Einführung in die Bioinformatik

Enno Ohlebusch

Abteilung Theoretische Informatik
Universität Ulm

October 18, 2016

Überblick

Datenbanken

exponentielles Wachstum der Sequenzdaten

1996: knapp 60 Datenbanken
Januar 2004: ca. 540 Datenbanken

Primäre DB: Nukleotid- und Proteinsequenzen, meist keine "richtigen" Datenbanken, sondern flat files

Es gibt Datenbanken, die auf bestimmte Organismen oder Organismengruppen spezialisiert sind.

Abgeleitete DB: gefilterte und interpretierte Sequenzinformation

Primäre Datenbanken

Primäre Sequenzdatenbanken:

- Genbank am National Center for Biotechnology Information (NCBI), USA; `http://www.ncbi.nlm.nih.gov`, Suchmaschine ENTREZ
- Europea Molecular Biology Laboratory (EMBL) am European Bioinformatics Institute (EBI) in Hinxton, England; `http://www.embl-heidelberg.de/`, Suchmaschine SRS (Sequence Retrieval System)
- DNA Databank of Japan (DDBJ); `http://www.nig.ac.jp/`, eigene einfache Suchmaschine getentry, aber auch SRS

täglicher Abgleich der Einträge auf Grund einer Kooperation von Genbank, EMBL und DDBJ

Suche in Datenbanken

Prinzip der Datenbanksuche:

ähnliche DNA Sequenzen

$\Rightarrow$ ähnliche Aminosäuresequenzen

$\Rightarrow$ ähnliche Proteinstruktur

$\Rightarrow$ ähnliche Funktion

Vorsicht!

Obige Vorgehensweise liefert nur Arbeitshypothesen, die zwar oft, aber nicht immer, zum Ziel führen.

Nicht-redundante Datenbanken

Genbank, EMBL und DDBJ sind redundant, weil die Daten nicht überprüft werden.
Jeder Wissenschaftler kann seine Sequenzen selbst eintragen.
Daher treten Sequenzen häufig mehrfach auf.

Eine nicht-redundante Datenbanken ist UniProt (Universal Protein Resource). Entstand 2002 durch den Zusammenschluß von

- Swiss-Prot: alle Einträge sind manuell annotiert, viele Querverweise auf andere Datenbanken
- TrEMBL (translated EMBL): automatische Translation und Annotation der proteinkodierenden Sequenzeinträge aus EMBL
- PIR-PSD (Protein Information Resource-Protein Sequence Database): ebenfalls manuell annotiert

FASTA Format

Einfaches und weitverbreitetes Sequenzformat.

erste Zeile: beginnt mit >, gefolgt vom Sequenznamen und evtl. Beschreibung der Sequenz.

zweite Zeile: eigentliche Sequenz.

Beispiel:

>emb|AL096836| Pyrococcus abyssi complete genome
GGGCTTTAGCCTCCTTCACCGCTTCCACGATTTTCTGCCTGTCAAAG
GGTTTTTAATTAAAAATTCAAGGTGGAGTAAAAAGGGATGTTTTTAA
TTCTCAAATAGCTCGTCGTAAACCCCTTCATCTATTTCTCTCTGAAC
CGGTAACTCCCATGCTTAAAGCCGTTCCAATGACTTCCTTGGCGGCA
CTGGTTTCTCTTCATCTTAGCTATCTTGATAACTTGCTCCATCGTTA
GGCTCACCGCTGCCCTTCTCGAGCCCTAGTTCCTTCTTTATCAACTG

Alignments

Alignment = Ausrichtung zweier Sequenzen

$$\begin{aligned} Seq\ 1: &\ t\ a\ t\ a - t\ a\ c\ g\ c\ t\ a\ g\ c\ a \\ Seq\ 2: &\ t\ a\ t\ a\ a\ t\ a\ g\ g\ c\ t - g\ c\ a \end{aligned}$$

Sind zwei ausgerichtete Nukleotide identisch $\Rightarrow$ Match

Sind zwei ausgerichtete Nukleotide nicht identisch $\Rightarrow$ Mismatch

Daneben gibt es Lücken, die dadurch entstehen, dass in die Sequenzen Positionen eingefügt (Insertionen) oder aber entfernt werden (Deletionen).

Obiges Alignment hat 80% Identität, denn 12 von den 15 ausgerichteten Positionen sind identisch.

Alignments

Offensichtlich gibt es viele Möglichkeiten zwei Sequenzen auszurichten (zu alignieren).

Folgendes Alignment hat nur 60% Identität.

$$\begin{aligned} Seq\ 1: &\ - \ t \ a \ t \ a \ t \ a \ c \ g \ c \ t \ a \ g \ c \ a \\ Seq\ 2: &\ t \ a \ t \ a \ a \ t \ a \ g \ g \ c \ t \ - \ g \ c \ a \end{aligned}$$

Nukleotidsequenzen: Unter allen Alignments zweier DNA Sequenzen, finde eines mit maximaler Identität (= mimimaler Anzahl von Mismatches, Insertionen und Deletionen).

Aminosäuresequenzen: Unter allen Alignments zweier Sequenzen, finde eines mit maximaler Ähnlichkeitbewertung bzgl. eines vorgegebenen Ähnlichkeitsmaßes (similarity score).

Substitutionsmatrizen

Ein Ähnlichkeitsmaß bewertet Insertionen, Deletionen und die Substitution einer Aminosäure durch eine andere Aminosäure. Am häufigsten werden in der Praxis PAM und BLOSUM Matrizen verwendet.

PAM-Matrizen: PAM steht für Percent Accepted Mutation; die Matrizen wurden in den 70er Jahren von Margaret Dayhoff entwickelt.

BLOSUM (Blocks Substitution Matrix): 1992 von Jorja und Steven Henikoff aufgestellt.

PAM250

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

BLOSUM62

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

Dotplot

Einfachste Art eines paarweisen Vergleichs von Sequenzen: Dotplot (Punktdiagramm)

Die erste Sequenz wird auf der X-Achse, die zweite auf der Y-Achse abgetragen.

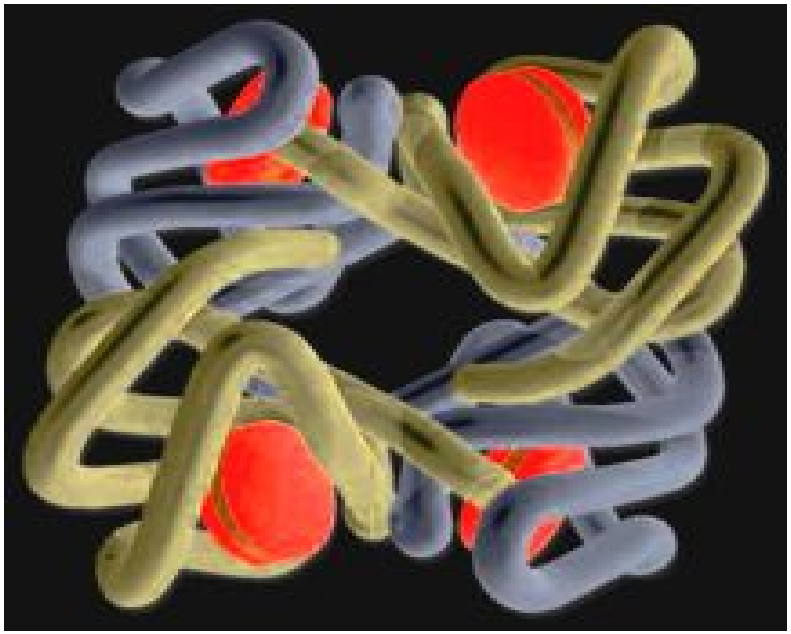
Identitäten visualisieren: Überall dort, wo man identische Positionen findet wird ein Punkt gemacht.

k-Wortmethode als Dotplot-Filter: Es werden nur dann Punkte eingezeichnet, wenn k aufeinanderfolgende Positionen identisch sind (also ein exakter Match der Länge k vorliegt).

Ähnlichkeit visualisieren: Überall dort, wo der Eintrag in der verwendeten Substitutionsmatrix einen vorgegebenen Schwellwert überschreitet, wird ein Punkt gemacht.

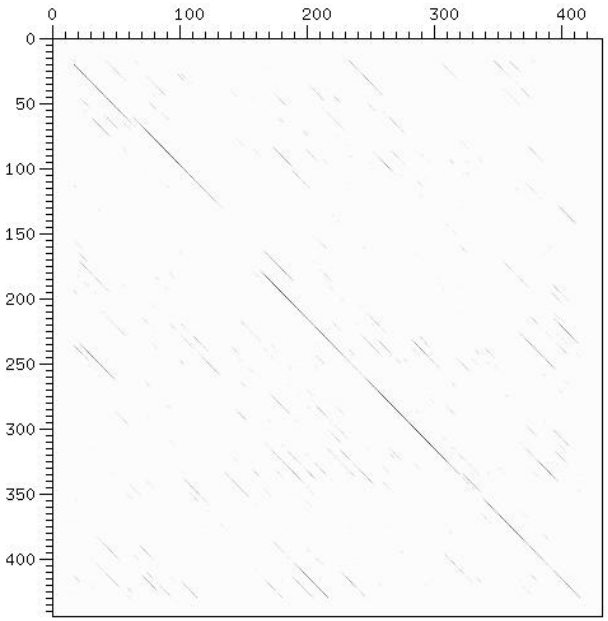
Hämoglobin

Modell eines Hämoglobinmoleküls. Rot die 4 Häm-Gruppen, die je ein Sauerstoff-Molekül binden können.
Zu jedem Häm gehört eine Globinkette. Es gibt zwei verschiedene Ketten (alpha- und beta-Ketten).



Dotplot

Dotplot zweier kodierender DNA Sequenzen.
Horizontale Achse: Alpha Kette des menschlichen Hämoglobins.
Vertikale Achse: Beta Kette des menschlichen Hämoglobins.



Dotplot

Fenstermethode als Dotplot-Filter: In einem Fenster vorgegebener Größe (z.B. 15 Felder) wird mit Hilfe der verwendeten Substitutionsmatrix jedem Feld der entsprechende Wert zugeordnet.

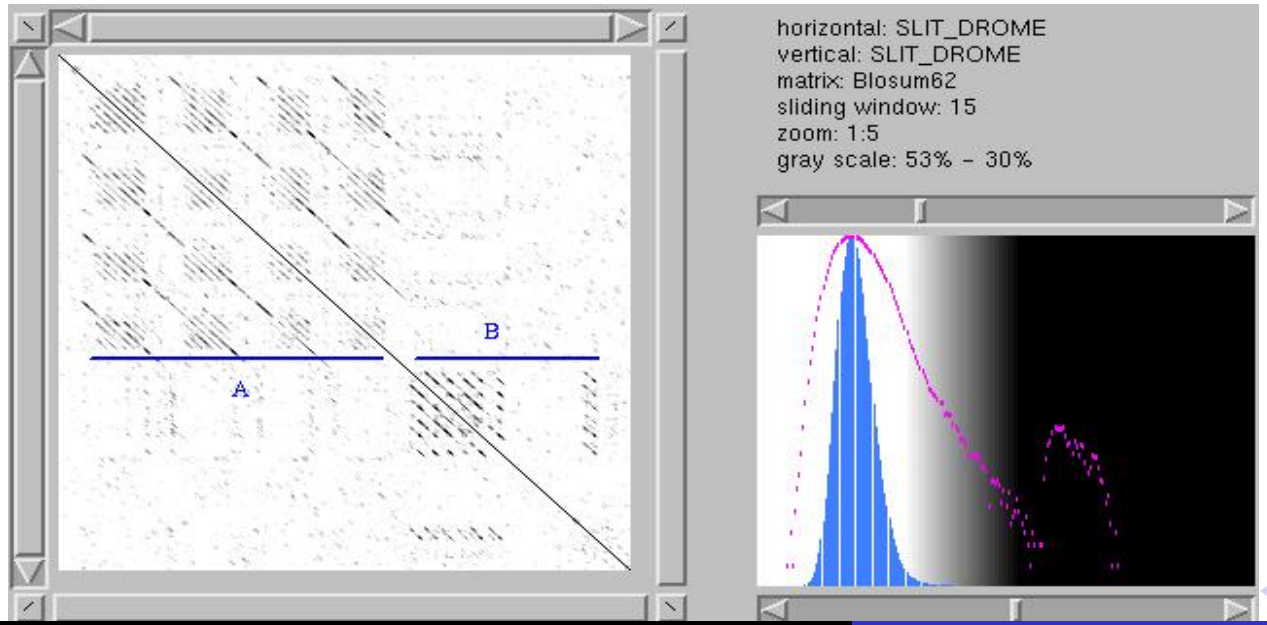
Dann wird die Summe aus den Werten gebildet.

Ist die Summe größer oder gleich dem vorgegebenen Schwellwert, so wird in der Mitte des Fensters ein Punkt gesetzt.

Dann wird das Fenster um ein Feld verschoben und erneut die Summe gebildet etc.

Dotplot

Dotplot des SLIT Proteins von Drosophila melanogaster gegen sich selbst. Im N-Terminus (A), gibt es vier wiederholte Bereiche, die selbst aus kleineren Repeat-Einheiten aufgebaut sind. Es gibt einen weiteren Bereich, der in einem Cluster wiederholt vorkommt (B) und auch nahe dem C-Terminus auftritt.



Globales Alignments

Nukleotidsequenzen: Unter allen Alignments zweier DNA Sequenzen, finde ein optimales, d.h. eines mit mimimaler Anzahl von Mismatches, Insertionen und Deletionen.

Einheitskostenmatrix δ:

δ	-	A	C	G	T
-		1	1	1	1
A	1	0	1	1	1
C	1	1	0	1	1
G	1	1	1	0	1
T	1	1	1	1	0

Kosten eines Alignments bzgl. δ: Anzahl von Mismatches, Insertionen und Deletionen im Alignment

Dynamic Programming

Dynamic Programming Algorithmus zur Berechnung eines optimalen Alignments zweier Nukleotidsequenzen S_1 und S_2 benutzt folgende Rekursiongleichungen:

$$E(0,0) = 0 \tag{1}$$

$$E(i,0) = i \tag{2}$$

$$E(0,j) = j \tag{3}$$

$$E(i,j) = \min \left\{ \begin{array}{l} E(i-1,j)+1 \\ E(i,j-1)+1 \\ E(i-1,j-1)+\delta(S_1[i],S_2[j]) \end{array} \right\} \tag{4}$$

Dynamic Programming, Beispiel

	$E(i,j)$		t	g	a	t	a	t
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
g	1	1	1	1	2	3	4	5
c	2	2	2	2	2	3	4	5
a	3	3	3	3	2	3	3	4
c	4	4	4	4	3	3	4	4
t	5	5	4	5	4	3	4	**4**

Die minimale Anzahl von Mismatches, Insertionen und Deletionen um $S_1 = gcact$ in $S_2 = tgatat$ zu überführen ist **4**.

Dynamic Programming, Beispiel

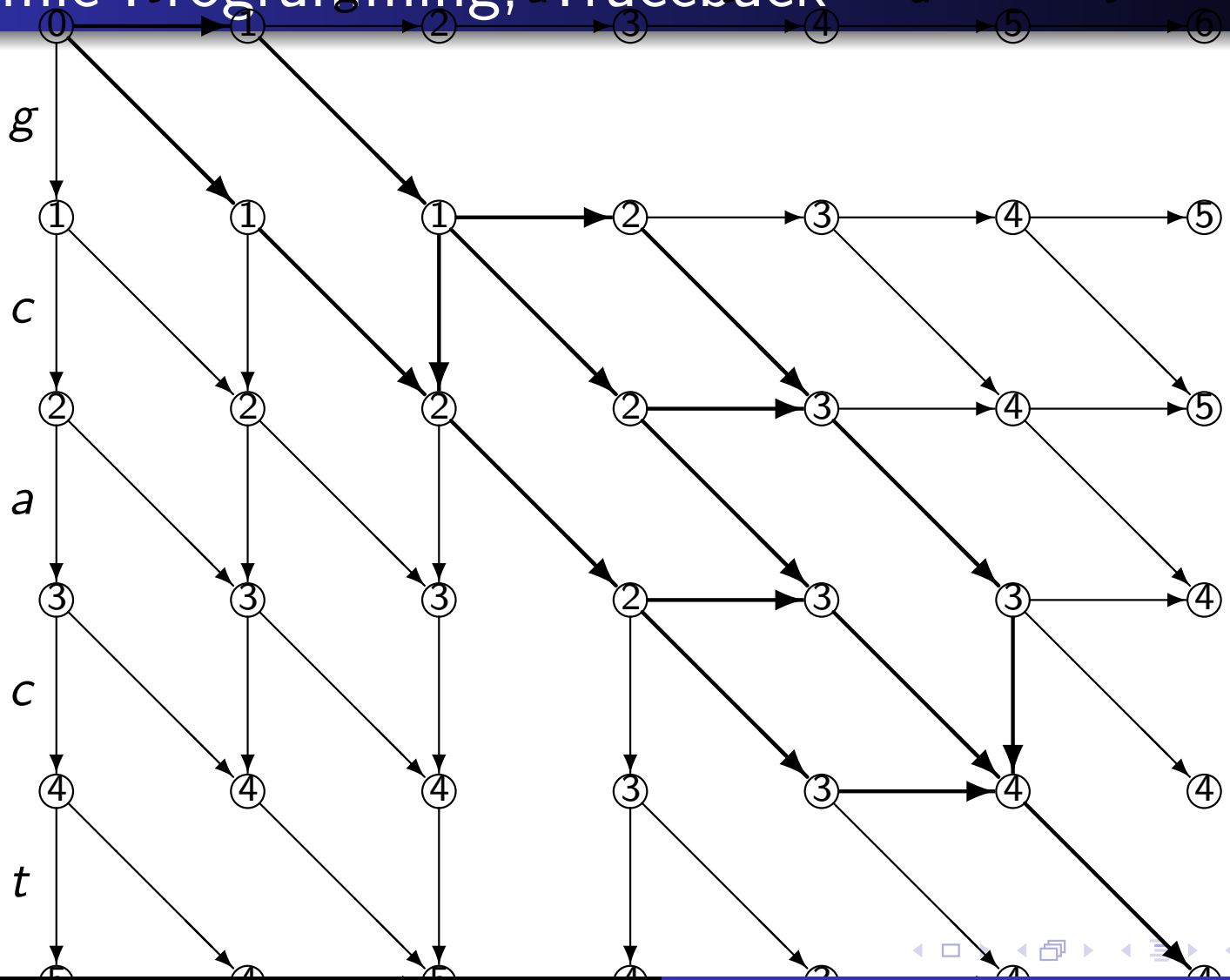
Um ein optimales Alignment der Sequenzen S_1 (Länge m) und S_2 (Länge n) zu erhalten, muß man sich merken, auf welchem Weg ein Eintrag $E(i, j)$ in der Matrix enstanden ist.

Wenn man alle minimierenden Kanten in die Matrix einzeichnet, erhält man einen gerichteten Graphen (s. nächste Folie).

Jeder Pfad vom Knoten $E(0, 0)$ zum Knoten $E(m, n)$ entspricht dann einem optimalen Alignment (nach rechts gehen entspricht einer Insertion, nach unten gehen entspricht einer Deletion und über die Diagonale zu gehen entspricht einer Substitution).

Diese Pfade findet man am einfachsten, indem man ausgehend vom Knoten $E(m, n)$ rückwärts Wege zum Knoten $E(0, 0)$ sucht (Traceback).

Dynamic Programming, Traceback



Globales Alignment

Unter allen Alignments zweier Sequenzen, finde ein optimales, d.h. eines mit maximaler Ähnlichkeitbewertung.
Needleman & Wunsch Algorithmus:

$$S(0,0) = 0$$
$$S(i,0) = i \cdot -g$$
$$S(0,j) = j \cdot -g$$
$$S(i,j) = \max \left\{ \begin{array}{l} S(i-1,j) - g \\ S(i,j-1) - g \\ S(i-1,j-1) + score(S_1[i], S_2[j]) \end{array} \right\}$$

wobei g die Kosten einer Lücke der Länge 1 bezeichnet und $score(A, B)$ die Bewertung eines Austausches von A und B gemäß verwendeter Substitutionsmatrix ist.

Lokales Alignment

In vielen Fällen sind zwei Proteine global nicht sehr ähnlich, haben aber Bereiche (Domänen), die sehr ähnlich zueinander sind. Der Smith & Waterman Algorithmus erlaubt es solche lokalen Ähnlichkeiten zu bestimmen.

$$S(i,0) = 0$$
$$S(0,j) = 0$$
$$S(i,j) = \max \left\{ \begin{array}{l} 0 \\ S(i-1,j) - g \\ S(i,j-1) - g \\ S(i-1,j-1) + score(S_1[i], S_2[j]) \end{array} \right\}$$

Lokales Alignment

Die vierte Alternative in der Rekursiongleichung, die 0, stellt sicher, dass überall ein "neues" Alignment begonnen werden kann.

Nach der Berechnung der Dynamic Programming Matrix bestimmt man einen Eintrag mit maximalem Wert und folgt den maximierenden Kanten rückwarts, bis man einen Eintrag mit Wert 0 erreicht (backtrace). Dies liefert ein bestes lokales Alignment der Sequenzen.

Allgemeine Gap-Kosten

Sei g eine Kostenfunktion, die nur $g(1) \leq g(2) \leq g(3) \leq \ldots$ erfüllen muß, wobei $g(k)$ die Kosten einer Lücke der Länge k bezeichnet. Dynamic Programming Algorithmus zur Berechnung eines globalen optimalen Alignments:

$$S(0,0) = 0$$
$$S(i,0) = -g(i)$$
$$S(0,j) = -g(j)$$
$$S(i,j) = \max \left\{ \begin{array}{l} \max_{1 \leq k \leq i} \{S(i-k,j) - g(k)\} \\ \max_{1 \leq k \leq j} \{S(i,j-k) - g(k)\} \\ S(i-1,j-1) + score(S_1[i], S_2[j]) \end{array} \right\}$$

Die bisherigen Algorithmen haben die Zeitkomplexität $O(mn)$, dieser jedoch $O(mn(m+n))$

Affine Gap-Kosten

Lücken in einem Proteinalignment kommen selten vor. Wenn es aber zu einer Lücke kommt, so erstreckt sich diese meistens über einen längeren Bereich.
Affine Gap-Kosten der Art $g(k) = a + b(k-1)$ tragen dem Rechnung. Dabei ist a die "gap-open penalty" und b die "gap-extension penalty", wobei $a > b$ (z.B. $a = 12$ und $b = 2$).
Rekursionsgleichung:

$$
\begin{aligned}
S(i,j) &= \max\{E(i,j), F(i,j), S(i-1,j-1) + score(S_1[i], S_2[j])\} \\
E(i,j) &= \max_{1 \le k \le i} \{S(i-k,j) - (a + b(k-1))\} \\
F(i,j) &= \max_{1 \le k \le j} \{S(i,j-k) - (a + b(k-1))\}
\end{aligned}
$$

Affine Gap-Kosten

Es gilt:

$$
\begin{aligned}
& F(i,j) \\
&= \max_{1\le k\le j}\{S(i,j-k)-(a+b(k-1))\} \\
&= \max\{S(i,j-1)-a, \max_{2\le k\le j}\{S(i,j-k)-(a+b(k-1))\}\} \\
&= \max\{S(i,j-1)-a, \max_{1\le k\le j-1}\{S(i,j-k-1)-(a+bk))\}\} \\
&= \max\{S(i,j-1)-a, \max_{1\le k\le j-1}\{S(i,j-k-1)-(a+b(k-1))\}-b\} \\
&= \max\{S(i,j-1)-a, F(i,j-1)-b\}
\end{aligned}
$$

Affine Gap-Kosten

Damit erhalten wir folgende Rekursionsgleichung:

$$\begin{aligned} S(i,j) &= \max\{E(i,j), F(i,j), S(i-1,j-1) + score(S_1[i], S_2[j])\} \\ E(i,j) &= \max\{S(i-1,j) - a, E(i-1,j) - b\} \\ F(i,j) &= \max\{S(i,j-1) - a, F(i,j-1) - b\} \end{aligned}$$

D.h. jeder der $m \cdot n$ Einträge der Matrix entsteht aus dem Maximum von 5 Werten. Also ist die Zeitkomplexität dieses von Gotoh entwickelten Algorithmus $O(mn)$.

PAM Matrizen

Wir betrachten zwei Sequenzen $S_1 = x_1 x_2 \dots x_n$ und $S_2 = y_1 y_2 \dots y_n$ und deren Alignment (ohne Lücken, d.h. ohne Insertionen und Deletionen) unter zwei konkurrierenden Modellen.

In dem Zufallsmodell R ist die Annahme, dass jede Aminosäure a unabhängig mit einer Wahrscheinlichkeit p_a auftritt.

Die Wahrscheinlichkeit eines Alignments von S_1 und S_2 im Zufallsmodell ist:

$$P(S_1, S_2 \mid R) = \prod_{i=1}^{n} p_{x_i} \prod_{j=1}^{n} p_{y_j}$$

PAM Matrizen

In dem Mutationsmodell M ist die Annahme, dass ein Alignment von S_1 und S_2 mit Hilfe von Mutationen erklärbar ist.

Die Wahrscheinlichkeit eines Alignments von S_1 und S_2 im Mutationsmodell ist:

$$P(S_1, S_2 \mid M) = \prod_{i=1}^{n} p_{x_i} p_{x_i, y_i}$$

wobei $p_{a,b}$ die Wahrscheinlichkeit ist, dass eine Aminosäure a zu einer Aminosäure b mutiert. (Wir nehmen an $p_{a,b} = p_{b,a}$.)

PAM Matrizen

Wir vergleichen nun beide Modelle

$$\frac{P(S_1, S_2 \mid M)}{P(S_1, S_2 \mid R)} = \prod_{i=1}^{n} \frac{p_{x_i} p_{x_i, y_i}}{p_{x_i} p_{y_j}}$$

Ist dieser Wert größer als 1, so beschreibt das Mutationsmodell das Alignment besser als das Zufallsmodell; sonst ist es umgekehrt.

Logarithmieren ergibt ein additives Maß:

$$score(S_1, S_2) = \sum_{i=1}^{n} score(x_i, y_i) = \sum_{i=1}^{n} \log \frac{p_{x_i} p_{x_i, y_i}}{p_{x_i} p_{y_j}}$$

$$\text{also } score(a, b) = \log \frac{p_{a,b}}{p_b}$$

PAM Matrizen

Dayhoff et al. erhielten eine Menge von sogenannten akzeptierten Mutationen (accepted point mutations) aus Gruppen von eng verwandten (höchstens 15% verschiedenen) Proteinen

$p_b :=$ relative Häufigkeit von b in allen Sequenzen

Wie erhält man $p_{a,b}$?

Dayhoff et al. stellten dazu die Matrix A der akzeptierten Mutationen auf.

Matrix A der akzeptierten Mutationen

Xpdf: BioinformaticsMolMed.pdf

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y
Ala	A																			
Arg	R	30																		
Asn	N	109	17																	
Asp	D	154	0	532																
Cys	C	33	10	0	0															
Gln	Q	93	120	50	76	0														
Glu	E	266	0	94	831	0	422													
Gly	G	579	10	156	162	10	30	112												
His	H	21	103	226	43	10	243	23	10											
Ile	I	66	30	36	13	17	8	35	0	3										
Leu	L	95	17	37	0	0	75	15	17	40	253									
Lys	K	57	477	322	85	0	147	104	60	23	43	39								
Met	M	29	17	0	0	0	20	7	7	0	57	207	90							
Phe	F	20	7	7	0	0	0	0	17	20	90	167	0	17						
Pro	P	345	67	27	10	10	93	40	49	50	7	43	43	4	7					
Ser	S	772	137	432	98	117	47	86	450	26	20	32	168	20	40	269				
Thr	T	590	20	169	57	10	37	31	50	14	129	52	200	28	10	73	696			
Trp	W	0	27	3	0	0	0	0	0	3	0	13	0	0	10	0	17	0		
Tyr	Y	20	3	36	0	30	0	10	0	40	13	23	10	0	260	0	22	23	6	
Val	V	365	20	13	17	33	27	37	97	30	661	303	17	77	10	50	43	186	0	17

PAM Matrizen

Dayhoff et al. berechneten daraus die Matrix M der Mutationswahrscheinlichkeiten für das Zeitintervall, in dem 1% aller Aminosäuren mutieren.

Dabei wird die Evolution durch einen Markov-Prozess erster Ordnung modelliert.

Bei einem Markov-Prozess erster Ordnung hängt die Zukunft des Systems nur von der Gegenwart (dem aktuellen Zustand) und nicht von der Vergangenheit ab.

Beispiel eines Markov-Prozesses

	Regen	**S**onne	**W**olken
R	0,5	0,1	0,4
S	0,2	0,6	0,2
W	0,3	0,3	0,4

Matrix der Übergangswahrscheinlichkeiten für eine einfaches Wettermodell.

Beispiel eines Markov-Prozesses

Xpdf: script.pdf

1.2 Markov-Ketten

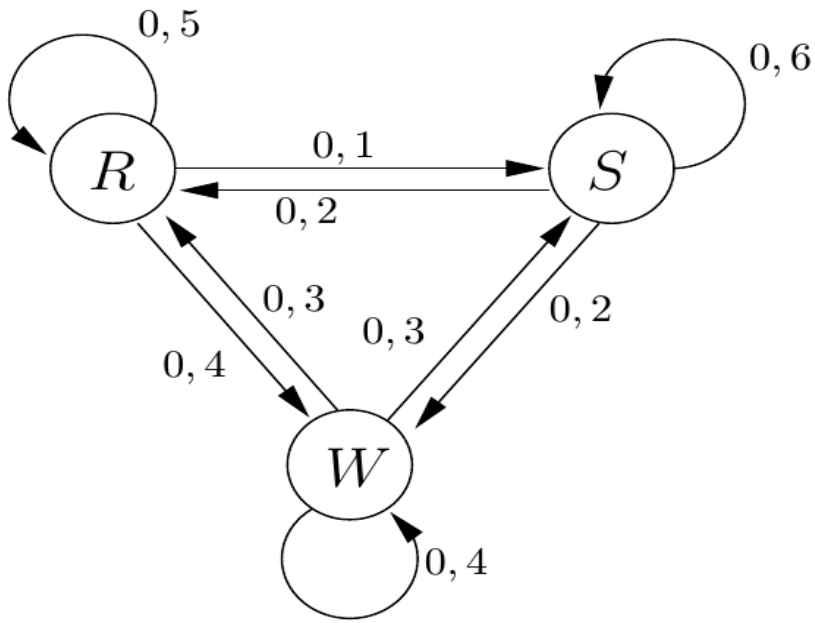


Abbildung 1.1: Eine Markov-Kette für das Wettermodell

PAM1: Matrix der Mutationswahrscheinlichkeiten

dayhoff1978.pdf

File Edit View Go Bookmarks Help

348 (4 of 8) 180,00%

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A Ala	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R Arg	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N Asn	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D Asp	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C Cys	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q Gln	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E Glu	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G Gly	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H His	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I Ile	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L Leu	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K Lys	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M Met	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F Phe	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P Pro	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S Ser	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T Thr	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9

PAM Matrizen

Bei einem Markov-Prozess kann man durch Kenntnis einer begrenzten Vorgeschichte ebenso gute Prognosen über die zukünftige Entwicklung machen wie bei Kenntnis der gesamten Vorgeschichte des Prozesses.

Die Mutationswahrscheinlichkeiten nach t Zeitintervallen (sodass $t\%$ aller Aminosäuren mutieren) ist dann die Matrix M^t (die Matrix M wird t mal mit sich selbst multipliziert). Dies ist eine grundlegende Eigenschaft des Markov-Prozesses.

PAM250: Matrix der Mutationswahrscheinlichkeiten

dayhoff1978.pdf

File Edit View Go Bookmarks Help

350 (6 of 8) 200%

REPLACEMENT AMINO ACID		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
A	Ala	13	6	9	9	5	8	9	12	5	8	6	7	7	4	11	11	11	2	4	9
R	Arg	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	Asn	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	Asp	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	Cys	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	Gln	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	Glu	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	Gly	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	His	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	Ile	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	Leu	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	Lys	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	Met	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	Phe	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	Pro	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	Ser	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	Thr	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6

PAM Matrizen

Die Log-Odds Form der PAM250 erhält man mit:

$$score(a, b) = \log \frac{p_{a,b}}{p_b} = \log \frac{M^{250}_{a,b}}{p_b}$$

PAM250: Log-Odds Form

Xpdf: BioinformaticsMolMed.pdf

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	Ala	.18																			
R	Arg	-.15	.61																		
N	Asn	.02	0	.20																	
D	Asp	.03	-.13	.21	.39																
C	Cys	-.20	-.36	-.36	-.51	1.19															
Q	Gln	-.04	.13	.08	.16	-.54	.40														
E	Glu	.03	-.11	.14	.34	-.53	.25	.38													
G	Gly	.13	-.26	.03	.06	-.34	-.53	.25	.38												
H	His	-.14	.16	.16	.07	-.34	.29	.07	-.21	.65											
I	Ile	-.05	-.20	-.18	-.24	-.23	-.20	-.20	-.26	-.24	.45										
L	Leu	-.19	-.30	-.29	-.40	-.60	-.18	-.34	-.41	-.21	.24	.59									
K	Lys	-.12	.34	.10	.01	-.54	.07	-.01	-.17	0	-.19	-.29	.47								
M	Met	-.11	-.04	-.17	-.26	-.52	-.10	-.21	-.28	-.21	.22	.37	.04	.64							
F	Phe	-.35	-.45	-.35	-.56	-.43	-.47	-.54	-.48	-.18	.10	.18	-.53	.02	.91						
P	Pro	.11	-.02	-.05	-.10	-.28	.02	-.06	-.05	-.02	-.20	-.25	-.11	-.21	-.46	.59					
S	Ser	.11	-.03	.07	.03	0	-.05	0	.11	-.08	-.14	-.28	-.02	-.16	-.32	.09	.16				
T	Thr	.12	-.09	.04	-.01	-.22	-.08	-.04	0	-.13	.01	-.17	0	-.06	-.31	.03	.13	.26			
W	Trp	-.58	.22	-.42	-.68	-.78	-.48	-.70	-.70	-.28	-.51	-.18	-.35	-.42	.04	-.56	-.25	-.52	1.73		
Y	Tyr	-.35	-.42	-.21	-.43	.03	-.40	-.43	-.52	-.01	-.09	-.09	-.44	-.24	.70	-.49	-.28	-.27	-.02	1.01	
V	Val	.02	-.25	-.17	-.21	-.19	-.19	-.18	-.14	-.22	.37	.19	-.24	.18	-.12	-.12	-.10	.03	-.62	-.25	.43

PAM Matrizen

Die finale Form der PAM250 Matrix erhält man, indem jeder Eintrag mit einem Faktor 10 multipliziert und danach gerundet wird (dies dient lediglich der besseren Lesbarkeit).

In der finalen Matrix ist die Ordnung der Aminosäuren so gewählt, dass man die Gruppen von chemisch ähnlichen Aminosäuren klar erkennt:

- STPAG: kleine hydrophile Aminosäuren
- NDEQ: saure und amidierte Aminosäuren
- HRK: basische Aminosäuren
- MILV: kleine hydrophobe Aminosäuren
- FYW: aromatische Aminosäuren

PAM250: Finale Form

Xpdf: BioinformaticsMolMed.pdf

Cys	C	12																
Ser	S	0	2															
Thr	T	-2	1	3														
Pro	P	-3	1	0	6													
Ala	A	-2	1	1	1	2												
Gly	G	-3	1	0	-1	1	5											
Asn	N	-4	1	0	-1	0	0	2										
Asp	D	-5	0	0	-1	0	1	2	4									
Glu	E	-5	0	0	-1	0	0	1	3	4								
Gln	Q	-5	-1	-1	0	0	-1	1	2	2	4							
His	H	-3	-1	-1	0	-1	-2	2	1	1	3	6						
Arg	R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6					
Lys	K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5				
Met	M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6			
Ile	I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5		
Leu	L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6	
Val	V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4
Phe	F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9	
Tyr	Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10
Trp	W	8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	-3	-3	-4	-5	-2	-6	0	0	17

PAM Matrizen

Xpdf: dayhoff1978.pdf

Correspondence between Observed Differences and the Evolutionary Distance

Observed Percent Difference	Evolutionary Distance in PAMs
1	1
5	5
10	11
15	17
20	23
25	30
30	38
35	47
40	56
45	67
50	80
55	94
60	112
65	133

Multiples Alignment

Ein multiples Alignment ist ein Alignment von mehr als zwei Sequenzen, z.B.

```
N - F L S
N - F - S
N K Y L S
N - Y L S
```

Formale Definition

Ein globales multiples Alignment von m Strings $S^1, \ldots, S^m$ über dem Alphabet Σ ist eine $(m \times N)$ Matrix A mit:

1. $A(i,j) \in \Sigma \cup \{-\}$, wobei $-$ ein spezielles Lückensymbol (gap symbol) ist, das nicht in Σ vorkommt.
2. Wenn man alle Lückensymbole entfernt, dann stimmen die k-te Zeile von A und der String S^k überein.
3. Keine Spalte der Matrix A besteht ausschließlich aus Lückensymbolen.

Man beachte, dass Bedingung (2) $\max_{1 \leq k \leq m}\{n_k\} \leq N$ impliziert und Bedingung (3) $N \leq \sum_{k=1}^{m} n_k$ impliziert.

Bewertung

Bewertung eines multiplen Alignments A:

$$score(A) = \sum_{j=1}^{N} score(A(1,j), A(2,j), \ldots, A(m,j))$$

wobei *score* eine Funktion ist, die für alle Kombinationen von m Symbolen (inklusive dem Gap Symbol $-$) eine Bewertung liefert. Für $m = 2$ Aminosäuresequenzen wird solch eine Bewertungsfunktion durch eine Substitutionsmatrix (PAM oder BLOSUM) und eine Gap-Kostenfunktion definiert.

Für $m > 2$ Aminosäuresequenzen gibt es jedoch keine!

Sum-of-pairs Score

Der sum-of-pairs score (SP-Score) eines multiplen Alignments A der Strings $S^1, \ldots, S^m$ ist definiert durch:

$$score_{SP}(A) = \sum_{1 \leq i < j \leq m} score(\pi_{i,j}(A))$$

wobei

- *score* eine Bewertungsfunktion von paarweisen Alignments ist, die durch die Vereinbarung $score(-,-) = 0$ erweitert wird.
- $\pi_{i,j}(A)$ das paarweise Alignment ist, das durch die Projektion auf die i-te und j-te Zeile von A entsteht.

Im Folgenden verwenden wir immer den sum-of-pairs score.

Beispiel

Der SP-Score unseres Beispielalignments A^* bzgl. der PAM250 Matrix und den Gap-Kosten $g = 8$ erechnet sich wie folgt:

$$\begin{aligned}
score(\pi_{1,2}(A)) &= 2+0+9-8+2 = 5 \\
score(\pi_{1,3}(A)) &= 2-8+7+6+2 = 9 \\
score(\pi_{1,4}(A)) &= 2+0+7+6+2 = 17 \\
score(\pi_{2,3}(A)) &= 2-8+7-8+2 = -5 \\
score(\pi_{2,4}(A)) &= 2+0+7-8+2 = 3 \\
score(\pi_{3,4}(A)) &= 2-8+10+6+2 = 12
\end{aligned}$$

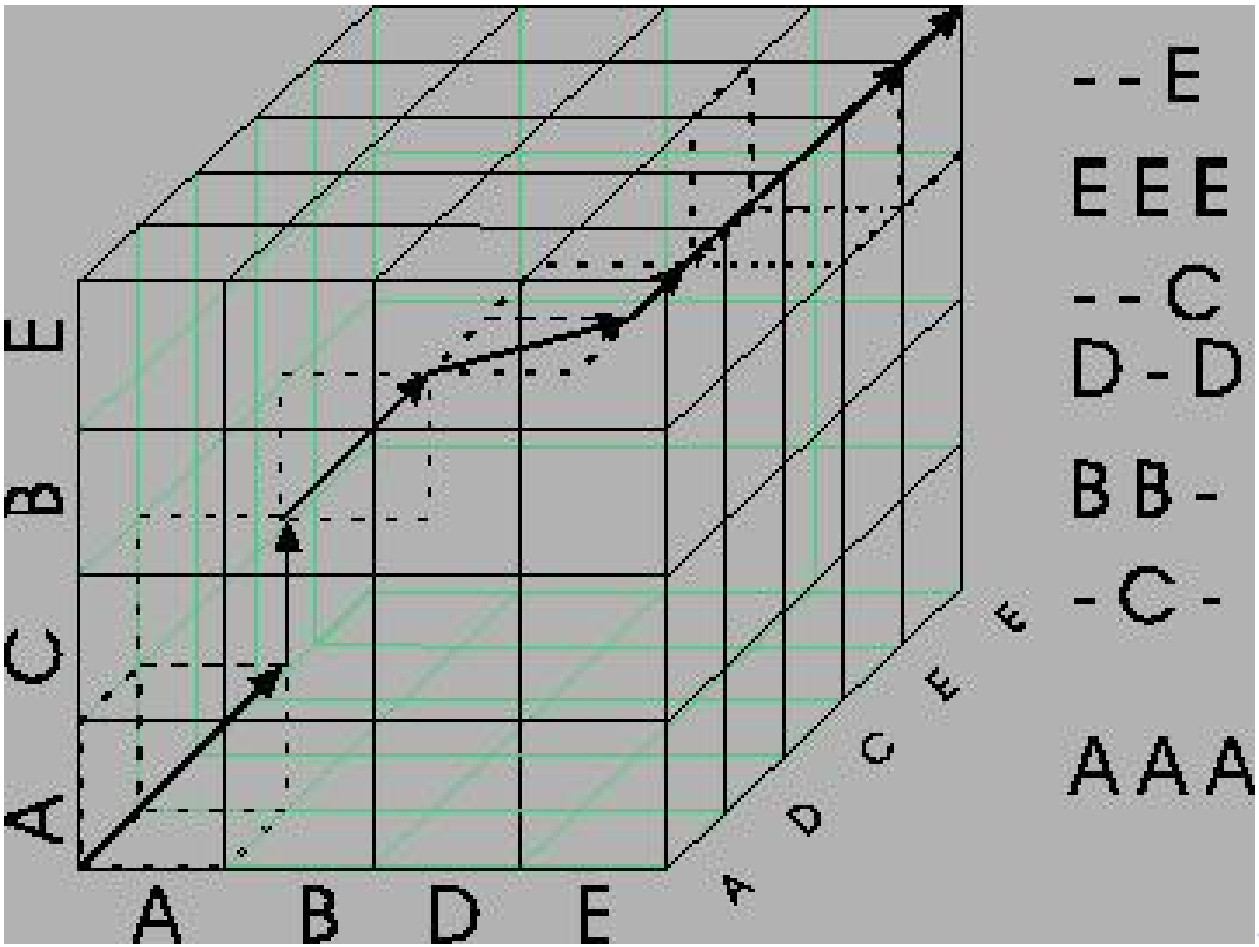
Also $score_{SP}(A) = 5+9+17-5+3+12 = 41$.

Dynamic Programming Algorithmus

Es ist möglich die Dynamic Programming Algorithmen zur Berechnung von optimalen paarweisen Alignments auf mehr als zwei Sequenzen zu erweitern, z.B. den Needleman & Wunsch Algorithmus für $r = 3$ Sequenzen:

$$S(i,j,k) = \max \begin{cases} S(i-1,j-1,k-1) + score_{SP}(S^1[i], S^2[j], S^3[k]) \\ \\ S(i-1,j-1,k) + score_{SP}(S^1[i], S^2[j], -) \\ S(i,j-1,k-1) + score_{SP}(-, S^2[j], S^3[k]) \\ S(i-1,j,k-1) + score_{SP}(S^1[i], -, S^3[k]) \\ \\ S(i-1,j,k) + score_{SP}(S^1[i], -, -) \\ S(i,j-1,k) + score_{SP}(-, S^2[j], -) \\ S(i,j,k-1) + score_{SP}(-, -, S^3[k]) \end{cases}$$

Dynamic Programming Algorithmus



Multiples Alignment

Die Zeitkomplexität der Dynamic Programming Algorithmen zur Berechnung von multiplen Alignments ist allerdings exponentiell in der Anzahl der Sequenzen.

Schlimmer noch: Die Berechnung eines optimalen multiplen Alignments bzgl. des sum-of-pairs score wurde als NP-vollständig nachgewiesen.

Multiples Alignment

Es gibt prinzipiell drei Möglichkeiten weiter vorzugehen:

1. Versuche einen Algorithmus zu entwickeln, der den "Suchraum" zur Berechnung eines multiplen Alignments einschränkt, ohne auf eine optimale Lösung zu verzichten.
2. Versuche einen effizienten Approximationsalgorithmus zu entwickeln. Dieser liefert dann eine approximative Lösung, die sich nur um einen (kleinen) konstanten Faktor von der optimalen Lösung unterscheidet.
3. Versuche einen effizienten heuristischen Algorithmus zu entwickeln, der in der Praxis gute Ergebnisse liefert, jedoch keine optimale Lösung garantiert.

Einschränkung des Suchraums

Sei δ eine Kostenfunktion und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Es gilt für jedes weitere Alignment A^{heur}:

$$\begin{aligned}
\delta(A^{heur}) &\geq \delta(A^{opt}) \\
&= \sum_{k<l} \delta(\pi_{k,l}(A^{opt})) \\
&= \delta(\pi_{p,q}(A^{opt})) + \sum_{k<l,(k,l)\neq(p,q)} \delta(\pi_{k,l}(A^{opt})) \\
&\geq \delta(\pi_{p,q}(A^{opt})) + \sum_{k<l,(k,l)\neq(p,q)} edist_\delta(S^k, S^l)
\end{aligned}$$

für alle Paare (p, q), $1 \leq p < q \leq m$.

Einschränkung des Suchraums

Sei δ eine Kostenfunktion und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Es gilt für jedes weitere Alignment A^{heur}:

$$\delta(A^{heur}) \quad \geq \quad \delta(\pi_{p,q}(A^{opt})) + \sum_{k<l,(k,l)\neq(p,q)} edist_\delta(S^k, S^l)$$

für alle Paare (p, q), $1 \leq p < q \leq m$. Also ist

$$U_{p,q} := \delta(A^{heur}) - \sum_{k<l,(k,l)\neq(p,q)} edist_\delta(S^k, S^l)$$

eine obere Schranke für $\delta(\pi_{p,q}(A^{opt}))$, d.h. $U_{p,q} \geq \delta(\pi_{p,q}(A^{opt}))$.

Einschränkung des Suchraums

Für jedes Paar (p, q), definiere für $1 \leq i \leq n_p$ und $1 \leq j \leq n_q$

$$B_{p,q}(i,j) = edist_\delta(S^p[1..i], S^q[1..j]) + edist_\delta(S^p[i+1..n_p], S^q[j+1..n_q])$$

d.h. $B_{p,q}(i,j)$ ist das Minimum der Kosten aller Pfade im Editiergraphen von S^p und S^q, die über den Knoten (i,j) laufen.

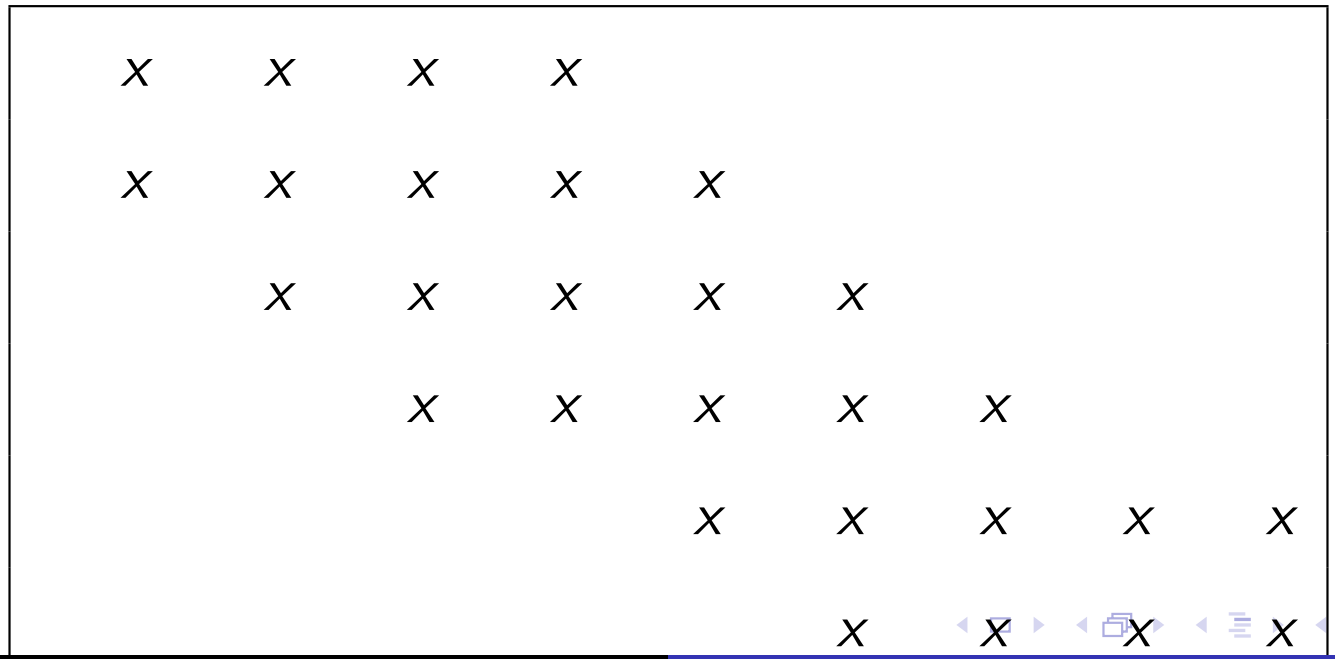
$B_{p,q}(i,j)$ kann durch die Kombination der *dynamic programming* Matrizen

- $D_{p,q}$ (Berechnung der Editierdistanz von S^p und S^q) und
- $D^{rev}_{p,q}$ (Berechnung der Editierdistanz von $(S^p)^{rev}$ und $(S^q)^{rev}$)

in $O(n_p n_q)$ Zeit berechnet werden.

Methode von Carillo & Lipman

In der Methode von Carillo und Lipman berechnet der m-dimensionale *dynamic programming* Algorithmus nur Werte für die Knoten $(i_1, i_2, \ldots, i_m)$ im Editiergraphen von $S^1, \ldots, S^m$ für die gilt: $B_{p,q}(i_p, i_q) \leq U_{p,q}$ für alle Paare (p, q).



Methode von Carillo & Lipman

Warum kann man die anderen Knoten vernachlässigen?
Nehmen wir an, es gibt einen Pfad im Editiergraphen eines multiplen Alignments A der Strings $S^1, \ldots, S^m$, der über einen Knoten $(i_1, i_2, \ldots, i_m)$ verläuft, für den ein Paar (k, l) existiert mit $B_{k,l}(i_k, i_l) > U_{k,l}$. Dann folgt aus $\delta(\pi_{k,l}(A)) \geq B_{k,l}(i_k, i_l)$, dass $\delta(\pi_{k,l}(A)) > U_{k,l}$ gilt. Gemäß obiger Diskussion kann das Alignment A dann aber nicht optimal sein.
Natürlich muss man noch einen optimalen Pfad vom Knoten $(0, \ldots, 0)$ zum Knoten $(n_1, \ldots, n_m)$ in dem reduzierten Editiergraphen finden. Dies kann z.B. durch Dijkstras Algorithmus zur Bestimmung kürzester Wege oder durch den so genannten A^*-Algorithmus erfolgen.

Methode von Carillo & Lipman

Die Methode von Carillo & Lipman ist in dem Programm MSA implementiert. MSA kann in akzeptabler Zeit ca. acht Aminosäuresequenzen mit durchschnittlicher Proteinlänge alignieren.
Eine andere Implementierung benutzt eine Heuristik, um den Suchraum weiter einzuschränken. Dort wird ein weiterer Parameter $\epsilon_{p,q}$ eingeführt, und ein Knoten im Editiergraphen wird als irrelevant betrachtet, wenn $B_{p,q}(i_p, i_q) \leq U_{p,q} - \epsilon_{p,q}$ gilt. Man beachte, dass diese Heuristik kein optimales Alignment garantiert!

Ein 2-Approximationsalgorithmus

Es seien m Strings $S^1, \ldots, S^m$ gegeben. Der *Center-String* dieser Strings ist derjenige String, dessen Distanz zu den restlichen Strings minimal ist. Genauer gesagt minimiert der *Center-String* S^c, $1 \leq c \leq m$, die Summe

$$\sum_{i=1}^{m} edist_\delta(S^c, S^i)$$

Das *Center-Star-Alignment* A^c wird dann durch die Kombination der paarweisen optimalen Alignments des *Center-Strings* mit den restlichen Strings konstruiert.

Ein 2-Approximationsalgorithmus

Wir zeigen nun, dass

$$\delta_{SP}(A^c) \leq \frac{2(m-1)}{m} \delta_{SP}(A^{opt})$$

gilt.
D.h. die *Center-Star-Methode* liefert einen 2-Approximationsalgorithmus für das globale multiple Alignmentproblem.

Ein 2-Approximationsalgorithmus

Lemma: Sei S^c der *Center-String* und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Dann gilt

$$\frac{m}{2}\sum_{i=1}^{m} edist_\delta(S^i, S^c) = \frac{1}{2}\underbrace{(\sum_{i=1}^{m} edist_\delta(S^i, S^c) + \cdots + \sum_{i=1}^{m} edist_\delta(S^i, S^c))}_{m-mal}$$

$$\leq \frac{1}{2}(\sum_{i=1}^{m} edist_\delta(S^i, S^1) + \cdots + \sum_{i=1}^{m} edist_\delta(S^i, S^m))$$

$$= \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{m} edist_\delta(S^i, S^j)$$

Ein 2-Approximationsalgorithmus

Lemma: Sei S^c der *Center-String* und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Dann gilt

$$\begin{aligned}
\frac{m}{2} \sum_{i=1}^{m} edist_\delta(S^i, S^c) &\leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} edist_\delta(S^i, S^j) \\
&\leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{i,j}(A^{opt})) \\
&= \sum_{1 \leq i < j \leq m} \delta(\pi_{i,j}(A^{opt})) \\
&= \delta_{SP}(A^{opt})
\end{aligned}$$

Ein 2-Approximationsalgorithmus

Es seien S^c der *Center-String*, A^c das *Center-Star-Alignment* und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Dann gilt

$$\begin{aligned}
\delta_{SP}(A^c) &= \sum_{1 \le i < j \le m} \delta(\pi_{i,j}(A^c)) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{i,j}(A^c)) \\
&\le \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} (\delta(\pi_{i,c}(A^c)) + \delta(\pi_{c,j}(A^c))) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} (\delta(\pi_{i,c}(A^c)) + \delta(\pi_{j,c}(A^c)))
\end{aligned}$$

Ein 2-Approximationsalgorithmus

Es seien S^c der *Center-String*, A^c das *Center-Star-Alignment* und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Dann gilt

$$\begin{aligned}
\delta_{SP}(A^c) &\leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} (\delta(\pi_{i,c}(A^c)) + \delta(\pi_{j,c}(A^c))) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{i,c}(A^c)) + \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{j,c}(A^c)) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{i,c}(A^c)) + \frac{1}{2} \sum_{j=1}^{m} \sum_{i \neq j} \delta(\pi_{j,c}(A^c))
\end{aligned}$$

Ein 2-Approximationsalgorithmus

Es seien S^c der *Center-String*, A^c das *Center-Star-Alignment* und A^{opt} ein optimales Alignment der Strings $S^1, \ldots, S^m$. Dann gilt

$$\begin{aligned}
\delta_{SP}(A^c) &\leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{i,c}(A^c)) + \frac{1}{2} \sum_{j=1}^{m} \sum_{i \neq j} \delta(\pi_{j,c}(A^c)) \\
&= (m-1) \sum_{i=1}^{m} \delta(\pi_{i,c}(A^c)) \\
&= (m-1) \sum_{i=1}^{m} edist_{\delta}(S^i, S^c) \\
&\leq \frac{2(m-1)}{m} \, \delta_{SP}(A^{opt}) \qquad (vgl.\ obiges\ Lemma)
\end{aligned}$$

Ein 2-Approximationsalgorithmus

In der Komplexitätsanalyse der *Center-Star-Methode* nehmen wir zur Vereinfachung an, dass alle Strings ungefähr die gleiche Länge n haben. Um den *Center-String* S^c zu bestimmen, müssen $\binom{m}{2}$ paarweise Editierdistanzen berechnet werden.
Dies kostet $O(m^2 n^2)$ Zeit und $O(m^2 + n)$ Platz (wenn alle $O(m^2)$ Editierdistanzen gespeichert werden).
Die Berechnung der $m-1$ vielen optimalen paarweisen Alignments von S^c mit den restlichen Strings erfordert $O(mn^2)$ Zeit und $O(mn)$ Platz (wenn Hirschbergs Algorithmus benutzt wird).
Die Kombination der paarweisen optimalen Alignments zu einem multiplen Alignment erfordert Zeit, die proportional zur Größe des multiplen Alignments ist, also $O(mn)$ Zeit.
Die Gesamtkomplexität ist damit $O(m^2 n^2)$ Zeit und $O(mn)$ Platz.

Progressive Alignments (Heuristiken)

Progressive Alignmentverfahren gehen nach folgendem Schema vor:

1. Für jedes Paar p, q mit $1 \leq p < q \leq r$ berechne eine "approximative evolutionäre Distanz".
2. Ausgehend von den paarweisen Distanzen, berechne einen "approximativen phylogenetischen Baum" (guide tree).
3. Aligniere die Sequenzen sukzessive in der durch den Baum vorgegebenen Ordnung.

Progressive Alignmentverfahren sind heuristisch: Sie optimieren keine Bewertungsfunktion.

Dafür sind sie schnell und liefern in vielen Fällen ein vernünftiges Alignment.

Feng-Doolittle Methode

(1) Für jedes Paar p, q mit $1 \leq p < q \leq r$ berechne die Distanz

$$D_{p,q} = -\log S_{eff} = -\log \frac{S_{pair} - S_{rand}}{S_{aver} - S_{rand}}$$

wobei S_{pair} der Score eines optimalen paarweisen Alignments der Sequenzen S^p und S^q ist, S_{aver} der Mittelwert der Scores von Alignments der beiden Sequenzen S^p und S^q mit sich selbst und S_{rand} ist der erwartete Score eines Alignments zweier "zufälliger" Sequenzen gleicher Länge.

Sequenzen sind identisch: $S_{pair} = S_{aver} \Rightarrow S_{eff} = 1 \Rightarrow D_{p,q} = 0$.
Keine Ähnlichkeit: $S_{pair} = S_{rand} \Rightarrow S_{eff} = 0 \Rightarrow D_{p,q} = \infty$.

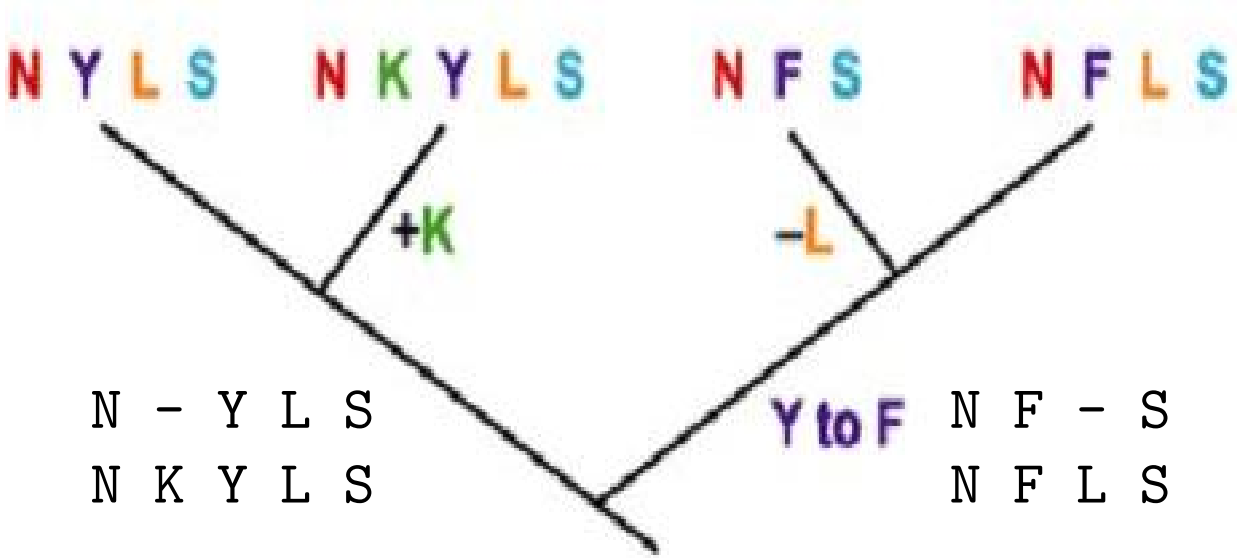
Feng-Doolittle Methode

(2) Ausgehend von den paarweisen Distanzen, berechne einen guide tree mit Hilfe des Clustering-Verfahrens von Fitch & Margoliash (vgl. UPGMA Clustering-Verfahren im Kapitel Phylogenetische Rekonstruktion).

(3) Aligniere die Sequenzen sukzessive in der durch den guide tree vorgegebenen Ordnung.

Im folgenden Beispiel werden gemäß dem guide tree zuerst die Sequenzen `N Y L S` und `N K Y L S` sowie die Sequenzen `N F S` und `N F L S` mit Hilfe des Needleman & Wunsch Algorithmus aligniert. Die beiden paarweisen Alignments werden dann zu einem multiplen Alignment vereint (alignment of alignments).

Alignment mit einem Guide Tree



Feng-Doolittle Methode

Nachdem ein Alignment aufgestellt wurde, werden alle Gap-Symbole durch den "neutralen" Buchstaben `X` ersetzt (`X` kommt im Aminosäurenalphabet nicht vor).

Feng und Doolittle nennen diese Regel "once a gap, always a gap."

`X` kann ohne Kosten mit jedem anderen Buchstaben (inklusive des Gap-Symbols) aligniert werden.

Dies stellt sicher, dass ein "alignment of alignments" konsistent ist.

Wie werden nun Alignments aligniert?

Feng-Doolittle Methode

Eine Sequenz wird mit einer Gruppe von Sequenzen (einem Alignment) aligniert, indem sie mit allen Sequenzen aus der Gruppe paarweise aligniert wird. Das beste paarweise Alignment legt dann fest, wie die einzelne Sequenz zur Gruppe aligniert wird.

Eine Gruppe von Sequenzen (ein Alignment) wird mit einer anderen Gruppe von Sequenzen (einem anderen Alignment) aligniert, indem alle möglichen paarweisen Alignments von Sequenzen aus verschiedenen Gruppen berechnet werden. Das beste paarweise Alignment legt dann fest, wie die Gruppen aligniert werden.

CLUSTALW

CLUSTALW ist ebenfalls ein progressives Alignmentverfahren.
Es ist eines der am meisten genutzten Alignmentprogramme.

1. Für jedes Paar von Sequenzen wird eine Distanz mit Hilfe der Kimura Korrektur berechnet.
2. Ausgehend von den paarweisen Distanzen, wird ein guide tree mit Hilfe des neighbor-joining Verfahrens von Saitou & Nei berechnet.
3. Die Sequenzen werden sukzessive in der durch den guide tree vorgegebenen Ordnung aligniert, wobei Profile von Alignments benutzt werden.

CLUSTALW benutzt eine Fülle von Heuristiken, die z.B. in Durbin et al. Kapitel 6 diskutiert werden.

FASTA

FASTA (FAST Alignments) ist eine Heuristik, um signifikante Übereinstimmungen zwischen einer Anfragesequenz q und einer Datenbank d zu finden.

Die verfolgte Strategie besteht darin, die "besten" Diagonalen im Dotplot bzw. in der Dynamic Programming Matrix zu finden.

Im ersten Schritt werden mit Hash-Methoden alle hot-spots, d.h. alle exakten Matches der Länge k gesucht (Voreinstellung: $k = 6$ für DNA Sequenzen bzw. $k = 2$ für Aminosäuresequenzen).

FASTA

Ein hot-spot wird repräsentiert durch das Paar (i, j), seinen Anfangspositionen in q und d.

Ein hot-spot (i, j) liegt auf der Diagonalen $i - j$ im Dotplot.

Die Hauptdiagonale hat die Nummer 0, die darüberliegenden Diagonalen haben positive Nummern, die darunterliegenden negative.

Ein Diagonal Run ist eine Folge von hot-spots, die auf derselben Diagonale liegen und "eng" beieinanderliegen (bei der Wahl $k = 2$ für Proteine darf der Abstand höchstens 16 sein).

FASTA

0	1	2	3	4	5	6	7	8	9	
-1		H	E	I	T	E	I	T	E	I
-2	F									
-3	R									
-4	E		↘			↘			↘	
-5	I			↘			↘			↘
-6	H	↘								
-7	E		↘			↘			↘	
-8	I			↘			↘			↘
	T				↘			↘		

FASTA

Der Algorithmus bestimmt die besten Diagonal Runs.

Jeder dieser Diagonal Runs entspricht einem lokalen Alignment ohne Lücken, welches durch eine Substitutionsmatrix bewertet wird (z.B. PAM250 Matrix).

Der Algorithmus bestimmt nun die besten lokalen Alignments.

Dann wird versucht die lokalen Alignments durch die Einführung von Lücken zu größeren lokalen Alignments zu kombinieren.

Schließlich wird ein beschränkter (banded) Smith-Waterman Algorithmus benutzt, um ein optimales lokales Alignment in den oben bestimmten Regionen zu finden.

BLAST

BLAST (Basic Local Alignment Search Tool) ist das am meisten benutzte Programm zur Suche einer Anfragesequenz q in einer Datenbank d.

Proteine: Zuerst wird die Menge M aller k langen Aminosäuresequenzen bestimmt, deren Ähnlichkeit (bzgl. der zu Grunde gelegten Substitutionsmatrix, z.B. BLOSUM62) zu einem der k langen Teilwörter von q einen Schwellwert T überschreitet (Voreinstellung: $k = 3$).

DNA Sequenzen: Hier besteht M aus der Menge aller k langen Teilwörter von q (Voreinstellung: $k = 11$).

BLAST

Nun wird die Datenbank nach allen exakten Matches mit einer der Sequenzen aus M durchsucht (z.B. mit dem Aho-Corasick Algorithmus). Solche Matches werden Hits genannt.

Seit der BLAST Version 2.0 wird die so genannte two-hit Strategie verwendet. Falls zwei Hits auf derselben Diagonale im Dotplot liegen und höchstens A Positionen auseinanderliegen, werden sie bei der weiteren Suche berücksichtigt. Alle anderen Hits werden verworfen.

BLAST

Jeder Diagonalabschnitt, der durch zwei Hits begrenzt wird, wird bidirektional ausgedehnt, solange bis sich der Score nicht mehr erhöhen lässt. Überschreitet der so erhaltene Score einen Schwellwert S (cutoff score), so spricht man von einem high scoring pair, kurz HSP.

Alle HSPs werden statistisch bewertet und nach ihrer Signifikanz geordnet ausgegeben.

Phylogenetische Rekonstruktion

Gegeben: eine Menge von Spezies (Tier- oder Pflanzenarten).

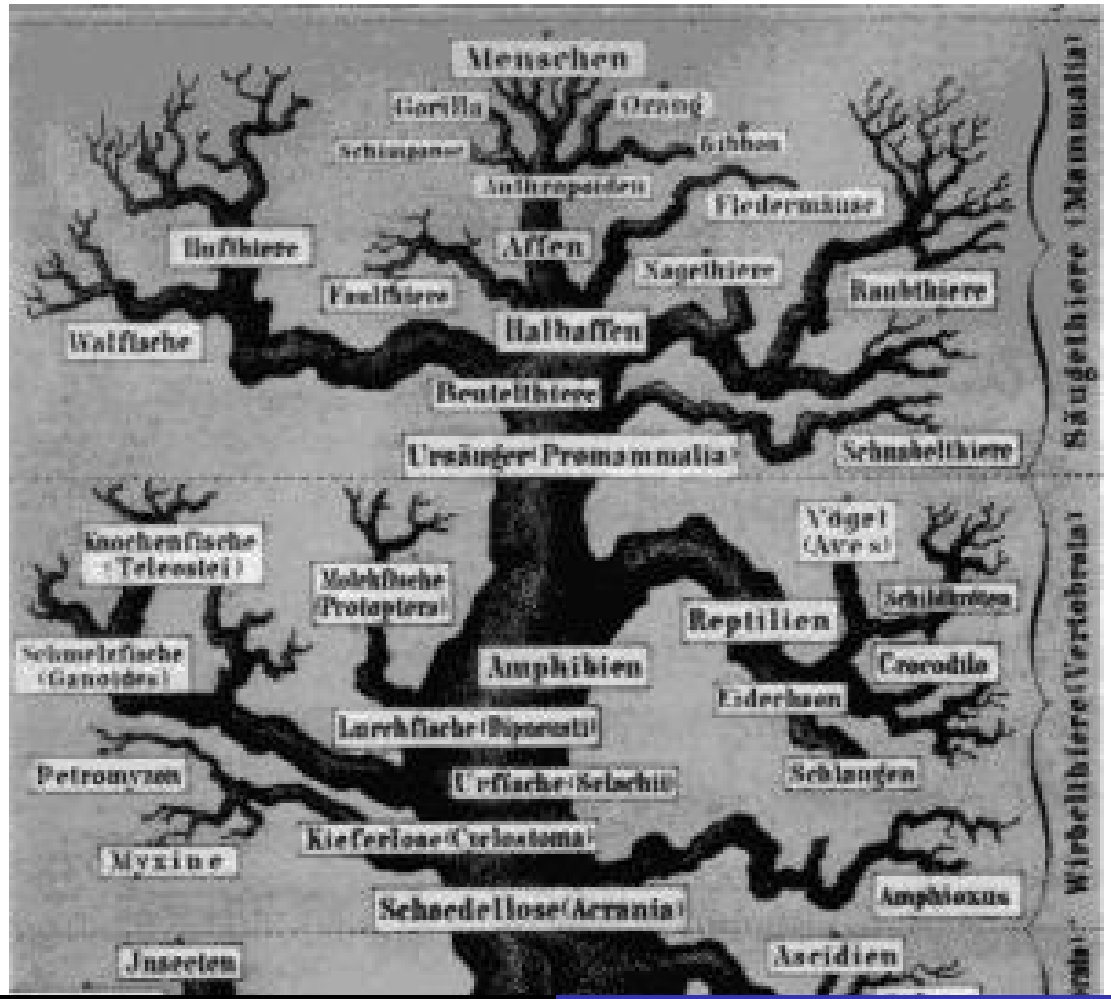
Gesucht: die evolutionäre Beziehung zwischen den Spezies.

Man nimmt an, dass Artenbildung ein verzweigender Prozeß ist: Eine Population von Organismen wird getrennt in zwei Teilpopulationen. Im Laufe der Evolution entwickeln sich diese in zwei verschiedene Spezies, die sich nicht (mehr) kreuzen.

Ziel der phylogenetischen Rekonstruktion: Ein Abstammungsbaum, wobei die Spezies als Blätter im Baum repräsentiert werden und ein gemeinsamer Elternknoten einen gemeinsamen Vorfahren repräsentiert.

Dieser phylogenetische Baum repräsentiert dann die evolutionäre Beziehung zwischen den Spezies.

Haeckels Baum des Lebens (1866)



Phylogenetischer Baum: Formale Definition

Sei $X = \{x_1, \ldots, x_n\}$ eine Menge von Taxa, wobei ein Taxon x_i ein Repräsentant einer Gruppe von Individuen ist.
Ein phylogenetischer Baum (bzgl. X) ist ein Triple $T = (V, E, \lambda)$, wobei (V, E) ein azyklischer zusammenhängender Graph und λ eine Beschriftung der Blätter ist mit den Eigenschaften:

- Jedes Blatt wird mit genau einem Taxon aus X beschriftet.
- Jedes Taxon aus X erscheint genau einmal als Beschriftung.

T ist ein ungewurzelter Baum, wenn jeder innere Knoten einen Verzweigungsgrad ≥ 3 hat.

T ist ein gewurzelter Baum, wenn es genau einen inneren Knoten (die Wurzel) mit Verzweigungsgrad 2 gibt.

Beispiel: Molekulare Anthropologie

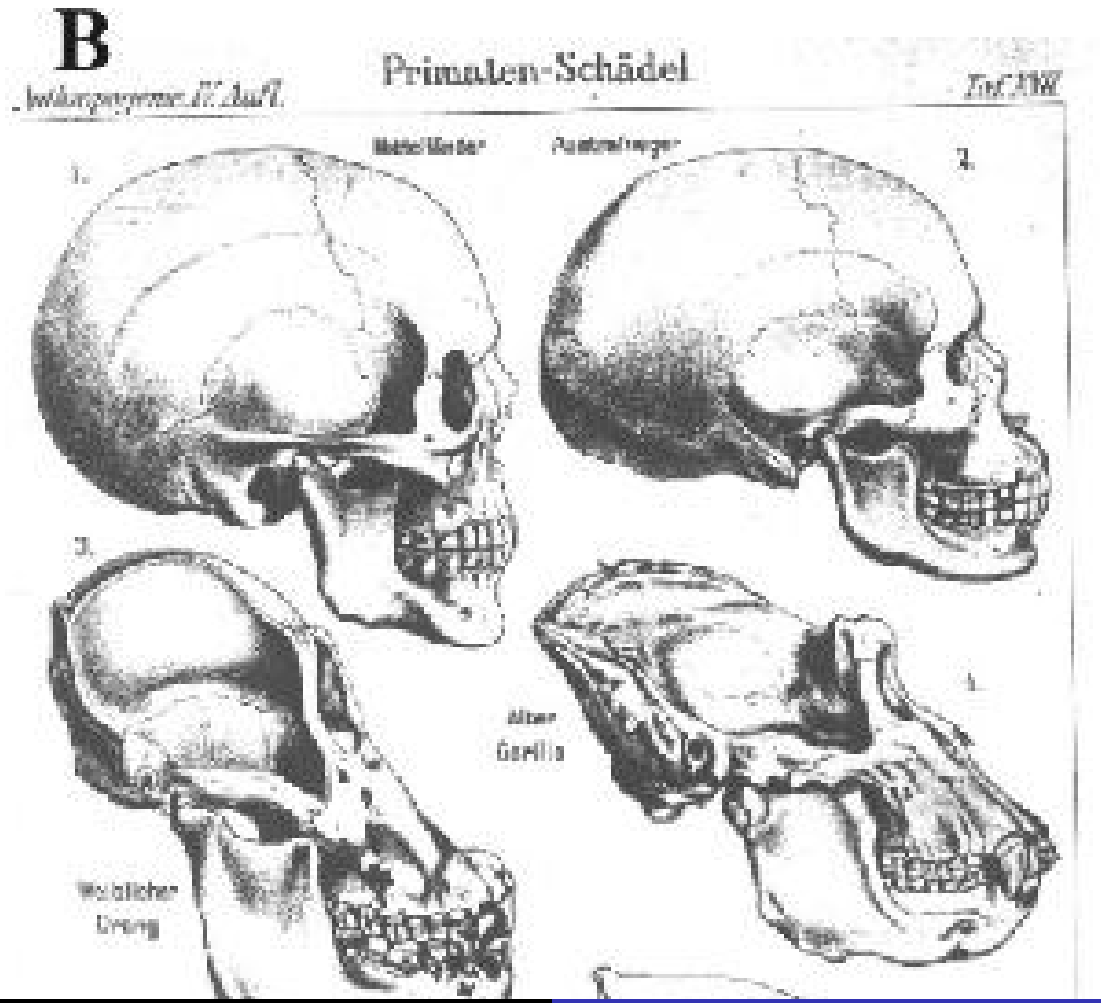
Woher kommen wir?

Seit über 100 Jahren versuchen Anthropologen diese Frage durch die morphologische Analyse von fossilen Funden zu beantworten.

Hypothesen:

1. *Homo erectus* entwickelte sich in Afrika und verbreitete sich vor 1-2 Millionen Jahren in der Welt
2. *Homo sapiens* entwickelte sich aus archaischen Menschen in verschiedenen Regionen der Welt (multi-regionale Entwicklung)

Früher: Morphologische Analyse



Heute: Molekulare Anthropologie

Grundlage der phylogenetischen Rekonstruktion ist der Vergleich von DNA von lebenden Menschen, z.B. mitochondrischer DNA (mtDNA):

1. Mitochondrien haben ein eigenes Genom, das ca. 16500 bp groß ist und 13 Protein kodierende Gene, 22 tRNAs und 2 rRNAs enthält
2. mtDNA hat eine nahezu konstante Rate von Substitutionen (Mutationen)
3. mtDNA wird von der Mutter ererbt
4. mtDNA unterliegt keiner Rekombination

Molekulare Anthropologie

Ingman et al. (Nature 408, S.708-713, 2000) sequenzierten die vollständige mtDNA von 53 Menschen diversen Ursprungs.

Als Distanzmaß wurde folgende "normalisierte" Hammingdistanz zwischen zwei mtDNA-Sequenzen α und β benutzt:

$$d(\alpha, \beta) = \frac{\text{Anzahl der Mismatches zwischen } \alpha \text{ und } \beta}{\text{Länge der mtDNA}}$$

Aus den paarweisen Distanzen wurde mit der neighbor-joining Methode ein phylogenetischer Baum erstellt.

Molekulare Anthropologie

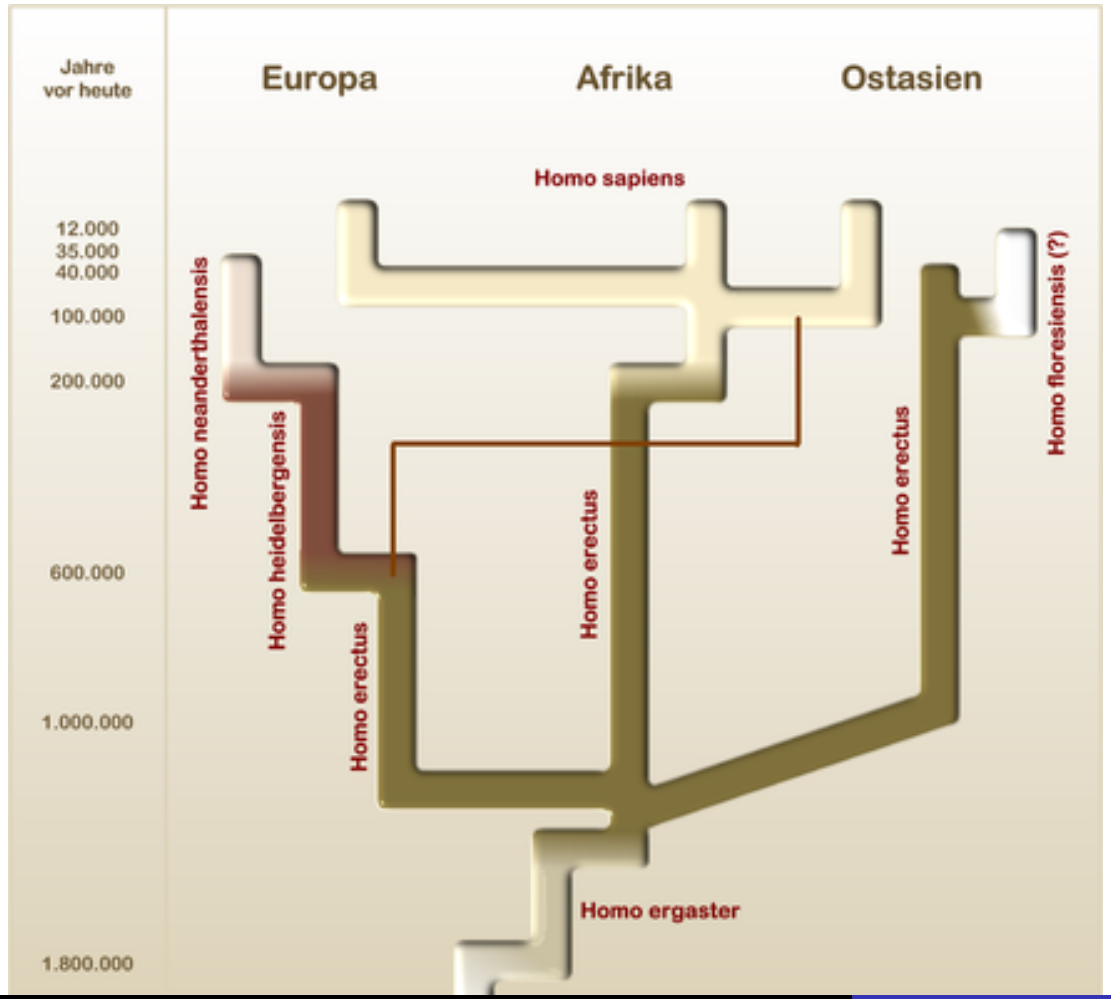
Aus der Mutationsrate (geschätzt: $1.7 \cdot 10^{-8}$ Substitutionen pro Position pro Jahr) ergeben sich folgende Konsequenzen:

1. die letzte gemeinsame Vorfahrin aller 53 Personen lebte vor ca. 171.500 ± 50.000 Jahren in Afrika
2. widerlegt die Hypothese der multi-regionalen Entwicklung (weil die letzte gemeinsame Vorfahrin sonst vor viel längerer Zeit gelebt haben müßte)
3. vor ca. 52.000 ± 27.500 Jahren wanderten die modernen Menschen aus Afrika aus und verdrängten die archaischen Menschen in der gesamten Welt

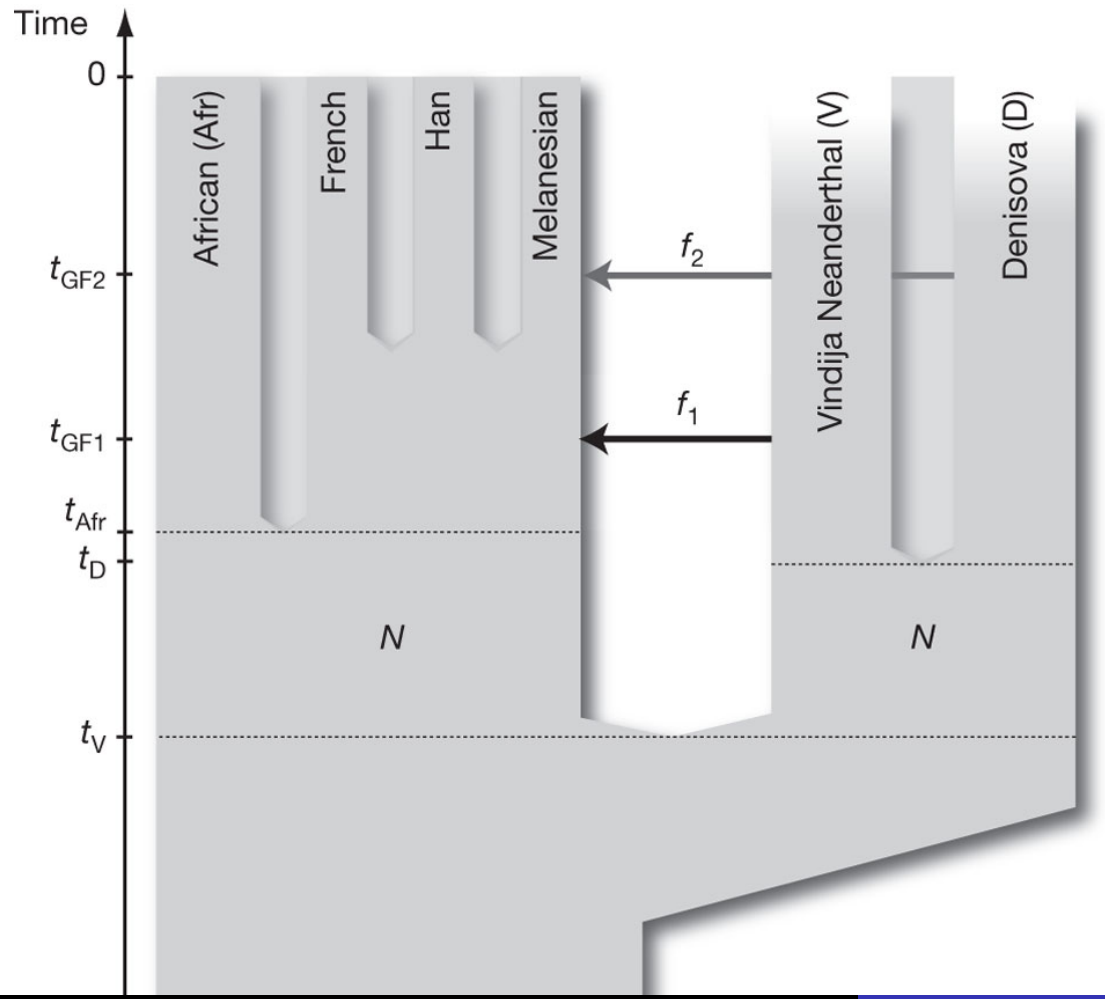
Die Gattung *Homo*

- *Homo habilis* (lebte vor 2,3 bis 1,6 Millionen Jahren),
- *Homo ergaster* (lebte vor 1,8 bis 1,5 Millionen Jahren),
- *Homo erectus* (lebte vor 1,7 Millionen bis 40.000 Jahren),
- *Homo heidelbergensis* (lebte vor 700.000 bis 200.000 Jahren),
- *Homo neanderthalensis* (lebte vor 200.000 bis 30.000 Jahren),
- *Homo floresiensis* (lebte vor 74.000 bis 12.000 Jahren),
- Denisova hominin (lebte vor ca. 40.000 Jahren)
- *Homo sapiens* (vor 200.000 Jahren bis heute),

Die Gattung *Homo*



Die Gattung *Homo*



Phylogenetische Rekonstruktionsverfahren

Gegeben: Eine Menge von Taxa—hier Nukleotidsequenzen bzw. Aminosäuresequenzen verschiedener Spezies.

- Distanzmethoden: Benutzen paarweise Distanzen zwischen den Sequenzen zur Berechnung eines phylogenetischen Baums.
- Maximum Parsimony Methode: Bestimmt einen phylogenetischen Baum, der mit den wenigsten Substitutionen auskommt, um die Unterschiede in den Sequenzen zu erklären.
- Maximum Likelihood Methode: Bestimmt einen phylogenetischen Baum, der am wahrscheinlichsten die Verwandschaft der Sequenzen wiedergibt.

Auf Grund der Zeitbeschränkung müssen wir uns auf eine simple Distanzmethode beschränken.

Genetische Distanzen

Gegeben sei ein Alignment zweier Nukleotidsequenzen. Ein Schätzer der Distanz der beiden Sequenzen ist:

$$p = \frac{\text{Anzahl der unterschiedlichen Nukleotide}}{\text{Länge des Alignments}}$$

Jukes-Cantor-Korrektur von p (berücksichtigt Rückmutationen):

$$d = -\frac{3}{4}\ln(1 - \frac{4}{3}p)$$

Kimura-2-Parameter-Korrektur von p:

$$K = -\frac{1}{2}\ln((1 - 2P - Q)\sqrt{1 - 2Q})$$

wobei P bzw. Q die Anzahl der Transitionen bzw. Transversionen geteilt durch die Länge des Alignments ist.

Genetische Distanzen, Beispiel

Die Basen A und G sind Purine, während C und T Pyrimidine sind. Ein Purin/Purin bzw. Pyrimidin/Pyrimidin Austausch wird Transition, ein Purin/Pyrimidin bzw. Pyrimidin/Purin Austausch Transversion genannt. Transitionen treten viel häufiger auf als Transversionen.

Beispiel: In einem Alignment der Länge 200 (ohne Lücken) treten 50 Transitionen und 16 Transversionen auf.

$$p = \frac{66}{200} = 0.33 \text{ und } d = -\frac{3}{4}\ln(1 - \frac{4}{3} \cdot 0.33) = 0.435$$

$$P = \frac{50}{200} = 0.25 \text{ und } Q = \frac{16}{200} = 0.08$$

$$K = -\frac{1}{2}\ln((1 - 2 \cdot 0.25 - 0.08)\sqrt{1 - 2 \cdot 0.08}) = 0.477$$

Distanzmethode UPGMA

Michener & Sokal entwickelten UPGMA = unweighted pair group method using arithmetic averages

Setzt eine konstante Evolutionsrate (molekulare Uhr) voraus.

Idee: Initial bildet jede Sequenz ein eigenes Cluster. Danach faßt man die zwei Sequenzen bzw. Cluster zusammen, die die geringste Distanz zueinander haben.

Distanz zwischen zwei Clustern C_i und C_j:

$$d(i,j) = \frac{1}{n_i n_j} \sum_{\alpha \in C_i, \beta \in C_j} d(\alpha, \beta)$$

wobei $n_i = |C_i|$ und $n_j = |C_j|$.

Inkrementelle Berechnung der Distanz von Clustern

Falls $C_k = C_i \cup C_j$ und C_ℓ ist beliebiges anderes Cluster, dann gilt:

$$d(k, \ell) = \frac{n_i \cdot d(i, \ell) + n_j \cdot d(j, \ell)}{n_i + n_j}$$

Beweis

$$
\begin{aligned}
d(k,\ell) &= \frac{1}{n_k\, n_\ell} \sum_{\alpha \in C_k, \beta \in C_\ell} d(\alpha,\beta) \\
&= \frac{1}{(n_i + n_j)\, n_\ell} \sum_{\alpha \in C_i \cup C_j, \beta \in C_\ell} d(\alpha,\beta) \\
&= \frac{1}{(n_i + n_j)\, n_\ell} \left(\sum_{\alpha \in C_i, \beta \in C_\ell} d(\alpha,\beta) + \sum_{\alpha \in C_j, \beta \in C_\ell} d(\alpha,\beta) \right) \\
&= \frac{1}{(n_i + n_j)\, n_\ell} \left(n_i\, n_\ell\, d(i,\ell) + n_j\, n_\ell\, d(j,\ell) \right) \\
&= \frac{n_i\, d(i,\ell) + n_j\, d(j,\ell)}{n_i + n_j}
\end{aligned}
$$

Algorithmus UPGMA

Initialisierung:

1. jede Sequenz i definiert ein Cluster C_i der Grösse $n_i = 1$
2. jede Sequenz bildet ein Blatt im Baum T auf Höhe 0.
3. berechne die Distanzen $d(i, j)$

Algorithmus UPGMA

Iteration, solange die Anzahl der Cluster ≥ 2 ist:

1. bestimme C_i und C_j mit $d(i,j)$ minimal
2. $C_k = C_i \cup C_j$ ist neues Cluster der Grösse $n_k = n_i + n_j$
3. in T wird ein neuer Knoten erzeugt, dessen Kinder die zu C_i und C_j korrespondierenden Knoten werden;
   Höhe des Knotens: $\frac{d(i,j)}{2}$
4. füge zur Matrix d eine Reihe/Spalte für das Cluster C_k hinzu mit

$$d(k,\ell) = \frac{n_i \cdot d(i,\ell) + n_j \cdot d(j,\ell)}{n_i + n_j}$$

   und lösche die Reihen/Spalten der Cluster C_i und C_j

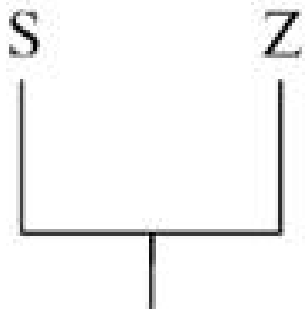
UPGMA, Beispiel

Jukes-Cantor-Distanzen aus mitochondrialen rDNA-Daten der kleinen Ribosomenuntereinheit verschiedener Hominiden (Hixson & Brown 1986). Das zugehörige Alignment umfaßt 939 bp.

	S	Z	G	M	O
Schimpanse (S)					
Zwergschimpanse (Z)	0.0118				
Gorilla (G)	0.0427	0.0416			
Mensch (M)	0.0382	0.0327	0.0371		
Orang Utan (O)	0.0953	0.0916	0.0965	0.0928	

UPGMA, Beispiel

Zunächst werden die beiden Arten mit minimalem Abstand zu einem neuen Cluster (SZ) zusammengefasst.

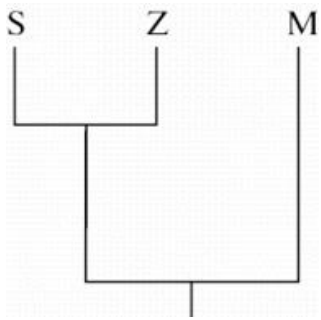


und dann die neue Distanzmatrix inkrementell berechnet:

	SZ	G	M	O
G	0.0422			
M	0.0355	0.0371		
O	0.0935	0.0965	0.0928	

UPGMA, Beispiel

Wiederum werden die beiden Cluster mit minimalem Abstand zu einem neuen Cluster (SZM) zusammengefasst.

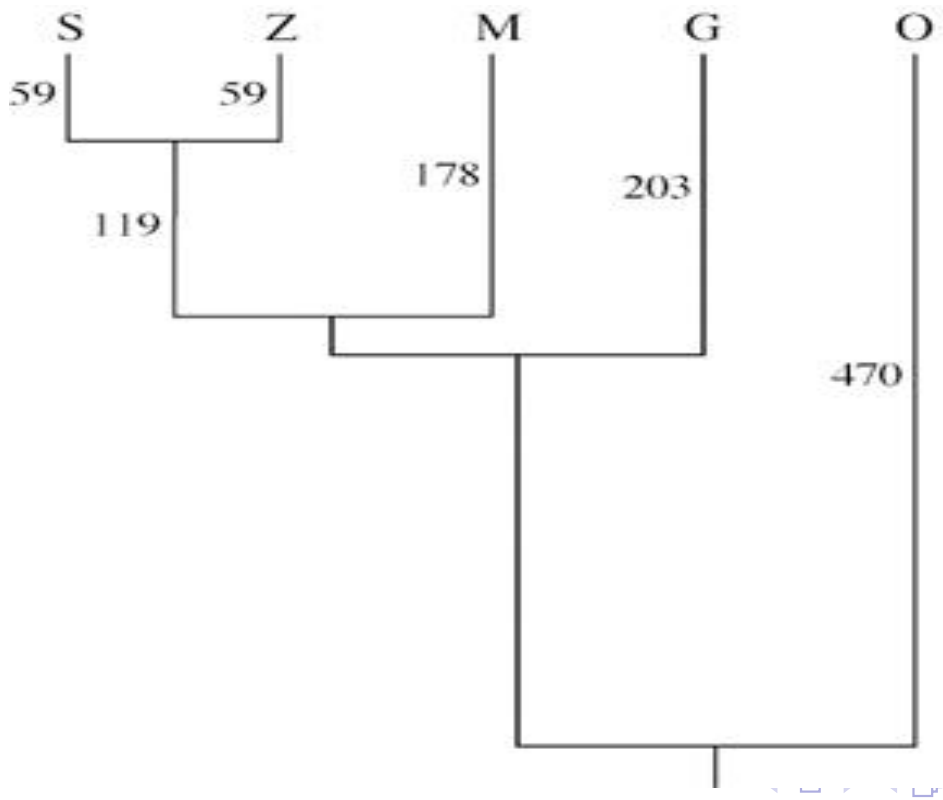


und die neue Distanzmatrix berechnet

	SZM	G	O
G	0.0405		
O	0.0932	0.0965	

UPGMA, Beispiel

Der resultierende phylogenetische Baum sieht folgendermaßen aus (die Distanzen im Baum sind mit 10^{-4} zu multiplizieren):



Prokaryonten

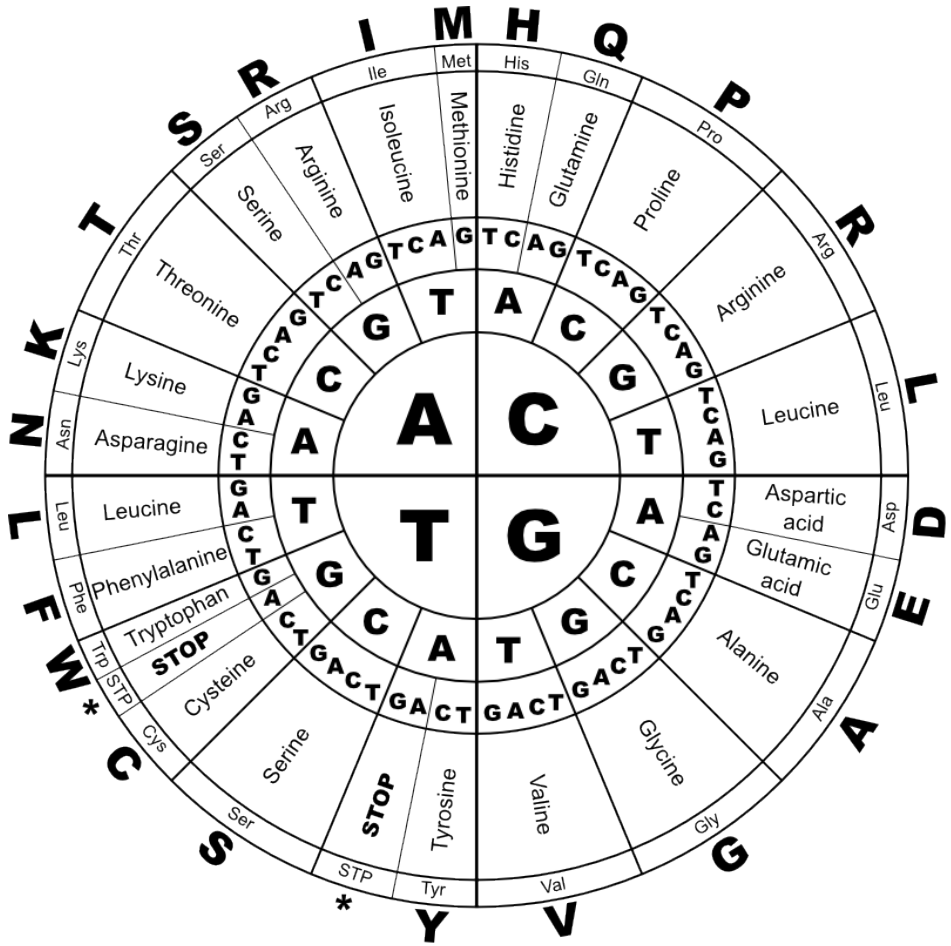
Genvorhersage ist die Aufgabe, ausgehend von der genomischen DNA Sequenz, aller Gene zu finden.

Einfachste Methode: Bestimme Open Reading Frames (ORFs). Ein ORF ist eine Folge von Codons, die mit dem Startcodon (ATG) beginnt, mit einem Stopcodon (TAA, TAG oder TGA) endet und keine weiteren Stopcodons enthält.
Es gibt 6 verschiedene Reading Frames (3 auf dem Vorwärtsstrang und 3 auf dem Rückwärtsstrang).

Die durchschnittliche Distanz zwischen zwei Stopcodons in einer "zufälligen" DNA-Sequenz ist $\frac{64}{3} \approx 21$.

Da die durchschnittliche Anzahl von Aminosäuren (Codons) in Proteinen 300 ist, deuten lange ORFs auf Gene hin.

Genetischer Code



Codon Usage

Viele Methoden zur Genvorhersage beruhen auf Codon Usage.

Die Häufigkeit des Auftretens von Codons in ORFs weicht oft stark von der Häufigkeit in nicht-kodierenden Bereichen der DNA-Sequenz ab.

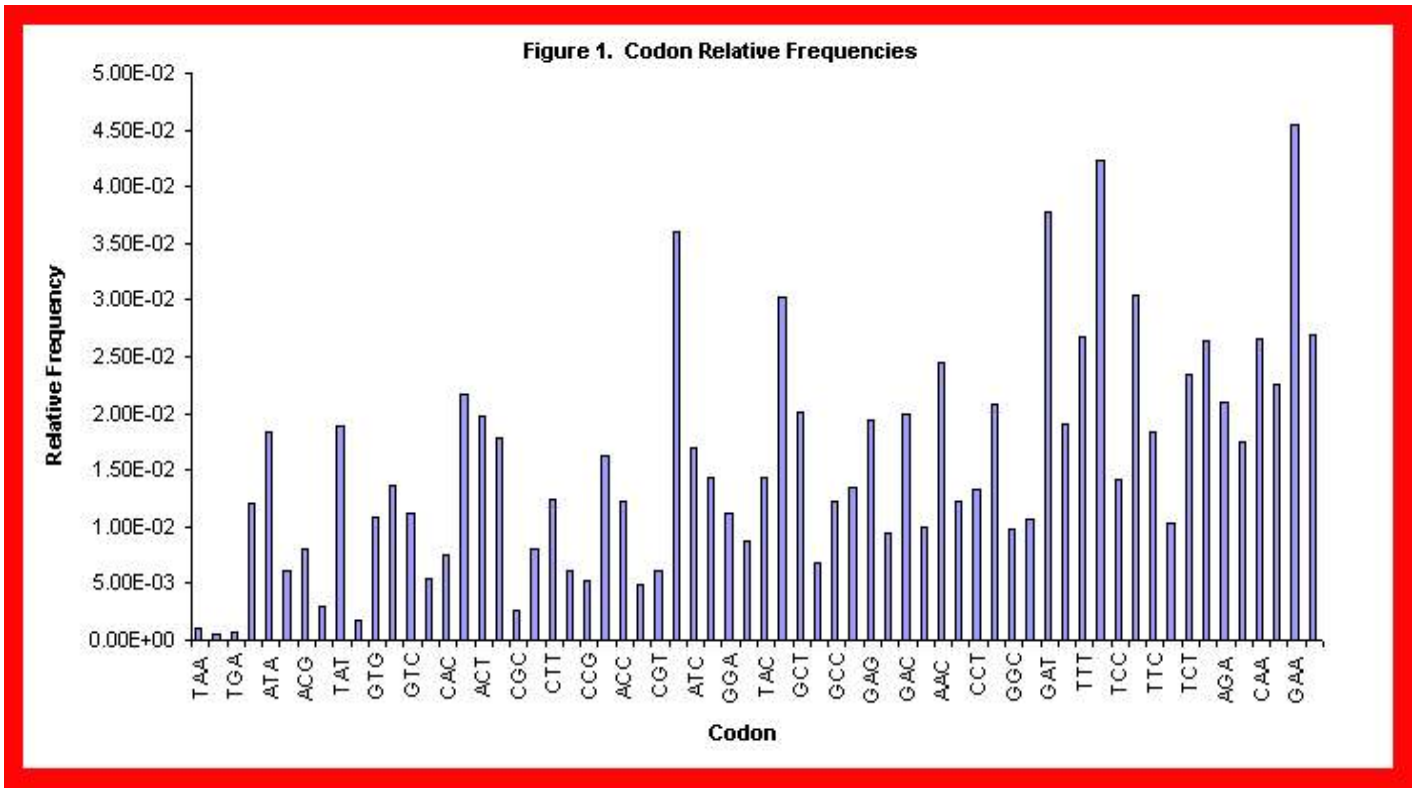
Beispiel: Codon Usage in 6161 ORFs von *S. cerevisiae*.

Von den 3 Stopcodons trat TAA 2939 mal, TGA 1824 mal und TAG 1398 mal auf.

Unter den Aminosäure-kodierenden Codons trat das Codon GAA (kodiert für Glutaminsäure) am häufigsten auf (4.6% der $2,851,170$ Codons), während CGG (kodiert für Arginin) am seltensten (0.18% der Codons) auftrat.

Codon Usage

Codon Usage in 6161 ORFs von *S. cerevisiae*.



Weitere Signale

G/C-Gehalt: Der G/C-Gehalt in Genen ist sehr oft höher als in intergenischen Regionen.

Vorkommen von Hexameren: Ein Hexamer ist eine Nukleotid-Sequenz der Länge 6.
Interpretiert man ein Hexamer als zwei aufeinanderfolgende Codons, so entspricht es zwei aufeinanderfolgenden Aminosäuren.
Es hat sich herausgestellt, dass Hexamer-Statistiken sich gut zur Unterscheidung zwischen kodierenden und nicht-kodierenden Regionen eignen.

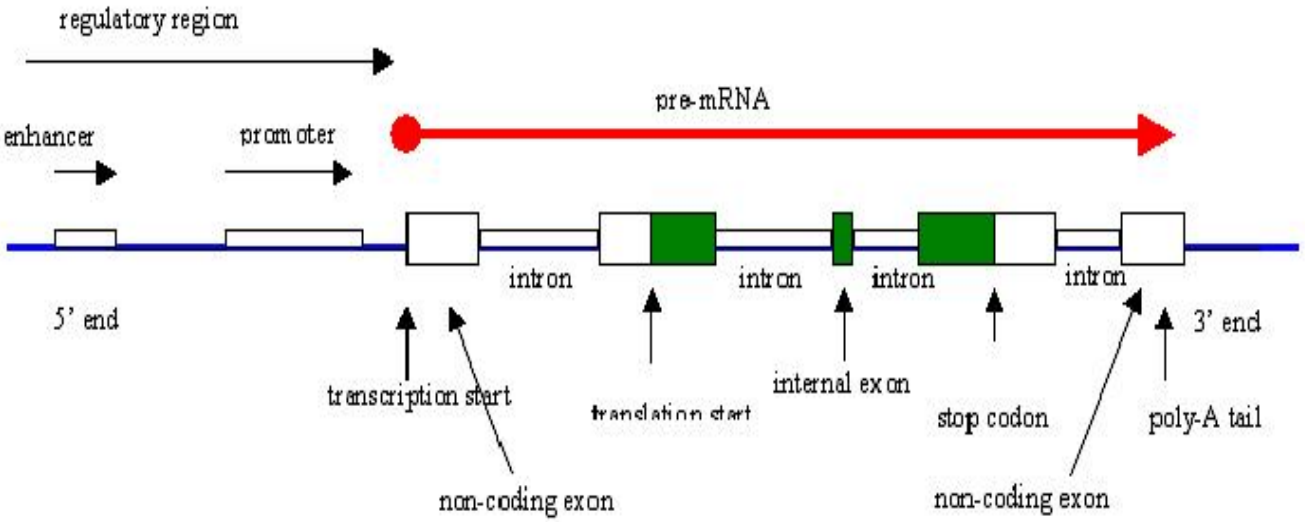
Eukaryonten

Eukaryontische Gene sind aus Exons und Introns zusammengesetzt. Durch das sogenannte Splicing nach der Transkription werden die Introns herausgeschnitten und die mRNA aus den Exons zusammengesetzt.

Genvorhersage ist die Aufgabe, ausgehend von der genomischen DNA Sequenz, die korrekte Exon/Intron Struktur aller Gene zu bestimmen.

Genvorhersage wird erschwert durch überlappende Gene, alternatives Splicing etc.

Struktur von eukaryontischen Genen



Genvorhersagemethoden

Grob kann man Genvorhersagemethoden in folgende drei Kategorien einteilen:

- *ab initio* Methoden, die die inhärenten Eigenschaften der Sequenz analysieren (z.B. Hidden Markov Models, künstliche neuronale Netze).
- Homologie-basierte Methoden, die Protein, cDNA oder EST-Datenbanken benutzen.
- Methoden, die auf einem Genomvergleich beruhen.

Kombinationen der verschiedenen Methoden sind möglich.

Spliced Alignments

Der Homologie-basierte Spliced Alignment Ansatz von Gelfand et al. zur Bestimmung der Exon-Intron Struktur eines potenziellen Genes geht davon aus, dass die cDNA eines orthologen Genes einer anderen Spezies in einer Datenbank vorhanden ist (eine Erweiterung der Methode kann auch mit Proteinen statt mit cDNA arbeiten).

Wie alle anderen Homologie-basierten Genvorhersagemethoden kann diese Methoden nur schon “bekannte” Gene finden.

Spliced Alignments

Sei $G = g_1 g_2 \dots g_n$ ein String (die neu sequenzierte DNA).

Sei E die (endliche) Menge aller potenziellen Exons von G (Teilstrings von G, die von den Dinukleotiden AG und GT flankiert werden). Elemente aus E werden Blöcke genannt.

Seien $B = g_i g_{i+1} \dots g_j$ und $B' = g_{i'} g_{i'+1} \dots g_{j'}$ Blöcke. Wir schreiben $B \prec B'$, falls $j < i'$ gilt.

Eine Folge $\Gamma = B_1, B_2, \dots, B_s$ von Blöcken aus E wird als Kette bezeichnet, falls gilt:

$$B_1 \prec B_2 \prec \dots \prec B_s$$

Mit $\Gamma^* = B_1 B_2 \dots B_s$ bezeichnen wir die Konkatenation der Blöcke.

Spliced Alignment Problem

Sei $T = t_1 t_2 \dots t_m$ ein weiterer String, die so genannte Target-Sequenz (hier die cDNA).

Das Spliced Alignment Problem besteht darin, eine Kette Γ von Blöcken aus E zu finden, so dass der Score $S(\Gamma^*, T)$ des Alignments von Γ^* und T maximal ist.

Wir verwenden folgende Bezeichnungen:
Sei $B = g_f \dots g_i \dots g_l$ ein Block aus E.

- $first(B) = f$
- $last(B) = l$
- $E(i) = \{B_k \in E \mid last(B_k) < i\}$
- $B(i) = g_f \dots g_i$ und $T(j) = t_1 t_2 \dots t_j$

Spliced Alignments

Sei $\Gamma = B_1, \ldots, B_k, \ldots, B_t$ eine Kette, so dass B_k die Position i (d.h. g_i) enthält und definiere $\Gamma^*(i) = B_1 B_2 \ldots B_{k-1} B(i)$. Sei

$$S(i,j,k) = \max_{\text{alle Ketten } \Gamma, \text{ die } B_k \text{ enthalten}} S(\Gamma^*(i), T(j))$$

Um das Spliced Alignment Problem zu lösen, muss man das folgende Maximum bestimmen:

$$\max_k S(last(B_k), m, k)$$

Spliced Alignments

Die folgende Rekursionsgleichung ermöglicht dieses Berechnung des Maximums mittels Dynamic Programming:

$$S(i,j,k) =$$

$$\max \begin{cases} S(i-1,j-1,k) + score(g_i, t_j) & i \neq first(B_k) \\ S(i-1,j,k) + score(g_i, -) & i \neq first(B_k) \\ S(i,j-1,k) + score(-, t_j) & \\ max_{B_l \in E(i)}\{S(last(B_l), j-1, l)\} + score(g_i, t_j) & i = first(B_k) \\ max_{B_l \in E(i)}\{S(last(B_l), j, l)\} + score(g_i, -) & i = first(B_k) \end{cases}$$