

Seminar
Aktuelle Themen der Bioinformatik SS 07

Fachbereich Informatik und Mathematik
Johann Wolfgang Goethe-Universität, Frankfurt am Main

Thema:

Gibbs sampler for statistical multiple alignment - Jensen, Hein (2005)

Seminararbeit von Marten Rosselli

Inhalt

- 1. Übersicht**
- 2. Gibbs-Sampling-Algorithmus**
- 3. Das TKF91-Modell und Markovketten**
- 4. Unbeobachtbare Markovketten (HMMs)**
- 5. Der Algorithmus**
 - 5.1. Notation und Anwendung von Gibbs-Idee**
 - 5.2. Zustände und Transitionswahrscheinlichkeiten**
 - 5.3. Simulation eines „3-star-trees“**
 - 5.4. Unterschied zu Holmes und Bruno**
- 6. Mixing**
- 7. Parameterschätzung**
 - 7.1. ML-Ansatz**
 - 7.2. EM-Algorithmus**
- 8. Beispiel**
- 9. Fazit**

1. Übersicht

Ein Alignment, im Deutschen oft auch Alinierung genannt, dient dem Vergleich zweier oder mehrerer Sequenzen (Strings) und wird verwendet, um die funktionelle oder evolutionäre Verwandtschaft (Homologie) von z.B. DNA/RNA- oder Proteinsequenzen zu untersuchen oder um z.B. phylogenetische Bäume zu erstellen um gemeinsame Vorfahren zu bestimmen und Taxa zu ordnen.

Im Falle von zwei betrachteten Sequenzen spricht man von einem paarweisen Alignment (global, lokal, semiglobal).

Ein multiples Alignment steht im Gegenzug für das Analysieren mehrerer Sequenzen.

Beispiel für multiples Alignment:

Auszug der ClustalW-Output-Files eines Alignments von fünf Aminosäuresequenzen (Aminosäuren sind die Bausteine der Proteine).

Output: CLUSTAL W (1.83) multiple sequence alignment

```
FOS_RAT      MMFSGFNADYEASSRCSSASPAGDSLYYHSPADSSMGVNTQDFCADLSVSSANF
FOS_MOUSE   MMFSGFNADYEASSRCSSASPAGDSLYYHSPADSSMGVNTQDFCADLSVSSANF
FOS_CHICK   MMYQGFAGEYEAPSSRCSSASPAGDSLTYYPSPADSSMGVNSQDFCTDLAVSSANF
FOSB_MOUSE  -MFQAFPGDYDS-GSRCSS-SPSAESQ--YLSSVDSSPPAASQE-CAGLGEMPGSF
FOSB_HUMAN  -MFQAFPGDYDS-GSRCSS-SPSAESQ--YLSSVDSSPPAASQE-CAGLGEMPGSF
*:. . * . : * : : . * * * * * * * : : * * * . . * * * . . . . *
```

- * = Identische Aminosäuren/Basen in allen 5 Zeilen
- . = Sehr ähnliche Aminosäuren/Basen (semi-conserved substitutions)
- : = Sehr ähnliche Aminosäuren/Basen (conserved substitutions)

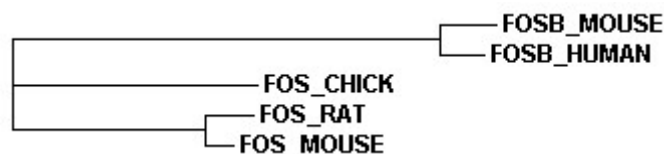


Abbildung 1: Phylogram Tree (ClustalW)

Multiples Alignment und die Bestimmung von Vorfahren beim Alignieren ist generell wichtig für die Biologie und Bioinformatik. Die Bestimmung eines multiplen Alignments ist wesentlich schwieriger als die Bestimmung eines paarweisen Alignments da man nicht auf solch effiziente Algorithmen wie den von Needleman-Wunsch (1970) oder Smith und Waterman (1981), die auf dynamischer Programmierung beruhen, zurückgreifen kann.

Um ein multiples Alignment zu finden benutzt beispielsweise das bekannte Programm ClustalW [1], auf welches später noch Bezug genommen wird, eine progressive Strategie: Es werden zuerst zwei Sequenzen aligniert um dann das Resultat gegen eine dritte Sequenz zu alignieren und so fort.

Der Ansatz in dem in dieser Ausarbeitung beschriebenen Paper ist ein statistischer Ansatz. Der Vorteil gegenüber Heuristiken oder progressiv alignierenden Programmen wie *ClustalW* ist die Miteinbeziehung der stochastischen Natur des Evolutionsprozesses. Ein Vorreiter des statistischen Sequenzalignments ist Zhu, Liu und Lawrence (1998).

Der Artikel „Gibbs sampler for statistical multiple alignment“ von Jens Ledet Jensen und Jotun Hein (2005) beschreibt einen Gibbs-Sampling-Algorithmus der für eine Menge von Sequenzen S_1, \dots, S_n die Sequenzen der Vorfahren und die dazugehörigen Alignments bestimmt. Sämtliche Sequenzen sind in einem binären Baum gespeichert und als Grundlage dient das Sequenz-Evolutionsmodell von Thorne, Kishino und Felsenstein (1991). Im Gegensatz zu den vielen Substitutionsmodellen werden im TKF91-Modell auch Insertionen und Deletionen berücksichtigt. Dieses Modell, auf dem der Artikel von Jensen und Hein fußt, führt uns unmittelbar zu unbeobachtbaren Markovketten welche Grundlage des hier präsentierten Gibbs-Samplers sind. Der Gibbs-Sampling-Algorithmus gehört zu der Klasse der Markov-Chain-Monte-Carlo (MCMC) Algorithmen. Bei der MCMC-Methode werden Zufallszahlen von einer Verteilung erzeugt indem man eine Markov-Kette konstruiert die die Zielverteilung als stationäre Verteilung hat und gegen diese konvergiert. Dabei ist die Geschwindigkeit wie schnell die Zustände der Markov-Kette gewechselt werden (*mixing-time*) von Bedeutung in Bezug darauf wie lange man die Kette laufen lassen muss bis die Kette im Gleichgewicht ist. Das Aussehen der Markovkette und die Transitionsmatrix werden in 5.2. (*Zustände und Transitionswahrscheinlichkeiten*) angegeben.

Der Gibbs-Sampling-Algorithmus simuliert die Vorfahren der Sequenzen schrittweise, d.h. zu jeweils drei Sequenzen (den Kindern eines Baum) wird der Vorfahre (der innere Knoten) und die drei Alignments (die Äste im Baum) vom inneren Knoten zu den drei äußeren Knoten bestimmt. Die Notation und das Aussehen des binären Baumes und eines *3-star-tree's* werden in 5.1. (*Notation und Anwendung von Gibbs-Idee*) erläutert.

Der Artikel von Jensen und Hein wurde zwar vier Jahre später als ein Artikel von Holmes und Bruno (2001) veröffentlicht, dennoch wurden beide Ideen zeitgleich entwickelt. Beide Artikel behandeln das gleiche Thematik, aber eben auf unterschiedliche Art und Weise. Im Anschluss wird ein kurzer Vergleich anhand von Tests von beiden Verfahren angegeben und es wird diskutiert unter welchen Rahmenbedingungen der hier angegebene Algorithmus den von Holmes und Bruno schlägt.

2. Gibbs-Sampling-Algorithmus

Gibbs-Sampling ist ein Algorithmus, um eine Folge von Stichproben der gemeinsamen Wahrscheinlichkeitsverteilung zweier oder mehrerer Zufallsvariablen zu erzeugen. Das Ziel ist es dabei, die unbekannte gemeinsame Verteilung zu approximieren. Der Algorithmus ist aufgrund der Ähnlichkeit des Sampling-Verfahrens mit Methoden der statistischen Physik nach dem Physiker Josiah Willard Gibbs benannt. Entwickelt wurde er von S. Geman und D. Geman. Gibbs Sampling ist ein Spezialfall des Metropolis-Hastings-Algorithmus, der wiederum zur Klasse der Markov-Chain-Monte-Carlo Algorithmen gehört. Oft wird er auch schlicht der Gibbs Sampler genannt.

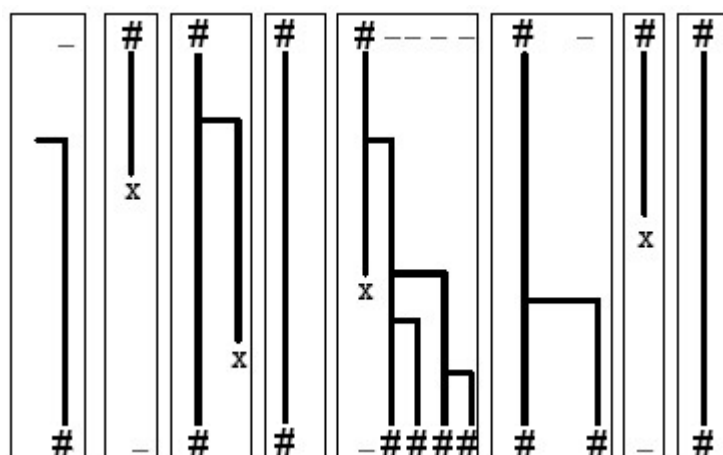
Gibbs-Sampling eignet sich besonders dann, wenn die gemeinsame Verteilung eines Zufallsvektors unbekannt, jedoch die bedingte Verteilung einer jeden Zufallsvariable bekannt ist. Das Grundprinzip besteht darin, in wiederholender Weise eine Variable auszuwählen und gemäß ihrer bedingten Verteilung einen Wert in Abhängigkeit der Werte der anderen Variablen zu erzeugen. Die Werte der anderen Variablen bleiben in diesem Iterationsschritt unverändert. Aus der entstehenden Folge von Stichprobenvektoren lässt sich eine Markow-Kette herleiten.

Der Artikel von Jensen und Hein beschreibt einen Gibbs-Sampler für die Simulation der gemeinsamen Vorfahren und deren Alignments bei gegebenen beobachtbaren Sequenzen.

3. Das TKF91-Modell und Markovketten

Im TKF91-Modell entwickelt sich jede Position innerhalb einer Sequenz unabhängig von den anderen Positionen der Sequenz. Die Veränderung wird für jede Position durch die Geburtsrate λ (typischer Wert wäre z.B. $\lambda=0,099$) und Sterberate $\mu > \lambda$ (typischer Wert $\mu=0,1$) angegeben.

Wenn eine neue Position, also ein Buchstabe (Entweder Nukleotiden bei DNA (DNA-Basen): Adenin (A), Guanin (G), Cytosin (C), Thymin (T) oder Aminosäuren bei Proteinen) in einer Sequenz, geboren wird, dann wird er rechts angefügt. Eine Position kann überleben, substituiert werden, oder aussterben. Eine (eindeutige) grafische Darstellung sieht z.B. so aus:



(Abbildung 2: TKF91-Modell)

Das Symbol "#" steht hier für die Präsenz eines Buchstabens und "x" steht für das Verschwinden/Aussterben eines Buchstabens. Insertionen am Sequenzanfang sind Kinder einer hier unsichtbaren unsterblichen Position (*Immortal-Rate*).

Der neue Buchstabe wird durch eine Verteilung π bestimmt. Ganz links der Sequenz ist ein Geburtsprozess (immortal-rate) mit Geburtsrate λ der sicherstellt das die Sequenz nicht letztlich völlig ausstirbt wegen $\mu > \lambda$.

Solange ein Buchstabe existiert unterliegt er dem Markov'schen Substitutionsprozess mit den stationären Wahrscheinlichkeiten die durch π gegeben sind und der Transitions Wahrscheinlichkeit von w_1 zu w_2 innerhalb einer Zeitspanne τ zu substituieren mit $f(w_2|w_1, \tau)$.

Die stationäre Verteilung einer Sequenz S der Länge L ist gegeben durch:

$$P(S) = (1-\gamma)\gamma^L \prod_{i=1}^L \pi(S[i]) \quad \text{mit} \quad \gamma = \frac{\lambda}{\mu} \quad (1)$$

S[i] entspricht dem i-te Element der Sequenz.

Wenn eine Sequenz S_1 zu S_2 in einer Zeitspanne τ evolviert, dann können wir die Evolution der ersten Sequenz in Bezug auf das Alignment durch Deletionen, Insertionen, Substitution von Buchstaben oder einfaches Überleben eines Buchstaben zusammenfassen.

Das TKF-Modell kann durch diese vier möglichen Ereignisse an jeder Position als versteckte Markovkette umformuliert werden (Durbin, Eddy, Krogh Mitchison (1998) und Hein (2001)).

Die drei Zustände der Markovkette sind Match (M), Deletion (D) und Insertion (I). M bedeutet, dass eine Position in Sequenz eins mit der Position in Sequenz zwei aligniert wurde. Das heißt der Buchstabe kann entweder gleich geblieben sein oder er hat sich verändert. Ist er aber in Sequenz zwei ausgestorben (es entsteht ein Gap in der zweiten Sequenz) wird der Zustand D verwendet und wir eine neue Position in Sequenz zwei eingefügt und der alte Buchstabe in Sequenz eins stirbt zugleich aus (Gab in der ersten Sequenz) geht die Markovkette in den Zustand I. Zusätzlich gibt es den Ende-Zustand ϵ um die zufällige Länge der Markovkette zu simulieren.

Hein verwendet ab nun das Symbol # für die Präsenz eines Buchstabens (Nukleotid oder Aminosäure) und das Symbol - für die Abstinenz eines Buchstabens.

Folgende Definition um die Notation künftig zu vereinfachen:

$$\gamma = \frac{\lambda}{\mu} \quad \text{und} \quad \beta = \frac{(1 - \exp(-(\mu - \lambda)\tau))}{(1 - \gamma \exp(-(\mu - \lambda)\tau))}$$

Um die Transitionsmatrix der Markovkette angeben zu können werden außerdem noch folgende Terme eingeführt:

$$b(\#, \#) = \gamma\beta \quad \text{und entsprechend die Gegenwahrscheinlichkeit:} \quad b(\#, -) = 1 - b(\#, \#)$$

Dies bedeutet, dass ein # gematched wird und ein # (das zweite # in der Formel) geboren wird. Die Gegenwahrscheinlichkeit hat die Interpretation, dass das erste # gematched wird aber kein Buchstabe geboren wird.

$$b(-, \#) = 1 - \left(\frac{\beta}{1 - \exp(1 - \mu \tau)} \right) \text{ mit Gegenwahrscheinlichkeit: } b(-, -) = 1 - b(-, \#)$$

Dies bedeutet, dass erste # stirbt aus und es wird aber ein # geboren. Die Gegenwahrscheinlichkeit bedeutet das # stirbt aus und es gibt keine neue Geburt.

$$s(\#) = \exp(-\mu \tau) \text{ mit Gegenwahrscheinlichkeit: } s(-) = 1 - s(\#)$$

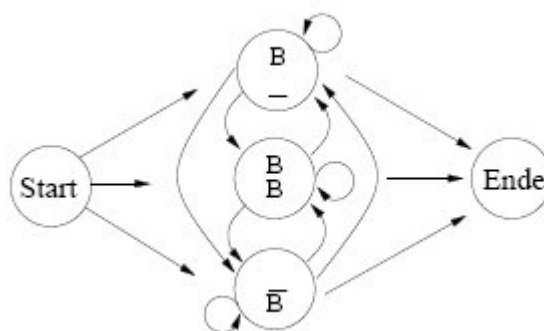
Die erste Wahrscheinlichkeit gibt das Überleben an und die Gegenwahrscheinlichkeit entsprechend das Aussterben.

Nach dieser Vorarbeit können wir endlich die Transitionsmatrix für die Markovkette eines Alignments angeben.

	M	D	I	ε
M	$b(\#, -)\gamma s(\#)$	$b(\#, -)\gamma s(-)$	$b(\#, \#)$	$b(\#, -)(1 - \gamma)$
D	$b(-, -)\gamma s(\#)$	$b(-, -)\gamma s(-)$	$b(-, \#)$	$b(-, -)(1 - \gamma)$
I	$b(\#, -)\gamma s(\#)$	$b(\#, -)\gamma s(-)$	$b(\#, \#)$	$b(\#, -)(1 - \gamma)$

(3)

Die passende Markovkette dazu sieht wie folgt aus:



(Abbildung: Markovkette eines Alignments)

Später werden wir darauf zurückgreifen um eine Übergangsmatrix für eine Markovkette im dreidimensionalen Fall mit multiplen Alignments zu formulieren.

4. Unbeobachtbare Markovketten (HMMs)

Eine unbeobachtbare Markovkette (Hidden Markov Model, HMM) hat die gleichen Eigenschaften wie eine *normale* Markovkette. D.h. auch hier darf der nächste Zustand nur vom jetzigen Zustand abhängen (Markov-Eigenschaft) und nicht von Teilen oder gar der ganzen Vergangenheit (abgesehen von Markovketten höherer Ordnungen). Das Besondere einer unbeobachtbaren Kette ist, dass es neben einem Zustandsraum noch einen Raum der für uns sichtbaren Emissionen der Kette gibt. Als Betrachter sehen wir nur die Emissionen und können dadurch nicht genau schließen in welchen Zustand die Kette nun tatsächlich ist, da mehrere Zustände aus dem Zustandsraum das gleiche Zeichen emittieren können.

Formal:

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

$$e_y(b) := W_S(S_i = b | X_i = y) = W_S(S_i = b | X_i = y, X_1, X_2, \dots, S_1, \dots, S_{i-1}, S_{i+1}, \dots) \quad [6]$$

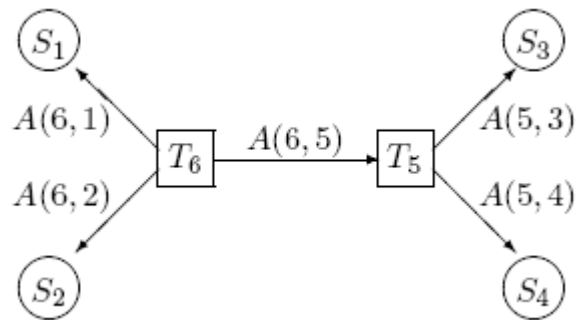
5. Der Algorithmus

5.1. Notation und Anwendung von Gibbs-Idee

Wir haben n beobachtbare Sequenzen S_1, \dots, S_n die ausschließlich in den Blättern des binären Baumes stehen. Es gibt dann $v = n - 2$ innere Knoten in denen die uns unbekannt Sequenzen, also die Vorfahren, stehen. Diese nennen wir T_{n+1}, \dots, T_{n+v} . Die Wurzel des Baumes ist der Knoten mit der Bezeichnung T_{n+v} . Jeder innere Knoten $n+1 \leq u < n+v$ hat einen Vorgänger $a(i)$ von den inneren Knoten $i+1, \dots, n+v$ und zwei Nachfolger $d_1(i)$ und $d_2(i)$ von den inneren Knoten $n+1, \dots, n+i-1$ und den Blättern des Baumes. Da die Wurzel keinen Vorgänger $a(n+v)$ hat, wird dort der Vorgänger durch einen Nachkommen ersetzt. Für ein Blatt j ist der Vorgänger $a(j)$ immer aus der Menge der innerer Knoten.

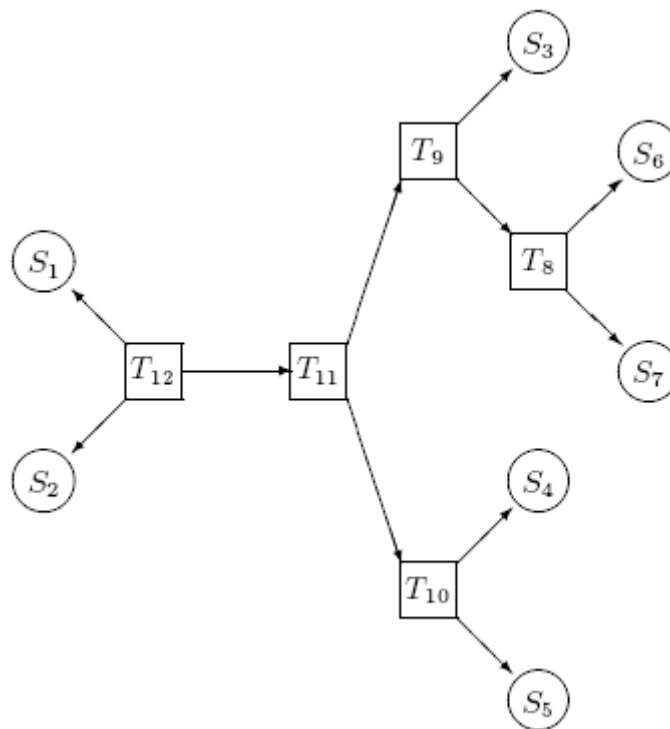
Beispiele:

Binärer Baum mit vier Sequenzen S_1, \dots, S_4 , mit inneren Knoten T_5, T_6 und der Wurzel T_6 :



(Abbildung 3: Baum mit vier Sequenzen)

Ein Baum mit sieben Sequenzen S_1, \dots, S_7 , mit inneren Knoten T_8, \dots, T_{12} und mit Wurzel T_{12} :



(Abbildung 4: Baum mit sieben Sequenzen)

Eine Kante vom Knoten $a(j)$ zum Knoten j wird mit j bezeichnet, so dass die Kantenmenge $j=1, \dots, n+v-1$ ist. Die Kante j hat die Länge τ_j und ein Alignment $A(a(j), j)$ besteht aus einer Sequenz mit den Zuständen M , D und I .

In jedem Schritt des Gibbs-Samplers wird ein *3-star-tree*, abhängig von den Sequenzen in den drei Blättern, simuliert. Wir betrachten für alle $r=n+1, \dots, n+v$ einen *3-star-tree* mit inneren Knoten r und Blättern $a(r)$, $d_1(r)$ und $d_2(r)$ und wir simulieren neue Werte für die Sequenz T_r und die Alignments $A(r, a(r))$, $A(r, d_1(r))$ und $A(r, d_2(r))$. Dies ist uns durch die bedingte Wahrscheinlichkeit, konditioniert auf die Sequenzen in den drei Blättern möglich. Diese bedingte Wahrscheinlichkeit ist in (11) weiter unten angegeben.

Beispiel:

Für den Baum in Abbildung 3 simulieren wir zuerst einen „3-star-tree“ mit der Wurzel $r=T_5$ und dann mit der neuen Wurzel $r=T_6$.

Um die Initialwerte der inneren Sequenzen T_{n+1}, \dots, T_{n+v} zu bekommen, verwenden wir einen fast gleichen Algorithmus wie der später hier beschriebene. Jedoch mit dem Unterschied dass nur *2-star-tree's* simuliert werden. Wir haben dann die gleiche Formel (11) für die Wertebestimmung eines inneren Knotens. Für $r=n+1, \dots, n+v$ simulieren wir T_r gegeben die Sequenzen von $d_1(r)$ und $d_2(r)$.

5. 2. Zustände und Transitionswahrscheinlichkeiten

Nun betrachten wir eine Markovkette die auf der Kette aus Punkt 3 (*Das TKF91-Modell und Markovketten*) unmittelbar aufbaut. Der Unterschied, der in diesem Abschnitt beschrieben wird, ist nun, dass es nicht nur ein Alignment zwischen zwei Sequenzen gibt sondern immer drei Alignments zwischen vier Sequenzen. Somit steigt die Anzahl der Zustände der ursprüngliche Kette auf 15 und die Transitionsmatrix wird entsprechend komplizierter.

Wir betrachten einen „3-star-tree“ mit T als innerer Knoten und den Blättern S_1, S_2, S_3 . Die evolutionäre Zeiten entlang den Kanten seien τ_1, τ_2, τ_3 .

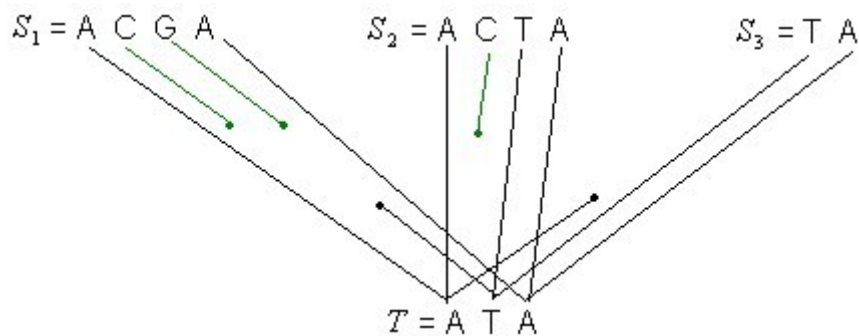
Zuerst einmal gibt es zwei Mengen von Zuständen.

Die erste Menge beschreibt die Menge der Blätter J in denen ein Zeichen aus der Ursequenz T in einem bestimmten Blatt überlebt. $j \in J$ bedeutet, dass der Buchstabe im Blatt j überlebt hat und $j \notin J$ bedeutet entsprechend, dass der Buchstabe im Blatt j nicht überlebt hat. Wir bezeichnen einen solchen (Match-)Zustand mit $M(J)$, wobei $J=0$ (da es möglich sein muss, dass die Position in allen drei Blättern ausstirbt. Durch $M(J)$ ist nämlich indirekt auch gegeben in welchen Blättern ein Zeichen deletiert wurde und deshalb gibt es auch keine Deletionszustände in der unteren Transitionsmatrix) möglich ist.

Die zweite Menge beschreibt eine Untermenge von J , in denen eine Geburt (Insertion) eines

neuen Zeichens stattgefunden hat. Wir nennen diese Menge $I(J)$ mit $J \neq \emptyset$.
 Wie schon bei der Markovkette mit zwei Sequenzen gibt es auch einen Endzustand ε .
 Von einem Zustand $M(J)$ können wir in jeden anderen Zustand wechseln. Von einem Zustand $I(J_1)$ können wir in jeden Zustand $M(J_2)$, aber nur in Zustände $I(J_2)$ mit $J_2 \subseteq J_1$ (Durch die Teilmengenrestriktion wird sichergestellt, dass der Algorithmus eine bestehende Sequenz mit einer Insertion mit weiteren Insertionen verlängern kann, aber die Anzahl der Gaps der inneren Sequenz so nicht größer als die längste Insertionssequenz wird. Ansonsten könnte der Algorithmus die innere Sequenz mit beliebig langen Gaps füllen und das Ziel ist aber die Anzahl der Gap-Positionen möglichst klein zu halten).
 Die Menge der 15 Zustände wird im Folgenden mit \mathcal{E} bezeichnet.
 Es sind genau 15 Zustände weil wir einmal die Potenzmenge (einschließlich der leeren Menge) $2^n = 2^3$ der $M(J)$ und einmal die der $I(J)$ Zuständen haben. Das sind dann 16 Zustände, aber bei $I(J)$ haben wir oben die leere Menge ausgeschlossen weshalb wir dann auf die Zahl 15 kommen.

Beispiel:



Markovkette:

$$M(\{1,2\}) \rightarrow I(\{1,2\}) \rightarrow I(\{1\}) \rightarrow M(\{2,3\}) \rightarrow M(\{1,2,3\})$$

Um die Transitionswahrscheinlichkeiten anzugeben, definieren wir $\beta(j)$, $b(\#, \#; j)$, $b(\#, -; j)$, $b(-, \#; j)$, $b(-, -; j)$, $s(\#; j)$ und $s(-; j)$ wobei hier τ durch τ_j mit $j=1,2,3$ (j ist der neu eingeführte Parameter den man im *TKF91-Modell* mit nur jeweils zwei Sequenzen natürlich nicht benötigt) ersetzt wird. Weiterhin definieren wir für einen Zustand x der Form $M(J)$ oder $I(J)$ für $j=1,2,3$, dass $x_j = \#$ wenn $j \in J$ und $x_j = -$ wenn $j \notin J$.

Die Transitionsmatrix der Übergangswahrscheinlichkeiten $p(x, y)$:

	$y=M(J_2)$	$y=I(J_2)$	$y=\varepsilon$
$y=M(J_1)$	$B(\#, \#)\gamma\left(\prod_{j=1}^3 s(y_j; j)\right)$	$B(\#, -)$	$B(\#, \#)(1-\gamma)$
$y=I(J_1)$	$B(-, \#)\gamma\left(\prod_{j=1}^3 s(y_j; j)\right)$	$B(-, -)$	$B(-, \#)(1-\gamma)$

(4)

Mit:

$B(\#, \#)=\left(\prod_{j=1}^3 b(x_j, -; j)\right)$ wobei ein einzelnes $b(x_j, -; j)$ dem $b(\#, -)$ aus 3 (*Das TKF91-Modell und Markovketten*) entspricht. Zur Erinnerung: Dies bedeutet, dass ein # gematched wird (das # der Formel) und kein Zeichen geboren wird (das „-“, der Formel) wodurch also klar ist, dass wir in einem Match-Zustand sind und nicht etwa in einem Insertion-Zustand.

Weiterhin ist $B(\#, -)=\left(\prod_{j=1}^3 b(x_j, y_j; j)\right)$, $B(-, \#)=\left(\prod_{j=1}^3 b(\#, -; j)\right)$ und

$$B(-, -)=\left(\prod_{j=1}^3 b(\#, y_j; j)\right) .$$

Beispiel:

Für den Übergang von dem Match-Zustand $y=M(J_1)$ in den Match-Zustand $y=M(J_2)$ haben wir zum einen $B(\#, \#)=\left(\prod_{j=1}^3 b(x_j, -; j)\right)$ (Info: Es werden hier unabhängige Wahrscheinlichkeiten multipliziert), wobei $b(x_j, -; j)$ eben angibt, dass wir in einem Match-Zustand sind (und nicht etwa in einem Insertion-Zustand). Zum anderen haben wir $\gamma\left(\prod_{j=1}^3 s(y_j; j)\right)$ wobei ein $s(y_j; j)$ angibt ob ein Zeichen in einem Alignment überlebt (es also gematched wird).

Der Startzustand entspricht $M=(\{1,2,3\})$ und emittiert keine Zeichen. Ein Zustand $M(J)$ emittiert ein Zeichen w_0 vom inneren Knoten und emittiert ein Zeichen w_j an den Blättern $j \in J$. Ein Zustand $I(J)$ emittiert nur ein Zeichen w_j an den Blättern $j \in J$.

Die Emissionswahrscheinlichkeiten sind:

$$p_e^0(w|M(J))=\pi(w_0)\prod_{\{j \in J\}} f(w_j|w_0; \tau_j) , \quad p_e^0(w|I(J))=\prod_{\{j \in J\}} \pi(w_j) \quad (5)$$

Summieren wir über alle möglichen w_0 erhalten wir

$$p_e(w|M(J)) = \sum_{w_0} \pi(w_0) \prod_{\{j \in J\}} f(w_j|w_0; \tau_j) \quad , \quad p_e^0(w|I(J)) = \prod_{\{j \in J\}} \pi(w_j) \quad (6)$$

5.3. Simulation eines „3-star-trees“

Wir bezeichnen die Länge der Sequenz S_j mit L_j , $j=1,2,3$. Eine Teilsequenz die mit an der Stelle a beginnt und an der Stelle b endet sei mit $S_j[a:b]$ bezeichnet. Für den Fall $a > b$ sei die Teilsequenz $S_j[a:b]$ leer. Für Spaltenvektoren u und v mit Integereinträgen bezeichne $S[u:v]$ die drei Teilsequenzen $S_j[u_j:v_j]$, $j=1,2,3$. Für einen Zustand $x=M(J)$ definieren wir $t(x)=1$ und einen 3-dimensionalen Vektor $l(x)$ mit Einseinträgen an den Koordinaten $j \in J$ und Nulleinträgen an den restlichen Stellen. Für einen Zustand $x=I(J)$ definieren wir $t(x)=0$ und $l(x)$ wie oben.

Somit erhält man die Länge der inneren Sequenz T wenn man die $t(x)$ für alle Positionen aufsummiert. Der dreidimensionale Vektor $l(x)$ hilft uns hingegen die Längen der äußeren Sequenzen zu berechnen. Auch hier muss man dazu nur alle $l(x)$ aufsummieren.

Wollen wir die Längen von den entsprechenden Sequenzen bis zu einer Position i wissen können wir folgende Formeln verwenden:

$$L^i = l(x^1) + \dots + l(x^i) \quad \text{und} \quad t^i = t(x^1) + \dots + t(x^i)$$

Das multiple Alignment für einen 3-star-tree ist gegeben durch die Zustände x^0, \dots, x^N wobei es sich um eine Folge aus Ξ handelt und es klar ist, dass Zustände aus Ξ bei $N > 15$ doppelt vorkommen müssen. x^0 sei der Startzustand I , $x^i \in \Xi$ mit $i=1, \dots, N$ die „inneren“ Zustände und x^{N+1} sei der Endzustand ε .

Nun geben wir eine Formel für die gemeinsame Verteilung der Sequenzen und der Alignments unter Verwendung der Formel für die Emissionswahrscheinlichkeiten von oben an.

$$P(N=n, x^1, \dots, x^n, T, S) = p(x^n, \varepsilon) \prod_{i=1}^n p(x^{(i-1)}, x^i) p_e^0(T[t^{(i-1)}+1:t^i], S[L^{(i-1)}+1:L^i] | x^i) \quad (7)$$

Wobei n und x^0, \dots, x^N so gewählt werden, dass $L^n = l(x^1) + \dots + l(x^n) = L$ und L der Vektor der Längen der (äußeren) Sequenzen ist. Summieren wir diesen Ausdruck über die möglichen Buchstaben der inneren Sequenz T wird p_e^0 durch $p_e(S[L^{(i-1)}+1:L^i] | x^i)$ ersetzt (d.h. T fällt weg).

Info: Der Ausdruck $T[t^{(i-1)}+1:t^i]$ ist Null für den Fall $t^{(i-1)} = t^i$ und ansonsten gibt $T[t^{(i-1)}+1:t^i]$ eine Position innerhalb der Sequenz T an.

Um die Wahrscheinlichkeit für einen Teil des Alingments zu bekommen, führen wir die Funktion $F(K|x^0)$ ein. K ist ein Spaltenvektor mit Integer-Einträgen und x^0 ist irgendein (Initial-) Zustand aus Ξ .

$$F(K|x^0) = \sum_{n=0}^{\infty} \sum_{x^1, \dots, x^n \in \Xi: K+L^n=L} p(x^n, \varepsilon) \prod_{i=1}^n p(x^{(i-1)}, x^i) p_e(S[K+L^{(i-1)}+1:K+L^i]|x^i) \quad (8)$$

Wobei die innere Summe Null ist wenn es keine x^0, \dots, x^N mit $K+L^n=L$ gibt. D.h. $F(K|x^0)=0$ wenn es j 's mit $K_j > L_j$ gibt.

Diese Funktion gibt uns die Wahrscheinlichkeit der Sequenzen $S[K+1:L]$ an, gegeben den Initialzustand x^0 .

Dann ergibt sich daraus, dass die Wahrscheinlichkeit für die 3 Sequenzen in den Blättern

$P(S)=F(0|I)$ ist. Die Null gibt an, dass man alle Positionen der drei Alignments berücksichtigt und I ist der Startzustand der Markovkette der hier die erste Position sein soll.

Aus (7) und (8) erhalten wir folgende Formel:

$$\begin{aligned} & P(N \geq k, x^1, \dots, x^k, T[1:t^k], S) \\ &= \left(\prod_{i=1}^k p(x^{(i-1)}, x^i) p_e^0(T[t^{(i-1)}+1:t^i], S[L^{(i-1)}+1:L^i]|x^i) \right) F(L^k|x^k) \end{aligned}$$

Diesen Ausdruck teilen wir nun durch die Wahrscheinlichkeit für die drei Sequenzen in den Blättern $P(S)=F(0|I)$ und wir erhalten

$$\begin{aligned} & P(N \geq k, x^1, \dots, x^k, T[1:t^k]|S) \\ &= \left(\prod_{i=1}^k p(x^{(i-1)}, x^i) p_e^0(T[t^{(i-1)}+1:t^i], S[L^{(i-1)}+1:L^i]|x^i) \right) F \frac{(L^k|x^k)}{(F(0|I))} \end{aligned} \quad (10)$$

Nun teilen wir diesen Ausdruck durch den gleichen Ausdruck jedoch mit k ausgetauscht durch $k-1$ und erhalten:

$$\begin{aligned} & P(N \geq k, x^k, T[t^{(k-1)}+1:t^k]|S, N \geq (k-1), x^1, \dots, x^{(k-1)}, T[1:t^{(k-1)}]) \\ &= p(x^{(k-1)}, x^k) p_e^0(T[t^{(k-1)}+1:t^k], S[L^{(k-1)}+1:L^k]|x^k) F \frac{(L^k|x^k)}{(F(L^{(k-1)}|x^{(k-1)}))} \end{aligned} \quad (11)$$

Mit (11) können wir nacheinander $(x^1, T[1:t^1])$, $(x^2, T[1:t^2])$, ... simulieren wenn wir $F(K|x)$ für alle x kennen.

$F(K|x)$ berechnen wir mit der Rekursion

$$F(K|x) = \sum_{x \in \mathcal{E}} p(x, z) p_e(S[K+1:K+l(z)]|x) F(K+l(z)|z)$$

Wir haben also (8) aufgebrochen in einmal die Summe nur über x^1 und dann den Rest ausgedrückt durch $F(K+l(z)|z)$.

Haben wir das Alignment x^1, \dots, x^N simuliert für einen *3-star-tree* mit innerem Knoten r und Blättern $a(r)$, $d_1(r)$ und $d_2(r)$ dann können wir direkt die Alignments $A(r, a(r))$, $A(r, d_1(r))$ und $A(r, d_2(r))$ ablesen.

5.4. Unterschied zu Holmes und Bruno

Der Algorithmus von Holmes und Bruno (2001) [6] funktioniert etwas anders als der hier vorgestellte Algorithmus. Da aber beide Algorithmen die gleiche Aufgabe haben ist es sinnvoll zu schauen und zu testen, in welcher Situation einer der beiden Algorithmen besser ist.

Bei dem Holmes/Bruno-Algorithmus wird im ersten Schritt jeweils drei „normale“ Alignments von der inneren Sequenz T zu den äußeren Sequenzen S_1, \dots, S_3 gemäß dem TKF91-Modell berechnet. Man verzichtet also auf den zusätzlichen Parameter j und berechnet stattdessen drei voneinander unabhängige Alignments. Es werden Zustände x^1, \dots, x^N der Form M , D oder I simuliert.

Der zweite Schritt besteht darin, die drei unabhängigen Alignments zu einem einzigen Alignment mit Zuständen der Form $M(J)$ und $I(J)$ zusammenzuführen.

6. Mixing

Der hier präsentierte Algorithmus simuliert die innere Sequenz und die Alignments direkt aus der Wahrscheinlichkeit konditioniert über die Sequenzen in den Blättern. Deshalb gibt es per Konstruktion schon bei einem *3-star-tree* keine Korrelation, da das Alignment in einem Schritt und in Abhängigkeit von allen 3 Sequenzen berechnet wird. Bei Bäumen mit mehr als 3 Blättern gibt es dann zwar eine Korrelation, aber diese ist sehr gering (s. Abb. 6, Baum mit 4 Sequenzen).

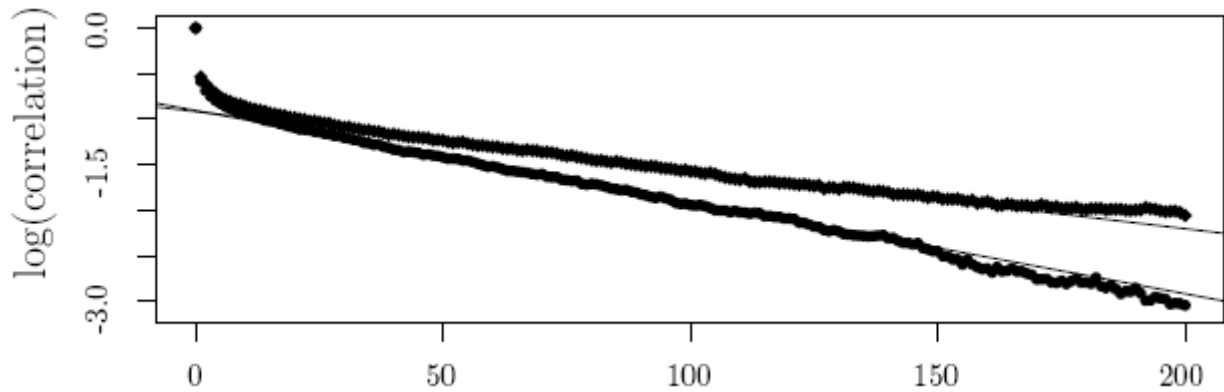
Simuliert man nach der Methode von Holmes und Bruno, stellt man fest, dass die Daten nur langsam transformiert (Mixing) werden und die Zustände nur langsam wechseln. Dazu betrachten wir die Werte der Autokorrelationen über einen Zeitraum in der der Algorithmus ausgeführt wird.

Info: Vergleicht man eine Folge mit sich selbst, so spricht man von Autokorrelation. Da jede unverschobene Folge mit sich selbst am Ähnlichsten ist, hat die Autokorrelation für die unverschobenen Folgen den höchsten Wert. Wenn zwischen den Gliedern der Folge eine Beziehung besteht, die mehr als zufällig ist, hat auch die Korrelation der ursprünglichen Folge mit der verschobenen Folge in der Regel einen Wert, der signifikant von Null abweicht. Man sagt dann, die Glieder der Folge sind autokorreliert.

In der folgenden Abbildung sehen wir die ersten 200 Autokorrelationen aufgetragen auf einer logarithmischen Y-Achseinteilung. Bei der oberen Kurve wurden 3 Sequenzen mit jeweils einer Länge von 150 verwendet und bei der unteren Kurve 3 Sequenzen mit jeweils einer Länge von 75. Vernachlässigen wir die Initialphase so stellen wir fest, dass es einen exponentiellen Abfall der

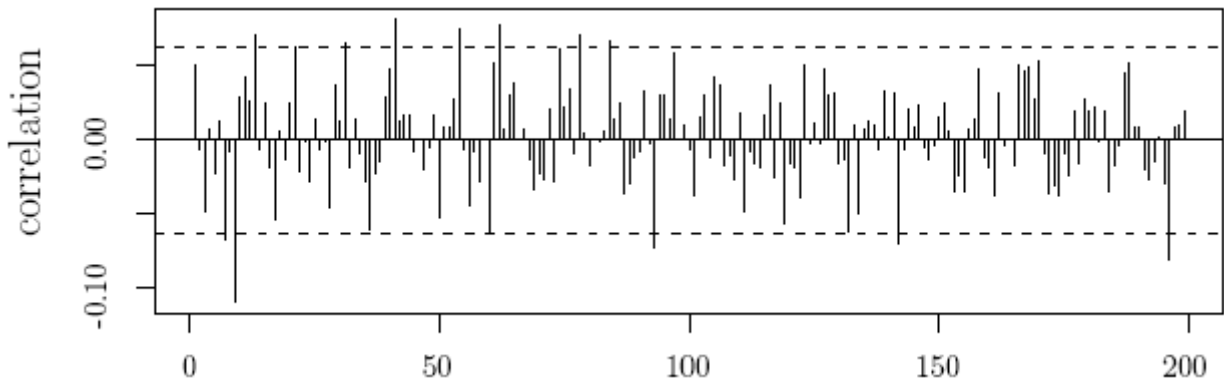
Kurven (hier fast Geraden) gibt.

Zu beachten ist, dass der Logarithmus von Werten zwischen Null bis Eins negativ ist und man so in diesem Beispiel erst eine Korrelation von fast Null (entspricht hier ungefähr $\log(\text{autocorrelation}) = -0,3$) hat (unabhängig!) wenn man auf der Zeitachse bei 200 angekommen ist.



(Abbildung 5, Autokorrelation bei Holmes-Bruno)

Im Gegensatz dazu schwankt die Autokorrelation bei dem Jensen-Hein-Algorithmus von Anfang an um (mit nur einer sehr geringen Abweichung) Null und das, obwohl bei diesem Beispiel 4 Sequenzen (siehe Abbildung 3) statt 3 Sequenzen aus oben genannten Grund verwendet wurden.



(Abbildung 6, Autokorrelation bei Jensen-Hein)

Um nun zu sehen, wie oft man nach Holmes-Bruno simulieren muss um die gleiche Präzision verglichen mit der Situation von n (wenn n groß ist) unabhängigen Werten zu haben nutzen wir eine bekannte Formel aus der Statistik.

$n \text{VAR}(x) \approx 1 + 2 \sum_k r_k$, wobei:

x ist der Durchschnitt, $\sum_k r_k$ die Summe der Autokorrelationen und $n \rightarrow \infty$.

Ist z.B. $1 + 2 \sum_k r_k = 100$, so ist die Interpretation dass wir $100n$ mal simulieren müssen um um die Genauigkeit verglichen mit der Situation von n zufälligen Werten zu haben.

Für den Holmes-Bruno-Algorithmus bedeutet das, dass man diesen Algorithmus länger laufen lassen muss um (ungefähr) die Genauigkeit des Jensen-Hein-Algo. zu erreichen. An dieser Stelle muss man aber auch sagen, dass die Laufzeit des ersteren Alg. besser ist (s. nächstes Kapitel).

In der folgenden Tabelle ist die Steigung (von 50 bis 150) der Kurven des Holmes-Bruno-Algorithmus für verschiedene Datensätze und der Wert für $1 + 2 \sum_k r_k$ angegeben.

Länge	3-star, 75	3-star, 150	4-Seq, 75	4-Seq, 150	7-Seq, 75
Steigung	-0,0101	-0,0064	-0,0061	-0,0061	-0,0055
$1 + 2 \sum_k r_k$	82	127	130	140	176

Der Algorithmus von Jensen und Hein hat die Komplexität $O(n^3)$, wobei n die typische Länge einer Sequenz ist. $O(n^2)$ ist die Komplexität des Holmes und Bruno Algorithmus.

Um nun ein fairen Vergleich beider Algorithmen anzugeben messen wir die Laufzeit in Sekunden bei Testdatensätzen und setzen diese Werte in Verhältnis zu der „Genauigkeit“ $1 + 2 \sum_k r_k$.

	3-star, 75	3-star, 150	4-seq, 75	4-seq, 150	7-seq, 75
H&B [s]	4,99	15,04	9,94	27,77	18,77
J&H [s]	158,5	775,4	330,3	1524,0	830,2
Ratio	32	52	33	55	44
$\frac{(1 + 2 \sum_k r_k)}{\text{Ratio}}$	2,6	2,5	3,9	2,6	4,0

Die letzte Zeile der Tabelle zeigt, dass der J&H-Algo in allen Fällen um den Faktor zwei bis vier effizienter als der H&B-Algo ist.

7. Parameterschätzung

7.1. ML-Ansatz

Allgemein: Um die Parameter des Modells zu schätzen können wir den Maximum-Likelihood (ML)-Ansatz verwenden. Für eine einzige Sequenz bedeutet das erstmal:

Gegeben sei eine Sequenz $s_1, s_2, \dots, s_n \in A$ von Beobachtungen aus einem HMM auf einem Zustandsraum Z . Gesucht seien nun die Parameterwerte $\theta = (P_{xy}, e_x(s))_{(x, y \in Z, s \in A)}$.

$\hat{\theta}$ ist der *Maximum-Likelihood-Schätzer* und das ist der Wert für θ der die Daten möglichst wahrscheinlich werden lässt.

$$\hat{\theta} := \arg \max_{\theta} Ws_{\theta}(S=(s_1, \dots, s_n))$$

Gesucht ist also die Maximalstelle der Likelihood-Funktion $L_{(s_1, \dots, s_n)}(\theta) := Ws_{\theta}(S=(s_1, \dots, s_n))$ [6].

Für zwei Sequenzen S_1 und S_2 mit einem Alignment $A = \{z^1, \dots, z^n\}$, wobei z^i der Form M , D oder I ist, gilt für die Wahrscheinlichkeit von S_2 und A gegeben S_1 :

$$P(S_2, A | S_1; \tau) = \tilde{p}(z^n, \varepsilon; \tau) \prod_{i=1}^n \tilde{p}(z^{(i-1)}, z^i; \tau) \tilde{p}_e^c(S_2[L_2^{(i-1)}+1:L_2^i] | S_1[L_1^{(i-1)}+1:L_1^i], z^i; \tau)$$

L_1 und L_2 sind die Längen der Sequenzen,

$\tilde{p}(\cdot, \cdot; \tau)$ ist die Transitions-Ws aus (3),

und \tilde{p}_e^c ist die bedingte Emissions-Ws und z^i ist ein Zustand der Form M , D oder I .

In unserem Fall mit mehreren Sequenzen und multiplen Alignments können wir diese Formel nutzen (für alle j) und damit eine *Likelihood-Funktion* angeben.

$$L_f(\theta) = P(T_{(n+v)}) \prod_{j=1}^{n+v-1} P(S_j, A(a(j), j) | T_{(a(j))}; \tau_j)$$

Hier taucht also nur $P(T_{(n+v)})$ neu auf und dieser Ausdruck kann ganz einfach mit (1) berechnet werden.

Wir suchen hier also auch wieder den Parameter θ der den oberen Ausdruck maximiert.

7.2. EM-Algorithmus

Der *Expectation-Maximization-Algorithmus* wird verwendet um einen unbekanntem Parameter zu

schätzen. Die Parameter θ dieses Modells sind die stationären Wahrscheinlichkeiten $\pi(\cdot)$, die Geburtsrate λ , die Sterberate μ , die Evolutionsdistanzen entlang der Äste τ und ein zusätzlicher Parameter ψ für die Substitutionswahrscheinlichkeiten der den Unterschied zwischen Transversionen (TV, erhalten den Charakter der Nukleobase und sind eine chemisch gesehen einfachere Reaktion und somit häufiger) und Transitionen (TS) beeinflusst.

Zuerst schätzen wir die π aus der empirischen Häufigkeit in der beobachteten Sequenzen

S_1, \dots, S_n . $\gamma = \frac{\lambda}{\mu}$ schätzen wir aus der durchschnittlichen Länge der beobachteten Sequenzen.

$$\hat{\pi}(a) = \sum_{j=1}^n \sum_{i=1}^{L_j} 1(S_j[i]=a) / \sum_{j=1}^n L_j,$$

$$\hat{\gamma} = \frac{(\bar{L})}{(1 + \bar{L})} \text{ mit } \bar{L} = \frac{1}{n} \sum_{i=1}^n L_i.$$

Wenn nun π und γ konstant sind, können wir die *Maximum-Likelihood-Funktion* schreiben als:

$$L_f(\theta) = \prod_{j=1}^{n+v-1} \{ b(\#, \#; j)^{N(\#, \#; j)} b(\#, -; j)^{N(\#, -; j)} b(-, \#; j)^{N(-, \#; j)} b(-, -; j)^{N(-, -; j)} \\ s(\#; j)^{N(\#; j)} s(-; j)^{N(-; j)} \prod_{w_1, w_2} f(w_2 | w_1; j)^{K(w_1, w_2; j)} \} \quad (20)$$

mit $\theta = (\mu, \psi, \tau_j; j=1, \dots, n+v-1)$.

$N(\#, \#; j)$ zählt wie oft der Term $b(\#, \#; j)$ der Transitionswahrscheinlichkeiten in dem Alignment $A(a(j), j)$ vorkommt. Alle anderen Zählerexponenten sind ähnlich definiert. $K(w_1, w_2; j)$ ist die Anzahl der Substitutionen von w_1 zu w_2 entlang des Astes j .

Um den EM-Algorithmus anwenden zu können müssen wir Erwartungswerte berechnen, da man mit Erwartungswerten effizient rechnen kann aufgrund der Linearität. Wir simulieren die Erwartungswerte (E-Schritt) der Zählerexponenten gegeben die beobachtbaren Sequenzen unter z.B. dem Parameter θ_1 . Dann finden wir einen neuen (besseren) Parameterwert θ_2 wenn wir (20) maximieren (M-Schritt).

Wir verwenden diese iterative Prozedur um (20) zu maximieren.

Unsere Hoffnung ist, dass unser immer weiter verbesserter Parameter $\hat{\theta}$ gegen $\tilde{\theta}$ konvergiert und wir iterieren solange bis sich der Wert nicht mehr ändert.

Jensen und Hein haben diesen EM-Algorithmus auf einen *3-star-tree* und den Baum aus Abbildung 3 angewendet (je 30 Iterationen und für jede Iteration 1000 Update-Schritte für Alignments und den Vorgänger). ψ ist hier aber konstant (=0.2), also $\theta = (\mu, \tau_1, \tau_2, \tau_3)$ bzw. $\theta = (\mu, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5)$.

Die andere Werte sind:

$$\pi=(0.2,0.3,0.2,0.3) \quad , \quad \mu=0,1 \quad , \quad \lambda=0,099$$

3-seq	μ	τ_1	τ_2	τ_3	\bar{l}_f , $\log(L_f)=l_f$
Start	0.100	0.80	0.80	0.80	19.41
Iteration 5	0.085	1.15	0.87	1.12	-2.65
Iteration 10	0.079	1.27	0.83	1.39	-6.58
Iteration 15	0.078	1.32	0.76	1.48	-4.44
Iteration 20	0.077	1.35	0.68	1.53	-1.82
Iteration 25	0.077	1.38	0.63	1.57	-0.87
Iteration 30	0.078	1.39	0.61	1.59	0.00

4-seq	μ	τ_1	τ_2	τ_3	τ_4	τ_5	\bar{l}_f
Start	0.100	0.80	0.80	0.80	0.80	0.80	-1.68
Iteration 5	0.101	0.94	0.79	0.71	0.72	0.75	4.83
Iteration 10	0.102	1.01	0.76	0.68	0.69	0.71	7.40
Iteration 15	0.107	1.09	0.72	0.70	0.67	0.67	3.58
Iteration 20	0.110	1.11	0.66	0.69	0.66	0.67	3.14
Iteration 25	0.113	1.14	0.64	0.70	0.66	0.67	1.34
Iteration 30	0.115	1.15	0.62	0.69	0.65	0.66	0.00

8. Beispiel

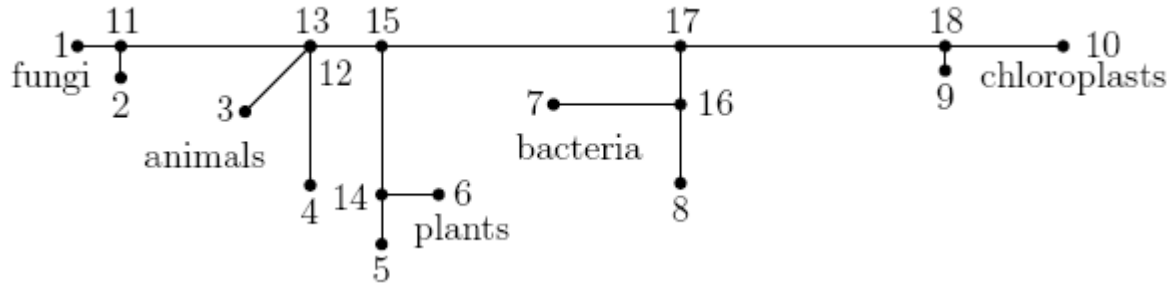
Wir betrachten nun ein Beispiel mit 10 *5S-RNA* (eine bestimmte und sehr kleine – um die 100 Nukleotiden - Art der RNA-Arten) Sequenzen deren gemeinsamer Vorfahre vermutlich vor ca. 3 Milliarden Jahren existiert hat. Die gesamte Kantenlänge in dem dazugehörigen phylogenetischen Baum beträgt insgesamt ca. 10 Milliarden Evolutionsjahre.

Eine wesentliche Funktion der RNA (Ribonukleinsäuren, ribonuclein acid) in der Zelle ist die Umsetzung von genetischer Information in Proteine. RNA ist hierbei sowohl als Informationsträger beteiligt (mRNA, RNA-Viren), als auch als katalytisches Molekül bei der Übersetzung dieser Information in ein Protein (rRNA, tRNA).

In der RNA kommen die folgenden organischen Basen (Nukleinbasen) vor: Adenin, Guanin, Cytosin und Uracil. Die ersten drei Basen kommen auch in der DNA vor. Uracil dagegen ersetzt Thymin als komplementäre Base zu Adenin. Vermutlich nutzt RNA Uracil, da dieses energetisch weniger aufwändig herzustellen ist (keine Methyl-Substituierung).

Historisch gesehen, waren *5S-RNA* die ersten Moleküle die man in Bezug auf Phylogenie untersucht

hat, weil sie kurz sind.



In der Abbildung sieht man den phylogenetischen Baum für 10 *5S-RNA*-Sequenzen. Die Länge der Kanten entsprechen den geschätzten Zeitabständen und somit den Parametern τ_i . Die 10 *5S-RNA*-Sequenzen sind also die *Kinder* des Baumes und die Vorfahren sind die inneren Knoten. Der evolutionäre Abstand von Knoten 12 und 13 ist so kurz, dass er hier in der Abbildung auf den gleichen Knoten fällt, was daran liegt, dass *5S-RNA* kurz sind.

Die 10 Sequenzen sind:

1: *Auricularia auricula-judae* ((Speise-)Pilz), 2: *Auricularia edulis* (Pilz), 3: *Caenorhabditis elegans* (ein Fadenwurm), 4: *Gallus gallus* (Haushuhn), 5: *Equisetum arvense* (Ackerschachtelhalm), 6: *Cycad revoluta*, 7: *Bacillus brevis*, 8: *Bacillus firmus*, 9: *Jungermannia subulata* und 10: *Dryopteris acuminata*.

Um die Verlässlichkeit der hier vorgestellten Methode zu zeigen betrachten wir zuerst die Ergebnisse von CLUSTALW für diese 10 Sequenzen:

```

1:  A----TCCACGGCCATAGGACTCTGAAAGCACTGCATCCCGT-CCGATCTGCAAAGTTAAACCAGAG
2:  A----TCCACGGCCATAGGACTGTGAAAGCACCGCATCCCGT-CTGATCTGCGCAGTTAAACACAG
3:  G----CTTACGACCATATCACGTTGAATGCACGCCATCCCGT-CCGATCTGGCAAGTTAAGCAAAG
4:  G----CCTACGGCCATCCCACCGTGGTAAACGCCGATCTCGT-CTGATCTCGGAAGCTAAGCAGGG
5:  GT---GGTGC GGTCATAACCAGCGTAATGCACCGGATCCCAT-CAGAACTCCGCAGTTAAGCGCGC
6:  G----GGTGC GGTCATAACCAGCGTAATGCACCGGATCCCAT-CAGAACTCCGCAGTTAAGCGCGC
7:  T----CTGGTGATGATGGCGGAGGGGACACACCCGTTCCCATACCGAAACACGGCCGTTAAGCCCTC
8:  T----CTGGTGGCGATAGCGAGAAAGTTCACACCCGTTCCCATACCGAAACACGGAAAGTTAAGCTTCT
9:  T---TCTGGTGTCTCAGGCGTGGAGGAACCACACCAATCCATCCCGAACTTGGTGGTGAAACTCTA
10: T-ATTCTGGTGTCTCCAGGCGTAGAGGAACCACACCGATCCATCTCGAACTTGGTGGTGAAACTCTG
11: A----TCCACGGCCATAGGACTCTGAAAGCACCGCATCCCGT-CCGATCTGCGAAGTTAAACACAG
12: G----CCTACGACCATAACCACCGTAAAGCACCCCATCCCGT-CCGATCTCGGAAGTTAAGCAGGG
13: G----CCTACGACCATAACCACCGTAAAGCACCCCATCCCGT-CCGATCTCGGAAGTTAAGCAGGG
14: G----GGTGC GGTCATAACCAGCGTAATGCACCGGATCCCAT-CAGAACTCCGCAGTTAAGCGCGC
15: G----CGTGC GACCATACCAGCGTAAAGCACCCGATCCCAT-CCGAACTCGGAAGTTAAGCGCGC
16: T----CTGGTGACGATGGCGGGGAGGTCACACCCGTTCCCATACCGAAACACGGAAAGTTAAGCTCGT
17: T----CTGGTGACGATGGCGGGGAGGTCACACCCGATCCCATACCGAACTCGGAAGTTAAGCTCGT
18: T---TCTGGTGTCTCAGGCGTGGAGGAACCACACCAATCCATCCCGAACTTGGTGGTGAAACTCTA
      *                * *          * *   * * *   * * * *
1:  TACCGCC-CAGTTAG-TACCACGGTGGGGGACCAACGCGGGAA-TCCTGGGTGCTG-T-GGT-T---
2:  TGCCGCC-TAGTTAG-TACCATGGTGGGGGACCACATGGGAA-TCCTGGGTGCTG-T-GGT-T---
3:  TTGAGTC-CAGTTAG-TACTTGGATCGGAGACGGCCTGGGAA-TCCTGGATGTTG-TAAGC-T---
4:  TCGGGCC-TGTTAG-TACTTGGATGGGAGACCTCCTGGGAA-TACCGGTGCTG-TAGGCTT---
5:  TTGGGCCAGAA-CAG-TACTGGGATGGGTGACCTCCCGGGAAGTCCTG-GTGCG-CACCC-C---
6:  TTGGGTTGGAG-TAG-TACTAGGATGGGTGACCTCCTGGGAAGTCCTA-ATATTG-CACCC-TT--
7:  CAGCGCC-AA--TGG-TACTTGCTCCGCAGGGA-GCCGGGAG-AGTAGGACGTGCCAGGC-----
8:  CAGCGCC-GA--TGG-TAGTT-AGGGGCTGTCC-CCTGTGAG-AGTAGGACGTGCCAGGC-----
9:  TTGCGGT-GA--CGA-TACTGTAGGGGAAGCCC-GATGGAAA-AATAGCTCGAAGCCAGGA---T-
10: CCGCGGT-AA--CCAATACTCGGGGGG-GGCC-TGCGGAAA-AATAGCTCGATGCCAGGA---TA
11: TACCGCC-TAGTTAG-TACCATGGTGGGGGACCACATGGGAA-TCCTGGGTGCTG-T-GGT-T---
12: TTGCGCC-GAGTTAG-TACTTGGATGGGAGACCACATGGGAA-TCCTGGGTGCTG-TAGGC-T---
13: TTGCGCC-GAGTTAG-TACTTGGATGGGAGACCACATGGGAA-TCCTGGGTGCTG-TAGGC-T---
14: TTGGGCCAGAG-TAG-TACTGGGATGGGTGACCTCCTGGGAAGTCCTG-GTGCTG-CACCC-T---
15: TTGCGCC-GAG-TAG-TACTTGGATGGGAGACCACCTGGGAA-TCCTGGGTGCTG-CAGGC-T---
16: CAGCGCC-GA--TGG-TACTTGAGCGGCAGGCC-CCTGGGAG-AGTAGGACGTGCCAGGC-----
17: CAGCGCC-GA--TGG-TACTTGAGTGGCAGGCC-CCTGGGAA-AATAGGGCGCTGCCAGGC-----
18: CTGCGGT-GA--CGA-TACTGTAGGGGAAGCCC-TATGGAAA-AATAGCTCGATGCCAGGA---T-
      *          **      *          * *          *

```

(Eine Spalten die mit einem „*“ gekennzeichnet ist, ist an allen Stellen identisch.)

Man lässt dann den hier vorgestellten Algorithmus laufen um mehrere Beispielsequenzen von der a posteriori Verteilung der Alignments gegeben die beobachtbaren Sequenzen zu erhalten. Hein und Jensen verwendeten die Parameter des EM-Algorithmus und sie simulierten 1000 Updateschritte und berechneten die Häufigkeit mit der eine bestimmte Spalte identisch in allen Stellen ist. Die

meisten Stellen die auch von CLUSTALW markiert wurden hatten eine a posteriori Wahrscheinlichkeit identisch zu sein von teilweise über 95%. In wenigen Fällen war die Wahrscheinlichkeit unter 80% und in einem Fall war sie 16%. Solche prozentualen Aussagen kann man mit einem Programm wie CLUSTALW alleine nicht treffen.

9. Fazit

Es wurde gezeigt, dass es möglich ist ein multiples Alignment im Rahmen des TKF-Modells mit einem Gibbs-Sampler zu simulieren bei dem in jedem Schritt ein *3-star-tree* aktualisiert wird. Dabei ist die Korrelation sehr gering und die Zustände werden schnell gewechselt als bei Holmes-Bruno. Außerdem ist die Laufzeit des hier angegebenen Algorithmus bei den hier verwendeten Daten um den Faktor 2 bis 4 besser.

Der hier vorgestellte Algorithmus ist auch nicht allein beschränkt auf das TKF-Modell sondern ein allgemeineres Markov-Modell wäre denkbar wenn man weitere Parameter einführt.

Referenzen:

[1] European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI), *ClustalW*.

<http://www.ebi.ac.uk/clustalw/>

[2] J. L. Jensen, J. Hein (2005),
Gibbs sampler for statistical multiple alignment.
Statistica Sinica **15** (2005), 889-907.

[3] J. Zhu, J. S. Liu, C. E. Lawrence (1998),
Bayesian adaptive sequence alignment algorithms,
Bioinformatics **14**, 25-39.

[4] J. L. Thorne, H. Kishino, J. Felsenstein (1991),
An evolutionary model for maximum likelihood alignment of DNA sequences.
J. Mol. Evol. **33** (1991), 114-24.

[5] I. Holmes, W. J. Bruno (2001),
Evolutionary HMMs: a Bayesian approach to multiple alignment.
Bioinformatics Vol. **17** no. 9 (2001), 803-820.

[6] D. Metzler (2007),
Algorithmen und Modelle der Bioinformatik, Skript zur Vorlesung WS 06/07.

[7] R. Durbin, S. Eddy, A. Krogh, G. Mitchison (1998),
Biological Sequence Analysis: Probabilistic Models of Protein and Nucleic Acids.
Cambridge University Press, Cambridge, UK.

[8] J. Hein (2001),
An algorithm for statistical alignment of sequences related by a binary tree.
Pacific Symposium on Biocomputing, **179-190**. World Scientific, Singapore.