1	
K												
C				1		1					1	
R								1				
N									1			
J									1			
C				1		1					1	
J									1			

Primer assignem els valors de similitud entre les parelles d'elements. En aquest cas hem assignat un 1 als iguals i un 0 als diferents

	M	P	R	C	L	C	Q	R	J	N	C	B	A
P	0	1	0	0	0	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1	1	1	2	1	
R	0	0	2	1	1	1	1	2	1	1	1	2	
C	0	0	1	3	2	3	2	2	2	3	2	2	
K	0	0	1	2	3	3	3	3	3	3	3	3	
C	0	0	1	3	3	4	3	3	3	4	3	3	
R	0	0	2	2	3	3	4	?					
N													
J									1				
C				1		1					1		
J									1				
A													1

Després busquem la puntuació màxima (la del millor alineament que acabi a la cel·la) per cada cel·la de la matriu, seguint el procediment descrit abans. La nova cel·la es calcularia doncs com :

Hij=max{Hi-1, j-1 +s(ai,bj), max{Hi-k,j-1 -Wk +s(ai,bj)}, max{Hi-1, j-l -Wl +s(ai,bj)}}

	M	P	R	C	L	C	Q	R	J	N	C	B	A
P	0	1	0	0	0	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1	1	1	1	2	1
R	0	0	2	1	1	1	1	2	1	1	1	1	2
C	0	0	1	3	2	3	2	2	2	2	3	2	2
K	0	0	1	2	3	3	3	3	3	3	3	3	3
C	0	0	1	3	3	4	3	3	3	3	4	3	3
R	0	0	2	2	3	3	4	5	4	4	4	4	4
N	0	0	1	2	3	3	4	4	5	6	5	5	5
J	0	0	1	2	3	3	4	4	6	5	6	6	6
C	0	0	1	3	3	4	4	4	5	6	7	6	6
J	0	0	1	2	3	3	4	4	6	6	6	7	7
A	0	0	1	2	3	3	4	4	5	6	6	7	8

Com veiem, la puntuació màxima es troba a la part exterior de la matriu i per construir l'alineament que porta a aquesta puntuació tan sols cal tornar enrera per la matriu.

```
MP-RCLCQR-JNCBA
  | ||  |  |  |  |  |
-PBRCKC-RNJ-CJA
```

Un altre variant de la distància d'edició és la distància de Smith-Waterman, que compara segments de totes les possibles longituds (ara els alineaments són **locals**, en contraposició a la distància de Needleman-Wunch) i tria aquella que maximitza la mesura de similitud. Calcula **tots** els camins que porten a cada cel·la, podent ésser aquests camins de qualsevol longitud i podent contenir insercions i supressions. Només treballa bé quan s'empren penalitzacions als forats, dins de les quals s'acostuma a considerar per separat el cost d'*obrir* un forat al cost de *continuar-lo*; per tal d'entendre el perquè d'aquest fet posarem un exemple extret d'unes transparències que ens sembla molt ilustratiu :

Inserció simple :

William W. Cohen

William W. Cohen ‘Don’t call me Dubya’

Inserció múltiple :

> Intuitively, a single long insertion is “cheaper” than a lot of short insertions
>
> **Intuitively, a**re **s**p**r**i**ngl**est hu**long**ru poi**nsertim**o**n** ext**is**n’t **“cheaper” than a lot of short insertions**

Sembla bastant clar que les cadenes obtingudes l’una fruit d’una inserció simple a l’altra són molt més semblants que les cadenes obtingudes de la inserció múltiple. Podem entendre doncs que les distàncies que emprin el concepte de cost per obrir un forat i cost per continuar-lo variaran considerablement de rendiment en funció de com ajustem aquests costos.

Per tal de fer-ho emprarem també una matriu i seguirem els següents passos :

1. Comencem assignant un 0 com a valor inicial de les cel·les com en els casos anteriors
2. Mirem a la subcolumna i a la subfila com al Needleman-Wunch i ara també a la diagonal, per una puntuació que sigui la màxima entre la puntuació d’alineament i la penalització per forat
3. Hi ha quatre maneres de formar un camí
    a. Alinear amb el següent residu de la seqüència destí. La puntuació és la anterior més la puntuació de similitud entre els dos residus
    b. Esborrat (Aliniem el resudi de la seqüència d’origen amb un forat) La puntuació és l’anterior menys la penalització del forat (que dependrà de la mida del forat)
    c. Inserció (Aliniem el residu de la seqüència destí amb un forat) La puntuació serà l’anterior menys la penalització del forat (que dependrà de la mida del forat)
    d. Stop. La puntuació es 0
4. La puntuació a cada cel·la ara sera la màxima possible per un alineament de qualsevol mida que acabi justament a les coordenades de la cel·la. La cel·la amb la puntuació més alta pot estar a qualsevol lloc de la matriu
5. Al igual que abans, per tal de reconstruir l’alineament de les estructures discretes que porti a aquest emparellament, es parteix de la cel·la amb el màxim emparellament i es torna enrera per la matriu.

Al igual que ens els casos anteriors, a continuació veurem un exemple del seu funcionament per acabar d’aclarir com es faria el seu càlcul.

		C	A	G	C	C	U	C	G	C	U	U	A	G
		0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
A		0.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.7
U		0.0	0.0	0.8	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.7
G		0.0	0.0	1.0	0.3	0.0	0.0	0.7	1.0	0.0	0.0	0.7	0.7	1.0
C		1.0	0.0	0.0	2.0	1.3	0.3	1.0	0.3	2.0	0.7	0.3	0.3	0.3
C		1.0	0.7	0.0	1.0	3.0	1.7	?						
A														
U														
U														
G														
A														
C														
G														
G														

Ens saltem el primer pas.
El cost per forat de l’exemple es : -1+1/3k, on k es la longitud del forat
El d’error es -1/3
Ara també mirarem alineaments locals. La nova cel·la es calcularia com :

Hij=max{Hi-1, j-1 +s(ai,bj), max{Hi-k,j -Wk}, max{Hi, j-l -Wl}, 0}

		C	A	G	C	C	U	C	G	C	U	U	A	G
		0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
A		0.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.7
U		0.0	0.0	0.8	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.7
G		0.0	0.0	1.0	0.3	0.0	0.0	0.7	1.0	0.0	0.0	0.7	0.7	1.0
C		1.0	0.0	0.0	2.0	1.3	0.3	1.0	0.3	2.0	0.7	0.3	0.3	0.3
C		1.0	0.7	0.0	1.0	3.0	1.7	1.3	1.0	1.3	1.7	0.3	0.0	0.0
A		0.0	2.0	0.7	0.3	1.7	2.7	1.3	1.0	0.7	1.0	1.3	1.3	0.0
U		0.0	0.7	1.7	0.3	1.3	2.7	2.3	1.0	0.7	1.7	2.0	1.0	1.0
U		0.0	0.3	0.3	1.3	1.0	2.3	2.3	2.0	0.7	1.7	2.7	1.7	1.0
G		0.0	0.0	1.3	0.0	1.0	1.0	2.0	3.3	2.0	1.7	1.3	2.3	2.7
A		0.0	1.0	0.0	1.0	0.3	0.7	0.7	2.0	3.0	1.7	1.3	2.3	2.0
C		1.0	0.0	0.7	1.0	2.0	0.7	1.7	1.7	3.0	2.7	1.3	1.0	2.0
G		0.0	0.7	1.0	0.3	0.7	1.7	0.3	2.7	1.7	2.7	2.3	1.0	2.0
G		0.0	0.0	1.7	0.7	0.3	0.3	1.3	1.3	2.3	1.3	2.3	2.0	2.0

Ara la màxima puntuació no es troba a la part exterior de la matriu, però veiem que el procediment que ens porta a l’alineament es el mateix :

```
GCC-UCG
GCCAUUG
```

Veurem ara una comparació entre les anomenades variants ponderades de la distància d’edició o distància de Levenshtein :

Needleman-Wunsch	**Smith-Waterman**
Alineaments globals	Alineaments locals
La puntuació d’alinea-ment dels residus ha de ser sempre positiva	La puntuació d’alinea-ment dels residus pot ser positiva o negativa
No requereix l’us de penalitzacions per forat	Requereix l’us de pena-litzacions per forat per tal de funcionar bé
La puntuació no pot decréixer entre dos cel·les d’una camí	La puntuació entre dos cel·les d’una camí pot créixer, decréixer o mantenir-se igual

4 Variants heurístiques

Motivació

Emprant algorismes de programació dinàmica com el de Needleman-Wunsch o bé el de Smith-Waterman queda garantit trobar l’alineament òptim : tots els possibles alineaments són considerats i aquell amb la puntuació més alta es selecciona. Desafortuna-dament però, aquests algorismes son usualment computacionalment complexos (en velocitat i ús de memòria). Les cerques heuristiques són menys sensitives, però bastant bones tot i així i molt més ràpides. La idea es basa en el fet que la majoria d’alineaments bons tenen petits fragments de seqüències sense forats amb una puntuació d’alineament molt alta.

Introducció

Desafortunadament, ben poc es sabut sobre la distribució aleatòria de les puntuacions dels alineaments globals. En canvi, els paràmetres

estadístics de les puntuacions dels alineaments locals es ampliament conegut, especialment sense considerar l'existència de forats. Un alineament local sense forats consisteix simplement en un alineament entre un parell de segments d'igual longitud, un de cadascuna de les seqüències que volem comparar. Una modificació del Smith-Waterman trobarà totes les parelles de segments amb una puntuació que no pot ser millorada per l'extensió o el retallat. Aquestes parelles amb màxima puntuació s'anomenen HSPs (high-scoring segment pairs).

Si tenim seqüències suficientment llargues (de longituds m i n), les propietats estadístiques de les puntuacions dels HSP estan caracteritzats per dos parametres, k i lambda. El nombre esperat de HSPs amb una puntuació almenys de S en el cas suposat vindria donat per la fórmula :

$$E = Kmn\, e^{-\lambda S}$$

Anomenem a aquest coeficient el *E-value* per la puntuació S. Intuitivament, doblant la longitud d'alguna de les dues seqüències doblem també el nombre de HSPs que obtenen una certa puntuació. També podem observar com E baixa de forma exponencial conforme incrementem S. El nombre de parelles aleatòries amb una puntuació major o igual a S és descrit per una distribució de Poisson, i la probabilitat de trobar exactament *a* HSPs amb una puntuació major o igual a S ve donada per :

$$e^{-E} \cdot (E^a / a!)$$

On E es l'anomenat anteriorment *E-value*. Per tant, la probabilitat de no trobar cap HSP amb una puntuació major o igual a S és e^{-E} , i la probabilitat de trobar-ne almenys un :

$$P = 1 - e^{-E}$$

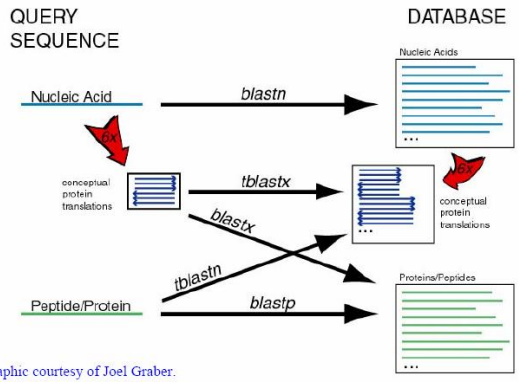
Anomenem a aquest coeficient el *P-value* per la puntuació S. Quan el que pretenem es trobar alineaments d'una seqüència contra les seqüències d'una base de dades, no tenim dues longituds, m i n, sino tan sols una. Com que l' *E-value* és una mesura de significància estadística, modificarem les fórmules anteriors en funció de si ens interessa considerar per igual els alineaments amb seqüències curtes o llargues o bé donar una importància proporcional a la longitud.

BLAST: Basic Local Alignment Search Tool

Aplicacions de l'algorisme :

- Cerca a la base de dades de seqüències similars a la seqüència d'entrada
    - Identificar seqüències prèviament caracteritzades
    - Trobar relacions filogenètiques entre seqüències
    - Identificar possibles funcions basades en similituds entre seqüències

Tipus en funció de la naturalesa de la seqüència d'entrada i de les seqüències de la base de dades :



Graphic courtesy of Joel Graber.

Funcionament de l'algorisme :

1. Cerca a la base de dades paraules amb una mida predeterminada, considerant encert també a paraules d'aquesta mida amb un cert nombre de caràcters diferents (controlat per un paràmetre de *treshold T*)
2. Exten l'encert fins que la puntuació cau per sota de la màxima puntuació assolida, permetent un cert marge de violació *X*.

Com podem veure l'èxit/fràcas de l'algorisme anirà molt lligat a l'assignació de la puntuació, que obviament ha de cumplir :

- L'esperança de la puntuació de dos residus aleatoris s'espera negativa
- Un alineament perfecte ha de tenir puntuació màxima
- Combinacions de residus que es poden intercambiar a les proteines han de tenir puntuacions positives

Per tal d'assignar aquestes puntuacions de forma que ens permetin cumplir les restriccions imposades per la puntuació, emprarem les matrius de substitució, calculades segons :

- Observacions empíriques de les freqüències de substitució
- Puntuacions altes per substitucions amb residus similars
- Substitucions aleatòries han de donar puntuacions negatives

Tipus de matrius de substitució :

- Matrius PAM (Percent Accepted Mutation)
    - Comencen amb seqüències properes i extrapolen les

probabilitats de substitució per seqüències més distants

- 1 Unitat PAM equival a una mutació cada 100 bases
- Exemple : PAM120 calculada considerant 120 mutacions cada 100 bases

PAM120

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W
A	3	-3	-1	0	-3	-1	0	1	-3	-1	-3	-2	-2	-4	1	1	1	-7
R	-3	6	-1	-3	-4	1	-3	-4	1	-2	-4	2	-1	-5	-1	-1	-2	1
N	-1	-1	4	2	-5	0	1	0	2	-2	-4	1	-3	-4	-2	1	0	-4
D	0	-3	2	5	-7	1	3	0	0	-3	-5	-1	-4	-7	-3	0	-1	-8
C	-3	-4	-5	-7	9	-7	-7	-4	-4	-3	-7	-7	-6	-6	-4	0	-3	-8
Q	-1	1	0	1	-7	6	2	-3	3	-3	-2	0	-1	-6	0	-2	-2	-6
E	0	-3	1	3	-7	2	5	-1	-1	-3	-4	-1	-3	-7	-2	-1	-2	-8
G	1	-4	0	0	-4	-3	-1	5	-4	-4	-5	-3	-4	-5	-2	1	-1	-8
H	-3	1	2	0	-4	3	-1	-4	7	-4	-3	-2	-4	-3	-1	-2	-3	-3
I	-1	-2	-2	-3	-3	-3	-3	-4	-4	6	1	-3	1	0	-3	-2	0	-6
L	-3	-4	-4	-5	-7	-2	-4	-5	-3	1	5	-4	3	0	-3	-4	-3	-3
K	-2	2	1	-1	-7	0	-1	-3	-2	-3	-4	5	0	-7	-2	-1	-1	-5
M	-2	-1	-3	-4	-6	-1	-3	-4	-4	1	3	0	8	-1	-3	-2	-1	-6
F	-4	-5	-4	-7	-6	-6	-7	-5	-3	0	0	-7	-1	8	-5	-3	-4	-1
P	1	-1	-2	-3	-4	0	-2	-2	-1	-3	-3	-2	-3	-5	6	1	-1	-7
S	1	-1	1	0	0	-2	-1	1	-2	-2	-4	-1	-2	-3	1	3	2	-2
T	1	-2	0	-1	-3	-2	-2	-1	-3	0	-3	-1	-1	-4	-1	2	4	-6
W	-7	1	-4	-8	-8	-6	-8	-8	-3	-6	-3	-5	-6	-1	-7	-2	-6	12
Y	-4	-5	-2	-5	-1	-5	-5	-6	-1	-2	-2	-5	-4	4	-6	-3	-3	-2
V	0	-3	-3	-3	-3	-3	-3	-2	-3	3	1	-4	1	-3	-2	-2	0	-8

Aquestes serien les característiques bàsiques d'aquests tipus de matrius. Més concretament, la matriu PAM1 va ser construida partint de 71 famílies de proteines. Les seqüències d'almenys 85% de coincidència van ésser alineades. Aquest requirement fou degut a

- El resultat de l'alineament no era ambigu
- La probabilitat de tenir dues substitucions a la mateixa posició és petita

La matriu PAM1 es basa en les probabilitats Pr(b|a) d'una substitució de l'aminoàcid *a* per l'aminoàcid *b* a les seqüències proteiques a una certa distància d'evolució amb una mitja de substitució del 1% del nombre de posicions. És a dir, exposant una proteina a un canvi evolucionari en aquest interval de temps resulta en una mitja de substitucions d'una per cada 100 aminoàcids. Per extrapolar cap a temps d'evolució majors podem asumir que les substitucions són com una cadena de Markov (time-reversible), per tant si tenim la matriu PAM1 amb les entrades Pr(b|a,t=1) la matriu PAM2 serà construida de la següent manera :

$$PR(b \mid a, t=2) = \sum_c \Pr(c \mid a, t=1) \Pr(b \mid c, t=1)$$

és a dir, les substitucions de *a* cap a *b* via un aminoàcid arbitrari *c* (sumant tots els possibles aminoàcids). La matriu PAMx es construeix per iteració :

$$\Pr(b \mid a, t=x) = \sum_c \Pr(c \mid a, t=x-1) \Pr(b \mid c, t=1).$$

La matriu de puntuacions s'obté :

$$s(a,b \mid t) = \ln \frac{\Pr(b \mid a,t)}{q_b}.$$

Un gran inconvenient de les matrius PAM es que mentre l'error d'estimació és petit a la matriu PAM1 aquest creix amb la distància, esdevenint massa gran a la matriu PAM250. Per tractar de posar remei a aquest problema van sorgir les matrius de substitució BLOSUM.

- Matrius BLOSUM (Block Substitution Matrices)
  - Valors inferits de seqüències amb un cert percentatge de coincidència
  - Exemple : BLOSUM80 derivada de seqüències amb no més d'un 80% de coincidència

BLOSUM80

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	7	-3	-3	-3	-1	-2	-2	0	-3	-3	-3	-1	-2	-4	-1	2	0	-5	-4	-1
R	-3	9	-1	-3	-6	1	-1	-4	0	-5	-4	3	-3	-5	-3	-2	-2	-5	-4	-4
N	-3	-1	9	2	-5	0	-1	-1	1	-6	-6	0	-4	-6	-4	1	0	-7	-4	-5
D	-3	-3	2	10	-7	-1	2	-3	-2	-7	-7	-2	-6	-6	-3	-1	-2	-8	-6	-6
C	-1	-6	-5	-7	13	-5	-7	-6	-7	-2	-3	-6	-3	-4	-6	-2	-2	-5	-5	-2
Q	-2	1	0	-1	-5	9	3	4	1	-5	-4	2	-1	-5	-3	-1	-1	-4	-3	-4
E	-2	-1	-1	2	-7	3	8	-4	0	-6	-6	1	-4	-6	-2	-1	-2	-6	-5	-4
G	0	-4	-1	-3	-6	-4	-4	9	-4	-7	-7	-3	-5	-6	-5	-1	-3	-6	-6	-6
H	-3	0	1	-2	-7	1	0	-4	12	-6	-5	-1	-4	-2	-4	-2	-3	-4	3	-5
I	-3	-5	-6	-7	-2	-5	-6	-7	-6	7	2	-5	2	-1	-5	-4	-2	-5	-3	4
L	-3	-4	-6	-7	-3	-4	-6	-7	-5	2	6	-4	3	0	-5	-4	-3	-4	-2	1
K	-1	3	0	-2	-6	2	1	-3	-1	-5	-4	8	-3	-5	-2	-1	-1	-6	-4	-4
M	-2	-3	-4	-6	-3	-1	-4	-5	-4	2	3	-3	9	0	-4	-3	-1	-3	-3	1
F	-4	-5	-6	-6	-4	-5	-6	-6	-2	-1	0	-5	0	10	-6	-4	-4	0	4	-2
P	-1	-3	-4	-3	-6	-3	-2	-5	-4	-5	-5	-2	-4	-6	12	-2	-3	-7	-6	-4
S	2	-2	1	-1	-2	-1	-1	-1	-2	-4	-4	-1	-3	-4	-2	7	2	-6	-3	-3
T	0	-2	0	-2	-2	-1	-2	-3	-3	-2	-3	-1	-1	-4	-3	2	8	-5	-3	0
W	-5	-5	-7	-8	-5	-4	-6	-6	-4	-5	-4	-6	-3	0	-7	-6	-5	16	3	-5
Y	-4	-4	-4	-6	-5	-3	-5	-6	3	-3	-2	-4	-3	4	-6	-3	-3	3	11	-3
V	-1	-4	-5	-6	-2	-4	-4	-6	-5	4	1	-4	1	-2	-4	-3	0	-5	-3	7

Equivalències entre els tipus de matrius (per similituds entre distàncies que es calculen en base a elles) :

PAM120	BLOSUM80
PAM160	BLOSUM62
PAM250	BLOSUM45

Les matrius BLOSUM són més tolerants a les substitucions hidrofòbiques però en canvi menys tolerants a les hidrofíliques que les matrius PAM.

La suma de les puntuacions de les matrius de substitució ens dona les puntuacions HSPs nominals, que les podem normalitzar per tenir les puntuacions bit a bit (per tal de poder comparar puntuacions d'alineament amb altres mètodes) amb la següent fórmula :

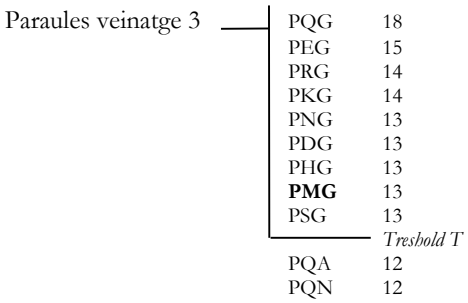
$$S' = (\lambda S - \ln K) / \ln 2$$

on K i λ són paràmetres estadístics que relacionen la puntuació calculada amb la probabilitat de trobar un encert amb almenys aquesta puntuació

A més de la puntuació nominal i la puntuació bit a bit les eines que apliquen aquest algorisme ens

acostumen a donar els coeficients vistos a la introducció dels mètodes heurístics. Novament ho veurem més clarament amb un exemple :

Query : SLAALLNKCKT**PQG**QRLVNQWIKQPL



Extensió de la paraula pels dos extrems :

```
SLAALLNKCKTPQGQRLVNQWIKQPL
+LA++L+    TP G R++ +W+   P+
TLASVLDCTVTPMGSRMLKRWLHMPR
```

El símbol + denota les puntuacions positives a la matriu de substitució.

En definitiva, cerquem paraules de longitud W (per defecte 3 a la variant blastp) que tinguin una puntuació almenys de *T* al alinear-les amb la paraula de consulta i puntuar-les amb la matriu de substitució. Les paraules a la base de dades que cumpleixin aquests requisits són exteses en les dues direccions per tal de trobar un alineament sense forats localment òptim o el que és el mateix un HSP (high scoring pair) amb una puntuació almenys de S o un *E-value* menor a l'especificat per un cert llindar. Els HSPs que cumpleixin aquest criteri seran retornats pel BLAST.

FASTA: Una aproximació eficient a Smith-Waterman

FASTA consta bàsicament de dos passos :

- El primer pas es basa una cerca de segments similars a les dues seqüencies. En aquesta cerca es fixa la mida de la paraula i es cerquen coincidències en una taula bidimensional similar a la que hem vist a l'algorisme Smith-Waterman.
- El segon pas correspon a aplicar l'algorisme d'alineament Smith-Waterman vist abans centrat en aquelles diagonals que corresponen a l'alineament de segments molt similars de les dues seqüències. La regió sobre la qual aplicar l'alineament de Smith-Waterman està acotada per una mida de finestra, que limita el nombre d'insercions o supressions que una seqüència pot acumular en respecte de l'altre a l'alineament. És a dir, estem guanyant eficiència en respecte al Smith-Waterman original gràcies a les restriccions a priori en l'espai d'alineaments així com en el fet que tan sols emprarem aquest algorisme quan trobem diagonals que corresponguin a alineaments entre segments de les seqüència amb un alt grau de similitud.

L'algorisme FASTA és doncs una aproximació heurística de l'algorisme de Smith-Waterman. Les heuristiques usades per FASTA el permeten executar-se molt més ràpidament que l'algorisme original, però com succeeix habitualment amb els mètodes heurístics a costa de perdre alguna sensibilitat. Concretament empra dos heurístiques, ambdues poden ser interpretades com restriccions del model d'evolució de seqüències emprat en la comparació. La primera és implementada amb el paràmetre de la mida de la paraula i la segona amb el paràmetre de la mida de la finestra, essent el segon molt més determinant que el primer fins al punt que una mala tria pot fer-nos concloure que dues seqüències no són homologues quan un anàlisi complet amb Smith-Waterman hagués determinat que si ho eren.

Per cerques de nucleòtids l'algorisme FASTA és més sensible que el BLAST.

5 Aplicacions

Ja hem vist a les aproximacions heurístiques com un dels usos més importants d'aquests algorismes es troben dins del camp del que podem anomenar seqüències biològiques (cadenes d'ADN i de proteines), si bé els algorismes de programació dinàmica vistos són emprats en la detecció de plagis, en eines de control de versions per fer el merge i dur un control dels canvis fets sobre els arxius i en general com a mètriques, que podrem després fer servir en molts algorismes clàssics dins de l'anàlisi de dades i data mining.

7 Referències

[1] Needleman S.B. and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins *J Mol Biol*, 48(3):443-53. 1970

[2] Smith T.F. and M.S.Waterman. Identification of common molecular subsequences *J Mol Biol*, 147(1):195-7. 1981

[3] Pearson W.R. and D.J. Lipman. Improved tools for biologial sequence comparison. *Proc Natl Acad Sci USA*, 85(8):2444-8, 1988

[4] Pearson W.R. Effective protein sequence comparison. *Methods Enzymol*, 266:227-58, 1996

[5] S.F. Altschul, W. Gish, et al. Basic local alignment search tool *J Mol Biol*, 215(3):403-10. 1990

[6] Altschul S.F., T.L. Madden, et al. Gapped BLAST and PSI-BLAST : a new generation of protein database search programs *Nucleic Acids Res*, 25(17):3389-402, 1997