Algorithmen und Modelle der Bioinformatik

Dirk Metzler
www.informatik.uni-frankfurt.de/∼metzler
metzler@informatik.uni-frankfurt.de

Wintersemester 2007/2008

Vorlesung mit Übung, 4+2 Semesterwochenstunden

Inhalt u.a. Paarweises Alginment mit dynamischer Programmierung, BLAST, Mustersuche, Multiple Alignments, Modelle der Sequenzevolution, Hidden Markov Modelle, parsimonische und maximum-likelihood-basierte Verfahren der Stammbaumschätzung, Markoffketten-Monte-Carlo-Methoden, Vorhersage von RNA-Sekundärstrukturen mit stochastischen kontextfreien Grammatiken, Populationsgenetik: Importance Sampling für den Coalescent.

Empfohlene Literatur

R.C. Deonier, S. Tavaré, M.S. Waterman *Computational Genome Analysis.* Springer, 2005

R. Durbin, S. Eddy, A. Krogh, G. Mitchison: *Biological sequence analysis.* Cambridge Univ. Press, 1998

W. Ewens, G. Grant: *Statistical Methods in Bioinformatics.* Springer, 2001.

J. Felsenstein: *Inferring Phylogenies*. Sinauer, 2004.

D. Gusfield: *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, 1999.

R. Nielsen (Ed.): *Statistical Methods in Molecular Evolution*. Springer, 2005

M.S. Waterman: *Introduction to Computational Biology*. Chapman & Hall, 1995

Inhaltsverzeichnis

1 Klassische Alignment-Algorithmen für Sequenzpaare

Gegeben seien zwei DNA- oder Protein-Sequenzen $x, y \in \mathcal{A}^*$, ($\mathcal{A}$ bezeichne das entsprechende Alphabet, $\mathcal{A}^*$ die Menge der endlichen Wörter über $\mathcal{A}$).

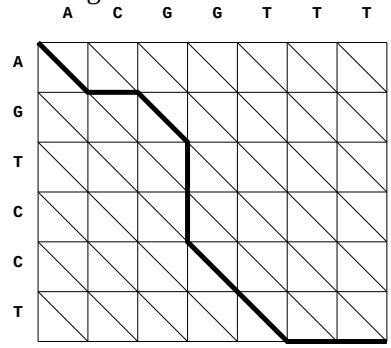
Frage: Welche Positionen in den Sequenzen gehören zueinander? (z.B. bzgl. Abstammung oder Funktion)

Z.B. seien die DNA-Sequenzen `ACGGTTT` und `AGTCCT` gegeben. Plausibel erscheint dann z.B. folgendes Alignment:

```
ACG__GTTT
A_GTCCT__
```

Wenn an zwei miteinander alignierten Positionen die gleichen Buchstaben aus $\mathcal{A}$ (also Basen bzw. Aminosäuren) stehen, sprechen wir von einem **Match**, bei ungleichen von einem **Mismatch**. Eine Folge von Strichen, die gegenüber direkt aufeinanderfolgenden nicht alignierten Positionen stehen, wird als **Gap** bezeichnet. Obiges Alignment enthält also 3 Matches, ein Mismatch und drei Gaps, von denen einer der Länge 1 und zwei der Länge 2 sind. Wenn die Sequenzen gemeinsamen Ursprungs sind und das Alignment darstellt, welche Positionen zueinander **homolog** sind, dann entsprechen Gaps **Insertionen** und **Deletionen**, die sich seit dem jüngsten gemeinsamen Vorfahren der Sequenzen ereignet haben.

Folgendes Beispiel zeigt, wie man Alignments als Pfade durch Graphen darstellen kann:



Alignierte Positionspaare werden hier als diagonale Stücke dargestellt, Gaps in der oberen Sequenz entsprechen vertikalen und Gaps in der anderen Sequenz horizontalen Linien.

Es handelt sich hierbei um ein **globales Alignment**, d.h. wir gehen von vornherein davon aus, dass die Sequenzen gemeinsamen Ursprungs sind (oder zumindest eine ähnliche Funktion haben), und wollen sie möglichst komplett alignieren, d. h. der Pfad beginnt in der linken oberen Ecke des Graphen und endet in der rechten unteren. Gaps am Anfang und Ende werden genauso behandelt wie Gaps innerhalb der Sequenzen.

Im Gegensatz dazu geht es beim **lokalen Alignment** darum, ob innerhalb der (evtl. sehr langen) Sequenzen (von denen eine oft eine Datenbank ist) kürzere Teilsequenzen gut gegeneinander alignierbar sind. Es stellt sich dann auch die Frage der Signifikanz: Können die beobachteten Übereinstimmungen rein zufällig zustande kommen?

1.1 Score-Optimierung mit dynamischer Programmierung

Als Kriterium, wie "gut" ein Alignment ist, wird ein Score definiert. Dabei werden i. d. R. Matches belohnt und Mismatches und Gaps bestraft. Ein einfaches Beispiel: Belohne jeden Match mit +1, bestrafe jeden Mismatch mit der Mismatch penalty $-m$ und jeden Gap der Länge g mit $-\gamma(g) = -d-(g-1)\cdot e$. Das obige Alignment würde also den Score $3 - m - 3d - 2e$ erhalten. Die positiven Zahlen d und e heissen *gap open penalty* und *gap extension penalty* und es gilt $d \geq e$. Da γ affin in g ist, spricht man von einer *affinen Gap penalty*, und mit *linearer Gap penalty* bezeichnet man den Spezialfall $d = e$.

Auch andere monoton steigende Funktionen für γ werden gelegentlich benutzt.

Insbesondere bei Protein-Sequenzen verwendet man Score-Schemata, die nicht nur zwischen Match und Mismatch unterscheiden, sondern jedem Paar $(a, b) \in \mathcal{A}^2$ einen Score $s(a, b)$ zuordnen (etwa durch PAM- oder BLOSUM-Matrizen; dazu später mehr). Für $a = b$ ist $s(a, b)$ normalerweise positiv und für $a \neq b$ eher (aber nicht immer) negativ. Das Score-Schema wird üblicherweise so gewählt, dass für zufällige, nicht verwandte Sequenzen ein negativer Score zu erwarten ist. Oft wird $s(a, b) > 2\gamma(1)$ gefordert, so dass ein Mismatch zwischen a und b nicht strenger bestraft wird als zwei einzelne Gap-Positionen, durch die man ihn vermeiden könnte.

1.1.1 Globales Alignment

Gegeben: zwei Sequenzen $x = (x_1, x_2, \ldots, x_n) \in \mathcal{A}^n$ und $y = (y_1, \ldots, y_m) \in \mathcal{A}^m$ und ein Score-Schema $s(a, b)$ für $a, b \in \mathcal{A}$ mit affiner Gap penalty $\gamma(g) = d + (g - 1) \cdot e$.

Gesucht: globales Alignment von x und y mit maximalem Score.

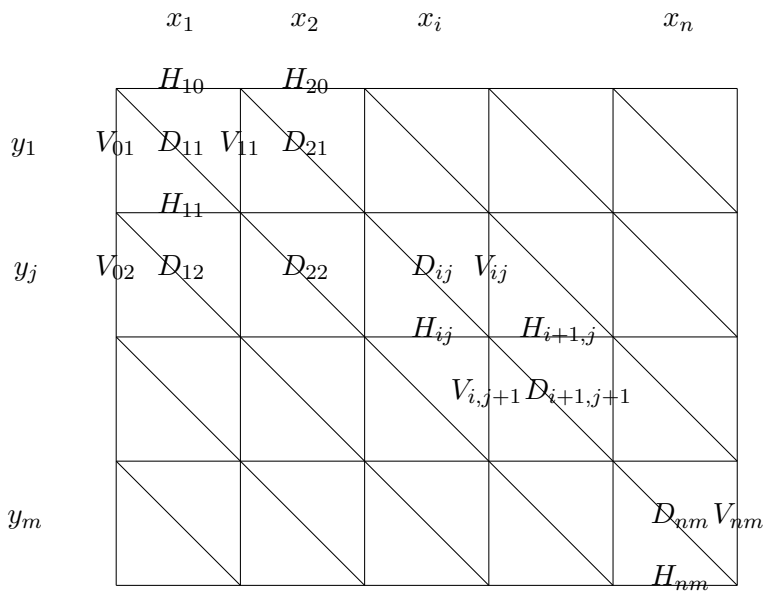
erster Ansatz (naiv): Berechne für jedes denkbare Alignment den Score und nimm das Maximum.

Problem: Es gibt so etwa $3^{\max(n,m)}$ mögliche Alignments, und wir wollen nicht exponential lange rechnen.

Ausweg: Dynamische Programmierung nach Needleman und Wunsch (1970)

Beschrifte jede Kante des Alignment-Graphen mit dem Score des besten dort endenden Alignments der Sequenzanfänge.

Sei D_{ij} (bzw. H_{ij}, bzw. V_{ij}) der Score des besten Alignments von $(x_1, \ldots, x_i)$ und $(y_1, \ldots, y_j)$ das mit $\begin{smallmatrix} x_i \\ y_j \end{smallmatrix}$ (bzw. $\begin{smallmatrix} x_i \\ - \end{smallmatrix}$, bzw. $\begin{smallmatrix} - \\ y_j \end{smallmatrix}$) endet ($D$, H und V stehen für "diagonal", "horizontal", "vertikal").



Wenn es gelingt, alle D_{ij}, V_{ij}, H_{ij} effizient auszurechen, erhalten wir auch leicht den maximal erreichbaren Score S^* eines globalen Alignments von x und y, denn es gilt:

$$S^* = \max\{D_{nm}, V_{nm}, H_{nm}\}$$

Für die Beschriftungen der Kanten am linken und am oberen Rand gilt:

$$\begin{aligned}
D_{11} &= s(x_1, y_1) \\
D_{1j} &= s(x_1, y_j) - d - e \cdot (j-2) \\
D_{i1} &= s(x_i, y_1) - d - e \cdot (i-2) \\
H_{i0} &= -d - e \cdot (i-1) \\
H_{1j} &= -2d - e \cdot (j-1) \\
V_{0j} &= -d - e \cdot (j-1) \\
V_{i1} &= -2d - e \cdot (i-1)
\end{aligned}$$

Für $i, j \geq 1$ gilt:

$$\begin{aligned}
D_{i+1,j+1} &= s(x_{i+1}, y_{j+1}) + \max\{D_{ij}, H_{ij}, V_{ij}\} \\
H_{i+1,j} &= \max\{D_{ij} - d, H_{ij} - e, V_{ij} - d\} \\
V_{i,j+1} &= \max\{D_{ij} - d, H_{ij} - d, V_{ij} - e\}
\end{aligned}$$

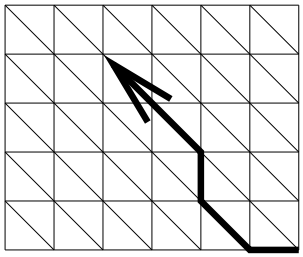
(Je nach Score-Schema kann es sein, dass ein direkter Übergang zwischen horizontalen und vertikalen Kanten, also Gaps in der einen und der anderen Sequenz, bei Score-optimierten Alignments ausgeschlossen ist.)

Nach der Idee der dynamischen Programmierung verwenden wir diese Rekursionen nun in einer günstigen Reihenfolge, nämlich Zeile für Zeile, so dass die nötigen Zwischenergebnisse jeweils bereitstehen:

```
Berechne H_i0 und V_0j für alle i ≤ n und j ≤ m.

Für j von 1 bis m:
   Für i von 1 bis n:
     Berechne D_ij, H_ij und V_ij nach obigen Gleichungen
```

Damit erhalten wir also in Zeit $O(nm)$ den optimalen Score, denn es müssen $3nm + n + m$ Kanten beschriftet werden, und der Zeitaufwand pro Kante ist unabhängig von n und m.



Wie finden wir aber das zum optimalen Score gehörige Alignment? Dazu laufen wir im Alignment-Graphen von rechts unten nach links oben. Zuerst wählen wir die Kante aus, die die höchste Beschriftung D_{nm}, H_{nm} oder V_{nm} hat. Dann wählen wir jeweils die Kante, die bei der Berechnung der Beschriftung der zuletzt gewählten Kante das "Maximum" geliefert hat. Das machen wir solange bis wir links oben angekommen sind. Der Zeitaufwand ist offensichtlich linear in n um m.

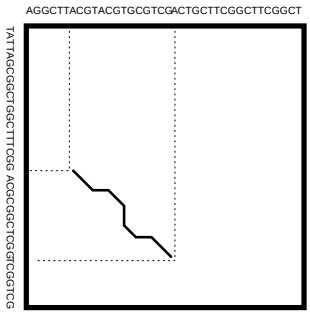
Insgesamt liefert also der Needleman-Wunsch-Algorithmus das Alignment mit optimalem Score mit Zeit- und Speicherplatzaufwand $O(nm)$.

Nicht-affine Gap-Penalty: Wenn γ nicht affin ist, z.B. $\gamma(g) = 1 + \log(g)$, sehen die Rekursionen ein bisschen anders aus, z.B.:

$$H_{ij} = \max_{k=1,\ldots,i-1} \{\max\{D_{kj}, V_{kj}\} - \gamma(i-k)\}$$

Der Aufwand in jedem Schritt wächst also linear mit n bzw. m, da jeweils über linear viele Werte für k maximiert werden muss. Als Zeitabschätzung erhalten wir damit bei Verwendung dieser Rekursion $O(nm(n+m))$.

1.1.2 Lokales Alignment



Nun geht es also darum, ob und ggf. welche Teile zweier Sequenzen (von denen eine oft eine ganze Datenbank ist) miteinander alignierbar sind. Die Strategie ist dabei, Alignments von Teilsequenzen so lange zu verfolgen, wie der Score positiv bleibt. Das Score-Schema muss also so gewählt sein, dass globale Alignments unverwandter Sequenzen höchstwahrscheinlich negativen Score bekommen. Bezeichnet h_a die relative Häufigkeit von $a \in \mathcal{A}$, so muss also insbesondere gelten:

$$\sum_{a,b \in \mathcal{A}} h_a \cdot h_b \cdot s(a,b) < 0$$

Sonst ist bereits für gaplose Alignments unverwandter Sequenzen ein positiver Score zu erwarten.

Wir verwenden hier wieder eine affine Gap-Penalty-Funktion $\gamma(g) = d + (g-1)e$.

Der Algorithmus von Smith und Waterman (1981) zum Finden des optimalen lokalen Alignments bei affiner (in der Orginalarbeit linearer) Gap-Penalty ist eine Variante des Needleman-Wunsch-Algorithmus. Diesmal sind die Beschriftungen der Kanten des Alignment-Graphen D_{ij}, H_{ij}, V_{ij} die Scores der besten *lokalen* Alignments, die in der jeweiligen Kante enden. Z. B. ist jetzt also D_{ij} der beste Score eines Alignments von Teilsequenzen $(x_k, \ldots, x_i)$ und $(y_l, \ldots, y_j)$, das mit $\begin{smallmatrix} x_i \\ y_j \end{smallmatrix}$ endet, jedoch einschließlich des Falls, dass die Teilsequenzen leer sind, was dem Score 0 entspricht. Dieser Fall hat die Interpretation, dass es sich nicht lohnt, an der Stelle (i, j) ein Alignment weiterzuführen, und dass man lieber daneben mit einem neuen lokalen Alignment anfängt.

Wir erhalten also die Rekursionsgleichungen

$$\begin{aligned} D_{i+1,j+1} &= \max\{0, D_{ij} + s(x_{i+1}, y_{j+1}), H_{ij} + s(x_{i+1}, y_{j+1}), V_{ij} + s(x_{i+1}, y_{j+1})\} \\ H_{i+1,j} &= \max\{0, D_{ij} - d, H_{ij} - e, V_{ij} - d\} \\ V_{i,j+1} &= \max\{0, D_{ij} - d, H_{ij} - d, V_{ij} - e\} \end{aligned}$$

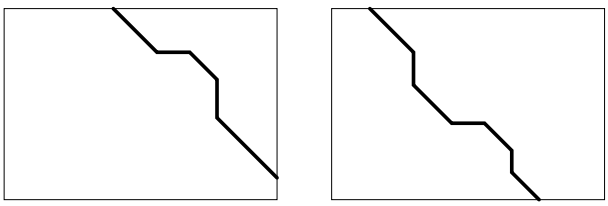
und entsprechende Gleichungen für die Kanten am linken und oberen Rand.

Damit beschriften wir alle Kanten des Alignment-Graphen und merken uns, welche Kante den maximalen Wert erhielt. Wir laufen dann von dieser Kante aus (und nicht wie bei Needleman-Wunsch von rechts unten) im Alignment-Graphen nach links und/oder oben, indem wir wieder jeweils die Kante wählen, die den maximalen Beitrag zur aktuellen Kante geliefert hat. Dies tun wir so lange, bis wir auf eine Kante treffen, die mit dem Wert 0 beschriftet ist. Damit haben wir dann das optimale Alignment zurückverfolgt (vgl. Abbildung 1).

Ebenso wie der Needleman-Wunsch-Algorithmus benötigt der Smith-Waterman-Algorithmus Zeit und Speicherplatz $O(nm)$. (Das ist nicht schlecht, aber nicht gut genug für die Suche in großen Datenbanken. Das Programm BLAST etwa läuft wesentlich schneller, kann dafür aber nicht garantieren, wirklich das optimale Alignment zu finden. Wir werden später darauf zurückkommen.)

1.1.3 Mischformen von lokal und global

Alternativ zum lokalen oder globalen Alignment kann auch vorgegeben werden, dass eine Sequenz komplett mit einem Teil der anderen Sequenz aligniert werden soll oder dass der Anfang einer Sequenz mit dem Ende der anderen aligniert werden soll. Also sollen folgende Typen von Alignment-Pfaden bevorzugt werden:



Das lässt sich leicht erreichen: Wir müssen nur am Needleman-Wunsch-Algorithmus für globales Alignment zwei Änderungen vornehmen. Zum einen ist die Initialisierung nun:

$$D_{1j} = s(x_1, y_j), \quad D_{i1} = s(x_i, y_1), \quad V_{0j} = H_{i0} = 0, \quad V_{i1} = H_{1j} = -d$$

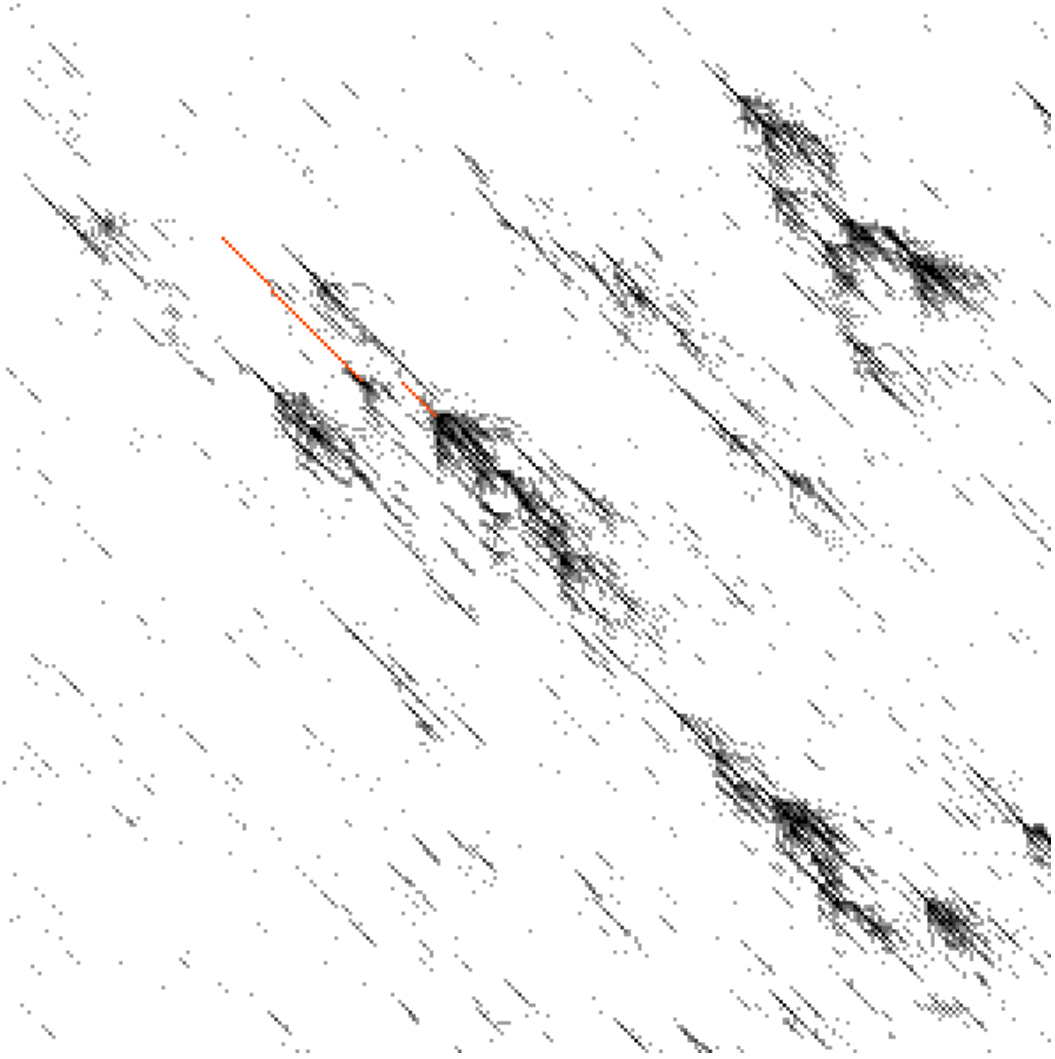


Abbildung 1: Für zwei Sequenzen von je 300 rein zufälligen Nukleotiden wurde mit Smith-Waterman das optimale lokale Alignment berechnet (rot). Die Grauwerte zeigen die “Inseln” der Punkte (i, j), in denen D_{ij} große Werte annahm. Das rechte untere Ende des optimalen lokalen Alignments liegt dort, wo D_{ij} maximal wird.

(Die Rekursionsgleichungen für die übrigen Kantenbeschriftungen bleiben dieselben wie bei Needleman-Wunsch.)

Zum anderen beginnt das Zurückverfolgen des optimalen Alignments nicht unbedingt in der rechten unteren Ecke des Alignment-Graphen, sondern am rechten oder unteren Rand, und zwar dort, wo D_{ij} mit $i = n$ oder $j = m$ maximal wird.

1.1.4 Globales Alignment mit linearem Speicheraufwand

Wir behandeln nun einen Algorithmus, der auf Hirschberg (1975) und auf Myers und Miller (1988) zurückgeht.

Zunächst einmal bemerken wir: Wenn wir nur den Score des optimalen globalen Alignments haben wollen, aber nicht das Alignment selbst, dann können wir beim Needleman-Wunsch-Algorithmus, jeweils nachdem wir für ein fest gewähltes j für jedes i die Werte D_{ij}, H_{ij} und V_{ij} (also eine Zeile im Alignment-Graphen) berechnet haben, die vorherige Zeile (also die Werte $D_{i,j-1}$, $H_{i,j-1}$ und $V_{i,j-1}$) löschen. Damit haben wir nur linearen Speicheraufwand. Allerdings verwendet der Needleman-Wunsch-Algorithmus die Werte der vorherigen Zeilen, um das optimale Alignment zurückzuverfolgen.

Eine Alternative beruht auf der Idee, zunächst zu ermitteln, was das optimale Alignment mit der Mitte der Sequenz $y = (y_1, \ldots, y_m)$ tut, und genau das dann rekursiv für die erste und die zweite Hälfte von y aufzurufen.

Sei also $k := \lfloor \frac{m}{2} \rfloor$ und es sei $\widetilde{D}_{ik}$ für $i \leq n$ der Score des besten Alignments unter allen, in denen die Paarung $\begin{smallmatrix} x_i \\ y_k \end{smallmatrix}$ vorkommt. $\widetilde{V}_{ik}$ sei der Score des besten Alignments unter allen, in denen die k-te Position der Sequenz y gegenüber einem Gap zwischen der i-ten und der $i+1$-ten Position der Sequenz x steht.

Für die Berechnung von $\widetilde{D}_{ik}$ und $\widetilde{V}_{ik}$ verwenden wir noch D'_{ij}, H'_{ij} und V'_{ij}, den Score des besten mit $\begin{smallmatrix} x_i \\ y_j \end{smallmatrix}$ (bzw. $\begin{smallmatrix} x_i \\ - \end{smallmatrix}$, bzw. $\begin{smallmatrix} - \\ y_j \end{smallmatrix}$) beginnenden Alignments der Sequenz-Enden $(x_i, x_{i+1}, \ldots, x_n)$ und $(y_j, y_{j+1}, \ldots, y_m)$.

Wir können D'_{ij}, H'_{ij} und V'_{ij} für alle i, j mit dynamischer Programmierung gemäß Needleman-Wunsch berechnen, da es gerade die Werte $D_{n-i+1,m-j+1}$, $H_{n-i+1,m-j+1}$ und $V_{n-i+1,m-j+1}$ für die invertierten Sequenzen $(x_n, x_{n-1}, \ldots, x_1)$ und $(y_m, y_{m-1}, \ldots, y_1)$ sind. Anders ausgedrückt: Wir können D'_{ij}, H'_{ij} und V'_{ij} mit analogen Rekursionen wie bei Needleman-Wunsch berechnen, fangen dabei aber nicht bei der ersten Zeile sondern mit der letzten Zeile des Graphen an, also mit den Werten D'_{im}, H'_{im} und V'_{im}, die wir in der Reihenfolge $i = n, n-1, \ldots, 1$ berechnen. Z.B. gilt:

$$D'_{ij} = s(x_i, y_j) + \max\{D'_{i+1,j+1}, H'_{i+1,j}, V'_{i,j+1}\}$$

Um nun $\widetilde{D}_{ik}$ und $\widetilde{V}_{ik}$ für alle i zu berechnen, berechnen wir zunächst mittels dynamischer Programmierung alle Werte D_{ij}, H_{ij} und V_{ij} für $i \leq n$ und $j < k$ **(!)** sowie alle Werte D'_{ij} und V'_{ij} für $i \leq n$ und $j > k$ sowie H'_{ij} für $i \leq n$ und $j \geq k$. Um Speicher zu sparen, löschen wir dabei die jeweils älteren Zeilen, die wir nicht mehr brauchen.

Mit den Gleichungen

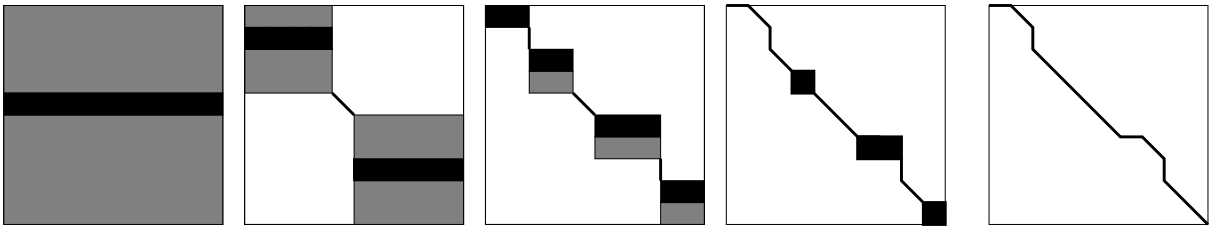
$$\widetilde{D}_{ik} \quad = \quad s(x_i, y_k) + \max\{D_{i-1,k-1}, H_{i-1,k-1}, V_{i-1,k-1}\} + \max\{D'_{i+1,k+1}, H'_{i+1,k}, V'_{i,k+1}\}$$

$$\widetilde{V}_{ik} \quad = \quad \max\{D_{i,k-1} - d, H_{i,k-1} - d, V_{i,k-1} - e\} + \max\{D'_{i+1,k+1}, H'_{i+1,k}, V'_{i,k+1} - e + d\}$$

berechnen wir dann die Werte $\widetilde{D}_{ik}$ und $\widetilde{V}_{ik}$ für $i \leq n$ (also die mittlere Zeile im Alignment-Graphen) und suchen davon das Maximum, womit wir ermittelt haben, was das optimale Alignment mit der k-ten Position von y anstellt. Wir haben dazu $3nm + n + m \approx 3nm$ Kanten mit Werten beschriftet.

Sei l die Position in x, für die $\widetilde{D}_{lk}$ (oder $\widetilde{V}_{lk}$) das gefundene Maximum ist. Dann wissen wir, dass das optimale Alignment die Sequenzanfänge $(x_1, \ldots, x_{l-1})$ (oder $(x_1, \ldots, x_l)$) und $(y_1, \ldots, y_{k-1})$ miteinander aligniert sowie die Sequenzenden $(x_{l+1}, \ldots, x_n)$ mit $(y_{k+1}, \ldots, y_m)$. Um für beide Sequenzpaare jeweils wieder herauszufinden, was das optimale Alignment mit der Mitte der zweiten Sequenz tut, müssen wir insgesamt höchstens $3l(k-1) + l + k - 1 + 3(n-l)(m-k) + (n-l) + (m-k) \approx 3nm/2$ Kanten beschriften. Bei den folgenden rekursiven Aufrufen wird die Anzahl der zu berechnenden Kanten (symbolisiert durch die dunklen Flächen in den nachfolgenden Abbildungen A bis E; die schwarzen Streifen stellen jeweils die mittleren Zeilen der (Teil-)Sequenz (von) y dar) jeweils im Wesentlichen halbiert.

A B C D E



Insgesamt muss also ungefähr $3nm(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots) \approx 6nm$ mal ein Wert für eine Kante berechnet werden. Damit ist die Rechenzeit immer noch $O(nm)$ (wenn auch etwa doppelt so groß wie beim Needleman-Wunsch-Algorithmus) und der Speicherbedarf ist $O(n)$.

1.2 Scores für Matches und Mismatches

1.2.1 PAM-Matrizen

Dayhoff, Schwarz und Orcutt (1978) haben sich bei der Konstruktion der PAM-Matrizen, eines Score-Schemas für Matches und Mismatches von Aminosäuren, von der Idee leiten lassen, dass die Scores reflektieren sollten, wie häufig die Aminosäuren durch Punktmutationen ineinander überführt werden.

Der Einfachheit halber bezeichnen wir hier die Aminosäuren mit $A_1, A_2, \ldots, A_{20}$. Gesucht ist also zunächst eine Matrix

$$M = \begin{pmatrix} m_{11} & m_{12} & \ldots & m_{1,20} \\ m_{21} & m_{22} & \ldots & m_{2,20} \\ \vdots & \vdots & \ddots & \vdots \\ m_{20,1} & m_{20,2} & \ldots & m_{20,20} \end{pmatrix},$$

deren Einträge m_{ij} brauchbare Schätzungen für die Wahrscheinlichkeit sind, dass an einer Stelle, wo A_i stand, nach einer Zeiteinheit die Aminosäure A_j steht. Die Zeiteinheit ist dabei 1 PAM (**P**rozent **a**kzeptierter **M**utationen), also die Zeitspanne, in der der pro Position 0.01 Substitutionen stattfinden,

wobei wir hier und im Folgenden nur solche Substitutionen betrachten, die nicht wieder durch Selektion verschwinden.

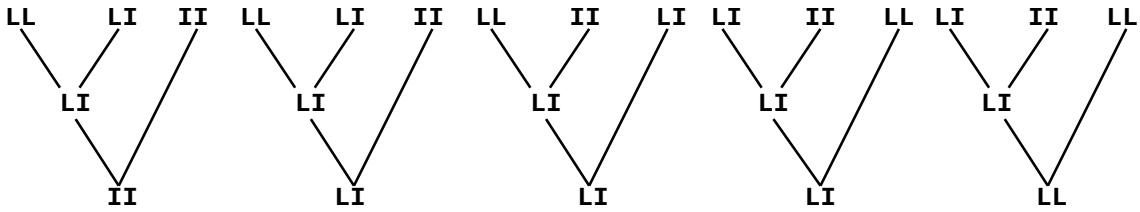
Damit die Einträge von M als Wahrscheinlichkeiten interpretiert werden können, muss für alle i, j gelten: $0 \leq m_{ij} \leq 1$ und $\sum_k m_{ik} = 1$. Außerdem wird, um die Modellklasse etwas überschaubarer zu halten, noch Reversibilität gefordert, d. h. es sollen im Schnitt genauso viele Positionen von A_i nach A_j mutieren wie umgekehrt. Das bedeutet, dass $h_i m_{ij} = h_j m_{ji}$ gelten muss, wobei h_i die relative Häufigkeit bezeichnet, mit der die Aminosäure A_i vorkommt (d. h. auch hier gilt $\sum_i h_i = 1$).

Dayhoff et. al (1978) haben die Matrix M aus damals zur Verfügung stehenden Protein-Datensätzen geschätzt. Es wurden 71 Blöcke von Teilsequenzen verwendet, die gaplos und sehr eindeutig aligniert waren. Je 2 Sequenzen innerhalb eines Blocks durften sich höchstens in 15% der Positionen unterscheiden.

Für jeden solchen Block wurden dann gewurzelte parsimonische Stammbäume mit Sequenzbelegungen der inneren Kanten konstruiert, also Bäume, die mit möglichst wenigen Mutationen auskommen (wie das effizient zu machen ist, wird in einem späteren Kapitel der Vorlesung thematisiert). Zum Beispiel gibt es für den Block aus drei Sequenzen der Länge zwei

```
LL
LI
II
```

folgende fünf maximal-parsimonische Bäume mit inneren Sequenzen:



Wir beobachten folgende Anzahlen an Übergängen längs einer Kante:

	insgesamt	das macht im Schnitt pro Baum (C_{ij})
L→L	15	3
I→I	15	3
L→I	5	1
I→L	5	1

In diesem Beispiel wurden genausoviele Übergänge L→I wie I→L beobachtet. Das ist eher untypisch und wird normalerweise nicht der Fall sein. Da wir aber von Reversibilität ausgehen, symmetrisieren wir die mittleren Anzahlen C_{ij} der Übergänge pro Baum, d. h. wir rechnen weiter mit $A_{ij} = \frac{1}{2}(C_{ij} + C_{ji})$.

Eine naheliegende Schätzung für die Wahrscheinlichkeit, dass ein in einem inneren Knoten eines der Bäume stehenden A_i am anderen Ende einer ausgehenden Kante zu einem A_j geworden ist, ist nun:

$$a_{ij} = \frac{A_{ij}}{\sum_m A_{im}}$$

Das ist natürlich ganz grob, denn hier wird nicht berücksichtigt, dass die Kanten alle unterschiedliche Länge haben, d. h. für unterschiedliche evolutionäre Zeitspannen stehen. Genauso grob gehts weiter: Da wir nicht die Längen der Kanten in den parsimonischen Bäumen als Zeiteinheit benutzen wollen, sondern 1 PAM, skalieren wir durch Multiplikation mit einer geeigneten positiven Konstante:

$$m_{jk} := c \cdot a_{jk} \qquad \text{für } j \neq k$$

Damit die Zeilen der Matrix M in Summe 1 ergeben, müssen wir außerdem setzen:

$$\forall_j : m_{jj} := 1 - \sum_{k \neq j} m_{jk}$$

Bei dieser Vorgehensweise werden Sättigungseffekte durch Rück- und Mehrfachmutationen, die sich über längere Zeitspannen an einer Position ereignen können, vernachlässigt. Das erscheint vertretbar, wenn die evolutionären Zeitspannen, um die es hier geht, also die Kantenlängen und das PAM, sehr klein sind. Insbesondere muss c so klein sein, dass alle m_{ij} kleiner als 1 sind.

Da c so gewählt sein soll, dass sich in einem durch M beschriebenen Schritt 1 Mutation pro 100 Positionen ereignet, muss gelten:

$$\frac{1}{100} = \sum_i \sum_{j \neq i} h_i m_{ij} = c \cdot \sum_i \sum_{j \neq i} h_i a_{ij}$$

Wir wählen also $c = (100 \cdot \sum_i \sum_{j \neq i} h_i a_{ij})^{-1}$.

Die Matrix

$$M = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1,20} \\ m_{21} & m_{22} & \dots & m_{2,20} \\ \vdots & \vdots & \ddots & \vdots \\ m_{20,1} & m_{20,2} & \dots & m_{20,20} \end{pmatrix},$$

beschreibt die Übergangswahrscheinlichkeiten für einen 1 PAM-Zeitschritt in folgendem Sinne: Sind $h^{(0)} = (h_1^{(0)}, \dots, h_{20}^{(0)})$ die relativen Häufigkeiten der Aminosäuren in einer Sequenz, so kann man die erwarteten Häufigkeiten $\mathbb{E}h^{(1)}$, die sich ergeben, wenn die Sequenz einen PAM-Zeitschritt evolviert, berechnen, indem man die Matrix von rechts auf den Vektor anwendet:

$$\begin{aligned} \mathbb{E}h^{(1)} &= h^{(0)} \cdot M = (h_1^{(0)}, \dots, h_{20}^{(0)}) \cdot \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1,20} \\ m_{21} & m_{22} & \dots & m_{2,20} \\ \vdots & \vdots & \ddots & \vdots \\ m_{20,1} & m_{20,2} & \dots & m_{20,20} \end{pmatrix} \\ &= \left(\sum_{i=1}^{20} h_i^{(0)} m_{i1}, \sum_{i=1}^{20} h_i^{(0)} m_{i2}, \dots, \sum_{i=1}^{20} h_i^{(0)} m_{i,20} \right) \end{aligned}$$

Um die erwarteten Häufigkeiten nach zwei Zeitschritten zu berechnen, muss man M ein weiteres Mal von rechts draufmultiplizieren: $\mathbb{E}h^{(2)} = h^{(0)} \cdot M \cdot M = h^{(0)} \cdot M^2$ usw. Allgemein gilt also, dass

die Übergangswahrscheinlichkeiten für eine Gesamtdauer von n Zeitschritten durch

$$M^n = \begin{pmatrix} m_{11}^{(n)} & m_{12}^{(n)} & \dots & m_{1,20}^{(n)} \\ m_{21}^{(n)} & m_{22}^{(n)} & \dots & m_{2,20}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ m_{20,1}^{(n)} & m_{20,2}^{(n)} & \dots & m_{20,20}^{(n)} \end{pmatrix},$$

die n-fache Potenz von M, beschrieben wird. (**Achtung:** Bitte nicht die rein symbolische Schreibweise $m_{ij}^{(n)}$ mit m_{ij}^n verwechseln!)

Nebenbemerkung: Bei dem Mutationsmodell haben wir stillschweigend angenommen (und werden dies auch weiterhin tun), dass die Wahrscheinlichkeit einer Mutation und des neuen Aminosäure-Typs nur von der gegenwärtig an der Position befindlichen Aminosäure abhängt, aber nicht von der Vergangenheit, also nicht von den Aminosäuren, die an dieser Position früher in der Evolutionsgeschichte einmal standen. Allgemein bezeichnet man einen Prozess, bei dem von Zeitschritt zu Zeitschritt zufällige Zustandsänderungen stattfinden, deren Wahrscheinlichkeiten nur vom jeweiligen gegenwärtigen Zustand abhängen, als **Markoff-Kette** (Markov chain). Die Wahrscheinlichkeiten der Zustandsänderungen kann man dann durch Übergangsmatrizen beschreiben. Wir werden in späteren Kapiteln der Vorlesung noch viel mit Markoff-Ketten zu tun haben.

Nun aber zum Score: Als Score für eine Paarung der Aminosäuren A_i und A_j in Sequenzen, die etwa n PAM-Zeitschritte voneinaner entfernt sind, käme nun $h_i m_{ij}^{(n)}$ in Frage, also die Wahrscheinlichkeit, mit der A_i an der Position in der ersten Sequenz steht und nach n Zeitschritten zu A_j geworden ist. (Beachte, dass wegen der Forderung der Reversibilität das selbe herauskommt, wenn man die Rollen der beiden Sequenzen vertauscht.) Allerdings will man diese Wahrscheinlichkeit in Bezug setzen zur Wahrscheinlichkeit, dass diese Aminosäuren an zwei Positionen stehen, die eigentlich gar nichts miteinander zu tun haben. Das ist ja die Situation, die sich ergibt, wenn die Aminosäuren in einem falschen Alignment willkürlich übereinander geschoben werden. Diese Wahrscheinlichkeit ist das Produkt der Häufigkeiten $h_i h_j$. Als Score käme also jetzt der sogenannte Likelihoodquotient $\frac{h_i m_{ij}^{(n)}}{h_i h_j} = m_{ij}^{(n)}/h_j$ in Frage, der uns sagt, um welchen Faktor dieses Paar von Aminosäuren (un-)wahrscheinlicher ist, wenn die Positionen aligniert sind, gegenüber der nicht-alignierten Situation.

Der Likelihoodquotient eines gaplosen Alignments

$$\begin{matrix} A_{i_1} & A_{i_2} & \dots & A_{i_k} \\ A_{j_1} & A_{j_2} & \dots & A_{j_k} \end{matrix}$$

(also der Quotient aus der Wahrscheinlichkeit der Aminosäurebelegung im Falle der Richtigkeit des Alignments und der Wahrscheinlichkeit der Aminosäurebelegung im Falle der Unabhängigkeit der Sequenzen) ist dann (nach elementaren Regeln der Stochastik) das Produkt der Likelihoodquotienten der einzelnen Positionen:

$$\frac{h_{i_1} m_{i_1 j_1}^{(n)} \cdot h_{i_2} m_{i_2 j_2}^{(n)} \cdots h_{i_k} m_{i_k j_k}^{(n)}}{h_{i_1} h_{j_1} \cdot h_{i_2} h_{j_2} \cdots h_{i_k} h_{j_k}} = \frac{m_{i_1 j_1}^{(n)}}{h_{j_1}} \cdot \frac{m_{i_2 j_2}^{(n)}}{h_{j_2}} \cdots \frac{m_{i_k j_k}^{(n)}}{h_{j_k}}$$

A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
9890	5	5	6	12	9	11	12	5	2	5	6	9	2	10	29	14	1	2	17
4	9907	5	2	2	16	4	3	8	1	2	30	2	0	3	5	5	4	3	2
3	4	9888	18	2	8	5	6	13	1	1	10	1	1	2	13	8	1	3	1
4	2	21	9905	0	7	28	5	6	0	0	5	0	0	3	7	5	0	1	0
3	1	1	0	9946	0	0	1	1	1	1	0	1	1	0	3	1	1	1	2
4	11	7	5	1	9856	18	2	14	1	3	14	6	1	4	5	5	1	1	2
8	5	6	30	0	28	9890	2	7	1	1	15	3	0	4	7	5	1	1	3
11	4	9	7	2	4	3	9952	3	0	1	3	1	0	2	10	2	2	1	1
1	4	7	3	1	9	3	1	9895	1	1	3	2	2	1	2	2	1	9	1
2	1	2	0	2	2	1	0	2	9878	22	2	26	7	1	1	5	2	2	42
5	4	2	0	3	8	2	1	3	35	9919	3	48	22	4	3	4	5	5	19
5	33	13	5	0	22	15	2	8	2	2	9883	5	1	4	6	9	1	2	2
3	1	1	0	2	4	1	0	2	10	12	2	9859	5	0	2	3	1	1	4
1	0	1	0	3	1	0	0	4	5	10	0	9	9923	0	1	1	10	28	3
6	2	2	2	0	5	3	1	2	1	2	3	0	1	9943	6	5	0	1	2
23	5	17	8	9	9	7	8	6	1	2	7	4	1	8	9862	32	2	4	2
11	5	11	6	4	8	5	2	7	6	2	9	7	2	7	33	9879	1	2	12
0	1	0	0	1	1	0	0	1	0	1	0	1	3	0	0	0	9956	4	0
1	2	2	1	3	1	1	0	13	1	2	1	2	22	1	2	1	10	9924	2
15	2	1	0	8	3	4	1	2	51	14	3	12	5	2	3	14	1	4	9884

Tabelle 1: PAM-1-Substitutions-Matrix M (Alle Einträge sind mit 10000 multipliziert.)

Wir wollen aber keine Score-Funktion haben, die wir von Position zu Position aufmultiplizieren müssen, sondern wir wollen eine haben, die wir von Position zu Position addieren können (und die trotzdem noch eine gewisse Interpretation hat). Also ziehen wir einfach den Logarithmus, dessen wichtigste Eigenschaft es ist, Produkte in Summen zu verwandeln:

$$\log\left(\frac{m_{i_1j_1}^{(n)}}{h_{j_1}}\cdot\frac{m_{i_2j_2}^{(n)}}{h_{j_2}}\cdots\frac{m_{i_kj_k}^{(n)}}{h_{j_k}}\right)=\log\frac{m_{i_1j_1}^{(n)}}{h_{j_1}}+\log\frac{m_{i_2j_2}^{(n)}}{h_{j_2}}+\cdots+\log\frac{m_{i_kj_k}^{(n)}}{h_{j_k}}$$

Solche log-Likelihoodquotienten spielen allgemein in der Statistik ein wichtige Rolle. Die Einträge der von Dayhoffs PAM-n-Score-Matrizen sind log-Likelihoodquotienten, die allerdings aus praktischen Gründen noch mit einem konstanten Faktor skaliert und gerundet werden:

$$s^{(n)}(A_i,A_j)=\text{round}\left(\text{const}\cdot\log\frac{m_{ij}^{(n)}}{h_j}\right)$$

Positive (bzw. negative) Werte bedeuten, dass das jeweilige Paar von Aminosäuren wahrscheinlicher (bzw. unwahrscheinlicher) ist, wenn die Positionen homolog sind, also im wahren Alignment übereinander stehen, als wenn die Positionen unabhängig voneinander sind. Matches erhalten immer positiven Score. Für kleine n erhalten Mismatches beim PAM-n-Scoreschema negativen Score, da Mutationen über kurze Zeiträume selten vorkommen. Bei großem n gibt es jedoch auch positiven Score für bestimmte Mismatches zwischen Aminosäuren, die bei Mutationen oft ineinander überführt werden vgl. Tabellen 1,2,3 und 4. **Achtung:** Wir haben hier die die Mutationsratenmatrizen von rechts auf Zeilenvektoren angewandt, so wie es heute in der Theorie der Markoffketten üblich ist, siehe Abschnitt 4.4. Traditionell werden PAM-Substitutionsmatrizen eigentlich von links auf Spaltenvektoren angewandt. Daher ergeben in PAM-Substitutionsmatrizen nicht die Zeilen,sondern die Spalten in Summe 0 (vgl. Tabellen 1 und 3).

C	17.2																			
S	-18.5	12.1																		
T	-21.6	-12.7	12.0																	
P	-33.2	-18.6	-19.5	13.4																
A	-18.1	-14.3	-17.5	-18.8	11.0															
G	-25.2	-18.7	-25.3	-24.9	-18.2	11.3														
N	-24.1	-15.5	-17.5	-24.0	-22.3	-19.1	13.4													
D	-32.1	-18.7	-20.0	-22.7	-21.2	-20.5	-14.0	12.7												
E	-35.3	-19.4	-20.8	-21.6	-18.6	-23.7	-19.5	-12.8	12.3											
Q	-28.7	-18.4	-18.9	-19.7	-19.4	-22.8	-17.4	-18.7	-13.2	14.2										
H	-22.1	-20.2	-19.7	-22.8	-22.1	-24.1	-15.3	-19.4	-19.4	-14.3	16.2									
R	-24.2	-20.4	-20.6	-23.1	-21.7	-22.4	-20.3	-24.4	-21.0	-15.2	-18.2	12.7								
K	-33.3	-19.6	-18.3	-21.7	-20.8	-23.9	-16.7	-20.4	-16.0	-14.2	-18.6	-12.5	12.3							
M	-21.6	-21.7	-19.3	-31.0	-19.3	-27.6	-25.3	-32.7	-23.6	-17.8	-21.1	-24.3	-20.7	16.4						
I	-25.5	-26.2	-20.4	-27.5	-25.0	-34.6	-25.5	-33.8	-27.2	-25.8	-25.3	-26.9	-24.2	-13.4	12.3					
L	-24.5	-25.7	-23.9	-23.8	-22.3	-31.0	-27.7	-35.3	-27.4	-21.0	-24.4	-24.0	-24.3	-12.9	-14.2	10.2				
V	-19.3	-24.3	-17.1	-24.6	-16.6	-29.5	-27.2	-33.8	-22.7	-23.9	-26.5	-24.7	-24.1	-17.5	-11.3	-17.0	11.5			
F	-21.9	-28.0	-24.8	-29.5	-25.1	-32.2	-26.5	-33.1	-32.5	-26.3	-20.5	-31.1	-30.2	-16.5	-19.4	-16.3	-21.7	13.9		
Y	-21.0	-21.9	-24.8	-26.6	-25.7	-29.6	-21.5	-25.8	-27.1	-24.1	-14.2	-23.0	-24.7	-22.7	-23.7	-22.4	-22.9	-11.7	14.8	
W	-22.4	-25.3	-28.6	-32.0	-28.5	-26.5	-28.3	-37.5	-29.5	-24.1	-22.8	-21.5	-30.5	-22.8	-25.3	-22.8	-28.5	-15.9	-15.1	18.8
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Tabelle 2: Von M abgeleitete PAM-1-Dayhoff-Score-Matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	1350	677	733	733	884	748	781	884	647	649	597	714	671	461	834	1006	899	342	468	803
R	460	1583	568	493	323	751	589	424	616	304	320	995	361	253	429	512	507	366	351	337
N	425	485	1092	747	299	534	563	496	601	239	228	552	275	225	366	558	507	197	326	271
D	501	496	881	1593	260	666	1007	549	591	221	213	604	268	188	453	597	536	164	284	273
C	212	114	124	91	2660	107	94	119	138	145	134	100	153	157	93	193	166	149	169	187
Q	359	530	442	468	215	705	558	300	496	243	261	538	303	208	359	390	379	201	253	264
E	580	645	722	1095	293	863	1334	483	635	315	305	768	370	237	525	614	572	215	312	376
G	827	583	801	752	466	585	608	3387	530	264	264	567	332	224	512	799	569	291	292	346
H	194	271	310	259	173	309	256	170	946	143	153	267	175	231	182	225	221	194	387	149
I	480	330	304	239	446	375	314	208	353	1460	1094	354	1027	726	320	379	510	381	489	1185
L	717	566	473	374	673	653	492	339	612	1779	2390	576	1798	1501	561	575	707	793	942	1443
K	535	1097	713	662	311	840	774	454	667	358	359	1240	425	276	510	600	605	264	360	394
M	195	154	138	114	185	183	144	103	170	403	435	165	608	326	130	166	198	181	217	330
F	239	193	202	143	340	224	165	124	399	510	649	191	583	2041	171	213	246	937	1312	419
P	484	367	366	385	225	434	410	318	353	251	271	396	259	191	2614	495	467	144	222	299
S	774	580	741	671	619	625	636	657	578	394	368	616	439	315	656	997	844	285	393	475
T	707	587	687	616	544	621	605	478	580	542	462	635	535	372	632	862	1101	274	397	622
W	57	90	57	40	103	70	48	52	108	86	110	59	104	301	41	62	58	3398	336	72
Y	195	215	235	173	293	220	175	130	539	276	327	200	311	1054	160	213	211	843	1955	253
V	708	437	413	352	689	486	446	326	440	1415	1060	464	1004	712	454	545	698	382	535	1501

Tabelle 3: PAM-250-Substitutions-Matrix M^{250} (Alle Einträge sind mit 10000 multipliziert.)

C	11.5																			
S	0.1	2.2																		
T	-0.5	1.5	2.5																	
P	-3.1	0.4	0.1	7.6																
A	0.5	1.1	0.6	0.3	2.4															
G	-2.0	0.4	-1.1	-1.6	0.5	6.6														
N	-1.8	0.9	0.5	-0.9	-0.3	0.4	3.8													
D	-3.2	0.5	-0.0	-0.7	-0.3	0.1	2.2	4.7												
E	-3.0	0.2	-0.1	-0.5	-0.0	-0.8	0.9	2.7	3.6											
Q	-2.4	0.2	0.0	-9.2	-0.2	-1.0	0.7	0.9	1.7	2.7										
H	-1.3	-0.2	-0.3	-1.1	-0.8	-1.4	1.2	0.4	0.4	1.2	6.0									
R	-2.2	-0.2	-0.2	-0.9	-0.6	-1.0	0.3	-0.3	0.4	1.5	0.6	4.7								
K	-2.8	0.1	0.1	-0.6	-0.4	-1.1	0.8	0.5	1.2	1.5	0.6	2.7	3.2							
M	-0.9	-1.4	-0.6	-2.4	-0.7	-3.5	-2.2	-3.0	-2.0	-1.0	-1.3	-1.7	-1.4	4.3						
I	-1.1	-1.8	-0.6	-2.6	-0.8	-4.5	-2.8	-3.8	-2.7	-1.9	-2.2	-2.4	-2.1	2.5	4.0					
L	-1.5	-2.1	-1.3	-2.3	-1.2	-4.4	-3.0	-4.0	-2.8	-1.6	-1.9	-2.2	-2.1	2.8	2.8	4.0				
V	-0.0	-1.0	0.0	-1.8	0.1	-3.3	-2.2	-2.9	-1.9	-1.5	-2.0	-2.0	-1.7	1.6	3.1	1.8	3.4			
F	-0.8	-2.8	-2.2	-3.8	-2.3	-5.2	-3.1	-4.5	-3.9	-2.6	-0.1	-3.2	-3.3	1.6	1.0	2.0	0.1	7.0		
Y	-0.5	-1.9	-1.9	-3.1	-2.2	-4.0	-1.4	-2.8	-2.7	-1.7	2.2	-1.8	-2.1	-0.2	-0.7	-0.0	-1.1	5.1	7.8	
W	-1.0	-3.3	-3.5	-5.0	-3.6	-4.0	-3.6	-5.2	-4.3	-2.7	-0.8	-1.6	-3.5	-1.0	-1.8	-0.7	-2.6	3.6	41	14.2
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Tabelle 4: Von M^{250} abgeleitete PAM-250-Dayhoff-Score-Matrix

1.2.2 BLOSUM-Matrizen

Hinter den PAM-Matrizen steckt die Idee, das Substitutionsgeschehen über sehr kurze Zeiträume zu betrachten (praktisch bedeutet das, sehr ähnliche Sequenzen zum Schätzen der Substitutionswahrscheinlichkeiten zu benutzen) und die Schätzungen dann bei Bedarf mittels Matrix-Multiplikation auf längere evolutionäre Zeiträume hochzurechnen. Nun ist aber nicht auszuschließen, dass bei relativ unähnlichen Sequenzen evolutionäre Mechanismen eine Rolle gespielt haben, die sich qualitativ von denen zwischen sehr ähnlichen Sequenzen unterscheiden. Die BLOSUM-Score-Matrizen von Henikoff und Henikoff (1992) wurden daher mit Datensätzen von Proteinsequenzen konstruiert, die größere Unterschiede aufweisen. Auch die BLOSUM-Matrizen enthalten geschätzte log-Likelihoodquotienten. Die Datensätze lagen wieder als gaplos alignierte Blöcke vor, wobei man sich bzgl. des Alignments wegen der größeren Unterschiede zwischen den Sequenzen nicht ganz so sicher war wie bei der Konstruktion der PAM-Matrizen.

Da in den Datensätzen oft Sequenzen mehrfach vorkommen, evtl. mit leichten Veränderungen, werden sehr ähnliche Sequenzen geclustert und wie eine einzelne Sequenz gewichtet. Bei der Konstruktion der BLOSUM-n-Matrix werden Sequenzen zusammengefasst, wenn sie in mindestens n% der Positionen übereinstimmen.

Wenn es z.B. um BLOSUM-75 geht und der Datensatz so aussieht

```
V  I  I  L
V  I  I  I
L  I  V  V
L  L  V  I
```

fassen wir die ersten beiden Zeilen zu einem Cluster zusammen:

```
V  I  I  I,L
L  I  V   V
L  L  V   I
```

Wir behandeln den Cluster wie eine Sequenz, bei der an der letzten Position jeweils zur Hälfte `I` und `L` steht. Wenn wir nun aus den insgesamt zwölf Positionen die relativen Aminosäure-Häufigkeiten schätzen, so erhalten wir also $h_{\mathtt{L}} = \frac{3{,}5}{12}$, $h_{\mathtt{I}} = \frac{4{,}5}{12}$ und $h_{\mathtt{V}} = \frac{4}{12}$. Für die Häufikeiten mit denen die Aminosäure-Paare im Datensatz beobachtet werden, erhalten wir entsprechend: $h_{\mathtt{LL}} = \frac{1}{12}$, $h_{\mathtt{LV}} = \frac{2{,}5}{12}$, $h_{\mathtt{LI}} = \frac{2{,}5}{12}$, $h_{\mathtt{VV}} = \frac{1}{12}$, $h_{\mathtt{VI}} = \frac{3{,}5}{12}$ und $h_{\mathtt{II}} = \frac{1{,}5}{12}$. (Dass auch hier wieder 12 als Nenner auftaucht, ist ein dummer Zufall. Es sind halt 3 Sequenzen und die Anzahl der Paare pro Position ist damit $\binom{3}{2} = 3$, sorry!) Die beobachteten Häufigkeiten verwenden wir als Schätzungen für die Wahrscheinlichkeit, in einer (anderen) Sequenz an einer Stelle die entsprechende Aminosäure steht bzw. dass bei einem homologen Paar von Positionen zweier Sequenzen die beiden Aminsäuren stehen.

Als Score schlagen Henikoff und Henikoff (1992) dann den sich aus diesen Schätzungen ergebenden log-Likelihoodquotienten vor (mit Vorfaktor 2 und Basis 2 des log):

$$s(a,b) = \begin{cases} 2\log_2 \frac{h_{aa}}{h_a^2} & \text{falls} \quad a = b \\ 2\log_2 \frac{h_{ab}}{2h_a h_b} & \text{falls} \quad a \neq b \end{cases}$$

Dabei taucht der Faktor 2 im Nenner des zweiten Bruchs auf, da wir nicht zwischen ab und ba unterscheiden und beide Fälle in h_{ab} zählen.

Die von Henikoff und Henikoff (1992) verwendeten Daten kamen aus öffentlichen Datenbanken. Die ersten daraus geschätzten Score-Matrizen wurden dann verwendet, um das Alignment noch einmal zu verfeinern, und damit wurden dann auch die Score-Matrizen noch einmal neu geschätzt. Dies wurde dreimal iteriert.

Man beachte einen Unterschied bei der Bezeichnung von PAM- und BLOSUM-Matrizen: Die Zahl im Namen der zu verwendenen Matrix sollte bei PAM umso größer und bei BLOSUM umso kleiner sein, je verschiedener die zu alignierenden Sequenzen sind.

1.3 Mustersuche und BLAST

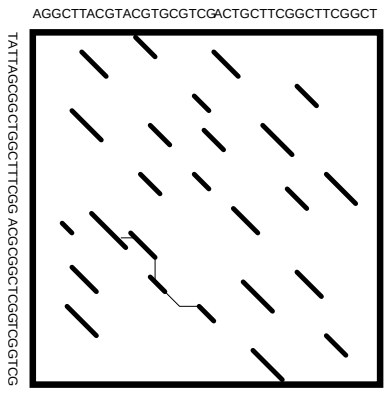
1.3.1 Die Suchstrategie von BLAST

Gegeben seien zwei sehr lange Sequenzen S und T der Längen n und m über dem Alphabet $\mathcal{A}$ und ein Score-Schema mit affinen Gap-Penalties, das die Forderungen erfüllt, die wir in Abschnitt 1.1.2 an Score-Schemata für lokale Alignments gestellt haben. Man denke an sehr lange Sequenzen, z. B. sei T die Aneinanderreihung der Sequenzen einer ganzen Datenbank.

Gesucht sei ein lokales Alignment mit möglichst hohem Score.

Diesen könnten wir mit dem Smith-Waterman-Algorithmus finden, der jedoch mit Laufzeit $O(nm)$ zu langsam für Datenbankvergleiche im großen Stil ist. Im Programm-Paket BLAST, dessen ursprüngliche Version in Altschul et. al (1990), dem inzwischen meistzitiertem biologischen Paper, vorgestellt wird, wird die Stategie verfolgt, zunächst mal gute gaplose Alignments zu finden, was wesentlich schneller geht, und diese dann zu noch besseren Alignments zu erweitern oder zusammenzusetzen, die dann evtl. auch Gaps enthalten. Damit ist nicht garantiert, dass tatsächlich das optimale Alignment gefunden wird, aber in der Praxis funktioniert es offensichtlich zur Zufriedenheit der meisten Anwender.

Oft genug enthalten wohl die besten Alignments tatsächlich auch ein recht gutes gaploses Alignment. Interessant sind in diesem Zusammenhang insbesondere die **High-score Segment-Paare (HSP)**. Das sind lokale Alignments, die einen gewissen Mindestscore überschreiten und die in dem Sinne lokal optimal sind, als dass sich ihr Score verschlechtern würde, wenn man sie in der einen oder anderen Richtung verlängern würde, und wenn sich der Score auch nicht verbessern ließe, indem man sie verkürzt. Das HSP mit dem höchsten Score bezeichnet man auch als **Maximales Segment-Paar (MSP)**.



Strategie von BLAST (klassische Version)

1. Ist S gegen eine sehr lange Sequenz T zu alignieren, so wird zunächst eine Liste aller Wörter in S (also zusammenhängender Teilsequenzen von S) der Länge w erstellt. (Default-Werte sind $w = 11$ bei DNA und $w = 3$ bei Proteinen.) Zumindest bei Proteinen wird für jedes solche Wort W auch jedes Wort V der Länge w in die Liste geschrieben, welches W ähnlich ist, d. h. sich mit einem Score oberhalb einer vorgegeben Schranke t gegen W gaplos alignieren lässt.

2. Baue für die Liste der Wörter einen endlichen Automaten (genauer gesagt einen Mealy-Automaten, vgl. Durbin et al., 1998) ähnlich dem Schlüsselwort-Baum im Aho-Corasick-Algorithmus; Das geht fast in linearer Zeit. Mehr dazu im Abschnitt 1.3.3.

3. Schiebe die Datenbank durch den Automaten und finde so alle w-langen Wörter, die mit w-langen Wörtern in S mit Score $\geq t$ alignierbar sind. Auch dazu mehr in Abschnitt 1.3.3.

4. Verlängere die gefundenen Alignments vom Score $\geq t$ zu HSPs. Verlängere sie dazu nach links und rechts, bis der Score unter eine gewisse Schranke fällt, und optimiere dann das Alignment in diesem Bereich. Gib das HSP aus, falls der Score eine Schranke C übersteigt.

Gapped BLAST Altschul et al. (1997) stellen eine Variante von BLAST vor, die zum einen schneller ist und zum anderen auch bessere Chancen hat, Alignments mit Gaps zu finden. Im Wesentlichen wurden zum klassischen BLAST zwei Veränderungen vorgenommen: Zum einen werden nur solche HSPs weiterverfolgt, die in der Nähe eines weiteren HSPs auf derselben "Diagonallinie" im Alignment-Graphen liegen, die also beide Teil eines gemeinsamen gaplosen Alignments sind, und zum anderen werden solche HSPs dann mittels dynamischer Programmierung innerhalb einer gewissen Umgebung zu lokalen Alignments mit Gaps erweitert.

1.3.2 Ein Muster in einem Text suchen: Der Knuth-Morris-Pratt-Algorithmus

Bevor wir uns dem Teilproblem von BLAST zuwenden, alle Stellen in einer langen Sequenz zu finden, an denen ein Wort aus einer vorgegebnen Menge beginnt, betrachten wir als Vorüberlegung das **Pattern Matching Problem**:

Gegeben ist ein Wort (in diesem Kontext auch Pattern oder Muster genannt) $S = (S_1, S_2, \ldots, S_n) \in \mathcal{A}^n$ und ein Text $T = (T_1, \ldots, T_m) \in \mathcal{A}^m$.

Gesucht $i \leq m - n + 1$ mit $S_1 = T_i$, $S_2 = T_{i+1}$,..., $S_n = T_{i+n-1}$, falls es existiert.

Idee des Knuth-Morris-Pratt Algorithmus (KMP): Es sei $S^q := (S_1, \ldots, S_q)$ der Präfix von S der Länge $q \leq n$. Angenommen S^q matcht, S^{q+1} jedoch nicht:

$$\begin{array}{cccccccccc} T_1 & \ldots & T_{i-1} & T_i & T_{i+1} & \ldots & T_{i+q-1} & T_{i+q} & \ldots & T_m \\ & & & = & = & \cdots & = & \neq & & \\ & & & S_1 & S_2 & \ldots & S_q & S_{q+1} & & \end{array}$$

Dann braucht man S erst wieder mit einem T_{i+j}, T_{i+j+1},..., $T_{i+j+n-1}$ zu vergleichen, falls der Präfix von S der Länge $q-j$ gleichzeitig Suffix von S^q ist:

$$\begin{array}{cccccccccccc} T_1 & \dots & T_{i-1} & T_i & T_{i+1} & \dots & T_{i+j} & T_{i+j+1} & \dots & T_{i+q-1} & T_{i+q} & \dots & T_m \\ & & & = & = & \cdots & = & = & \dots & = & & & \\ & & & S_1 & S_2 & \dots & S_{j+1} & S_{j+2} & \dots & S_q & & & \\ & & & & & & = & = & \dots & = & & & \\ & & & & & & S_1 & S_2 & \dots & S_{q-j} & & & \end{array}$$

Die Anzahl an Positionen, um die wir S längs T verschieben können bevor wir wieder vergleichen, ist also $q - \pi(q)$, wobei:

$$\pi(q) = \max\{k \mid k < q, S^k \text{ ist Suffix von } S^q\}$$

Nach der Verschiebung vergleichen wir $S_{\pi(q)+1}$ mit T_{i+q} und bei Erfolg $S_{\pi(q)+2}$ mit T_{i+q+1} usw. Passt $S_{q'+1}$ als erster nicht mehr, so verschieben wir um $q' - \pi(q')$.

Angenommen, die Abbildung $\pi : q \mapsto \pi(q)$ ist bekannt. Dann kommen wir nach jedem Vergleich in gewissem Sinne in T weiter nach rechts: Bei "Match" wird die Stelle, an der wir vergleichen, um eins nach rechts verschoben und bei "Mismatch" bleibt sie, wo sie ist, aber das Muster S wird dann längs T nach rechts verschoben, was aber höchstens so oft passieren kann, bis der S_1 an der Stelle von T steht, wo verglichen wird. Also kommen wir mit weniger als $2m$ Vergleichen aus.

Zuvor müssen wir π berechnen. Wir verwenden dabei die selben Präfix-Suffix-Ideen und benutzen zur Berechnung von $\pi(q)$ die Werte $\pi(1), \dots, \pi(q-1)$, die wir zuvor berechnet haben. (Die dynamische Programmierung lässt grüßen!)

$\pi(1) = 0$
`Für` $q := 1, \dots, n-1$`:`
    $x := S_{q+1}$
    $v := \pi(q)$
    `Solange` $S_{v+1} \neq x$ `und` $v \neq 0$`:`
        $v := \pi(v)$
    `Falls` $S_{v+1} = x$`:`
        $\pi(q+1) := v + 1$
    `sonst :`
        $\pi(q+1) := 0$

Zur Laufzeit der Berechnung von π muss man sich überlegen, wie oft die Körper der äußeren und der inneren Schleife durchlaufen werden. Das ist äquivalent zur Frage, wie oft v seinen Wert ändert. Bei der Zuweisung $v := \pi(q)$ kann v höchstens um 1 erhöht werden, und das passiert höchstens $n-1$ mal. Bei der Zuweisung $v := \pi(v)$ wird v hingegen kleiner. Da v zu Beginn 0 ist und nie kleiner als 0 werden kann, kann die Zuweisung $v := \pi(v)$ nicht öfter als die Zuweisung $v := q$ aufgerufen werden. Also ist die Laufzeit der Berechnung $O(n)$.

Damit ist gezeigt: KMP findet ein Muster der Länge n in einem Text der Länge m mit Laufzeit $O(n+m)$.

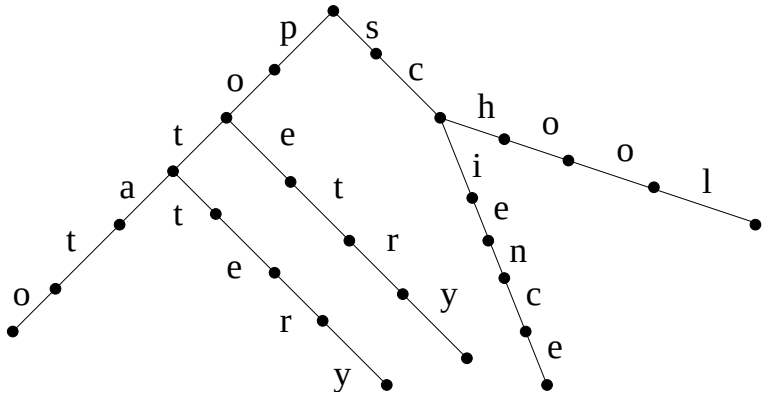


Abbildung 2: Schlüsselwort-Baum für $\mathcal{P} = \{\text{potato, poetry, pottery, science, school}\}$

1.3.3 Mehrere Muster in einem Text suchen: Der Aho-Corasick-Algorithmus

Zu suchen sind alle Stellen in einem Text T der Länge m, an denen ein Muster aus einer Menge $\mathcal{P} = \{P_1, P_2, \ldots, P_z\}$, $P_i \in \mathcal{A}^*$, vorkommt. Indem man für jedes Muster einen Linearzeit-Algorithmus wie den von Knuth, Morris und Pratt anwendet, gelingt dies in Zeit $O(n+zm)$, wobei n die Gesamtlänge der Muster in $\mathcal{P}$ ist. In diesem Abschnitt werden wir sehen, dass es auch in Zeit $O(n+m+k)$ geht, wobei k die Anzahl der zu findenden Textstellen ist. Das geht z. B. mit dem Algorithmus von Aho und Corasick (1975), bei dem man zunächst einen Schlüsselwort-Baum (keyword tree) für die Muster konstruiert und dann ähnliche Ideen wie bei Knuth-Morris-Pratt anwendet.

Ein **Schlüsselwort-Baum** (keyword tree) $\mathcal{K}$ für die Menge $\mathcal{P}$ ist ein gewurzelter gerichteter Baum, bei dem jede Kante mit einem $a \in \mathcal{A}$ beschriftet ist, so dass man für jedes Muster $P_i \in \mathcal{P}$ ein Blatt von $\mathcal{K}$ findet, so dass die Kanten von der Wurzel bis zum Blatt gerade P_i buchstabieren, und dies muss eine Bijektion zwischen $\mathcal{P}$ und der Menge der Blätter des Baumes definieren. Außerdem müssen je zwei verschiedene Kanten, die aus demselben Knoten des Baumes wachsen, jeweils mit verschiedenen Elementen von $\mathcal{A}$ beschriftet sein.

Das Beispiel in Abbildung 2 ist (wie auch Abbildung 3) dem Buch von Gusfield (1999) entnommen.

Ein Schlüsselwort-Baum für $\mathcal{P}$ kann auf die offensichtliche Art und Weise in Zeit $O(n)$ konstruiert werden. Ein Schlüsselwort-Baum kann dazu benutzt werden, um in Zeit $O(n)$ zu überprüfen, ob eines der Muster an einer bestimmten Position des Textes vorkommt. Wenn wir das mit jeder Position tun wollen, erhalten wir aber einen $O(nm)$-Algorithmus. Zur Beschleunigung fließen nun noch Ideen aus dem KMP-Algorithmus ein.

Wir nehmen der Einfachheit halber folgendes an: KEIN MUSTER IN $\mathcal{P}$ IST TEILSTRING EINES ANDEREN MUSTERS IN $\mathcal{P}$.

Wir bezeichnen die Knoten des Schlüsselwort-Baums im Folgenden mit $\mathcal{K}$ und dessen Wurzel mit r. Das Wort, das man erhält, wenn man die Kanten von r bis zu einem Knoten $v \in \mathcal{K}$ abliest, bezeichnen wir mit $\mathcal{L}(v)$. Wenn W Suffix von $\mathcal{L}(v)$ und zugleich Präfix eines Musters in $\mathcal{P}$ ist, dann existiert genau ein $w \in \mathcal{K}$ mit $W = \mathcal{L}(w)$. Ist $\mathcal{L}(w)$ der längste Suffix von $\mathcal{L}(v)$, der auch Präfix eines Musters in $\mathcal{P}$

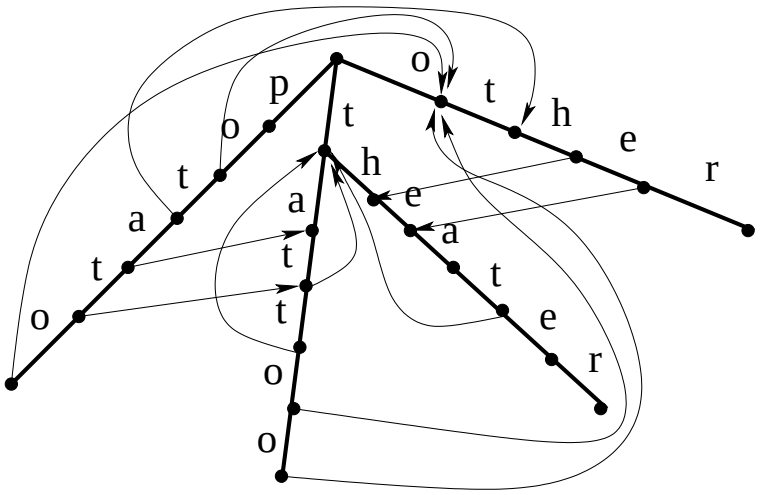


Abbildung 3: Schlüsselwort-Baum für $\mathcal{P} = \{\text{potato, tattoo, theatre, other}\}$ mit Failure-Links $(v, f(v))$ (außer solchen, bei denen $f(v)$ die Wurzel ist)

ist, so setzen wir $f(v) := w$. Falls kein solches w existiert, setzen wir $f(v) := r$. Wir können dann in den Schlüsselwort-Baum zusätzliche Kanten $(v, f(v))$ einzeichnen, die als "failure links" bezeichnet werden, vgl. Abbildung 3. (Mit diesen Kanten ist der Schlüsselwort-Baum dann natürlich kein Baum mehr.)

Mit $t(v)$ bezeichnen wir die Tiefe von $v \in \mathcal{K}$, d. h. die Anzahl der Kanten zwischen r und v.

Die Failure-Links können wir dann analog zur Funktion π des KMP-Algorithmus benutzen und erhalten:

Aho-Corasick-Suche (AC)

$l := 1$
$c := 1$
$w := r$
`wiederhole:`
    `Falls` $w = r$`:`
        `solange keine Kante` (w, w') `mit Beschriftung` T_c `existiert:`
            $c := c + 1$
            $l := l + 1$
    `solange eine Kante` (w, w') `mit Beschriftung` T_c `existiert:`
        `Falls` w' `Blatt ist:`
            `gib` l `und das entsprechende Muster aus`
        $w := w'$
        $c := c + 1$
    $w := f(w)$

$l := c - t(w)$

`bis` $c > m$

Dabei ist c jeweils die aktuelle Vergleichs-Position in T und l die Position, die dem Beginn des entsprechenden Musters und damit der Wurzel des Baums entspricht.

Laufzeit der AC-Suche Wegen $t(f(w)) < t(w)$ wird bei jedem Aufruf der inneren oder der äußeren Schleife c oder l erhöht. Da immer gilt $0 \leq l \leq c \leq m$ ist die Laufzeit $O(m)$.

Berechnung von $f : v \mapsto f(v)$

```
Für i = 1, 2, ..., max_{v∈K} t(v):
    Für alle v ∈ K mit t(v) = i:
        v' := Vater von v
        x := Beschriftung der Kante (v', v)
        w := f(v')
        Solange ∄ Kante (w, u) mit Beschriftung x und w ≠ r:
            w := f(w)
        Falls ∃ Kante (w, u) mit Beschriftung x:
            f(v) := u
        Sonst:
            f(v) := r
```

Laufzeit der Berechnung von f Sei $P \in \mathcal{P}$ und $v_1, \ldots, v_y$ die dazugehörigen Knoten. Wir betrachten nur die Durchläufe der inneren "Für"-Schleife mit $v = v_1, \ldots, v_y$. Speziell betrachten wir $t(w)$ jeweils nach Aufruf von $w := f(v')$ und $w := f(w)$. Beim Durchlauf mit $v = v_i$ kann $t(w)$ um 1 größer sein als bei $v = v_{i-1}$, und zwar falls $f(v) := u$ ausgeführt wurde, ohne dass die Solange-Bedingung erfüllt war. Sonst wird $t(w)$ immer kleiner. Wegen $t(w) \geq 0$ und da $f(v) := u$ entlang $v_1, \ldots, v_y$ nur $O(y)$-oft ausgeführt wird, folgt: Es werden $O(y)$ Rechenschritte bei den Aufrufen der Für-Schleifen mit $v = v_1, \ldots, v_y$ ausgeführt, und damit $O(n)$ Schritte für alle $v \in \mathcal{K}$.

Falls $\mathcal{P}$ Muster enthält, die Teilstrings anderer Muster in $\mathcal{P}$ sind, werden im Suchbaum zusätzliche gerichtete Kanten eingefügt, die bei der Ausgabe eines Strings benutzt werden, um auch die entsprechenden Teilstrings auszugeben. Details siehe Gusfield (1999). Ist k die Anzahl der Positionen, an denen in T ein Muster aus $\mathcal{P}$ beginnt, so ist auch in diesem Fall die Laufzeit der AC-Suche $O(n + m + k)$ (einschließlich der Berechnung von f).

1.4 Signifikanz lokaler Alignments

1.4.1 Fragestellung und Allgemeines zur Signifikanz

Zwei DNA- oder Proteinsequenzen $X = (X_1, \ldots, X_n)$ und $Y = (Y_1, \ldots, Y_m)$ werden mit einem bestimmten Score-Schema lokal aligniert. Das beste lokale Alignment hat einen Score b. Bedeutet das was? Ist das signifikant? Oder sieht das eher nach Zufall aus? Wir nehmen zunächst an, dass wir nur gaplose Alignments zulassen.

Wir verfolgen den klassischen Ansatz der Statistik: Angenommen, die Sequenzen wären unabhängig. Wir versuchen zu zeigen, dass ein so extremes Ergebnis (also ein Score so hoch wie b – oder gar noch höher), unglaublich unwahrscheinlich ist. Um das rechnerisch zu behandeln brauchen wir ein Modell:

Modell: Es gibt eine Wahrscheinlichkeitsverteilung $(p_i)_{i \in \mathcal{A}}$ auf unserem Alphabet $\mathcal{A} = \{A_i\}$ (z.B. (p_1, p_2, p_3, p_4) für Nukleotide oder $(p_1, \ldots, p_{20})$ für Aminosäuren). Alle Positionen sind stochastisch unabhängig, d.h. die Wahrscheinlichkeit der Sequenz $A_{i_1} A_{i_2} \ldots A_{i_n}$ ist $p_{i_1} \cdot p_{i_2} \cdots p_{i_n}$.

Null-Hypothese: Diese möchten wir gerne verwerfen. Sie lautet: Die Sequenzen sind voneinander unabhängig, d.h. für alle $k \leq n$, $l \leq m$ und $A_i, A_j \in \mathcal{A}$ gilt:

$$\mathrm{Ws}(X_k = A_i, Y_l = A_j) = p_i \cdot p_j$$

Alternative: Die möchten wir gerne glauben, wenn es uns gelingt, die Null-Hypothese zu verwerfen. Sie lautet: Es gibt ein "wahres" lokales (gaploses, wie wir zunächst annehmen werden) Alignment von X und Y. Für entsprechende Positionspaare (k, l) gilt:

$$\mathrm{Ws}(X_k = A_i, Y_l = A_j) = q(i, j) \neq p_i \cdot p_j$$

Strategie: Gehe von der Null-Hypothese aus und zeige, dass dann Score $\geq b$ unglaublich unwahrscheinlich ist. Berechne dazu den ***p*-Wert**, also die Wahrscheinlichkeit unter der Annahme der Null-Hypothese, dass ein mindestens so extremes Ergebnis wie das beobachtete (in unserem Fall also ein maximaler Score $\geq b$) auftritt. Falls der p-Wert kleiner als das **Signifikanzniveau** α ist (oft wählt man $\alpha = 5\%$), wird die Nullhypothese verworfen, und (vielleicht) die Alternative geglaubt. Dies ist das allgemeine Prinzip des statistischen Hypothesentests.

Speziell bei lokalen Alignments misst man die Signifikanz oft statt mit p-Werten mit sogenannten ***e*-Werten**. Das ist, grob gesagt, die erwartete Anzahl nichtüberlappender Alignments vom Score $\geq b$.

1.4.2 e-Werte für hohe Scores bei HSPs

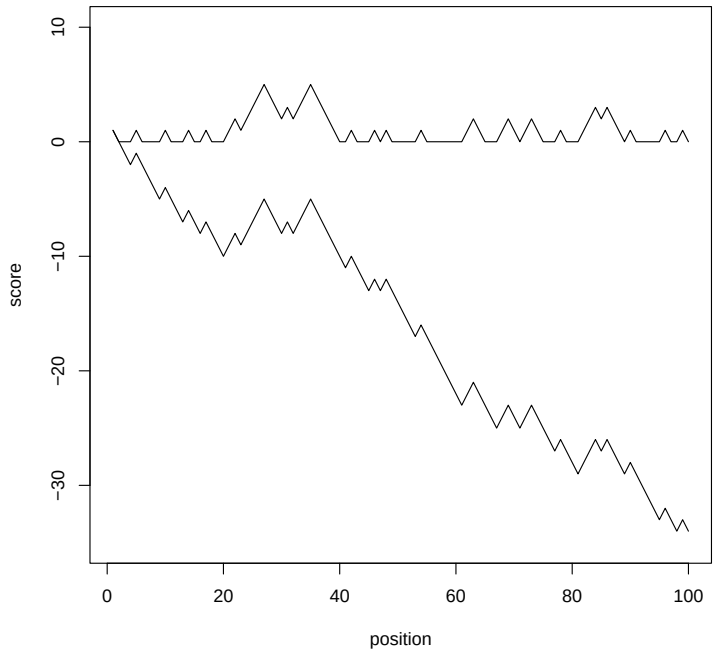
Wir betrachten zunächst nur gaplose Alignments, also HSPs. Da HSPs per definitionem nicht überlappen, bietet es sich an, den e-Wert als die erwartete Anzahl der HSPs vom Score $\geq b$ zu definieren.

Wir legen jetzt die Null-Hypothese zu Grunde. Wenn wir an einer wahllos herausgegriffenen Stelle (i, j) des Alignment-Graphen beginnen, gaplos zu alignieren, also längs einer Diagonalen des Alignmentgraphen

$$\begin{array}{cccc} X_i & X_{i+1} & X_{i+2} & \dots \\ Y_j & Y_{j+1} & Y_{j+2} & \dots \end{array}$$

so sind die Score-Zuwächse, die wir beobachten können, voneinander stochastisch unabhängig (zumindest längs einer Diagonalen) und identisch verteilt: Mit Wahrscheinlichkeit $p_k \cdot p_l$ kommt im Alignment als nächstes $\begin{smallmatrix} A_k \\ A_l \end{smallmatrix}$ und dann wird zum Score $s(k, l)$ hinzuaddiert. (Gemeint ist hier der Score des Alignments ab (i, j); wir fangen also *nicht* neu an, wenn der Score unter 0 fällt.) Ein solcher Prozess mit unabhängigen, identisch verteilten Zuwächsen wird in der Stochastik als *Irrfahrt* (random walk) bezeichnet.

Wir betrachten zunächst ein sehr einfaches Beispiel. Angenommen wir unterscheiden nur zwischen Match und Mismatch. Die Belohnung für einen Match beträgt $s(i, i) = 1$ und die Strafe für einen Mismatch $s(i, j) = -1$ (für $i \neq j$). Die Irrfahrt geht also mit Wahscheinlichkeit $p := \sum_i p_i^2$ um 1 nach oben und mit Wahrscheinlichkeit $1 - p$ um 1 nach unten. Einen typischen Verlauf einer solchen Irrfahrt zeigt die untere Linie in der folgenden Abbildung (für $p = 0.4$):



Als **Leiterpunkte** bezeichnen wir alle Stellen, an denen die Irrfahrt einen so niedrigen Wert annimmt wie nie zuvor. Ein Abschnitt der Irrfahrt zwischen zwei Leiterpunkten wird als **Exkursion** bezeichnet. Häufig folgen zwei Leiterpunkte direkt aufeinander. Dann hat die Exkursion die Länge 0. Man kann die Irrfahrt als Aneinanderreihung voneinander stochastisch unabhängiger Exkursionen auffassen. Irrfahrten wurden in der Stochastik sehr gut untersucht (siehe z.B. Feller, 1968). Die Resultate müssen

also “nur” auf die Score-Irrfahrten übertragen werden. Die obere Linie in der Abbildung ist aus der Irrfahrt entstanden, indem der Prozess in jedem Leiterpunkt auf den Wert 0 gesetzt wurde. Wir sehen also die Exkursionen der Irrfahrt oberhalb der x-Achse. Die Irrfahrt entspricht dem Verlauf des Scores eines gaplosen Alignments längs einer Diagonale im Alignment-Grafen bei festem Startpunkt. Die obere Linie zeigt für jeden Punkt auf der Diagonalen den Score des jeweils besten lokalen Alignments. Man mache sich klar, dass alle High-Scored Segment Pairs (HSPs) gerade den Abschnitten entsprechen, die in einem Leiterpunkt beginnen und in einem Maximum der entsprechenden Exkursion enden. Die Frage wie wahrscheinlich es ist, dass ein HSP den Score b erreicht, entspricht also der Frage wie wahrscheinlich es ist, dass eine Exkursion die Höhe b erreicht. Wir behandeln diese Frage hier nur für den oben beschriebenen einfachen Fall mit $s(i,j) = -1$ (für $i \neq j$).

Sei w_h die Wahrscheinlichkeit, dass eine solche bei h startende Irrfahrt den Wert b vor dem Wert a erreicht. Dann gilt offensichtlich $w_b = 1$, $w_a = 0$ und

$$(*) \qquad w_h = p \cdot w_{h+1} + (1-p) \cdot w_{h-1} \quad \text{für } a < h < b,$$

denn mit Wahrscheinlichkeit p wird von h aus als nächstes $h+1$ erreicht und dann ist w_{h+1} die Wahrscheinlichkeit für das Ereignis und entsprechendes gilt für $h-1$, welches mit Wahrscheinlichkeit $1-p$ als nächstes erreicht wird. Einen solchen Ansatz nennen wir “Zerlegung nach dem ersten Schritt”. Wir können diese Gleichung auch als homogene Differenzengleichung umschreiben:

$$w_{h+1} - w_h = \frac{1-p}{p} \cdot (w_h - w_{h-1})$$

Also verändert sich die Steigung von w_h von Schritt zu Schritt um einen konstanten Faktor. Das kennen wir vom exponentiellen Wachstum und wir kommen somit zu dem Ansatz $w_h = k \cdot e^{\lambda h} + c$ für geeignte Konstanten λ, k, c. Setzen wir das in $(*)$ ein, so erhalten wir:

$$k \cdot e^{\lambda h} + c = p \cdot (k \cdot e^{\lambda(h+1)} + c) + (1-p) \cdot (k \cdot e^{\lambda(h-1)} + c)$$

Wir ziehen davon auf jeder Seite c ab, teilen durch $k \cdot e^{\lambda h}$ und erhalten:

$$1 = p \cdot e^{\lambda} + (1-p) \cdot e^{-\lambda}$$

Das lässt sich umformen in:

$$0 = p \cdot (e^{\lambda})^2 - e^{\lambda} + 1 - p$$

Diese quadratische Gleichung in $x = e^{\lambda}$ hat die beiden Lösungen $e^{\lambda} = 1$ und $e^{\lambda} = \frac{1-p}{p}$. Das Erste ist gleichbedeutend mit $\lambda = 0$ und das Zweite mit $\lambda = \log \frac{1-p}{p}$. Im Folgenden meinen wir mit λ den letzten Ausdruck. Hat eine homogene Differenzengleichung zwei verschiedene Lösungen u_h und v_h, so bilden die Linearkombinationen $c_1 \cdot u_h + c_2 \cdot v_h$ den Lösungsraum der Gleichung. Die Lösungen von $(*)$ sind also die Ausdrücke der Form

$$w_h = C_1 \cdot 1 + C_2 \cdot e^{\lambda h} \qquad \text{mit Konstanten } C_1, C_2$$

Indem wir C_1 und C_2 so wählen, dass die Randbedingungen $w_a = 0$ und $w_b = 1$ erfüllt sind, erhalten wir die eindeutige Lösung

$$w_h = \frac{e^{\lambda h} - e^{\lambda a}}{e^{\lambda b} - e^{\lambda a}}.$$

Setzen wir nun für a den Wert -1 und für b eine sehr große Zahl ein, so gilt:

$$w_0 = \frac{e^0 - e^{-\lambda}}{e^{\lambda b} - e^{-\lambda}} \sim (1 - e^{-\lambda}) \cdot e^{-\lambda b}$$

Hier und im Folgenden bedeutet "$\sim$", dass der Quotient der beiden Seiten gegen 1 konvergiert (hier falls b gegen ∞ geht).

Um nun die Wahrscheinlichkeit W_b zu berechnen , dass in einem Positionenpaar (i, j) ein HSP vom Score $\geq b$ beginnt, müssen wir das obige Ergebnis noch mit der Wahrscheinlichkeit multiplizieren, dass (i, j) ein Leiterpunkt ist. Asymptotisch erhalten wir also einen Ausdruck der Form $k \cdot e^{-\lambda b}$.

Dembo, Karlin und Zeitouni (1994) konnten zeigen, dass für alle praktisch bedeutsamen Score-Schemata Konstanten k und λ existieren, so dass die Asymptotik $W_b \sim k \cdot e^{-\lambda b}$ gilt, oder zumindest Konstanten k_1, k_2 und λ, so dass gilt:

$$k_1 \cdot e^{-\lambda b} \lesssim W_b \lesssim k_2 \cdot e^{-\lambda b}$$

Auch im allgemeinen Fall lässt sich λ zurch Zerlegung nach dem ersten Schritt ermitteln. Das sieht dann so aus:

$$k \cdot e^{-\lambda b} \sim \sum_{i,j \in \mathcal{A}} p_i \cdot p_j \cdot k \cdot e^{-\lambda \cdot (b - s(i,j))}$$

Das kann man umformen in

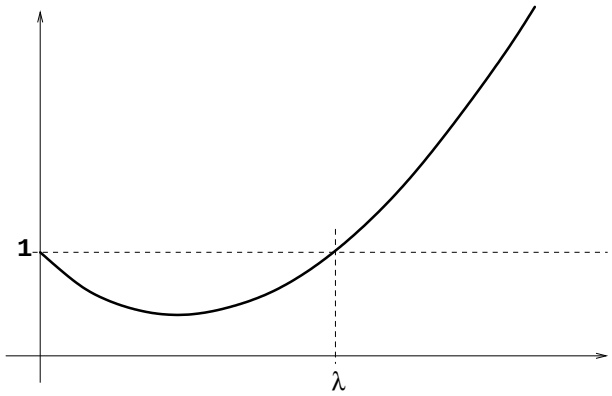
$$1 = \sum_{i,j \in \mathcal{A}} p_i \cdot p_j \cdot e^{\lambda \cdot s(i,j)}$$

und numerisch lösen, z.B. mit dem Newton-Verfahren (vgl. Anhang B).

Für $f(x) := \sum_{i,j} p_i p_j e^{x \cdot s(i,j)}$ rechnet mal leicht folgende Gleichungen nach:

$$\begin{aligned}
f(0) &= \sum_{i,j} p_i p_j = 1 \\
\lim_{x \to \infty} f(x) &= \sum_{i,j} p_i p_j \lim_{x \to \infty} e^{xs(i,j)} = \infty \\
f'(x) &= \sum_{i,j} p_i p_j e^{xs(i,j)} \cdot s(i,j) \\
f'(0) &= \sum_{i,j} p_i p_j s(i,j) < 0 \qquad \text{(nach Voraussetzung an Score-Schema)} \\
f''(x) &= \sum_{ij} p_i p_j e^{xs(i,j)} \cdot (s(i,j))^2 > 0
\end{aligned}$$

Aus dieser Kurvendiskussion folgt, dass f in etwa so einen Verlauf nehmen muss:



Insbesondere hat f nur ein lokales (und damit globales) Minimum und für genau ein $\lambda > 0$ gilt $f(\lambda) = 1$. Zur Anwendung des Newton-Verfahrens wählen wir nun einen Startwert x_0 mit $f'(x_0) > 0$ und approximieren f um Punkt $(x_0, f(x_0))$ durch die Tangente, also die Gerade mit Steigung $f'(x_0)$. Diese Tangente nimmt an der Stelle $x_1 = \frac{1-f(x_0)}{f'(x_0)} + x_0$ den Wert 1 an, denn es gilt ja $f'(x_0) = \frac{f(x_1)-f(x_0)}{x_1-x_0}$. Verwende nun x_1 an Stelle von x_0 und iteriere das Verfahren bis ein x_n gefunden wird, für das $f(x_n)$ hinreichend nahe bei 1 liegt. Dieses x_n ist unser gesuchtes λ.

Die Berechnung von k_1 und k_2 ist etwas komplizierter. Sei dazu S_i der (möglicherweise negative) Score des globalen gaplosen Alignments

$$\begin{array}{cccc} X_1 & X_2 & X_3 & \dots & X_i \\ Y_1 & Y_2 & Y_3 & \dots & Y_i \end{array}$$

der ersten i Positionen zufälliger Sequenzen mit den vorgegebenen Nukleotid- oder Aminosäurehäufigkeiten $(p_j)_j$. Nach Voraussetzung gilt $\mathbb{E}S_1 = \sum_{i,j} p_i p_j s(i,j) < 0$. In der Regel liegen alle Werte, die von S_i angenommen werden können auf einem Gitter $\mathcal{S} = \{x \mid \exists_i \ \Pr(S_i = x) > 0\}$. Zum Beispiel gilt $\mathcal{S} \subseteq \mathbb{N}$, falls $\forall_{i,j}\ s(i,j) \in \mathbb{N}$). Sei

$$\begin{aligned} \delta &:= \min\{|x-y| \ : \ x,y \in \mathcal{S}, x \neq y\} \\ &= \max\{r \ : \ \forall_{i,j} \exists_{k \in \mathbb{N}}\ r \cdot k = s(i,j)\} \end{aligned}$$

der kleinste Gitterabstand. Sei außerdem

$$\begin{aligned} P &:= \exp\left(-\sum_{k=1}^{\infty} \frac{1}{k} \Pr\left(S_k \geq 0\right)\right) \\ Q &:= \exp\left(-\sum_{k=1}^{\infty} \frac{1}{k} \cdot \sum_{s \in \mathcal{S} \cap]-\infty, 0[} e^{\lambda s} \Pr(S_k = s)\right). \end{aligned}$$

Dann gilt

$$\begin{aligned} k_1 &= \frac{\delta P^2 Q^2}{(e^{\lambda\delta} - 1) \cdot \mathbb{E}\left(S_1 e^{\lambda S_1}\right)} \\ k_2 &= k_1 \cdot e^{\lambda\delta} \end{aligned}$$

Wir verzichten hier auf den recht technischen Beweis dieses Resultats, siehe z.B. Ewens, Grant (2001).

Hilfreich für die Berechnung der Werte P und Q sind die Wahrscheinlichkeiten

$$f[k,s] := \Pr(S_k = s \ \wedge \ \forall_{i<k}\, S_i \geq 0),$$

dass die in 0 startende Score-Irrfahrt nach k Schritten in $s \in \mathcal{S}$ ist und zuvor nicht negativ war. Da die Irrfahrt eine negative Drift hat (also im Mittel und damit langfristig sicher nach unten geht), ist $f[k,s]$ vernachlässigbar klein, falls k groß ist. Wir brauchen also $f[k,s]$ nur für relativ kleine k zu berechnen. Dazu sei

$$r_z := \sum_{i,j \text{ mit } s(i,j)=z} p_i p_j$$

die Wahrscheinlichkeit, dass die Irrfahrt einen Schritt z macht und Y_x sei die Menge der $y \geq 0$, die von 0 aus in x Irrfahrtsschritten erreichbar sind. Bei der Berechnung der Werte $f(x,y)$ kommt wiedermal dynamische Programmierung zum Einsatz:

$Y_0 := \{0\}$
$Y_1 := \{z \mid \exists\ i,j : s(i,j) = z\}$
`Für` $z \in Y_1$`:`
    $f[1,z] := r_z$
$k := 2$
`Wiederhole:`
    $Y_k := \emptyset$
    $s := 0$
    `Für` $y \in Y_{k-1}$ `und` $z \in Y_1$ `mit` $z + y \geq 0$ `:`
        `Falls` $z + y \in Y_k$ `:`
            $f[k, y+z] := f[k, y+z] + f[k-1, y] \cdot r_z$
            $s := s + f[k-1, y] \cdot r_z$
        `Sonst :`
            $Y_k := Y_k \cup \{y+z\}$
            $f[k, y+z] := f[k-1, y] \cdot r_z$
            $s := s + f[k, y+z]$
    $k := k + 1$
`solange` $s > \varepsilon$

(Dabei ist $\varepsilon > 0$ sehr klein, und alles, was weniger ist, betrachten wir als vernachlässigbar.)

Sei nun u_b die Wahrscheinlichkeit, dass eine in 0 startende Score-Irrfahrt b übersteigt, bevor sie negative Werte erreicht. Da u_b (sehr vereinfacht ausgedrückt) für große b ähnlich fällt wie $\max_k f[k,b]$, kann man mit λ und $f[k,y]$ Konstanten c_1 und c_2 finden, so dass gilt:

$$c_1 \cdot e^{-\lambda b} \lesssim u_b \lesssim c_2 \cdot e^{-\lambda b}$$

Außerdem kann man aus den Werten $f[k,y] \cdot r_z$ mit $y + z < 0$ die erwartete Anzahl ρ der Schritte zwischen zwei Leiterpunkten berechnen. Dann gilt:

$$k_1 = c_1/\rho \text{ und } k_2 = c_2/\rho$$

Die letzten Argumente waren eher skizzenhaft. Details findet man bei Feller (1968), Karlin und Altschul (1990), Mott und Tribe (1999), Ewens und Grant (2001).

Im Folgenden vernachlässigen wir den Unterschied zwischen k_1 und k_2 und verwenden nur noch $k := k_1 \approx k_2$.

Anhand der bisherigen Überlegungen erhalten wir eine Asymptotik für den e-Wert $E_{S \geq b}$, also die erwartete Anzahl an HSPs mit Score$\geq b$ bei Sequenzen der Längen n und m:

$$E_{S \geq b} \sim n \cdot m \cdot k \cdot e^{-\lambda b}$$

Dies gilt, da es ungefähr $n \cdot m$ Positionspaare gibt, an denen ein lokales Alignment beginnen könnte.

1.4.3 Was heißt das alles für die Wahl des Score-Schemas?

Anhand von e-Werten kann man ausgegebene lokale Alignments nach ihrer Bedeutung sortieren: Je kleiner der e-Wert desto größer ist die Signifikanz. Der Score ist als Maß für die Signifikanz problematisch, da verschiedene Alignments evtl. mit verschieden strengen Score-Schemata gefunden wurden. Einen Kompromiss bildet jedoch die Verwendung des normalisierten Score S', der bei BLAST wegen der Verwendung der 2er-Logarithmen auch "Bit-Score" genannt wird:

$$S' := \frac{\lambda S - \ln k}{\ln 2} \qquad \text{d.h. } S = \frac{S' \cdot \ln 2 + \ln k}{\lambda}$$

Also folgt:

$$\begin{aligned} E_{S' \geq b} &= E_{S \geq \frac{b \cdot \ln 2 + \ln k}{\lambda}} \\ &\sim mn \cdot k \cdot e^{-\lambda \frac{b \ln 2 + \ln k}{\lambda}} \\ &= mn \cdot k \cdot e^{-b \cdot \ln 2} \cdot e^{-\ln k} \\ &= mn \cdot 2^{-b} \end{aligned}$$

Die Beziehung zwischen den Bit-Scores und den e-Werten hängt also asymptotisch nicht von λ und k ab. Daher sind Bit-Scores besser miteinander zu vergleichen.

Wir kommen nun noch einmal auf die Wahl der Score-Funktion zurück. Bereits im Zusammenhang mit PAM- und BLOSUM-Matrizen hieß es: Wenn $(p_i)_{i \in \mathcal{A}}$ die Basen- bzw. Aminosäure-Häufigkeiten sind und bei wahren Alignments $i \in \mathcal{A}$ und $j \in \mathcal{A}$ mit Wahrscheinlichkeit q_{ij} an homologen Positionspaaren auftreten, dann wähle man für den Match/Mismatch-Score

$$s(i,j) = c \cdot \log \frac{q_{ij}}{p_i \cdot p_j},$$

wobei c eine Konstante ist, um die es jetzt gehen soll. Wir können auch von $s(i,j)$ ausgehen und q_{ij} gerade so definieren, dass obige Gleichung gilt, d.h. wir setzen:

$$q_{ij} = p_i p_j e^{\frac{1}{c}s(i,j)}$$

Damit q_{ij} eine Wahrscheinlichkeitsverteilung auf $\mathcal{A} \times \mathcal{A}$ sein kann, muss gelten:

$$1 = \sum_{i,j} q_{ij} = \sum_{i,j} p_i p_j e^{\frac{1}{c}s(i,j)}$$

Das bedeutet aber, dass $c = 1/\lambda$ ist, denn λ war gerade so definiert. In Karlin und Altschul (1990) und Karlin (1994) wird gezeigt, dass nicht nur die HSPs in wahren Alignments (falls denn die Alternativhypothese stimmt) sondern auch die HSPs mit optimalen Scores bei sehr langen *unverwandten* Sequenzen gemäß der Verteilung $(q_{ij})_{i,j\in\mathcal{A}}$ zusammengesetzt sind. Bei so zusammengesetzten Alignments ist also der Erwartungswert für den Score-Beitrag eines Positionspaares:

$$\sum_{i,j} q_{ij} \cdot \lambda^{-1} \cdot \log \frac{q_{ij}}{p_i p_j} = \mathrm{H}/\lambda$$

Dabei ist H (der griechische Buchstabe Eta) die relative Entropie der Verteilung $(q_{ij})_{(i,j)\in\mathcal{A}^2}$ zur Verteilung $(p_i \cdot p_j)_{(i,j)\in\mathcal{A}^2}$.

1.4.4 p-Werte für hohe Scores bei HSPs

Vom Alignment mal abgesehen ist der p-Wert als Maß für Signifikanz weitaus gebräuchlicher als der e-Wert. Der p-Wert ist im Alignment-Kontext die Wahrscheinlichkeit, dass *mindestens ein* HSP mit Score$\geq b$ existiert, wobei b der Score des besten Alignments für die zu untersuchenden Sequenzen sei.

Ein Problem bei der Berechnung von p-Werten bilden die vielen stochastischen Abhängigkeiten: Sei I_{ij} die **Indikatorvariable** des Ereignisses $\{$An Positionspaar (i,j) beginnt ein HSP mit Score$\geq b\}$, d.h. falls das Ereignis eintritt, gilt $I_{ij} = 1$ und sonst gilt $I_{ij} = 0$.

Wir können dann den e-Wert schreiben als $E_{S\geq b} = \mathbb{E}\sum_{ij} I_{ij}$. Bei der Berechnung des p-Werts $\mathrm{Ws}(\sum_{ij} I_{ij} > 0)$ müssten eigentlich stochastische Abhängigkeiten berücksichtigt werden, wie etwa, dass I_{ij} und $I_{i+1,j+1}$ nicht beide den Wert 1 annehmen können.

Karlin, Dembo und Zeitouni (1992) zeigen aber, dass diese Abhängigkeiten insgesamt gesehen sehr schwach sind und in den praktisch relevanten Fällen vernachlässigt werden können. Da es ungefähr $n \cdot m$ Indikatorvariablen I_{ij} gibt und alle (bis auf die mit $i \approx n$ oder $j \approx m$, die wir hier vernachlässigen) mit einer Wahrscheinlichkeit von ungefähr $ke^{-\lambda b}$ den Wert 1 annehmen, ist $\sum_{ij} I_{ij}$ ungefähr binomialverteilt zu den Parametern $(nm, ke^{-\lambda b})$, d.h. es gilt:

$$\mathrm{Ws}(\sum_{ij} I_{ij} = l) \approx \binom{nm}{l} \left(ke^{-\lambda b}\right)^l \left(1 - ke^{-\lambda b}\right)^{nm-l}$$

Da wir sehr viele Ereignisse (bzw. Indikatorvariablen) haben, die alle mit sehr kleiner Wahrscheinlichkeit eintreten (bzw. den Wert 1 annehmen), können wir die Bionmialverteilung durch die Poissonverteilung

zum Paramter $\mu = nmke^{-\lambda b}$ approximieren (siehe Anhang A.7). Wir erhalten damit:

$$\mathrm{Ws}(\sum_{ij} I_{ij} = l) \approx \frac{\mu^l}{l!} e^{-\mu}$$

Insbesondere ist der gesuchte p-Wert:

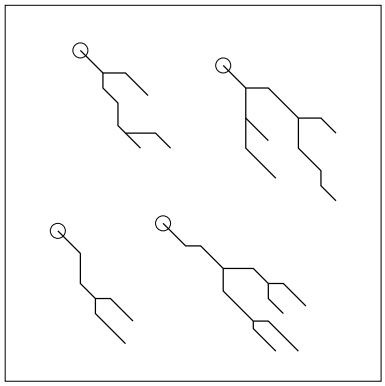
$$1 - \mathrm{Ws}(\sum I_i j = 0) \approx 1 - \frac{\mu^0}{0!} e^{-\mu} = 1 - e^{-nmk \cdot e^{-\lambda b}}$$

1.4.5 e-Werte für hohe Scores bei Alignments mit Gaps

Dieser Fall ist weitaus komplizierter und es ist bei weitem nicht alles bewiesen, was man vermutet und was den e-Werten, die die neueren Versionen von BLAST ausgeben, an Annahmen zu Grunde liegt. Dazu später mehr.

Zunächst müssen wir uns klar machen, was die e-Werte eigentlich zählen sollen (genauer ausgedrückt: von welcher Zahl sie der Erwartungswert sein sollen). Bei den HSP's haben wir nicht jedes gaplose Alignment gezählt, das einen hohen Score enthält, sondern eben nur die HSPs, also solche gaplose Alignments, die sich nicht noch durch Verkürzen oder Verlängern verbessern lassen und somit in gewissem Sinne lokal optimal sind. Damit haben wir vermieden, dass wir Alignments doppelt zählen, die sich eigentlich nicht wesentlich unterscheiden und auf denselben Übereinstimmungen zwischen Sequenz-Positionen beruhen.

Auch im Fall von Alignments mit Gaps möchte man etwas tun, was manchmal als "declumping", also "Entklumpung", bezeichnet wird: lokale Alignments, die einen "Klumpen" bilden, die sich also miteinander überschneiden (z.B. weil sie im Wesentlichen dieselben HSPs verbinden), sollen nicht einzeln gezählt werden. Es sollen nur die Klumpen gezählt werden, in denen ein Alignment vom Score$\geq b$ vorkommt.



Eine Möglichkeit, das zu konkretisieren, ist der Ansatz von Altschul, Bundschuh, Olsen und Hwa (2001), die vorschlagen, dass man sich beim Smith-Waterman-Algorithmus jeweils merkt, wo das lokale Alignment, dessen Score an die jeweiligen Kante geschrieben wird, begann. Das ist mit dynamischer Programmierung effizient zu machen: Wenn man beim Kanten-Beschriften einen Score-Anteil von einer anderen Kante übernimmt, dann übernimmt man auch deren Information über den Anfangspunkt des Alignments. In den Anfangspunkt schreibt man jeweils den maximalen Score eines dort beginnenden Alignments. Alle Alignments, die im selben Punkt beginnen, bilden eine "Insel". (Wir übernehmen hier die Terminologie von Altschul et al. (2001) und sprechen von Inseln statt von Klumpen. Klingt ja auch irgendwie besser.) Die Inseln entsprechen den grauen Gebilden in Abbildung 1. Etwas schematischer ist die Darstellung hier am Seitenrand (Kringel markieren den jeweiligen Anfangspunkt).

Es wird vermutet, dass es auch im Fall von lokalen Alignments mit Gaps Zahlen λ und k gibt, so dass eine Asymptotik der Form

$$E_{S \geq b} \sim mn \cdot k \cdot e^{-\lambda b}$$

gilt. Das ist allerdings nicht wirklich bewiesen, und es ist auch noch nicht hinreichend klar, welche Werte für k und λ zu wählen sind (falls denn die Asymptotik überhaupt gilt). Altschul et al. (2001) schlagen vor – und so ähnlich wird es dann auch für BLAST gemacht –, dass man für gängige Score-Schemata Sequenzdaten simuliert und dann für alle b ab einer gewissen Größe die in den Simulationsergebnissen vorliegenden Anzahlen $c(b)$ der Inseln mit Score$\geq b$ ermittelt. Dann suche man λ und k, so dass für die relevanten Werte für b die Approximation $c(b) \approx mnke^{-\lambda b}$ möglichst gut ist. Das klingt einfacher als es ist. Unter anderem gibt es folgende Probleme:

- Da λ als Exponent eingeht, muss es sehr genau bestimmt werden. Kleine Änderungen von λ haben nämlich einen großen Einfluss aufs Ergebnis.
- Leider gelten die Asymptotiken nur für sehr große Werte für n und m. Bei kürzeren Sequenzen müssen zusätzliche Korrekturterme verwendet werden. Das ist ein Thema für sich.
- Repetitive Elemente, Microsatelliten und Ähnliches werden im Nullmodell nicht berücksichtigt. Alignments, die so etwas enthalten, müssen gesondert behandelt werden. (Gilt natürlich auch schon für Alignments ohne Gaps.)
- Da Simulationen nur für bestimmte Scoring-Schemata durchgeführt wurden, kann BLAST e-Werte nur für diese ausgeben.

Auch wenn die Asymptotik für die Signifikanz lokaler Alignments mit Gaps nicht bewiesen ist, gibt es doch außer Simulationsstudien auch mathematische Teilergebnisse. Siegmund und Yakir (2000) konnten zeigen, dass die Asymptotik gilt, wenn die Gap-Open-Penalty hinreichend groß ist (von der Größenordnung $\log b$). In diesem Fall stimmt λ mit dem λ des gaplosen Falls überein. Die Struktur solcher Alignments wird in Metzler, Grossmann, Wakolbinger (2002) diskutiert. Mott und Tribe (1999) schlagen einen Greedy-Algorithmus vor, der möglicherweise nicht das optimale Alignment findet. Für die ausgegebenen Alignments können aber e-Werte berechnet werden. Grossmann (2003) schlägt eine Methode zur Berechnung von λ für Alignments mit Gaps vor. In Metzler (2006) wird eine Methode diskutiert, mit der man k und λ für Alignments mit Gaps direkt aus den jeweils gegebenen Sequenzen schätzen kann.

2 Hidden Markov Modelle

2.1 Ein Hidden Markov Modell für CpG-Inseln

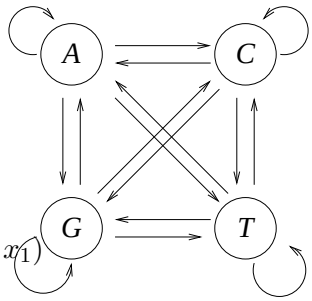
In der menschlichen DNA folgt G nur selten auf C, da das C in diesem Fall häufig methyliert wird und somit zu einem T mutiert. Es gibt aber auch Bereiche, etwa in Promoter-Regionen, wo G auf C folgen

muss. In solchen sogenannten CpG-Inseln wird die Methylierung durch Enzyme verhindert. (Das p in CpG steht für die Phosphorsäure und soll die Verwechslung mit dem Watson-Crick-Paar CG verhindern.)

Als Anwendungsbeispiel für Hidden Markov Modelle (HMM) fragen wir uns, wie wir CpG-islands in einer langen DNA-Sequenz finden können.

Zunächst geht es darum, für einen gegebenen Sequenzabschnitt zu entscheiden, ob es sich um ein CpG-Inseln handelt oder nicht.

Dazu modellieren wir die DNA-Sequenz durch eine Markoff-Kette. Eine Folge $X_1, X_2, X_3, \ldots$ von Zufallsvariablen, die Werte in einem Zustandsraum $\mathcal{Z}$ (z.B. $\{A, C, G, T\} = \mathcal{Z}$) annehmen, heißt **Markoff-Kette** (Markov chain) auf $\mathcal{Z}$, falls die Wahrscheinlichkeitsverteilung von X_i nur von X_{i-1} abhängt und darüber hinaus nicht von $X_1, X_2, \ldots, X_{i-2}$.



Formal ausgedrückt muss also für alle $x, x_1, \ldots, x_{i-1} \in \mathcal{Z}$ gelten:

$$\mathrm{Ws}(X_i = x | X_{i-1} = x_{i-1}) = \mathrm{Ws}(X_i = x | X_{i-1} = x_{i-1}, X_i = x_i, \ldots, X_1 = x_1)$$

Die Markoff-Kette sei *homogen*, d.h. die Übergangswahrscheinlichkeiten sind für alle Positionen dieselben:

$$\forall i: \ \mathrm{Ws}(X_i = y | X_{i-1} = x) = \mathrm{Ws}(X_2 = y | X_1 = x) =: P_{xy}$$

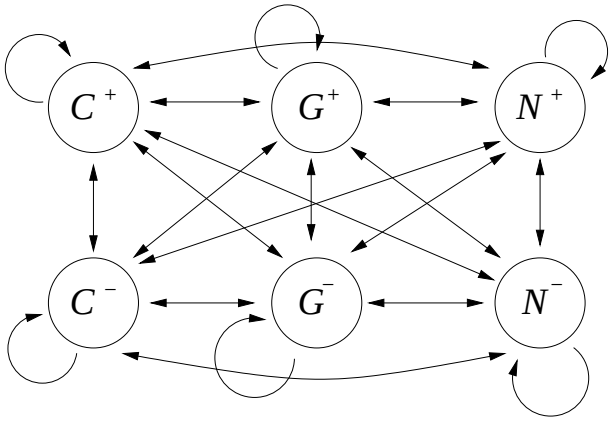
Ähnlich wie wir es bereits im Zusammenhang mit PAM-Matrizen in Abschnitt 1.2.1 gesehen hatten, kann man die Übergangswahrscheinlichkeiten der Markoff-Kette dann auch über mehrere Schritte hinweg mit Hilfe der Übergangsmatrix berechnen, die beim Zustandsraum $\mathcal{Z} = \{A, C, G, T\}$ von folgender Form ist:

$$P = \begin{pmatrix} P_{AA} & P_{AC} & P_{AG} & P_{AT} \\ P_{CA} & P_{CC} & P_{CG} & P_{CT} \\ P_{GA} & P_{GC} & P_{GG} & P_{GT} \\ P_{TA} & P_{TC} & P_{TG} & P_{TT} \end{pmatrix}$$

Um nun CpG-islands zu finden, könnten wir zunächst P aus bekannten Daten getrennt für CpG-islands ("P^+") und sonstige Regionen ("P^-") schätzen und dann für die zu klassifizierende Sequenz $s = s_1 \ldots s_n$ den log-Likelihoodquotienten berechnen:

$$\log \frac{\mathrm{Ws}_+(s)}{\mathrm{Ws}_-(s)} = \sum_{i=2}^{n} \log \frac{P^+_{s_{i-1}s_i}}{P^-_{s_{i-1}s_i}}$$

Interessanter ist folgender Fall: Suche in der sehr langen Sequenz $s = s_1, \ldots, s_n$ nach CpG-islands. Es ist also innerhalb der Sequenz an unbekannten Positionen ein Wechsel zwischen CpG-islands und sonstigen Regionen möglich. Zur Unterscheidung markieren wir die Positionen innerhalb eines CpG-Inseln mit $+$, die anderen mit $-$ (diese Markierungen sind natürlich den Daten nicht unmittelbar zu entnehmen). Innerhalb der jeweiligen Regionen unterscheiden wir zwischen den Zuständen C, G und N. Letzteres steht für A oder T. Wir gehen also von einer Markoff-Kette aus, die auf folgendem Graphen umherspringt:



Hidden Markov Model (HMM): Eine unbeobachtbare Markoff-Kette $X_1, X_2, \ldots$ auf einem Zustandsraum $\mathcal{Z}$ mit Übergangswahrscheinlichkeiten $P_{xy} = \mathrm{Ws}(X_i = y | X_{i-1} = x)$ und Startverteilung $P_x = \mathrm{Ws}(X_1 = x)$ emittiert beobachtbare zufällige Signale $S_1, S_2, \ldots$ aus einem Alphabet $\mathcal{A}$. Die Emissionswahrscheinlichkeiten für S_i hängen jeweils nur von X_i ab, d. h. für $y \in \mathcal{Z}$ und $b \in \mathcal{A}$ gilt:

$$e_y(b) := \mathrm{Ws}(S_i = b | X_i = y) = \mathrm{Ws}(S_i = b | X_i = y, X_1, X_2, \ldots, S_1, \ldots, S_{i-1}, S_{i+1}, \ldots)$$

In unserem CpG-Beispiel verwenden wir $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\} = \mathcal{A}$ und $\{C_+, G_+, N_+, C_-, G_-, N_-\} = \mathcal{Z}$. Dabei kennzeichnet $+$ die Zustände innerhalb und $-$ die Zustände außerhalb von CpG-islands. N emittiert A oder T jeweils mit Wahrscheinlichkeit 0.5.

2.2 Der Viterbi Algorithmus findet den wahrscheinlichsten Pfad

Gegeben sei eine Sequenz von Emissionen $s = (s_1, \ldots, s_n)$ eines Hidden Markov Modells mit Übergangswahrscheinlichkeiten P_{xy} und Emissionswahrscheinlichkeiten $e_x(b)$ für $x, y \in \mathcal{Z}$ und $b \in \mathcal{A}$. Sei außerdem $P_x = \mathrm{Ws}(X_1 = x)$.

Frage: Welcher "Pfad" $x_1, \ldots, x_n$ ist der wahrscheinlichste für die versteckte Kette $X_1, \ldots, X_n$? (in unserem Beispiel: Was ist die wahrscheinlichste Aufteilung in CpG-islands und sonstige Bereiche?)

Zu suchen ist also

$$\begin{aligned}(x_1^*, \ldots, x_n^*) &= \arg \max_{(x_1,\ldots,x_n)} \mathrm{Ws}(X = (x_1, \ldots, x_n) | S = (s_1, \ldots, s_n)) \\ &= \arg \max_{(x_1,\ldots,x_n)} \mathrm{Ws}(X = (x_1, \ldots, x_n), S = (s_1, \ldots, s_n))\end{aligned}$$

Der Ansatz ist wiedermal dynamische Programmierung. Wir berechnen für alle $y \in \mathcal{Z}$ und $k \leq n$ den Wert

$$v_k(y) := \max_{(x_1,\ldots,x_{k-1}) \in \mathcal{Z}^{k-1}} \mathrm{Ws}(X_1 = x_1, X_2 = x_2, \ldots, X_{k-1} = x_{k-1}, X_k = y, S_1 = s_1, \ldots, S_k = s_k)$$

Viterbi-Algorithmus

Für $x \in \mathcal{Z}$:
    $v_1(x) := P_x \cdot e_x(s_1)$
Für $k = 2, \ldots, n$:
    Für $x \in \mathcal{Z}$:
        $v_k(x) := e_x(s_k) \cdot \max_y\{v_{k-1}(y) \cdot P_{yx}\}$
$x_n^* := \arg\max_y\{v_n(y)\}$
Für $k = n-1, \ldots, 1$:
    $x_k^* := \arg\max_y\{v_k(y) \cdot P_{yx_{k+1}^*}\}$
$\mathrm{Ws}(X = x^*, S = (s_1, \ldots, s_n)) := \max_y\{v_n(y)\}$

Die Laufzeit ist offensichtlich $O(n \cdot |\mathcal{Z}^2|)$.

Praktische Überlegung: In der Rekursion werden sehr kleine Wahrscheinlichkeiten aufmultipliziert. Um zu vermeiden, dass Rechenungenauigkeiten durch die winzigen Fließkommazahlen entstehen, kann man in der logarithmischen Skala rechen. Die Rekursion ist dann also:

$$\log(v_k(x)) = \log(e_x(s_k)) + \max_y(\log v_{k-1}(y) + \log P_{yx})$$

Statt der Werte $v_k(x)$ berechnen (und benötigen) wir also nur die Werte $\log(v_k(x))$

2.3 Wahrscheinlichste einzelne Zustände: Der vorwärts-rückwärts-Algorithmus

Gegeben sei eine Sequenz von Emissionen eines HMM. Wie wahrscheinlich sind die einzelnen möglichen inneren Zustände an einer festen Position?

Wir überlegen zunächst, wie wir die Wahrscheinlichkeit berechnen, dass das HMM eine bestimmte Folge $s = (s_1, s_2, \ldots, s_n)$ emittiert:

$$\mathrm{Ws}(S = s) = \sum_{x \in \mathcal{Z}^n} \mathrm{Ws}(S = s, X = x)$$

Das machen wir wieder mit dynamischer Programmierung, diesmal über:

$$f_k(y) := \mathrm{Ws}(S_1 = s_1, \ldots, S_k = s_k, X_k = y) \qquad \text{für } y \in \mathcal{Z}$$

Vorwärts-Algorithmus

Für alle $x \in \mathcal{Z}$:
    $f_1(x) := P_x \cdot e_x(s_1)$
Für $k = 2, \ldots, n$ und alle $x \in \mathcal{Z}$:
    $f_k(x) := e_x(s_k) \sum_{y \in \mathcal{Z}} f_{k-1}(y) P_{yx}$

$\mathrm{Ws}(S = (s_1, \ldots, s_n)) := \sum_{y \in \mathcal{Z}} f_n(y)$

Die $f_k(x)$ werden wir auch später noch brauchen; ebenso folgende Größen, die im Rückwärts-Algorithmus berechnet werden:

$$\begin{aligned}
b_k(y) &:= \mathrm{Ws}(S_{k+1} = s_{k+1}, \ldots, S_n = s_n \mid X_k = y) \\
&= \sum_{x \in \mathcal{Z}} \mathrm{Ws}(X_{k+1} = x, S_{k+1} = s_{k+1}, \ldots, S_n = s_n \mid X_k = y) \\
&= \sum_{x \in \mathcal{Z}} \mathrm{Ws}(S_{k+1} = s_{k+1}, \ldots, S_n = s_n \mid X_{k+1} = x, X_k = y) \cdot \mathrm{Ws}(X_{k+1} = x \mid X_k = y) \\
&\quad \text{(denn es gilt die Bayes-Formel auch für bedingte Verteilungen } \mathrm{Ws}(.|C) : \\
&\qquad \mathrm{Ws}(A, B|C) = \mathrm{Ws}(A|B, C) \cdot \mathrm{Ws}(B|C).) \\
&= \sum_{x \in \mathcal{Z}} \mathrm{Ws}(S_{k+1} = s_{k+1}, \ldots, S_n = s_n \mid X_{k+1} = x) \cdot \mathrm{Ws}(X_{k+1} = x \mid X_k = y) \\
&\quad \text{(wegen der Markoff-Eigenschaft von } X) \\
&= \sum_{x \in \mathcal{Z}} \mathrm{Ws}(S_{k+2} = s_{k+2}, \ldots, S_n = s_n \mid X_{k+1} = x) \cdot \mathrm{Ws}(S_{k+1} = s_{k+1} \mid X_{k+1} = x) \cdot \\
&\qquad \cdot \mathrm{Ws}(X_{k+1} = x \mid X_k = y) \\
&\quad \text{(da } S_{k+1} \text{ nur von } X_{k+1} \text{ abhängt.)} \\
&= \sum_{x \in \mathcal{Z}} b_{k+1}(x) \cdot e_x(s_{k+1}) \cdot P_{yx}
\end{aligned}$$

Wir erhalten also eine Rekursion für $b_k(y)$, und die benutzen wir im

Rückwärts-Algorithmus

`Für` $y \in \mathcal{Z}$`:`
  $b_n(y) := 1$
`Für` $k = n-1, \ldots, 1$`:`
  `Für` $y \in \mathcal{Z}$`:`
    $b_k(y) := \sum_{x \in \mathcal{Z}} b_{k+1}(x) \cdot P_{yx} \cdot e_x(s_{k+1})$
$\mathrm{Ws}(S = (s_1, \ldots, s_n)) := \sum_{x \in \mathcal{Z}} P_x \cdot b_1(x) \cdot e_x(s_1)$

Der Zeitaufwand für den Rückwärts-Algorithmus, ebenso wie für den Vorwärts- und den Viterbi-Algorithmus, ist offensichtlich $O(n \cdot |\mathcal{Z}|^2)$.

Praktische Überlegung II: Auch hier werden sehr viele kleine Zahlen aufmultipliziert und man kann die daraus evtl. entstehenden Probleme bzgl. der Rechnengenauigkeit umgehen, indem man in der logarithmischen Skala rechnet. Anders als beim Viterbi-Algorithmus gibt es hier allerdings das Problem, dass Summen vorkommen, d. h. man muss $\log(a + b)$ aus $\log(a)$ und $\log(b)$ berechnen. Dabei will man

nicht als Zwischenschritt über a und b gehen, denn das sind ja die problematisch kleinen Werte (klein im Sinne von "nahe bei 0"). Um das numerisch stabil hinzubekommen, kann man die Gleichung

$$\log(a+b) = \log(e^{\log(a)} + e^{\log(b)}) = \log(e^{\log(a)} \cdot (1 + e^{\log(b)-\log(a)})) = \log(a) + \log(1 + e^{\log(b)-\log(a)})$$

verwenden, wobei a die größere der beiden Zahlen sei. (Wenn a wesentlich größer als b ist, ist $e^{\log(b)-\log(a)}$ auf der rechten Seite ohnehin vernachlässigbar.)

Unser eigentliches Ziel in diesem Abschnitt war ja die Berechnung der Wahrscheinlichkeiten von inneren Zuständen in einzelnen Positionen. Das können wir unter Verwendung der Nebenprodukte des Vorwärts- und des Rückwarts-Algorithmus tun. Für $x \in \mathcal{Z}$ und $s = (s_1, \ldots, s_n)$ gilt nämlich:

$$\begin{aligned}
\mathrm{Ws}(X_k = x \mid S = s) &= \frac{\mathrm{Ws}(X_k = x, S = s)}{\mathrm{Ws}(S = s)} \\
&= \frac{\mathrm{Ws}(S = s \mid X_k = x) \cdot \mathrm{Ws}(X_k = x)}{\mathrm{Ws}(S = s)} \\
&= \mathrm{Ws}((S_1, \ldots, S_k) = (s_1, \ldots, s_k) \mid X_k = x) \cdot \mathrm{Ws}(X_k = x) \cdot \\
&\qquad \cdot \frac{\mathrm{Ws}((S_{k+1}, \ldots, S_n) = (s_{k+1}, \ldots, s_n) \mid X_k = x)}{\mathrm{Ws}(S = s)} \\
&\qquad \text{(Denn bedingt auf } X_k \text{ sind } (S_1, \ldots, S_k) \text{ und} \\
&\qquad (S_{k+1}, \ldots, S_n) \text{ voneinander unabhängig.)} \\
&= \frac{\mathrm{Ws}((S_1, \ldots, S_k) = (s_1, \ldots, s_k), X_k = x) \cdot b_k(x)}{\mathrm{Ws}(S = s)} \\
&= \frac{f_k(x) \cdot b_k(x)}{\mathrm{Ws}(S = s)}
\end{aligned}$$

Wir können also die gesuchte Wahrscheinlichkeit durch einen Bruch von Größen ausdrücken, die wir recht effizient mit dem Vorwärts- und dem Rückwärts-Algorithmus berechnen können.

2.4 EM für HMMs: Parameterschätzung mit dem Baum-Welch-Algorithmus

Gegeben sei eine Sequenz $s_1, s_2, \ldots, s_n \in \mathcal{A}$ von Beobachtungen aus einem HMM auf einem Zustandsraum $\mathcal{Z}$. Gesucht seien nun die Parameterwerte $\theta = (P_{xy}, e_x(s))_{x,y \in \mathcal{Z}, s \in \mathcal{A}}$. Wir setzen hier P_x als bekannt voraus.

Wir wollen zwei mögliche allgemeine Ansätze der Parameterschätzung kurz andiskutieren. Wir werden noch oft auf sie zurückkommen.

Der Bayes'sche Ansatz ist eine Möglichkeit, θ zu schätzen: Wir gehen davon aus, dass das wahre θ zufällig ist und wählen dafür das θ^* als Schätzwert, das die höchste a-posteriori-Wahrscheinlichkeit hat, also die höchste bedingte Wahrscheinlichkeit $\mathrm{Ws}(\theta = \theta^* | S = (s_1, \ldots, s_n))$ für die gegebenen

Beobachtungen. Mit dem Vorwärts-Algorithmus können wir $\text{Ws}(S = (s_1, \ldots, s_n) | \theta = \theta^*)$ berechnen und nach der Bayes-Formel gilt:

$$\begin{aligned}
\text{Ws}(\theta = \theta^* | S = (s_1, \ldots, s_n)) &= \frac{\text{Ws}(\theta = \theta^* \ , \ S = (s_1, \ldots, s_n))}{\text{Ws}(S = (s_1, \ldots, s_n))} \\
&= \frac{\text{Ws}(S = (s_1, \ldots, s_n) \mid \theta = \theta^*) \cdot \text{Ws}(\theta = \theta^*)}{\sum_{\theta'} \text{Ws}(S = (s_1, \ldots, s_n) \ , \ \theta = \theta')} \\
&= \frac{\text{Ws}(S = (s_1, \ldots, s_n) \mid \theta = \theta^*) \cdot \text{Ws}(\theta = \theta^*)}{\sum_{\theta'} \text{Ws}(S = (s_1, \ldots, s_n) \mid \theta = \theta') \cdot \text{Ws}(\theta = \theta')}
\end{aligned}$$

Aber was ist eigentlich $\text{Ws}(\theta = \theta^*)$? Das ist die sogenannte a-priori-Wahrscheinlichkeit von $\theta = \theta^*$. Wenn wir davon ausgehen, dass θ eine Zufallsvariable ist, dann müssen wir auch davon ausgehen, dass eine Wahrscheinlichkeitsverteilung dafür existiert. In der Bayes'schen Statistik werden für Parameter, deren Werte zu schätzen sind, recht willkürlich a-priori-Verteilungen angenommen. Wir werden später in verschiedenen Kontexten einen solchen Weg gehen. Hier wollen wir aber zunächst die Willkür einer a-priori-Verteilung vermeiden und verfolgen einen anderen Ansatz.

Der Maximum-Likelihood (ML)-Ansatz kommt ohne a-priori-Verteilung der zu schätzenden Parameter aus. Der *Maximum-Likelihood (ML)-Schätzer* $\widehat{\theta}$ ist der Wert für θ, der die Daten möglichst wahrscheinlich werden lässt:

$$\widehat{\theta} := \arg\max_{\theta} \text{W}s_{\theta}(S = (s_1, \ldots, s_n)).$$

Dabei bezeichnet Ws_θ die Wahrscheinlichkeit unter der Annahme, dass θ der gültige Parameterwert ist. Gesucht ist also die Maximalstelle der Likelihood-Funktion $l_{s_1,\ldots,s_n}(\theta) := \text{Ws}_\theta(S = (s_1, \ldots, s_n))$. Wegen ($a \leq b \Leftrightarrow \log a \leq \log b$) können wir genauso gut auch die Maximalstelle der Log-Likelihood $\log l_{s_1,\ldots,s_n}(\theta)$ suchen, was oft etwas übersichtlicher ist. Wir machen uns das zunächst an einem etwas einfacheren Beispiel klar:

Beispiel: ML-Schätzung bei der Binomialverteilung Von $n = 1000$ Positionen einer Sequenz seien $X = 20$ mutiert. Unter der Annahme, dass alle Positionen unabhängig und mit der selben Mutationswahrscheinlichkeit p mutieren, ist X binomial-verteilt:

$$\text{Ws}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Wir berechnen nun den ML-Schätzer $\widehat{p}$ zu p:

$$\begin{aligned}
\widehat{p} &= \arg\max_p \ \binom{n}{k} p^k (1-p)^{n-k} \\
&= \arg\max_p \ \log\left(\binom{n}{k} p^k (1-p)^{n-k}\right) \\
&= \arg\max_p \ \log\left(p^k (1-p)^{n-k}\right) \\
&= \arg\max_p \ k \log p + (n-k)\log(1-p)
\end{aligned}$$

Die Maximalstelle von $f(p) := k \log p + (n-k)\log(1-p)$ finden wir, indem wir die Nullstelle der Ableitung $f'(p) = k/p - (n-k)/(1-p)$ suchen. Es gilt also $k/\widehat{p} = (n-k)/(1-\widehat{p})$, und damit erhalten wir den ML-Schätzer für den Parameter p der Binomialverteilung:

$$\widehat{p} = k/n$$

In obigem Beispiel schätzen wir also $\widehat{p} = 20/1000 = 0.02$. Das erscheint irgendwie naheliegend. In der Tat wird man oft festellen, dass die naheliegende, "naive" Art der Parameterschätzung gerade mit dem ML-Schätzer übereinstimmt.

Nun also zur ML-Schätzung der Übergangs- und Emmissions-W'keiten bei einem HMM: Wären die versteckten Zustände $X_1, \ldots, X_n$ bekannt, so könnten wir die Anzahl A_{xy} der Übergänge $x \to y$ für $x, y \in \mathcal{Z}$ und die Anzahl B_{xb} eines $b \in \mathcal{A}$ aus einem $x \in \mathcal{Z}$ zählen und dann P_{xy} und $e_x(b)$ daraus schätzen. Die log-Likelihood von $\theta = (P_{xy}, e_x(b))_{x,y,b}$ für die gegebenen Anzahlen ist:

$$\log(P_{X_1}) + \sum_{x,y \in \mathcal{Z}} A_{xy} \log(P_{x,y}) + \sum_{x \in \mathcal{Z}, b \in \mathcal{A}} B_{xb} \log(e_x(b))$$

Ähnlich wie im obigen Beispiel erhält man folgende ML-Schätzer:

$$\widehat{P_{x,y}} = \frac{A_{xy}}{\sum_{z \in \mathcal{Z}} A_{xz}} \qquad \widehat{e_x(b)} = \frac{B_{xb}}{\sum_{a \in \mathcal{A}} B_{xa}}$$

Für die Schätzung von $\theta = (P_{xy}, e_x(b))_{x,y,b}$ sind die Anzahlen $(A_{xy}, B_{xb})_{x,y,b}$ ***suffizient***. Das heißt, sie enthalten für diese Schätzung genauso viel nützliche Information wie die gesamte versteckte Kette $X_1, \ldots, X_n$.

Ist umgekehrt $\theta = (P_{xy}, e_x(b))_{x,y,b}$ zusätzlich zu den beobachteten Emmissionen $S = (s_1, \ldots, s_n)$ bekannt, so kann man die Anzahlen $(A_{xy}, B_{xb})_{x,y,b}$ schätzen. Es handelt sich dabei nicht um Parameter sondern um zufallsabhängige Größen. Daher ergäbe der ML-Ansatz hier keinen Sinn. Statt dessen schätzen wir die Größen durch ihre bedingten Erwartungswerte. Um diese zu berechnen, verwenden wir die Ergebnisse des Vorwärts-Rückwärts-Algorithmus $f_i(x)$ und $b_i(x)$:

$$\begin{aligned}
\mathbb{E}_\theta(A_{xy} | S = (s_1, \ldots, s_n)) &= \sum_{i=1}^{n-1} \mathsf{Ws}_\theta(X_i = x, X_{i+1} = y | S = (s_1, \ldots, s_n)) \\
&= \sum_{i=1}^{n-1} \frac{\mathsf{Ws}_\theta(X_i = x, X_{i+1} = y, S = (s_1, \ldots, s_n))}{\mathsf{Ws}_\theta(S = (s_1, \ldots, s_n))} \\
&= \frac{\sum_{i=1}^{n-1} f_i(x) \cdot P_{xy} \cdot e_y(s_{i+1}) \cdot b_{i+1}(y)}{\mathsf{Ws}_\theta(S = (s_1, \ldots, s_n))} \\
\mathbb{E}_\theta(B_{xb} | S = (s_1, \ldots, s_n)) &= \sum_{\{i | s_i = b\}} \mathsf{Ws}_\theta(X_i = x | S = (s_1, \ldots, s_n)) \\
&= \sum_{\{i | s_i = b\}} \frac{\mathsf{Ws}_\theta(X_i = x, S = (s_1, \ldots, s_n))}{\mathsf{Ws}_\theta(S = (s_1, \ldots, s_n)} \\
&= \sum_{\{i | s_i = b\}} \frac{f_i(x) \cdot b_i(x)}{\mathsf{Ws}_\theta(S = (s_1, \ldots, s_n)}
\end{aligned}$$

(Man beachte dabei, dass die Größen P_{xy}, $e_x(b)$, $f_i(x)$ und $b_i(x)$ von θ abhängen.)

Der Baum-Welch-Algorithmus (Baum, 1972): Starte mit irgendeiner groben Schätzung θ_0 und iteriere mehrfach das Schätzen von A_{xy} und B_{xb} aus θ und das Schätzen von θ aus A_{xy} und B_{xb}. Verwende dabei jeweils $\mathbb{E}_\theta(A_{xy}|S=(s_1,\ldots,s_n))$ und $\mathbb{E}_\theta(B_{xb}|S=(s_1,\ldots,s_n))$ als Schätzer von A_{xy} und B_{xb}. Wir werden sehen, dass es sich hierbei um einen Spezialfall des Expectation Maximization (EM) Algorithmus handelt und dass von Iteration zu Iteration die Likelihood von θ steigt.

Eine Alternative: Viterbi-Training Hier werden nicht $\mathbb{E}_\theta(A_{xy}|S=(s_1,\ldots,s_n))$ und $\mathbb{E}_\theta(B_{xb}|S=(s_1,\ldots,s_n))$ als Schätzer für A_{xy} und B_{xb} verwendet sondern die Anzahlen, die im wahrscheinlichsten Pfad beobachtet werden. Dieser kann mit dem Viterbi-Algorithmus berechnet werden. Ein Problem ist, dass es z.B. für gewisse $x,y\in\mathcal{Z}$ passieren kann, dass $A_{xy}=0$ auf dem wahrscheinlichsten Pfad gilt. Man sollte dies dann dennoch nicht als Schätzung verwenden, da sonst in Zukunft alle Pfade, bei denen ein Übergang $x\to y$ erfolgt, gänzlich ausgeschlossen sind. Um dies zu vermeiden, kann man statt der tatsächlichen Anzahlen auf dem wahrscheinlichsten Pfad etwa die um eins erhöhten Anzahlen als Schätzer nehmen. Viterbi-Training kann schneller sein, aber im Allgemeinen ist Baum-Welch die theoretisch besser begründete Lösung.

Der Expectation-Maximization (EM) Algorithmus (Dempster, Laird, Rubin, 1977) Gegeben sei ein Paar voneinander stochastisch abhängiger Zufallsvariablen (X,Y), dessen Verteilung von einem unbekannten Parameter θ abhängt, sowie die Beobachtung $X=x$. Die Zufallsvariable Y ist unbeobachtbar. Gesucht ist der ML-Schätzer $\widehat{\theta}$ für den Parameter θ. Zu optimieren ist also

$$l(\theta)=\mathrm{Ws}_\theta(X=x)=\sum_y \mathrm{Ws}_\theta(X=x,Y=y)$$

(Bei Zufallsvariabelen mit kontinuierlichen Wertebereichen ist die Summe durch ein Integral zu ersetzen.) Der EM-Algorithmus konstruiert eine Folge $\theta_1,\theta_2,\ldots$, die gegen $\widehat{\theta}$ konvergieren soll. Um von einem θ_t zu θ_{t+1} zu gelangen, betrachten wir die Funktion

$$\theta\mapsto Q(\theta|\theta_t):=\mathbb{E}_{\theta_t}(\log\mathrm{Ws}_\theta(X=x,Y)|X=x)=\sum_y \log\left(\mathrm{Ws}_\theta(X=x,Y=y)\right)\cdot\mathrm{Ws}_{\theta_t}(Y=y|X=x)$$

Durch Logarithmieren der Bayes-Formel erhalten wir:

$$\log\mathrm{Ws}_\theta(X=x)=\log\mathrm{Ws}_\theta(X=x,Y=y)-\log\mathrm{Ws}_\theta(Y=y|X=x)$$

Daraus folgt durch Multiplizieren mit $\mathrm{Ws}_{\theta_t}(Y=y|X=x)$ und Aufsummieren über y:

$$\log\mathrm{Ws}_\theta(X=x)=Q(\theta|\theta_t)-\sum_y \mathrm{Ws}_{\theta_t}(Y=y|X=x)\cdot\log\mathrm{Ws}_\theta(Y=y|X=x)$$

Die Gleichung gilt natürlich auch dann, wenn wir θ_t für θ einsetzen. Durch Subtrahieren dieses Spezialfalls von obiger Gleichung erhalten wir:

$$\log\mathrm{Ws}_\theta(X=x)-\log\mathrm{Ws}_{\theta_t}(X=x)=Q(\theta|\theta_t)-Q(\theta_t|\theta_t)+\sum_y \mathrm{Ws}_{\theta_t}(Y=y|X=x)\log\frac{\mathrm{Ws}_{\theta_t}(Y=y|X=x)}{\mathrm{Ws}_\theta(Y=y|X=x)}$$

Die Summe ist gerade die relative Entropie der beiden (bedingten) Verteilungen (vgl. Ewens, Grant, 2001, S. 45 f). Da relative Entropien immer positiv sind, siehe Anhang A.5, folgt:

$$\log l(\theta) - \log l(\theta_t) = \log \mathrm{Ws}_\theta(X = x) - \log \mathrm{Ws}_{\theta_t}(X = x) \geq Q(\theta|\theta_t) - Q(\theta_t|\theta_t)$$

Wenn wir nun θ_{t+1} so wählen, dass gilt $Q(\theta_{t+1}|\theta_t) \geq Q(\theta_t|\theta_t)$, insbesondere also durch

$$\theta_{t+1} := \arg\max_\theta Q(\theta|\theta_t),$$

so erreichen wir also, dass auch $\log l(\theta_{t+1}) \geq \log l(\theta_t)$ gilt. Da die Folge $(\log l(\theta_t))_{t=1,2,\ldots}$ monoton steigt, nach oben aber durch 0 beschränkt ist, konvergiert sie. Die Hoffnung ist, dass der Limes das globale Maximum $\log l(\widehat{\theta})$ ist, und damit $(\theta_t)_{t=1,2,\ldots}$ gegen $\widehat{\theta}$ konvergiert.

Der EM-Algorithmus besteht also aus der Iteration der folgenden beiden Schritte:
E-Schritt: Berechne $Q(.|\theta_t)$
M-Schritt: $\theta_{t+1} := \arg\max_\theta Q(\theta|\theta_t)$

Der Grund dafür, dass diese Schritte meistens effizient durchführbar sind, liegt darin, dass $Q(\theta|\theta_t)$ ein Erwartungswert ist, und das Rechnen mit Erwartungswerten ist meistens angenehm, was unter anderem an der Linearität der Erwartung liegt.

Baum-Welch als Spezialfall von EM Die beobachteten Größen sind nun $S = s = (s_1, \ldots, s_n)$ und die unbeobachtbaren sind die Zustände der versteckten Kette $X = (X_1, \ldots, X_n)$. Wir erhalten also:

$$\begin{aligned}
Q(\theta|\theta_t) &= \sum_{x \in \mathcal{Z}^n} \mathrm{Ws}_{\theta_t}(X = x|S = s) \cdot \log \mathrm{Ws}_\theta(X = x, S = s) \\
&= \sum_{x \in \mathcal{Z}^n} \mathrm{Ws}_{\theta_t}(X = x|S = s) \cdot \left(\sum_{y \in \mathcal{Z}, b \in \mathcal{A}} B_{yb}(x, s) \cdot \log e_{y,\theta}(b) \right. \\
&\qquad \left. + \sum_{y,z \in \mathcal{Z}} A_{zy}(x) \cdot \log P_{zy,\theta} \right) \\
&= \sum_{y \in \mathcal{Z}, b \in \mathcal{A}} \mathbb{E}_{\theta_t}(B_{yb}|S = s) \cdot \log e_{y,\theta}(b) + \sum_{y,z \in \mathcal{Z}} \mathbb{E}_{\theta_t}(A_{zy}|S = s) \cdot \log P_{zy,\theta}
\end{aligned}$$

(Wir schreiben hier θ als Index an die Emissions- und Übergangswahrscheinlichkeiten, um zu verdeutlichen, dass diese durch θ bestimmt sind. Ebenso schreiben wir die Argumente (x, s) und (x) zu B_{yb} bzw. A_{zy} um zu verdeutlichen, dass sich diese Anzahlen auf die versteckte Kette x und die emittierte Folge s beziehen.)

Die letzte Zeile ist gerade das, was wir erhalten, wenn wir in die Formel für die log-Likelihood des Parameters θ_t für die Anzahl der Emissionen eines b von einem y den Wert $\mathbb{E}_{\theta_t}(B_{yb}|S = s)$ und für die Anzahl der Übergänge von Zustand y zu Zustand z den Wert $\mathbb{E}_{\theta_t}(A_{zy}|S = s)$ einsetzen. Baum-Welch wählt θ_{t+1} so, dass dieser Ausdruck maximiert wird, und EM tut genau dasselbe, denn wie wir gesehen haben, ist dieser Ausdruck ja gerade $Q(\theta|\theta_t)$. Also ist Baum-Welch ein Spezialfall von EM. Damit folgt, dass auch bei Baum-Welch die Folge $\theta_1, \theta_2, \ldots$ konvergiert und die Likelihood dabei monoton steigt.

2.5 Gene suchen mit HMMs

2.5.1 Gen-Suche bei Prokaryonten

Gene von Prokaryonten haben eine relativ einfache Struktur: Sie beginnen mit einem Start-Kodon, dann kommen Aminosäuren-kodierende Kodons, dann das Stopp-Kodon. Derartig aufgebaute DNA-Abschnitte werden als Open Reading Frames (ORFs) bezeichnet. Nicht alle ORFs sind wirklich ein Gen.

erste Frage: Wie unterscheiden wir Gene von sonstigen ORFs?

Durbin et al. (1998) betrachten 6500 ORFs von E. Coli der Länge > 100bp. Darunter sind 1100 bekannte Gene. 900 davon wurden verwendet, um ein Markoff-Modell zu "trainieren" (also Übergangswahrscheinlichkeiten zu schätzen), die restlichen 200, um dieses dann zu testen. Als Kriterium wurde der log-Likelihoodquotient $L = \log P_{\widehat{\theta}} / \log P_0$ verwendet. Dabei ist $P_{\widehat{\theta}}$ die Wahrscheinlichkeit der zu überprüfenden Sequenz im trainierten Markoff-Modell und P_0 ist ihre Wahrscheinlichkeit im 0-Modell, bei dem alle Positionen unabhängig gemäß der Basenhäufigkeiten mit Basen belegt werden.

Für das Markoff-Modell wurden 3 Ansätze ausprobiert:

1. Markoffkette auf den Positionen der DNA
2. Markoffkette zweiter Ordnung auf den Positionen der DNA (d.h. die Wahrscheinlichkeit einer Base darf von ihren beiden linken Nachbarn abhängen; dazu mehr im nächsten Abschnitt.)
3. Markoffkette auf den 64 Kodons (als Basen-Tripel)

Während die ersten beiden Ansätze relativ erfolglos blieben, ergab der dritte Ansatz bei den meisten der 200 bekannten Test-Genen einen höheren Wert für L als bei den meisten der übrigen ORFs.

zweite Frage: Wie finden wir Gene?

Auf der Basis obiger Resultate kann man ein HMM-Modell konstruieren, bei dem es einen inneren Zustand gibt, der für eine Position außerhalb der Gene steht und Basen gemäß der Basenhäufigkeiten emittiert. Wir halten uns jeweils geometrisch lange in diesem Zustand auf (d.h. wir entscheiden jeweils unabhängig von der bisherigen Verweildauer, ob wir dort weiter bleiben) und springen dann in einen Zustand, der ein Start-Kodon emittiert. Von dort aus beginnen wir mit der Markoffkette gemäß $P_{\widehat{\theta}}$ bis wir in einem Zustand sind, der dem Stopp-Kodon entspricht. Von dort springen wir wieder in den nicht-Gen-Zustand usw..

Wir können dann auf der Basis dieses Modells z.B. mit dem Viterbi-Algorithmus schätzen, wo die Gene liegen.

2.5.2 Markov-Ketten höherer Ordnung

Eine Folge von Zufallsvariablen $X_1, X_2, \ldots$ heißt *Markoffkette n-ter Ordnung*, falls für alle k gilt:

$$\begin{aligned}&\mathsf{Ws}(X_k = x | X_{k-1} = x_{k-1}, X_{k-2} = x_{k-2}, \ldots, X_1 = x_1)\\ &= \quad \mathsf{Ws}(X_k = x | X_{k-1} = x_{k-1}, X_{k-2} = x_{k-2}, \ldots, X_{k-n} = x_{k-n})\end{aligned}$$

D.h. X_k darf von seinen n linken Nachbarn abhängen, darüber hinaus aber nicht von Positionen, die noch weiter links stehen.

Jede Markoffkette X n-ter Ordnung auf $\mathcal{Z}$ kann man durch eine Markoffkette $\widetilde{X}$ auf $\mathcal{Z}^n$ simulieren, indem man folgende Übergangsdynamik wählt:

$$\begin{aligned}&\mathsf{Ws}(\widetilde{X}_i = (a_1, \ldots, a_n) | \widetilde{X}_{i-1} = (b_1, \ldots, b_n))\\ &:= \begin{cases} \mathsf{Ws}(X_i = a_n | X_{i-1} = b_n, X_{i-2} = b_{n-1}, \ldots, X_{i-n} = b_1) & \text{falls} \quad a_k = b_{k+1} \text{ für } k < n \\ 0 & \text{sonst} \end{cases}\end{aligned}$$

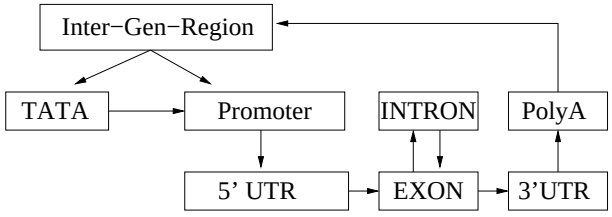
Lässt man die Markoffkette der $\widetilde{X}_i$ laufen und löscht man dann die Redundanz aus der Kette, indem man von jedem $\widetilde{X}_i$ z.B. nur den letzten Eintrag notiert (in der Sprache der HMMs könnte man auch sagen "emittieren läßt"), so erhält man eine zufällige Folge in $\mathcal{Z}$, die der ursprünglichen Markoffdynamik n-ter Ordnung folgt.

Eine andere Verallgemeinerung der Markoffketten, wie wir sie bisher kennengelernt haben, sind die *inhomogenen* Markoffketten, bei denen die Übergangswahrscheinlichkeiten $\mathsf{Ws}(X_i = x | X_{i-1} = y)$ von der Position i abhängen kann. Zum Beispiel könnte bei Genen die Wahrscheinlichkeit eines Basentyps nicht nur von dem oder den linken Nachbarn sondern auch von der Position innerhalb des Kodons abhängen, also von i mod 3.

Wir können insbesondere die in Abschnitt 2.5.1 betrachtete Markoffkette auf den Basentripletts auch als inhomogene Markoffkette höherer Ordnung auf den Basentypen auffassen oder auch als versteckte inhomogene Markoffkette auf einer größeren Menge von Zuständen, die jeweils eine Base emittieren.

2.5.3 Gen-Suche bei Eukaryonten

Burge und Karlin (1997) verwenden in ihrer Software GENSCAN sogenannte *Hidden Semi-Markov-Models* (HSMMs), um Gene in Eukaryonten-DNA zu suchen. Die verschiedenen Abschnitte eines Gens werden als Blöcke modelliert; hier ein vereinfachtes Schema:



Jeder dieser Blöcke emittiert eine Sequenz. Dabei kommen z.B. bei Exons Markoffmodelle fünfter Ordnung zum Einsatz. Die von den Blöcken emittierten Teilsequenzen haben charakteristische Längenverteilungen. Solche Längenverteilungen erfordern zum Teil spezielle algorithmische Lösungen, die uns hier zu weit führen würden...

2.6 Paarweise Alignieren mit pairHMM

PairHMMs unterscheiden sich von gewöhnlichen HMMs dadurch, dass es nicht nur eine Sequenz von Emissionen gibt sondern zwei. Die versteckten Zustände können in eine von beiden oder beide emittieren. Wir können das Alignieren von Sequenzpaaren in diesem Formalismus fassen. Dazu enthalte die Menge $\mathcal{Z}$ außer `Start` und `Ende` folgende Zustände:

D**:** entspricht einer diagonalen Kante im Alignment-Graphen, also einem homologen Paar von Positionen, und emittiert in beide Sequenzen, und zwar die Basen a und b mit Wahrscheinlichkeit e_{ab}.

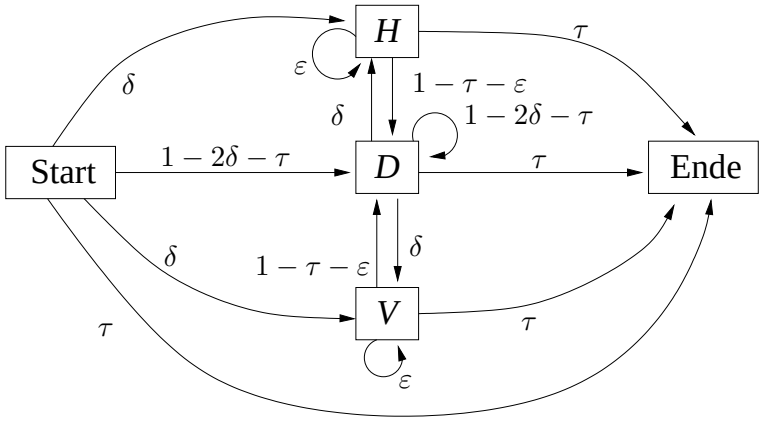
H**:** entspricht einer horizontalen Kante im Alignment-Graphen, also einem Gap in der zweiten Sequenz; emittiert in die erste Sequenz, und zwar die Base x mit Wahrscheinlichkeit q_x.

V**:** entspricht einer vertikalen Kante im Alignment-Graphen, also einem Gap in der ersten Sequenz; emittiert in die zweite Sequenz, und zwar die Base x mit Wahrscheinlichkeit q_x.

Eine Realisierung der versteckten Markoffkette $X_1, \ldots, X_n$ entspricht einem Alignment ohne Basenbelegung. So entspricht z.B. die Realisierung `Start`$DDHHDVDD$`Ende` dem Alignment

```
BBBBB_BB
BB__BBBB
```

Wenn wir verbieten, dass Gaps in der einen Sequenz direkt auf Gaps in der anderen Sequenz folgen, dann können wir uns auf folgende Übergänge beschränken (mit den Beschriftungen der Pfeile als Wahrscheinlichkeiten):



2.6.1 Das wahrscheinlichste Alignment

Das wahrscheinlichste Alignment für die emittierten Sequenzen $a_1, \ldots, a_n$ und $b_1, \ldots, b_m$ kann man mit einer adaptierten Version des Viterbi-Algorithmus berechnen:

Viterbi-Algorithmus für obige pairHMM

$v_D(0,0) := 1$

für $i = 1, \ldots, n$:

$v_H(i,0) := \delta \cdot \varepsilon^{i-1} \cdot \prod_{k=1}^{i} q_{a_k}$

$v_D(i,0) := 0$

$v_V(i,0) := 0$

für $j = 1, \ldots, m$:

$v_V(0,j) := \delta \cdot \varepsilon^{j-1} \cdot \prod_{k=1}^{j} q_{b_k}$

$v_D(0,j) := 0$

$v_H(0,j) := 0$

für $i = 1, \ldots, n; j = 1, \ldots, m$:

$$v_D(i,j) := e_{(a_i b_j)} \cdot \max \begin{cases} (1-2\delta-\tau) \cdot v_D(i-1,j-1) \\ (1-\varepsilon-\tau) \cdot v_H(i-1,j-1) \\ (1-\varepsilon-\tau) \cdot v_V(i-1,j-1) \end{cases}$$

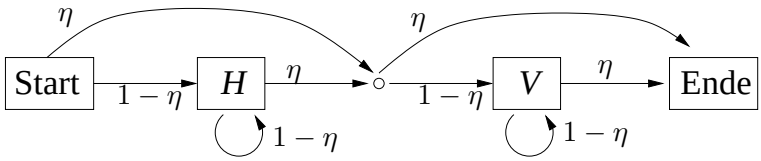
$$v_H(i,j) := q_{a_i} \cdot \max \begin{cases} \delta \cdot v_D(i-1,j) \\ \varepsilon \cdot v_H(i-1,j) \end{cases}$$

$$v_V(i,j) := q_{b_j} \cdot \max \begin{cases} \delta \cdot v_D(i,j-1) \\ \varepsilon \cdot v_V(i,j-1) \end{cases}$$

$v_E := \tau \cdot \max(v_D(n,m), v_H(n,m), v_V(n,m))$

v_E ist die Wahrscheinlichkeit des wahrscheinlichsten Pfades für die Eingabesequenzen. Um den wahrscheinlichsten Pfad zu ermitteln, verfolgt man wie üblich beginnend mit v_E die Argumente der Maxima bis $v_D(0,0)$ zurück.

Wenn man diesen Algorithmus auf der logarithmischen Skala rechnet (was man natürlich tun sollte), sieht das Ganze dem Needleman-Wunsch Algorithmus schon sehr ähnlich. Allerdings sollten die klassischen Alignment-Scores ja log-Likelihood*quotienten* sein. Im Nenner stehen dabei die Likelihoods unabhängig erzeugter Sequenzen. Das können wir – etwas künstlich – auch durch ein pairHMM ausdrücken:



Im Null-Modell unabhängiger Sequenzen haben $a = a_1 \ldots a_n$ und $b = b_1 \ldots b_m$ die Wahrscheinlichkeiten

$$\Pi_{i=1}^{n} q_{x_i} \Pi_{j=1}^{m} q_{x_j} \cdot \eta^2 (1-\eta)^{n+m}.$$

Wir können demnach $\log\left(P_\theta(a,b)/P_0(a,b)\right)$ mit

$$s(a_i, b_j) \quad := \quad \log \frac{e_{a_i b_j}}{q_{a_i} q_{b_j}} + \log \frac{1-2\delta-\tau}{(1-\eta)^2}$$

$$
\begin{aligned}
d &:= -\log \frac{\delta(1-\varepsilon-\tau)}{(1-\eta)(1-2\delta-\tau)} \\
e &:= -\log \frac{\varepsilon}{1-\eta}
\end{aligned}
$$

als $\sum_{i,j} s(a_i, b_j) - d \cdot \sharp\{\text{gaps}\} - e \cdot \sharp\{\text{gap extensions}\}$ schreiben.

Also entspricht der Needleman-Wunsch-Algorithmus einer Variante des Viterbi-Algorithmus zur Berechnung des optimalen log-Likelihoodquotienten-Alignment.

Ebenso wie der Viterbi-Algorithmus läßt sich auch der Vorwärts-Rückwärts-Algorithmus auf pairHMMs übertragen. Somit kann man auch für ein pairHMM für gegebene Sequenzen die Wahrscheinlichkeit berechnen, dass Positionen i und j homolog sind.

2.6.2 Alignment Sampling

Durch ein pairHMM werden den Alignments Wahrscheinlichkeiten zugewiesen. Insbesondere ist damit auch (falls die Modellparameter gegeben sind) die bedingte Wahrscheinlichkeit eines Alignments für ein gegebenes Sequenzpaar $a = a_1 \dots a_n$, $b = b_1 \dots b_m$ definiert. Das wahrscheinlichste Alignment weist besonders wenige Mutationen auf; es muss allerdings nicht das wahre sein. In vielen Fällen hat das wahre Alignment einige Mutationen mehr als das wahrscheinlichste. In diesem Sinne ist das wahrscheinlichste Alignment also atypisch.

Als Alternative zur Ausgabe nur des wahrscheinlichsten Alignments bietet sich an, mehrere Alignments auszugeben, die jeweils eine hohe Wahrscheinlichkeit haben, und sich so ein Bild davon zu machen, was so alles an Alignments in Frage kommt. Unser Ziel ist also nun, aus der bedingten Verteilung der Alignments für die gegebenen Daten zu samplen, d.h. wir suchen einen Algorithmus, der uns zufällige Alignments ausgibt, so dass die Wahrscheinlichkeit, dass ein bestimmtes Alignment ausgegeben wird, gerade dessen bedingter Wahrscheinlichkeit im pairHMM für die gegebenen Daten entspricht.

Dazu berechnen wir mit dem Vorwärtsalgorithmus die Werte $f_z(i,j)$ für $z \in \{D, H, V\}, i \leq n, j \leq m$, also die Wahrscheinlichkeit, dass die versteckte Markoffkette die Sequenzanfänge $a_1 \dots a_i$, $b_1 \dots b_j$ emittiert und dann im Zustand z ist. Die Rekursionsgleichungen für $f.(.,.)$ lassen sich mit den Übergangs- und Emissionswahrscheinlichkeiten wie üblich hinschreiben, z.B. gilt:

$$
\begin{aligned}
f_D(i,j) &= [f_D(i-1,j-1) \cdot (1-2\delta-\tau) \\
&\quad + f_H(i-1,j-1) \cdot (1-\varepsilon-\tau) \\
&\quad + f_V(i-1,j-1) \cdot (1-\varepsilon-\tau)] \cdot e_{a_i b_j}
\end{aligned}
$$

Anders als bei den Alignment-Optimierungsalgorithmen entscheiden wir uns beim "Zurückverfolgen" nicht immer für den Zustand, der den höchsten Beitrag zum Wert $f_z(i,j)$ des aktuellen Zustands z an der Stelle (i,j) geliefert hat, sondern wir entscheiden uns zufällig. Dabei entsprechen die Wahrscheinlichkeiten der möglichen Zustände ihren relativen Beiträgen zu $f_z(i,j)$. Ist z.B. bei $(i+1, j+1)$ der Zustand D gewählt worden, so wählen wir für (i,j) ebenfalls D mit Wahrscheinlichkeit $f_D(i,j) \cdot (1 - 2\delta - \tau) \cdot e_{a_{i+1} b_{j+1}} / f_D(i+1, j+1)$.

Wir erhalten damit ein Alignment, d.h. eine Folge $x_1, \ldots, x_n \in \{D, H, V\}^n$ gemäß der Wahrscheinlichkeit

$$\text{Ws}\{X = (x_1, \ldots, x_n, \texttt{Ende}) | \text{Sequenzen}\}$$

Es gilt nämlich Folgendes, wobei $\epsilon_k(i,j)$ das Ereignis bezeichnet, dass $X_1, \ldots, X_k$ die Sequenzen $a_1, \ldots, a_i$ und $b_1, \ldots, b_j$ emittiert:

$$\begin{aligned}
&\text{Ws}\left(\exists_k X_k = D, \epsilon_k(i,j) | \exists_k X_{k+1} = D, X_{k+2} = z_{k+2}, \ldots, X_n = z_n, \epsilon_{k+1}(i+1, j+1)\right) \\
&= \quad \text{Ws}\left(\exists_k X_k = D, \epsilon_k(i,j) | \exists_k X_{k+1} = D, \epsilon_{k+1}(i+1, j+1))\right) \\
&= \quad \frac{\text{Ws}\left(\exists_k X_k = D, X_{k+1} = D, \epsilon_{k+1} = (i+1, j+1))\right)}{\text{Ws}\left(\exists_k X_{k+1} = D, \epsilon_{k+1} = (i+1, j+1))\right)} \\
&= \quad \frac{f_D(i,j) \cdot P_{DD} \cdot e_{a_{i+1} b_{j+1}}}{f_D(i+1, j+1)}
\end{aligned}$$

Entsprechendes folgt völlig analog für die anderen Zustandsübergänge.

2.6.3 Das akkurateste Alignment

Mit dem Vortwärts-Rückwärts-Algorithmus können wir die auf gegebene Sequenzen bedingte Wahrscheinlichkeit berechnen, dass die Positionen i und j homolog sind. Wir bezeichnen diese Wahrscheinlichkeit im Folgenden mit h_{ij}. Wenn wir davon ausgehen, dass das wahre Alignment Resultat eines Zufallsprozesses gemäß des pairHMM ist, können wir uns fragen, was die erwartete Anzahl an Positionen ist, an denen ein gegebenes Alignment mit dem wahren Alignment übereinstimmt. Diesen auf die gegebenen Sequenzen bedingten Erwartungswert bezeichnen wir als die *Akkuratheit* des Alignments.

Das *akkurateste* Alignment ist also das Alignment, welches $\sum_{(i,j)} h_{ij}$ maximiert, wobei (i,j) die Menge der Paare homologer Positionen durchläuft, die im jeweiligen Alignment gegeben sind. Das akkurateste Alignment finden wir wieder mit dynamischer Programmierung. Dazu sei $A_{(i,j)}$ die Akkuratheit des akkuratesten Alignments der Sequenzanfänge $a_1, \ldots, a_i$ und $b_1, \ldots, b_j$. Es gilt dann die einfache Rekursion

$$A(i,j) = \max \begin{cases} A(i-1, j-1) + h_{ij} \\ A(i-1, j) \\ A(i, j-1) \end{cases}$$

Wir berechnen damit A_{nm}, die Akkuratheit des akkuratesten Alignments der kompletten Sequenzen und ermitteln anschließend das akkurateste Alignment, indem wir zurückverfolgen, welche Positionspaare zu A_{nm} beigetragen haben.

3 Multiples Alignment

Gegeben seien nun n Sequenzen

$$\begin{aligned} x^1 &= (x_1^1, \ldots, x_{l_1}^1) \\ x^2 &= (x_1^2, \ldots, x_{l_2}^2) \\ &\vdots \\ x^n &= (x_1^n, \ldots, x_{l_n}^n) \end{aligned}$$

mit $x_j^i \in \mathcal{A}$. Gesucht ist ein gemeinsames Alignment für diese Sequenzen.

3.1 Score-Schemata für Multiple Alignments

Sei m ein multiples Alignment, m_i dessen i-te Spalte und m_i^j das Symbol in Spalte i und Sequenz j.

Zum Beispiel für $m = \begin{matrix} \texttt{AACG\_\_\_TT\_} \\ \texttt{A\_\_GCCTTA\_} \\ \texttt{A\_\_GG\_\_TTA} \end{matrix}$ ist $m_5 = \begin{matrix} - \\ \texttt{C} \\ \texttt{G} \end{matrix}$ und $m_5^2 = \texttt{C}$.

Wir beschränken uns im Folgenden auf Score-Schemata der Form $S(m) = G + \sum S(m_i)$, wobei G die Gaps bestraft (z.B. mit affiner Gap-Penalty) und $S(.)$ die Mismatches und Matches bestraft bzw. belohnt.

Wenn das Alignment nicht einfach Ähnlichkeiten sondern Homologien der Sequenzen darstellt, sollte das Score-Schema den Stammbaum der Sequezen berücksichtigen. Sinnvoll wäre dann etwa die Summe der Scores der Alignments entlang jeder Kante. Das ist jedoch meistens sehr schwierig, da der Stammbaum in der Regel nicht bekannt ist und selbst wenn dem so wäre, würde ein Alignment der Sequenzen an den Blättern noch nicht die volle Information über die Sequenzen an den inneren Knoten und Alignments beinhalten.

Mit c_{ia} bezeichnen wir die Anzahl der $a \in \mathcal{A}$ in m_i und es sei $c_i = (c_{ia})_{a \in \mathcal{A}}$.

Der *Entropie-Score* ist dann definiert durch

$$S(m_i) := -\sum_a c_{ia} \log \frac{c_{ia}}{\sum_b c_{ib}}.$$

Ein weiterer Score für multiple Alignments ist der Summe-aller-Paare(SP)-Score mit

$$S(m_i) := \sum_{k<l} s(m_i^k, m_i^l).$$

Dabei ist $s(m_i^k, m_j^l)$ die Match/Mismatch-Penalty eines paarweisen Alignment-Score-Schemas, also etwa entsprechend einer PAM- oder BLOSUM-Matrix.

Stehen also in einer Spalte zwei $\texttt{A}$ und zwei $\texttt{G}$, so beträgt der Score $s(\texttt{A},\texttt{A}) + s(\texttt{G},\texttt{G}) + 4 \cdot s(\texttt{A},\texttt{G})$. Stehen hingegen in der Spalte drei $\texttt{A}$ und nur ein $\texttt{G}$ in der Spalte, ist der Score $3 \cdot s(\texttt{A},\texttt{A}) + 3 \cdot s(\texttt{A},\texttt{G})$. Wie gut der Unterschied motiviert ist, ist nicht ganz klar. Beide Spalten könnten mit nur einer Transition erklärt werden.

3.2 Multiples Alignment mit Dynamischer Programmierung

Im Prinzip kann man auch optimale multiple Alignments bei affiner Gap-Strafe mit dynamischer Programmierung berechnen. Sei dazu $F_{i_1,i_2,\ldots,i_n}$ der Score des besten Alignments der Sequenzanfänge $x_1^1 \ldots x_{i_1}^1$, $x_1^2 \ldots x_{i_2}^2$, …, $x_1^n \ldots x_{i_n}^n$. Wir beschränken uns hier auf lineare Gap-Penalty. Dann sieht die Rekursion zur Berechnung von F so aus:

$$F_{i_1,\ldots,i_n} = \max_{\{(\delta_1,\ldots,\delta_n)|\delta_1+\cdots+\delta_n>0\}} \{F_{i_1-\delta_1,\ldots,i_n-\delta_n} + S(\delta_1 x_{i_1}^1, \ldots, \delta_n x_{i_n}^n)\}$$

Dabei ist für $x \in \mathcal{A} \cup \{\_\}$:

$$\delta_i x := \begin{cases} x & \text{für} \quad \delta_i = 1 \\ \_ & \text{für} \quad \delta_i = 0 \end{cases}$$

In jedem Schritt muss also über $2^n - 1$ mögliche Werte maximiert werden. Damit ist die Laufzeit exponentiell in der Anzahl der Sequenzen.

3.2.1 Bei SP-Scores Zeit sparen: MSA

Carillo und Lipman (1988) zeigen, dass man bei der Berechnung des optimalen multiplen SP-Score-Alignments mit dynamischer Programmierung nicht alle Werte $F_{i_1,\ldots,i_n}$ berechnen muss.

Ist m ein multiples Alignment, so sei m^{kl} das darin enthaltene Alignment der Sequenzen x^k und x^l. Es gilt also $S(m) = \sum_{k<l} S(m^{kl})$. Es sei m das optimale multiple Alignment. Ist β der Score irgendeines anderen multiplen Alignments, man denke an ein schon recht gutes, das durch eine schnelle Heuristik gefunden wurde, so gilt klarerweise $\beta \leq S(m)$. Ebenso trivial ist, dass für das beste paarweise Alignment m_*^{kl} der Sequenzen x^k und x^l die Ungleichung $S(m^{kl}) \leq S(m_*^{kl})$ gilt. Damit erhalten wir:

$$\beta \leq \sum_{i<j} S(m^{ij}) \leq S(m^{kl}) - S(m_*^{kl}) + \sum_{i<j} S(m_*^{ij})$$

Daraus folgt:

$$S(m^{kl}) \geq \beta + S(m_*^{kl}) - \sum_{i<j} S(m_*^{ij}) =: \beta^{kl}$$

Der MSA-Algortihmus von Carillo und Lipman ist nun:

- Berechne β^{kl} für jedes Sequenzpaar (x^k, x^l).
- Berechne für jedes Sequenzpaar (x^k, x^l) die Menge B^{kl} der Positionspaare (i, j), für die das beste Alignment durch (i, j) mindestens Score β^{kl} hat. (Das geht mit einer "Viterbi-Variante" des Vorwärts-Rückwärts-Algorithmus in Zeit $O(L^2)$.)
- Berechne dann $F_{i_1,i_2,\ldots,i_n}$ für alle $(i_1, i_2, \ldots, i_n)$, für die $(i_k, i_l) \in B^{kl}$ für alle enthaltenen Paare (i_k, i_l) gilt. (Alle anderen spielen keine Rolle.)

Die Rechenzeit des letzten Schrittes bleibt im Allgemeinen exponentiell in der Anzahl der Sequenzen, aber man kann auf diese Weise einen nicht unbedeutenden Faktor einsparen. Veranschaulicht wird diese Idee in einer Abbildung in Durbin et al. (1998) auf Seite 144. MSA soll bis etwa 7 Protein-Sequenzen praktikabel sein.

3.3 Progressives Alignment (CLUSTALW)

Der Ansatz, Multiples Alignment progressiv zu betreiben, besteht darin, zunächst zwei Sequenzen zu alignieren, dann eine dritte gegen das gefundene Alignment zu alignieren und so weiter...

Das bekannteste progressiv alignierende Programm ist CLUSTALW, siehe Thompson, Higgins und Gibson (1994). Die groben Teilschritte von CLUSTALW sind:

1. Aligniere jedes Paar von Sequenzen mittels dynamischer Programmierung.
2. Weise jedem alignierten Paar eine evolutionäre Distanz zu.
3. Konstruiere aus den Distanzen einen phylogenetischen Baum der Sequenzen mittels Neighbour-Joining (siehe Abschnitt 4.1.2).
4. Aligniere längs des Baumes die Sequenz-Paare, Sequenz/Profil-Paare und Profil-Paare (siehe Abschnitt 3.4).

CLUSTALW ist recht schnell und liefert ansehnliche Ergebnisse, was u.a. durch folgende “Tricks” erreicht wird:

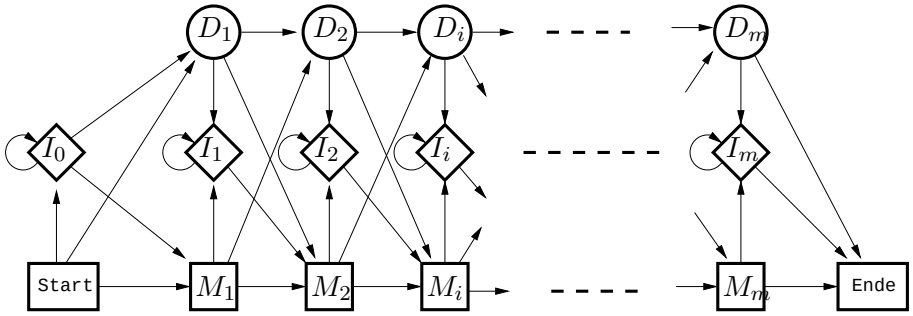
- Verwendet werden SP-Scores für die Profil-Alignments, die so gewichtet werden, dass große Teilfamilien sehr ähnlicher Sequenzen keinen übermäßig großen Einfluss haben.
- Substitutionsmatrizen werden je nach erwarteter Ähnlichkeit der Sequenzen flexibel ausgewählt.
- Gap-Penalties sind bei Protein-Sequenzen strenger in hydrophoben Regionen und milder in hydrophilen.
- Gap-Penalties werden erhöht, falls an anderen, aber benachbarten Positionen Gaps auftreten. Damit werden Alignments zusätzlich bevorzugt, die Gaps möglichst an dieselbe Stelle bringen.
- Der verwendete Baum kann automatisch verändert werden, falls ein Profil-Alignment einen sehr schlechten Score erbringt.

Natürlich ist CLUSTALW problematisch, wenn aus dem multiplen Alignment letztlich ein phylogenetischer Baum geschätzt werden soll. Es besteht dann die Gefahr einer gewissen Bias zum verwendeten Neighbour-Joining-Baum. Auf Ansätze, multiple Alignments und phylogenetische Bäume simultan zu schätzen, werden wir in Abschnitt 4.7 noch einmal kurz zurückkommen.

3.4 ProfilHMM-Training

Mit einem ProfilHMM kann man für eine Grupppe von Sequenzen modellieren, welche Positionen typischerweise mit bestimmten Basen besetzt sind und welche Basen dort ansonsten auch noch toleriert werden, welche Positionen bei einigen Sequenzen fehlen und wo etwas eingefügt werden kann, was nicht typisch für diese Gruppe von Sequenzen ist.

Die versteckte Markoffkette eines ProfilHMM bewegt sich durch folgenden gerichteten Graphen:



Die Zustände M_i stehen für die Positionen einer Konsensus-Sequenz der betrachteten Sequenzen. Die Zustände I_i in der mittleren Zeile stehen für Positionen, die nur in einzelnen Sequenzen vorkommen, aber nicht für die Sequenz-Gruppe typisch sind. Die Zustände D_i in der oberen Zeile werden durchlaufen, wenn in einer Sequenz Positionen fehlen, die in der Konsensus-Sequenz vorkommen.

Die Modellvorstellung ist also, dass eine zu dieser Gruppe gehörige Sequenz dadurch erzeugt wird, dass die versteckte Markoffkette von Start bis Ende läuft und dabei Zustände M_i und I_i Aminosäuren bzw. Basen emittieren. Die entsprechenden Emissionswahrscheinlichkeiten bezeichnen wir mit $e_{M_i}(.)$ und $e_{I_i}(.)$. Die Übergangswahrscheinlichkeiten bezeichnen wir mit $P_{M_iM_{i+1}}$, $P_{M_iD_{i+1}}$, $P_{I_iI_i}$ etc...

3.4.1 Alignment einer Sequenz gegen ein Profil

Wir können nun etwa eine neue Sequenz $x = x_1 \dots x_n$ gegen das Profil einer Gruppe von Sequenzen alignieren, indem wir den Viterbi-Algorithmus auf ProfilHMMs übertragen. Sei dazu $V_j^X(i)$ der Logarithmus aus dem Quotienten der Wahrscheinlichkeit des besten Pfades für $x_1 \dots x_i$ im Teil-ProfilHMM bis zur Spalte der Zustände M_j, I_j und D_j, wobei der Pfad im Zustand X_j enden muss, und der Wahrscheinlichkeit der Teilsequenz im Nullmodell, bei dem alle Postionen unabhängig voneinander gemäß den Basen- bzw. Aminosäurehäufigkeiten $q_\cdot$ zusammengwürfelt wurden. Die Rekursionsgleichungen zur Berechnung von $V_\cdot^\cdot(.)$ lassen sich leicht hinschreiben, z.B. gilt:

$$V_j^M(i) = \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log P_{M_{j-1}M_j} \\ V_{j-1}^D(i-1) + \log P_{D_{j-1}M_j} \\ V_{j-1}^I(i-1) + \log P_{I_{j-1}M_j} \end{cases}$$

Wir können also V und damit das optimale Alignment der Sequenz x gegen ein ProfilHMM, dessen Übergangs- und Emissionswahrscheinlichkeiten bekannt sind, in Zeit $O(nm)$ berechnen.

3.4.2 ProfilHMM-Training mit unalignierten Sequenzen

Gegeben seien unalignierte Sequenzen, die in einem biologischen Sinne als zu einer Gruppe gehörig aufgefasst werden. Wie sollte man dann für ein ProfilHMM der Länge m die Übergangs- und Emissionswahrscheinlichkeiten schätzen? (Für m wählen wir etwa die durchschnittliche Länge der Sequenzen.)

Naheliegend ist, das Baum-Welch-Training oder das Viterbi-Training auf ProfilHMMs zu übertragen, was auch ganz direkt geht. Eine Besonderheit bei ProfilHMMs ist allerdings, dass sehr viele Parameter

zu schätzen sind, die wir als einen Vektor $\theta = (P_{\cdot}(.), e_{\cdot}(.))$ in einem sehr hochdimensionalen Raum zusammenfassen können. In hochdimensionalen Räumen ist viel Platz für lokale Optima. Daher ist bei ProfilHMMs an Optimierungsalgorithmen zu denken, die nicht nur gierig optimieren, sondern auch die Chance haben, lokale Optima zu verlassen. Ein solcher Algorithmus ist *Simulated Annealing*.

Die Idee des Simulated Annealing kommt aus der Physik. Hat eine Konfiguration x eines physikalischen Systems eine Energie $E(x)$ und wird die Konfiguration des Systems durch eine Temperatur T etwas durcheinander gebracht, so entspricht in gewissen Modellen die Wahrscheinlichkeit(sdichte), dass das System gerade im Zustand x ist, der Gibbs-Boltzmann-Wahrscheinlichkeit(sdichte) $P(x) = e^{E(x)/T}/z$, wobei z lediglich eine Normierungskonstante ist, die dafür sorgt, dass die Summe der Wahrscheinlichkeiten (bzw. das Integral über die Dichten) den Wert 1 annimmt. Je niedriger die Temperatur ist, desto näher ist das System am energetischen Optimum. Es gibt physikalische Systeme, etwa gewisse Kristall-Strukturen, die nur dann in den Bereich des energetischen Optimums kommen, wenn sie langsam von einer hohen zu einer niedrigen Temperatur abgekühlt werden.

Das Prinzip des Simulated Annealing lässt sich auf viele Optimierungsprobleme anwenden, etwa folgendermaßen auf die Suche nach den optimalen Parametern $\widehat{\theta}$ eines ProfilHMMs für eine Gruppe von Sequenzen:

Wähle zu Beginn eine "simulierte Temperatur" T_0 und einen vorläufigen Parametervektor θ_0. Wähle dann für $i = 0, 1, \ldots$ jeweils in der Nähe von θ_i einen Kandidaten θ' für θ_{i+1} und akzeptiere diesen mit Wahrscheinlichkeit

$$\min\left\{1, (P_{\theta'}/P_{\theta_i})^{1/T_i}\right\},$$

wobei P_θ die Wahrscheinlichkeit ist, dass ein ProfilHMM mit Parametervektor θ die beobachteten Daten emittiert, die sich mit dem Vorwärtsalgorithmus berechnen lässt. Bei nicht-Akzeptanz setze $\theta_{i+1} := \theta_i$. Dabei lässt man T_i (vorsichtig) gegen 0 konvergieren (für $i \to \infty$). Wenn alles gut geht, konvergiert dann θ_i gegen $\widehat{\theta}$.

Viterbi-Training mit Simulated Annealing Bei dieser von Eddy (1995) vorgeschlagenen Variante des Viterbi-Trainings wird nicht jeweils der wahrscheinlichste Pfad durch den ProfilHMM-Graphen zur nächsten Iteration der Parameterschätzung verwendet, sondern ein zufälliger Pfad z_i, der mit Wahrscheinlichkeit

$$\frac{\text{Ws}_{\theta_i}(z_i, \text{Daten})^{1/T_i}}{\sum_z \text{Ws}_{\theta_i}(z, \text{Daten})^{1/T_i}}$$

aus allen möglichen Pfaden ausgewählt wird. Dies ist algorithmisch effizient machbar, indem beim Vorwärtsalgorithmus statt der aktuellen Parameter $\theta_i = (P_{\cdot,i}(.), e_{\cdot,i}(.))$ die mit $1/T_i$ potenzierten Parameter $(P_{\cdot,i}(.)^{1/T_i}, e_{\cdot,i}(.)^{1/T_i})$ verwendet werden und man beim Zurückverfolgen des Pfades jeweils zufällig gemäß der Beiträge der eingehenden Kanten entscheidet (ähnlich wie beim Alignment-Sampling, vgl. Abschnitt 2.6.2). Man mache sich klar, dass man damit tatsächlich den Pfad z_i mit der angestrebten Wahrscheinlichkeit erhält!

Wie beim Viterbi-Training üblich, werden die Übergangs- und Emissions-Wahrscheinlichkeiten aus dem jeweils aktuellen Pfad jeder Iteration geschätzt. Wie beim Simulated Annealing üblich, lassen wir

T_i gegen 0 gehen.

4 Phylogenetische Bäume und Modelle der Sequenzevolution

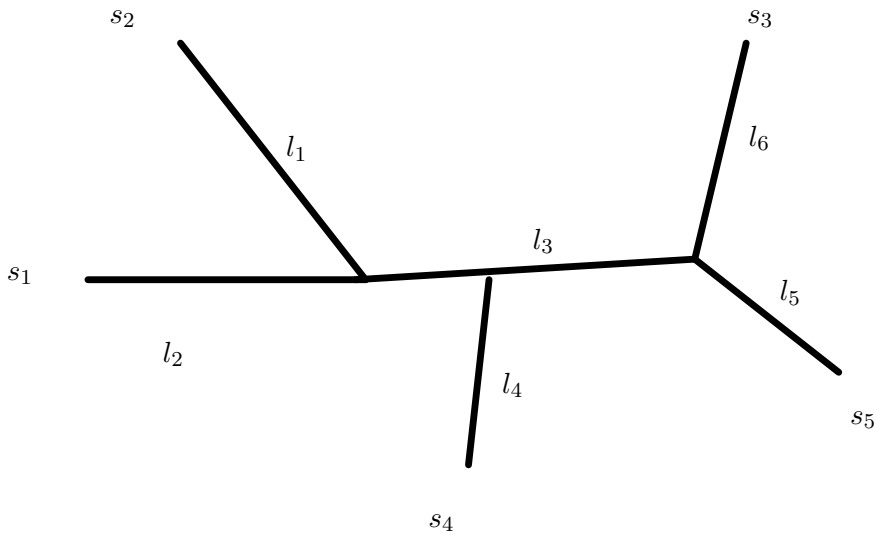
4.1 Distanzbasierte Phylogenieschätzung

Gegeben ist eine Menge $S = \{s_1, s_2, \ldots, s_n\}$ (z.B. von Sequenzen, Arten, Individuen, ...) und eine Distanzmatrix $(d_{ij})_{ij \leq n}$, d.h. es gilt für alle $i, j, k \leq n$:

$$d_{ij} = d_{ji}, \qquad d_{ij} + d_{jk} \geq d_{ik} \quad \text{(Dreiecksungleichung)}, \qquad i = j \Leftrightarrow d_{ij} = 0$$

d_{ij} ist der (evtl. geschätzte) Abstand zwischen s_i und s_j.

Gesucht: ein binärer Baum, dessen Blätter mit $s_1, s_2, \ldots, s_n$ und dessen Kanten mit Längen $l_1, l_2, \ldots, l_k \in \mathbb{R}_{>0}$ beschriftet sind, so dass die Abstände der Knoten im Baum *möglichst gut* mit den d_{ij} übereinstimmen.



4.1.1 Ein Cluster-Verfahren: UPGMA

Cluster sind Teilmengen von S. Hierarchische Clusterverfahren beginnen mit den einelementigen Clustern und vereinigen schrittweise die Cluster minimaler Distanz. Clusterverfahren unterscheiden sich vor allem darin, wie Distanzen zwischen mehrelementigen Clustern definiert sind, Beispiele sind:

single linkage: $d(C, C') = \min_{s_i \in C, s_j \in C'} d_{ij}$

complete linkage: $d(C, C') = \max_{s_i \in C, s_j \in C'} d_{ij}$

means: $d(C, C') =$ mittlere Distanz der Elemente von C und C'.

UPGMA (Sokal & Michener, 1985) ist ein “means”-Clusterverfahren. UPGMA steht für **U**nweighted **P**airwise **G**rouping **M**ethod using **A**rithmetic means.

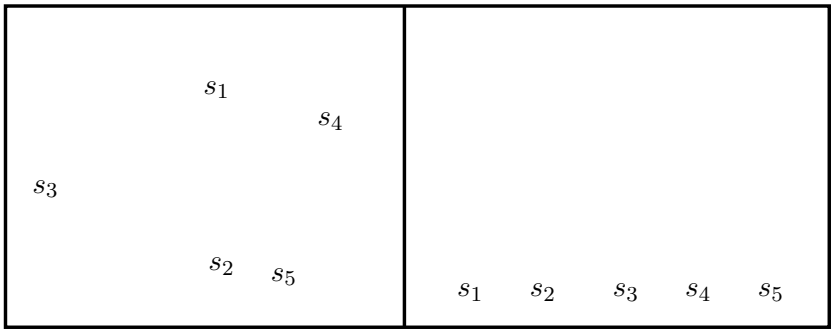
für $i \leq n$ sei $C_i := \{s_i\}$
$\mathcal{C} := \{C_1, \ldots, C_n\}$
$m := n$ wiederhole

- $m := m + 1$
- Suche $C_i, C_j \in \mathcal{C}$ mit minimalem $d_{ij} > 0$
- $C_m := C_i \cup C_j$
- $\mathcal{C} := \mathcal{C} \cup \{C_m\} \setminus \{C_i, C_j\}$
- Für alle $C_k \in \mathcal{C}$ setze

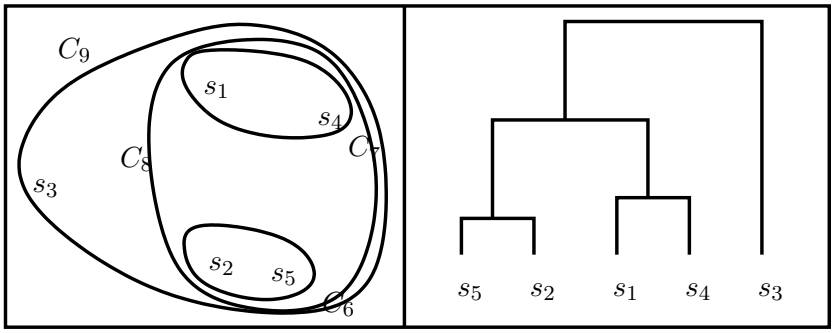
$$d_{km} := d_{mk} := \frac{1}{|C_k| \cdot |C_m|} \sum_{s_x \in C_k, s_y \in C_m} d_{xy}$$

bis $C_m = \{s_1, \ldots, s_n\}$.

Beispiel für UPGMA Distanzen entsprechen euklidischen Abständen auf linker Seite

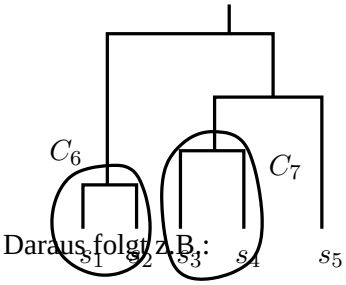


UPGMA wird folgendermaßen Clustern:



Wähle Astlängen entsprechend der Abstände der Cluster.

Wann findet UPGMA den richtigen Baum? Ideale Situation:



Angenommen die Distanzen $\{d_{ij}\}$ sind exakt zu einem Baum kompatibel, der die Eigenschaft der molekularen Uhr erfüllt, d.h. alle Blätter sind gleich weit von einer Wurzel entfernt.

Dann gilt

$$d_{13} = d_{14} = d_{23} = d_{24}$$

Daraus folgt z.B.:

$$d_{67} = \frac{1}{2 \cdot 2} \cdot (d_{13} + d_{14} + d_{23} + d_{24}) = d_{13}$$

UPGMA behandelt also jedes Cluster wie ein Blatt und weist genau die Distanzen zu, die die in den Clustern enthaltenen Blätter tatsächlich haben.

Wir sehen also: **Unter der Annahme der molekularen Uhr wird UPGMA den exakt passenden Baum finden, falls er existiert.**

Kann man bereits den Distanzen ansehen, ob sie mit einem Baum mit molekularer Uhr verträglich sind?

Lemma *Sei D eine Distanz-Matrix für $(s_1, \ldots, s_n)$. Dann sind äquivalent:*

(a) *Es gibt einen Baum, der die molekulare-Uhr-Eigenschaft hat, und dessen Blätter die in D gegebenen Distanzen haben (als Summen der dazwischenliegenden Kantenlängen).*

(b) *D ist ultrametrisch, d.h. es gilt*

$$\forall_{i,j,k} \exists_{i \in \{i,j,k\}} : d_{jk} < d_{ij} = d_{ik}$$

Beweis: (a)⇒(b): Sei h der jüngste gemeinsame Vorfahr von i, j und k, und sei oBdA d_{jk} das Minimum von $\{d_{ij}, d_{jk}, d_{ik}\}$. Also sind j und k direkte Nachbarn. Aus der molekularen Uhr Eigenschaft folgt, dass i, j und k alle die selbe Distanz zu h haben. Dann hat i sowohl zu j als auch zu k die doppelte Distanz wie zu h. Also gilt $d_{ij} = d_{ik}$.

(b)⇒(a): Induktion über n: Angenommen der Baum kann für jeweils höchstens n Taxa $(s_1, \ldots, s_n)$ rekonstruiert werden. Jetzt seien $n+1$ Taxa $(s_1, \ldots, s_{n+1})$ gegeben und es seien S_L und S_R der linke und der rechte Teilbaum des Teilbaums für $(s_1, \ldots, s_n)$. (Was rechts und was links ist, ist hier beliebig. Jedenfalls unterteilen wir den Baum an der Wurzel.) Sei $x \in S_L$ und $y \in S_R$. Wir unterscheiden drei Fälle:

erster Fall: $d(s_{n+1}, x) = d(s_{n+1}, y)$

Wegen der Ultrametrik-Eigenschaft gilt $d(s_{n+1}, s_i) = d(s_{n+1}, x)$ für alle s_i mit $i \leq n$ (Übung!). Also erhalten wir einen Baum, der die molekulare Uhr Eigenschaft erfüllt, indem wir an den Baum für $(s_1, \ldots, s_n)$ eine neue Wurzel im Abstand $(d(s_{n+1}, x) - d(x, y))/2$ anhängen und dort das neue Blatt s_{n+1} an eine Kante der Länge $d(s_{n+1}, x)/2$.

zweiter Fall: $d(s_{n+1}, x) < d(s_{n+1}, y) = d(x, y)$

Dann gilt $d(s_{n+1}, x) < d(x, z) = d(s_{n+1}, z)$ für alle $z \in S_R$.

Falls $d(s_{n+1}, x) = d(s_{n+1}, u)$ für alle $u \in S_L$ gilt, hängen wir s_{n+1} an eine Kante der Länge $d(s_{n+1}, x)/2$ zwischen S_L und die Wurzel, und zwar $d(s_{n+1}, x)/2$ von den Blättern in S_L entfernt. Wir erhalten so den passenden Baum.

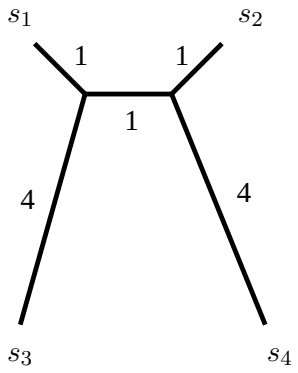
Falls hingegen ein $u \in S_L$ mit $d(s_{n+1}, x) \neq d(s_{n+1}, u)$ in S_L existiert, kann s_{n+1} nach Induktionsannahme in den Teilbaum S_L eingefügt werden und das Ergebnis erfüllt weiterhin die molekulare Uhr Eigenschaft. Da $d(s_{n+1}, z) = d(x, z)$ für alle $z \in S_R$ gilt, stimmen die Distanzen im ganzen Baum.

dritter Fall: $d(s_{n+1}, y) < d(s_{n+1}, x) = d(x, y)$

Analog zum zweiten Fall.

□

Wann findet UPGMA <u>nicht</u> den richtigen Baum? Wir betrachten als Beispiel den folgenden Baum:



$d_{12} = 3 \qquad d_{23} = 6$
$d_{13} = 5 \qquad d_{24} = 5$
$d_{14} = 6 \qquad d_{34} = 9$

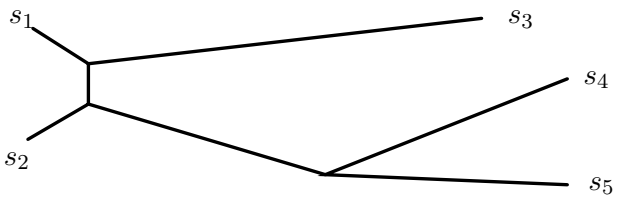
Was wuerde UPGMA tun? Es würde zuerst s_1 und s_2 vereinigen und damit sofort in die Irre laufen!

Frage: Gibt es einen Algorithmus, der einen exakt passenden Baum sicher findet? Antwort: Ja, sofern ein solcher Baum existiert; Siehe nächster Abschnitt.

4.1.2 Neighbour Joining (Saitou & Nei, 1987)

Idee: Verwende gewichtete Distanzen

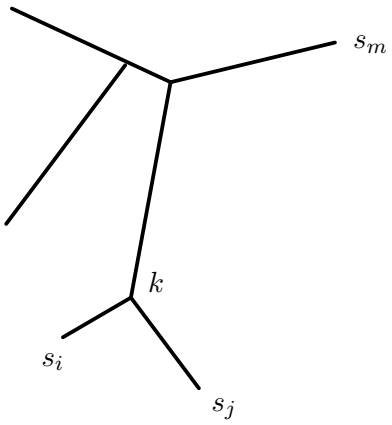
$$D_{ij} := d_{ij} - (r_i + r_j), \qquad \text{mit} \quad r_i \;=\; \frac{1}{n-2} \sum_k d_{ik} \;=\; \frac{n-1}{n-2} \cdot \overline{d_{i\cdot}}$$



Neighbour Joining Algorithmus Eingabe $T = \{s_1, \ldots, s_n\}$ mit Distanzen $(d_{ij})_{i,j \leq n}$

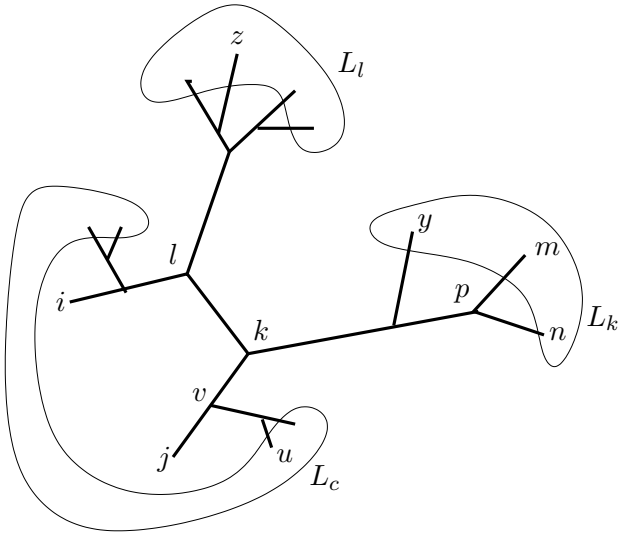
NeighbourJoining(T):

- fertig falls $n = 1$
- berechne alle D_{ij}
- finde in T die Knoten s_i und s_j mit minimalem D_{ij}
- definiere inneren Knoten k mit $\forall_m : d_{km} := \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$
- NeighbourJoining($\{k\} \cup T \setminus \{s_i, s_j\}$)



Satz (Neighbour-Joining-Theorem, Studier & Keppler, 1988): Falls ein Baum existiert, dessen Blätter $\{s_i\}_{i \leq n}$ exakt die Distanzen $(d_{ij})_{i,j \leq n}$ haben (als Summen der Kantenlängen), wird *Neighbour Joining* diesen Baum finden.

Beweis: Zu zeigen ist, dass i und j Nachbarn sind, falls D_{ij} minimal ist. Wir beweisen dies durch Widerspruch. Seien also i und j *nicht* Nachbarn. Es gibt also mindestens zwei benachbarte innere Knoten k und l zwischen j und i. k sei der, der näher an j liegt. L_k seien die Blätter am Teilbaum, der durch k von i und j getrennt wird, und L_l die Blätter des entsprechenden Teilbaumes an l. OBdA sei $|L_k| \leq |L_l|$. Die Menge aller Blätter sei L und es sei $L_c := L \setminus (\{i, j\} \cup L_k \cup L_l)$. Wir beschränken uns auf den Fall $|L_k| > 1$. Der Fall $|L_k| = 1$ wird in einer Übung behandelt. m und n seien benachbarte Knoten in L_k und p sei der innere Knoten zwischen ihnen.



Da die Distanzen die Summen der Kantenlängen sind, gilt $d_{iy} + d_{jy} = d_{ij} + 2d_{ky}$ und $d_{my} + d_{ny} = d_{mn} + 2d_{py}$. Daraus folgt:

$$d_{iy} + d_{jy} - d_{my} - d_{ny} = d_{ij} + 2d_{ky} - 2d_{py} - d_{mn}.$$

Analog gilt für alle $z \in L_l$:

$$d_{iz} + d_{jz} - d_{mz} - d_{nz} = d_{ij} - d_{mn} - 2d_{pk} - 2d_{lk}$$

Nach Definition der $D_{..}$ gilt:

$$D_{ij} - D_{mn} = d_{ij} - d_{mn} - \frac{1}{N-2}\left(\sum_u d_{iu} + d_{ju} - d_{mu} - d_{nu}\right)$$

Wegen obiger Gleichung gilt aber:

$$\begin{aligned}
\sum_u d_{iu} + d_{ju} - d_{mu} - d_{nu} &= \left(\sum_{y\in L_k} d_{iy} + d_{jy} - d_{my} - d_{ny}\right) + \left(\sum_{z\in L_l} d_{iz} + d_{jz} - d_{mz} - d_{nz}\right) \\
&\quad + d_{ii} + d_{ij} - d_{mi} - d_{ni} + d_{ij} + d_{jj} - d_{mj} - d_{nj} + \left(\sum_{u\in L_c} d_{iu} + d_{ju} - d_{mu} - d_{nu}\right) \\
&= \left(\sum_{y\in L_k} d_{ij} + 2d_{ky} - 2d_{py} - d_{mn}\right) + \left(\sum_{z\in L_l} d_{ij} - d_{mn} - 2d_{pk} - 2d_{lk}\right) \\
&\quad + 2d_{ij} - d_{mi} - d_{ni} - d_{mj} - d_{nj} + \left(\sum_{u\in L_c} d_{ij} - d_{mn} + 2d_{uv} - 2d_{pu}\right) \\
&= (N-2)(d_{ij} - d_{mn}) + \left(\sum_{y\in L_k} (2d_{ky} - 2d_{py})\right) - \left(\sum_{z\in L_l} (2d_{pk} + 2d_{lk})\right) \\
&\quad - 4d_{kp} - 2d_{mn} - 2\sum_{u\in L_c} d_{pv}
\end{aligned}$$

Daraus folgt:

$$D_{ij} - D_{mn} = \frac{\left(\sum_{y \in L_k} 2d_{py} - 2d_{ky}\right) + \left(\sum_{z \in L_l} 2d_{pk} + 2d_{lk}\right) + \left(\sum_{u \in L_c} 2d_{pv}\right) + 4d_{kp} + 2d_{mn}}{N-2}$$

Wegen $d_{py} - d_{ky} > -d_{pk}$ folgt daraus:

$$D_{ij} - D_{mn} > 2d_{pk}(|L_l| - |L_k|)/(N-2) > 0$$

Also kann D_{ij} nicht minimal sein. □

Problem: Meistens sind die Distanzen mit keinem Baum verträglich (z.B. weil es nur mehr oder weniger ungenaue Schätzungen sind). Dann kann uns das Neighbour-Joining-Theorem auch nicht wirklich helfen.

Oft sind dann Verfahren erfolgreicher, die *mehr Informationen* als nur die ungefähren Distanzen ausnutzen!

4.2 Parsimonische Baumrekonstruktion

Gegeben: n homologe Sequenzen, z.B. DNA oder Proteine

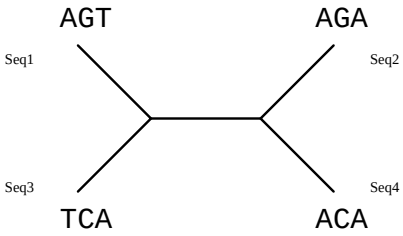
$$\begin{array}{rcccc} x^1 & = & x_1^1, & x_2^1, \ldots, & x_m^1 \\ x^2 & = & x_1^2, & x_2^2, \ldots, & x_m^2 \\ \vdots & \vdots & & \ddots & \vdots \\ x^n & = & x_1^n, & x_2^n, \ldots, & x_m^n \end{array}$$

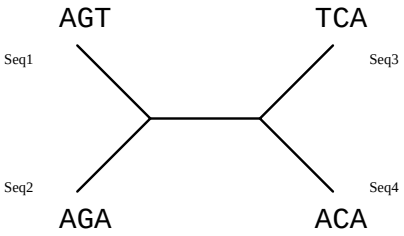
z.B. $n = 4$:

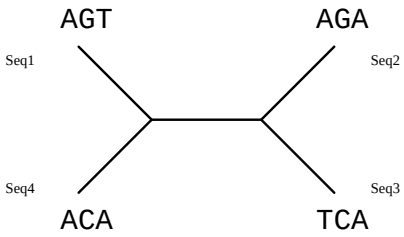
```
Seq1    GCAGGGTAC
Seq2    GCAGGGAAC
Seq3    GCTGGCAAC
Seq4    GCAGGCAAC
```

Gesucht: Stammbaum, der mit möglichst wenigen Substitutionen auskommt, mit anderen Worten: **Welcher Baum ist der parsimonischste?**

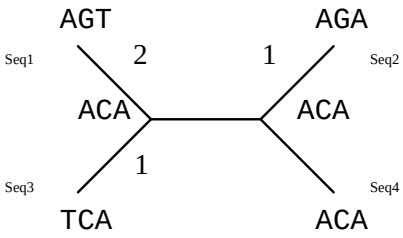
Wir können uns auf die segregierenden Sites beschränken, d. h. auf die Spalten, in denen mindestens zwei verschieden Basen (bzw. Aminosäuren) stehen.

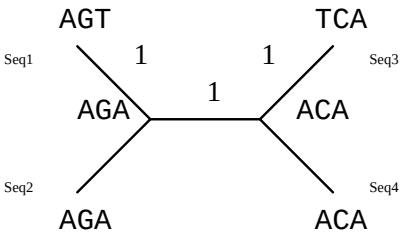


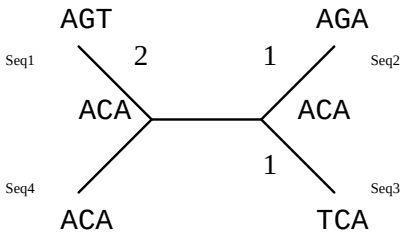




Welcher Baum ist der parsimonischste? **Der da ↓**



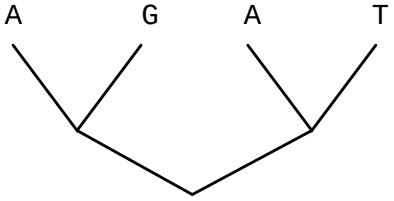


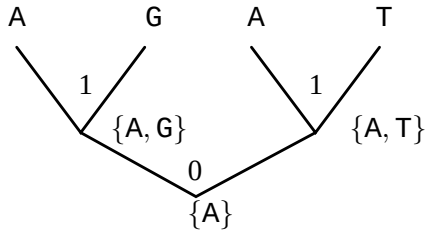


Gegeben: Baum (oBdA mit Wurzel), dessen Blätter mit Sequenzen beschriftet sind.
Gesucht: Mindestanzahl an nötigen Substitutionen

4.2.1 Algorithmus von Fitch (1971)

Idee: Beschrifte jeden Knoten k mit Menge M_k aller Beschriftungen, die optimal parsimonisch für den darüberliegenden Teilbaum wären. (dynamische Programmierung!)





Algorithmus von Fitch, formal:

Führe folgende Schritte für jede Position aus:

$C := 0$

Für jedes Blatt b mit Beschriftung x setze $M_b := \{x\}$

Gehe "von oben nach unten" alle Knoten k mit Kindern i, j durch und setze:

Falls $M_i \cap M_j = \emptyset$:

- $M_k := M_i \cup M_j$
- $C := C + 1$

Sonst: $M_k := M_i \cap M_j$

4.2.2 Gewichtete Parsimonie

Jede Substitution von $a \in \mathcal{A}$ nach $b \in \mathcal{A}$ kostet $S(a, b)$.

gegeben: ein gewurzelter Baum mit Sequenzen an den Blättern

gesucht: Gesamtkosten bei optimaler Belegung der inneren Knoten.

Strategie: Berechne mit Dynamischer Programmierung für jede Position und jeden Knoten k und jedes $a \in \mathcal{A}$ die Mindestkosten $S_k(a)$ für die Belegung des aus k herauswachsenden Teilbaums unter der Annahme, dass bei k ein a steht.

Für Blätter k, an denen an der entsprechenden Position ein a steht, ist $S_k(a) = 0$ und $S_k(b) = \infty$ für $b \neq a$. Für innere Knoten k mit Nachkommen i und j gilt:

$$S_k(a) = \min_{b \in \mathcal{A}}(S(a, b) + S_i(b)) + \min_{c \in \mathcal{A}}(S(a, c) + S_j(c))$$

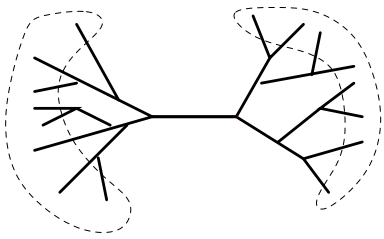
Dieser Wert kann mit dynamischer Programmierung ausgehend von den Blättern bis hin zur Wurzel r berechnet werden. Das Ergebnis ist dann $\min_{a \in \mathcal{A}} S_r(a)$.

4.2.3 Das Problem der perfekten Parsimonie

gegeben: n homologe Sequenzen der Länge m. An jeder Position kommen 1 oder 2 verschiedene Zustände vor.

gesucht: Gibt es einen perfekt parsimonischen Baum, d.h. einen in dem für jede Position höchstens eine Substitution vorkommt?

Idee: Jede Kante des Baumes entspricht genau einem "Split" der Menge der Blätter L in zwei Teilemengen A und B, das heißt $L = A \cup B$ und $A \cap B = \emptyset$. Positionen, an denen 2 verschiedene Zustände auftreten, definieren einen Split.



Also: Unterteile die Menge der Blätter nach und nach für jede segregierende Position weiter auf, bis ein Widerspruch auftritt – oder eben nicht.

Lemma: *Ein Widerspruch tritt genau dann auf, wenn zwei Splits mit $L = A \cup B = C \cup D$ existieren, so dass $A \cap C$, $A \cap D$, $B \cap C$ und $B \cap D$ alle $\neq \emptyset$ sind.*

Beweis: Wenn eine Kante L in A und B trennt, so kann offensichtlich jede weitere Kante nur A oder B weiter unterteilen. Deswegen gilt "$\Leftarrow$". Für die umgekehrte Richtung "$\Rightarrow$" überlege man sich, dass beim iterativen Einfügen von Kanten die Menge der Blätter nach und nach feiner partitioniert wird. Ein Widerspruch kann nur dann auftreten, wenn ein Split $L = A \cup B$ mehr als eine der in der aktuellen Partitionierung enthaltenen Mengen betrifft. Betrifft er zwei Mengen und kam eine der dazwischen liegenden Kanten von einer Partitionierung $L = C \cup D$, bedeutet das gerade, dass die vier Schnittmengen $A \cap C$, $A \cap D$, $B \cap C$ und $B \cap D$ alle $\neq \emptyset$ sind.

□

4.2.4 Das verallgemeinerte Problem der perfekten Parsimonie

gegeben: n homologe Sequenzen der Länge m. An jeder Position kommen bis zu r verschiedene Zustände vor.

gesucht: Gibt es einen perfekt parsimonischen Baum, d.h. einen, in dem an jeder Position weder Rückmutationen noch zwei Substitutionen zum selben Symbol vorkommen?

Komplexität: Falls r groß werden darf, ist dieses Problem NP-vollständig.

Aber: Für jedes feste $r \in \mathbb{N}$ existieren Polynomialzeitalgorithmen.

4.2.5 Das Problem der maximalen Parsimonie

gegeben: n homologe Sequenzen der Länge m. An jeder Position kommen 1 oder 2 verschiedene Zustände vor.

gesucht: ein Baum, der die minimale Anzahl an Substitutionen braucht

Komplexität: NP-vollständig

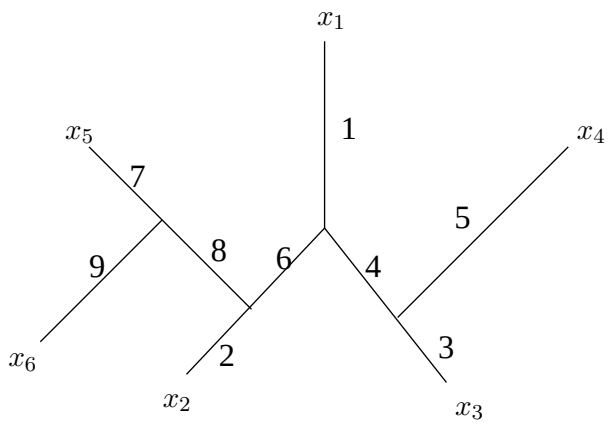
Aber: Immerhin gibt es einen Algorithmus, der einen Baum liefert, der mit weniger als dem Doppelten der Mindestanzahl auskommt. Für die Praxis relevanter sind jedoch heuristische Lösungsverfahren.

Exakte Lösung für wenige Sequenzen: Branch & Bound: Wir überlegen uns zunächst, wie wir alle Bäume aufzählen können. Dazu kodieren wir die ungewurzelten Bäume mit $n \geq 3$ Blättern $x_1, \ldots, x_n$

durch $[i_3][i_5][i_7]\ldots[i_{2n-5}]$ mit $i_k \in \{1,\ldots,k\}$. Einen solchen Ausdruck übersetzt man folgendermaßen in einen Baum:

- Starte mit dem dreiblättrigen Baum, bei dem die Blätter mit x_1, x_2, x_3 und die dazugehörigen Kanten mit 1,2,3 beschriftet sind.
- Wiederhole für $j = 4,\ldots,n$:
  - $k := 2j - 5$
  - Füge Kante mit Blatt x_j an Kante i_k ein.
  - Nenne den neuen Teil der Kante i_k, der näher an x_1 liegt $k+1$ und die neue Kante $k+2$.

Beispiel: $[3][2][7]$ steht für:

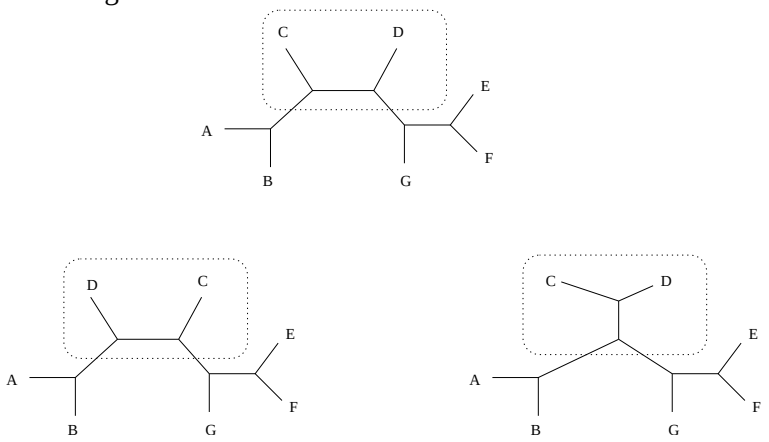


Offensichtlich läßt sich umgekehrt jeder Baum mit Blattbeschriftungen $x_1,\ldots,x_n$ als Zahlenfolge schreiben. Mit Hilfe dieser Kodierung können wir nun die Bäume aufzählen, indem wir $[i_3][i_5][i_7]\ldots[2n-5]$ im Stile eines Kilometerzählers alle erlaubten Werte durchlaufen lassen. Wir starten dabei mit $[1][0][0]\ldots[0]$. Nullen am Ende bedeuten dabei, dass die entsprechenden Taxa noch nicht in den Baum eingefügt wurden. "Zählerstände", bei denen Nullen links von Nicht-Nullen stehen, werden übersprungen. Für jeden "Zählerstand" bzw. Baum berechnen wir den Parsimonie-Score. Wir merken uns den Score C des bisher parsimonischsten Baums. Diese Vorgehensweise gestattet uns nicht nur, den Raum der Bäume halbwegs übersichtlich darzustellen, sondern legt auch eine Möglichkeit nahe, wie wir die Suche beträchtlich abkürzen können: Wenn der Score zu $[i_1]\ldots[i_m][0]\ldots[0]$ bereits größer als C ist, brauchen wir die Möglichkeiten, die übrigen Taxa einzufügen (d.h. die Nullen durch Nicht-Nullen zu ersetzen) nicht durchzugehen und können statt dessen gleich i_m um 1 erhöhen.

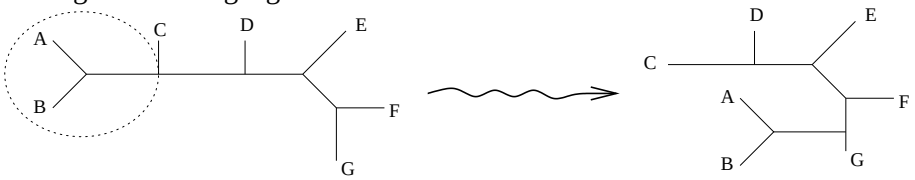
heuristische Methoden Man beginnt mit irgendeinem Baum und versucht, ihn durch kleine Veränderungen (etwa das sogenannte **branch swapping**) nach und nach zu verbessern. Die Hoffnung ist, somit den optimalen Baum zu finden. Das wesentliche Problem ist aber die Gefahr in lokalen Optima hängen zu bleiben. Die Verfahren unterscheiden sich in den Veränderungen, die für den jeweils aktuellen Baum

in Betracht gezogen werden. Um einen ersten Baum zu bekommen, bietet sich **stepwise addition** an, das heißt man beginnt mit 3 Taxa und fügt nach und nach alle weiteren Taxa hinzu, indem eine Kante durch einen neuen Knoten unterteilt und an den neuen Knoten eine neue Kante angefügt wird, die zu dem neuen Blatt führt. Bezüglich der Eingabe-Reihenfolge der Taxa gibt es verschiedene Varianten, z. B. kann man jeweils das Taxon als nächstes nehmen, das die größte Ähnlichkeit zu den bisherigen hat.

Für das branch swapping gibt es wiederum mehrere Möglichkeiten. Eine ist **nearest neighbour interchange**: Dabei wird ein Paar benachbarter Blätter zufällig ausgewählt und probiert, ob man zu einem besseren Score kommt, indem man ihre Verwandtschaft zum Rest des Baumes ändert. Dazu hat man jeweils drei Möglichkeiten, wie folgendes Beispiel verdeutlichen soll. Werden bei einem der folgenden Bäume die Blätter C und D ausgewählt, so wird im nächsten der Baum von den dreien weiterverwendet, der den höchsten Score bringt.



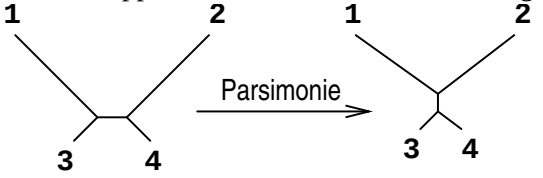
Eine andere Möglichkeit ist **subtree pruning & regrafting**. Dabei wird ein Teilbaum zufällig ausgewählt und es wird überprüft, ob man den Score verbessern kann, indem man den Teilbaum an einen neuen Knoten an einer anderen Kante des restlichen Teilbaums ansetzt. Folgendes Bild zeigt ein Beispiel für einen somit möglichen Übergang:



Diese und weitere Heuristiken stehen etwa in dem Programmpaket PAUP von David Swofford (2003) zur Verfügung.

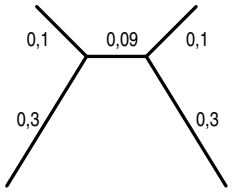
4.2.6 Grenzen des Parsimonie-Prinzips

Problematisch wird's, wenn Äste so lang sind, dass Rück- und Doppelmutationen aufreten. Die Länge solcher Äste wird systematisch unterschätzt.



Besser wäre also ein Verfahren, das die Möglichkeit von Rück- und Doppelmutationen berücksichtigt.

Vergleich von Verfahren zur Phylogenieschätzung Durbin et al. (1998) haben für mehrere Sequenzlängen je 1000 Quartette von $\{A, B\}$-Sequenzen längs dieses Baumes simuliert und damit die Verfahren verglichen. Die Astlängen sind mittlere Mutationshäufigkeiten pro Position.



Anteil richtig geschätzter Bäume:

Sequenzlänge	Maximale Parsimonie	Neighbour Joining	Maximum Likelihood
20	39.6%	47.7%	41.9%
100	40.5%	63.5%	63.8%
500	40.4%	89.6%	90.4%
2000	35.3%	99.5%	99.7%

Wir sehen also, dass bei diesem Baum, der sehr lange Kanten enthält, das parsimonische Verfahren auch bei sehr langen Sequenzen schlechte Ergebnisse liefert. Neighbour Joining liefert bessere Ergebnisse obwohl es nur die Distanzen zwischen den Sequenzen verwendet. Noch bessere Ergebnisse liefert bei langen Sequenzen allerdings das Maximum-Likelihood-Verfahren (ML), um das es im nächsten Abschnitt geht.

4.3 Das ML-Prinzip in der Phylogenieschätzung

Modell: Die Daten D sind Ergebnis eines zufälligen Prozesses. Ihre Wahrscheinlichkeit $\Pr_\theta(D)$ hängt vom unbekannten Parameter θ ab.

Gesucht: θ soll geschätzt werden!

ML-Prinzip: Der Schätzer $\widehat{\theta}$ ist der Wert, für den die Likelihood-Funktion

$$L_D(\theta) := \Pr_\theta(D)$$

maximal wird.

$D =$ Sequenzdaten (n homologe Sequenzen der Länge m)

$\theta =$ ein Baum

ML-Prinzip: $\widehat{\theta}$ ist der Baum, der $L_D(.)$ maximiert (der also die Sequenzdaten am wahrscheinlichsten macht).

Damit $L_D(\theta) := \Pr_\theta(D)$ definiert ist, benötigt man ein stochastisches Modell für den Substitutionsprozess, der längs der Kanten eines Baumes θ abläuft.

Spezifiziere also die Wahrscheinlichkeit

$$P_{x \to y}(t),$$

dass an einer Position, an der ein x steht, nach Zeit t ein y steht.

Dazu brauchen wir ein Modell für die Sequenz-Evolution. Wir verwenden hier zunächst die DNA-Variante eines sehr einfaches Modell von Jukes und Cantor (1969).

Dabei sollten wir wie immer, wenn es ans Modellieren geht ein Bonmot von G.E.P. Box beherzigen: **All models are wrong — but some are useful.**

4.3.1 DNA-Substitutionsmodell von Jukes & Cantor (1969)

Basenhäufigkeiten in Ursequenz: $(\pi(\mathtt{A}), \pi(\mathtt{C}), \pi(\mathtt{G}), \pi(\mathtt{T})) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$

Alle Positionen mutieren unabhängig voneinander mit Rate 1, d.h. die Wartezeit T einer Position auf die nächste Mutation ist exponential-1-verteilt, also:

$$\Pr(T > t) = e^{-t}$$

Wenn eine Mutation geschieht, wird die Base durch eine rein zufällige aus $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ ersetzt (bleibt also evtl. unverändert). Es folgt:

$$P_{x \to y}(t) = \begin{cases} \left(1 - e^{-t}\right) \cdot \frac{1}{4} & \text{falls } x \neq y \\ e^{-t} + \left(1 - e^{-t}\right) \cdot \frac{1}{4} & \text{falls } x = y \end{cases}$$

Jukes-Cantor ist sehr einfach! Allgemeinere Modelle für DNA-Substitutionen (z.B. von Hasegawa, Kishino & Yano, 1985) berücksichtigen die ungleichen Basenhäufigkeiten und die Tatsache, dass Transitionen eine höhere Rate haben als Transversionen. Wir werden uns später mit solchen Modellen befassen. Auch z.B. die Positionsabhängigkeit der Mutationsraten kann berücksichtigt werden (Tamura & Nei, 1993). Substitutionsmodelle für Proteine sind implizit durch Score-Matrizen wie PAM und BLOSUM gegeben.

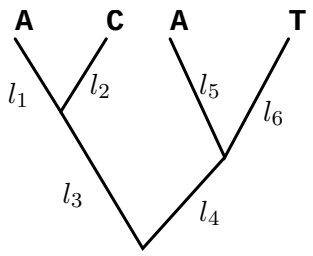
4.3.2 Berechnung der Likelihood eines Baums

Gegeben: (gewurzelter) Baum θ mit Astlängen $(l_i)_i$, Sequenzdaten D.

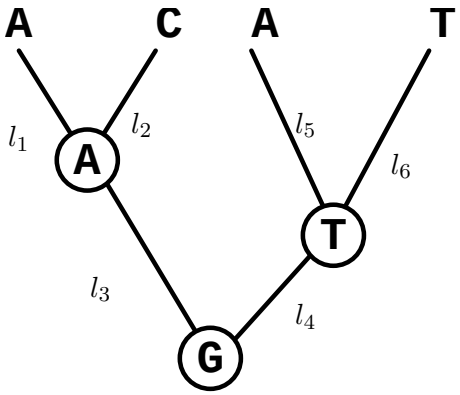
Gesucht: Die Likelihood $L_D(\theta) = \Pr_\theta(D)$ des Baumes für die gegeben Daten.

erste Idee: Berechne Likelihood positionsweise und multipliziere.

oBdA sei also die Sequenzlänge $m = 1$.



Angenommen D kennt auch die inneren Knoten:



Dann ist es einfach:

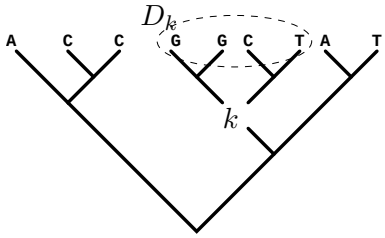
$$L_D(\theta) = \pi(\mathsf{G}) \cdot P_{\mathsf{G}\to\mathsf{A}}(l_3) \cdot P_{\mathsf{G}\to\mathsf{T}}(l_4) \cdot P_{\mathsf{A}\to\mathsf{A}}(l_1) \cdot P_{\mathsf{A}\to\mathsf{C}}(l_2) \cdot P_{\mathsf{T}\to\mathsf{A}}(l_5) \cdot P_{\mathsf{T}\to\mathsf{T}}(l_6)$$

Was tun, wenn die inneren Knoten nicht bekannt sind?

Felsensteins Pruning-Algorithmus (1981)

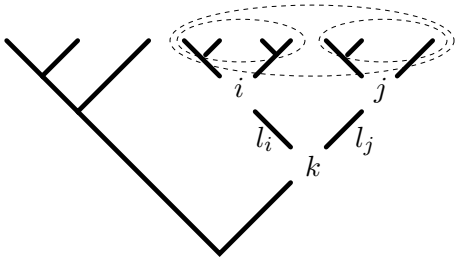
Wiedermal dynamische Programmierung!

D_k sei der Datensatz an den Blättern über dem Knoten k.



Für Base x sei $W_k(x)$ die Wahrscheinlichkeit von D_k unter der Bedingung, dass bei k ein x steht.

Ist b ein Blatt mit Base y, so ist $W_b(y) = 1$ und $W_b(x) = 0$ für $x \neq y$. Man rechne dann der Idee der dynamischen Programmierung folgend von den Blättern bis zur Wurzel alle Werte $W_k(x)$ aus, indem man folgende Rekursion anwendet:



Ist k ein Knoten mit Kindern i und j und Astlängen l_i und l_j, so gilt:

$$W_k(x) = \left(\sum_{y \in \{\mathsf{A},\mathsf{C},\mathsf{G},\mathsf{T}\}} P_{x\to y}(l_i) \cdot W_i(y) \right) \cdot \left(\sum_{z \in \{\mathsf{A},\mathsf{C},\mathsf{G},\mathsf{T}\}} P_{x\to z}(l_j) \cdot W_j(z) \right)$$

Da für die Wurzel r die Gleichung $D_r = D$ gilt, erhalten wir das Ergebnis:

$$L_D(\theta) = \sum_{x \in \{\mathtt{A,C,G,T}\}} \pi(x) \cdot W_r(x)$$

4.3.3 Den ML-Baum finden

Das ist das schwierige Problem:

Gegeben n Sequenzen. Finde den Baum mit der höchsten Likelihood.

Dabei sind zwei Teilprobleme zu unterscheiden:

(a) Gegeben eine Baumtopologie, also ein Baum ohne Angabe von Kantenlängen bzw. Zeitabstände, welche Evolutionsraten und Astlängen maximieren die Likelihood?

(b) Suche die Topologie, für die (a) zu einer möglichst hohen Likelihood führt.

zu (a): Felsenstein (1981) wendet einen EM-Algorithmus zur Optimierung der Astlängen an. Es werden also abwechselnd alle Kantenlängen und Mutationsraten aus den erwarteten Anzahlen an Substitutionen der einzelnen Basen- bzw. Aminosäuretypen geschätzt und erwartete Substitutionshäufigkeiten bezüglich der jeweils aktuellen Schätzwerte neu berechnet.

Alternativ dazu kann man auch die mehrdimensionale Newton-Methode (siehe Anhang B) zur Optimierung der Astlängen anwenden. Dazu muss man die Ableitung der Likelihood nach den Astlängen berechnen. Das kann man mit einer Variante des Felsenstein-Prunings effizient machen, da die dort verwendeten Rekursionen linear sind. (Die Details überlege man sich!)

zu (b): Felsenstein (1981) beginnt mit einem dreiblättrigen Baum für die ersten drei Taxa und fügt nach und nach alle weiteren Taxa hinzu. Beim Einfügen des k-ten Blattes probiert man alle $2k - 5$ Kanten als mögliche Einfügestelle für die neue Blatt-Kante aus und entscheidet sich für diejenige, für die der entstehende Baum die höchste Likelihood hat. Wenn alle Kanten eingefügt sind, wird versucht, die Likelihood des Baums durch lokale Veränderungen zu erhöhen. Außerdem werden mehrere zufällige Eingabe-Reihenfolgen für die Taxa durchprobiert.

einige weitere Lösungsansätze:

- Falls n klein ist: Probiere alle Baumtopologien durch und variiere die Astlängen.
- Gehe z.B. von NeighbourJoining-Baum aus und versuche, durch leichtes Verändern des Baums nach und nach die Likelihood zu erhöhen, ähnlich wie das bei der Suche nach dem Parsimonischsten Baum gemacht wird.
- Nimm jeweils 4 Sequenzen und suche deren ML-Baum. Baue am Ende die Viererbäume zusammen. (“PUZZLE”; Strimmer, von Haeseler, 1996, `http://www.tree-puzzle.de/`)

Ein wichtiges Argument für den Maximum-Likelihood-Schätzer ist dessen **Konsistenz**, das heißt im Grenzfall unendlich langer Sequenzen wäre der ML-Baum immer der wahre Baum, siehe Abschnitt 4.3.4. Das Problem ist allerdings, dass der ML-Baum nicht so leicht zu berechnen ist.

4.3.4 Konsistenz des ML-Baums

Der Maximum-Likelihood-Schätzer für den Stammbaum gegebener Sequenzen ist *konsistent*, d.h. zumindest wenn die Modellannahmen stimmen, dann wird die Wahrscheinlichkeit, dass die ML-Methode den richtigen Baum liefert, gegen 1 konvergieren, wenn die Sequenzlänge gegen ∞ geht. Zu den Modellannahmen gehört, dass die Sequenzen korrekt aligniert sind und dass alle Positionen unabhängig voneinander nach den in der Analyse angenommenen Substitutionsraten ohne Selektion evoluiert sind.

Unter diesen starken Annahmen mag als selbstverständlich erscheinen, dass der wahre Baum gefunden wird. Man beachte aber, dass z.B. der maximal-parsimonische Baum im Gegensatz zum ML-Baum nicht konsistent ist.

Beweisskizze zur Konsistenz des ML-Schätzers: Wir gehen der Einfachheit halber davon aus, dass Insertionen und Deletionen keine Rolle spielen, so dass das Alignment der Sequenzen keine Gaps enthält. Seien $x_1, \ldots, x_m$ die verschiedenen Typen von Spalten, die in dem Alignment vorkommen können, und seien $n_1, \ldots, n_m$ die Anzahlen der jeweiligen Spaltentypen in den vorliegenden Daten D. Die Likelihood eines Baumes T ist dann

$$L(T) = \Pr(D \mid T) = \prod_{i=1}^{m} \Pr(x_i \mid T)^{n_i}$$

Wie so oft bevorzugen wir die logartihmische Darstellung:

$$\ln L(T) = \sum_{i=1}^{m} n_i \ln \Pr(x_i \mid T)$$

Die Grundidee des Beweises ist, dass die Wahrscheinlichkeiten $\Pr(x_i \mid T)$ für den Baum charakteristisch sind und sich in den relativen Häufigkeiten $R_1, \ldots, R_m$ der Spaltentypen $x_1, \ldots, x_m$ in den Daten widerspiegeln. Indem wir obige Gleichung durch die Sequenzlänge n teilen, erhalten wir

$$\frac{1}{n} \ln L(T) = \sum_{i=1}^{m} R_i \ln \Pr(x_i \mid T)$$

Da der Logarithmus monoton ist, wird $L(T)$ für den selben Baum maximal wie $\frac{1}{n} \ln L(T)$. Seien $p_1, \ldots, p_m$ die für den wahren Baum geltenden Wahrscheinlichkeiten der Spaltentypen. Wenn die Sequenzen hinreichend lang sind, gilt mit beliebig hoher Wahrscheinlichkeit $R_i = p_i \pm \varepsilon$ für beliebig kleines ε. Mit $\Pr(x_i \mid T) =: q_i$ gilt also

$$\frac{1}{n} \ln L(T) \stackrel{n \to \infty}{\longrightarrow} \sum_{i=1}^{m} p_i \ln q_i$$

Für den wahren Baum T^* gilt $\frac{1}{n} \ln L(T) \stackrel{n\to\infty}{\longrightarrow} \sum_{i=1}^{m} p_i \ln p_i$. Für $p \neq p$ gilt

$$\sum_{i=1}^{m} p_i \ln q_i < \sum_{i=1}^{m} p_i \ln p_i.$$

Das folgt nämlich mit

$$\sum_{i=1}^{m} p_i \ln p_i - \sum_{i=1}^{m} p_i \ln q_i = \sum_{i=1}^{m} p_i \ln \frac{p_i}{q_i} > 0,$$

daraus, dass die relative Entropie ungleicher Verteilungen positiv ist, siehe Anhang A.5.

Also gilt für hinreichend lange Sequenzen mit beliebig hoher Wahrscheinlichkeit $L(T^*) > L(T)$ für jeden Baum T, dessen Spaltentypenwahrscheinlichkeiten sich von denen des wahren Baumes unterscheiden. Es bleibt noch zu zeigen, dass ein falscher Baum nicht die selben Spaltentypenwahrscheinlichkeiten haben kann wie der richtige. Das letzteres gelten muss, kann man sich aber mit einem Umweg über das Neighbor-Joining-Theorem klar machen, siehe Abschnitt 4.3.6.

4.3.5 Maximum Parsimony aus probabilistischer Sicht

Wenn wir von einem probabilistischen Modell ausgehen und $s(a, b) = -\log P_{a\to b}(1)$ setzen, so können wir die Werte $s(a, b)$ als Scores für gewichtete Parsimonie benutzen. Dann ist der Score eines Baumes eine Approximation für die Likelihood des Baums für den Fall, dass alle Kanten die Länge 1 haben. Es ist selbst für diesen recht unplausibel erscheinenden Fall nur eine Approximation, da nur die wahrscheinlichste Möglichkeit der Belegung der inneren Kanten berücksichtigt wird. (Ähnlich wie bei Viterbi nur der wahrscheinlichste Pfad berücksichtigt wird statt einer gewichteten Summe aller Pfade.) Wie wir zuvor gesehen haben, kann Parsimonie daher in die Irre laufen, wenn es sehr lange Kanten im wahren Baum gibt. Ist dies nicht der Fall, kann Parsimonie recht gute Ergebnisse liefern, und zwar vergleichsweise schnell, da nicht über die Kantenlängen maximiert wird.

4.3.6 Maximum Likelihood und Pairwise Distances

Gegeben ein reversibles Sequenzevolutionsmodell mit bekannten Parametern ist die ML-Distanz d_{xy}^{ML} zwischen Sequenz $x = (x_1, x_2, \ldots, x_n)$ und Sequenz $y = (y_1, \ldots, y_n)$:

$$\begin{aligned} d_{xy}^{\mathrm{ML}} &= \arg\max_t \{\Pi_i \pi_{x_i} \cdot P_{x_i\to y_i}(t)\} \\ &= \arg\max_t \{\Pi_i P_{x_i\to y_i}(t)\} \end{aligned}$$

Zum Beispiel im Jukes-Cantor-Modell erhalten wir im Falle von k Mismatches:

$$\Pi_i P_{x_i\to y_i}(t) = \left(\frac{1}{4}(1 - e^{-t\alpha})\right)^k \left(\frac{1}{4}(1 + 3e^{-t\alpha})\right)^{n-k}$$

Indem wir davon mit den üblichen Mittel das Extremum suchen, erhalten wir:

$$d_{xy}^{\mathrm{ML}} = -\frac{1}{4\alpha} \cdot \ln\left(1 - \frac{4k}{3n}\right)$$

Der ML-Schätzer ist konsistent, d.h. im Limes sehr langer Sequenzen konvergiert d^{ML} gegen die wahre Distanz. Dann kann mit Neighbour Joining der wahre Baum gefunden werden. Im Limes sehr langer Sequenzen konvergiert auch der ML-Baum gegen den wahren Baum. Wenn es von der Rechenzeit her möglich ist, sollte man den ML-Baum bevorzugen, da dieser für weniger lange Sequenzen zuverlässiger ist.

4.4 Modelle der DNA-Sequenzevolution in kontinuierlicher Zeit

Sei $P_{a\to b}(t)$ die Wahrscheinlichkeit, dass ein $a \in \mathcal{A}$ nach Zeit (bzw. Kantenlänge) t zu einem b geworden ist und sei

$$S(t) := \begin{pmatrix} P_{A\to A}(t) & P_{A\to C}(t) & P_{A\to G}(t) & P_{A\to T}(t) \\ P_{C\to A}(t) & P_{C\to C}(t) & P_{C\to G}(t) & P_{C\to T}(t) \\ P_{G\to A}(t) & P_{G\to C}(t) & P_{G\to G}(t) & P_{G\to T}(t) \\ P_{T\to A}(t) & P_{T\to C}(t) & P_{T\to G}(t) & P_{T\to T}(t) \end{pmatrix}$$

Die Zeilen summieren sich zu 1 auf.

Für sehr kleine $\varepsilon > 0$ ist $P_{x\to x}(\varepsilon)$ nahe bei 1 und es gibt eine Matrix R, die sogenannte **Ratenmatrix** (manchmal auch Q-Matrix genannt), so dass gilt $S(\varepsilon) \approx (I + R \cdot \varepsilon)$, wobei I die Einheitsmatrix bezeichnet:

$$I = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Damit ergibt sich $S(t+\varepsilon) = S(t) \cdot S(\varepsilon) \approx S(t)(I + R\varepsilon) = S(t) + S(t)R\varepsilon$ und damit

$$\lim_{\varepsilon\to 0} \frac{S(t+\varepsilon) - S(t)}{\varepsilon} = S(t)R$$

$S(t)R$ ist also so etwas wie die Ableitung des Prozesses. Man beachte, dass die Zeilensummen 0 sind. In der Diagonalen der Matrix stehen die mit negativem Vorzeichen versehenen Erwartungswerte für die Zeit, die man im jeweiligen Zustand bleibt. Die übrigen Einträge sind die Mutationsraten für die jeweiligen Substitutionen.

Das Jukes-Cantor-Modell für DNA-Sequenzen hat die Ratenmatrix

$$\begin{pmatrix} -\frac{3}{4}\alpha & \frac{1}{4}\alpha & \frac{1}{4}\alpha & \frac{1}{4}\alpha \\ \frac{1}{4}\alpha & -\frac{3}{4}\alpha & \frac{1}{4}\alpha & \frac{1}{4}\alpha \\ \frac{1}{4}\alpha & \frac{1}{4}\alpha & -\frac{3}{4}\alpha & \frac{1}{4}\alpha \\ \frac{1}{4}\alpha & \frac{1}{4}\alpha & \frac{1}{4}\alpha & -\frac{3}{4}\alpha \end{pmatrix}.$$

Das Modell F81 von Felsenstein (1981) berücksichtigt die möglicherweise ungleichen Basenhäufigkeiten

$(\pi_A, \pi_C, \pi_G, \pi_T)$. Die entsprechende Ratenmatrix ist:

$$\begin{pmatrix} -\alpha + \alpha\pi_A & \alpha\pi_C & \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & -\alpha + \alpha\pi_C & \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & \alpha\pi_C & -\alpha + \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & \alpha\pi_C & \alpha\pi_G & -\alpha + \alpha\pi_T \end{pmatrix}.$$

Das HKY-Modell (Hasegawa, Kishino, Yano, 1985) berücksichtigt zusätzlich, dass Transitionen im Allgemeinen häufiger sind als Transversionen durch Verwendung eines zusätzlichen Parameters β. Die Ratenmatrix ist

$$Q := \begin{pmatrix} -\alpha\pi_G - \beta(\pi_C + \pi_T) & \beta\pi_C & \alpha\pi_G & \beta\pi_T \\ \beta\pi_A & -\alpha\pi_T - \beta(\pi_A + \pi_G) & \beta\pi_G & \alpha\pi_T \\ \alpha\pi_A & \beta\pi_C & -\alpha\pi_A - \beta(\pi_C + \pi_T) & \beta\pi_T \\ \beta\pi_A & \alpha\pi_C & \beta\pi_G & -\alpha\pi_C - \beta(\pi_A + \pi_G) \end{pmatrix}.$$

$(\pi_A, \pi_C, \pi_G, \pi_T)$ ist die **stationäre Verteilung** (oder auch **Gleichgewichtsverteilung**) für diese Ratenmatrix, das heißt es gilt:

$$(\pi_A, \pi_C, \pi_G, \pi_T) \cdot Q = (0, 0, 0, 0)$$

Daraus folgt insbesondere: Falls $(\pi_A, \pi_C, \pi_G, \pi_T)$ die Basenhäufigkeiten der Ursequenz sind, bleiben diese Häufigkeiten in Erwartung erhalten, wenn die Sequenz gemäß HKY evolviert. Es gilt also:

$$(\pi_A, \pi_C, \pi_G, \pi_T) \cdot S(t) = (\pi_A, \pi_C, \pi_G, \pi_T)$$

Diese Stationarität gilt ebenso für das F81-Modell.

4.4.1 Verweildauer in einem Zustand

Wie lange bleibt man bei einem durch eine Ratenmatrix beschriebenen Markoff'schen Evolutionsmodell in einem Zustand, bevor man in einen anderen mutiert? Wir betrachten zunächst den Fall mit diskreten Zeitschritten. Angenommen man ist in einem Zustand, von dem man innerhalb einer Generation mit Wahrscheinlichkeit p wegmutiert. Dann ist die Wahrscheinlichkeit, in der k-ten Generation den Zustand erstmalig zu verlassen, offensichtlich $(1-p)^{k-1} \cdot p$.

Allgemein heißt eine Zufallsvariable X mit Werten in $\{1, 2, \ldots\}$ **geometrisch verteilt** wenn, gilt $\mathrm{Ws}(X = k) = (1-p)^{k-1} \cdot p$.

Es gilt dann

$$\mathbb{E}X = \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} \cdot p = \frac{1}{p}$$

Das kann man leicht nachrechnen (falls man glaubt, dass $\mathbb{E}X$ existiert und endlich ist):

$$\mathbb{E}X \quad = \quad \sum_{k=0}^{\infty} (k+1) \cdot (1-p)^k \cdot p$$

$$
\begin{aligned}
&= \sum_{k=1}^{\infty} k \cdot (1-p)^k \cdot p + \sum_{k=0}^{\infty} \cdot (1-p)^k \cdot p \\
&= (1-p) \cdot \mathbb{E}X + p \cdot \frac{1}{p} \\
\Rightarrow \qquad & \mathbb{E}X = \frac{1}{p}
\end{aligned}
$$

Die geometrische Verteilung ist im folgenden Sinne gedächtnislos:

$$\mathrm{Ws}(X = k + n \mid X > k) = \mathrm{Ws}(X = n)$$

Diese Eigenschaft charakterisiert die geometrische Verteilung unter allen Wahrscheinlichkeitsverteilungen auf den natürlichen Zahlen. (Man kann die geometrische Verteilung allerdings auch auf $\{0, 1, 2, \ldots\}$ definieren; das findet man auch oft in der Literatur.)

Das kontinuierliche Analogon zur geometrischen Verteilung ist die **Exponentialverteilung:** Eine Zufallsvariable Y heißt **exponentialverteilt** auf $[0, \infty)$, falls gilt:

$$\mathrm{Ws}(Y > z) = e^{-\lambda z}$$

Das heißt:

$$\mathrm{Ws}(Y \in [a, b]) = e^{-\lambda b} - e^{-\lambda a}$$

Es gilt dann:

$$\mathbb{E}Y = \int_0^{\infty} z \lambda e^{-\lambda z} = \frac{1}{\lambda}$$

Für kleine p und große k gilt

$$(1-p)^k \approx e^{-pk}.$$

Die Exponentialverteilung kann also zur Approximation der geometrischen Veteilung verwendet werden. Auch die Exponentialverteilung ist durch ihre Gedächtnislosigkeit charakterisiert:

$$\mathrm{Ws}(Y > x + z | Y > x) = \mathrm{Ws}(Y > z)$$

Bei Substitutionsmodellen mit kontinuierlicher Zeit bleibt man exponentiell lange in einem Zustand. Zum Beispiel bei HKY bleibt man in A exponentiell lange mit Erwartungswert $1/(\alpha\pi_G + \beta(\pi_C + \pi_T))$ und entscheidet sich beim Wegsprung mit Wahrscheinlichkeiten $\beta\pi_C \cdot c$, $\alpha\pi_G \cdot c$ und $\beta\pi_T \cdot c$ (mit $c = 1/(\alpha\pi_G + \beta(\pi_C + \pi_T))$) für C,G, bzw. T.

4.4.2 Berechnung von $S(t)$ aus R

Wie man aus der Ratenmatrix die Übergangsmatrix für eine feste Zeit berechnet, überlegen wir uns zunächst wieder fürs zeitdiskrete Analogon: $S(n)$ berechnet man, wie wir bereits im Zusammenhang mit PAM-Matrizen gesehen hatten, für $n \in S(n)$ aus $S(1)$ durch Potenzieren:

$$S(n) = S(1)^n$$

Das macht man effizient, indem man $S(1)$ zunächst diagonalisiert, das heißt man sucht eine Matrix U und eine Diagonalmatrix D, so dass gilt $S(1) = U \cdot D \cdot U^{-1}$. In D stehen in der Diagonalen die Eigenwerte von $S(1)$ (und sonst nur Nullen), d.h. die Werte μ, so dass ein Vektor v mit $Dv = \mu v$ existiert. Ein solcher Vektor heißt Eigenvektor. D beschreibt geometrisch gesehen dieselbe Abbildung wie $S(1)$, aber ausgehend von einer Basis des Vektorraums, die aus Eigenvektoren der Abbildung besteht. (Das sollte bei Bedarf mit Hilfe eines Lehrbuchs über Lineare Algebra nachgearbeitet werden!) Es gilt dann $S(n) = UDU^{-1} \cdot UDU^{-1} \cdot UDU^{-1} \cdots UDU^{-1} = UD^nU^{-1}$. Diagonalmatrizen sind leicht zu potenzieren. Für

$$D = \begin{pmatrix} \mu_1 & 0 & \dots & 0 \\ 0 & \mu_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mu_m \end{pmatrix}$$

gilt:

$$D^n = \begin{pmatrix} \mu_1^n & 0 & \dots & 0 \\ 0 & \mu_2^n & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mu_m^n \end{pmatrix}.$$

Im kontinuierlichen Fall verhält es sich ähnlich. Für $t \in [0, \infty)$ gilt $S(t) = U \cdot T^t \cdot U^{-1}$ mit

$$T^t = \begin{pmatrix} e^{\lambda_1 t} & 0 & \dots & 0 \\ 0 & e^{\lambda_2 t} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & e^{\lambda_m t} \end{pmatrix},$$

wobei $\lambda_1, \lambda_2, \dots, \lambda_m$ die Eigenwerte von R sind (mit $m = 4$ bei DNA und $m = 20$ bei Proteinen). Das kann man sich folgendermaßen klar machen: Für sehr kleine $\varepsilon > 0$ gilt

$$S(t) = S(\varepsilon)^{t/\varepsilon} \approx (I + R \cdot \varepsilon)^{t/\varepsilon} = U_\varepsilon \cdot D_\varepsilon^{t/\varepsilon} \cdot U_\varepsilon^{-1}.$$

Dabei ist D_ε eine Diagonalmatrix, die als Diagonaleinträge die Eigenwerte μ_i von $I + \varepsilon \cdot R$ enthält, d.h. für die zugehörigen rechten Eigenvektoren v_i gilt $(I + \varepsilon \cdot R) \cdot v_i = \mu_i \cdot v_i$, und damit $R \cdot v_i = \frac{\mu_i - 1}{\varepsilon} \cdot v_i$. Also ist $\lambda_i := \frac{\mu_i - 1}{\varepsilon}$ ein Eigenwert von R (sofern $\mu_i \neq 1$). Wir können die Diagonaleinträge von $D_\varepsilon^{t/\varepsilon}$ also als $(\varepsilon\lambda_i + 1)^{t/\varepsilon}$ schreiben, und das konvergiert gegen $e^{\lambda_i t}$ für $\varepsilon \to 0$, denn bekanntlich geht $(1 + \varepsilon)^{1/\varepsilon}$ für $\varepsilon \to 0$ gegen e.

Mit einer ähnlichen Argumentation überzeugt man sich davon, dass die Spalten der in der Formel $S(t) = UT^tU^{-1}$ auftretenden Matrix U die rechten Eigenvektoren der Ratenmatrix R sind: Wir erhalten U als Limes für $\varepsilon \to 0$ der Matrix U_ε aus der Gleichung $(I + R \cdot \varepsilon)^{t/\varepsilon} = U_\varepsilon \cdot D_\varepsilon^{t/\varepsilon} \cdot U_\varepsilon^{-1}$. Die Spalten von U_ε sind Eigenvektoren v_i der Matrix $I + \varepsilon R$. Wie wir im vorherigen Absatz nebenbei bemerken konnten, sind die v_i damit auch Eigenvektoren von R und daran ändert sich nichts, wenn ε gegen 0 geht (wenn wir von $\mu_i \neq 1$ ausgehen).

Der Fall, dass ein Eigenwert von $I + \varepsilon R$ für $\varepsilon \to 0$ gegen 1 konvergiert, korrespondiert mit einer stationären Verteilung des Prozesses, die dann linker Eigevektor zum Eigenwert 1 ist. Der dazugehörige rechte Eigenvektor tritt dann ebenfalls in U auf und der dazugehörige Diagonaleintrag in T^t ist $e^0 = 1$.

4.4.3 Ein Modell, in dem man ohne Eigenvektoren rechnen kann

Das F84-Modell (Felsenstein, 1984) ist dem HKY-Modell sehr ähnlich. Man kann es aber gut auch ohne Ratenmatrix beschreiben. Jede Position der DNA-Sequenz hat eine Abstammungslinie, in die mit Rate λ "Kreuzchen" und mit Rate μ "Kringel" eingestreut werden. Beide bewirken, dass die Position ihren Basentyp "vergisst". Bei den Kringeln bleibt aber die Information erhalten, ob es sich um ein Purin oder ein Pyrimidin handelte. Bei Kreuzchen wird die neue Base gemäß der Verteilung $(\pi_A, \pi_C, \pi_G, \pi_T)$ gezogen, bei Kringeln gemäß $(\pi_A/(\pi_A+\pi_G), \pi_G/(\pi_A+\pi_G))$ bzw. gemäß $(\pi_C/(\pi_C+\pi_T), \pi_T/(\pi_C+\pi_T))$. Letzteres entspricht also den Wahrscheinlichkeiten der Basentypen bedingt auf die Information, ob es sich um ein Purin oder Pyrimidin handelt. Man beachte, dass weder ein Kreuzchen noch ein Kringel eine Substitution im eigentlichen Sinne bedeuten muss. Wie man sich leicht überlegen oder nachrechen kann, hat eine Position, die erst von einem Kreuzchen und dann von einem Kringel getroffen wird, unabhängig von ihrer Urbase die stationäre Wahrscheinlichkeitsverteilung $(\pi_A, \pi_C, \pi_G, \pi_T)$, so als wäre sie nur von einem Kreuzchen getroffen worden. Wir können nun die Übergangswahrscheinlichkeiten leicht herleiten. So ist etwa $P_{A\to C}(t)$ das Produkt aus der Wahrscheinlichkeit, dass die Position in der Zeit bis t von mindestens einem Kreuzchen getroffen wurde, und der Wahrscheinlichkeit, dass beim letzten Kreuzchen oder Kringel ein `C` eingefügt wurde. Da die Zeit bis zum ersten Kreuzchen exponentialverteilt mit Rate λ ist und die Verteilung einer Position, die von mindestens einem Kreuzchen getroffen wurde, der stationären Verteilung entspricht, folgt:

$$P_{A\to C}(t) = \left(1 - e^{-\lambda t}\right) \cdot \pi_C$$

Zur Berechnung von $P_{A\to G}(t)$ müssen wir zwei Möglichkeiten in Betracht ziehen: Entweder die Position wurde von mindestens einem Kreuzchen getroffen oder von keinem Kreuzchen und von mindestens einem Kringel. Wir erhalten damit:

$$P_{A\to G}(t) = \left(1 - e^{-\lambda t}\right) \cdot \pi_G + e^{-\lambda t}\left(1 - e^{-\mu t}\right) \cdot \pi_G/(\pi_A + \pi_G)$$

Wir brauchen also die Ratenmatrix eigentlich nicht, um die Übergangswahrscheinlichkeiten zu berechnen. Aber es ist auch nicht schwer, auf die Ratenmatrix zu kommen:

$$\begin{pmatrix}
-\lambda(1-\pi_A) - \frac{\mu\pi_G}{\pi_A+\pi_G} & \lambda\pi_C & \lambda\pi_G + \frac{\mu\pi_G}{\pi_A+\pi_G} & \lambda\pi_T \\
\lambda\pi_A & -\lambda(1-\pi_C) - \frac{\mu\pi_T}{\pi_C+\pi_T} & \lambda\pi_G & \lambda\pi_T + \frac{\mu\pi_T}{\pi_C+\pi_T} \\
\lambda\pi_A + \frac{\mu\pi_A}{\pi_A+\pi_G} & \lambda\pi_C & -\lambda(1-\pi_G) - \frac{\mu\pi_A}{\pi_A+\pi_G} & \lambda\pi_T \\
\lambda\pi_A & \lambda\pi_C + \frac{\mu\pi_C}{\pi_C+\pi_T} & \lambda\pi_G & -\lambda(1-\pi_T) - \frac{\mu\pi_C}{\pi_C+\pi_T}
\end{pmatrix}$$

4.4.4 Konvergenz in die stationäre Verteilung

Sei $X = (X_1, X_2, \ldots)$ bzw. $(X_t)_{t\in\mathbb{R}_{\leq 0}}$ eine Markoff-Kette mit endlichem Zustandsraum $\mathcal{Z}$ und Übergangswahrscheinlichkeiten $P_{x\to y}(t)$ für $t \in \mathbb{N}$ bzw. $t \in \mathbb{R}_{\geq 0}$.

Die Übergangsdynamik P heißt **irreduzibel**, falls $\forall_{x,y\in\mathcal{Z}}\exists_t : \; P_{x\to y}(t) > 0$. Im Fall diskreter Zeit heißt P **periodisch**, falls ein $z \in \mathcal{Z}$ und ein $k > 1$ existieren mit $P_{z\to z}(n)$ für alle $n \in \mathbb{N} \setminus \{k, 2k, 3k, \ldots\}$, falls also der Zustand nur zu gewissen periodisch wiederkehrenden Zeiten besucht werden kann. Sonst heißt P **aperiodisch**.

Wichtig zu wissen ist, dass jede aperiodische irreduzible Übergangsdynamik P auf einem endlichen Zustandsraum $\mathcal{Z}$ genau eine stationäre Verteilung $(\pi_z)_{z\in\mathcal{Z}}$ besitzt und gegen dieses Gleichgewicht konvergiert, das heißt:

$$\forall_{x,z} \lim_{t\to\infty} P_{x\to z}(t) = \pi_z$$

Zum Beispiel konvergieren die Evolutionsdynamiken in den Modellen F81, F84 und HKY gegen ihr Gleichgewicht $(\pi_A, \pi_C, \pi_G, \pi_T)$.

Wir wollen obigen Konvergenzsatz nicht in allen technischen Details beweisen. Wir wollen aber mit einem stochastischen Argument verstehen, wieso jede aperiodische irreduzible Kette auf einem endlichen Zustandsraum gegen eine stationäre Verteilung (deren Existenz wir mal voraussetzen) konvergieren muss. Das Argument beruht auf dem Koppeln zweier Prozesse: Wir betrachten zwei Markoff-Ketten $X_1, X_2, X_3, \ldots$ und $Y_1, Y_2, Y_3, \ldots$, die beide derselben irreduziblen aperiodischen Übergangsdynamik $P_{x\to y}$ folgen. X_1 sei gemäß der stationären Verteilung gewählt. Damit ist bereits klar, dass X_i für jedes i gemäß der stationären Verteilung verteilt ist. Y_1 sei beliebig verteilt. Wir wollen zeigen, dass Y gegen die stationäre Verteilung $(\pi_z)_z$ konvergiert. Dazu stellen wir zusätzliche Forderungen an das Verhalten von X, ohne die obigen Annahmen zu verletzen. Wir lassen nämlich X unabhängig von Y gemäß der Übergangsdynamik unabhängig von Y laufen, bis sich die beiden Ketten zum ersten Mal treffen. Ab dann lassen wir X immer dasselbe machen wie Y, was offensichtlich obigen Annahmen nicht widerspricht, in Formeln:

$$X_i = Y_i \Rightarrow \forall_{j>i} X_j := Y_j$$

Es sei q_j die Wahrscheinlichkeit, dass sich X und Y im j-ten Schritt noch nicht getroffen haben. Dann gilt $|\mathrm{Ws}(Y_j = z) - \pi_z| = |\mathrm{Ws}(Y_j = z) - \mathrm{Ws}(X_j = z)| = |\mathrm{Ws}(Y_j = z, X_j = Y_j) + \mathrm{Ws}(Y_j = z, X_j \neq Y_j) - \mathrm{Ws}(X_j = z, X_j = Y_j) - \mathrm{Ws}(X_j = z, X_j \neq Y_j)| = |\mathrm{Ws}(Y_j = z, X_j \neq Y_j) - \mathrm{Ws}(X_j = z, X_j \neq Y_j)| \leq \max\{\mathrm{Ws}(Y_j = z, X_j \neq Y_j)\ ,\ \mathrm{Ws}(X_j = z, X_j \neq Y_j)\} \leq q_j$. Um zu zeigen, dass Y gegen die stationäre Verteilung konvergiert, genügt es also zu zeigen, dass q_j gegen 0 konvergiert. Wegen der Irreduzibilität besucht X jeden Zustand z beliebig oft. Bei jeder der dazwischenliegenden Exkursionen hat X eine gewisse positive Mindestwahrscheinlichkeit, Y zu treffen, bevor es wieder in z ist. Also fällt q_j exponentiell in j. Man beachte, dass hier die Irreduzibilität und die Aperiodizität eingehen: Wäre $P_{x\to y}$ reduzibel, könnten in X und Y in zwei verschiedenen Teilmengen des Zustandsraums gefangen sein. Wäre die Übergangskette periodisch, könnten sich X und Y ebenfalls ausweichen. So könnte der Zustandsraum in zwei Teilmengen unterteilt sein, und für gerade bzw. ungerade j ist X in einer der beiden Teilmengen und Y in der jeweils anderen. □

Wir werden es fast nur mit irreduziblen, aperiodischen Markoff-Ketten zu tun haben.

Eine Markoff'sche Übergangsdynamik P mit stationärer Verteilung $(\pi_z)_{z\in\mathcal{Z}}$ heißt **reversibel**, falls

gilt

$$\forall_{z,y\in\mathcal{Z}}:\ \pi_z \cdot P_{z\to y}(t) = \pi_y \cdot P_{y\to z}(t)$$

Anschaulich bedeutet das, dass unter der Voraussetzung, dass der Prozess im Gleichgewicht startet, nicht zu unterscheiden ist, ob der Prozess vorwärts oder rückwärts läuft (im Sinne eines Films, den man sich rückwärts ansieht). Die obige Gleichung wird auch als "detaillierte Balance-Gleichung" bezeichnet. Man mache sich klar, dass aus ihr insbesondere folgt, dass $(\pi_z)_{z\in\mathcal{Z}}$ stationäre Verteilung für die Übergangsdynamik P ist.

Die Evolutionsdynamiken, die durch Jukes-Cantor, F81, F84, HKY sowie PAM-Matrizen beschrieben werden, sind reversibel.

Hier ein Beispiel für einen nicht reversiblen Prozess: Sei $P_{A\to A}(1) = P_{A\to C}(1) = P_{C\to C}(1) = P_{C\to G}(1) = P_{G\to G}(1) = P_{G\to T}(1) = P_{T\to T}(1) = P_{T\to A}(1) = 0.5$ und alle anderen Übergangswahrscheinlichkeiten dementsprechend 0. Für diese Kette ist $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ die stationäre Verteilung, gegen die der Prozess auch konvergiert, denn er ist irreduzibel und aperiodisch. Der Prozess ist aber offensichtlich nicht reversibel.

Wenn wir von einem reversiblen Evolutionsmodell ohne molekulare Uhr ausgehen, hängt die Likelihood eines Baumes nicht von der Position der Wurzel ab. Wie bei parsimonischen Verfahren sollte dann also nur der ungewurzelte Baum ausgegeben werden. Wenn wir nämlich auf einer Kante die Wurzel r mit Abstand s bzw. t zu den beiden Knoten einfügen (vorausgesetzt die Kante hat Gesamtlänge $s+t$), dann erhalten wir für die Wahrscheinlichkeit, dass sich die Knoten in den Zuständen x und y befinden, unter Voraussetzung der Reversibilität:

$$\begin{aligned}\sum_z \pi_z \cdot P_{z\to x}(s) \cdot P_{z\to y}(t) &= \sum_z \pi_x \cdot P_{x\to z}(s) \cdot P_{z\to y}(t)\\ &= \pi_x \cdot P_{x\to y}(s+t)\\ &= \pi_y \cdot P_{y\to x}(s+t)\end{aligned}$$

Die Wahrscheinlichkeit hängt also nur von der Länge der Kante ab und nicht davon, wo wir die Wurzel genau einfügen. Ebenso kann man sich auch überlegen, dass es egal ist, auf welcher Kante man eine Wurzel einfügt.

4.4.5 DNA-Substitutionsmodelle im Überblick

Nun folt ein Überblick über die Mutatiosraten der meistbenutzten DNA-Substitutionsmodelle. Bei den ersten beiden ist exemplarisch auch die Ratenmatrix angegeben.

Beim **Jukes-Cantor-Modell (JC)** wird der Nukleotidtyp nicht berücksichtigt:

von\ nach	A	C	G	T
A	—	α	α	α
C	α	—	α	α
G	α	α	—	α
T	α	α	α	—

$$\begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}.$$

Bei **Kimuras 2-Parameter-Modell (K2)** wird berücksichtigt, dass Transitionen häufiger vorkommen als Transversionen.

von\ nach	A	C	G	T
A	—	α	β	α
C	α	—	α	β
G	β	α	—	α
T	α	β	α	—

$$\begin{pmatrix} -2\alpha-\beta & \alpha & \beta & \alpha \\ \alpha & -2\alpha-\beta & \alpha & \beta \\ \beta & \alpha & -2\alpha-\beta & \alpha \\ \alpha & \beta & \alpha & -2\alpha-\beta \end{pmatrix}.$$

Felsenstein (1981) **(F81)** berücksichtigt die möglicherweise ungleichen Basenhäufigkeiten $(\pi_A, \pi_C, \pi_G, \pi_T)$.

von\ nach	A	C	G	T
A	—	$\alpha\pi_C$	$\alpha\pi_G$	$\alpha\pi_T$
C	$\alpha\pi_A$	—	$\alpha\pi_G$	$\alpha\pi_T$
G	$\alpha\pi_A$	$\alpha\pi_C$	—	$\alpha\pi_T$
T	$\alpha\pi_A$	$\alpha\pi_C$	$\alpha\pi_G$	—

Im Modell von **Hasegawa, Kishino und Yano (HKY)** sind sowohl die Basenhäufigkeiten als auch der Unterschied zwischen Transitions- und Transversionshäufigkeiten berücksichtigt.

von\ nach	A	C	G	T
A	—	$\alpha\pi_C$	$\beta\pi_G$	$\alpha\pi_T$
C	$\alpha\pi_A$	—	$\alpha\pi_G$	$\beta\pi_T$
G	$\beta\pi_A$	$\alpha\pi_C$	—	$\alpha\pi_T$
T	$\alpha\pi_A$	$\beta\pi_C$	$\alpha\pi_G$	—

Felsenstein (1984) **(F84)** berücksichtigt ebenfalls Basenhäufigkeiten und Unterschiede zwischen Transverions- und Transitionsrate. Außerdem benötigt man bei F84 keine Matrizenrechnug zur Berechnung der Übergangswahrscheinlichkeiten.

von\ nach	A	C	G	T
A	—	$\lambda\pi_C$	$\lambda\pi_G + \frac{\mu\pi_G}{\pi_A+\pi_G}$	$\lambda\pi_T$
C	$\lambda\pi_A$	—	$\lambda\pi_G$	$\lambda\pi_T + \frac{\mu\pi_T}{\pi_C+\pi_T}$
G	$\lambda\pi_A + \frac{\mu\pi_A}{\pi_A+\pi_G}$	$\lambda\pi_C$	—	$\lambda\pi_T$
T	$\lambda\pi_A$	$\lambda\pi_C + \frac{\mu\pi_C}{\pi_C+\pi_T}$	$\lambda\pi_G$	—

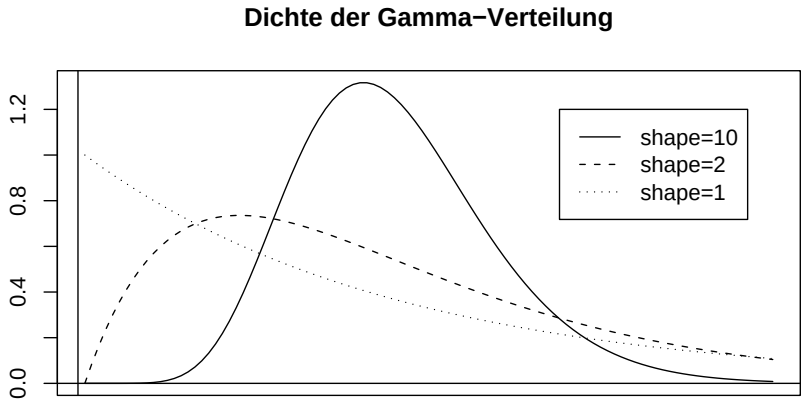
Im **Allgemeines Zeit-reversibles Modell (GTR)** werden weitere Unterschiede zwischen den Mutationsraten zwischen den einzelnen Nukletidtypen berücksichtigt.

von\ nach	A	C	G	T
A	—	$\alpha\pi_C$	$\beta\pi_G$	$\gamma\pi_T$
C	$\alpha\pi_A$	—	$\delta\pi_G$	$\epsilon\pi_T$
G	$\beta\pi_A$	$\delta\pi_C$	—	$\eta\pi_T$
T	$\gamma\pi_A$	$\epsilon\pi_C$	$\eta\pi_G$	—

Bei den Modellen F81, F84, HKY und GTR $(\pi_A, \pi_C, \pi_G, \pi_T)$ ist stationäre Verteilung, bei JC und K2 $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$. Jedes dieser Modelle ist reversibel.

4.4.6 Positionsabhängige Mutationraten

Die bisher beschriebenen Substitutionsmodelle kann man um die Möglichkeit erweitern, dass sich die verschiedenen Positionen in ihren Mutationsraten unterscheiden. Wenn man modellieren will, dass es verschiedene Regionen gibt, die unterschiedlich schnell evoluieren, ohne dass die Grenzen dieser Abschnitte im vorhinein bekannt sind, sind hidden-Markoff-Modelle ein geeigneter Ansatz. In diesem Abschnitt geht es aber um den Fall, dass sich die Mutationraten bei benachbarten Positionen ebenso unterscheiden können wie bei entfernten. Der Ansatz ist hier, dass es für jede Position i des Sequenzalignments einen Faktor r_i gibt, so dass $R_i = r_i \cdot Q$ die Substitutionsratenmatrix für Position i ist. Dabei ist die Matrix Q, durch die zum Beispiel definiert ist, um welchen Faktor sich die Raten für Transitionen und Transversionen unterscheiden, für alle Positionen gleich. Man könnte $r_1, \ldots, r_n$ als zusätzliche Parameter auffassen, wobei n die Sequenzlänge ist. Das wären dann allerdings in der Regel so viele, dass eine Schätzung aller Mutationsparameter aus den Daten nicht mehr sinnvoll möglich wäre. Praktikabel ist hingegen, davon auszugehen, dass $r_1, \ldots, r_n$ zufällige Faktoren sind, die unabhängig voneinander aus einer gemeinsamen Verteilung kommen. Es genügt dann ein Parameter α, mit dem man festlegen kann, wie stark die r_i variieren sollen. Um die gemeinsame Verteilung der r_i zu modellieren, ist die Familie der Gammaverteilungen gut geeignet, deren Dichten je nach Wahl des sogenannten Formparameters (engl. *shape*) α ganz unterschiedliche Formen annehmen kann, wie die folgende Abbildung zeigt, und damit in recht vielen Fällen die Ratenheterogenität gut annähern kann.



Eigentlich hat die Familie der Gammaverteilungen noch einen zweiten Parameter, den sogenannten Skalenparameter β. Wie der Name andeutet, kann man diesen als konstanten Faktor auffassen. Er legt also fest, wo in der vorherigen Abbildung auf der x-Achse die 1 liegt. Das ist im hier diskutierten Kontext nicht relevant, denn wir können die Skala festlegen, indem wir die Einträge der Matrix Q oder die Kantenlängen des Stammbaums mit einer Konstante multiplizieren. In der obigen Abbildung wurde $\beta = 1/\alpha$ verwendet, so dass jede der drei Verteilungen, deren Dichten in der Abbildung gezeigt werden, Erwartungwert 1 hat. Allgemein hat die Gammverteilung mit Formparameter α und Skalenparameter β

Erwartungswert $\alpha \cdot \beta$ und Varianz $\alpha \cdot \beta^2$.

Die Dichte der Gammaverteilung mit Formparameter α und Skalenparameter β an der Stelle $x \in [0, \infty]$ ist

$$g_{\alpha,\beta}(x) := \frac{x^{\alpha-1} \cdot e^{-x/\beta}}{\beta^\alpha \cdot \Gamma(\alpha)},$$

wobei Γ die Gammafunktion mit $\Gamma(a) = \int_0^\infty x^{a-1} \cdot e^{-x} dx$ bezeichnet. Die Wahrscheinlichkeit, dass eine so verteilte Zufallvariable einen Wert zwischen u und v annimmt (mit $0 \leq u \leq v$), ist also

$$\frac{1}{\beta^\alpha \cdot \Gamma(\alpha)} \int_u^v x^{\alpha-1} \cdot e^{-x/\beta} dx.$$

Wir gehen im Folgenden von $\beta = 1/\alpha$ aus und verwenden die Dichte $g_\alpha := g_{\alpha,\, 1/\alpha}$.

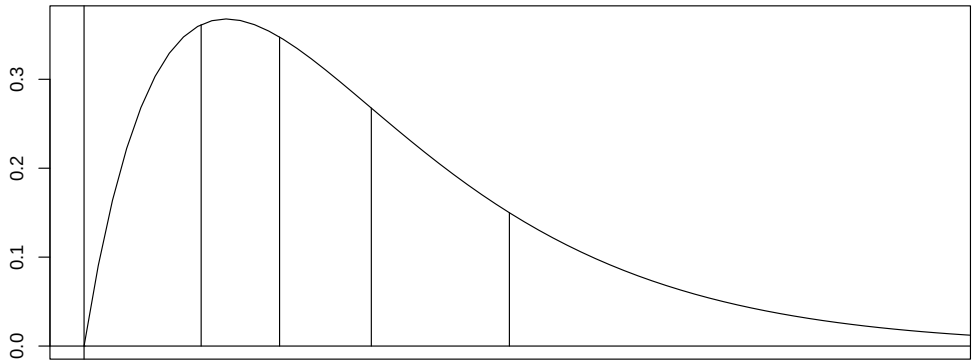
Wie berechnen wir in diesem Modell die Likelihood eines Baumes bzw. eines Baumes und eines bestimmten Werts für α, falls dieser Parameter nicht als bekannt vorausgesetzt wird? Wie üblich, ist die Likelihood des Baumes (bzw. des Baumes und des Werts für α) die Wahrscheinlichkeit der Daten unter Annahme des Baumes und des Werts für α, und diese Wahrscheinlichkeit ist das Produkt der entsprechenden Wahrscheinlichkeiten aller Positionen i. Mit Felsensteins Pruning-Algorithmus können wir die Wahrscheinlichkeit der Daten D_i an Position i ermitteln, wenn wir die Ratenmatrix $R_i = r_i \cdot Q$ kennen. Da Q als gegeben vorausgesetzt ist (oder bestimmte Werte für Q eingesetzt werden, um deren Likelihood zu berechnen), können wir mit Felsenstein-Pruning zumindest die auf einen festen Wert x für r_i bedingte Wahrscheinlichkeit $\Pr(D_i \mid r_i = x)$ der Daten D_i berechnen. Um daraus die nicht-bedingte Wahrscheinlichkeit zu ermitteln, müssten wir für x sämtliche Werte zwischen 0 und ∞ einsetzen und die entsprechend bedingten Wahrscheinlichkeiten aufintegrieren, die dabei jeweils mit der zu r_i gehörigen Wahrscheinlichkeitsdichte $g_\alpha(x)$ an der Stelle x zu gewichten sind:

$$\Pr(D_i) = \int_0^\infty \Pr(D_i \mid r_i = x) \cdot g_\alpha(x)\, dx.$$

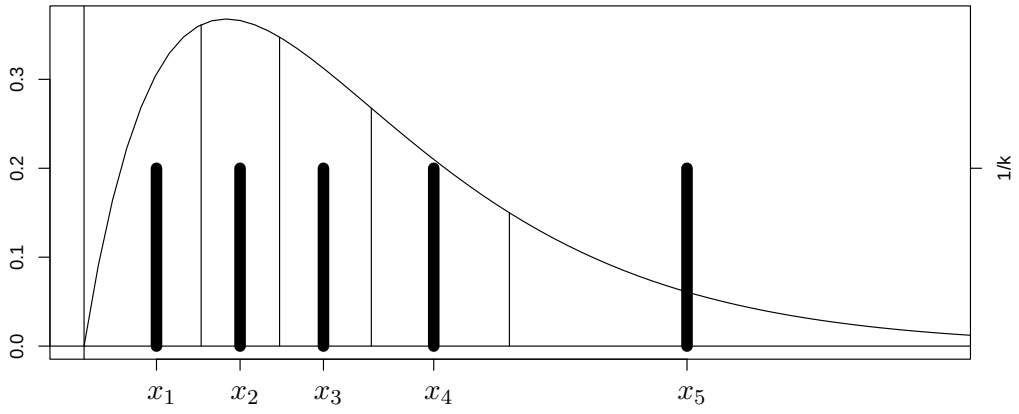
Dieses Integral können wir nicht exakt berechnen, daher nähern wir es an, indem wir für einige Werte $x_j = x_1, x_2, \ldots, x_k$ jeweils mit Felsenstein-Pruning $\Pr(D_i \mid r_i = x_j)$ berechnen und dann das Integral durch eine gewichtete Summe approximieren:

$$\Pr(D_i) = \int_0^\infty \Pr(D_i \mid r_i = x) \cdot g_\alpha(x)\, dx \approx \sum_{j=1}^{k} w_j \cdot \Pr(D_i \mid r_i = x_j).$$

Es gibt mehrere Möglichkeiten, die Punkte x_j und dazu passende Gewichte w_j zu wählen und somit die Gammaverteilung zu diskretisieren. Häufig wird die Methode von Yang (1994) verwendet. Dazu wird das Intervall $[0, \infty]$ zunächst in k Abschnitte gleicher Wahrscheinlichkeit aufgeteilt. Für jeden solchen Abschnitt $[a, b]$ muss also $\int_a^b g_\alpha(x) dx = 1/k$ gelten. Anschaulich bedeutet das, dass die Fläche unter der Dichte-Kurve der Gamma-Verteilung in k gleich große Stücke unterteilt wird, hier für $k = 5$:



Der Wert x_j ist dann der Erwartungswert einer Gamma-verteilten Zufallsvariablen, die darauf bedingt ist, in den j-ten Abschnitt zu fallen. Anschaulich entspricht dies dem Schwerpunkt das zugehörigen Flächenstücks auf die x-Achse. Alle Gewichte w_j werden dann auf $1/k$ gesetzt. In folgender Abbildung ist dies für den Fall $k = 5$ dargestellt:



Man kann das Modell auch gleich so formulieren, dass für die Faktoren r_i nur jeweils die Werte $x_1, \ldots, x_k$ zulässig sind, und zwar jeweils mit uniformer a-proiori-Wahrscheinlichkeitsverteilung. Mit dem Parameter α kann man dann steuern, ob eher die kleinen x_j-Werte oder eher die mittleren x_j-Werte nahe beieinander liegen.

Als Erweiterung des Modells mit gammaverteilten Raten kann man noch zulassen, dass sich jede Position mit einer bestimmten Wahrscheinlichkeit dafür entscheidet, invariant zu sein. An diesen Positionen sind dann keine Mutationen erlaubt.

4.4.7 Modellwahl

Angesichts der oben dargestellten Fülle an Modellen für die Evolution von DNA-Sequenzen stellt sich die Frage, welches Modell verwendet werden sollte, z.B. wenn aus gegebenen Sequenzdaten ein Stammbaum geschätzt werden soll.

Wie gut ein Modell M zu einem Datensatz D passt, kann man daran messen, wie hoch die $L_D(M) = \mathrm{Ws}_M(D)$ oder auch die log-Likelihood $\log(L_D(M))$ wird, wenn die Modellparamter optimal an die Daten angepasst werden. Allerdings kann man diese (log-)Likelihood immer steigern, indem man die Modelle komplexer macht, also mehr Parameter hinzunimmt. Je größer die Anzahl d der Parameter des Modells M, desto größer das Risiko einer Überanpassung (engl. Overfitting). In diesem Fall werden die vielen Parameter benutzt, um das Modell auch an geringe Zufallsschwankungen in den Daten anzupassen. Wendet man das Modell dann auf neue Daten an, die aus der selben Verteilung kommen wie die ursprünglichen "Trainingsdaten", so passt das Modell weniger gut als ein einfacheres Modell, das nur die wesentlichen Parameter enthält.

Akaike (1974) zeigt, dass bei bestimmten Modellen, die auf normalverteilten Zufallsvariablen basieren, der Fehler, den man macht, wenn man ein Modell mit d Parametern auf einen Datensatz anpasst und dann auf einen vergleichbaren Datensatz überträgt, ungefähr Akaike-Informationskriterium entspricht:

$$\mathrm{AIC} = -2 \log L_D(M) + 2d$$

Man wählt dann das Modell, bei dem AIC minimal wird.

Ein anderer Ansatz führt zum Bayes-Informations-Kriterium

$$\mathrm{BIC} = -2 \log L_D(M) + d \log n.$$

Dabei ist n die Anzahl der unabhängigen Messungen. Der Bayessche Ansatz setzt dabei voraus, dass alle Modelle *a priori* gleich wahrscheinlich sind. Die *a posteriori* Wahrscheinlichkeit des Modells M ist

$$\mathrm{Ws}(M \mid D) = \frac{\mathrm{Ws}(M, D)}{\mathrm{Ws}(D)} = \frac{\mathrm{Ws}(D \mid M) \cdot \mathrm{Ws}(M)}{\mathrm{Ws}(D)}$$

Unter gewissen Annahmen gilt dann:

$$\log \frac{\mathrm{Ws}(M_1 \mid D)}{\mathrm{Ws}(M_2 \mid D)} \approx (2 \log L_D(M_1) - d_1 \log n) - (2 \log L_D(M_2) - d_2 \log n)$$

AIC bestraft also jeden neuen Parameter mit 2 und BIC mit $\log n$, was in der Regel größer als 2 ist. Also bevorzugt BIC im Vergleich zu AIC die einfacheren Modelle.

Wenn ein Modell M_1 ein Spezialfall von M_2 ist, so dass man M_1 aus M_2 erhält, indem man bestimmte Parameter auf gewisse Werte festlegt, sagt man M_1 sei eingebettet (engl. nested) in M_2. In gewissem Sinne wird dann M_2 mindestens genauso gut auf die Daten passen wie M_1, denn allgemein gilt $\mathrm{Ws}_{M_1}(D) \leq \mathrm{Ws}_{M_2}(D)$. In den meisten Fällen gilt $L_D(M_1) = \mathrm{Ws}_{M_1}(D) < \mathrm{Ws}_{M_2}(D) = L_D(M_2)$. Die Frage ist dann ob $L_D(M_2)$ *signifikant* höher ist als $L_D(M_2)$. Dies kann man entscheiden, indem man als Teststatistik den log-Likelihoodquotienten betrachtet.

$$\frac{L_D(M_2)}{L_D(M_1)} \qquad (\mathrm{mit} L_D(M) := \mathrm{Ws}_M(D))$$

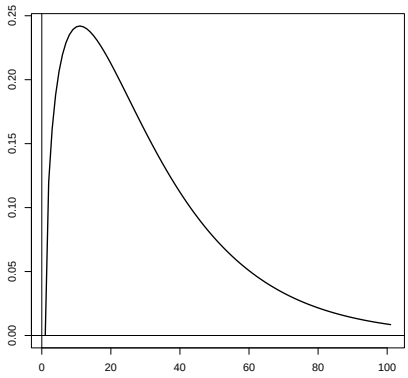
Dieser ist unter der Nullhypothese, dass Modell M_1 zutrifft, und die Erhöhung der Likelihood in M_2 nur auf Zufallsschwankungen zurückzuführen ist, in den meisten Fällen asymptotisch χ^2-verteilt (sprich: chi-quadrat) mit $d = d_2 - d_1$ Freiheitsgraden, wobei d_i die Anzahl der Parameter im Modell M_i bezeichnet. "Asymptotisch" heißt hier, dass es um den Grenzwert sehr vieler unabhängiger Messungen geht, in unserem Fall also um sehr lange Sequenzen. In symbolischer Schreibweise:

$$\mathcal{L}_{M_1}\left(\log\frac{L_D(M_2)}{L_D(M_1)}\right) \sim \chi^2_{\sharp\{\text{zusätzliche Parameter}\}}$$

Die χ^2-Verteilung mit d Freiheitsgraden besitzt die Dichte

$$\frac{x^{\frac{d}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{d}{2}}\cdot\Gamma\left(\frac{d}{2}\right)}.$$

Folgende Abbildung zeigt die Dichte der χ^2-Verteilung mit $d = 3$ Freiheitsgraden.

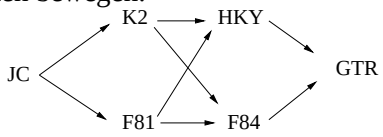


Man kann eine mit d Freiheitsgraden χ^2-verteilte Zufallsvariable simulieren, indem man d unabhängige standardnormalverteilte Zufallsvariablen quadriert und aufaddiert. Anders formuliert: Ist V ein d-dimensionaler standardnormalverteilter Zufallsvektor, so ist $||V||^2 =< V, V >$, also das Quadrat seiner Länge, χ^2-verteilt mit d Freiheitsgraden.

Um zu beurteilen, wie signifikant der log-Likelihoodquotient für M_2 spricht, muss man also ermitteln wie unwahrscheinlich es ist, dass eine χ^2-verteilte Zufallsvariable mit der passenden Anzahl an Freiheitgraden einen mindestens so hohen Wert annimmt. Das kann man entweder durch numerische Berechnung tun, etwa indem man einen Befehl in einem Softwarepaket wie R (`www.r-project.org`) aufruft, oder indem man in einer Tabelle nachschaut, siehe z.B:
`http://de.wikibooks.org/wiki/Mathematik:_Statistik:_Tabelle_der_Chi-Quadrat-Verteilung`

Wenn die Gültigkeit der χ^2-Asymptotik zweifelhaft ist, z.B. wenn die Sequenzen nicht sehr lang sind, kann man sich mit Simulationsstudien behelfen. Dazu simuliert man mehrere Datensätze gemäß M_1, passt die Modelle an diese Daten an und überprüft wie häufig es vorkommt, dass der log-Likelihoodquotient bei den simulierten Daten (mindestens) so hoch ist wie bei den Originaldaten.

Durch die Likelihoodquotiententests (LRTs) kann man zwei Modelle nur dann direkt vergleichen, wenn eines davon in das andere eingebettet ist. Um zwischen mehreren Modellen zu entscheiden, kann man mit einem davon beginnen und sukzessive einzelne Parameter hinzunehmen oder weglassen, je nachdem LRT Signifikanz anzeigt. Auf diese Weise kann man sich etwa bei der Wahl eines Substitutionsmodells durch folgenden Graphen bewegen:



Das Ergebnis hängt dann allerdings davon ab, mit welchem Modell man beginnt und welche Zwischenschritte man zulässt. Man sollte sich außerdem klar machen, dass der Signifikanzbegriff bei dieser Vorgehensweise leicht zweckentfremdet und möglicherweise überstrapaziert wird. Wenn ein statistischer Test keine Signifikanz ergibt, kann die die Nullhypothese zunächst nicht verworfen werden. Daraus folgt jedoch nicht, dass die Nullhypothese, in unserem Fall also das eingebettete Modell, gültig oder allgemein zu bevorzugen ist.

Die Modellwahl sollte unter anderem folgende Punkte berücksichtigen:

- Die Komplexität und Eigenschaften des zu modellierenden Systems
- Die Größe und die Qualität des zur Verfügung stehenden Datensatzes
- Die Fragestellung

Der letzte Punkt, die Fragestellung, wird weder von AIC oder BIC noch von LRT berücksichtigt. Wer den Aufwand einer sorgfältigen Analyse nicht scheut, kann zur Modellwahl Simulationsstudien durchführen. Stehen mehrere Modelle $M_1, M_2, \ldots, M_k$ zur Auswahl und ist $\widehat{\theta}_i$ jeweils die auf M_i basierende Schätzung, z.B. der geschätzte Stammbaum, so kann man ähnlich wie beim Bootstrap untersuchen, wie gravierend die Fehler wären, die man machen würde, wenn $\widehat{\theta}_i$ tatsächlich der gesuchten Wahrheit entspräche und man die Analyse auf das Modell M_j stützen würde. Dabei sollte man auch den Fall $i = j$ einbeziehen. Es ist durchaus möglich, dass bei Daten, die gemäß einem Modell M_i erzeugt wurden, ein anderes Modell M_j in der Analyse bessere Ergebnisse liefert als M_i selbst. Das kann etwa dann der Fall sein, wenn M_j in M_i eingebettet ist und die wahren Parameterwerte so liegen, dass M_j fast gilt. Ein Beispiel: Angenommen DNA-Sequenzdaten entsprechen dem HKY-Modell und die theoretischen Basenwahrscheinlichkeiten $(\pi_A, \pi_C, \pi_G, \pi_T)$ liegen alle sehr nahe bei 1/4. Dann ist es wahrscheinlich, dass phylogenetische Analysen, die auf dem Kimura-2-Parametermodell (K2) beruhen, bessere Ergebnisse liefern, als solche, die auf HKY basieren.

4.5 Die Markoffketten-Monte-Carlo-Methode (MCMC)

Sei D ein Datensatz von n Sequenzen der Länge m. Wenn wir auf der Menge $\mathcal{T}$ der in Frage kommenden m-blättrigen Binärbäume mit Astlängen eine a-priori-Wahrscheinlichkeitsverteilung $(P(T))_{T\in\mathcal{T}}$ definieren (z.B.: alle Bäume bis zu einer gewissen Gesamtastlänge sind gleichwahrscheinlich, der Rest hat

Wahrscheinlichkeit 0), dann können wir die a-posteriori-Verteilung betrachten:

$$\mathrm{Ws}(T|D) = \frac{\mathrm{Ws}(T \cap D)}{\mathrm{Ws}(D)} = \frac{\mathrm{Ws}(D|T) \cdot P(T)}{\mathrm{Ws}(D)}$$

Mit diesem Bayes'schen Ansatz bekommen die Bäume nicht nur eine Likelihood sondern eine Wahrscheinlichkeit. In der Wahl der a-priori-Wahrscheinlichkeit steckt eine gewisse Beliebigkeit. Dafür können wir jetzt nicht nur den wahrscheinlichsten Baum ausgeben (analog zum ML-Baum beim klassischen Ansatz), sondern wir können uns eine Menge möglicher (man könnte auch sagen für die beobachteten Daten typischer) Bäume betrachten, indem wir uns zufällige Bäume gemäß der Wahrscheinlichkeitsverteilung $\mathrm{Ws}(T|D)$ erzeugen. Wie ist das effizient machbar, wenigstens approximativ? Eine Schwierigkeit, die bei Betrachtung der obigen Formeln auffällt, ist, dass die Konstante

$$\mathrm{Ws}(D) = \int_{T \in \mathcal{T}} \mathrm{Ws}(D \cap T) \mathrm{d}T$$

schwierig zu berechnen ist. Vielleicht kann man das aber umgehen!?

Abhilfe schafft der Metropolis-Hastings-Algorithmus, mit dem man approximativ aus Verteilungen samplen kann, die nur bis auf eine Konstante bekannt sind. Der Metropolis-Hastings-Algorithmus ist eine Markoffketten-Monte-Carlo-Methode.

Markoffketten-Monte-Carlo-Methoden (MCMC): Um eine Verteilung $(\pi_z)_{z \in \mathcal{Z}}$ zu simulieren, konstruiere eine aperiodische irreduzible Markoff-Kette mit Übergangsmatrix $(P_{x \to y})_{xy}$, die (π_z) als stationäre Verteilung hat. Starte einen Prozess mit dieser Übergangsdynamik, lass ihn lange genug laufen und gib dann nach einer gewissen Anzahl Schritten das $Z \in \mathcal{Z}$ aus, in dem er sich gerade befindet. Da der Prozess gegen die stationäre Verteilung konvergiert, gilt

$$\mathrm{Ws}(Z = z) \approx \pi_z.$$

Da man in der Regel nicht wissen kann, wie gut der Startpunkt der Kette gewählt ist, lässt man die Kette zunächst sehr lange laufen ("burn in"), bis man einen Wert ausgibt, von dem man dann annimmt, dass seine Verteilung hinreichend nahe am Gleichgewicht ist. Dann kann man gleich mehrere Werte ausgeben. Da man allerdings in der Regel (annähernd) stochastisch unabhängige Realisierungen aus der Gleichgewichtsverteilung haben möchte, sollte man auch zwischen den einzelnen Ausgaben hinreichend viele Schritte abwarten. Wieviele Schritte man warten sollte, ist eine schwierige Frage. Man kann zwar theoretisch bestimmen, wie schnell die Markoff-Kette konvergiert, praktisch ist das aber sehr schwierig. Letztlich muss man die Ergebnisse von MCMC-Verfahren einer strengen Analyse unterziehen, um etwa nach stochastischen Abhängigkeiten zu fahnden.

4.5.1 Metropolis-Hastings-Algorithmus

Der Metropolis-Hastings-Algorithmus ist eine Möglichkeit, einen Prozess zu konstruieren, der gegen $(\pi_z)_{z \in Z}$ konvergiert: Zunächst benötigt man dazu irgendeine irreduzible Übergangsverteilung $Q_{x \to y}$ auf

$\mathcal{Z}$ und lasse sie Vorschläge machen ("proposal step"). Wenn wir uns gerade in x befinden und "Q" schlägt vor (genauer gesagt ein Zufallsgenerator für Q), nach y zu gehen, dann akzeptiere das mit Wahrscheinlichkeit

$$\min\left\{1\ ,\ \frac{\pi_y}{\pi_x}\cdot\frac{Q_{y\to x}}{Q_{x\to y}}\right\}.$$

Andernfalls bleibe einen weiteren Schritt in x ("rejection step"). Zunächst fällt auf, dass die zu simulierende Verteilung nur als Quotient π_y/π_x eingeht. Daher muss sie nur bis auf einen konstanten Faktor bekannt sein, denn der Faktor kürzt sich ohnehin heraus. Wir erhalten somit eine Übergangsdynamik P, die irreduzibel ist, da Q irreduzibel ist. Da man gelegentlich mit einer gewissen Wahrscheinlichkeit bleibt, wo man ist, wird P selbst dann aperiodisch sein, wenn Q periodisch ist. Wir überlegen uns jetzt noch das Wesentliche, nämlich dass $(\pi_z)_{z\in\mathcal{Z}}$ stationäre Verteilung für P ist. Wir zeigen sogar, dass es sich um ein reversibles Gleichgewicht handelt ("detaillierte Balance"). Sei oBdA $\pi_x Q_{x\to y} \geq \pi_y Q_{y\to x}$. Dann gilt:

$$P_{y\to x} = Q_{y\to x}$$

und

$$P_{x\to y} = Q_{x\to y}\cdot\frac{\pi_y}{\pi_x}\cdot\frac{Q_{y\to x}}{Q_{x\to y}} = \frac{\pi_y}{\pi_x}\cdot Q_{y\to x}.$$

Also gilt $\pi_x P_{x\to y} = \pi_y P_{y\to x}$ und damit ist $(\pi_z)_{z\in\mathcal{Z}}$ reversibles Gleichgewicht für P und P konvergiert gegen diese Verteilung. □.

Wir haben nicht viel über die "proposal chain" Q (auch "reference chain", "Referenzkette") vorausgesetzt. Wie schnell P letztlich konvergiert, hängt aber selbstverständlich von der Wahl von Q ab. Die Kunst ist, ein Q zu finden, dessen Vorschläge möglichst oft akzeptiert werden und das gleichzeitig "gut mischt", d.h. es wäre nachteilig, wenn es $x, y \in \mathcal{Z}$ gäbe, so dass Q höchstwahrscheinlich extrem viele Schritte bräuchte, um von x nach y zu gelangen.

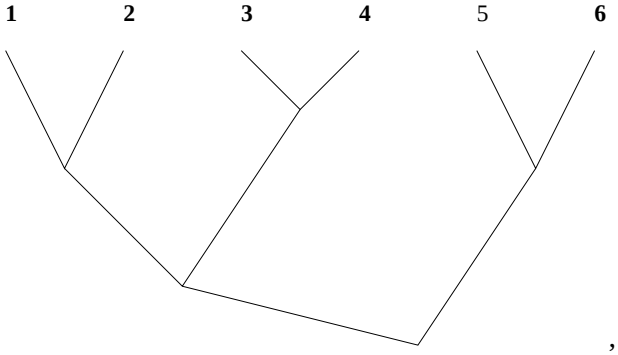
4.5.2 MCMC und simulated Annealing

MCMC-Methoden und speziell der Metropolis-Hastigs-Algorithmus sind dem Simulated-Annealing-Ansatz sehr ähnlich. Der wesentliche Unterschied ist, dass man bei Simulated Annealing nicht aus einer Verteilung samplen will, sondern eine Funktion optimieren will, was also etwa der Suche nach dem wahrscheinlichsten Element der Zustandsmenge entspricht. Daher verringert man beim Simulated Annealing nach und nach die Akzeptanzwahrscheinlichkeit für Schritte, die den zu optmierenden Funktionswert (z.B. die Wahrscheinlichkeit) verkleinern. Beim MCMC-Sampling muss man die Akzeptanz-Wahrscheinlichkeit sorgfältiger wählen als beim Simulated Annealing. Es genügt nicht, am Ende immer in Richtung des Optimums zu laufen; man muss sich jederzeit gemäß einer Übergangsdynamik bewegen, die gegen die Zielverteilung konvergiert.

4.5.3 Bäume samplen nach Mau, Newton und Larget

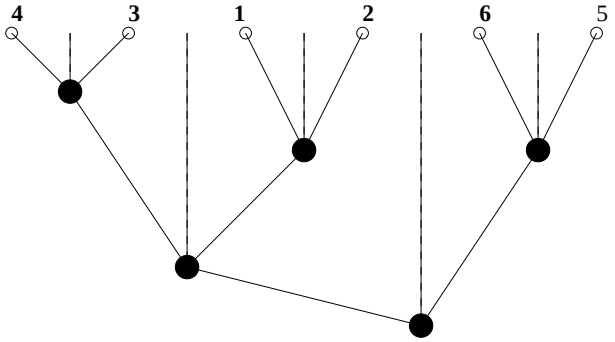
Mau, Newton und Larget (1996) schlagen für einen Metropolis-Hastings-Verfahren zum Samplen von Bäumen für gegebene Sequenzdaten unter Annahme der molekularen Uhr folgende Referenzkette Q vor:

Als aktueller Zustand sei ein gewurzelter Baum gegeben, z.B. folgender:

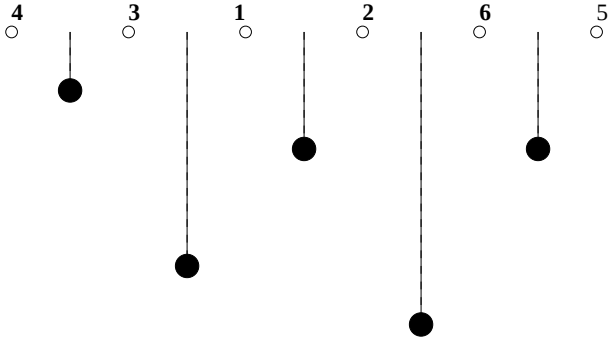


,

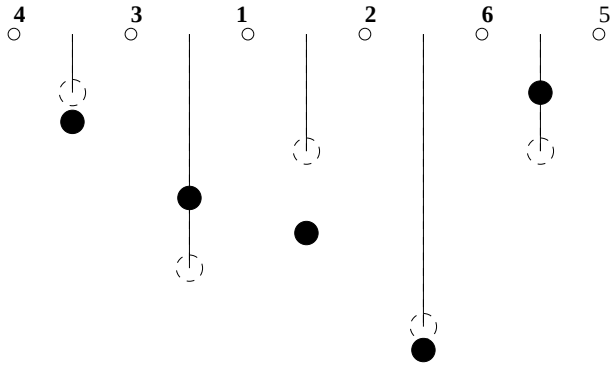
Zunächst wird der Baum in jedem Knoten rein zufällig gedreht, d.h. per Münzwurf wird entschieden welcher der beiden herauswachsenden Teilbäume als linker und als rechter anzusehen ist. Der Baum wird so in die Ebene gemalt, dass jeder Knoten horizontal gesehen zwischen den darüberliegenden Teilbäumen liegt:



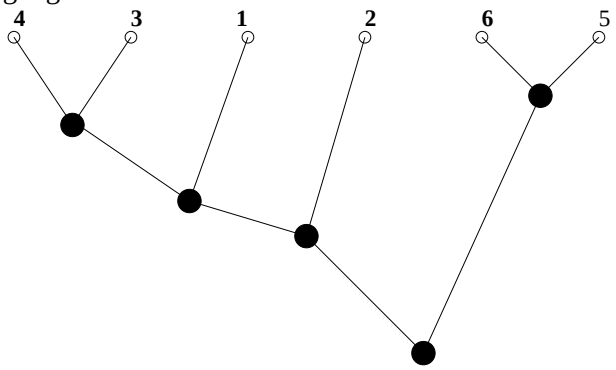
Dann brauchen wir die Kanten gar nicht einzuzeichnen. Aus der Lage der inneren Knoten einschließlich der Wurzel könnten wir den Baum komplett rekonstruieren:



Nun verschieben wir die inneren Knoten und die Wurzel jeweils um einen kleinen zufälligen Betrag:



Wenn wir jetzt die Kanten für die neue Lage der Knoten einzeichnen, erhalten wir einen Baum, bei dem sich auch die Topologie geändert haben kann:



Das ist der Baum, den Q vorschlägt. Entsprechend dem Metropolis-Hastigs-Algorithmus kann dieser nun in Abhängigkeit von seiner a-posteriori-Wahrscheinlichkeit und der des ursprünglichen Baumes sowie von den Übergangswahrscheinlichkeiten der Referenzkette akzeptiert oder verworfen werden.

4.6 Bootstrap

Ziel von Bootstrap-Methoden ist es, abzuschätzen, wie zuverlässig eine Schätzung ist. Die Anwendung in der Phylogenieschätzung wäre also, eine Idee davon zu bekommen, auf welche Kanten eines geschätzen Baumes man sich mehr verlassen kann, und auf welche weniger. Wir beginnen mit einem Beispiel aus der klassischen Populationsgenetik:

Beispiel: Gemäß Hardy-Weinberg gibt es in einer diploiden panmiktischen Population im Gleichgewicht bei zwei selektiv neutralen Allelen N und M die Genotypen MM, MN und NN in den relativen Häufigkeiten $(1-\theta)^2$, $2\theta(1-\theta)$ und θ^2, wobei θ die relative Häufigkeit von N ist.

In einer Stichprobe aus der Population werden folgende Häufigkeiten gefunden:

MM	MN	NN	Gesamt
342	500	187	1029
X	Y	Z	

Der ML-Schätzer für θ ist $\widehat{\theta} = \frac{2Z+Y}{2(X+Y+Z)} \approx 0.4247$ (Übung!). Dabei beachte man, dass das Tripel

multinomialverteilt ist:

$$\mathrm{Ws}_\theta(X = x, Y = y, Z = z) = \left(\frac{1029}{x! \cdot y! \cdot z!}\right)(1-\theta)^{2x}(2\theta(1-\theta))^y\theta^{2z}$$

Wie zuverlässig ist der Schätzer $\widehat{\theta}$?

Angenommen, $\widehat{\theta}$ wäre der wahre Parameterwert und jemand würde eine neue Stichprobe vom Umfang 1029 ziehen und den Parameter schätzen. Sei θ^* das Ergebnis. Wie weit ist dann θ^* von $\widehat{\theta}$ entfernt? Dem Bootstrap-Gedanken liegt die Idee zu Grunde, dass θ^* in Verteilung von $\widehat{\theta}$ ungefähr so weit weg ist wie $\widehat{\theta}$ vom wahren Wert θ, dass also die Verteilungen von $\theta^* - \widehat{\theta}$ und $\widehat{\theta} - \theta$ einander sehr ähnlich sind. Das ist zumindest dann plausibel, wenn $\widehat{\theta}$ nicht ganz danebenliegt. Wir simulieren obiges Gedankenexperiment 1000 mal und erhalten die simulierten Schätzwerte $\theta_1, \theta_2, \ldots, \theta_{1000}$. Aus diesen schätzen wir die Standardabweichung von $\widehat{\theta}$:

$$s_{\widehat{\theta}} \approx \sqrt{\frac{\sum_i (\theta_i^* - \widehat{\theta})^2}{1000}}$$

Das Beispiel ist Rice (1995) entnommen. Das dort erhaltene Ergebnis ist $s_{\widehat{\theta}} \approx 0{,}011$

Wir sind in diesem Beispiel nach dem **parametrischen Bootstrap** vorgegangen, d.h. wir haben unserer Simulation eine *Verteilungsannahme* zu Grunde gelegt und für den benötigten Parameter den Schätzwert eingesetzt.

Beim **nichtparametrischen Bootstrap** verzichtet man auf eine Verteilungsannahme und greift direkt auf die beobachteten Daten zu. Man erzeugt sich nämlich eine simulierte Stichprobe, indem man aus den Originaldaten mit Zurücklegen zieht. Auch hierzu ein einfaches

Beispiel: Aus einer Fischpopulation werden n Individuen gezogen. Der i-te Fisch der Stichprobe ist l_i cm lang und wiegt g_i Gramm. Sei $\hat{l}$ cm die durchschnittliche Länge und $\hat{g}$ Gramm das durchschnittliche Gewicht der Fische in der Stichprobe. Wie gut kann man das durchschittliche Gewicht $\bar{g}$ und die durchschnittliche Länge $\bar{l}$ der Fische in der gesamten Population durch $\hat{g}$ und $\hat{l}$ schätzen?

Um diese Genauigkeit zu schätzen, simulieren wir das Ziehen aus der Population, indem wir aus unserer Stichprobe n-mal mit Zurüklegen ziehen, d.h. einige Fische werden doppelt ausgewählt, andere gar nicht. Die empirische Verteilung der Gewichte und Längen der Fische in der Stichprobe soll also die entsprechende Verteilung der Fische in der Population approximieren. Seien g^* und l^* die Durchschnittswerte in der mit Zurücklegen gezogenen "Bootstrap"-Stichprobe. Wir wiederholen diesen Vorgang sehr oft, z.B. 1000×, und schauen uns jeweils die Differenz $\hat{g} - g^*$ bzw. $\hat{l} - l^*$ zwischen den Schätzwerten und den Boostrap-Schätzwerten an. Wir gehen dann davon aus, dass die Differenz $g - \hat{g}$ bzw. $l - \hat{l}$ ähnliche Werte annimmt.

Bootstrap in der Phylogenie-Schätzung anzuwenden, um die Ergebnisse eines Baum-Schätzverfahrens zu berurteilen, schlägt Felsenstein (1985) vor. Um einen simulierten Datensatz zu bauen, ziehe man aus den n homologen Sequenzen der Länge m jeweils m Positionen (also Spalten des Alignments) mit Zurücklegen. Auf den so zusammengesetzten Datensatz wende man das Baumschätzverfahren an. Das

wiederhole man ganz oft, z.B. 1000 mal, und beschrifte jede Kante des aus den Orginaldaten geschätzten Baumes mit der relativen Anzahl der Bootstrap-Bäume, die diese Kante auch haben (im Sinne eines Splits der Blatt-Menge).

Die Interpretation der Felsenstein'schen Bootstrap-Werte ist nicht ganz unumstritten. Hillis und Bull (1993) stellen fest, dass die Bootstrap-Bäume schlechter sind als der aus den Orginaldaten geschätzte Baum und ziehen daraus den Schluss, dass die Bootstrap-Werte kleiner sind als die Wahrscheinlichkeiten, dass die Kanten stimmen. Efron, Halloran, Holmes (1996) stellen klar, dass in der Tat beim Bootstrappen im allgemeinen $\mathrm{Ws}(\theta^* = \theta) < \mathrm{Ws}(\widehat{\theta} = \theta)$ gilt, dass aber $\mathrm{Ws}(\theta^* = \widehat{\theta}) \approx \mathrm{Ws}(\widehat{\theta} = \theta)$ gilt, dass also der Unterschied zwischen den gebootstrapten und dem ursprünglich geschätzten Baum Aufschluß gibt über den Unterschied zwischen dem geschätzten und dem wahren Baum. Sie schlagen vor, abermals Bootstrap anzuwenden, um die Genauigkeit der ersten Bootstrap-Methode zu schätzen.

4.7 Modelle der Sequenz-Evolution mit Insertionen und Deletionen

4.7.1 TKF91

Den vielen Substitutionsmodellen stehen nur wenige Sequenz-Evolutionsmodelle gegenüber, die Insertionen und Deletionen berücksichtigen. Das erste solche Modell ist das TKF91-Modell von Thorne, Kishino und Felsenstein (1991). Wie wir sehen werden, führt es zu einer Hidden-Markov-Struktur auf den Alignments und ist daher mit effizienten Alignment-Algorithmen verträglich.

Jede Position wird deletiert mit einer Rate μ und zwischen je 2 Positionen und am Anfang und Ende der Sequenz werden mit Rate λ einzelne neue Positionen eingefügt. Existierende Positionen evolvieren unabhängig voneinander gemäß einem der einschlägigen Substitutionsmodelle, etwa HKY. Liegt eine Sequenz der Länge N vor, so kann jede der N Positionen gelöscht werden und es kann an $N + 1$ Stellen etwas eingefügt werden. Damit daraus kein Ungleichgewicht erwächst, was bedeuten würde, dass die Sequenzen immer länger würden, ist $\lambda < \mu$ vorausgesetzt. Wir erhalten damit ein reversibles Gleichgewicht auf der Sequenzlänge: Sei π_n die Wahrscheinlichkeit, dass die Sequenz die Länge n hat. Damit $(\pi_n)_n$ reversibles Gleichgewicht ist, muss $\pi_n \cdot \lambda \cdot (n+1) = \pi_{n+1} \cdot \mu \cdot (n+1)$ gelten. Löst man diese Rekursion, so folgt wegen $\sum_{n \geq 0} \pi_n = 1$:

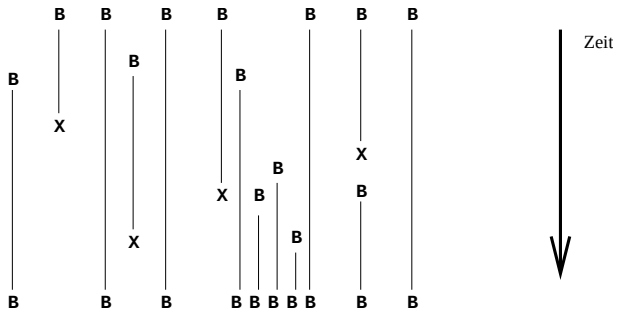
$$\pi_n = \left(\frac{\lambda}{\mu}\right)^n \cdot \left(1 - \frac{\lambda}{\mu}\right)$$

Die Sequenzlänge im Gleichgewicht ist also geometrisch verteilt und ihr Erwartungswert ist $\sum_{n=1}^{\infty} n \cdot \left(\frac{\lambda}{\mu}\right)^n \left(1 - \frac{\lambda}{\mu}\right) = \frac{\lambda}{\mu - \lambda}$. Im TKF91-Modell wird davon ausgegangen, dass die Länge der Ursequenz zufällig ist und gemäß der stationären Verteilung verteilt.

Wegen der Reversibilität des TKF91-Modells können wir bei zwei zu alignierenden Sequenzen oBdA davon ausgehen, dass eine der beiden die Urseqenz der anderen ist.

Wie kommt man nun von dem Insertions-Deletions-Prozess zum Alignment? Angenommen man weiß genau, wie der Prozess abgelaufen ist. Kann man dann ein Alignment hinschreiben?

Betrachten wir folgende Realisierung des Prozesses als Beispiel (B steht für eine vorhandene oder eingefügte "Base", ein Kreuzchen für eine Deletion):



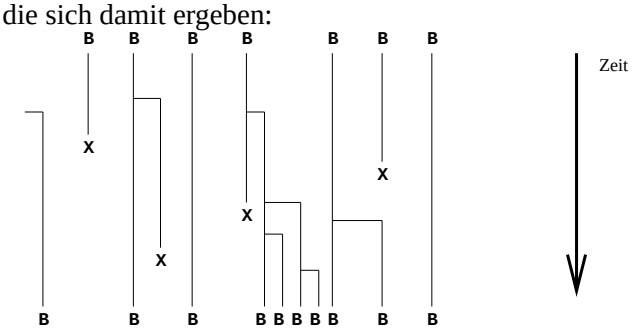
Das dazugehörige Alignment wird so notiert:

```
_BBBB____B_BB
B_BB_BBBBBB_B
```

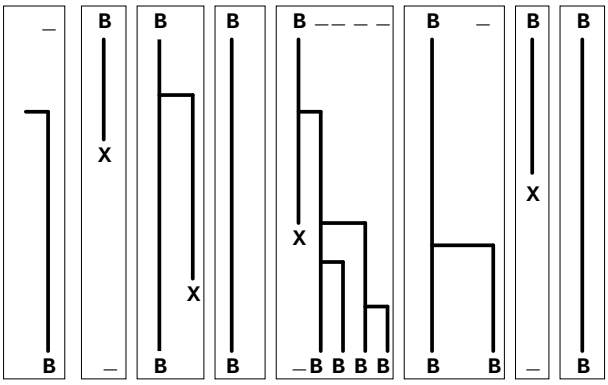
Eigentlich könnte man es genauso gut auch so notieren:

```
_BBBB____BB_B
B_BB_BBBBB_BB
```

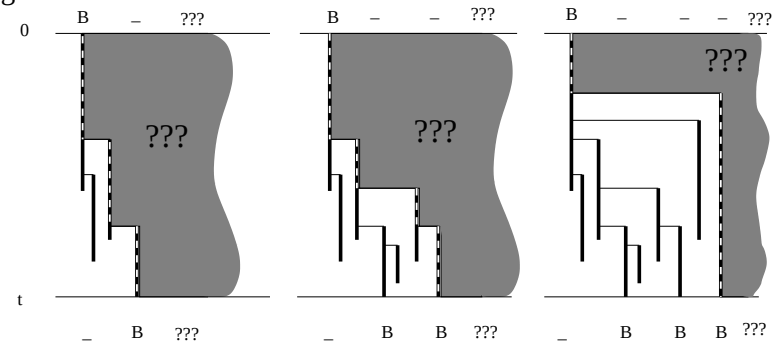
Um jedoch die Notation (man könnte auch sagen die Abbildung von den Evolutionsgeschichten auf die Alignments) eindeutig zu definieren, wird im TKF91-Modell folgende Konvention verwendet: jede eingefügte Base wird als “Kind” des linken Nachbarn aufgefasst und mölichst nahe an diesem notiert. (Insertionen am Sequenzanfang sind Kinder einer unsichtbaren unsterblichen Position.) Folgendes Bild zeigt die Stammbäume, die sich damit ergeben:



Diese Konvention klärt nicht nur die Abbildung von den Evolutionsgeschichten auf die Alignments und stattet damit die Alignments mit Wahrscheinlichkeiten aus, sie sorgt auch dafür, dass die Alignments eine HMM-Struktur erhalten. Zunächst können wir das Alignment in Blöcke aufteilen. An jeder Position der Ursequenz beginnt ein solcher und umfasst deren “Nachkommenschaft”:



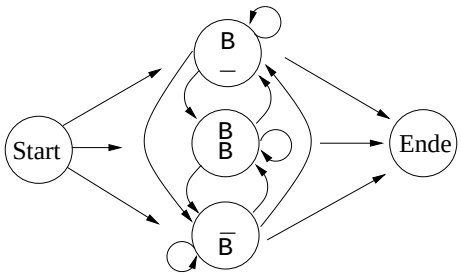
Die Blöcke sind voneinander stochstisch unabhängig. Wenn ein neuer Block beginnt, stellt sich zunächst die Frage, ob die Urposition überlebt. Das tut sie offensichtlich mit Wahrscheinlichkeit $e^{-\mu t}$. Wenn die Urposition mindestens einen überlebenden Nachkommen hat (einschließlich ihrer selbst), ist die Anzahl der überlebenden Nachkommen geometrisch verteilt. Das kann man sich folgendermaßen klar machen: Dass es mindestens einen Nachkommen gibt, ist äquivalent dazu, dass es einen gibt, der unter allen Nachkommen am weitesten links steht. Wo könnten dann weitere Nachkommen herauswachsen? Nur aus der Abstammungslinie des am weitesten links stehenden, also der gestrichelten Linie im linken Teil des folgenden Bildes:



Setzt man die Teilstücke dieser Abstammungslinie zusammen, so erhält man ein Intevall $[0, t]$. Falls daraus wieder mindestens ein überlebender Nachkomme entspringt, so betrachte man wieder den linkesten davon und mache sich klar, dass alles weitere nur aus dessen Abstammungslinie herauswachsen kann (der gestrichelten Linie im mittleren Teil des obigen Bildes). Auch diese Abstammungslinie entspricht dem Intervall $[0, t]$. Wenn daraus wieder mindestens ein Nachkomme überlebt, ist man wieder in der selben Situation, und so weiter. Daher hängt die Anzahl der noch dazukommenden überlebenden Nachkommen, also der Gap-Extensions in der Ursequenz, nicht davon ab, wieviele überlebende Nachkommen bzw. Gap-Positionen in der Ursequenz wir schon gesehen haben. Das ist aber gerade die Gedächtnislosigkeit, die die geometrische Verteilung eindeutig charakterisiert. Auch Markoff-Ketten bleiben jeweils geometrisch lange in einem Zustand. Zusammen mit der stochastischen Unabhängigkeit der einzelnen Blöcke des TKF91-Modells folgt die Markoff-Eigenschaft der Alignments des TKF91-Modells.

Das heißt: Falls man ein Alignment gemäß dem TKF91-Modell simulieren will, braucht man nicht den Insertions-Deletions-Prozess im Rechner ablaufen zu lassen, um dann daraus das Alignment zu extrahieren. Man kann stattdessen auch gleich eine Markoff-Kette auf dem abgebildeten Graphen um-

herspringen lassen. Die Übergangswahrscheinlichkeiten hängen dabei von λ, μ und t ab. Wenn es nur um zwei Sequenzen geht, kann man die Mutationsraten stets so skalieren, dass $t = 1$ gilt. Entscheidender als die Möglichkeit, Sequenzpaare effizient zu simulieren, ist natürlich, dass wir für das Alignment-Problem und alles, was damit zusammenhängt, die in Abschnitt 2 beschriebenen effizienten HMM-Algorithmen verwenden können. So können wir etwa auch Maximum-Likelihood-Methoden einsetzen, um aus einem Paar unalignierter Sequenzen λ, μ und Substitutionsraten zu schätzen.



Details der Berechnung der Übergangswahrscheinlichkeiten für das TKF91-Modell findet man in Thorne, Kishino, Felsenstein (1991) und Metzler, Fleißner, Wakolbinger, von Haeseler (2001). Wir beschränken uns hier auf eine vereinfachte Variante des TKF91-Modells, bei der $\lambda = \mu$ gilt. Damit gibt es dann auf $\mathbb{N}_0$ keine Gleichgewichtsverteilung mehr für die Sequenzlänge. Theoretisch operieren die Mutationen auf unendlich langen Sequenzen, und wir gehen davon aus, dass die zu alignierenden Sequenzen aus diesen langen Sequenzen herausgeschnitten wurden und dass die Positionen, die links und rechts der zu alignierenden Sequenzen liegen, jeweils in allen Sequenzen homolog sind. Daher benötigen wir in diesem Modell keinen Start- und keinen End-Zustand. Stattdessen gehen wir davon aus, dass wir jeweils in einem Zustand $\begin{smallmatrix}\mathsf{B}\\ \mathsf{B}\end{smallmatrix}$, der nichts in die beobachteten Daten emittiert, beginnen und enden. Dass dieses Modell in einem geeigneten Sinne reversibel ist, wird in Metzler, Fleißner, Wakolbinger, von Haeseler (2005) gezeigt.

Ein Vorteil des vereinfachten Modells ist, dass sich die Übergangswahrscheinlichkeiten leichter berechnen lassen. Wir gehen im Folgenden von $t = 1$ aus. Mit den selben Argumenten wie oben gilt auch im vereinfachten TKF91-Modell, dass die Anzahl X der zur Zeit $t = 1$ überlebenden Nachkommen (ggf. einschließlich der Position selbst) einer Ursequenz-Position geometrisch verteilt ist, sofern darauf bedingt wird, dass mindestens ein Nachkomme überlebt. Also gibt es eine Wahrscheinlichkeit p mit $\Pr(X = k \mid X > 0) = (1-p)^{k-1} \cdot p$ und $\mathbb{E}(X \mid X > 0) = 1/p$. Die Rate für das Hinzukommen von Positionen diesselbe ist wie die Rate für das Verschwinden von Positionen, bleibt die erwartete Anzahl an Nachkommen konstant. Also gilt $\mathbb{E}X = 1$. Wenn eine Position überlebt oder mindestens einen überlebenden Nachkommen hat, so ist der Erwartungswert für die Anzahl an Linien, die bis zum Zeitpunkt $t = 1$ aus der entsprechenden Abstammungslinie nach rechts herauswachsen, gleich der Rate λ. Da jede der herauswachsenden Linien im Mittel einen überlebenden Nachkommen zur Zeit 1 hat, folgt $\mathbb{E}(X \mid X > 0) = 1 + \lambda$ und somit $p = 1/(1+\lambda)$. Aus

$$\begin{aligned} 1 &= \mathbb{E}X = \Pr(X > 0) \cdot \mathbb{E}(X|X > 0) + \Pr(X = 0) \cdot \mathbb{E}(X|X = 0) \\ &= \Pr(X > 0) \cdot (1 + \lambda) + \Pr(X = 0) \cdot 0 \end{aligned}$$

folgt $\Pr(X > 0) = 1/(\lambda + 1)$.

Für die Berechnung der Übergangswahrscheinlichkeiten zwischen den Zuständen $\begin{smallmatrix}\mathsf{B}\\ \bar{\mathsf{B}}\end{smallmatrix}$, $\begin{smallmatrix}\mathsf{B}\\ \_\end{smallmatrix}$ und $\begin{smallmatrix}\mathsf{B}\\ \mathsf{B}\end{smallmatrix}$ folgen nun zwei Beispiele. Bedingt darauf, dass der aktuelle Zustand $\begin{smallmatrix}\mathsf{B}\\ \mathsf{B}\end{smallmatrix}$ ist, ist die Wahrscheinlichkeit, dass der

nächste Zustand $\bar{\mathtt{B}}$ ist, also dass die Position einen weiteren Nachkommen hat,

$$P(\mathtt{B} \to \bar{\mathtt{B}}) = 1 - p = 1 - \frac{1}{1+\lambda} = \frac{\lambda}{1+\lambda}.$$

Das Ereignis $X > 0$ kann dadurch zustandekommen, dass die Urposition überlebt oder dadurch, dass sie stribt und einen überlebenden Nachkommen hat. Also gilt $e^{-\lambda} + (1 - e^{-\lambda}) \cdot \Pr(\mathtt{B} \to \bar{\mathtt{B}}) = \Pr(X > 0) = 1/(\lambda + 1)$ und somit

$$\Pr(\mathtt{B} \to \bar{\mathtt{B}}) = \frac{1 - e^{-\lambda}(1+\lambda)}{(1 - e^{-\lambda})(1+\lambda)}.$$

Die Berechnung der übrigen Übergangswahrscheinlichkeiten wird den Leserinnen und Lesern als Übung überlassen.

4.7.2 TKF92

Eine Schwäche des TKF91-Modells ist, dass nur Insertionen und Deletionen einzelner Positionen zugelassen sind. Das erscheint biologisch unrealistisch. Eine Weiterentwicklung ist das TKF92-Modell (Thorne, Kishino, Felsenstein, 1992). Es erlaubt auch Insertionen und Deletionen längerer Fragmente und stattet ebenso wie das TKF91-Modell die Alignments mit einer HMM-Struktur aus. Der Unterschied zu TKF91 ist lediglich, dass nicht einzelne Positionen eingefügt und herausgenommen werden, sondern Fragmente, die geometrisch lang sind. Die Wahrscheinlichkeit eines Fragments, Länge n zu haben, ist also $r^{n-1}(1-r)$, wobei r ein zusätzlicher Modellparameter ist. Die Fragmentierung bleibt erhalten, d. h. Fragmente können nur einzeln und als Ganzes deletiert werden und neue Fragmente können nur zwischen anderen Fragmenten eingefügt werden. Die Ursequenz ist bereits auf unbeobachtbare Art zufällig in Fragmente geometrisch verteilter Länge unterteilt. Wenn ein Fragment der Ursequenz mindestens einen überlebenden Nachkommen zur Zeit t hat, dann ist die Anzahl der Fragmente, die vom Urfragment abstammen und zur Zeit t noch leben, geometrisch verteilt, denn bei TKF92 verhalten sich die Fragmente so wie die Positionen bei TKF91. Jedes Fragment enthält geometrisch viele Positionen (mit einem anderen Parameter als die geometrisch verteilte Anzahl der Fragmente). Damit ist die Anzahl der überlebenden Positionen in dem Block wieder geometrisch verteilt. Allgemein gilt nämlich, dass geometrisch lange Summen identisch unabhängig geometrisch verteilter Zufallsvariablen wieder geometrisch verteilt sind. Davon kann man sich leicht überzeugen: Sei dazu $(1-p)^{n-1}$ die Wahrscheinlichkeit eines Fragments, Länge n zu haben und sei $R(p)$ eine Prozedur, die nach Eingabe von p mit Wahrscheinlichkeit p eine 1 und sonst eine 0 liefert. Dann können wir die Gesamtlänge N der Überlebenden Fragmente, bedingt auf $N > 1$, mit folgender Prozedur simulieren:

$N := 0$
`Repeat`
    `Repeat`
        $N := N + 1$
    `until` $R(p) = 1$
`until` $R(1-r) = 1$

`Ausgabe` N

Dieses Programm ist offensichtlich äquivalent zu folgendem:

$N := 0$

`Repeat`

$N := N + 1$

`until` $R(1-r) \cdot R(p) = 1$

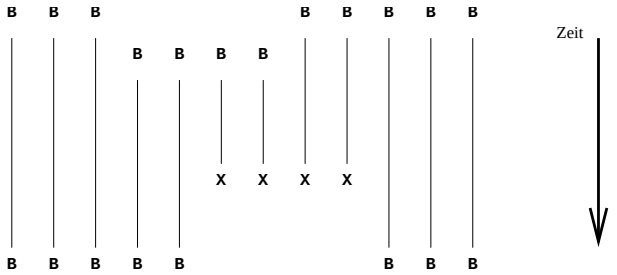
`Ausgabe` N

Damit ist klar, dass N mit Wahrscheinlichkeit $(1 - p(1-r))^{k-1} \cdot p(1-r)$ den Wert k annimmt. Also ist N ebenfalls geometrisch verteilt. Man bleibt also auch beim TKF92-Modell geometrisch lange im Zustand $\binom{\mathtt{B}}{\bar{\mathtt{B}}}$. Entsprechendes folgt auch für die Zustände $\binom{\mathtt{B}}{\mathtt{B}}$ und $\binom{\mathtt{B}}{\_}$ aus der geometrischen Länge der Fragmente. Zusammen mit der stochastischen Unabhängigkeit der Blöcke, die wie bei TKF91 auch bei TKF92 gilt, folgt die Markoff-Eigenschaft der Alignments im TKF92-Modell. Dasselbe gilt auch in einer vereinfachten Variante des TKF92-Modells mit $\lambda = \mu$ (siehe Metzler 2003). Wir berechnen als Beispiel eine Übergangswahrscheinlichkeit. Sei dazu γ die erwartete Länge eines Fragments. Dann ist $\Pr(\binom{\mathtt{B}}{\mathtt{B}} \to \binom{\mathtt{B}}{\_})$ im vereinfachten TKF92-Modell das Produkt der Wahrscheinlichkeit γ^{-1}, dass das Fragment mit dem homologen Positionenpaar endet, der Wahrscheinlichkeit, dass dieses Fragment keinen weiteren überlebenden Nachkommen hat, und der Wahrscheinlichkeit, dass das nachfolgende Fragment deletiert wurde. Das sich die Fragmente wie Einzelpositionen im vereinfachten TKF91-Modell verhalten, folgt

$$\Pr(\tbinom{\mathtt{B}}{\mathtt{B}} \to \tbinom{\mathtt{B}}{\_}) = \frac{1 - e^{-\lambda}}{\gamma \cdot (\lambda + 1)}.$$

Alle weiteren Übergangswahrscheinlichkeiten des vereinfachten TKF92-Modells werden den Leserinnen und Lesern zur Übung überlassen.

Man beachte, dass diese Eigenschaft dadurch erkauft ist, dass einiges im TKF92-Modell verboten ist, z.B. folgendes:



Ob solche Einschränkungen beim Alignieren eines Sequenzpaares von praktischer Bedeutung sind, wird in Metzler (2003) untersucht.

4.8 Phylogenieschätzung aus unalignierten Daten

Multiples Alignment funktioniert, wie wir früher schon einmal diskutiert haben, gut, wenn man die Phylogenie der Sequenzen kennt. Will man aber aus den Sequenzen letztlich einen Baum schätzen, steckt

man in einem Dilemma: Nimmt man zum Alignieren einen vorläufigen Baum an, wie etwa bei ClustalW, kann das Alignment von diesem Baum beeinflusst sein, so dass eine darauf basierende Phylogenieschätzung in Richtung des vorläufigen Baumes verzerrt ist.

4.8.1 Paarweise ML-Parameterschätzung

Einen Ausweg aus diesem Dilemma, der auf dem TKF92-Modell beruht, schlagen Thorne und Kishino (1992) vor: Gegeben seien n Sequenzen $s_1, s_2, \ldots, s_n$. Für jedes Paar (s_i, s_j) sei t_{ij} die zeitliche Distanz im wahren Baum. Thorne und Kishino schätzen zunächst für jedes Sequenzenpaar die Parameter $t_{ij}\lambda$, $t_{ij}\mu$, r und $t_{ij}S$, wobei S die Substitutionsmatrix ist, und konstruieren dann aus den Distanzen, gewichtet mit den Streuungen der Schätzungen, den Baum mittels eines Distanz-Verfahrens.

Im Detail: Mit einer Kombination aus dem Viterbi-Algorithmus und numerischer Optimierung werden zunächst die ML-Schätzer $\widehat{St_{ij}}$, $\widehat{\mu t_{ij}}$ und $\widehat{r_{ij}}$ für jedes Sequenzpaar berechnet. Es sei $\rho_{ij} := \widehat{st_{ij}}/\widehat{\mu t_{ij}}$, wobei die mittlere Substitutionsrate ist (gemittelt wird über alle Aminosäuren gemäß ihren Häufigkeiten). Die wahren Werte für ρ und r sind im ganzen Baum gleich, aber die Schätzungen aus den einzelnen Sequenzpaaren werden sich zunächst unterscheiden. Als Schätzer ρ^* und r^* für diese Größen werden daher die Mediane von $\{\rho_{ij} \mid ij\}$ bzw. $\{\widehat{r_{ij}} \mid ij\}$ verwendet. Anschließend werden ρ^* und r^* festgehalten und in einem zweiten Durchgang werden noch einmal die Werte St_{ij} für alle Sequenzpaare geschätzt. Die Ergebnisse d_{ij} dieses zweiten Durchgangs sind die Distanzen, die zur Baumschätzung verwendet werden.

Für jedes d_{ij} wird die Varianz der Schätzung approximiert. Dies geschieht mit Hilfe der Fisher-Information $I(d_{ij}) = -\mathbb{E}\frac{\partial^2}{\partial d_{ij}^2} L(d_{ij})$, also der erwarteten Krümmung der log-Likellihood an der Maximalstelle. (Die Krümmung ist zufällig, da sie von den Daten abhängt!) Aus der Statistik ist bekannt, dass der Kehrwert der Fisher-Information die Varianz des ML-Schätzers asymptotisch annähert. Thorne und Kishino (1992) schätzen aus dem Verlauf der Likelihood-Kurve der jeweiligen Distanz die Fisher-Information und daraus die Varianz des Schätzers. Sei z_{ij}^2 die so gewonnene Approximation der Varianz der Schätzung d_{ij}. Sei T_{ij} die Distanz von s_i und s_j im Baum T. Mit einem Algorithmus von Fitch und Margoliash (1967) wird dann der Baum gesucht, der die Summe der gewichteten quadrierten Differenzen zwischen den geschätzten Distanzen und denen im Baum

$$\sum_{i<j} \frac{(d_{ij} - T_{ij})^2}{z_{ij}^2}$$

minimiert. Thorne und Kishino schlagen vor, anschließend mit einem "Pseudo-Bootstrap-Verfahren" die Baumschätzung zu beurteilen: Die "Pseudo-Bootstrap-Werte" d_{ij}^* werden dabei aus einer Normalverteilung mit Mittelwert d_{ij} und Varianz z_{ij}^2 gezogen.

Da das Verfahren von Thorne und Kishino auf paarweisen Distanzen basiert, vernachlässigt es einen Teil der in den Sequenzen enthaltenen Information.

4.8.2 Aktuelle Ansätze

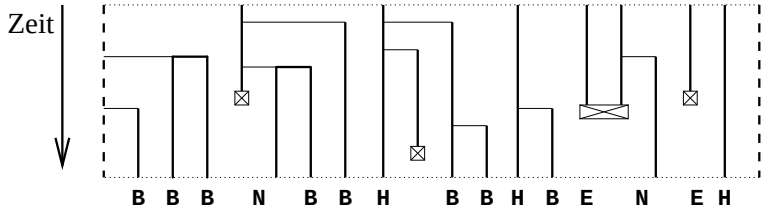
Es wäre wünschenswert, ausgehend von einem Modell wie TKF92 für gegebene Sequenzen simultan das multiple Alignment und den Stammbaum schätzen zu können.

Ein Problem ist bereits: Wie berechnet man die Likelihood eines Baumes für gegebene unalignierte Sequenzen? Die Konvention zur Alignment-Notation der TKF-Modelle kann man so auf Multiple Alignments verallgemeinern, dass eine Multiple-Hidden-Markov-Struktur entsteht. Dazu muss man allerdings im Falle des TKF92-Modells fordern, dass die Fragment-Struktur im ganzen Baum erhalten bleibt, was etwas problematischer ist als wenn es nur um paarweise Alignments geht. Einige Ansätze probieren mehrere Sequenzen für die die inneren Knoten durch. Das Multiple Alignment ergibt sich dann jeweils aus den paarweisen Alignments längs der Kanten. Dynamische Programmierung über alle Kanten und alle denkbaren Alignments der Sequenzen in den Blättern ist im worst case exponentiell in der Anzahl der Blätter. Außerdem sind die Algorithmen sehr unübersichtlich.

Einige Ansätze für simultanes Alignment und Baumschätzen versuchen es mit MCMC-Sampling oder simulated Annealing. Dabei werden Baum-Sampling-Verfahren wie das von Mau, Newton und Larget mit Alignment sampling kombiniert.

Aktuelle Arbeiten, die sich mit dem Problem befassen: Holmes, Bruno (2001), Lunter, Miklós, Song, Hein (2003), Fleißner, Metzler, von Haeseler (2005), Metzler, Fleißner, Wakolbinger, von Haeseler (2005). Wir befassen und nun noch mit einigen Ideen aus diesen Arbeiten.

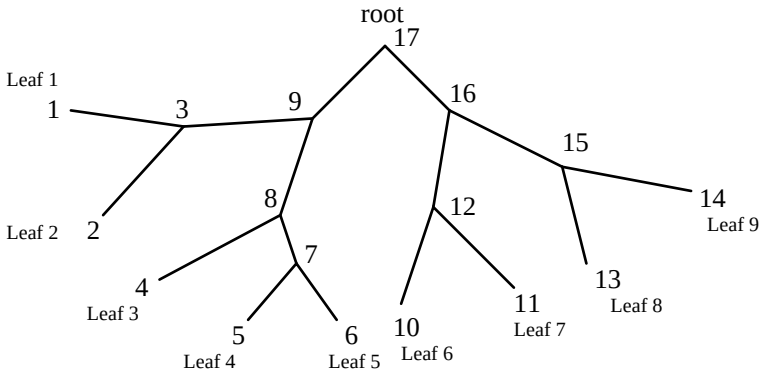
Um die TKF-Modelle auf den Fall multipler Alignments zu verallgemeinern, ist eine von Lunter et al. (2003) vorgeschlagene, kompaktere Notation nützlich, bei der paarweise Alignments (ohne Informationen über die Nukleotid- bzw. Aminosäuretypen) durch jeweils eine Zeile dargestellt wird. Wir schreiben $\begin{smallmatrix}\mathsf{B}\\ \mathsf{B}\end{smallmatrix}$ als `H` (für **h**omologes Paar), $\begin{smallmatrix}\bar{}\\ \mathsf{B}\end{smallmatrix}$ als `B` (für **B**irth, Geburt) und $\begin{smallmatrix}\mathsf{B}\\ \bar{}\end{smallmatrix}$ als `E` (für **E**nde). Für $\begin{smallmatrix}\bar{}\\ \mathsf{B}\end{smallmatrix}\begin{smallmatrix}\mathsf{B}\\ \bar{}\end{smallmatrix}$ schreiben wir nicht `EB` sondern `N` (für **n**icht homolog), denn dadurch werden die Übergangswahrscheinlichkeiten stark vereinfacht, zumindest im TKF91-Modell. Hier ein Beispiel für die Notation:



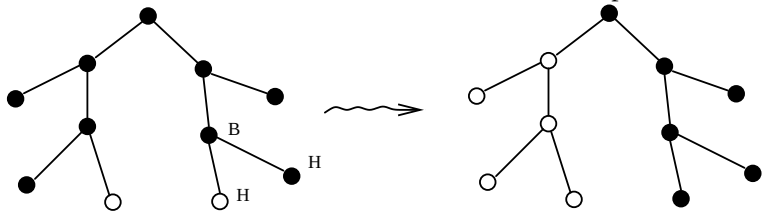
Stammt eine Position i einer Sequenz von einer Position j in der Ursequenz in einer Weise ab, die durch `H` oder `N` dargestellt wird, so bezeichnen wir die Position i als *Erben* der Position j. Ist dann eine Position k in einer weiteren Sequenz ein Erbe von i, so ist k auch Erbe von j. Eine Position, die selbst kein Erbe ist, heißt *Gründer*. Das sind genau die Positionen, die einem `B` entsprechen. Ein *Tihl* (Abk. für tree-indexed heirs-line, baumindizierte Erbenlinie) ist die Menge aller Positionen in Sequenzen in verschiedenen Knoten, die Erben desselben Gründers sind.

Wenn nach zwei homologen Positionen im Stammbaum auf verschiedenen Kanten neue Gründer eingefügt werden, die überlebende Erben in den Blättern des Baumes haben, so muss geklärt werden, welche der Positionen im Alignment zuerst (also weiter links) notiert werden. Es wird also eine Verallge-

meinerung von Thorne, Kishino und Felsensteins Alignmentnotationskonvention benötigt. Dazu werden die Knoten des Baumes in einer Weise durchnummeriert, so dass jeweils der Vater eines Knotens eine größere Nummer bekommt, als der Sohn selbst, also z.B. so:



Im Zweifelsfall wird dann das Tihl zuerst notiert, dessen Gründer am Knoten mit der kleineren Nummer steht. Im Umkehrschluß bedeutet das aber folgendes, wenn man das Alignment von links nach rechts liest: Wird an einem Knoten ein Gründer eingefügt, so kann an einem Knoten, der eine kleiere Nummer hat und bei dem kein Erbe des Gründers steht, im nächsten Schritt kein Gründer eingefügt werden. Dieser hätte nämlich vor dem erstgenannten Gründer notiert werden müssen. In diesem Sinne deaktiviert also ein Tihl solche Knoten. Alle Knoten, an denen Erben des Gründers stehen, werden aktiviert. Tihls können dann nur an aktiven Knoten eingefügt werden. Folgendes Beispiel zeigt die Aktivierung und Deaktivierung von Knoten durch einen Tihl. Aktive Knoten sind schwarz, passive Knoten weiß dargestellt.



Bei der Berechnung der Wahrscheinlichkeit eines Tihls geht als Faktor die Wahrscheinlichkeit ein, dass an der entsprechenden Stelle bei keinem der deaktivierten Tihls ein Gründer eingefügt wird. Beim (evtl. vereinfachten) TKF91-Modell hängt die Wahrscheinlichkeit eines Tihls nur davon ab, wie das Tihl zusammengesetzt ist und welche Knoten gerade aktiv sind. Somit erhalten wir ein multiples Hidden-Markoff-Modell, wobei die versteckte Kette gerade die Abfolge der Mengen aktiver Knoten ist.

Für einen Multiindex $k = (k_1, \ldots, k_m) \in \mathbb{N}^m$, wobei m die Anzahl der Blätter des Baumes ist, sei $P(k)$ die Wahrscheinlichkeit, dass das hier beschriebene multiple HMM für jedes Blatt b_i die dort stehende Sequenz bis zu Position k_i emittiert und dann ohne weitere Emission in den Endzustand springt, bzw. im vereinfachten TKF91-Modell, wo es keinen Endzustand gibt, als nächstes ein Tihl einfügt, das seinen Gründer in der Wurzel des Baumes hat und alle Kanten mit `H` beschriftet. Um eine Rekursion für $P(k)$ zu definieren, die wir dann benutzen können, um mit dynamischer Programmierung die Wahrscheinlichkeit der emittierten Daten zu berechnen, benötigen wir noch einige weitere Bezeichnungen. Für ein Tihl e sei $v_e \in \{0,1\}^n$ ein Vektor, der an i-ter Stelle genau dann eine 1 hat, wenn das Blatt b_i als Gründer oder Erbe zum Tihl e gehört. $p(e)$ sei die Wahrscheinlichkeit des Tihls e, also die Wahrscheinlichkeit, dass eine

Position vor dem Gründerknoten von e inseriert wird und so wie durch e angegeben entlang des Baumes evoluiert. Dass der Gründerknoten aktiv ist, wird dabei vorausgesetzt. $\vartheta(e,k)$ ist die Wahrscheinlichkeit, dass eine Position, die gemäß e evoluiert, in jedem Blatt b_i, auf dem ein Erbe oder Gründer von e liegt, die Aminosäure bzw. das Nukleotid emittiert, das in der betreffenden Eingabesequenz an Stelle k_i steht. $q(e)$ sei die Wahrscheinlichkeit, dass an keinem der Erben-Knoten ein `B` eingefügt wird, wobei wir darauf bedingen, dass alle Erben-Knoten von e aktiv und alle Knoten mit höherer Priorität (also kleinerer Nummer) passiv sind. Mit diesen Bezeichnungen gilt

$$P(k) = \sum_e p(e) \cdot q(e) \cdot P_{\mathcal{R}}(k - v_e) \cdot \vartheta(e,k).$$

Eine Herleitung dieser Gleichung findet man bei Lunter et al. (2003) und Metzler et al. (2005). Ein Problem ist dabei, dass für einige Tihls $v_e = (0,\ldots,0)$ gelten kann. Daher tritt $P(k)$ auch auf der linken Seite der Gleichung auf und wir müssen die Gleichung nach $P(k)$ auflösen, um eine saubere Rekursion zu erhalten:

$$\begin{aligned} P(k) &= \sum_{e\,:\,v_e \neq \vec{0}} p(e) \cdot q(e) \cdot P(k - v_e) \cdot \vartheta(e,k) + \sum_{e\,:\,v_e = \vec{0}} p(e) \cdot q(e) \cdot P(k) \\ \Rightarrow & \\ P(k) &= \frac{\sum_{e\,:\,v_e \neq \vec{0}} p(e) \cdot q(e) \cdot P(k - v_e) \cdot \vartheta(e,k)}{1 - \sum_{e\,:\,v_e = \vec{0}} p(e) \cdot q(e)} \end{aligned}$$

Lunter et al. (2003) verwenden eine etwas andere Rekursion, bei der jeweils mehrere Tihls gemeinsam behandelt werden, wodurch der Rechenaufwand bei der dynamischen Programmierung vermindert wird. Dennoch hängt der Rechenaufwand exponentiell von der Anzahl der Blätter des Baumes ab.

Beim (evtl. vereinfachten) TKF92-Modell sind die Abhängigkeiten komplizierter. Dort ist in den Zuständen der versteckten Kette nicht nur die Information enthalten, welche Knoten aktiv sind, sondern auch für jeden Knoten ob an der linken Nachbarposition eine Deletion oder Insertion erfolgt war. Damit werden nicht nur die Formeln komplizierter, sondern auch die Berechnungen aufwändiger. Praktisch stößt dann die dynamische Programmierung bereits bei vierblättrigen Bäumen an ihre Grenzen.

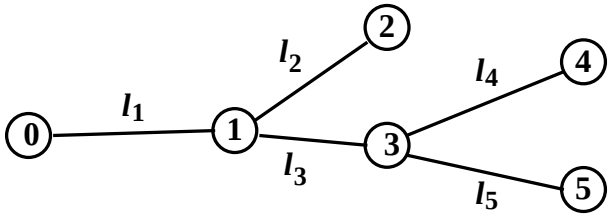
Die Hoffnung liegt nun auf Verfahren, bei denen mit den hier beschriebenen Methoden jeweils die Alignments von Sequenztripeln schrittweise verbessert werden. Außerdem wird versucht, ein Bayessches Phylogenie-Sampling-Verfahren wie das von Mau und Newton so zu ergänzen, dass zusammen mit den Stammbäumen auch die Alignments gemäß der a-posteriori-Verteilung erzeugt werden. Dazu sollen während des Verfahrens lokale Veränderungen an den Alignments von Sequenztripeln vorgenommen werden.

4.9 Modelle für die Evolution quantitativer Merkmale

Manchmal sollen anhand von quatitativen Messungen Stammbäume für Arten rekonstruiert werden. Dabei kann es sich zum Beispiel um Anzahlen von Genen handeln, um Mutationsraten oder um morphologische Daten wie Gewicht, Größe oder bestimmte Körpermaße. Um auch hier Maximum-Likelihood-Methoden anwenden zu können, benötigen wir wieder ein Modell dafür, wie solche Merkmale längs der

Kanten eines Baumes evoluieren. Bei Merkmalen mit kontinuierlichem Wertebereich bietet es sich an, die Evolution des Merkmals längs des Baumes (nach geeigneter Transformation, z.B. mit dem Logarithmus) durch eine *Brownsche Bewegung* zu modellieren. Es handelt sich dabei um einen markoffschen stochastischen Prozess, bei dem über eine Zeitspanne t jeweils zufällige normalverteilte Wertzuwächse mit Mittelwert 0 und Varianz $\sigma^2 = t$ hinzukommen (siehe Anhang A.6).

Als Beispiel betrachten wir folgenden Baum mit dem Knoten 0 als Wurzel.



Für ein Merkmal, das sich gemäß dem Brownsche-Bewegung-Modell entlang des Baumes entwickelt, sei X_i der Wert im Knoten i. Der Prozess starte im Knoten 0 mit einem nicht-zufälligen Wert $X_0 = x_0 \in \mathbb{R}$. Damit gilt $\mathbb{E}X_i = x_0$ für alle i. Die Varianz (vgl. Anhang A.4) entspricht jeweils der Entfernung von der Wurzel, also gilt z.B. $\text{var}(X_5) = l_1 + l_2 + l_5$. Anhand eines Beispiels machen wir uns klar, dass die Kovarianz (vgl. Anhang A.4) zwischen X_i und X_j der Varianz von X_k enspricht, wobei k der jüngste gemeinsame Vorfahr von i und j sei:

$$\begin{aligned}
\text{cov}(X_5, X_4) &= \text{cov}(X_5 - X_3 + X_3, X_4 - X_3 + X_3) \\
&= \text{cov}(X_5 - X_3, X_4 - X_3) + \text{cov}(X_5 - X_3, X_3) \\
&\quad + \text{cov}(X_3, X_4 - X_3) + \text{cov}(X_3, X_3) \\
&= \text{var}(X_3) \;=\; l_1 + l_3
\end{aligned}$$

Wir untersuchen nun die gemeinsame Verteilung der X_i noch etwas genauer. Sei v_i jeweils der Vater des Knotens i. Dann sind die Werte $\left(\frac{X_i - X_{v_i}}{\sqrt{l_i}}\right)_{i=1,\ldots,n}$ (in obigem Beispiel ist $n = 5$, aber was hier erklärt wird, gilt allgemein) voneinander unabhängig und standardnormalverteilt. Sie bilden also einen standardnormalverteilten Zufallsvektor. Außerdem ist die Abbildung

$$Y := \begin{pmatrix} \frac{X_1 - X_{v_1}}{\sqrt{l_1}} \\ \vdots \\ \frac{X_n - X_{v_n}}{\sqrt{l_n}} \end{pmatrix} \mapsto \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = X$$

eine affine Transformation, also mit einer passenden Matrix M und einem Vektor v darstellbar als $Y \mapsto v + MY = X$. Damit ist auch X normalverteilt und seine Verteilung ist durch die Erwartungswerte und Kovarianzen der X_i bestimmt.

Es seien nun die Werte X_i in den Blättern gegeben. Wie schätzen wir den "Startwert" x_0 in der Wurzel und den Baum, oder zumindest dessen Astlängen bei bekannter Topologie? Wir streben auch hier wieder einen ML-Ansatz an. Wir betrachten zunächst ein einfaches

Beispiel: Für $i = 1, \ldots, p$ seien jeweils ein Merkmal gegeben, das ausgehend vom jüngsten gemeinsamen Vorfahren zweier Taxa im unbekannten Wert $x_{0,i}$ gestartet ist und nun bei den beiden Taxa den

Wert x_{1i} bzw. $x_2 i$ hat. Außer den Werten x_{0i1} sind auch die evolutionären Distanzen ℓ_1 und ℓ_2 der beiden Taxa zu ihrem jüngsten gemeinsamen Vorfahren zu schätzen. Wir gehen davon aus, dass bei gegebenem Startwert x_{0i} der Wert nach evolutionärer Distanz ℓ normalverteilt mit Mittelwert x_{0i} und Varianz $\ell\sigma_i^2$ ist. Zu maximieren ist also folgende Likelihood:

$$\begin{aligned} L(x_0, \ell_1, \ell_2) &= \prod_{i=1}^{p} \frac{1}{\sqrt{2\pi\sigma_i^2\ell_1}} \cdot e^{-\frac{(x_{i1}-x_{i0})^2}{2\ell_1\sigma_i^2}} \cdot \frac{1}{\sqrt{2\pi\sigma_i^2\ell_2}} \cdot e^{-\frac{(x_{i2}-x_{i0})^2}{2\ell_2\sigma_i^2}} \\ &= \prod_{i=1}^{p} \frac{1}{2\pi\sigma_i^2\sqrt{\ell_1\ell_2}} \cdot e^{-\frac{1}{2\sigma_i^2}\left(\frac{(x_{1i}-x_{0i})^2}{2\ell_1}+\frac{(x_{2i}-x_{0i})^2}{2\ell_2}\right)} \\ &= \frac{1}{\prod_{i=1}^{p}\sigma_i^2} \cdot \left(\frac{1}{2\pi\sqrt{\ell_1\ell_2}}\right)^p \cdot e^{-\frac{1}{2}\cdot\left(\sum_{i=1}^{p}\frac{1}{\sigma_i^2}\cdot\left(\frac{(x_{1i}-x_{0i})^2}{\ell_1}+\frac{(x_{2i}-x_{0i})^2}{\ell_2}\right)\right)} \end{aligned}$$

Zunächst zur Wahl von x_0. Um die Likelihood zu maximieren, müssen wir im Exponenten die Werte

$$\frac{(x_{1i}-x_{0i})^2}{\ell_1} + \frac{(x_{2i}-x_{0i})^2}{\ell_2}$$

minimieren. Das erreichen wir mit

$$\widehat{x_{0i}} = \frac{\frac{x_{1i}}{\ell_1}+\frac{x_{2i}}{\ell_2}}{\frac{1}{\ell_1}+\frac{1}{\ell_2}}$$

Setzt man das oben ein, so bleiben ℓ_1 und ℓ_2 so zu wählen, dass

$$\left(\frac{1}{\sqrt{\ell_1\ell_2}}\right) \cdot e^{-\sum_{i=1}^{p}\frac{(x_{1i}-x_{2i})^2}{2\sigma_i^2\cdot(\ell_1+\ell_2)}}$$

möglichst groß wird. Dazu lassen wir $\ell_1\ell_2$ möglichst klein und $\ell_1+\ell_2$ möglichst groß werden, indem wir $\ell_1 = 0$ setzen und ℓ_2 gegen ∞ gehen lassen (oder umgekehrt). Hingegen wird die Likelihood besonders **klein** für $\ell_1 = \ell_2$.

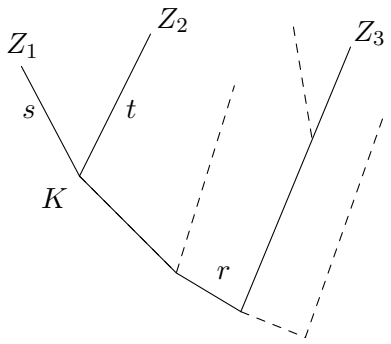
Was ist der Grund für dieses absurde Verhalten? Eine heuristische Erklärung: Irgendwie haben wir eine Variable zuviel in unserem Modell. Diese fällt bei $\ell_1 = 0$ weg, denn dann stimmt x_{0i} mit x_{1i} überein. Es gibt verschiedene Ansätze, dieses Problem, das auch bei Analysen mit mehr als 2 Taxa auftritt, von vornherein zu vermeiden. Eine Möglichkeit ist, zu fordern, dass die Molekulare-Uhr-Eigenschaft gilt, das also alle Blätter des Stammbaums dieselbe Distanz zur Wurzel aufweisen, vgl. Thompson (1975). Die REML-Methode (REduced Maximum-Likelihood) von Felsenstein (siehe z.B. Felsenstein, 2004) basiert hingegen auf der Idee, die Wurzel zu vermeiden und sich nur auf ungewurzelte Bäume zu beziehen. Im obigem Beispiel würde man also statt x_0, ℓ_1 und ℓ_2 nur die Gesamtlänge $\ell_1 + \ell_2$ der beiden Kanten schätzen. In obigem Beispiel ergibt die ML-Schätzung

$$\widehat{\ell_1+\ell_2} = \frac{1}{p}\sum_{i=1}^{p}\left(\frac{x_{1i}-x_{2i}}{\sigma_i}\right)^2.$$

Nun zur Berechnung der Likelihood eines Baumes im Brownsche-Bewegung-Modell, also der Wahrscheinlichkeitsdichte, dass entlang eines Baumes mit gegebener Kantenlängen mehrere Merkmale so mutieren, dass die in den Blättern beobachteten Werte dabei herauskommen. Wir gehen davon aus, dass alle

Merkmale unabhängig voneinander ohne Selektion mutieren. Wir können wieder o.B.d.A. von einem einzelnen Merkmal ausgehen, denn die Gesamtwahrscheinlichkeit ist das Produkt der Wahrscheinlichkeiten jedes einzelnen Merkmals. Sei $Z = (Z_1, \ldots, Z_m)^T$ der Zufallsvektor der Werte, die das Merkmal in den Blättern $b_1, \ldots, b_m$ annimmt. Da es sich bei den Werten Z_i um Summen der normalverteilten Zuwächse auf den jeweiligen Kanten handelt, ist Z ein normalverteilter Zufallsvektor. Gemäß dem REML-Ansatz interessieren uns die Komponenten von Z nur bis auf einen konstanten Summanden. Anders formuliert: uns interessieren nur die Differenzen zwischen den Komponenten Z_i von Z. Die Differenzen bilden ebenfalls aus den oben genannten Gründen einen normalverteilten Zufallsvektor. Um mit dessen Verteilung besser rechnen zu können, etwa um die Likelihood des Baumes zu ermitteln, möchten wir diesen Zufallsvektor auf einen affin transformierten standardnormalverteilten Zufallsvektor Y zurückführen. Auf diese Weise erhalten wir dann auch eine Matrix M und einen Vektor v, so dass $Z = M \cdot Y + v$ gilt. Es gibt viele Tripel (M, Y, v), die diese Gleichung erfüllen. Eine Variante von Felsensteins Pruning-Algorithmus liefert uns eine Lösung mit einem standardnormalverteilten Zufallsvektor Y, der besonders gut zu interpretieren ist.

Wir beginnen mit zwei benachbarten Blättern b_1 und b_2. Seien Z_1 und Z_2 die gemäß den Modellannahmen zufälligen Werte, die das Merkmal in diesen Blättern annimmt. Wie in der nebenstehenden Skizze habe der jüngste gemeinsame Vorfahre k zu den beiden Blättern die Abstände s und t. Der Wert K des Merkmals in diesem Knoten wird nicht beobachtet. Wir weisen ihm aber einen normalverteilten Wert W zu, der von Z_1 und Z_2 abhängt, aber nicht mit der Differenz $Z_1 - Z_2$ korreliert ist. Dazu setzen wir $W := x \cdot Z_1 + (1 - x) \cdot Z_2$ und bestimmen x so, dass $\text{cov}(x \cdot Z_1 + (1 - x) \cdot Z_2, Z_1 - Z_2) = 0$ gilt:



$$
\begin{aligned}
&\text{cov}(x \cdot Z_1 + (1 - x) \cdot Z_2, Z_1 - Z_2) \\
&\quad = \; x \cdot \text{var}(Z_1) - x \cdot \text{cov}(Z_1, Z_2) + (1 - x) \cdot \text{cov}(Z_2, Z_1) - (1 - x) \cdot \text{var}(Z_2) \\
&\quad = \; x \cdot \text{var}(Z_1) - x \cdot \text{var}(K) + (1 - x) \cdot \text{var}(K) - (1 - x) \cdot \text{var}(Z_2) \\
&\quad = \; x \cdot \text{var}(Z_1 - K) - (1 - x) \cdot \text{var}(Z_2 - K) \\
&\quad = \; x \cdot s - (1 - x) \cdot t
\end{aligned}
$$

Wir setzen also $x = \frac{t}{s+t}$ und entsprechend $W := \frac{t}{s+t} \cdot Z_1 + \frac{s}{s+t} \cdot Z_2$. Sei r der Abstand vom Knoten k zu einem Blatt $Z_3 \notin \{Z_1, Z_2\}$. Damit gilt $\text{var}(K - Z_3) = r$. Man sollte W nicht als Schätzung für K auffassen, denn es gilt

$$
\begin{aligned}
\text{var}(W - Z_3) &= \text{var}\left(\frac{t}{t+s} \cdot Z_1 + \frac{s}{t+s} \cdot Z_2 - Z_3\right) \\
&= \text{var}\left(\frac{t}{t+s} \cdot (Z_1 - K) + \frac{s}{t+s} \cdot (Z_2 - K) + K - Z_3\right) \\
&= \left(\frac{t}{s+t}\right)^2 \cdot \text{var}(Z_1 - K) + \left(\frac{s}{s+t}\right)^2 \cdot \text{var}(Z_2 - K) + \text{var}(K - Z_3)
\end{aligned}
$$

$$= \frac{t^2}{(s+t)^2} \cdot s + \frac{s^2}{(s+t)^2} \cdot t + r = \frac{st}{s+t} + r.$$

Also ist $\text{var}(W - Z_3)$ für jedes Blatt $Z_3 \notin \{Z_1, Z_2\}$ um $\frac{st}{s+t}$ größer als $\text{var}(W - Z_3)$. Das entspricht dem Baum, den wir erhalten, wenn wir beim ursprünglichen Baum den zweiblättrigen Teilbaum mit den Blättern Z_1 und Z_2 ersetzen durch einen einzelnen Knoten, der an einer um $\frac{st}{s+t}$ verlängerten Kante hängt und in dem das Merkmal den Wert $\frac{s}{t+s} \cdot Z_1 + \frac{t}{t+s} \cdot Z_2$ annimmt. Die Differenz $Z_1 - Z_2$ ist zu allen in diesem neuen Baum auftretenden Differenzen unkorreliert. In dem neuen Baum wählen wir nun wieder ein Paar direkt benachbarter Blätter und wenden darauf den selben "pruning"-Schritt an, wie zuvor auf den $\{Z_1, Z_2\}$-Teilbaum. Man wiederholt diesen Schritt $m - 2$ mal, so dass am Ende nur noch ein zweiblättriger Baum übrig bleibt. Die Differenz zwischen den Merkmalswerten seiner Blätter und die $m - 2$ Differenzen zwischen den Merkmalswerten der jeweiligen beiden Blätter, die in einem der pruning-Schritte zusammen weggeschnitten bzw. durch ein einzelnes Blatt ersetzt wurden, sind normalverteilt. Außerdem sind diese Differenzen zueinander unkorreliert. Indem wir jede solche Differenz durch ihre Standardabweichung (Wurzel aus der Varianz) teilen, erhalten wir $m-1$ Werte $y_1, \ldots, y_{m-1}$, die zusammen einen standardnormalverteilten Zufallsvektor Y bilden. Da alle Transformationen, mit denen wir Y aus Z berechnet haben, affin sind und die Hintereinanderausführung affiner Transformationen wieder affin ist, gibt es auch eine Matrix M und ein Vektor v, so dass $Z = M \cdot Y + v$ gilt.

Im Prinzip könnte man M und v aus den pruning-Schritten berechnen. Interessanter ist es allerdings, zu überprüfen, ob die Werte $y_1, \ldots, y_{m-1}$ tatsächlich von unabhängigen standardnormalverteilten Zufallsvariablen kommen. Kann dies durch einen statistischen Test verworfen werden, so bedeutet das, dass etwas an den Voraussetzungen nicht stimmt. Um allgemeine Abweichungen von der Nullhypothese zu testen, kann man als Statistik $y_1^2 + \ldots + y_{m-1}^2$ verwenden. Es ist bekannt, dass die quadrierte Länge eines standardnormalverteilten n-dimensionalen Zufallsvektors χ^2-verteilt ist mit n Freiheitsgraden. Wenn man eine bestimmte Alternative zur Nullhypothese testen will, kann man einen Likelihoodquotienten-Test verwenden oder sich einfach auf den entsprechenden Teil der y_i beschränken. Wenn man damit unter Beachtung der Problematik des multiplen Testens zum Beispiel zeigen kann, dass einer der Werte y_i signifikant zu groß oder klein ist, so kann das daran liegen, dass auf dem entsprechenden Teilbaum Selektionsdruck auf das Merkmal wirkte. Hier erweist es sich als Vorteil, dass Z nicht einfach irgendwie in einen standardnormalverteilten Vektor transformiert wurde, sondern so, dass seine einzelnen Komponenten mit Teilbäumen das Baumes assoziiert sind.

Auch wenn man zwischen zwei oder mehr möglichen Stammbäumen entscheiden will, kann man die Merkmalsvektoren wie oben beschrieben jeweils für jeden der beiden Bäume transformieren. Die Dichte der $m-1$-dimensionalen Normalverteilung an der durch den transformierten Vektor gegebenen Stelle kann man dann als Likelihood des jeweiligen Baumes auffassen. Werden mehrere voneinander unabhängig neutral evoluierende Merkmale betrachtet, entspricht die Likelihood dem Produkt der Dichten.

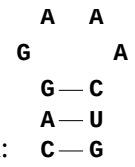
Fundamentalkritik Analysen, die auf phänotypischen Merkmalen beruhen, sind häufig problematisch, denn es ist kaum auszuschließen, dass die Merkmale unter Selektionsdruck stehen oder dass die verschiedenen Merkmale miteinander korreliert evoluieren. Zum Teil kann man solche Korrelatio-

nen oder die auf die Merkmale wirkende Selektion in die Modelle einbeziehen. Man muss dann allerdings viel Vorwissen in die Modelle einbringen, denn es erscheint als ziemlich aussichtslos, sowohl die Korrelations- als auch die Selektionsparameter aus den Daten zu schätzen.

5 RNA-Strukturanalyse mit stochastischen kontextfreien Grammatiken

Gegeben sei eine RNA-Sequenz und gesucht sei ihre Sekundärstruktur: Welche Positionen bilden miteinander Stems, welche gehören zu Loops, etc.

Beispiel: CAGGAAACUG könnte so aussehen:



Gesucht sind also "komplementär-palindromartige" Abhängigkeiten längs der Sequenz. Können wir diese Abhängigkeiten mit (versteckten) Markoff-Ketten modellieren? Nein, denn (H)MMs sind das stochastische Analogon zu regulären Grammatiken. Bei regulären Grammatiken sind Regeln der Formen $A \to aB$ und $A \to a$ für nichtterminale $A, B \in \mathcal{N}$ und terminale $a \in \mathcal{T}$ erlaubt. Bei **stochastischen regulären Grammatiken** können solche Regeln nicht nur erlaubt oder verboten sein, sondern haben eine gewisse Wahrscheinlichkeit $P(A \to aB)$. Dabei gilt:

$$\forall_{A \in \mathcal{N}} \sum_{B \in \mathcal{N}, a \in \mathcal{T}} P(A \to aB) = 1$$

Beginnend mit einem nichtterminalen Start-Symbol S werden diese Regeln gemäß ihren Wahrscheinlichkeiten auf das jeweilige nichtterminale Symbol angewandt. So entsteht letztlich eine Folge von terminalen Symbolen. Hidden-Markoff-Modelle können wir als stochastische reguläre Grammatiken auffassen. Die Menge der nichtterminalen Symbole $\mathcal{N}$ entspricht dabei der versteckten Zustandsmenge $\mathcal{Z}$ und die Menge der terminalen Symbole $\mathcal{T}$ entspricht dem Alphabet der Emissionen $\mathcal{A}$. Um für eine HMM die Übergangswahrscheinlichkeiten der entsprechenden Grammatik zu definieren, setzen wir $P(A \to aB) := P_{AB} \cdot e_A(a)$.

Mit regulären Grammatiken kann man bekanntlich keine Palindrom-Sprachen erzeugen, also werden HMMs bzw. stochastische reguläre Grammatiken wohl auch nicht ausreichen, um die Abhängigkeiten in RNA-Sequenzen zu modellieren. Also versuchen wir's mal mit dem stochastischen Analogon der nächsten Stufe in Chomsky's Hierachie der Transformationsgrammatiken, also mit **stochastischen kontextfreien Grammatiken** (SCFG). Bei kontextfreien Grammatiken sind Regeln der Form $A \to \alpha$ zugelassen, wobei $A \in \mathcal{N}$ und α eine beliebige Folge von Elementen aus $\mathcal{N} \cup \mathcal{T}$ sein darf. (Nicht erlaubt sind z.B. Regeln der Form $\alpha A \beta \to \alpha \gamma \beta$; das wäre kontextsensitiv.) Bei Stochastischen Regulären Grammatiken starten wir wieder mit einem $S \in \mathcal{N}$ und wenden jeweils auf alle nichtterminalen Symbole die Regeln mit gegebenen Wahrscheinlichkeiten an, wobei gelten muss:

$$\forall_{A \in \mathcal{N}} \sum_{\alpha \in (\mathcal{N} \cup \mathcal{T})^*} P(A \to \alpha) = 1$$

In der von Knudsen und Hein (1999) vorgeschlagenen Grammatik zur Modellierung der Abhängigkeiten in RNA-Sequenzen sind folgende Übergänge mit positiven Wahrscheinlichkeiten belegt ("|" steht für "oder"):

$$
\begin{aligned}
S &\rightarrow LS|L \\
F &\rightarrow LS|aFa|aFc|aFg|aFu|cFa|cFc|cFg|cFu|gFa|gFc|gFg|gFu|uFa|uFc|uFg|uFu \\
L &\rightarrow a|c|g|u|aFa|aFc|aFg|aFu|cFa|cFc|cFg|cFu|gFa|gFc|gFg|gFu|uFa|uFc|uFg|uFu
\end{aligned}
$$

L kann Positionen in Loops emittieren und F erzeugt Stems. Man beachte, dass in Stems auch Mismatches vorkommen, deswegen würde es nicht genügen, F nur Basenkomplemente emittieren zu lassen. So ist etwa die G-U-Paarung recht häufig in Stems vorzufinden. Natürlich haben Übergänge, bei den komplementäre Basenpaare emittiert werden, eine höhere Wahrscheinlichkeit.

Für die Beschreibung der Algorithmen gehen wir im folgenden davon aus, dass die SCFG in Chomsky-Normalform gegeben ist, dass also nur Regeln der Formen $A \rightarrow BC$ und $A \rightarrow a$ für $A, B, C \in \mathcal{N}$ und $a \in \mathcal{T}$ vorkommen. Man mache sich klar, dass man jede SCFG auf Chomsky-Normalform bringen kann.

Gegeben sei nun eine SCFG mit bekannten Übergangswahrscheinlichkeiten und eine Sequenz $x = (x_1, \ldots, x_L) \in \mathcal{T}^L$. Wie berechnen wir die Wahrscheinlichkeit, dass die SCFG die Sequenz x emittiert? Analog zum Vorwärts-Algorithmus gibt es bei SCFGs den **Inside Algorithmus**. Er berechnet mit dynamischer Programmierung für $1 \leq i \leq j \leq L$ und alle $A \in \mathcal{N}$ die Wahrscheinlichkeit $\alpha(i, j, A)$, dass das Sequenz-Teilstück $x_i, \ldots, x_j$ aus einem A entsteht.

Inside-Algorithmus

Für $i = 1, \ldots, L$ und $A \in \mathcal{N}$:
    $\alpha(i, i, A) := P(A \rightarrow x_i)$
Für $d = 1, \ldots, L-1$:
    Für $i = 1, \ldots, L-d$:
        $j := i + d$
        Für $k = i, \ldots, j-1$ und $A \in \mathcal{N}$:
            $\alpha(i, j, A) := \sum_{B \in \mathcal{N}} \sum_{C \in \mathcal{N}} \sum_{k=i}^{j-1} P(A \rightarrow BC) \cdot \alpha(i, k, B) \cdot \alpha(k+1, j, C)$
Ausgabe $\mathrm{Ws}_\theta(x) = \alpha(1, L, S)$

Um den Expectation-Maximization-Algorithmus auf SCFGs übertragen zu können, benötigen wir noch für $1 \leq i \leq j \leq L$ und jedes $A \in \mathcal{N}$ die Wahrscheinlichkeit $\beta(i, j, A)$, dass aus dem Startsymbol S letzlich eine Sequenz von terminalen Symbolen entsteht, die mit $x_1, \ldots, x_{i-1}$ beginnt, mit $x_{j+1}, \ldots, x_L$ endet und dazwischen etwas stehen hat, das aus einem A entstanden ist. Diese Wahrscheinlichkeiten kann man effizient mit dem Outside-Algorithmus berechnen wenn man zuvor alle Werte $\alpha(i, j, A)$ mit dem Inside-Algorithmus berechnet hat:

Outside-Algorithmus

Für $A \in \mathcal{N}$:
    Falls $A = S$:
        $\beta(1, L, A) := 1$

Sonst:

$\beta(1, L, A) := 0$

Für $d = L-2, \ldots, 0$:

Für $i = 1, \ldots, L-d$:

j:=i+d

Für $A \in \mathcal{N}$:

$$\beta(i,j,A) := \sum_{B,C\in\mathcal{N}} \sum_{k=1}^{i-1} \alpha(k, i-1, C) \cdot \beta(k, j, B) \cdot P(B \to CA) + \sum_{B,C\in\mathcal{N}} \sum_{k=j+1}^{L} \alpha(j+1, k, C) \cdot \beta(i, k, B) \cdot P(B \to AC)$$

Parameterschätzung für SCFGs Gegeben seien eine (oder mehrere) terminale Sequenzen (z.B. in unserer Anwendung RNA-Sequenzen) und eine SCFG. Wie schätzt man dann die Wahrscheinlichkeiten der Regeln? Analog zur Parameterschätzung bei HMMs kann man das mit Expectation-Maximization (EM) machen. Im E-Schritt schätzen wir wie oft die einzelnen nichtterminalen Symbole bei der Entstehung der Sequenzen (im SCFG-Modell) vorkamen. Diese Schätzungen nennen wir $c(A)$ für $A \in \mathcal{N}$). Außerdem schätzen wir wie häufig die einzelnen Regeln verwendet wurden, z.B. sei $c(A \to BC)$ die geschätzte Anzahl der Anwendungen der Regel $A \to BC$ beim Erzeugen der Sequenzen. Diese Schätzungen berechnen wir mit den jeweils aktuellen (und noch zu verfeinernden) Schätzungen für die Wahrscheinlichkeiten der Regeln, und zwar indem wir diese Wahrscheinlichkeiten zunächst im Inside-Outside-Algorithmus verwenden und die darin berechneten Werte folgendermaßen zu Schätzung der entsprechenden Anzahlen verwenden:

$$\begin{aligned}
c(A) &:= \frac{\sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \alpha(i,j,A) \cdot \beta(i,j,A)}{\mathrm{Ws}(x)} \\
c(A \to BC) &:= \frac{\sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} \beta(i,j,A) \cdot P(A \to BC) \cdot \alpha(i,k,B) \cdot \alpha(k+1,j,C)}{\mathrm{Ws}(x)} \\
c(A \to a) &:= \frac{\sum_{\{i|x_i=a\}} \beta(i,i,A) \cdot P(A \to a)}{\mathrm{Ws}(x)}
\end{aligned}$$

Im M-Schritt werden dann die geschätzten Anzahlen $c(\ldots)$ verwendet, um die Schätzungen für die Wahrscheinlichkeiten der Regeln verbessern:

$$\widehat{P}(A \to BC) := \frac{c(A \to BC)}{c(A)} \qquad \widehat{P}(A \to a) := \frac{c(A \to a)}{c(A)}$$

E- und M-Schritt werden dann solange iteriert, bis die Parameterschätzungen zu konvergieren scheinen.

Die wahrscheinlichste Entstehungsgeschichte Nun kommt noch das Analogon zum Viterbi-Algorithmus. Gegeben sei eine terminale Sequenz $x_1, \ldots, x_L$ und eine SCFG mit bekannten Wahrscheinlichkeiten. Mit der stochastischen Variante des Cocke-Younger-Kasami(CYK)-Algorithmus kann man die (genauer: eine) wahrscheinlichste Entstehungsgeschichte der terminalen Sequenz berechnen:

Für $i = 1, \ldots, L$ und $A \in \mathcal{N}$:

$\gamma(i, i, A) := \log P(A \to x_i)$

$\tau(i,i,A) := (0,0,0)$

Für $d = 1,\ldots,L-1$, für $i = 1,\ldots,L-d$, für $A \in \mathcal{N}$:

j:=i+d

$\gamma(i,j,A) := \max_{B,C\in\mathcal{N},k\in i,\ldots,j-1}\{\gamma(i,k,B) + \gamma(k+1,j,C) + \log P(A \to BC)\}$

$\tau(i,j,A) := \arg\max_{B,C\in\mathcal{N},k\in i,\ldots,j-1}\{\gamma(i,k,B) + \gamma(k+1,j,C) + \log P(A \to BC)\}$

CYK-Traceback (Zurückverfolgen den wahrscheinlichsten Geschichte) läßt sich gut mit einem Stack aufschreiben und programmieren:

Push $(1,L,S)$

Wiederhole solange Stack nicht leer:

Pop (i,j,A)

$(B,C,k) := \tau(i,j,A)$

Falls $\tau(i,j,A) = (0,0,0)$:

x_i ist Kind von A

Sonst:

B und C sind Kinder von A

Push (i,k,B)

Push $(k+1,j,C)$

Komplexität Während der Viterbi und der vorwärts-rückwärts-Algorithmus für HMMs einen Zeitbedarf von $O(LM^2)$ hat und $O(LM)$ Speicher braucht, wobei L die Länge der Eingabe und M die Anzahl der Elemente von $\mathcal{Z} = \mathcal{N}$ ist, ist die Rechenzeit des inside-outside und des CYK-Algorithmus für SCFGs offensichtlich $O(L^3M^3)$ und der Speicherbedarf ist $O(L^2M)$.

6 Modelle und Algorithmen der Populationsgenetik

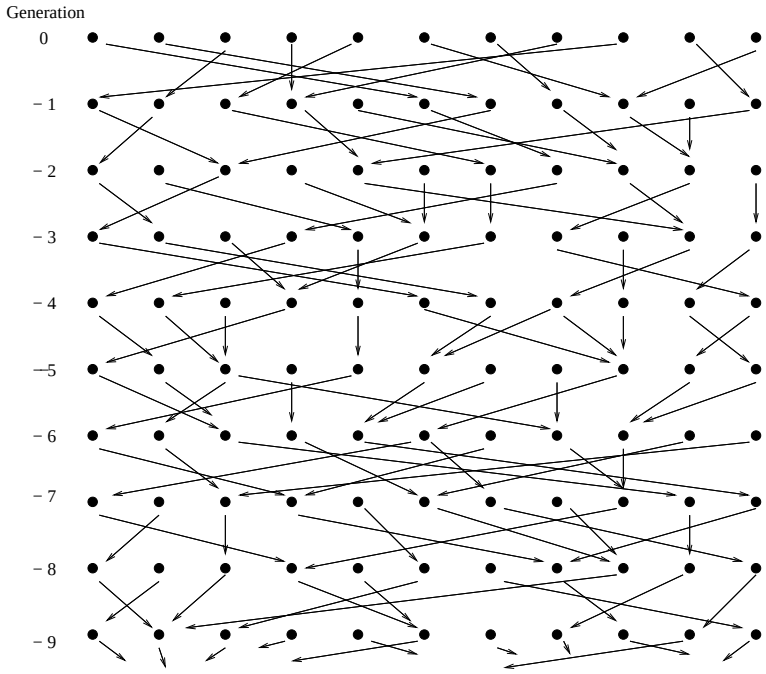
Wir stellen und vor, dass ein Biologe Individuen aus einer Population sammelt und Merkmale dieser Individuen erfasst (z.B. DNA-Sequenzen), die einen gewissen Informationsgehalt über die Verwandtschaft der gesammelten Individuen tragen. Wie können wir diese Informationen nutzen, um etwas über die Geschichte der Population auszusagen, z.B. ob es Selektionsdruck gab, ob die Population in der Vergangenheit gewachsen ist, ob die Population aus Subpopulationen besteht, zwischen denen Migration stattfindet, etc..

Als Literatur zum Thema dieses Abschnitts sind neben den Arbeiten, auf die in den nachfolgenden Unterabschnitten verwiesen wird, die Kapitel 26 bis 28 in Felsenstein (2004) zu empfehlen.

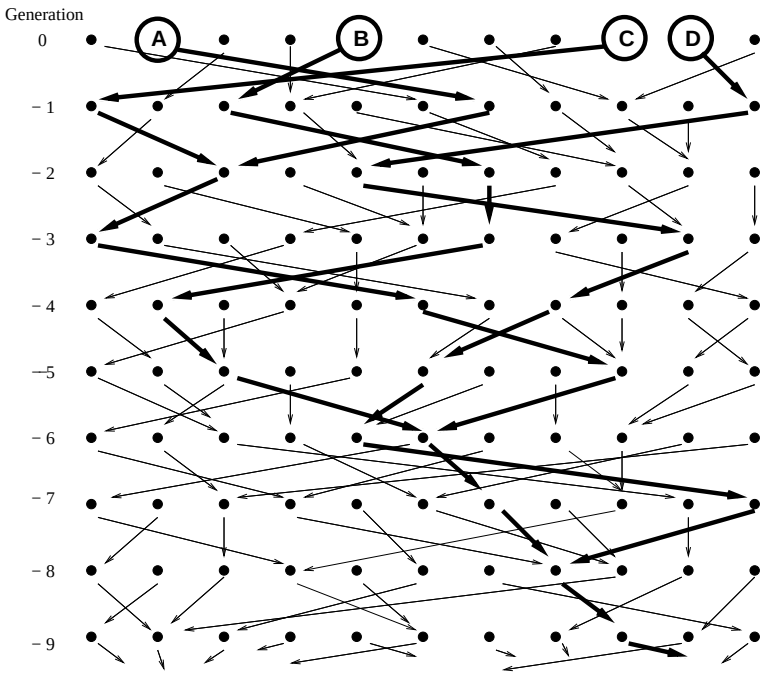
6.1 Das Fisher-Wright-Modell und der Coalescent

Gegeben sei eine Hardy-Weinberg-Population der konstanten Größe N; Sie sei panmiktisch (also ohne räumliche oder sonstige Struktur), haploid und neutral (d.h. es gibt keinen Selektionsdruck). Wir können das Folgende auch anwenden wenn wir es mit einer diploiden Population zu tun haben, indem wir dann jeweils die homologen Chromosomen als haploide Population auffassen.

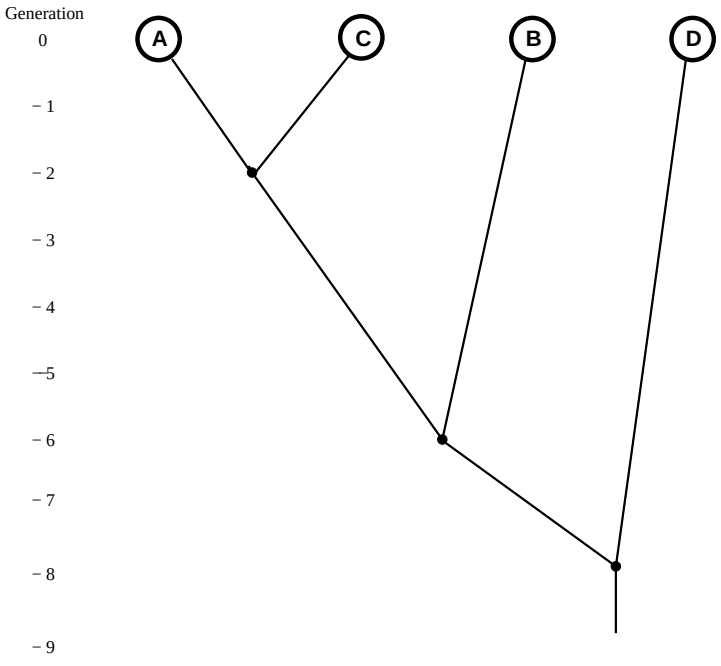
Das Fischer-Wright-Modell geht von einer solchen Population mit diskreten Generationen aus. Wir betrachten die Abstammungslinien eines Individuums. Jedes Individuum sucht sich seinen Vorfahren in der vorherigen Generation rein zufällig:



Damit haben wir die Population modelliert. Um das Zustandekommen der Daten zu modellieren, müssen wir noch den Biologen modellieren. Dazu nehmen wir an, dass er eine Stichprobe von k *rein zufällig* ausgewählten Individuen aus der heutigen Generation entnimmt. Diese haben Abstammungslinien, die wir nicht kennen (aber vielleicht aus z.B. Sequenzdaten zumindest grob schätzen können):



Übersichtlicher wird’s wenn wir die Individuen etwas umsortieren und alle Individuen weglassen, die weder in unserer Stichprobe noch Vorfahren von Individuen unserer Stichprobe sind:



Der Prozess der rückwärts in Zeit verschmelzenden Ahnenlinien heißt **Coalescent**. Seine Astlängen enthalten Informationen, die wir verwenden wollen, um zu untersuchen, ob es in der Entwicklung der Population Effekte nachzuweisen, die von dem einfachen Fischer-Wright-Modell abweichen (z.B. Selektion oder Populationswachstum oder eine räumliche Strukturierung). Ein Problem dabei ist, dass wir

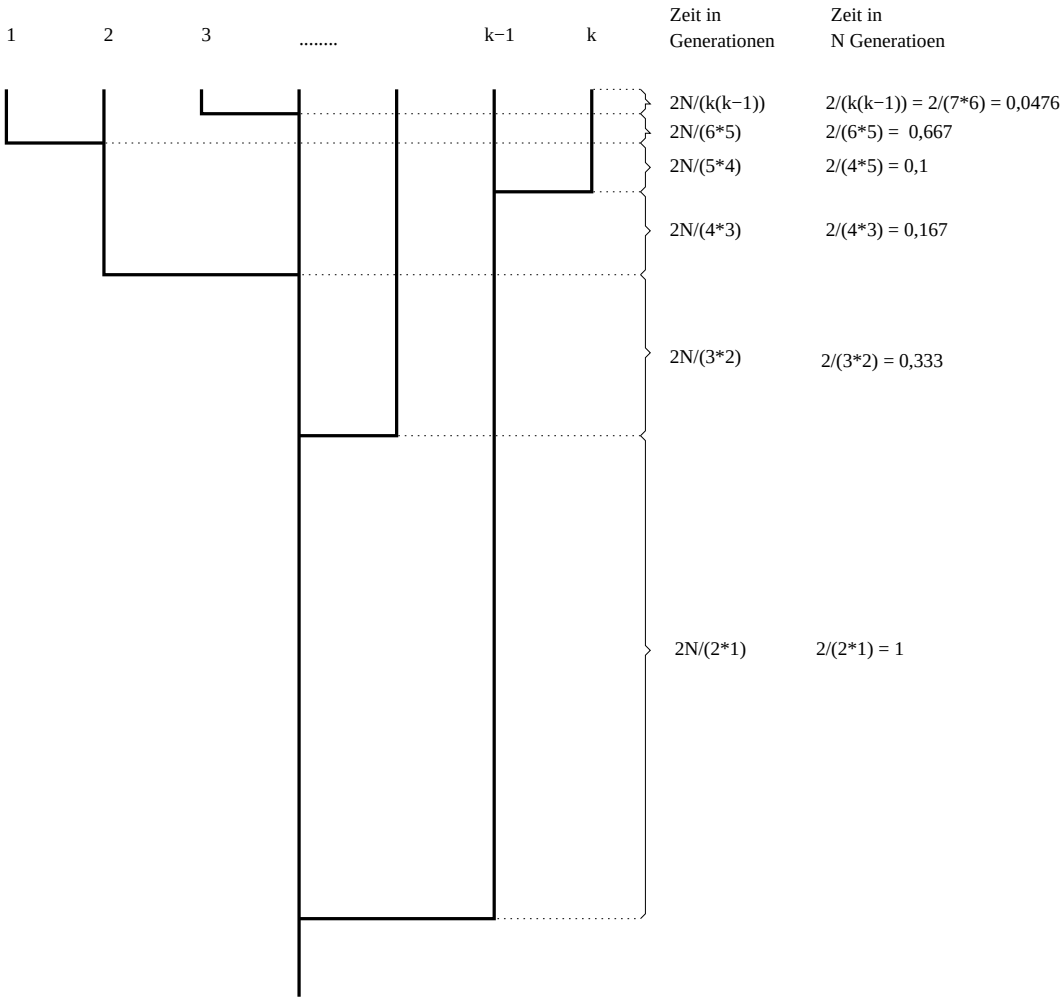
die Astlängen in der Regel nicht sehr genau schätzen können, aber dazu später mehr. Zunächst überlegen wir uns, wie lang die Äste im Fisher-Wright-Modell typischerweise sind, oder um es präziser zu sagen:

Wie ist die Wahrscheinlichkeitsverteilung der Kantenlängen des Coalescent?

Dazu überlegen wir uns erst mal wie lange es (in Verteilung) dauert bis die Ahnenlinien zweier ausgewählter Individuen verschmelzen. Die Wahrscheinlichkeit, dass dies bereits in der Elterngeneration geschieht, ist offensichtlich $1/N$, denn mit dieser W'keit wählt das zweite Individuum den selben Ahnen wie das erste. Wenn die Ahnenlinien in einer Generation nicht verschmelzen (coalieren), dann tun sie das in der nächst-früheren Generation jeweils mit W'keit $1/N$. Also ist $\left(1-\frac{1}{N}\right)^t = \left(\frac{N-1}{N}\right)^t$ die Wahrscheinlichkeit, dass nach dem Zurückgehen um t Generationen immernoch keine Verschmelzung eingetreten ist, d.h. die Zeit bis zur Verschmelzung ist geometrisch verteilt mit Erwartungswert N.

Wir gehen davon aus, dass die Anzahl N der Individuen in der Population sehr groß ist. Zum einen können wir dann die geometrische Verteilung durch eine Exponentialverteilung approximieren (vgl. Abschnitt 4.4.1), zum anderen können wir dann unter der zusätzlichen Annahme, dass die Anzahl k der untersuchten Individuen im Vergleich eher klein ist, die Wahrscheinlichkeit, dass mehr als zwei Ahnenlinien zum selben Zeitpunkt verschmelzen, vernachlässigen. Wir gehen also im Folgenden nicht mehr von diskreten Generationen, sondern von einer kontinuiertlichen Zeitachse in Richtung Vergangenheit aus.

Wenn wir also $m \in \{k, k-1, \ldots, 2, 1\}$ Ahnenlinien vorliegen habe, hat jedes Paar davon die Rate $1/N$ zu verschmelzen. Da es dann insgesamt $\binom{m}{2} = \frac{m\cdot(m-1)}{2}$ solche Paare gibt, tritt die nächste Verschmelzung mit Rate $\frac{m\cdot(m-1)}{2} \cdot \frac{1}{N}$ ein, d.h. die Wartezeit bis zur nächsten Verschmelzung ist $\frac{2N}{m\cdot(m-1)}$. Hier taucht Bemerkenswerterweise die Populationsgröße N als konstanter Faktor auf. Da es eigentlich egal ist, in welchen Einheiten wir die Zeit messen, machen wir es uns einfach, indem wir die Zeit in Einheiten von je N Generationen messen und uns so des Faktors N entledigen. Der Prozess des auf dieser Zeitskala gemessenen Verschmelzens von Ahnenlinien im Limes unendlich großer Populationen heißt nach seinem Entdecker **Kingman-Coalescent** (Kingman, 1982a, 1982b). Eine "durchschnittliche" Genealogie von $k = 7$ Ahnenlinien gemäß der Coalescent-Verteilung sieht dann also so aus (dass die Zeitintervalle genau den Erwartungswerten entsprechen ist natürlich eigentlich völlig untypisch) :



Mit der N-Generationen-Zeitskalierung verschmelzen die letzten beiden Linien (wie jedes Paar von Linien) in Erwartung in 1 Zeiteinheit. Demgegenüber verschmelzen alle k Linien in Erwartung in Zeit $\sum_{m=2}^{k} \frac{2}{m \cdot (m-1)}$. Im Limes $k \to \infty$ (also selbst wenn der fleißige Biologie-Doktorand unendlich viele Individuen gesammelt hat!) hat die Zeit bis alle Linien verschmolzen sind endliche Erwartung

$$\sum_{m=2}^{\infty} \frac{2}{m \cdot (m-1)} \qquad = \qquad 2.$$

Das heißt also: Egal wieviel Linien verschmelzen müssen, dauert das Verschmelzen der letzten beiden Linien im Mittel mindestens genauso lange wie das Verschmelzen aller anderen Linien.

6.2 Der Coalescent mit Mutationen

Wir gehen hier davon aus, dass die Daten, die Informationen über die Genealogie der gesammelten Individuen geben, DNA- oder Protein-Sequenzen sind. Jede Ahnenlinie ist in jeder Generation mit Wahrscheinlichkeit μ von einer Mutation betroffen. Wie im Infinite-Sites-Modell vernachlässigen wir in diesem Unterabschnitt Rückmutationen, nehmen also an, dass alle Mutationen, die die Ahnen der gesammelten seit dem jüngsten gemeinsamen Vorfahren erlitten haben, in den Daten sichtbar sind. Um die

Mutatiosrate auf die N-Generationen-Zeitskala zu bringen, setzen wir $\theta := 2N\mu$. Damit ist dann $\theta/2$ die Rate, mit der Mutationen in die Linien des Kingman-Coalescent eingestreut werden, d.h. auf jeder Ahnenlinie ist die Wartezeit bis zur nächsten Mutation exponentialverteilt mit Erwartungswert $2/\theta$.

Eine Schätzer für θ Wenn wir zwei Individuen aus der Population auswählen und die Mutationen zählen, die zwischen den beiden stattgefunden haben, so kommen wir auf eine zufällige Zahl mit Erwartungswert θ, denn die Zeit bis die Linien zum jüngsten gemensamen Vorfahren verschmelzen hat (in der N-Generationen-Zeitskala) Erwartungswert 1 und in einer Zeiteinheit sind auf jeder der beiden Ahnenlinien jeweils $\theta/2$ Mutationen zu erwarten. Wenn wir also n Unterschiede bei in den beiden Sequenzen sehen, wäre n eine naheliegende Schätzung für θ. Wenn wir nicht nur 2 sondern $k > 2$ Individuen vergleichen, führen wir diese Schätzung für jede der $\binom{k}{2}$ Paare von Individuen durch. Dann kann der Mittelwert Δ dieser Schätzungen als Schätzer für θ ausgegeben werden.

Noch ein Schätzer für θ Im Infinite-Sites-Modell gehen wir davon aus, dass die Sequenzen sehr lang und Mutationen an jeder einzelnen Position sehr unwahrscheinlich sind, und dass somit die Wahrscheinlichkeit, dass eine Position seit dem jüngsten gemeinsamen Vorfahren aller k gesammelten Individuen, von mehr als einer Mutation betroffen war, zu vernachlässigen ist. Ist m die Anzahl der segregierenden Positionen in den Sequenzen, so gehen wir jetzt mal davon aus, dass es in den Ahnenlinien unserer Individuen auch nur m Mutationen seit dem jüngsten gemeinsamen Vorfahren gegeben hat. Das vergleichen wir mit der erwarteten Gesamtlänge aller Linien des Coalescents. Die erwartete Zeit der Existenz von k Linien ist $2/(k\cdot(k-1))$, dann existieren $k-1$ für eine erwartete Zeit von $2/((k-1)\cdot(k-2))$ Einheiten und so weiter.

Die erwartete Gesamtlänge aller Linien beträgt also

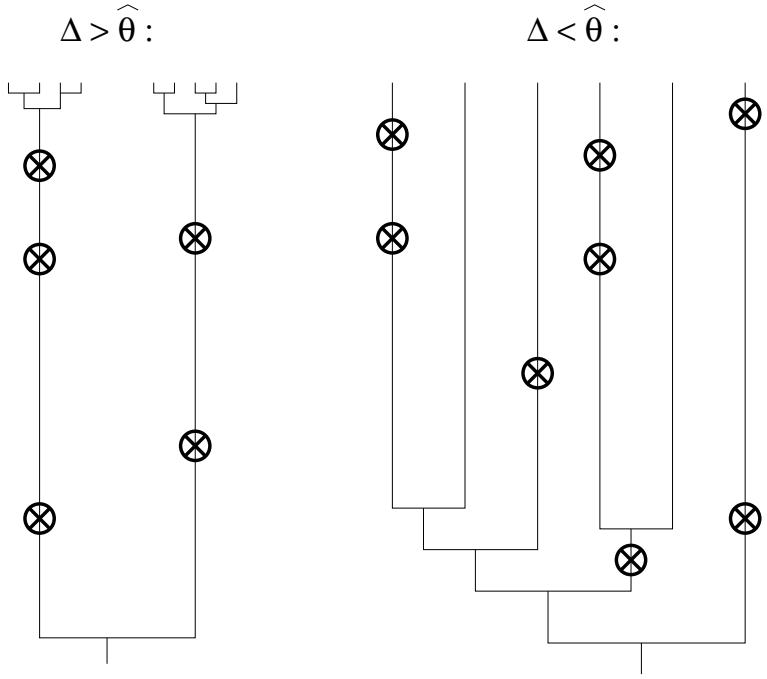
$$k\cdot\frac{2}{k\cdot(k-1)}+(k-1)\cdot\frac{2}{(k-1)\cdot(k-2)}+\ldots+2\cdot\frac{2}{2\cdot 1} \qquad = \qquad 2\cdot\sum_{i=1}^{k-1}1/i$$

Da auf jede Längeneinheit der Linien in Erwartung $\theta/2$ Mutationen entfallen, erscheint es naheliegend, θ zu schätzen durch

$$\widehat{\theta}=\frac{m}{\sum_{i=1}^{k-1}1/i}$$

Was ist wenn Δ und $\widehat{\theta}$ sehr unterschiedlich sind? Bei $\widehat{\theta}$ werden alle Mutationen gleich gewichtet in die Berechnung einbezogen. Man mache sich klar, dass hingegen bei Δ solche Mutationen, die innere Kanten betreffen (also solche, die nahe beim jüngsten gemeinsamen Vorfahren liegen), stärker gewichtet werden als Mutationen, die nahe der Blätter auftreten. Wenn sich Δ und $\widehat{\theta}$ signifikant unterscheiden, dann bedeutet das also, dass die inneren Kanten entweder untypisch lang (im Fall $\Delta > \widehat{\theta}$) oder untypisch kurz (im Fall $\Delta < \widehat{\theta}$) im Vergleich zu den äußeren Kanten sind. Der erste Fall ist z.B. durch eine räumliche Strukturierung der Populatioen zu erklären, der zweite dadurch, dass die Population gewachsen ist oder

eine Anpassung in Folge von Selektion stattgefunden hat. (Wieso solche Umstände die Form der Genealogie auf diese Weise beeinflussen, sollte man sich klar machen!) Die beiden Fälle sind in folgender Abbildung dargestellt, in der die Mutationen durch $\otimes$ symbolisiert werden:



Es stellt sich die Frage wie die Signifikanz einer Abweichung zwischen Δ und $\widehat{\theta}$ zu quantifizieren ist. Dazu Berechnet man einen Schätzer $\widehat{\sigma}$ für die Standardabweichung von $\Delta - \widehat{\theta}$, siehe Durrett (2002). Die Teststatistik ist dann Tajimas $D := (\Delta - \widehat{\theta})/\widehat{\sigma}$. Die zur Beurteilung der Signifikanz nötige Wahrscheinlichkeitsverteilung von Tajimas D wurde durch Simulationsstudien untersucht (Tajima, 1989).

6.3 Der Coalescent mit Mutationen und Migration

Gegeben sind s Subpopulationen der Größen $N_1, \ldots, N_s$. Man denke etwa an Vögel, die auf verschiedenen Inseln leben. Die Mutationsrate ist für jedes Individum μ pro Generation. Für $i, j \leq s$ ist m_{ij} die Wahrscheinlichkeit, dass ein Individuum von der i-ten zur j-ten Insel migriert. Wir verwenden hier das **Fisher-Wright-Modell mit Migration**: In jeder Generation leben auf der j-ten Insel N_j Individuen. Jedes dieser Individuen wählt dann eine Herkunftsinsel, und zwar die Insel i mit Wahrscheinlichkeit m_{ij}. Also muss gelten $\sum_i m_{ij} = 1$, und m_{jj} sollte nahe bei 1 liegen. Unter allen Individuen der vorherigen Generation auf der Herkunftsinsel wird dann der Vorfahr rein zufällig ausgewählt.

Analog zum Standard-Fall setzen wir $\theta_i = 2N_i\mu$ und $\gamma_{ij} = 4N_j m_{ij}$. Allerdings ist hier die Populationsgröße als Zeitskala ungeeignet, da die Subpopulationen ungleich groß sind. Das bedeutet, dass Verschmelzungen von Ahnenlinien auf verschiedenen Inseln unterschiedlich schnell erfolgen und wir keine Zeitskala finden können, auf der Verschmelzungen immer mit Rate 1 erfolgen. Stattdessen skalieren wir die Zeit so, dass Mutationen mit Rate 1 erfolgen, in dem wir $1/\mu$ Generation als eine Zeiteinheit verwenden. Für die Rate des Verschmelzens von Linien auf Insel i erhalten wir damit $1/\theta_i$ und betreffend

der Zuordnung der Ahnenlinien zu den Inseln verhält es sich ähnlich: Rückwärts in der Zeit betrachtet, springt eine auf Insel i angesiedelte Ahnenlinie zur Insel j mit Rate $M_{ji} := \frac{\gamma_{ji}}{\theta_i} = \frac{4N_i m_{ji}}{\theta_i} = \frac{m_{ji}}{\mu}$.

6.4 Ein Modell mit Selektion

ACHTUNG BAUSTELLE!

Bis zur Fertigstellung dieses Abschnitts siehe Neuhauser (1999). Ein Simulationsprogramm für den beschriebenden Prozess ist bei mir auf Anfrage erhältlich.

6.5 Importance Sampling für Genealogien

Geben sei nun ein Datensatz D von Sequenzen von Individuen einer Population. (Im Fall einer nicht-panmiktischen Population mit Migration sei bekannt aus welchen Subpopulationen die Sequenzen kommen.)

Aus den Daten soll der Mutationsparameter $\Theta := \theta$ geschätzt werden, bzw. im Fall mit Migration der Parametersatz $\Theta := (\theta_i, M_{ij})_{ij}$.

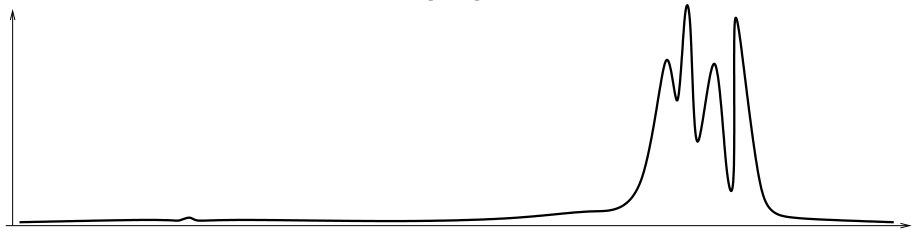
Wir möchten einen Maximum-Likelihood-Ansatz verwenden und möchten daher die Likelihood

$$L_D(\Theta) = \mathrm{Ws}_\Theta(D) = \sum_G \mathrm{Ws}_\Theta(G) \cdot \mathrm{Ws}_\Theta(D \mid G).$$

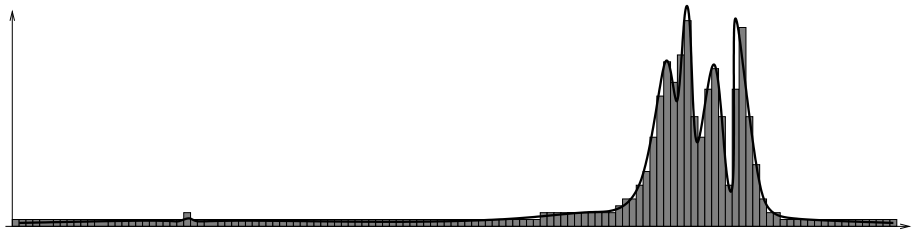
Summiert wird hierbei über alle Genealogien G das Produkt der Wahrscheinlichkeit $\mathrm{Ws}_\Theta(G)$, dass der Coalescent-Prozess unter Annahme der Parameterwerte Θ die Genealogie G hervorbringt und der Wahrscheinlichkeit $\mathrm{Ws}_\Theta(D \mid G)$, dass dann längs G die beobachteten Daten entstehen. **Aber Moment mal!** Da G auch Astlängen kennt, für die jeweils kontinuierlich viele Werte möglich sind, müssen wir hier über überabzählbar viele mögliche G summieren, also eigentlich unter Verwendung der durch den Coalescentprozess auf dem Raum der Genealogien definierten Wahrscheinlichkeitsdichte P_Θ integrieren:

$$L_D(\Theta) = \mathrm{Ws}_\Theta(D) = \int_{\text{alle Genealogien}} \mathrm{Ws}_\Theta(D \mid G)\, P_\Theta(G) dG.$$

Um dieses Integral über den unübersichtlichen Raum der Genealogien zu berechnen, werden verschiedene Varianten einer recht universell einsetzbaren numerischen Integrationsmethode namens **Importance Sampling** verwendet. Um die grundlegende Idee dieser Methode kennen zu lernen, stellen wir uns nun zunächst vor, dass wir eine Funktion $h : [a, b] \to \mathbb{R}$ integrieren wollen:



Eine altbekannte Strategie ist die Approximation von h durch Treppenfunktionen:

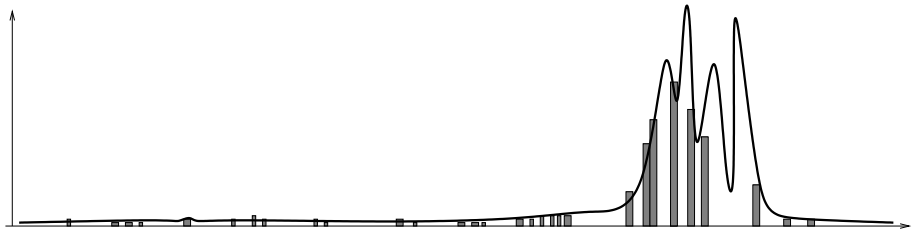


Sind $x_1, \ldots, x_k$ die Mittelpunkte der Zerlegungsintervalle und ist $c = \frac{b-a}{k}$ die Breite der Intervalle, so führt uns das auf die Approximation

$$\int_a^b h(x)\,dx \approx \sum_{i=1}^{k} c \cdot h(x_i) = \frac{b-a}{k} \sum_{i=1}^{k} h(x_i).$$

Im Prinzip funtioniert das auch genauso in höherdimensionalen Räumen. Statt ein Intervall in Teilintervalle zu Zerlegen, muss man dann eben das Volumen, über das Integriert werden soll, z.B. in kleine Würfel zerlegen. c ist dann das Volumen dieser Würfel und $h(x_i)$ der Funktionswert im Mittelpunkt x_i des Würfels. Allerdings braucht man in hochdimensionalen Räumen dann dermaßen viele Würfel, um die zu integrierende Funktion hinreichend genau zu approximieren, dass das Ganze meistens nicht in praktisch vertretbarer Rechenzeit durchführbar ist.

Auch hier hilft aber ein Schlüsselkonzept der moderenen Informatik: die Randomisierung! Statt den ganzen Raum mit Punkten x_i gleichmäßig zu übersähen, an denen wir $h(x_i)$ berechnen, nehmen wir nur Stichproben, ziehen also zufällig nur einige Punkte X_i aus der Menge, über die integriert werden soll. Wenn wir etwa aus dem Intervall $[a, b]$ uniform verteilt ziehen, dann ergibt sich für unser Beispiel folgendes Bild:



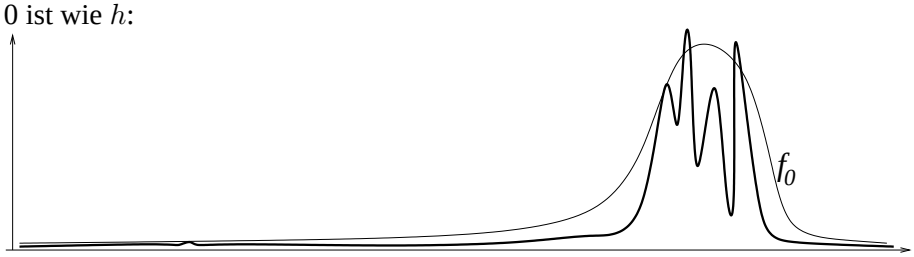
Ist k die Anzahl der zufällig gezogenen Punkte, so könnten wir auch hier folgendermaßen approximieren:

$$\int_a^b h(x)\,dx \approx \frac{b-a}{k} \sum_{i=1}^{k} h(X_i) = \frac{1}{k} \sum_{i=1}^{k} \frac{h(X_i)}{\frac{1}{b-a}}.$$

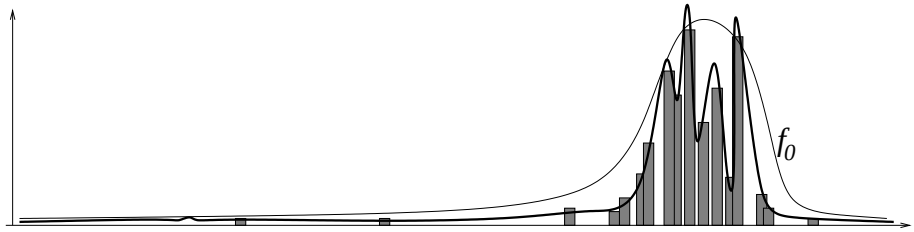
Hier ist schon mal zu beachten, dass $\frac{1}{b-a}$ die (konstante) Dichte der uniformen Verteilung auf dem Intervall $[a, b]$ ist; dazu später mehr.

Wie das vorherige Bild bereits suggeriert, wird eine solche Approximation aber nicht sehr genau sein, wenn in wesentlichen Bereichen nicht genügend Punkte x_i landen. Wir möchten gerne in Bereichen, die viel zum Integral beitragen, etwas genauer messen, und dazu benötigen wir dort mehr Messpunkte. Andererseits möchten wir den Rechenaufwand nicht wesentlich erhöhen, also sparen möchten wir dort Messpunkte einsparen, wo h ohnehin nahe bei 0 liegt. Das können wir natürlich nur erreichen, wenn wir eine gewisse Ahnung davon haben, wo die Funktion h ihre interessanten Bereiche hat, wenn wir

also etwa eine rechnerisch gut zu handhabende Wahrscheinlichkeitsdichte f_0 kennen, die an den selben Stellen nahe 0 ist wie h:



Dann können wir die Messpunkte x_i gemäß der entsprechenden Wahrscheinlichkeitsverteilung ziehen und erhalten damit mehr Information über die interessanten Stellen (und weniger über die, die zum Integral ohnehin wenig beitragen):



Allerdings kann man hier die Punkte X_i nicht mehr als so etwas wie repräsentative Stichprobe auffassen. Punkte, bei denen f_0 (und damit wohl auch $|h|$) groß ist, werden überrepräsentiert. Um das auszugleichen, müssen wir zu einer gewichten Summe übergehen, bei der (wie eigentlich auch schon im Spezialfall der uniformen Verteilung) der Beitrag jedes Messpunktes durch die Wahrscheinlichkeitsdichte an der entsprechenden Stelle geteilt wird. Wir erhalten damit die **Importance-Sampling-Formel**

$$\int h(x)\,dx \approx \frac{1}{k}\sum_{i=1}^{k}\frac{h(X_i)}{f_0(X_i)}.$$

Wie gut die Approximation dann ist, hängt außer von der Anzahl der gezogengen Messpunkte davon ab wie gut f_0 mit const$\cdot|h|$ übereinstimmt (und natürlich auch ein bißchen vom Zufall).

Eine weitere Möglichkeit, sich die Importance-Sampling-Formel (in etwas anderem Kontext) klarzumachen, ist folgende: Wir nehmen an, dass der Erwartungswert $\mathbb{E}_f g(X)$ von $g(X)$ berechnet werden soll, wobei f die Wahrscheinlichkeitsdichte der Verteilung von X ist. Angenommen wir können die Verteilung mit Dichte f nicht simulieren, dafür aber eine (hoffentlich ähnliche) Verteilung mit Dichte f_0. Es gilt

$$\mathbb{E}_f g(X) = \int g(x)\cdot f(x)dx = \int g(x)\cdot\frac{f(x)}{f_0(x)}\cdot f_0(x)dx = \mathbb{E}_{f_0}\left[g(X)\cdot\frac{f(X)}{f_0(X)}\right].$$

Ein naheliegender Schätzer für einen Erwartungswert ist der Mittelwert über ensprechende Stichproben. Sind also $X_1,\ldots,X_k$ gemäß der W'keitsdichte f_0 erzeugt, so schätzen wir

$$\mathbb{E}_f g(X) = \int g(x)\cdot f(x)dx = \mathbb{E}_{f_0}\left[g(X)\cdot\frac{f(X)}{f_0(X)}\right] \approx \frac{1}{k}\sum_{i=1}^{k} g(X_i)\cdot\frac{f(X_i)}{f_0(X_i)}$$

und erhalten so die **Importance-Sampling-Formel** in diesem Kontext. (Den vorherigen Fall erhalten wir mit $h = f\cdot g$.)

Soviel also zur allgemeinen Importance-Sampling-Idee. Mit $h(G) := Ws_\Theta(D \mid G) \cdot P_\Theta(G)$ möchten wir diese nun zur Berechnung von $L_D\Theta = \mathrm{Ws}_\Theta(D) = \int h(G)\, dG$ einsetzen.

Wir werden in den vier folgenden Unterabschnitten mehrere Importance-Sampling-Ansätze zur Berechnung von $L_D\Theta$ kennenlernen.

6.5.1 Beerli, Felsenstein, Kuhner, Yamato

Die Gruppe um Joe Felsenstein kombiniert Importance Sampling mit Markoffketten-Monte-Carlo-Methoden, siehe etwa Kuhner, Yamato und Felsenstein (1995) und Beerli und Felsentein (2001). Um das Θ zu suchen, welches $L_D\Theta$ optimiert, starten wir mit einem Wert Θ_0 und samplen Genealogien $G_1, \ldots, G_k$ gemäß $f_0(G) = P_{\Theta_0}(G \mid D)$. Mit den gesampelten G_i können wir dann für andere Werte $\Theta \approx \Theta_0$ gemäß der Idee des Importance Sampling $L_D\Theta$ recht effizient approximieren und somit ein Θ_1 mit $L_D\Theta_1 > L_D\Theta_0$ suchen. Genauer gesagt sei Θ_1 das Optimum von

$$\Theta \mapsto \frac{1}{k} \sum_{i=1}^{k} \frac{P_\Theta(G_i) \cdot \mathrm{Ws}_\Theta(D \mid G_i)}{P_{\Theta_0}(G_i) \cdot \mathrm{Ws}_{\Theta_0}(D \mid G_i)} \approx \frac{L_D(\Theta)}{L_D(\Theta_0)}$$

innerhalb einer geeignet gewählten Umgebung von Θ_0. Diese Umgebung sollte nicht zu groß sein, da sonst obige Importance-Sampling-Approximation zu ungenau wird.

Wir verwenden dann Θ_1 zur weiteren Optimierung wie zuvor Θ_0 usw., und erhalten schließlich eine Folge $\Theta_0, \Theta_1, \Theta_2, \ldots$, von der wir hoffen, dass sie gegen das globale Maximum von L_D, also den ML-Schätzer für Θ, konvergiert.

Zu klären bleibt wie wir $G_1, \ldots, G_k$ gemäß der Dichte $f_0(G) = P_{\Theta_0}(G \mid D)$ samplen. Dazu verwenden Beerli et al. einen Metropolis-Hastings-Ansatz. Ausgehend von einer Genealogie G wird also jeweils mit W'keit $Q(G \to G')$ die Genalogie vorgeschlagen und akzeptiert mit W'keit

$$\min\left\{1\,,\ \frac{Q(G' \to G)}{Q(G \to G')} \frac{P_{\Theta_0}(G') \cdot \mathrm{Ws}_{\Theta_0}(D \mid G')}{P_{\Theta_0}(G) \cdot \mathrm{Ws}_{\Theta_0}(D \mid G)}\right\}.$$

Auf diese Weise erhalten wir eine Markoffkette, aus der wir auf die übliche Weise samplen.

Nun bleibt noch die Frage nach der Proposal Chain $Q(G \to G')$ offen. Zur Erzeugung von G' wird aus G rein zufällig ein Knoten x (ungleich der Wurzel) gewählt und die Kante zwischen x und seinem Vater gelöscht, wodurch auch der Vaterknoten verschwindet. Die von x in die Vergangenheit zurückreichende Kante lässt man dann an einer zufälligen Stelle in den bestehenden Coalescent hineinwachsen. Die Rate, mit der dies geschieht, beträgt 1 für jede zum jeweiligen Zeitpunkt noch existierende Ahnenlinie, genauso wie es der Fall ist wenn der Coalescent ohne die Bedingung auf Daten simuliert wird.

Die Methoden sind im Software-Paket LAMARC implementiert, das auf der Seite `http://evolution.genetics.washington.edu/lamarc.html` kostenlos erhältlich ist.

6.5.2 Wilson und Balding

Wilson und Balding (1998) schlagen ein sehr ähnliches MCMC-Verfahren vor wie Felsensteins Gruppe. Der Unterschied liegt darin, dass beim MCMC-Sampling der Genealogien $G_1, \ldots, G_k$ für die inneren

Knoten Sequenz-Daten (bzw. Microsatelliten-Längen im Orginal-Paper) geschätzt werden. Bei der Wahl des neuen Vaters für x werden dann solche Ahnenlinien bevorzugt, die zu einem Konten y mit ähnlichem Genotyp wie x führen. Dem neu entstehenden Vaterknoten wird dann ein zufälliger Genotyp zugeordnet, und zwar bedingt auf die Nachbarknoten gemäß dem verwendeten Evolutionsmodell.

Durch die Zuweisungen von Genotypen zu den inneren Knoten können intelligentere Proposals gemacht werden, dafür wird der zu durchwandernde Zustandsraum viel größer, denn er umfasst nun auch für jeden inneren Knoten den Genotyp. Was letztlich besser ist, hängt von den jeweiligen Daten ab. Das Verfahren von Wilson und Balding soll gut für Microsatelliten geeignet sein.

6.5.3 Griffiths und Tavaré

Wie sich herausstellt, können die Astlängen bei der Parameterschätzung ohne Informationsverlust vernachlässigt werden. Griffiths und Tavaré (1994) verwenden in ihrem Importance-Sampling-Ansatz statt der Genealogien (mit Astlängen) lediglich die sog. *Historien*.

Eine Historie ist eine Folge von $H = (H_1, H_2, \ldots, H_\ell)$, die Rückwärts in der Zeit die Ereignisse der Arten {Linie i und Linie j coalieren} und {Mutation auf Linie i} aufzählt, die sich längs des Coalescent ereignet haben. Das erstgenannte Ereignis hat die Wahrescheinlichkeit $1/\left(\binom{k}{2} + k\theta\right)$, das letztgenannte $\theta/\left(\binom{k}{2} + k\theta\right)$, wobei k die Anzahl der (zuvor) existierenden Ahnenlinien ist.

Für die Importance Sampling Methode werden mehrere Historien $H^{(1)}, H^{(2)}, \ldots, H^{(M)}$ erzeugt. Jeweils für die Historie $H^{(i)}$ werden die Ereignisse $H_1^{(i)}, H_2^{(i)}, \ldots$ nach und nach (also ausgehend von den Blättern) erzeugt. Gegeben die Daten in den Blättern sind nicht alle Ereignisse möglich. Linien, deren Genotypen sich unterscheiden, können nicht coalieren. Je nach Mutationsmodell sind auch nicht alle Mutationen möglich. Griffiths und Tavaré ziehen jeweils alle erlaubten Ereignisse in Betracht. Sei $b_{ij}(\theta_0)$ die Wahrscheinlichkeit des j-ten Ereignis $h = H_j^{(i)}$ in $H^{(i)}$ enthaltenen Ereignisses und $(a_{ijk}(\theta_0))_k$ seien die Wahrscheinlichkeiten aller Ereignisse, die auch erlaubt gewesen wären. Griffiths und Tavaré ziehen dann das entsprechende Ereignis h mit Wahrscheinlichkeit $b_{ij}(\theta_0)/\sum_k b_{ijk}(\theta_0)$. Entsprechend ist dann $\prod_j b_{ij}(\theta_0)/\sum_k a_{ijk}(\theta_0)$ die Importance Sampling Wahrscheinlichkeit $Q_{\theta_0}(H^{(i)})$ der gesamten Historie $H^{(i)}$. Mit der Importance Sampling Formel erhalten wir daraus:

$$L_{(D)}\theta \approx \frac{1}{M}\sum_{i=1}^{M} \mathrm{Ws}_\theta(H^{(i)}) \cdot \prod_j \frac{\sum_k a_{ijk}(\theta_0)}{b_{ij}(\theta_0)}$$

Vorteile der Griffiths-Tavaré-Methode liegen darin, dass die gesampelten Historien anders als die Genealogien, die bei den MCMC-Verfahren entstehen, tatsächlich (und nicht nur fast) stochastisch unabhängig sind, und darin, dass das Verfahren in vielen Fällen recht schnell ist, insbesondere wenn für die Daten das infinite-sites-Modelle angenommen werden kann, bei dem keine Rück- und Doppelmutationen erlaubt sind. In anderen Fällen kann das Verfahren jedoch auch sehr langsam sein, wenn nämlich zu viele H_i gesampelt werden, die für die Daten doch recht unwahrscheinlich sind. Verbesserungsmöglichkeiten werden im nächsten Abschnitt behandelt. Das Verfahren ist in der Software GENETREE implementiert, die kostenlos erhältlich ist unter

`http://www.stats.ox.ac.uk/~griff/software.html`

6.5.4 Stephens und Donnelly

Stephens und Donnelly (2000) konnten das Verfahren von Griffiths und Tavaré im Fall des klassischen Coalescent (also ohne Migration oder ähnliches) verbessern. Auch hier werden die Historien gesampelt, allerdings schreiben wir eine Historie $\mathcal{H}$ nun als Folge $(H_{-m}, H_{-(m-1)}, \ldots, H_0)$, wobei nun (anders als im vorherigen Abschnitt) H_{-i} die ungeordnete Liste der Genotypen ist, die in den Ahnenlinien i Ereignisse (Mutatinen und Verschmelzungen) vor der Gegenwart existieren. (”Ungeordnete Liste” bedeutet, dass festgehalten ist, wie oft welcher Genotyp enthalten ist.) H_0 ist also der gesampelte Datensatz.

Wenn (für $i < 0$) H_i in H_{i+1} übergeht, indem eine Linie vom Typ α nach β mutiert, schreiben wir symbolisch $H_{i+1} = H_i - \alpha + \beta$. Falls zwischen H_i und H_i zwei Linien vom Typ α verschmelzen, schreiben wir $H_{i+1} = H_i - \alpha$. Ist $P_{\alpha\beta}$ die Wahrscheinlichkeit, dass ein α, das von einer Mutation betroffen ist, zu einem β wird, und bezeichnen wir mit n_α die Anzahl der Linien in H_{i-1} vom Typ α und mit $n = \sum_\alpha n_\alpha$ die Gesamtzahl der Linien in H_{i-1}, so gilt:

$$\mathrm{Ws}_\theta(H_i \mid H_{i-1}) = \begin{cases} \frac{n_\alpha}{n} \cdot \frac{\theta}{n-1+\theta} P_{\alpha\beta} & \text{falls} \quad H_i = H_{i-1} - \alpha + \beta \\ \frac{n_\alpha}{n} \cdot \frac{n-1}{n-1+\theta} & \text{falls} \quad H_i = H_{i-1} + \alpha \\ 0 & \text{sonst} \end{cases}$$

Sei $\pi_\theta(.)$ die Verteilung des Genotypen-Vektors A_n einer Stichprobe der Größe n aus der Population. Anders als H_0 ist A_n also eine geordnete Liste und es gilt (wobei n_α die Anzahl der α in H_0 bezeichnet):

$$\pi_\theta(A_n \mid \mathcal{H}) = \pi_\theta(A_n \mid H_0) = \begin{cases} \left(\prod_\alpha n_\alpha!\right)/n! & \text{falls } H_0 \text{ mit } A_n \text{ kompatibel} \\ 0 & \text{sonst} \end{cases}$$

Erzeugt man unabhängig Historien $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots, \mathcal{H}^{(m)}$ gemäß einer Proposal Verteilung $Q_{\theta_0}(.)$, so folgt mit der Importance Sampling Formel:

$$L(\theta) \approx \frac{1}{M} \sum_{i=1}^{M} \pi_\theta(A_n \mid \mathcal{H}^{(i)}) \cdot \frac{P_\theta(\mathcal{H}^{(i)})}{Q_{\theta_0}(\mathcal{H}^{(i)})}$$

Mit dieses Bezeichnungen schauen wir uns nochmal das Griffiths-Tavaré-Schema an und stellen fest, dass die dort verwendete Verteilung Q_θ^{GT} erzeugt wird, indem für gegebenes H_0 die Zustände $H_{-1}, H_{-2}, \ldots$ Markoffsch erzeugt werden mit $q_\theta(H_{i-1} \mid H_i) \propto p_\theta(H_i \mid H_{i-1})$.

Sei $\mathcal{M}$ die Klasse der Proposal-Verteilungen, bei denen $H_{-1}, H_{-2}, \ldots$ Markoffsch mit Start in H_0 gesampelt erzeugt werden, mit $\mathrm{supp}\{q_\theta(. \mid H_i)\} := \{H_{i-1} : q_\theta(H_{i-1} \mid H_i) \neq 0\} = \{H_{i-1} : p_\theta(H_i \mid H_{i-1} > 0)\}$.

Optimal wäre $Q_\theta^*(\mathcal{H}) = P_\theta(\mathcal{H} \mid A_n)$, denn damit gilt:

$$\pi_\theta(A_n \mid \mathcal{H}) \frac{P_\theta(\mathcal{H})}{Q_\theta^*(\mathcal{H})} = \frac{P_\theta(\mathcal{H} \cap A_n)}{P_\theta(\mathcal{H} \mid A_n)} = \pi_\theta(A_n) = L(\theta)$$

Also würde die Importance Sampling Formel *exakt* stimmen und wäre nicht nur eine Approximation, und zwar für jedes beliebige $\mathcal{H}$. Man braucht also gar nicht zu sampeln wenn man $\pi_\theta(A_n \mid \mathcal{H})$, $P_\theta(\mathcal{H})$ und $Q_\theta^*(\mathcal{H})$ für irgendein $\mathcal{H}$ berechnen kann. Das folgende Theorem benennt ein q_θ^*, mit dem dies innerhalb der Klasse $\mathcal{M}$ zu realisieren wäre:

Theorem *Sei $\pi_\theta(\alpha \mid A_n) = \frac{\pi_\theta((A_n,\alpha))}{\pi_\theta(A_n)}$ die Wahrscheinlichkeit, dass die $n+1$-te Stichprobe aus der Population vom Typ α ist, bedingt darauf, dass die ersten n Typen die durch A_n gegebenen sind. Die optimale proposal-Verteilung Q^*_θ liegt in $\mathcal{M}$ und ist gegeben durch:*

$$q^*_\theta(H_{i-1} \mid H_i) = \begin{cases} \frac{\theta \cdot n_\alpha}{n\cdot(n-1+\theta)} \frac{\pi(\beta \mid H_i - \alpha)}{\pi(\alpha \mid H_i - \alpha)} P_{\beta\alpha} & \text{für} \quad H_{i-1} = H_i - \alpha + \beta \\ \\ \frac{n_\alpha \cdot (n_\alpha - 1)}{n\cdot(n-1+\theta)} \frac{1}{\pi(\alpha \mid H_i - \alpha)} & \text{für} \quad H_{i-1} = H_i - \alpha \end{cases}$$

Beweis: Wir betrachten den Fall $H_{i-1} = H_i - \alpha + \beta$

Wir bezeichnen den Typ der k-ten Linie zur Zeit t mit $a_k(t)$. Sei $\delta > 0$ und Y_m das Ereignis, dass es in den letzten δ Zeiteinheiten eine Mutation von $a_k(t-\delta) = \beta$ nach $a_k(t) = \alpha$ gab.

Dann gilt:

$$\begin{aligned} \mathrm{Ws}\{Y_m \cap A_k(t-\delta) = (\alpha_1, \ldots, \alpha_{k-1}, \beta) \mid A_k(t) = (\alpha_1, \ldots, \alpha_{k-1}, \alpha)\} & \\ = \quad & \frac{\pi(\alpha_1, \ldots, \alpha_{k-1}, \beta) \cdot \delta \cdot \theta \cdot P_{\beta\alpha}/2}{\pi(\alpha_1, \ldots, \alpha_{k-1}, \alpha)} + o(\delta) \\ = \quad & \delta \cdot \frac{\pi(\beta \mid A_k - \alpha)}{2\pi(\alpha \mid A_k - \alpha)} \cdot P_{\beta\alpha} + o(\delta) \end{aligned}$$

Die Behauptung folgt, wenn wir nun δ gegen 0 gehen lassen, mit n_α multiplizieren (denn statt α_k könnte jedes α betroffen sein und H_i ist ungeordnet) und durch die Gesamtrate teilen.

Die Behauptung für den Fall $H_{i-1} = H_i - \alpha$ folgt analog. □

Aber: Im allgemeinen sind $\pi(\alpha \mid A_n)$ schwer zu berechnen und dann können wir Q^*_θ nicht nutzen.

Ansatz: Wenn Du $\pi(\alpha \mid A_n)$ nicht berechnen kannst, dann approximiere es und setze diese Approximation in die im Theorem gegebenen Formeln ein.

Definition: Sei

$$\widehat{\pi}(\beta \mid A_n) = \sum_{\alpha \in E} \sum_{m=0}^{\infty} \frac{n_\alpha}{n} \left(\frac{\theta}{n+\theta}\right)^m \cdot \frac{n}{n+\theta} (P^m)_{\alpha\beta}.$$

Diese Wahrscheinlichkeitsverteilung kann man folgendermaßen simulieren: Wähle ein rein zufälliges Individuum aus A_n und mutiere es gemäß P geometrisch oft mit Parameter $\frac{\theta}{n+\theta}$.

Eigenschaften von $\widehat{\pi}$:

(a) Bei Eltern-unabhängiger Mutation (d.h. wenn nach einer Mutation der neue Typ stochastisch unabhängig ist vom alten) gilt $\widehat{\pi}(. \mid A_n) = \pi(. \mid A_n)$.

(b) Für reversibles P im Fall $n = 1$ gilt $\widehat{\pi}(. \mid A_n) = \pi(. \mid A_n)$.

(c) Die Verteilung $\widehat{\pi}(. \mid A_n)$ ist von der Form

$$\widehat{\pi}(\beta \mid A_n) = \sum_{\alpha} \frac{n_\alpha}{n} M^{(n)}_{\alpha\beta} \qquad (*)$$

für ein geeignetes $M^{(n)}$. Das heißt, man kann sie simulieren, indem man erst eine Linie rein zufällig wählt und dann aus einer Verteilung zieht, die nur von n und vom Typ der gezogenen Linie abhängt. (Im Fall von $\widehat{\pi}$ gilt speziell $M^{(n)} = (1-\lambda_n)(I-\lambda_n P)^{-1}$ mit $\lambda_n = \frac{\theta}{n+\theta}$.)

(d) $\widehat{\pi}$ ist die einzige Verteilung, die $(*)$ und (b) erfüllt sowie

$$\widehat{\pi}(\beta \mid A_n) = \sum_{\alpha} \widehat{\pi}(\alpha \mid A_n) \cdot \widehat{\pi}(\beta \mid (A_n, \alpha)) \qquad (**)$$

Das heißt: Gegeben die ersten n, ist das $n+1$-te genauso verteilt wie das $n+2$-te.

(e) $\widehat{\pi}(. \mid A_n)$ ist die stationäre Verteilung einer Markoffkette mit Übergangsmatrix

$T_{\alpha\beta} = \frac{\theta}{n+\theta} P_{\alpha\beta} + \frac{n_\alpha}{n+\theta}$

(a), (b) und (d) sagen, dass $\widehat{\pi}$ sinnvolle Eigenschaften hat, die man von einer Approximation für π gerne fordern möchte. (c) sagt, dass $\widehat{\pi}$ effizient eingesetzt werden kann.

Beweise der Eigenschaften

(a) Bei Eltern-unabhängiger Mutation, d.h. $P_{\alpha\beta} = P_\beta$, gilt $P = P^m$ und damit:

$$\pi(\beta \mid A_n) = \frac{n_\beta + \theta P_\beta}{n+\theta} = \widehat{\pi}(\beta \mid A_n)$$

(b) Seien X und Y die Typen der beiden Blätter, R der Typ der Wurzel, m_1 die Zahl der Mutationen zwischen R und X und m_2 die der Mutationen zwischen R und Y. Dann gilt:

$$\begin{aligned} \mathrm{Ws}(Y = \beta \mid R = \gamma) &= (P^{m_2})_{\gamma\beta} \\ \mathrm{Ws}(R = \beta \mid X = \alpha) &= (P^{m_1})_{\alpha\beta} \\ \mathrm{Ws}(Y = \beta \mid X = \alpha) &= (P^{m_1+m_2})_{\alpha\beta} \end{aligned}$$

Die Gesamtzahl der Mutationen zwischen X und Y ist geometrisch verteilt mit Parameter $\frac{\theta}{1+\theta}$

(c)

$$\begin{aligned} \widehat{\pi}(\beta \mid A_n) &= \sum_{\alpha} \sum_{m=0}^{\infty} \frac{n_\alpha}{n} \left(\frac{\theta}{n+\theta}\right)^m \frac{n}{n+\theta} (P^m)_{\alpha\beta} \\ &= \sum_{\alpha} \sum_{m=0}^{\infty} \frac{n_\alpha}{n} (1-\lambda_n) \left[(\lambda_n P)^m\right]_{\alpha\beta} \\ &= \sum_{\alpha} \frac{n_\alpha}{n} (1-\lambda_n) \left[(I-\lambda_n P)^{-1}\right]_{\alpha\beta} \end{aligned}$$

Die letze Gleichung folgt aus der geometrischen Summenformel für Matrizen $\sum_{m=0}^{\infty} M^m = (I-M)^{-1}$.

(d) Sei $\widetilde{\pi}(\beta \mid A_n) = \sum_\alpha \frac{n_\alpha}{n} M^{(n)}_{\alpha\beta}$ für irgendein $M^{(\cdot)}_{\cdot\cdot}$, welches die Eigenschaft $(**)$ erfüllt:

$$\begin{aligned}
\left(\frac{n_\alpha}{n}, \frac{n_\beta}{n}, \ldots, \frac{n_\gamma}{n}\right) \cdot M^{(n)}_{\alpha\beta} &= \widetilde{\pi}(\beta \mid A_n) \\
&= \sum_\alpha \widetilde{\pi}(\alpha \mid A_n)\widetilde{\pi}(\beta \mid (A_n, \alpha)) \\
&= \sum_\alpha \sum_\gamma \frac{n_\gamma}{n} M^{(n)}_{\gamma\alpha} \cdot \sum_\xi \frac{n_\xi + \delta_{\alpha\xi}}{n+1} M^{(n+1)}_{\xi\beta} \\
&= \sum_\gamma \sum_\xi \left(\frac{n_\gamma}{n} \frac{n_\xi}{(n+1)} M^{(n+1)}_{\xi\beta} + \frac{n_\gamma}{n \cdot (n+1)} M^{(n)}_{\gamma\xi} M^{(n+1)}_{\xi\beta} \right) \\
&\qquad \text{(denn } \forall\gamma : \sum_\alpha M^{(n)}_{\gamma\alpha} = 1) \\
&= \frac{1}{n+1} \left[\left(\frac{n_\alpha}{n}, \ldots, \frac{n_\gamma}{n}\right) \cdot \left(n M^{(n+1)} + M^{(n)} M^{(n+1)} \right) \right]_\beta
\end{aligned}$$

Da dies für alle Vektoren $\frac{n_\alpha}{n}, \ldots, \frac{n_\gamma}{n}$ gilt, folgt:

$$(n+1)M^{(n)} = n \cdot M^{(n+1)} + M^{(n)} \cdot M^{(n+1)}$$

Aus dieser Rekursion und dem durch (b) festgelgten"Startwert" $M^{(1)} = (1-\lambda_1)(I - \lambda_1 P)^{-1}$ folgt $M^{(n)} = (1-\lambda_n)(I-\lambda_n P)^{-1}$. Also muss gelten: $\widetilde{\pi} = \widehat{\pi}$.

(e) Für den etwas komplizierteren Beweis von (e) verweisen wir auf Stephens und Donnelly (2000).

□

Wie bereits zuvor angekündigt, definieren wir nun eine Proposal Verteilung Q^{SD}_θ mit $\widehat{q}$ analog zu Q^*_θ und q, indem wir für π die Approximation $\widehat{\pi}$ einsetzen.

Satz *Sei H_i gegeben. Es gilt $\sum_H \widehat{q}_\theta(H \mid H_i) = 1$ und man kann $\widehat{q}_\theta(. \mid H_i)$ folgendermaßen simulieren:*

(a) Wähle ein rein zufälliges Element in H_i; dessen Typ heiße im folgenden α.

(b) Für alle β berechne $\widehat{\pi}(\beta \mid H_i - \alpha)$

(c)

$$H_{i-1} := \begin{cases} H_i - \alpha + \beta & \textit{mit Wahrscheinlichkeit proportional zu} \quad \theta\widehat{\pi}(\beta \mid H_i - \alpha) \cdot P_{\beta\alpha} \\ H_i - \alpha & \textit{mit Wahrscheinlichkeit proportional zu} \quad n_\alpha - 1 \end{cases}$$

Das bedeutet, dass die Zahl der Paare (α, β), für die $\widehat{\pi}(\beta \mid H_i - \alpha)$ berechnet werden muss, klein gehalten werden kann, indem man erst α sampelt und dann zufällig entscheidet ob eine Mutation zu einem Typ β erfolgt oder eine coalition mit einem weiteren α. Da $\widehat{\pi}(\beta \mid H_i - \alpha)$ leicht zu berechnen ist, ist Q^{SD}_θ sehr effizient zu simulieren.

Beweis des Satzes: Die Wahrscheinlichkeit, dass eine Mutation von einer Linie vom Typ α involviert ist, ist:

$$p_m(\alpha) = \frac{1}{n(n-1+\theta)} \sum_{\beta} \frac{\theta}{2} n_\alpha \frac{\widehat{\pi}(\beta \mid H_i - \alpha)}{\widehat{\pi}(\alpha \mid H_i - \alpha)} P_{\beta\alpha}$$

Ist Wahrscheinlichkeit, dass 2 Linien vom Typ α coalieren, ist:

$$p_c(\alpha) = \frac{n_\alpha(n_\alpha - 1)}{n(n-1+\theta)} \cdot \frac{1}{\widehat{\pi}(\alpha \mid H_i - \alpha)}$$

Wie man leicht nachrechnet, gilt $p_m(\alpha) + p_c(\alpha) = 1$. □

Was tun bei Sequenzdaten??? Bei Sequenzdaten der Länge ℓ gäbe es 4^ℓ bzw. 20^ℓ verschiedene mögliche Genotypen $\alpha = (\alpha_1, \ldots, \alpha_\ell)$ und die Übergangsmatrix $(P_{\alpha\beta})_{\alpha\beta}$ wäre dementsprechend sehr groß. Statt der oben beschriebenen Vorgehensweise müssen wir daher ein etwas anderes Verfahren zum Samplen verwenden.

Die Mutationsrate sei $\theta/2$ pro Position, also insgesamt $\ell\theta/2$. Beim Samplen nach $\widehat{\pi}(. \mid A_n)$ ist also eine geometrisch verteilte Anzahl an Mutationen mit Parameter $\frac{\ell\theta}{n+\ell\theta}$ zu ziehen. Sei m das Ergebnis (also die Anzahl). Dann sind die m Mutationen unabhängig und gleichverteilt auf die Positionen zu verteilen. Äquivalent dazu kann man auch eine $\exp(1)$-verteilte Zeit t ziehen und dann für jede Position i eine Poisson-verteilte Mutationen-Anzahl m_i mit Erwartungswert $t\theta/n$ ziehen. Also gilt

$$\widehat{\pi}(\beta \mid A_n) = \sum_{\alpha \in A_n)} \frac{n_\alpha}{n} \int \exp(-t) F^{(\theta,t,n)}_{\alpha_1\beta_1} \cdots F^{(\theta,t,n)}_{\alpha_\ell\beta_\ell} \quad dt$$

mit

$$F^{(\theta,t,n)}_{\alpha_i\beta_i} = \sum_{m=0}^{\infty} \frac{(\theta t/n)^m}{m!} \exp(-\theta t/n)(P^m)_{\alpha_i\beta_i}$$

Das dabei auftretende Integral kann man numerisch mit der Gauß-Quadratur approximieren (siehe Press et al. (1992)) und erhält

$$\widehat{\pi}(\beta \mid A_n) = \sum_{\alpha \in A_n)} \sum_{i=1}^{s} \frac{n_\alpha}{n} w_i F^{(\theta,t_i,n)}_{\alpha_1\beta_1} \cdots F^{(\theta,t_i,n)}_{\alpha_\ell\beta_\ell}$$

für eine geeignete Wahl der Werte s, w_i und t_i. Die $F^{(\theta,t,n)}_{\alpha_i\beta_i} = \sum_{m=0}^{\infty} \ldots$ werden durch endliche Summen approximiert.

A Einige stochastische Grundlagen

Hier sollen einige Fakten aus der elementaren Stochastik aufgeführt werden, die für die Inhalte der Vorlesung grundlegend sind. Eine systematische Einführung kann an dieser Stelle nicht geboten werden. Ein gutes Lehrbuch (eigentlich ein Schulbuch, aber durchaus recht weit führend) ist das Buch von A. Engel (1987). Ebenfalls geeignet ist Krengel (1988). Für einen nächsten Schritt in Richtung statistische Datenanalyse sind Ewens und Grant (2001) und Rice (1995) zu empfehlen. Der Klassiker unter der Stochastik-Lehrbüchern ist Feller (1968).

A.1 Stochastische Unabhängigkeit

Zufallsvariablen $X_1, \ldots, X_n$, die Werte in Mengen $S_1, \ldots, S_n$ annehmen, sind stochastisch unabhängig falls für alle erlaubten $U_i \in S_i$ gilt:

$$\mathrm{Ws}(X_1 \in U_1, X_2 \in U_2, \ldots, X_n \in U_n) = \mathrm{Ws}(X_1 \in U_1) \cdot \mathrm{Ws}(X_2 \in U_2) \cdots \mathrm{Ws}(X_n \in U_n)$$

Wenn Zufallsexperimente völlig unabhängig voneinander durchgeführt werden, sind ihre Ergebnisse stochastisch unabhängig.

A.2 Bedingte Wahrscheinlichkeiten, Bayes-Formel

Seien $A, B \subseteq \Omega$ Ereignisse in einem Ereignisraum Ω. Ist Ω eine endliche Menge gleichwahrscheinlicher Elementarereignisse, so gilt $\mathrm{Ws}(A) = |A|/|\Omega|$. Ansonsten kann man sich $\mathrm{Ws}(A)$ anschaulich als Anteil des Volumens von A am Gesamtvolumen von Ω vorstellen. Wenn wir bereits wissen, dass B eintritt, so entspricht die W'keit, dass auch A eintritt, dem Anteil von $\mathrm{Ws}(A \cap B)$ an $\mathrm{Ws}(B)$: Die *bedingte Wahrscheinlichkeit von A gegeben B* ist

$$\mathrm{Ws}(A|B) = \frac{\mathrm{Ws}(A \cap B)}{\mathrm{Ws}(B)} \qquad \text{(Formel von Bayes).}$$

Sind A und B unabhängige Ereignisse, so folgt $\mathrm{Ws}(A|B) = \mathrm{Ws}(A)$.

Ereignisse können zum Beispiel sein, dass Zufallsvariable gewisse Werte oder Werte in gewissen Mengen annehmen, z.B. ist dann $\mathrm{Ws}(X \in U|Y \in V)$ definiert, wobei X und Y Zufallsvariablen und U und V Teilmengen aus ihren Wertebereichen sind. Es gibt dann auch $\mathrm{Ws}(X \in U|Y)$. Dieser Ausdruck ist eine Zufallsvariable, deren Wert von Y abhängt. Wenn Y den Wert y annimmt, so nimmt $\mathrm{Ws}(X \in U|Y)$ den Wert $\mathrm{Ws}(X \in U|Y = y)$ an.

A.3 Erwartungswert und bedingte Erwartung

Wir nehmen nun an, dass wir es mit Zufallsvariablen zu tun haben, die Werte in $\mathbb{R}$ oder einem Vektorraum über $\mathbb{R}$ annehmen. Wenn X eine solche Zufallsvariable ist, und für X nur abzählbar viele Werte in Frage kommen, können wir den *Erwartungswert* von X so definieren:

$$\mathbb{E}X := \sum_x x \cdot \mathrm{Ws}(X = x)$$

Hat X einen kontinuierlichen Wertebereich müssen wir die Summe durch ein Integral ersetzen:

$$\mathbb{E}X := \int x \cdot \mathsf{Ws}(X \in dx)$$

Eine wichtige Eigenschaft, die das Rechnen mit Erwartungswerten angenehm macht, ist die Linearität: Für Zufallsvariablen X, Y und Zahlen a, b rechnet mal leicht nach, dass folgendes gilt:

$$\mathbb{E}(a \cdot X + b \cdot Y) = a \cdot \mathbb{E}X + b \cdot \mathbb{E}Y$$

Ist V ein zufälliger Vektor und M eine (nicht-zufällige) Matrix passender Größe, so gilt entsprechend $\mathbb{E}(M \cdot V) = M \cdot \mathbb{E}(V)$.

Man beachte jedoch, dass $\mathbb{E}(f(X))$ im allgemeinen (also für nichtlineare Abbildungen f etwas anderes ist als $f(\mathbb{E}X)$. Von der Gültigkeit der Gleichung $\mathbb{E}(XY) = \mathbb{E}X \cdot \mathbb{E}Y$ kann nur ausgegangen werden wenn X und Y stochastisch unabhängig sind.

Die *bedingte Erwartung* von X gegeben ein Ereignis B ist definiert durch:

$$\mathbb{E}(X \mid B) := \sum_x x \cdot \mathsf{Ws}(X = x \mid B) \qquad \text{bzw.} \qquad \mathbb{E}(X \mid B) := \int x \cdot \mathsf{Ws}(X \in dx \mid B)$$

Ist Y eine Zufallsvariable, so ist $\mathbb{E}(X|Y)$ ebenfalls eine Zufallsvariable, und ihr Wert hängt von Y ab:

$$\{Y = y\} \Rightarrow \{\mathbb{E}(X|Y) = \mathbb{E}(X|Y = y)\}$$

Die Linearität gilt auch für die bedingte Erwartung. Wichtig ist auch die folgende Formel, die man leicht nachrechenen kann:

$$\mathbb{E}(\mathbb{E}(X|Y)) = \mathbb{E}X$$

A.4 Varianz und Kovarianz

Sind X und Y reelwertige Zufallsvariablen, so heißt

$$\mathsf{cov}(X, Y) := \mathbb{E}((X - \mathbb{E}X)(Y - \mathbb{E}Y)) = \mathbb{E}(XY) - \mathbb{E}X \cdot \mathbb{E}Y$$

die *Kovarianz* von X und Y. Die *Varianz* von X ist

$$\mathsf{var}(X) = \mathsf{cov}(X, X)$$

und die *Standardabweichung* von X ist SD(X)= $\sqrt{\mathsf{var}(X)}$.

Die Kovarianz ist bilinear. Ist Z eine weitere Zufallsvariable und sind a und b reelle Zahlen, so gilt also

$$\mathsf{cov}(a \cdot X + b \cdot Z, Y) = a \cdot \mathsf{cov}(X, Y) + b \cdot \mathsf{cov}(Z, Y)$$

und

$$\mathsf{cov}(X, a \cdot Y + b \cdot Z) = a \cdot \mathsf{cov}(X, Y) + b \cdot \mathsf{cov}(X, Z).$$

Daraus ergibt sich eine binomische Formel für die Varianz

$$\text{var}(X+Y) = \text{var}(X) + 2 \cdot \text{cov}(X,Y) + \text{var}(Y)$$

sowie

$$\text{var}(a \cdot X) = a^2 \cdot \text{var}(X).$$

Gilt cov(X, Y)=0, so sind X und Y unkorreliert. Man mache sich klar, dass unabhängige Zufallsvariablen immer unkorreliert sind, dass es aber unkorrelierte Zufallsvariablen gibt, die voneinander abhängig sind.

A.5 Die relative Entropie

Sind $p = (p_1, \ldots, p_n)$ und $q = (q_i, \ldots, q_n)$ Verteilungen auf derselben Menge, so heißt $H(p,q) = \sum_i p_i \log \frac{p_i}{q_i}$ (mit $0 \log 0 := 0$) *relative Entropie* von p zu q oder auch *Kullback-Leibler-Information*. Offensichtlich gilt $H(p,q) = 0$, falls $\forall_i\, p_i = q_i$, und $H(p,q) = \infty$, falls $\exists_i\, p_i \neq 0 = q_i$.

Da q eine Verteilung ist, gilt $q_n = 1 - \sum_{i=1}^{n-1} q_i$. Sei p gegeben. Durch Nullsetzen der Ableitungen von $H(p,q)$ nach q_j suchen wir die Verteilung q, für die $H(p,q)$ minimal wird. Für $j < n$ sei

$$f(q_j) := H(p,q) = p_n \log \frac{p_n}{1 - \sum_{i=1}^{n-1} q_i} + \sum_{i=1}^{n-1} p_i \log \frac{p_i}{q_i}$$

Ableiten ergibt $f'(q_j) = \frac{p_n}{q_n} - \frac{p_j}{q_j}$. Wenn man das für jedes j mit 0 gleichsetzt, ergibt sich, dass nur dort ein Minimum von $H(p,q)$ sein kann, wo $\forall_j\, \frac{p_j}{q_j} = \frac{p_n}{q_n}$ gilt. Wegen $\sum_i p_i = \sum_i q_i$ kann das nur für $p = q$ gelten. Das Minimum liegt also bei $H(p,p) = 0$ und die relative Entropie kann somit nicht negativ werden.

A.6 Die Normalverteilung

Eine $\mathbb{R}$-wertige Zufallsvariable X ist *normalverteilt* (oder auch *Gauß-verteilt*) mit Erwartungswert μ und Varianz σ^2 (bzw. Standardabweichung σ), kurz $X \sim \mathcal{N}(\mu, \sigma^2)$, falls X die Wahrscheinlichkeitsdichte $f : x \mapsto \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ besitzt, falls also gilt

$$\Pr(X \in [x, x+\varepsilon]) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \cdot \varepsilon + o(\varepsilon^2)$$

oder, was auf's selbe hinausläuft,

$$\Pr(X \in [a,b]) = \int_a^b \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx.$$

Gilt $X \sim \mathcal{N}(0,1)$, so heißt X *standardnormalverteilt*.

Die Binomialverteilung mit Parametern (n,p) kann man gut durch die Normalverteilung mit dem passenden Erwartungswert $n \cdot p$ und der passenden Varianz $n \cdot p \cdot (1-p)$ approximieren, falls n sehr

groß und p weder zu nahe bei 0 noch zu nah bei 1 ist. Eine Faustregel besagt, dass diese Approximation für viele Anwendungen hinreichend genau ist, falls $n \cdot p \cdot (1-p)$ größer als 9 ist.

Ein zufälliger Vektor

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix},$$

mit Werten in $\mathbb{R}^n$ heißt standardnormalverteilt, falls alle seine Komponenten X_i standardnormalverteilt und voneinander stochastisch unabhängig sind. Ist $v \in \mathbb{R}^m$ und M eine $m \times n$-Matrix, so ist $Y = v + MX$ ein normalverteilter Zufallsvektor mit Erwartungsvektor $\mathbb{E}Y = v$ und Kovarianzmatrix $K = M \cdot M^T$, wobei $.^T$ für "transponiert" steht. Der Eintrag K_{ij} in Zeile i und Spalte j der Kovarianzmatrix ist die Kovarianz von Y_i und Y_j, also $\mathbb{E}((Y_i - \mathbb{E}Y_i)(Y_j - \mathbb{E}Y_j))$. Jede m-dimensionale Normalverteilung ist durch ihren Erwartungsvektor und ihre Kovarianzmatrix eindeutig bestimmt.

A.7 Poisson-Approximation der Binomialverteilung

Eine Zufallsvariable X mit Werten in $\{0, 1, \ldots, n\}$ heißt *binomialverteilt* mit Parametern (n, p), wenn gilt

$$\mathrm{Ws}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Es gilt dann $\mathbb{E}X = np$ und $\mathrm{Var}(X) = np(1-p)$.

Das wesentliche Beispiel ist: Es werden n stochastisch unabhängige Zufallsereignisse betrachtet, die alle mit Wahscheinlichkeit p eintreten. Dann ist die Anzahl der eintretenden Ereignisse (n, p)-binomialverteilt. Zur Maximum-Likelihood-Schätzung des Parameters p siehe Seite 40.

Falls n sehr groß und p sehr klein ist, kann man die Binomialverteilung durch die Poisson-Verteilung zum Parameter $\lambda = np$ approximieren. Die Poisson-Verteilung ist dann rechnerisch etwas einfacher zu handhaben.

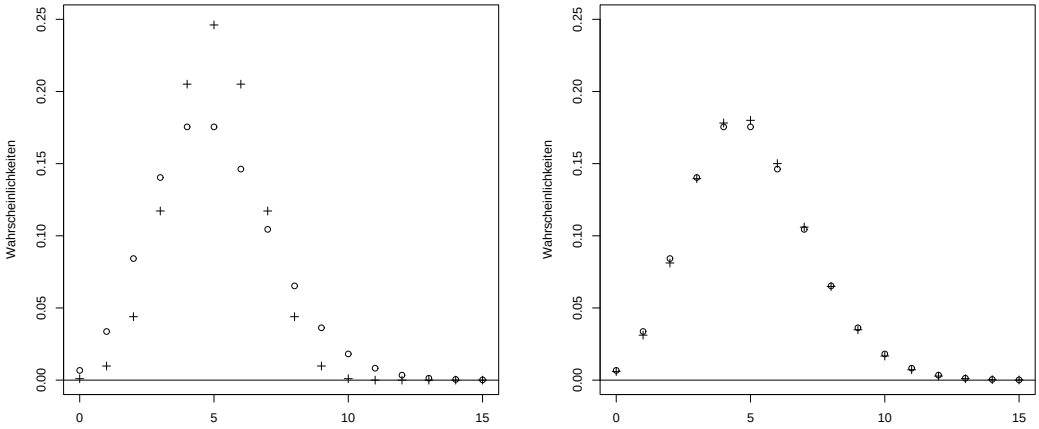
Eine Zufallsvariable Y mit Werten in $\{0, 1, 2, \ldots\}$ heisst Poisson-verteilt zum Parameter λ, falls gilt:

$$\mathrm{Ws}(Y = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

Es gilt dann $\mathbb{E}Y = \mathrm{Var}(Y) = \lambda$.

Die Poisson-Verteilung ist schon allein deshalb angenehm, da sie nur einen Einzigen Parameter hat. Durch ihren Erwartungswert sind bereits alle Wahrscheinlichkeiten bestimmt.

Die folgenden beiden Grafiken zeigen die Wahrscheinlichkeitsgewichte einer Poisson-Verteilung zum Parameter $\lambda = 5$ (Kreise) im Vergleich mit den entsprechenden Wahrscheinlichkeiten von Binomialverteilungen mit dem selben Mittelwert (Kreuze). Links sind die Parameter $(n, p) = (10, 0.5)$ und die Approximation funktioniert offensichtlich noch nicht sehr gut. Rechts sieht es mit $(n, p) = (100, 0.05)$ schon besser aus.



B Numerische Verfahren

Viele wichtige numerische Verfahren findet man in Press et al. (1992).

B.1 Das Newton-Verfahren zur Nullstellen-Suche

Um eine Nullstelle einer Funktion f zu finden, deren Ableitung man berechnen kann, starte man an einer Stelle x_0 und berechne $f(x_0)$ und $f'(x_0)$. Wenn f affin wäre (also von der Form $mx + b$), dann läge die Nullstelle bei $x_1 - f(x_0)/f'(x_0)$. Wir hoffen, dass f zumindest lokal um den Punkt x_0 herum durch eine affine Funktion approximiert werden kann, und dass x_1 somit näher an einer Nullstelle liegt. Dies iterieren wir. Sei also $x_{n+1} = x_n - f(x_n)/f'(x_n)$. Wenn f nicht zu wild ist, kommt man damit zumindest beliebig nahe an eine Nullstelle heran, und zwar meistens sehr schnell.

MEHRDIMENSIONALES NEWTON-VERFAHREN BESCHREIBEN... BIS DAHIN SIEHE PRESS ET AL.

Literatur

A. Aho, M. Corasick (1975) Efficient string matching: an aid to bibliographic search. *Comm. ACM*, **18**, 333–340

H. Akaike (1974) A new look at statistical model identification.
IEEE Trans. Autom. Control **19**, 716–723

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman (1990) Basic Local Alignment Search Tool *J. Mol. Biol.* **215**, 403–410

S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs
Nucleic Acids Research **25.17**, 3389–3402

A. Arribas-Gil, D. Metzler, J.-L. Plouhinec (2006+) Statistical alignment with a sequence evolution model allowing rate heterogeneity along the sequence.
Preprint.

L. E. Baum (1972) An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes.
Inequalities **3**, 1–8

P. Beerli, J. Felsenstein (2001) Maximum likelihood estimation of a migration matrix and effective population sizes in n subpopulations by using a coalescent approach
PNAS, **98.8**, 4563–4568

C. Burge, S. Karlin (1997) Prediction of complete gene structures in human genomic DNA.
Journal of Molecular Biology **268**, 78–94

H. Carillo, D. Lipman (1988) The multiple sequence alignment problem in biology.
SIAM Journal of Applied Mathematics **48**, 1073–1082

A. P. Dempster, N. M. Laird, D. B. Rubin (1977) Maximum likelihood from incomplete data via the EM algorithm.
Journal of the Royal Statistical Society B **39**, 1–38

A. Dembo, S. Karlin, O. Zeitouni (1994) Limit distribution of maximal non-aligned two-sequence segmental score.

Annals of Probability **22**, 2022–2039

R. Durbin, S. Eddy, A. Krogh, G. Mitchison (1998) *Biological sequence analysis*
Cambridge Univ. Press

R. Durrett (2002) *Probability Models for DNA Sequence Evolution*
Springer

S. R. Eddy (1995) Multiple alignment using hidden Markov models.
In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, S. Wodak (Eds.) *Proceedings of the Third Internationa Conference on Intelligent Systems for Molecular Biology* 114–120, AAAI Press

Efron, Halloran, Holmes (1996) Bootstrap confidence levels for phylogenetic trees
Proc. Natl. Acad. Sci. USA **93**, 13429–13434

A. Engel (1987) *Stochastik*
Klett Verlag

W. Ewens, G. Grant (2001) *Statistical Methods in Bioinformatics.*
Springer

W. Feller (1968) *An Introduction to Probability Theory and Its Applications*, Volume I, 3rd Edition.
John Wiley & Sons

J. Felsenstein (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach.
Journal of Molecular Evolution **17**, 368–376

J. Felsenstein (2004) *Inferring Phylogenies*.
Sinauer

R. Fleißner, D. Metzler, A. von Haeseler (2005) Simultaneous Statistical Multiple Alignment and Phylogeny Reconstruction.
Syst. Biol. **54**, 548-561

D. Gusfield (1999) *Algorithms on strings, trees and sequences: computer science and computational biology*.
Cambridge Univ. Press

R. Griffiths, S. Tavaré (1994) Simulating probability distributions in the coalescent

Theor. Popln. Biol. **46**, 131–159

S. Grossmann (2003) *Statistics of optimal sequence alignments*
Dissertation am Fachbereich Mathematik der Universität Frankfurt.

S. Henikoff, J. G. Henikoff (1992) Amino acid substitution matrices from protein blocks
Proc. Natl. Acad. Sci. USA **89**, 10915–10919

Hillis, Bull (1993) An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis
Syst. Biol. **2**, 182–192

D. S. Hirschberg (1975) A linear space algorithm for computing maximal common subsequences.
Communications of the ACM **18**, 341–343

I. Holmes, W. J. Bruno (2001) Evolutionary HMMs: A Bayesian approach to multiple alignment.
Bioinformatics, **17.9**, 803–820.

T. H. Jukes, C. Cantor (1969) Evolution of protein moelcules. In *Mammalian Protein Metabolism.* Academic Press. 21–132

S. Karlin, S.F. Altschul (1990) Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes.
Proceedings of the National Academy of Sciences **87**

J. F. C. Kingman (1982a) The Coalescent.
Stochastic Processes and their Applications **13**, 235–248.

J. F. C. Kingman (1982b) On the Genealogy of Large Populations.
Journal of Applied Probability **19A**, 27–43.

B. Knudsen, J. Hein (1999) Using stochastic context free grammars and molecular evolution to predict RNA secondary structure
Bioinformatics **15.5**, 446–454

U. Krengel (1998) *Einführung in die Wahrscheinlichkeitstheorie und Statistik*
Vieweg

M. Kuhner, J. Yamato, J. Felsenstein (1995) Estimating effective population size and mutation rate from

sequence data using Metropolis-Hasings sampling
Genetics, **140**, 1421–1430

G. A. Lunter, I. Miklós, Y. S. Song, J. Hein (2003) An efficient algorithm for statistical multiple alignment on arbitrary phylogenetic trees.
(erscheint im *J. Comp. Biol.*)

D. Metzler (2003) Statistical Alignment based on fragment insertion and deletion models.
Bioinformatics **19.4**, 490–499

D. Metzler (2006) Robust E-Values for Gapped Local Alignments.
Journal of Computational Biology **13**, 882–896

D. Metzler, R. Fleißner, A. Wakolbinger, A. von Haeseler (2001) Assessing variability by joint sampling of alignments and mutation rates.
J. Mol. Evol. **53.6**, 660–669

D. Metzler, R. Fleißner, A. Wakolbinger, A. von Haeseler (2005) Stochastic insertion-deletion processes and statistical sequence alignment.
In: J.D. Deuschel, A. Greven (Eds.) *Interacting Stochastic Systems*, Springer

D. Metzler, S. Grossmann, A. Wakolbinger (2002) A Poisson model for gapped local alignments.
Stat. Prob. Letters. **60**, 91–100

R. Mott, R. Tribe (1999) Approximate statistics of gapped alignments
Journal of Computational Biology **6**, 91–112

E. W. Myers, W. Miller (1988) Optimal alignments in linear space.
Computer Applications in the Biosciences **4**, 11–17

S. B. Needleman, C. D. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins.
Journal of Molecular Biology **48**, 443–453

C. Neuhauser (1999) The Ancestral Graph and Gene Genealogy under Frequency Dependent Selection.
Theoretical Population Biology **56**, 203–214.

P. Pevzner (2000) *Computational molecular biology: an algorithmic approach.*
MIT Press

H.W. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery (1992) *Numerical Recipes in C* (2nd Edition)
Cambridge University Press

J. A. Rice (1995) *Mathematical Statistics and Data Analysis*
Duxbury Press

G. Schnitger (2001) *Algorithmen der Bioinformatik.*
Skript zu Vorlesung vom WS 00/01

D. Siegmund, B. Yakir (2000) Approximate p-values for local sequence alignments.
Annals of Statistics **28**, 657–680

T. F. Smith, M. S. Waterman (1981) Identification of common molecular subsequences.
Journal of Molecular Biology **147**, 195–197

M. Stephens (2001) Inference under the Coalescent.
In: Balding et al. (eds.) *Handbook of Statistical Genetics*
John Wiley & Sons, Ltd.

M. Stephens, P. Donnelly (2000) Inference in molecular population genetics.
J. R. Statist. Soc. B. **62 (4)**, pp. 605–655

D. L. Swofford (2003) *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods).* Version 4. Sinauer Associates, Sunderland, Massachusetts.

F. Tajima (1989) Statistical method for testing the neutral mutation hypothesis by DNA polymorphism.
Genetics **123**, 585–595

E. A. Thompson (1975) *Human Evolutionary Trees.*
Cambridge University Press.

J. D. Thompson, D. G. Higgins, T. J. Gibson (1994) CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22**, 4673–4680

J. Thorne, H. Kishino (1992) Freeing phylogenies from artifacts of alignments.
Mol. Biol. Evol., **9.6**, 1148–1162

J. Thorne, H. Kishino, J. Felsenstein (1991) An evolutionary model for maximum likelihood alignment of DNA sequences.
J. Mol. Evol. **33**, 114–124

J. Thorne, H. Kishino, J. Felsenstein (1992) Inching towards reality: An improved likelihood model for sequence evolution.
J. Mol. Evol. **34**, 3–16

M. Waterman (1995) *Introduction to Computational Biology*.
Chapman & Hall

I. J. Wilson, D. J. Balding (1998) Genealogical inference from microsatellite data.
Genetics **150**, 499–510

Z. Yang (1994) Maximum-Likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods.
Journal of Molecular Evolution **39**: 306–314