Vortrag 2: *Proteinsequenzen und Substitutionsmatrizen*

Was sind Proteinsequenzen?

Die DNA-Forschung hat sich auf spezielle Abschnitte auf den Strängen der DNA-Moleküle konzentriert, den sog. Protein-codierenden Genen. Diese befinden sich auf beiden Einzelsträngen und werden dafür benutzt, um Proteine zu produzieren, lineare Polymere (lange Kettenmoleküle) aus 20 verschiedenen Aminosäuren, die durch Peptidbindungen miteinander verknüpft sind.

=> Proteinsequenzen (auch Aminosäuresequenzen genannt) sind demnach Verbindungen aus den Bausteinen des kanonischen Aminosäure-Alphabets {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}

Ein-Buschstaben-Code der 20 kanonischen Aminosäuren:

A	Alanin
R	Arginin
N	Asparagin
D	Asparaginsäure
C	Cystein
Q	Glutamin
E	Glutaminsäure
G	Glycin
H	Histidin
I	Isoleucin
L	Leucin
K	Lysin
M	Methionin
F	Phenylalanin
P	Prolin
S	Serin
T	Threonin
W	Tryptophan
Y	Tyrosin
V	Valin

Viele Verfahren, die ich in meinem Vortrag vorstellen werde, könnte man auch auf DNA- und RNA-Sequenzen anwenden. Ausserdem sind diese so allgemeingültig, dass sie auch für Sequenzen von Bausteinen jedes endlichen Buchstaben-Alphabets funktionieren. Zur Veranschaulichung benutzt man meistens ein 2- oder 3-Buchstaben-Alphabet {A, B} bzw. {A, B, C}.

Biologische Sequenzen, deren Strukturen und Funktionen schon untersucht worden sind, sind in Datenbanken erfasst, von denen viele öffentlich und für jeden einsehbar sind.

Was mache ich also, wenn ich ein Protein habe, über die ich noch keine Informationen bzgl. ihrer Struktur und Funktion habe?

Die Tatsache, dass neue DNA-, RNA- und Proteinsequenzen eher aus bereits existierenden Sequenzen entstehen, als dass sie von der Natur von Grund aus neu erfunden werden, legt uns den Ansatz nahe, die zu untersuchende Sequenz mit solchen zu vergleichen, deren Strukturen und Funktionen schon bekannt sind. Dieser Vergleich könnte dann zB. über Datenbanken erfolgen. Man versucht also, Aussagen über die funktionelle oder evolutionäre Verwandtschaft zweier oder mehrerer Sequenzen zu gewinnen.

Wie geht man bei diesem Vergleich vor?

Zunächst nimmt an, man hätte zwei verwandte Sequenzen, die während der Evolution aus einer gemeinsamen Stammsequenz entstanden seien, und nennt diese homolog. Wir müssen versuchen, diese Verwandtschaft mithilfe gegenwärtig verfügbarer Methoden zu zeigen.

Man schaut sich also die Sequenz-Ähnlichkeit an. Falls zwei Sequenzen sehr lang sind, ist es nicht einfach zu entscheiden, ob sie ähnlich sind oder nicht. Um zu sehen, ob sie ähnlich sind, müssen sie richtig aligniert (zugeordnet) werden, dh. es müssen Alignments gebildet werden. Dabei ist es möglich, nicht nur zwei Sequenzen zu alignieren, sondern auch mehrere (multiples Alignment). Wenn sich Sequenzen aus einem gemeinsamen Vorfahr entwickeln, können ihre Reste (= die einzelnen Aminosäuren) miteinander substituiert werden, dh. Aminosäuren können durch andere Aminosäuren ausgetauscht und ersetzt werden. Außerdem können noch weitere Ereignisse stattfinden, wie Insertionen (=Einfügungen) und Deletionen (=Verluste). Wenn man also versucht, das bestmögliche Alignment herzuleiten, müssen die Aminosäuren in der Lage sein, nicht nur mit anderen Aminosäuren aligniert zu werden, sondern auch mit Gaps (=Lücke/Leerstelle), die entweder bei einer Insertion oder einer Deletion entstehen.

Ein einfaches Beispiel dazu:

Seien x: TACCAGT und y: CCCGTAA zwei Sequenzen. Diese müssen jetzt so übereinander gelegt und gegeneinander ausgerichtet werden, dass eine möglichst gute Übereinstimmung erzielt wird:

x: TACCAGT
y: CCCGTAA

Falls nun Gaps im Alignment nicht erlaubt sin, gibt es nur eine Möglichkeit, die Sequenzen zu alignieren, da sie beide die gleiche Länge haben:

x: TACCAGT
y: CCCGTAA

Wie man sieht, gibt es nur eine einzige Übereinstimmung (=Match) (C-C). Der Rest ist unterschiedlich (=Mismatch).

Wenn Gaps doch erlaubt wären, gäbe es viele mögliche Alignments. Insbesondere scheint das folgende Alignment viel informativer zu sein, als das vorherige:

x: TACCAGT - -
y: C- CC - GTAA

Hier sieht man, dass es mehrere Übereinstimmungen gibt. Demnach könnte die Subsequenz (=Teilsequenz/Sequenzabschnitt) CCGT eine evolutionär konservierte Region sein, was bedeutet, dass x und y aus der gleichen Stammsequenz stammen könnten, in der dieser Abschnitt CCGT an einer angemessenen Stelle enthalten gewesen war. Ein weiteres Alignment ist:

x: TACCAGT - -
y: - - CCCGTAA

Welches ist also das bessere Alignment? Oder gäbe es sogar andere Alignments, die noch besser wären? Um diese Fragen beantworten zu können, muss man jedes mögliche Alignment bewerten (=Score zuordnen). Demnach sind diejenigen mit dem höchsten Score die besten bzw. die optimalsten Alignments (es könnte aber auch sein, dass es mehr als nur ein optimales Alignments gibt).

Das bekannteste Scoring-Modell nimmt an, dass die Spalten in einem Alignment unabhängig seien und setzt somit den Gesamt-Score für ein Alignment gleich der Summe der Scores für jede Spalte. Dafür muss man für jeden Match, Mismatch und Gap einen jeweiligen Score s(a,a), s(a,b)=s(b,a) und s(-,a)=s(a,-) (=Gap-Penalty; Kosten für eine Lücke) spezifizieren, mit a,b ϵ $\mathcal{A}$, wobei $\mathcal{A}$ ein Alphabet mit den 20 verschiedenen Aminosäuren ist (oder im Falle einer DNA- oder RNA-Sequenz ein 4-Buchstaben Alphabet).Die daraus resultierenden besten Alignments zwischen den Alignments hängen demnach von dem Scoring-Schema ab. Das bedeutet, dass für zwei verschiedene Scoring-Schema es ganz unterschiedliche Alignments geben könnte.

Wenn man nun als Bsp. die Scores s(a,a)=1 für Matches, s(a,b)=-1 mit a≠b für Mismatches und s(-,a)=s(a,-)=-2 hätte, hätte unser erstes Alignment

x: TACCAGT
y: CCCGTAA

den Score -1-1+1-1-1-1-1=-5 und die anderen beiden

x: TACCAGT - -
y: C- CC - GTAA

x: TACCAGT - -
y: - - CCCGTAA

auch -1-2+1+1-2+1+1-2-2=-5. Bei diesem Beispiel handelt es sich um ein lineares Modell (dh. die Gaps werden nach ihrer Länge bewertet).

Diese Werte für s(a,b)stellen am Ende eine Scoring- bzw. Substitutions-Matrix dar (die ich im 2. Teil meines Vortrages vorstellen werde). Substitutions-Matrizen sind immer symmetrisch und müssen einige Eigenschaften haben, um erfolgreich für einen Sequenzvergleich anwendbar zu sein (2. Teil d. Vortrages). Die bekanntesten Substitutions-Matrizen sind die PAM- und BLOSUM-Matrizen. Sie werden benutzt, um Aminosäure-Sequenzen zu alignieren. Ich werde im zweiten Teil meines Vortrages darauf eingehen. Bis dahin nehmen wir immer an, dass diese schon gegeben seien.

Welche Verfahren gibt es, um ein optimales Alignment zu bestimmen?

Eins, was für die ganzen Sequenzen das beste Alignment findet, ist das globale Alignment.

Nehmen an, man hätte ein lineares Gap-Modell (das ist: $s(-,a)=s(a,-)=d$ für $a \in \mathcal{A}$, mit $d>0$, so dass der Score für eine Gap-Region der Länge L gleich -dL ist) und versuche nun, alle optimalen globalen Alignments zu finden („Needleman-Wunsch-Algorithmus“). Die Idee ist es, ein optimales Sequenz-Alignment aus optimalen Subsequenz-Alignments zu erzeugen. Algorithmen, die eine Optimierung erreichen, indem sie eine Optimierung kleinerer Datenmengen durchführen (in diesem Fall Subsequenzen), werden dynamische Programmierungs-Algorithmen genannt.

Sagen wir, es seien zwei Sequenzen $x=x_1 \dots x_i\ x_{i+1} \dots x_n$ und $y=y_1 \dots y_j\ y_{j+1} \dots y_m$ gegeben. Dann konstruiert man eine (n+1)x(m+1)-Matrix F, in er das (i,j)-ste Element F(i,j) für i=1,…,n und j=1,…,m gleich dem Score eines optimalen Alignments zwischen $x_1 \dots x_i$ und $y_1 \dots y_j$ ist. F(i,0) für i=1,…,n ist der Score des Alignings von $x_1 \dots x_i$ zu einer Gap-Region der Lange i. Gleicherweise ist F(0,j) der Score für das Aligning von $y_1 \dots y_j$ für j=1,…,m zu einer Gap-Region der Länge j. F wird so konstruiert, dass die Matrix sich selbst initialisiert im Zustand F(0,0)=0, und wird so ausgefüllt, dass man in der oberen linken Ecke anfängt und bis zur unteren rechten Ecke alles einträgt. Damit hat man die Werte F(i-1,j-1), F(i-1,j) und F(i,j-1) für alles i und j, so dass man F(i,j) wie folgt berechnen kann:

$$F(i,j)=\max\begin{cases} F(i-1,j-1) + s(x_i,y_j) & x_i \text{ aligniert zu } y_j \\ F(i-1,j) - d & x_i \text{ aligniert zu einem Gap} \\ F(i,j-1) - d & y_j \text{ aligniert zu einem Gap} \end{cases}$$

Es gibt also drei Möglichkeiten, den besten Score F(i,j) zu erhalten. Durch die Berechnung von F(i,j) erhält man einen Pfeil zu der Option, aus dem F(i,j) berechnet wurde. Nachdem man F(n,m) erreicht hat, kann man diese Pfeile zurück verfolgen, so dass man die optimalen Alignments erhält (=Traceback) (es könnte sein, dass bei der Berechnung von F(i,j) für mehr als nur eine Option den selben Score erhält, und dementsprechend dann mehrere Alignments hat). Der Eintrag F(n,m) ist dann genau ihr Score.

Ein Beispiel dazu:

Seien x=CTTAGA und y=GTAA zwei Sequenzen, und nehme an, man hätte das Scoring-Schema: s(a,a)=1, s(a,b)=-1 für a≠b und s(-,a)=s(a,-)=-2. Die entsprechende Matrix F mit den Pfeilen ist gegeben durch:

F		0 -	1 G	2 T	3 A	4 A
0	-	0	-2	-4	-6	-8
1	C	-2	-1	-3	-5	-7
2	T	-4	-3	0	-2	-4
3	T	-6	-5	-2	-1	-3
4	A	-8	-7	-4	-1	0
5	G	-10	-7	-6	-3	-2
6	A	-12	-9	-8	-5	-2

wobei die Werte so berechnet werden:

F(0,0)=0 ,

F(0,1)=F(1,0)=-2

F(0,2)=F(2,0)=-4

F(0,3)=F(3,0)=-6

F(0,4)=F(4,0)=-8

$$F(1,1)=\max\begin{cases} F(0,0)+s(C,G) = 0-1 = -1 \\ F(0,1)-2 = -2-2 = -4 \\ F(1,0)-2 = -2-2 = -4 \end{cases}$$

$$F(1,2)=\max\begin{cases} F(0,1)+s(C,T) = -2-1 = -3 \\ F(0,2)-2 = -4-2 = -6 \\ F(1,1)-2 = -1-2 = -3 \end{cases}$$

$$F(1,3)=\max\begin{cases} F(0,2)+s(C,A) = -4-1 = -5 \\ F(0,3)-2 = -6-2 = -8 \\ F(1,2)-2 = -3-2 = -5 \end{cases}$$

$$F(1,4)=\max\begin{cases} F(0,3)+s(C,A) = -3-1 = -7 \\ F(0,4)-2 = -8-2 = -10 \\ F(1,3)-2 = -5-2 = -7 \end{cases}$$

$$F(2,1)=\max\begin{cases} F(1,0)+s(T,G) = -2-1 = -3 \\ F(1,1)-2 = -1-2 = -3 \\ F(2,0)-2 = -4-2 = -6 \end{cases}$$

$$F(2,2)=\max\begin{cases} F(1,1)+s(T,T) = -1+1 = 0 \\ F(1,2)-2 = -3-2 = -5 \\ F(2,1)-2 = -3-2 = -5 \end{cases}$$

$$F(2,3)=\max\begin{cases} F(1,2)+s(T,A) = -3-1 = -4 \\ F(1,3)-2 = -5-2 = -7 \\ F(2,2)-2 = 0-2 = -2 \end{cases}$$

$$F(2,4)=\max\begin{cases} F(1,3)+s(T,A) = -5-1 = -6 \\ F(1,4)-2 = -7-2 = -9 \\ F(2,3)-2 = -2-2 = -4 \end{cases}$$

usw…..

So ergeben sich drei optimale Alignments, die allen den Score -2 haben:

x: CTTAGA	x: CTTAGA	x: CTTAGA
y: G-TA - A	y: GT-A - A	y: - GTA - A

Ihre Tracebacks sind farblich abgesetzt.

Was macht man aber, wenn man sich nur für Sequenz-Abschnitte mit dem höchsten Score interessiert?

Dann gibt es die Möglichkeit, mithilfe der dynamischen Programmierung das optimale lokale Alignment zu finden. Dieses Verfahren geht so vor, dass es für Subsequenzen fortlaufender Elemente $x_i\, x_{i+1} \ldots x_{i+k}$, $1 \le i \le n$, $k \le n-i$, aus einer gegebenen Sequenz die Sequenz-Abschnitte aus $y=y_1 \ldots y_m$, zuordnet, so dass diese Alignments den höchsten Score haben. Der „Smith-Waterman-Algorithmus“ löst dieses Problem für ein lineares Gap-Modell. Man konstruiert dabei wieder eine (n+1)x(m+1)-Matrix F, wie im oberen Abschnitt auch, nur dass diesmal die Gleichung für die Scores etwas anders aussieht:

$$F(i,j)=\max \begin{cases} 0 & \text{Anfang eines neuen Alignm.} \\ F(i-1,j-1) + s(x_i,y_j) & x_i \text{ aligniert zu } y_j \\ F(i-1,j) - d & x_i \text{ aligniert zu einem Gap} \\ F(i,j-1) - d & y_j \text{ aligniert zu einem Gap} \end{cases}$$

Die erste Option in der Formel entspricht einem Anfang eines neuen Alignments: wenn das optimale Alignment zu einem Punkt hoch einen negativen Score hat, dann ist es besser, ein neues Alignment zu starten, als das laufende weiter fortzuführen. Außerdem kann bei diesem Verfahren das Alignment an jeder Stelle in der Matrix enden. Demnach fängt der Traceback im maximalen Element irgendwo in F an, und endet, wenn man eine Zelle mit dem Wert 0 erreicht hat.

Beispiel dazu:

Wenn man sich das Gap-Modell aus dem vorherigen Beispiel anschaut und diese Formel für die Berechnung der F(i,j) anwendet, erhält man:

F		0 -	1 G	2 T	3 A	4 A
0	-	0	0	0	0	0
1	C	0	0	0	0	0
2	T	0	0	1	0	0
3	T	0	0	1	0	0
4	A	0	0	0	**2**	1
5	G	0	1	0	0	1
6	A	0	0	0	1	1

Der maximale Wert ist hier 2, und genau hier beginnt der Traceback, der nach F(T,T) endet, da man danach eine Zelle mit dem Wert 0 erreicht.

Also hat man für die Sequenzen x=CTTAGA und y=GTAA das optimale lokale Alignment:

x: TA
y: TA

Als nächstes stelle ich ein Alignment mit einem affinen Gap-Modell vor. Dafür setzt man den Score einer Gap-Region der Länge L gleich -d-e(L-1) für d>0 und e>0. Hierbei steht das -d für die Gap-opening-penalty (Kosten für den Anfang eines Gaps) und -e für die Gap-extension-penalty (Kosten für die Erweiterung eins Gaps). Da eine Gap-Region anzufangen schwieriger ist als diese zu erweitern, ist e kleiner als d. Dazu werde ich den Algorithmus eines globalen Alignments erklären, wobei man solchen für ein globales Alignment auch sehr leicht wie in dem oberen Abschnitt erhalten kann.

Im Vergleich zu den vorherigen Verfahren, hat man hier drei Matrizen: eine (n+1)x(m+1)-Matrix und zwei n x m-Matrizen. Sei also M(i,j) für i=1,…,n und j=1,…,m der Score eines optimalen Alignments zwischen $x_1 \ldots x_i$ und $y_1 \ldots y_j$, wobei gegeben ist, dass das Alignment so endet, dass x_i mit y_j aligniert. Das Element M(i,0) für i=1,…,n ist der Score für das Alignieren von $x_1 \ldots x_i$ zu einer Gap-Region der Länge i. Ebenso ist M(0,j) für j=1,…,m der Score für das Alignieren von $y_1 \ldots y_j$ zu einer Gap-Region der Länge j. Desweiteren sei $I_x(i,j)$ der Score für ein optimales Alignment zwischen $x_1 \ldots x_i$ und $y_1 \ldots y_j$, so dass das

Alignment mit einer Zuordnung von x_i zu einem Gap endet, und $I_y(i,j)$ der Score dafür, dass das es mit einer Zuordnung von y_j zu einem Gap endet. Mit der Annahme, dass eine Deletion niemals direkt nach einer Insertion erfolgt, erhält man folgende Werte:

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i,y_j) \\ I_x(i-1,j-1) + s(x_i,y_j) \\ I_y(i-1,j-1) + s(x_i,y_j) \end{cases}$$

$$I_x(i,j) = \max \begin{cases} M(i-1,j) - d \\ I_x(i-1,j) - e \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1) - d \\ I_y(i,j-1) - e \end{cases}$$

Nachdem man den Ablauf durch M(0,0)=0 initialisiert hat, kann man die Matrizen M, I_x und I_y berechnen. Wenn nun aber für ein i und j eine der Optionen (auf der rechten Seite) nicht definiert ist (zB. enthält die Gleichung für M(1,2) die Werte $I_x(0,1)$ und $I_y(0,1)$), dann wird diese bei der Berechnung nicht beachtet. Der Score für ein optimales Alignment ist dann gegeben durch $\max \{M(n,m), I_x(n,m), I_y(n,m)\}$, und der Traceback beginnt dann bei dem Element (oder auch Elementen), der dieses Maximum einnimmt.

Beispiel dazu:

Seien x=ACGGTAC und y=GAGGT Sequenzen, wobei der Score für ein Match s(a,a)=1, für ein Mismatch s(a,b)=-1 für a≠b, die Gap-opening-penalty d=3 und die Gap-extension-penalty e=2 ist. Dann erhält man die folgenden Matrizen:

M	0 -	1 G	2 A	3 G	4 G	5 T
0 -	0	-3	-5	-7	-9	-11
1 A	-3	-1	-2	-6	-8	-10
2 C	-5	-4	-2	-3	-6	-8
3 G	-7	-4	-5	-1	-2	-7
4 G	-9	-6	-5	-4	0	-3
5 T	-11	-10	-7	-6	-5	1
6 A	-13	-12	-8	-8	-7	-4
7 C	-15	-14	-12	-9	-9	-6

I_x	1 G	2 A	3 G	4 G	5 T
1 A	-6	-8	-10	-12	-14
2 C	-4	-5	-9	-11	-13
3 G	-6	-5	-6	-9	-11
4 G	-7	-7	-4	-5	-10
5 T	-9	-8	-6	-3	-6
6 A	-11	-10	-8	-5	-2
7 C	-13	-11	-10	-7	**-4**

I_y	1 G	2 A	3 G	4 G	5 T
1 A	-6	-4	-5	-7	-9
2 C	-8	-7	-5	-6	-8
3 G	-10	-7	-8	-4	-5
4 G	-12	-9	-8	-7	-3
5 T	-14	-13	-10	-9	-8
6 A	-16	-15	-11	-11	-10
7 C	-18	-17	-15	-12	-12

wobei die Einträge wie folgt berechnet werden:

$M(0,1)=M(1,0)= -3$

$M(0,2)=M(2,0)=-3-2=-5$

$M(0,3)=M(3,0)=-5-2=-7$

$M(0,4)=M(4,0)=-7-2=-9$

$M(0,5)=M(5,0)=-9-2=-11$

$M(6,0)=-11-2=-13$

$M(7,0)=-13-2=-15$

$M(1,1)= M(0,0) + s(A,G) = -1$

$I_x(1,1)= M(0,1) - 3 = -3-3 = -6$

$I_y(1,1)= M(1,0) - 3 = -3-3 = -6$

$M(1,2)= M(0,1) + s(A,A) = -3+1 = -2$

$I_x(1,2)= M(0,2) - 3 = -5-3 = -8$

$I_y(1,2)=\max \begin{cases} M(1,1) - 3 & = -1-3 = -4 \\ I_y(1,1) - 2 & = -6-2 = -8 \end{cases}$

$M(1,3)= M(0,2) + s(A,G) = -5-1 = -6$

$I_x(1,3)= M(0,3) - 3 = -7-3 = -10$

$I_y(1,3)=\max \begin{cases} M(1,2) - 3 & = -2-3= -5 \\ I_y(1,2) - 2 & = -4-2 = -6 \end{cases}$

usw....

$M(2,1)= M(1,0) + s(C,G) = -3-1 = -4$

$I_x(2,1)=\max \begin{cases} M(1,1) - 3 & = -1-3 = -4 \\ I_x(1,1) - 2 & = -6-2 = -8 \end{cases}$

$I_y(2,1)= M(2,0) - 3 = -5-3 = -8$

$M(2,2)= \max \begin{cases} M(1,1) + s(C,A) & =-1-1 = -2 \\ I_x(1,1) + s(C,A) & = -6-1 = -2 \\ I_y(1,1) + s(C,A) & = -6-1 = -2 \end{cases}$

$$I_x(2,2)=\max \begin{cases} M(1,2) - 3 & = -2-3 = -5 \\ I_x(1,2) - 2 & = -8-2 = -10 \end{cases}$$

$$I_y(2,2)=\max \begin{cases} M(2,1) - 3 & = -4-3 = -7 \\ I_y(2,1) - 2 & = -8-2 = -10 \end{cases}$$

usw….

Man sucht sich jetzt das Maximum aus {M(7,5), I_x(7,5), I_y(7,5)}, was bei I_x(7,5)=-4 liegt, und beginnt hier mit dem Traceback: der Pfeil geht als nächstes immer in die Zelle, wo der höchste Eintrag neben, schräg nach links oder über der jetzigen Position aus allen drei Matrizen ist. Das entsprechende optimale Alignment ist

x: ACGGTAC
y: GAGGT - -

Fazit:

Es gibt noch viele andere Varianten des dynamischen Programmierungs-Algorithmus, aber die wichtigsten Überlegungen, die dahinter stecken, sind ziemlich ähnlich.

Es sei noch bemerkt, dass der dynamische Programmierungs-Algorithmus einfach modifiziert werden kann, um auf die Alignierung von Sequenzen aus verschiedenen Alphabeten anwendbar zu sein. Aber alle Algorithmen ergeben exakt den höchsten Score zu einem gegebenen Scoring-Schema und erzeugen die entsprechenden optimalen Alignments immer durch ein Traceback.

Die dynamische Programmierung liefert zwar optimale Lösungen, ist aber wegen der benötigten Computerresourcen in der Praxis nicht auf sehr lange Sequenzen oder sehr große Datenbanken anwendbar. Heuristische Algorithmen eignen sich besser zur Durchsuchung der großen, global verfügbaren Datenbanken, die sämtliche bekannten Sequenzen archivieren; sie garantieren zwar keine optimalen Ergebnisse, leisten aber dennoch sehr gute Dienste, wie zB. der BLAST-Algorithmus und FASTA, und erheblich schneller.

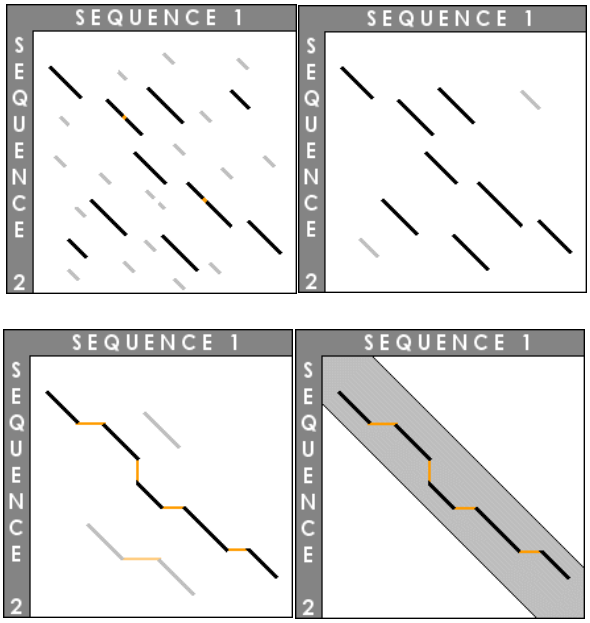
Was ist der FASTA-Algorithmus?

Wie man in den Beispielen zuvor gesehen hat, sind eigentlich nur winzige Teile in den Matrizen von Bedeutung; und zwar nur solche, in denen Zellen am Traceback beteiligt sind. Der FASTA-Algorithmus begrenzt die Bereiche der Matrizen, die die dynamische Programmierung untersucht. Solche Bereiche können zB. mithilfe der Dot-Matrix für zwei gegeben Sequenzen bestimmt werde. Diese wird so dargestellt, dass Dots an allen Stellen eingetragen werden, an denen Matches sind, sonst sind alle Einträge gleich -∞. Lokale Ähnlichkeiten können dann als diagonale Strecken fortlaufender Dots erkannt werden, die eine(vorher angegebene) Mindest-Länge k haben müssen (=Hot-spots). Kurze Darstellung des FASTA-Algorithmus:

FASTA-Algorithmus teilt sich in vier Schritte

1) Diagonale Regionen mit hoher Übereinstimmung ermitteln

3) Regionen in benachbarten Diagonalen durch Einfügen von Gaps zusammenfügen



2) Bewertung der einzelnen Regionen anhand einer Scoring-Matrix

4)

K-Band Alignment

(BLAST wird später in einem anderen Vortrag gestellt)

Was macht man nun, wenn man mehr als nur zwei Sequenzen hat, die man untersuchen will?

Um die Gemeinsamkeiten in einer Sammlung von Sequenzen zu ermitteln, muss ein optimales multiples Alignment für die gesamte Sammlung bestimmt werden. Wie auch im Fall von zwei Sequenzen, muss man auch hier jedes multiple Alignment mithilfe eines Scoring-Schemas bewerten können. Dazu wird wieder vorausgesetzt, dass die einzelnen Spalten eines Alignments, die keine Gaps enthalten, unabhängig voneinander sind, und somit eine Scoring-Funktion der Form

$$S(M) = G(M) + \sum_i s(M_i)$$

benutzt werden kann, wobei M ein multiples Alignment, M_i die i-te Spalte (ohne Gaps), $s(M_i)$ der Score von M_i und G eine Funktion für das Scoring einer Spalte mit Gaps bedeutet. In den Standard-Methoden (nicht wirklich ausreichend) werden die Spalten (ohne Gaps) mit der „Sum of Pairs“ (SP) Funktion bewertet, die für eine Spalte M_i definiert ist durch

$$s(M_i) = \sum_{k<l} s(M_i^k, M_i^l)$$ (M_i^k= Buchstabe in Spalte i und Sequenz k)

wobei hier die Summe über alle Paare (M_i^k, M_i^l), k<l, der Elemente von M_i genommen wird, und die Scores s(a,b) für a,b ϵ $\mathcal{A}$ aus einer Substitutions-Matrix (zB. PAM- oder BLOSUM-Matrizen) stammen. Nachdem dann wieder ein Scoring-Schema (siehe Fall mit zwei Sequenzen) für ein multiples Alignment bestimmt wurde, kann man einen Algorithmus für paarweise dynamische Programmierung zur Alignierung von Sequenzen n ≥ 3 verallgemeinern (= Verallgemeinerung des Needleman-Wunsch-Algorithmus, multidimensionale dynamische Programmierung):

Nehme an, man hätte Sequenzen

$$x^1 = x_{m_1}^1 \dots x_{m_1}^1 , \dots , \; x^n = x_1^n \dots x_{m_n}^n \quad .$$

Seien $i_1,\dots,i_n$ ganze Zahlen mit $0 \le i_j \le m_j$, j=1,…,n, wobei mindestens eine Zahl ungleich Null ist. $F(i_1,\dots,i_n)$ bezeichne den höchsten Score eines Alignments von Subsequenzen, die mit $x_{i_1}^1 \dots x_{i_n}^n$ enden (falls für ein j i_j=0 ist, dann sind die anderen Subsequenzen zu einer Gap-Region zugeordnet):

Dynamische Programmierung ist dann gegeben durch

$$F(i_1,\dots,i_n) = \max \begin{cases} F(i_1-1,\dots,i_n-1) + s(x_{i_1}^1,\dots, x_{i_n}^n) \\ F(i_1, i_2-1,\dots,i_n-1) + s(-, x_{i_2}^2,\dots, x_{i_n}^n) \\ F(i_1-1, i_2, i_3-1,\dots,i_n-1) + s(x_{i_1}^1, - , x_{i_3}^2,\dots, x_{i_n}^n) \\ \vdots \\ F(i_1-1,\dots, i_{n-1}-1, i_n) + s(x_{i_1}^1,\dots, x_{i_{n-1}}^{n-1}, -) \\ F(i_1, i_2, i_3-1,\dots, i_n-1) + s(- , - , x_{i_3}^3,\dots,x_{i_n}^n) \\ \vdots \end{cases}$$

wo alle Kombinationen von Gaps vorkommen, außer der, wo alle Reste durch Gaps ersetzt werden. Die Initialisierung ist durch F(0,…,0)=0 gegeben. Der Traceback beginnt bei $F(m_1,\dots,m_n)$ und ist analog zu dem von paarweisen Alignments. Die Matrix $\left(F(i_1,\dots,i_n) \right)$ mit $0 \le i_j \le m_j$, j=1,…,n ist eine $(m_1+1)x\dots x(m_n+1)$-Matrix.

Beispiel:

Finde alle optimalen Alignments der drei Sequenzen x=AATC, y=GTC und z=AAG anhand des folgenden Scoring-Schemas:

Der Score eines Alignments wird berechnet aus den Scores ihrer Spalten Mi ($S(M)= G(M) + \sum_i s(M_i)$), und zwar so, dass…

falls M_i drei identische Symbole hat, setze $s(M_i)= 2$;
falls M_i zwei identische Symbole hat, aber keine Gaps, setze $s(M_i)= 1$;
falls M_i drei unterschiedliche Symbole hat, aber keine Gaps, setze $s(M_i)= -1$;
falls M_i genau ein Gap hat, setze $s(M_i)= -2$;
falls M_i genau zwei Gaps hat, setze $s(M_i)= -4$.

Die Indizes i_1, i_2 und i_3 entsprechen jeweils den Sequenzen x, y und z.

So erhält man die Matrizen **F(*,*,0)** (hier ist in jeder Spalte das unterste Symbol (Symbol in z) ein Gap, dh die ersten beide Symbole in der Spalte alignieren immer zu einem Gap), **F(*,*,1)** (hier alignieren die ersten beiden Symbole zu dem ersten Symbol aus z (=A)), **F(*,*,2)** (hier alignieren die ersten beiden Symbole zu dem zweiten Symbol aus z (=A)) und **F(*,*,3)** (hier alignieren die ersten beiden Symbole zu dem dritten Symbol aus z (=G)):

F(*,*,0)		0 -	1 G	2 T	3 C
0	-	0	-4	-8	-12
1	A	-4	-2	-6	-10
2	A	-8	-6	-4	-8
3	T	-12	-10	-8	-6
4	C	-16	-14	-12	-10

F(*,*,1)		0 -	1 G	2 T	3 C
0	-	-4	-2	-6	-10
1	A	-2	1	-3	-7
2	A	-6	-3	-1	-5
3	T	-10	-7	-5	-3
4	C	-14	-11	-9	-7

F(*,*,2)	0 -	1 G	2 T	3 C
0 -	-8	-6	-4	-8
1 A	-6	-3	-1	-5
2 A	-4	-1	2	-2
3 T	-8	-5	-2	0
4 C	-12	-9	-6	-4

F(*,*,3)	0 -	1 G	2 T	3 C
0 -	-12	-10	-8	-6
1 A	-10	-7	-5	-3
2 A	-8	-5	-2	0
3 T	--6	-3	0	1
4 C	-10	-7	-4	-1

Ein paar Rechenbeispiele:

F(0,2,0)= 2 * (-4)= -8 (Gap der Länge 2, deshalb (-4)+(-4))

F(3,2,0)= F(2,1,0) + s(T,T,-)= -6-2= -8

F(4,3,0)= F(3,2,0) + s(C,C,-)= -8-2= -10

F(2,2,1)= max
- F(1,1,0) + s(A,T,A) = -2+1 = -1
- F(2,1,0) + s(-,T,A) = -6-2 = -8
- F(1,2,0) + s(A,-,A) = -6-2 = -8
- F(1,1,1) + s(A,T,-) = 1-2 = -1
- F(2,2,0) + s(-,-,A) = -4-4 = -8
- F(1,2,1) + s(A,-,-) = -3-4 = -7
- F(2,1,1) + s(-,T,-) = -3-4 = -7

F(3,3,2)= max
- F(2,2,1) + s(T,C,A) = -1-1 = -2
- F(3,2,1) + s(-,C,A) = -5-2 = -7
- F(2,3,1) + s(T,-,A) = -5-2 = -7
- F(2,2,2) + s(T,C,-) = 2-2 = 0
- F(3,3,1) + s(-,-,A) = -3-4 = -7
- F(2,3,2) + s(T,-,-) = -2-4 = -6
- F(3,2,2) + s(-,C,-) = -2-4 = -6

usw….

Man zeichnet wieder Pfeile ein, die anzeigen, aus welchen Elementen die Zahlen hergeleitet wurden, und startet nun den Traceback bei F(4,3,3)=-1und geht durch die Zellen, bis man F(0,0,0) erreicht hat. Dadurch erhält man drei Pfade:

F(4,3,3) → F(3,2,2) → F(2,1,1) → F(1,0,0) → F(0,0,0)

F(4,3,3) → F(3,2,2) → F(2,1,1) → F(1,1,1) → F(0,0,0)

F(4,3,3) → F(3,2,2) → F(2,2,2) → F(1,1,1) → F(0,0,0)

So dass man die drei optimalen Alignments mit dem Score -1 hat:

x: A A T C	x: A A T C	x: A A T C
y: - G T C	y: G - T C	y: G T - C
z: - A A G	z: A - A G	z: - A A G

Fazit:

Multiple Sequenz-Alignments sind zwar in der Lage, entferntere Ähnlichkeiten aufzuspüren und bieten besseres funktionelles Verständnis von Sequenzen und ihren Beziehungen, sind aber wegen ihres enormen Zeitaufwandes in der Praxis nicht anwendbar (dafür sind eher heuristische Methoden empfehlenswert).

Bis jetzt hatten wir immer angenommen, dass Substitutions-Matrizen, wie PAM- und BLOSUM-Matrizen, gegeben seien, Aber wie konstruiert man eigentlich solche?

Die allgemeine Form sieht so aus:

Um die Signifikanz des Scores eines lokalen oder globalen Gap-freien Alignments zwischen den Sequenzen x^0 und y^0 zu erhalten, muss man eine Substitutions-Matrix haben, die ganze Zahlen als Einträge hat und gewisse Voraussetzungen erfüllt (werden im Vortrag 6 erläutert), wobei $\{p_a\}$ und $\{p_b`\}$, hergeleitet aus x^0 und y^0, die jeweiligen Häufigkeiten sind, mit der die aus dem Alphabet $\mathcal{A}$ stammenden Buchstaben in den Sequenzen vorkommen. Diese Voraussetzungen sollen für ein Random-Modell erfüllt sein. Daraus erhält man zulässige Matrizen. Man nimmt an, dass $p_a > 0$ für alle $a \in \mathcal{A}$ und nennt die Werte $\{ p_a \}$ Background-Häufigkeiten (zu dem jetzigen Zeitpunkt nimmt man an, dass diese gegeben seien; wie man sie bestimmt, wird später erklärt). Die Elemente einer zulässigen Substitutions-Matrix werden wie folgt dargestellt

$$s(a,b)=C * \ln\left(\frac{p_{ab}}{p_a p_b}\right)$$

für eine bestimmt Wahl von $\{p_{ab}\}$ (alle grösser 0), also für ein bestimmtes Match-Modell, und ein $C > 0$, wobei $p_{ab} \neq p_a p_b$.

Der allgemeine Ansatz, zulässige Substitutions-Matrizen zu konstruieren, beginnt mit der Wahl eines Random-Modells $\{p_a\}$ und eines Match-Modells $\{p_{ab}\}$. Dann betrachtet man die Matrix für ein bestimmtes C. Wähle $\{p_{ab}\}$ so, dass $p_{ab} = p_{ba}$, damit die resultierende Matrix symmetrisch ist. Es könnte sein, dass nicht alle Einträge ganze Zahlen sind. Deshalb muss man sie zur nächsten

ganzen Zahl runden und dann etwas abändern (damit der größte gemeinsame Teiler der Einträge gleich 1 ist).

Wie konstruiert man die PAM-Matrix?

PAM steht für „*point accepted mutations*“. Dabei handelt es sich um Substitutionen von Aminosäuren durch andere Aminosäuren, allein durch natürliche Selektion hervorgerufen. Alle Aminosäure-Unterschiede sind demnach Punktmutationen. Bei dieser Art von Substitutions-Matrizen sollten die Sequenzen möglichst nah verwandt sein. Die Konstruktion der PAM-Matrizen (und auch der PAMn-Matrizen) fängt mit Blöcken an, die lokale Gap-freie multiple Alignements von Proteinen darstellen, dessen Sequenzen aus existierenden Datenbanken frei zugänglich sind. Die Sequenzen in einem Block dürfen nicht mehr als 15% voneinander abweichen. Die Blöcke werden basierend auf die Analyse von Matches und Mismatches zwischen den Sequenzen erstellt, das hauptsächlich durch die Verwendung der multidimensionalen Einheits-Matrix als Substitutions-Matrix für multiple Alignments geschieht. Bestimme nun, für ein gegebenes Random-Modell ein Match-Modell. Dafür muss man aus den Blöcken erst eine Matrix $S(1)=(S(1)_{ab})$ erstellen, die dann die Matrix der Übergangs-Wahrscheinlichkeiten einer Markov-Kette ist, dessen Elemente, die alle ungleich 0 sind, durch die Buchstaben von $\mathcal{A}$ bezeichnet werden (S(1) beschreibt die Aminosäure-Substitutionen nach zwei Schritten auf der Markov-Kette). Die stationäre Verteilung der Markov-Kette ist $\varphi=(p_1,\ldots,p_{20})$.

Kurze Zusammenfassung der Vorgehensweise:

Es seien Blöcke gegeben → Dazu bildet man die sparsamsten phylogenetischen Bäume → Zähle, wie oft die Aminosäure-Substitutionen ab=ba, a,b $\in \mathcal{A}$, in jedem Block vorkommen und teile die Ergebnisse jeweils durch die Anzahl der Bäume, die in dem Block sind, aus der das Ergebnis stammt → Bilde die Summe für jedes Paare und dann die Summe über alle Paare → Man erhält die symmetrische Matrix $H=(H_{ab})$ → Berechne die Background-Häufigkeiten p_a → Dann die Matrizen S(1) und $S(2)= S(1)^2$ → Dann die joint-probabilities p_{ab} (reflektiert die natürliche Austauschrate zweier Aminosäuren) → Berechne alle möglichen $s(a.b)=C * \ln\left(\frac{p_{ab}}{p_a p_b}\right)$ und runde die Ergebnisse zur nächsten ganzen Zahl, so dass man alle Einträge in der PAM2-Matrix erhält.

Fazit:

PAM-Substitutions-Matrizen treten dann auf, wenn die verfügbaren Daten überwiegend aus Sequenz-Familien sehr verwandter Proteine stammen. Aus solchen kann man sich nur erhoffen, dass man eine short-term (kurzzeitige)

evolutionäre Matrix S(1) ermitteln kann. Wenn Daten von weniger verwandten Daten vorliegen, zeigt die Matrix S(1) für große n nicht das reelle Schema von long-term (langzeitigen) Aminosäure-Substitutionen an.

Was macht man also, wenn man evolutionär distanziertere Proteine hat?

Dafür ist die BLOSUM-Familie (BLOcks SUbstitutions Matrices) von Substitutions-Matrizen besser. Wie der Name schon sagt, sind die Matrizen aus einer Datenbank von Blöcken hergeleitet. Wie auch bei PAM-Matrizen, sind die Blöcke anfangs basierend nur auf die Analyse von Matches und Mismatches der Sequenzen konstruiert. Am Anfang wird auch hier zur Bildung der Blöcke die Einheitsmatrix als Substitutions-Matrix genommen. Zur Bildung der BLOSUM-Matrizen beginnt man mit der Wahl von r und dann der Bildung von Clustern der Sequenzen in einem Block, und zwar so, dass man die Sequenzen eines Blockes so gruppiert, dass eine Sequenz in einem Cluster eine r%-ige oder höhere Sequenzübereinstimmung zu mindestens einer anderen Sequenz in dem Cluster aufweist.

Beispiel dazu:

Nehme an, $\mathcal{A}$ sei das 3-Buchstaben-Alphabet {A,B,C} und man hätte die 3 Blöcke:

x^1 : A B C A B	x^6 : A B C	x^{12} : A A A C B A B C
x^2 : A B C A C	x^7 : A B C	x^{13} : B A A C B A B C
x^3 : B B C A B	x^8 : A A C	x^{14} : A A A C B A C B
x^4 : C B C A C	x^9 : C B C	x^{15} : A A A C B A C C
x^5 : A A A C B	x^{10} : A A B	
	x^{11} : B A B	

Für r=80 erhält man

Zwei Cluster im ersten Block: $\{x^1,x^2,x^3,x^4\}$, $\{x^5\}$

Fünf Cluster im zweiten Block: $\{x^6,x^7\}$, $\{x^8\}$, $\{x^9\}$, $\{x^{10}\}$, $\{x^{11}\}$

Ein Cluster im dritten Block: $\{x^{12},x^{13},x^{14},x^{15}\}$

Das Clustering ist dafür da, um die Überpräsenz sehr verwandter Sequenzen zu reduzieren. Jedes Cluster wird wie eine Einzel-Sequenz behandelt. Als nächstes wird die Matrix $H=(H_{ab})$ konstruiert, mit den Werten, wie folgt: Die Matrix H ist symmetrisch, und ihre Spalten und Zeilen sind indiziert mit den Elementen aus $\mathcal{A}$. Man bestimme $a,b \in \mathcal{A}$ und sei zunächst $a \neq b$. Danach schaut man sich einen Block an und sucht sich genau zwei Cluster aus. Seien n und m die Anzahlen der Sequenzen in den Clustern. Man zählt nun, wie oft

a und b (als Paar ab) an der gleichen Position in den zwei Clustern vorkommen, aber nur solche dazu zählen, wo a un b nicht zum selben Cluster gehören, und teile das Ergebnis durch nm. Zuletzt summiert man die Ergebnisse über alle Cluster-Kombinationen in einem Block auf und danach über alle Blöcke. Man erhält die Einträge $H_{ab}=H_{ba}$. Für die Diagonal-Einträge (=Buchstaben-Übereinstimmungen aa) macht man dasselbe, und multipliziert hier die Ergebnisse mit 2:

(zur Verschnellerung habe ich direkt über alle Blöcke summiert Block 1 + Block 2 + Block 3)

A-A = $2*(\frac{2}{4}+\frac{2}{2}+\frac{2}{2}+1+1+1+1)=13$

A-B = $\frac{1}{4} + 4*(\frac{1}{4}) + \frac{2}{2} +1+1+ 3*(\frac{2}{2}) + 1+1+1= \frac{41}{4}$

A-C = $\frac{1}{4} + 4*(\frac{1}{4}) + 4*(\frac{1}{4}) + \frac{2}{2} +1+1= \frac{21}{4}$

B-C = $\frac{2}{4} +1+ 2*(\frac{2}{2}) +1+1+1+1= \frac{15}{2}$

B-B = $2* (\frac{2}{4} + \frac{2}{2} +1)= 5$

C-C = 2*3= 6

H	A	B	C
A	13	$\frac{41}{4}$	$\frac{21}{4}$
B	$\frac{41}{4}$	5	$\frac{15}{2}$
C	$\frac{21}{4}$	$\frac{15}{2}$	6

Man bestimme nun die Background-Häufigkeiten mithilfe der Formel $p_a = \frac{\sum_{b \in \mathcal{A}} H_{ab}}{D}$, wobei $D = \sum_{a,b \in \mathcal{A}} H_{ab}$ (Gesamtanzahl der Substitutionen). Nehme dabei an, dass $p_a > 0$ für alle $a \in \mathcal{A}$.

D = $13+5+6+ 2*(\frac{41}{4} + \frac{21}{4} + \frac{15}{2}) = 70$

$p_A = \frac{13+\frac{41}{4}+\frac{21}{4}}{70} = \frac{57}{140}$, $p_B = \frac{\frac{41}{4}+5+\frac{15}{2}}{70} = \frac{91}{280}$, $p_C = \frac{\frac{21}{4}+6+\frac{15}{2}}{70} = \frac{75}{280}$.

Setze für ein Match-Modell

$p_{ab} = \frac{H_{ab}}{D}$,

so dass man für die oberen Werte folgendes erhält:

$p_{AA} = \frac{13}{70}$, $p_{AB} = p_{BA} = \frac{41}{280}$, $p_{AC} = p_{CA} = \frac{21}{280}$, $p_{BC} = p_{CB} = \frac{15}{140}$

$p_{BB} = \frac{5}{70}$, $p_{CC} = \frac{6}{70}$

Mit der Formel $s(a,b)=C * \ln\left(\frac{p_{ab}}{p_a p_b}\right)$ und $C=\frac{2}{ln2}$ erhält man für die Blöcke die Level-1-BLOSUM80-Matrix

$$\text{Level-1-BLOSUM80} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix}$$

Jetzt wird diese Level-1-BLOSUMr-Matrix benutzt, um eine neue Menge von Blöcken zu erhalten, und das Verfahren wird wiederholt. Das führt zu einer Level-2-BLOSUMr-Matrix, die dann auch wieder angewandt wird, um eine dritte Datenbank von Blöcken zu erhalten. Die endgültige BLOSUMr-Matrix wird dann aus dieser auf dieselbe Art hergeleitet. Dabei kann der Clustering-Prozentsatz r im letzten Schritt anders sein, als die aus den ersten beiden Schritten.

Woher weiß man denn, wann man welche Matrix benutzen muss?

Wenn man einiges über die evolutionären Distanzen zwischen den zu betrachtenden Sequenzen weiß, dann kann man dementsprechend die BLOSUM-Matrix auswählen. Falls keine vorherigen Informationen vorhanden sind, werden oft BLOSUM62 und BLOSUM50 benutzt; BLOSUM62 für Gap-freie Matches, und BLOSUM50 um Alignments zu finden, die Gaps beinhalten.

Und welche ist nun besser, PAM- oder BLOSUM-Matrix?

PAM-Matrizen werden eher auf verwandte Proteine angewandt, BLOSUM hingegen für weniger verwandte.

Vergleich von PAM- und BLOSUM-Matrizen:

Enge Verwandtschaft (niedrige PAM, hohe BLOSUM)

Entferntere Verwandtschaft (hohe PAM, niedrige BLOSUM)

BLOSUM-Matrix geeigneter für lokale Alignments

PAM-Matrix geeigneter für globale Alignments

BLOSUM80	BLOSUM62	BLOSUM45
PAM1	PAM120	PAM250

Wenig unterschiedlich ⟷ deutlich unterschiedlich

Ungefähr gleichwertig sind:

PAM100 ⇨ BLOSUM90

PAM120 ⇨ BLOSUM80

PAM160 ⇨ BLOSUM60

PAM200 ⇨ BLOSUM52

PAM250 ⇨ BLOSUM45