Ein springendes Profil-Hidden-Markov-Modell und Anwendungen auf Rekombinationen in viralen Genomen

Diplomarbeit

vorgelegt von

Anne-Kathrin Schultz

aus
Göttingen

angefertigt

im Institut für Mikrobiologie und Genetik, Abteilung Bioinformatik,
der Georg-August-Universität zu Göttingen

2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungen

(C)RF	(**C**iculating) **R**ecombinant **F**orm
DNA	**D**eoxyribo**n**ucleic **A**cid (Desoxyribonukleinsäure (DNS))
HCV	**H**epatitis **C** **V**irus
HIV	**H**uman **I**mmunodeficiency **V**irus
HMM	**H**idden-**M**arkov-**M**odell
HMMER	**HMM**-Software-Programm
JALI	**J**umping-**Ali**gnment-Algorithmus
jP	Sprungwahrscheinlichkeit (**j**ump **P**robability) für ein jpHMM
jpHMM	Springendes Profil-HMM (**j**umping **p**rofile **HMM**)
LANL	**L**os **A**lamos **N**ational **L**aboratory
ML	**M**aximum **L**ikelihood
MSA	**M**ultiples **S**equenz**a**lignment
nt	**N**ukleo**t**id
Profil-HMM	**Profil**-**H**idden-**M**arkov-**M**odell
SAM	**S**equence **A**lignment and **M**odeling System
SimPlot	Programm-Paket (**Sim**ilarity **Plot**)
$\#\{...\}$	Anzahl der Elemente der Menge $\{...\}$
$\|x\|$ bzw. $\|s\|$	Länge eines Pfades x bzw. einer Sequenz s

Einleitung

Eine grosse Herausforderung in der Bioinformatik ist die Zuordnung von Sequenzen unbekannter oder nur teilweise bekannter Herkunft zu bereits bekannten Sequenzfamilien. Eine Besonderheit stellen dabei Genome von Viren dar, da für sie aufgrund ihrer Rekombinationsfähigkeit eine Vielzahl verschiedener Typen existiert. Zu den genetisch variabelsten Viren gehört das *Human Immunodeficiency Virus* (HIV) und das *Hepatitis C Virus* (HCV). Eine Klassifizierung dieser Viren ist besonders wichtig, um ihre weltweite Verbreitung beobachten und möglicherweise einen Impfstoff finden zu können.
Ein weiteres wichtiges Beispiel ist die Klassifizierung von Proteinen. Dabei spielt die Zuordnung von entfernt verwandten Proteinen zu einer Sequenzfamilie eine besondere Rolle, da diese Proteine häufig nur eine sehr geringe Sequenzähnlichkeit aufweisen.
In dieser Arbeit präsentieren wir eine neue Methode, ein sog. *springendes Profil-Hidden-Markov-Modell* (*jpHMM*), zum Vergleich von DNA- bzw. Proteinsequenzen mit einer gegebenen Sequenzfamilie. Dabei nehmen wir an, dass sich die Sequenzfamilie in verschiedene Subtypen, jeweils bestehend aus einer oder mehreren Sequenzen, einteilen lässt. Für eine Proteinfamilie können dies beispielsweise Unterfamilien der Familie sein.
Ein jpHMM ist ein spezielles Hidden-Markov-Modell. Es bestimmt die Ähnlichkeit einer Anfragesequenz zu der Sequenzfamilie bzw. zu den einzelnen Subtypen mithilfe eines Alignments der Sequenz zu einem multiplen Sequenzalignment der gegebenen Subtypen. Dabei wird für jeden Subtyp der Sequenzfamilie ein Profil-HMM entwickelt. Diese Profil-HMMs werden durch Übergänge, sog. Sprünge, miteinander verbunden. Die Anfragesequenz wird nun zu dem multiplen Sequenzalignment aligniert, indem jede Position der Sequenz zu einem *Referenzsubtyp* aligniert wird. Aufgrund der möglichen Sprünge zwischen den verschiedenen Profil-HMMs kann der Referenzsubtyp innerhalb des Alignment wechseln. Dadurch können verschiedene Bereiche der Sequenz zu unterschiedlichen Subtypen des Alignments aligniert werden, d.h jeder Position der Sequenz kann ein anderer Subtyp zugeordnet werden. Ein jpHMM kann also als eine Verallgmeinerung der *Jumping-Alignment*-Methode [24, 25] gesehen werden, die eine Sequenz zu dem Alignment einer Sequenzfamilie aligniert, indem verschiedene Bereiche der Sequenz zu verschiedenen Sequenzen des Alignments lokal aligniert werden. Der Vorteil eines jpHMMs gegenüber einem

Jumping Alignment ist dabei, dass nicht nur einzelne Sequenzen betrachtet werden, sondern jeweils das gesamte Profil eines Subtyps.
Durch Zurückverfolgen des besten Alignments der Anfragesequenz zum multiplen Sequenzalignment lässt sich dann feststellen, welche Bereiche der Sequenz welchen Subtypen der Sequenzfamilie am ähnlichsten sind und wo sich die Sprungstellen, die sog. *Bruchstellen*, befinden. Dadurch lässt sich eine Vorhersage über die Subtypzusammensetzung möglicherweise rekombinanter Sequenzen treffen.
Angewandt wurde das jpHMM zur Identifizierung von Rekombinationen in HIV- und HCV-Sequenzen. Das HI-Virus wird unterteilt in die beiden Stämme HIV-1 und HIV-2. Viren des Stamms HIV-1 lassen sich in drei phylogenetische Gruppen M (*Major*, *Main*), N (*Non-M, Non-O*) und O (*outlier*) einteilen. M ist die für die weltweite Ausbreitung von HIV verantwortliche Gruppe und kann in zehn Subtypen A - H, J - K unterteilt werden, die sich wiederum in mehrere Sub-Subtypen unterteilen lassen [19]. Viren des Stamms HIV-2 und der Gruppen N und O wurden bisher hauptsächlich in einigen afrikanischen Ländern entdeckt, und auch dort nur in sehr wenigen Fällen [16]. HCV-Sequenzen lassen sich in sechs Genotypen einteilen, die sich ebenfalls in eine Vielzahl von Subtypen unterteilen lassen.
Für beide Viren wurde bereits eine Vielzahl rekombinanter Sequenzen entdeckt, die sich aus den genannten Subtypen zusammensetzen. Rekombinante Viren mit derselben Struktur, d.h. bestehend aus denselben Subtypen mit Bruchstellen an ähnlichen Positionen, die in mehreren, nicht verwandten Individuen gefunden wurden, werden für HIV *CRF*s und für HCV *RF*s ((*circulating*) *recombinant forms*) genannt. Bisher bekannte RFs von HCV haben eine sehr einfache Struktur. Sie setzen sich jeweils nur aus zwei Subtypen zusammen und enthalten insgesamt nur eine Bruchstelle. CRFs von HIV können dagegen mit weiteren Subtypen rekombiniert werden, wodurch sie eine sehr viel komplexere Struktur als RFs haben können.
Je weiter die Ausbreitung der verschiedenen Subtypen von HIV und HCV weltweit fortschreitet, desto mehr (C)RFs mit immer komplexeren Strukturen entstehen, und desto schwieriger wird es, diese Sequenzen zu klassifizieren.
Bisher entwickelte Programme zur Identifizierung rekombinanter HIV-Sequenzen basieren auf einem paarweisen Vergleich einer Anfragesequenz zu einer Menge von Referenzsequenzen der bekannten Subtypen. Eines der am häufigsten verwendeten Werkzeuge zur Rekombinationsanalyse ist das Programm-Paket *Simplot*. Es beinhaltet zwei verschiedene Methoden, die jeweils die paarweise Ähnlichkeit der Anfragesequenz zu jeder Referenzsequenz mittels eines *sliding windows* auf dem multiplen Alignment aller betrachteten Sequenzen bestimmen. Eine Methode ist der sog. *similarity plot* [12], woraus sich auch der Name *Simplot* ableitet: Innerhalb des jeweils betrachteten Fensters wird die prozentuale Identität der Anfragesequenz zu jeder der Referenzsequenzen berechnet und der mittleren Position der Anfragesequenz in diesem Fenster zugeordnet. Die so berechneten Werte werden im *similarity plot* graphisch dargestellt. Die Referenzsequenz, der für eine bestimmte Position der Anfragesequenz die höchste Identität zugeordnet wird, gibt den Subtyp an, der für diese Position vorhergesagt wird. Die zweite in diesem Programm-Paket verwendete

Methode ist *bootscanning* [20]. Hier wird für die jeweils betrachteten Abschnitte der Sequenzen ein phylogenetischer Baum erzeugt. Positionen, an denen sich das Verzweigungsmuster des Baumes ändert, markieren Rekombinations-Bruchstellen.
In dieser Arbeit geben wir zunächst einen Überblick über die zur Entwicklung eines jpHMMs verwendeten Methoden. In Kapitel 1 werden Hidden-Markov-Modelle definiert und für einige der bekannten HMM-Algorithmen eine Verallgemeinerung gegeben und bewiesen. Dies ist notwendig, um diese Algorithmen auf ein Profil-HMM anwenden zu können. In den Kapiteln 2 und 3 stellen wir Profil-HMMs und Jumping Alignments vor, und vergleichen die jeweiligen Vorteile beider Methoden am Ende von Kapitel 3.
In Kapitel 4 definieren wir dann ein jpHMM, wobei wir die einem jpHMM zugrundeliegende Idee der Verknüpfung eines Profil-HMMs mit einem Jumping Alignment nochmal genau erläutern. Ausserdem stellen wir in diesem Kapitel den *beam-search*-Algorithmus [13, 17] vor, den wir zur Verringerung der Laufzeit und des Speicherbedarfs des für ein jpHMM implementierten Programmes nutzen. Die Implementation dieses jpHMM-Programmes wird in Kapitel 5 beschrieben.
In Kapitel 6 vergleichen wir für mehrere Datensätze von HIV- und HCV-Sequenzen die vom jpHMM vorhergesagten Rekombinationen und Bruchstellen mit den entsprechenden Ergebnissen von Simplot. Im Anschluss daran geben wir in Kapitel 7 einen kurzen Ausblick auf die Verwendung eines jpHMMs zur Proteinklassifikation.

Kapitel 1

Hidden-Markov-Modelle

Ein *Hidden-Markov-Modell* (*HMM*) [18] ist ein probabilistisches Modell. Um dieses genau definieren zu können, geben wir in diesem Kapitel zunächst eine kurze Einführung in *Markov-Ketten*. Im Anschluss an die Definition eines HMMs stellen wir dann zwei der bekanntesten HMM-Algorithmen vor, den *Vorwärts-* und den *Viterbi-Algorithmus*. Da wir diese Algorithmen ebenfalls für ein *Profil-HMM* (Kap. 2) nutzen wollen, werden wir bereits in der Definition eines HMMs den Fall eines Zustandsraumes mit *stummen Zuständen* berücksichtigen, und darauf aufbauend eine Verallgemeinerung der in der aktuellen Literatur gegebenen Algorithmen beweisen.

Sei in diesem und allen weiteren Kapiteln $\Sigma := \{\sigma_1, \ldots, \sigma_n\}$ eine endliche Menge von Symbolen bzw. Buchstaben, die wir *Alphabet* nennen. Eine endliche Folge $s = s_1 s_2 \ldots s_l$ von Buchstaben aus Σ heisst *Wort*. Für ein Wort $s = s_1 s_2 \ldots s_l$ bezeichnet $|s| = l$ die Länge des Wortes s. Σ^l ist die Menge aller Wörter der Länge l mit Buchstaben in Σ.

1.1 Definition eines Hidden-Markov-Modells

Definition 1.1 (Markov-Kette) *Sei $(\Omega, \mathscr{A}, P)$ ein Wahrscheinlichkeitsraum. Eine* Markov-Kette 1.Ordnung *ist eine Folge von Zufallsvariablen $X_1, X_2, \ldots$ auf Ω mit Werten in einer abzählbaren Menge Z, $X_i : \Omega \longrightarrow Z$, $i \geq 1$, die die folgende* Markovsche Eigenschaft *besitzt:*
Für alle $i \in \mathbb{N}$ und für alle $x_1, x_2, \ldots, x_{i+1} \in Z$ mit

$$P(X_1 = x_1, \ldots, X_i = x_i) > 0 \tag{1.1}$$

ist

$$P(X_{i+1} = x_{i+1} \,|\, X_1 = x_1, \ldots, X_i = x_i) = P(X_{i+1} = x_{i+1} \,|\, X_i = x_i). \tag{1.2}$$

Wir nennen die Menge Z einen *Zustandsraum* und interpretieren X_i als einen Zustand eines Systems zum Zeitpunkt i.
Die Folge der Zufallsvariablen hat also die Markovsche Eigenschaft, wenn die Wahrscheinlichkeit, zum Zeitpunkt $i+1$ in einen beliebigen Zustand zu gelangen, nur vom Zustand zum Zeitpunkt i und von i selbst abhängt, aber nicht davon, in welchen Zuständen das System früher war.

Folgerung 1.2 *Ist $X_1, X_2, \ldots$ eine Markov-Kette 1. Ordnung, so gilt für alle $i > 1$ und alle $x_1, \ldots, x_i \in Z$*

$$\begin{aligned} & P(X_1 = x_1, \ldots, X_i = x_i) \\ = \; & P(X_1 = x_1) P(X_2 = x_2 \,|\, X_1 = x_1) \cdots P(X_i = x_i \,|\, X_{i-1} = x_{i-1}). \end{aligned} \tag{1.3}$$

Definition 1.3 (Homogene Markov-Kette) *Eine Markov-Kette 1. Ordnung heisst* homogen*, wenn für alle $z_k, z_l \in Z$*

$$P(X_{i+1} = z_l \,|\, X_i = z_k) \tag{1.4}$$

unabhängig vom Zeitpunkt i ist.

Im Folgenden bezeichne $X[1,i]$ bzw. $X[1,i+1)$ eine beliebige Folge $X_1, \ldots, X_i$ von Zufallsvariablen.
Um die Verteilung einer Markov-Kette vollständig definieren zu können, muss die Verteilung von X_1 gegeben sein. Sei B ein sog. *Start-Zustand* und X_0 eine konstante Zufallsvariable so dass gilt $X_0 \equiv B$. Dann ist die Verteilung der Markov-Kette $X_0, X_1, \ldots$ durch die im Folgenden definierte *Übergangsmatrix* $\mathcal{T}$ gegeben:

Definition 1.4 (Übergangsmatrix) *Sei $(\Omega, \mathscr{A}, P)$ ein Wahrscheinlichkeitsraum. Sei Z ein Zustandsraum, $Z^+ := Z \cup \{B\}$ und $X_0, X_1, \ldots$ eine homogene Markov-Kette 1.Ordnung auf Ω mit Werten in Z^+, $X_0 \equiv B$. Dann definiert*

$$t_{z_k,z_l} := P(X_{i+1} = z_l \,|\, X_i = z_k), \quad z_k, z_l \in Z^+, \tag{1.5}$$

für alle $i \in \mathbb{N}_0$ die Wahrscheinlichkeit vom Zustand z_k in den Zustand z_l zu gelangen. Wir nennen t_{z_k,z_l} die Übergangswahrscheinlichkeit *von z_k nach z_l und*
$\mathcal{T} := (t_{z_k,z_l})_{z_k,z_l \in Z^+}$ die Übergangsmatrix.

Eine endliche Folge von Zuständen in Z^+ nennen wir einen *Pfad*. Für einen Pfad $x[1,r]$ bezeichnet $|x| = r$ die Länge des Pfades.
Z^r bzw. $Z^{+,r}$ sei die Menge aller Pfade der Länge r mit Zuständen in Z bzw. Z^+.
Z^* sei die Menge aller Pfade beliebiger Länge mit Zuständen in Z.

Wir betrachten nun Zustände, die entsprechend einer gegebenen Wahrscheinlichkeitsverteilung ein Symbol aus Σ erzeugen. D.h. jeder Pfad $X[1,r]$ mit Zuständen in Z erzeugt eine Folge von Symbolen aus Σ.

Definition 1.5 (Emissionsmatrix) *Sei $(\Omega, \mathscr{A}, P)$ ein Wahrscheinlichkeitsraum. Sei Z ein Zustandsraum, $X_0, X_1, \ldots$ eine homogene Markov-Kette 1.Ordnung auf Ω mit Werten in Z^+, und $Y_0, Y_1, \ldots$ eine Folge von Zufallsvariablen auf Ω mit Werten im Alphabet Σ. Dann definieren wir für alle $i \in \mathbb{N}_0$*

$$e_{z_k,\sigma_j} := P(Y_i = \sigma_j \,|\, X_i = z_k), \quad z_k \in Z^+, \sigma_j \in \Sigma. \tag{1.6}$$

Dies ist die Wahrscheinlichkeit, dass der Zustand z_k zum Zeitpunkt i das Symbol σ_j erzeugt. Wir sagen auch, dass der Zustand z_k das Symbol σ_j emittiert *und nennen e_{z_k,σ_j} die* Emissionswahrscheinlichkeit *von σ_j in z_k.*
Die Matrix $\mathcal{E} := (e_{z_k,\sigma_j})_{z_k \in Z^+, \sigma_j \in \Sigma}$ wird Emissionsmatrix *genannt.*

Aufgrund dieser Bezeichnungen wird das Alphabet Σ auch *Emissionsalphabet* genannt.

Der Start-Zustand B sei ein Zustand, der kein Symbol aus Σ emittiert, d.h. $e_{B,\sigma_j} = 0$, $\forall \sigma_j \in \Sigma$. Einen Zustand der kein Symbol aus Σ emittiert, nennen wir einen *stummen Zustand*. Für diesen Fall führen wir den Sonderbuchstaben ϵ ein. Ist ein Zustand z ein stummer Zustand, dann sagen wir, z emittiert mit Wahrscheinlichkeit 1 das Symbol ϵ. Wir definieren $\Sigma^+ := \Sigma \cup \{\epsilon\}$ und $\Sigma^{+,l} := (\Sigma \cup \{\epsilon\})^l$. Im Folgenden sei $\sigma_0 = \epsilon$.
Der Zustandsraum Z kann ebenfalls stumme Zustände enthalten:

Definition 1.6 (stummer Zustand) *Sei Z ein Zustandsraum und $Z^+ := Z \cup \{B\}$. Sei $X_0, X_1, \ldots$ eine homogene Markov-Kette 1.Ordnung mit Werten in Z^+, und $Y_0, Y_1, \ldots$ eine Folge von Zufallsvariablen mit Werten in Σ^+. Es sei*

$$e_{z,\epsilon} := P(Y_i = \epsilon \,|\, X_i = z), \quad z \in Z^+.$$

Dann wird ein Zustand $z \in Z$ stummer *Zustand genannt, wenn gilt*

$$e_{z,\sigma_j} = 0, \quad \forall \sigma_j \in \Sigma. \tag{1.7}$$

D.h. für einen stummen Zustand $z \in Z^+$ gilt $e_{z,\epsilon} = 1$.
Es sei

$$Z' := \{z_k \in Z \,|\, e_{z_k,\sigma_j} = 0, \forall \sigma_j \in \Sigma\} \subset Z \tag{1.8}$$

die Menge aller stummen Zustände in Z.

Zusätzlich zum Start-Zustand B führen wir einen, ebenfalls stummen, *End-Zustand* E ein. Wird dieser Zustand einmal angenommen, kann er nicht mehr verlassen werden, d.h. $t_{E,E} = 1$.

Definition 1.7 (Hidden-Markov-Modell)
Sei $Z := \{z_1, \ldots, z_m\}$ ein Zustandsraum und Z' die Menge aller stummen Zustände in Z. Sei B der Start-Zustand und E der End-Zustand. Es sei $Z^+ := Z \cup \{B, E\}$.

Ein Hidden-Markov-Modell (HMM) *mit Parametern* $M := (\Sigma^+, Z^+, \mathcal{T}, \mathcal{E})$ *ist eine endliche Folge von Zufallsvariablen*

$$X_0, Y_0, X_1, Y_1, \ldots, \tag{1.9}$$

wobei $X_0, X_1, \ldots$ *eine homogene Markov-Kette 1. Ordnung mit Werten in* Z^+, $X_0 \equiv B$, *mit Übergangsmatrix* $\mathcal{T} = (t_{z_k,z_l})_{z_k,z_l \in Z^+}$ *ist, wobei gilt*

$$\begin{aligned} t_{z_k,B} &= 0, \quad \forall z_k \in Z^+, && (1.10)\\ t_{B,E} &= 0, && (1.11)\\ t_{E,E} &= 1, && (1.12)\\ t_{z_k,E} &> 0, \quad \textit{für mind. ein } z_k \in Z, && (1.13) \end{aligned}$$

und $Y_0, Y_1, \ldots$ *eine Folge von Zufallsvariablen mit Werten in* Σ^+, $Y_0 \equiv \epsilon$, *mit Emissionsmatrix* $\mathcal{E} = (e_{z_k,\sigma_j})_{z_k \in Z^+, \sigma_j \in \Sigma^+}$, *wobei gilt*

$$\begin{aligned} e_{z_k,\sigma_j} &= P(Y_i = \sigma_j \mid X_s = z_k)\\ &= P(Y_i = \sigma_j \mid X[0,s) = x[0,s), X_s = z_k, Y[0,i) = y[0,i)), \end{aligned} \tag{1.14}$$

für alle $i, s > 0$, $x_0 = B$, $y_0 = \epsilon$, $z_k, x_1, \ldots, x_{s-1} \in Z$, $\sigma_j, y_1, \ldots, y_{i-1} \in \Sigma^+$, *und*

$$\begin{aligned} e_{z_k,\epsilon} &= \begin{cases} 0, & \forall z_k \in Z \backslash Z',\\ 1, & \forall z_k \in Z' \cup \{B, E\}, \end{cases} && (1.15)\\ e_{z_k,\sigma_j} &= 0, \quad \forall z_k \in Z' \cup \{B, E\}, \sigma_j \in \Sigma. && (1.16) \end{aligned}$$

Ein HMM erzeugt also eine Folge $Y[0,r] = y[0,r]$ von Emissionen in Σ^+ und einen Pfad $X[0,r] = x[0,r]$ mit Zuständen in Z^+, der dieser Folge von Emissionen zugrundeliegt.
Ein bestimmtes Symbol aus Σ kann jedoch von verschiedenen Zuständen aus Z emittiert werden. D.h. eine Folge von Emissionen kann nicht eindeutig einem ihr zugrundeliegenden Pfad zugeordnet werden. Der zugrundeliegende Pfad bleibt also *verborgen* (engl. *hidden*), wovon sich der Name *Hidden-Markov-Modell* ableitet. Mithilfe der folgenden Sätze bzw. Algorithmen können wir jedoch den der Folge $Y[0,r] = y[0,r]$ am wahrscheinlichsten zugrundeliegenden Pfad bestimmen.

Definition 1.8 *Sei* $Y_0, Y_1, \ldots, Y_r$, $r \geq 0$, *eine Folge von Emissionen. Dann definieren wir*

$$S_r := Y_0 Y_1 \ldots Y_r \tag{1.17}$$

S_r *ist die Verkettung der Folge von Emissionen* $Y[0,r]$ *und wird Beobachtung genannt.*

Da $Y_0 \equiv \epsilon$, gilt für $r \geq 1$ $S_r = Y_0 Y_1 \ldots Y_r = Y_1 \ldots Y_r$.

Lemma 1.9 *Sei $y[1,r]$ eine Folge von Emissionen in $\Sigma^{+,r}$ und $x[1,r]$ ein Pfad in Z^r. Sei $x_0 := B$. Dann ist die Wahrscheinlichkeit, dass $y[1,r]$ und $x[1,r]$* gemeinsam *durch den in (1.9) definierten Prozess erzeugt werden, gegeben durch*

$$P\big(X[1,r] = x[1,r], Y[1,r] = y[1,r]\big) = \prod_{i=1}^{r} t_{x_{i-1},x_i} e_{x_i,y_i}. \tag{1.18}$$

Beweis Da $P\big(X_1 = x_1\big) = P\big(X_1 = x_1 \,\big|\, X_0 = B\big) = t_{B,x_1} = t_{x_0,x_1}$,
$X_1, X_2, \ldots$ eine homogene Markov-Kette 1. Ordnung ist, und aufgrund von (1.14) gilt:

$$\begin{aligned}
& P\big(X[1,r] = x[1,r], Y[1,r] = y[1,r]\big) \\
= \; & P\big(Y[1,r] = y[1,r] \,\big|\, X[1,r] = x[1,r]\big) \cdot P\big(X[1,r] = x[1,r]\big) \\
= \; & \Big(\prod_{i=1}^{r} P\big(Y_i = y_i \,|\, X_i = x_i\big)\Big) \cdot \Big(P\big(X_1 = x_1\big) \cdot \prod_{i=1}^{r-1} P\big(X_{i+1} = x_{i+1} \,|\, X_i = x_i\big)\Big) \\
= \; & \prod_{i=1}^{r} e_{x_i,y_i} \cdot t_{x_0,x_1} \prod_{i=1}^{r-1} t_{x_i,x_{i+1}} \\
= \; & \prod_{i=1}^{r} t_{x_{i-1},x_i} e_{x_i,y_i}.
\end{aligned}$$

□

Die gemeinsame Wahrscheinlichkeit einer Beobachtung $s[1,l]$ in Σ^l, d.h. einer Folge von Emissionen in Σ, und eines Pfades $x[1,r]$ kann auf ähnliche Weise berechnet werden:

Definition 1.10 *Sei $z \in Z^+$. Dann definiert*

$$d_z := \begin{cases} 1, & \textit{falls} \quad z \in Z \backslash Z', \\ 0, & \textit{falls} \quad z \in Z' \cup \{B, E\}, \end{cases} \tag{1.19}$$

die Anzahl der Symbole aus Σ, die ein Zustand emittiert.

Definition 1.11 *Sei $x[1,r]$ ein Pfad in Z^r. Dann definiert*

$$D_{k,j} := \sum_{i=k}^{j} d_{x_i} \quad \forall \quad k, j = 1, \ldots, r, k \le j, \tag{1.20}$$

die Anzahl der Symbole aus Σ, die der Teilpfad $x[k,j]$ von $x[1,r]$ emittiert.

Lemma 1.12 *Sei $s[1,l]$ eine Beobachtung in Σ^l und $x[1,r]$ ein Pfad in Z^r. Sei $x_0 := B$.*

Dann ist die Wahrscheinlichkeit, dass $s[1,l]$ und $x[1,r]$ gemeinsam durch den in (1.9) definierten Prozess erzeugt werden, gegeben durch

$$\begin{aligned} & P\big(X[1,r] = x[1,r], S_r = s[1,l]\big) \\ = & \begin{cases} \prod_{i=1}^{r} t_{x_{i-1},x_i} e_{x_i,s[D_{1,i},D_{1,i}+d_{x_i})}, & \textit{falls } D_{1,r} = l, \\ 0, & \textit{falls } D_{1,r} \neq l. \end{cases} \end{aligned} \tag{1.21}$$

Beweis
Sei $D_{1,r} \neq l$ und $|S_r|$ die Länge der Beobachtung S_r.

$$\Rightarrow |S_r| = |Y_1 \dots Y_r| = D_{1,r} \neq l$$

$$\Rightarrow P(X[1,r] = x[1,r], S_r = s[1,l]) = 0.$$

da ein Pfad $x[1,r]$ der $D_{1,r} \neq l$ Symbole aus Σ erzeugt, keine Beobachtung der Länge l erzeugen kann.
Sei $D_{1,r} = l$.
Sei $s_0 := \epsilon$. Sei $y_1, ..., y_r$ eine Folge von Emissionen in $\Sigma^{+,l}$ mit

$$y_i := s[D_{1,i}, D_{1,i} + d_{x_i}), \qquad i = 1, \dots, r. \tag{1.22}$$

Dann gilt

$$\begin{aligned} y_1 \dots y_r &= s[D_{1,1}, D_{1,1} + d_{x_1}) \dots s[D_{1,r-1}, D_{1,r-1} + d_{x_{r-1}}) s[D_{1,r}, D_{1,r} + d_{x_r}) \\ &= s[d_{x_1}, d_{x_1} + d_{x_1}) \dots s[l - d_{x_r}, l - d_{x_r} + d_{x_{r-1}}) s[l, l + d_{x_r}) \\ &= s_1 \dots s_l, \qquad \text{da } D_{1,r} = l. \end{aligned}$$

$$\begin{aligned} \Rightarrow P(X[1,r] = x[1,r], S_r = s[1,l]) &= P(X[1,r] = x[1,r], Y[1,r] = y[1,r]) \\ &= \prod_{i=1}^{r} t_{x_{i-1},x_i} e_{x_i,y_i} \\ &= \prod_{i=1}^{r} t_{x_{i-1},x_i} e_{x_i,s[D_{1,i},D_{1,i}+d_{x_i})}. \end{aligned}$$

□

$P(X[1,r] = x[1,r], S_r = s[1,l])$ ist die Wahrscheinlichkeit, eine Beobachtung $s[1,l]$ und einen bestimmten Pfad $x[1,r]$ *gemeinsam* durch das HMM zu erzeugen. Eine Beobachtung kann jedoch durch verschiedene Pfade erzeugt werden. Wir wollen nun die Wahrscheinlichkeit berechnen, $s[1,l]$ durch das HMM zu erzeugen. Dies ist mithilfe des *Vorwärts-Algorithmus* (Abschnitt 1.2) möglich. Dazu führen wir zunächst einige Notationen und Definitionen ein.

Notation 1.13 Es sei

$$T := \{(z_k, z_l) \in Z \times Z \,|\, t_{z_k,z_l} > 0\} \tag{1.23}$$

die Menge aller Paare von Zuständen $(z_k, z_l) \in Z \times Z$ mit einer Übergangswahrscheinlichkeit t_{z_k,z_l} grösser als 0. Wir nennen T die Menge aller möglichen Übergänge zwischen Zuständen in Z.
Es sei

$$T' := \{(z_k, z_l) \in T \,|\, z_l \in Z'\} \subset T \tag{1.24}$$

die Menge aller Übergänge, die in einem stummen Zustand in Z enden.

Definition 1.14 (Zustandsgraph und Sub-Zustandsgraph) *Sei Z ein Zustandsraum. Seien T und T' definiert wie in Notation 1.13.*
Der Zustandsgraph *einer homogenen Markov-Kette 1.Ordnung $X_1, X_2, \ldots$ mit Werten in Z und Übergangsmatrix $\mathcal{T}$ ist definiert als der gerichtete Graph $G := (Z, T)$* [5]. *D.h. Z ist die Menge aller Knoten und T die Menge aller Kanten von G.*
Den gerichteten Graphen $G' := (Z, T')$, der nur die Kanten von G enthält, die in einem stummen Zustand enden, nennen wir den Sub-Zustandsgraphen *der Markov-Kette.*

Bemerkung 1.15 Sei $G := (V, E)$ ein azyklischer, gerichteter Graph, wobei V die Menge aller Knoten und E die Menge aller Kanten von G sei. Dann können alle Knoten $v \in V$ von G so linear angeordnet werden, dass für jede Kante (u, v) von G, $u, v \in V$, der Knoten u in der linearen Anordnung der Knoten von G vor dem Knoten v erscheint. Solch eine lineare Anordnung nennt man eine *topologische Sortierung* von G [5].
Eine topologische Sortierung eines zyklischen, gerichteten Graphen ist offensichtlich nicht möglich.

Im Folgenden fordern wir, dass der Sub-Zustandsgraph $G' = (Z, T')$ azyklisch ist. Wir können also G' topologisch sortieren, d.h. wir können die Zustände des Zustandsraums Z linear anordnen, so dass jeder stumme Zustand in Z in der linearen Anordnung rechts von allen seinen Vorgängern erscheint.
Aufgrund dieser Forderung ist es möglich, den Vorwärts- (Algo. 1.21) und den Viterbi-Algorithmus [27] (Algo. 1.31) ohne grosse Abwandlung der in der Literatur, siehe z.B. [6], veröffentlichten Algorithmen durchzuführen, wenn im Zustandsraum Z stumme Zustände zugelassen sind. Näheres dazu siehe Abschnitt 1.2 und 1.3.
Ein Beispiel für einen azyklischen Sub-Zustandsgraphen $G' = (Z, T')$ eines HMMs ist im Anhang abgebildet.

1.2 Vorwärts-Algorithmus

Definition 1.16 (Vorwärts-Variable) *Sei $s[1,l]$ ein Beobachtung in Σ^l. Dann definieren wir für alle $z \in Z$ die* Vorwärts-Variablen

$$\alpha_{z,0} := P\Big(\bigcup_{j\geq 1}\{X_j = z, S_j = \epsilon\}\Big), \tag{1.25}$$

$$\alpha_{z,i} := P\Big(\bigcup_{j\geq 1}\{X_j = z, S_j = s[1,i]\}\Big), \qquad i = 1,\ldots,l. \tag{1.26}$$

$\alpha_{z,i}$, $i = 1,\ldots,l$ ist also die Wahrscheinlichkeit, auf einem beliebigen Pfad beliebiger Länge j in den Zustand z zu gelangen, und dabei das Präfix $s[1,i]$ der Beobachtung $s[1,l]$ zu erzeugen. $\alpha_{z,0}$ ist die Wahrscheinlichkeit, auf einem beliebigen Pfad beliebiger Länge in den Zustand z zu gelangen und dabei kein einziges Zeichen aus Σ zu emittieren.

Lemma 1.17 *Sei $s[1,l]$ eine Beobachtung in Σ^l. Sei $x_0 := B$. Sei der Sub-Zustandsgraph $G' = (Z, T')$ azyklisch. Dann gilt für alle $z \in Z$, $i = 1,\ldots,l$,*

$$\alpha_{z,0} = \sum_{j\geq 1} P(X_j = z, S_j = \epsilon), \tag{1.27}$$

$$\alpha_{z,i} = \sum_{j\geq 1} P(X_j = z, S_j = s[1,i]). \tag{1.28}$$

Beweis
Sei $z \in Z$ und $i \geq 1$.
Sei $X[1,r]$ ein Pfad mit $X_j = X_k = z$, $j,k \in \{1,\ldots,r\}, j < k$, so dass
$S_j = S_k = s[1,i]$ bzw. $S_j = S_k = \epsilon$.
Sei $z \in Z'$ ein stummer Zustand. Dann gilt $X[j,k] = x[j,k] \in (Z')^{k-j+1}$ mit $x_j = x_k = z$. Dies ist jedoch nicht möglich, da Zyklen von stummen Zuständen ausgeschlossen sind.
Sei $z \in Z\backslash Z'$. Dann ist für alle $j < k$ mit $X_j = X_k = z$, $S_j \neq S_k$. Dies ist ein Widerspruch zu $S_j = S_k = s[1,i]$ bzw. $S_j = S_k = \epsilon$.
Daraus folgt für alle $z \in Z$, $k, j \geq 1, k \neq j$,

$$\{X_j = z, S_j = s[1,i]\} \cap \{X_k = z, S_k = s[1,i]\} = \emptyset,$$

und

$$\{X_j = z, S_j = \epsilon\} \cap \{X_k = z, S_k = \epsilon\} = \emptyset.$$

Daraus folgt

$$\alpha_{z,0} = P\Big(\bigcup_{j\geq 1}\{X_j = z, S_j = \epsilon\}\Big) = \sum_{j\geq 1} P(X_j = z, S_j = \epsilon)$$

Für $\alpha_{z,i}, i = 1,\ldots,l$, lässt sich dies analog beweisen. □

Lemma 1.18 *Sei $s[1,l]$ eine Beobachtung in Σ^l. Sei $x_0 := B$ und $\alpha_{B,0} := 1$. Sei $G' = (Z, T')$ azyklisch.*
Dann gilt für alle $z \in Z$

$$\alpha_{z,0} = \begin{cases} \sum_{z' \in Z' \cup \{B\}} \alpha_{z',0} t_{z',z} & , falls\ z \in Z', \\ 0 & , falls\ z \in Z \backslash Z', \end{cases} \tag{1.29}$$

$$\alpha_{z,i} = e_{z,s[i,i+d_z)} \sum_{z' \in Z \cup \{B\}} \alpha_{z',i-d_z} t_{z',z} \qquad , i = 1, \ldots, l. \tag{1.30}$$

Beweis:
Sei Z' die Menge aller stummen Zustände in Z.
Sei $z \in Z \backslash Z'$. Dann gilt

$$\alpha_{z,0} = \sum_{j \geq 1} P(X_j = z, S_j = \epsilon) = 0, \quad \text{da } S_j \neq \epsilon \text{ für } X_j = z \text{ mit } z \in Z \backslash Z'.$$

Sei $z \in Z'$. Dann gilt

$$\begin{aligned}
\alpha_{z,0} &= \sum_{j \geq 1} P(X_j = z, S_j = \epsilon) \\
&= \sum_{j \geq 1} \sum_{z' \in Z \cup \{B\}} P(X_j = z, Y_j = \epsilon, X_{j-1} = z', S_{j-1} = \epsilon) \\
&= \sum_{z' \in Z \cup \{B\}} t_{z',z} e_{z,\epsilon} \sum_{j \geq 1} P(X_{j-1} = z', S_{j-1} = \epsilon) \\
&= \sum_{z' \in Z \cup \{B\}} t_{z',z} \big(P(X_0 = z', S_0 = \epsilon) + \sum_{j \geq 2} P(X_{j-1} = z', S_{j-1} = \epsilon) \big) \\
&= \sum_{z' \in Z \cup \{B\}} t_{z',z} \big(P(X_0 = z', S_0 = \epsilon) + \sum_{j \geq 1} P(X_j = z', S_j = \epsilon) \big) \\
&= \sum_{z' \in Z \cup \{B\}} t_{z',z} \alpha_{z',0} \\
&= \sum_{z' \in Z' \cup \{B\}} t_{z',z} \alpha_{z',0},
\end{aligned}$$

da $X_0 \equiv B$, $P(X_j = B, S_j = \epsilon) = 0$, $\forall j \geq 1$, $\alpha_{B,0} = 1$ und $\alpha_{z',0} = 0$ für $z \in Z \backslash Z'$.
Sei $z \in Z$. Dann gilt

$$\begin{aligned}
\alpha_{z,i} &= \sum_{j \geq 1} P(X_j = z, S_j = s[1,i]) \\
&= \sum_{j \geq 1} \sum_{z' \in Z \cup \{B\}} P(X_j = z, Y_j = s[i, i+d_z), X_{j-1} = z', S_{j-1} = s[1, i-d_z])
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{z' \in Z \cup \{B\}} t_{z',z} e_{z,s[i,i+d_z)} \sum_{j \geq 1} P(X_{j-1} = z', S_{j-1} = s[1, i - d_z]) \\
&= \sum_{z' \in Z \cup \{B\}} t_{z',z} e_{z,s[i,i+d_z)} \cdot \\
&\qquad \cdot \big(P(X_0 = z', S_0 = \epsilon) + \sum_{j \geq 1} P(X_j = z', S_j = s[1, i - d_z]) \big) \\
&= e_{z,s[i,i+d_z)} \sum_{z' \in Z \cup \{B\}} t_{z',z} \alpha_{z',i-d_z}
\end{aligned}
$$

da $X_0 \equiv B$, $P(X_j = B, S_j = s[1, i - d_z]) = 0$, $\forall j \geq 1$, und $\alpha_{B,0} = 1$.

Definition 1.19 (Vorwärts-Wahrscheinlichkeit) *Sei $s[1,l]$ eine Beobachtung in Σ^l. Dann definieren wir die* Vorwärts-Wahrscheinlichkeit

$$\alpha_{*,l} := P\Big(\bigcup_{j \geq 1} X_{j+1} = E, S_j = s[1,l] \Big) \tag{1.31}$$

Die Vorwärts-Wahrscheinlichkeit $\alpha_{*,l}$ ist also die Wahrscheinlichkeit, die Beobachtung $s[1,l]$ durch das HMM zu erzeugen.

Lemma 1.20 *Es gilt*

$$\alpha_{*,l} = \sum_{z \in Z} \alpha_{z,l} \cdot t_{z,E} \tag{1.32}$$

Beweis

$$
\begin{aligned}
\alpha_{*,l} &= P\Big(\bigcup_{j \geq 1} X_{j+1} = E, S_j = s[1,l] \Big) \\
&= \sum_{j \geq 1} P\big(X_{j+1} = E, S_j = s[1,l] \big) \qquad \text{(siehe Beweis zu Lemma 1.17)} \\
&= \sum_{j \geq 1} \sum_{z \in Z} P\big(X_{j+1} = E, X_j = z, S_j = s[1,l] \big) \\
&= \sum_{z \in Z} t_{z,E} \sum_{j \geq 1} P\big(X_j = z, S_j = s[1,l] \big) \\
&= \sum_{z \in Z} t_{z,E} \alpha_{z,l}
\end{aligned}
$$

□

Algorithmus 1.21 (Vorwärts-Algorithmus)
Sei der Zustandsraum $Z = \{z_1, \ldots, z_m\}$ entsprechend des topologisch sortierten Sub-Zustandsgraphen $G' = (Z, T')$ sortiert. D.h. für alle $z_i, z_j \in Z$ gelte

$$i < j, \quad \text{falls} \quad t_{z_i,z_j} > 0 \text{ und } z_j \in Z'. \tag{1.33}$$

Sei $z_0 := B$. Dann lässt sich die Vorwärts-Wahrscheinlichkeit mithilfe des folgenden *Vorwärts-Algorithmus* berechnen:

Erzeuge das Feld $\alpha_{z,i}, z \in Z \cup \{B\}, i = 0, \ldots, l$.

1. Initialisierung:

$$\alpha_{B,0} = 1,$$
$$\alpha_{z,0} = 0, \quad \forall z \in Z \backslash Z',$$

für $j = 1, \ldots, m, z_j \in Z'$ berechne

$$\alpha_{z_j,0} = \sum_{z_k \in Z' \cup \{B\}, k<j} \alpha_{z_k,0} t_{z_k,z_j}$$

2. Rekursion:

für $i = 1, \ldots, l$,
  für $j = 1, \ldots, m$ berechne

$$\alpha_{z_j,i} = e_{z_j,s[i,i+d_{z_j})} \sum_{z_k \in Z \cup \{B\}, k<j} \alpha_{z_k,i-d_{z_j}} t_{z_k,z_j}$$

3. Termination:

berechne

$$\alpha_{*,l} = \sum_{z \in Z} \alpha_{z,l} t_{z,E}$$

Bemerkung 1.22 Für einen stummen Zustand $z \in Z'$ gilt $\alpha_{z,i} = \sum_{z' \in Z} \alpha_{z',i} t_{z',z}$, da $d_z = 0$ für alle $z \in Z'$. Ist der Zustandsraum Z entsprechend der Reihenfolge der Zustände im topologisch sortierten Sub-Zustandsgraphen G' sortiert (siehe 1.33), und werden die Vorwärts-Variablen aller Zustände für alle $i \geq 0$ in dieser Reihenfolge berechnet, ist sichergestellt, dass die Vorwärts-Variablen $\alpha_{z',i}$ aller Vorgänger z' von $z \in Z'$ mit $t_{z',z} > 0$ bereits berechnet wurden, und im Algorithmus nicht auf einen noch nicht berechneten Wert zugegriffen wird.
Für einen nicht-stummen Zustand $z \in Z \backslash Z'$ ergibt sich kein Problem in der Berechnung der Vorwärts-Variablen, da $d_z = 1$ für alle $z \in Z \backslash Z'$, und $\alpha_{z',i-1}$ für alle Vorgänger $z' \in Z$ bereits im vorherigen Schritt berechnet wurde.

Folgerung 1.23 (Laufzeit) *Sei $s[1, l]$ eine Beobachtung der Länge l und m die Anzahl aller Zustände in Z. Dann lassen sich sämtliche Vorwärts-Variablen, die Vorwärts-Wahrscheinlichkeit und somit die Wahrscheinlichkeit der Beobachtung s mithilfe des Vorwärts-Algorithmus in Zeit $O(m^2 l)$ berechnen.*

1.3 Viterbi-Algorithmus

Eine Beobachtung $s[1,l]$ kann durch verschiedene Pfade durch das HMM erzeugt werden. Die Wahrscheinlichkeit mit der jeder dieser Pfade die Beobachtung $s[1,l]$ erzeugt, kann jedoch unterschiedlich hoch sein. Unter allen Pfaden, die die Beobachtung erzeugen, suchen wir nun einen Pfad $v[1,r]$, für den diese Wahrscheinlichkeit maximal wird. Dabei ist zu beachten, dass es verschiedene Pfade durch das HMM geben kann, die dies erfüllen. Jeder dieser Pfade ist der wahrscheinlichste der Beobachtung zugrundeliegende Pfad und wird *Viterbi-Pfad* genannt. Für eine Beobachtung kann es also verschiedene Viterbi-Pfade geben. Unser Ziel ist es, für eine gegebene Beobachtung einen der Viterbi-Pfade zu finden.

Definition 1.24 (Viterbi-Pfad) *Sei $s[1,l]$ eine Beobachtung in Σ^l.*
Für einen Pfad $x \in Z^*$ sei l_x die Länge von x.
Ein Viterbi-Pfad *der Beobachtung $s[1,l]$ ist ein Pfad $v^* \in Z^*$ für den gilt:*

$$v^* \in \arg\max_{x \in Z^*} P\big(X[1,l_x] = x[1,l_x], X_{l_x+1} = E \,\big|\, S_{l_x} = s[1,l]\big). \tag{1.34}$$

Ein Viterbi-Pfad v^* einer Beobachtung $s[1,l]$ kann mithilfe eines Algorithmus berechnet werden, der dem Vorwärts-Algorithmus sehr ähnlich ist. Dazu definieren wir zunächst die *Viterbi-Variablen* und die *Viterbi-Wahrscheinlichkeit*:

Definition 1.25 (Viterbi-Variable) *Sei $s[1,l]$ ein Beobachtung in Σ^l.*
Sei $x_0 := B$. Dann definieren wir für alle $z \in Z$ die Viterbi-Variablen

$$\delta_{z,0} := \max_{j \geq 1} \max_{x[0,j) \in Z^*} P(X_j = z, X[0,j) = x[0,j), S_j = \epsilon), \tag{1.35}$$

$$\delta_{z,i} := \max_{j \geq 1} \max_{x[0,j) \in Z^*} P(X_j = z, X[0,j) = x[0,j), S_j = s[1,i]), \tag{1.36}$$

$i = 1, \ldots, l$.

Lemma 1.26 *Sei $s[1,l]$ eine Beobachtung in Σ^l. Sei $x_0 := B$, $\delta_{B,0} := 1$ und $\delta_{B,i} = 0$, $\forall i \geq 1$. Dann gilt für alle $z \in Z$*

$$\delta_{z,0} = \begin{cases} 0 & , falls\ z \in Z \backslash Z', \\ \max\limits_{z' \in Z' \cup \{B\}} \delta_{z',0} t_{z',z} & , falls\ z \in Z', \end{cases} \tag{1.37}$$

$$\delta_{z,i} = e_{z,s[i,i+d_z)} \max_{z' \in Z \cup \{B\}} \delta_{z',i-d_z} t_{z',z}, \qquad i = 1, \ldots, l. \tag{1.38}$$

Beweis:
Sei Z_B^* die Menge aller Pfade beliebiger Länge mit Start im Zustand B. Sei $x_0 := B$. Sei $z \in Z \backslash Z'$. Da $S_j \neq \epsilon$ für $X_j = z$ mit $z \in Z \backslash Z'$, gilt

$$\delta_{z,0} = \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} P(X_j = z, X[0,j) = x[0,j), S_j = \epsilon) = 0.$$

Sei $z \in Z'$. Dann gilt

$$\begin{aligned}
& \delta_{z,0} \\
= \; & \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} P\big(X_j = z, X[0,j) = x[0,j), S_j = \epsilon\big) \\
= \; & \max \Big\{ P\big(X_1 = z, X_0 = x_0, S_1 = \epsilon\big), \\
& \qquad \max_{j \geq 2} \max_{x[0,j) \in Z_B^*} P\big(X_j = z, X[0,j) = x[0,j), S_j = \epsilon\big) \Big\} \\
= \; & \max \Big\{ t_{B,z} e_{z,\epsilon} \,, \max_{j \geq 2} \max_{x[0,j-1) \in Z_B^*} \\
& \max_{x_{j-1} \in Z} P\big(X_j = z, Y_j = \epsilon, X_{j-1} = x_{j-1}, X[0,j-1) = x[0,j-1), S_{j-1} = \epsilon\big) \Big\} \\
= \; & \max \Big\{ t_{B,z} \,, \max_{z' \in Z} t_{z',z} e_{z,\epsilon} \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} P\big(X_j = z', X[0,j) = x[0,j), S_j = \epsilon\big) \Big\} \\
= \; & \max \Big\{ t_{B,z} \,, \max_{z' \in Z} t_{z',z} \delta_{z',0} \Big\} \\
= \; & \max_{z' \in Z' \cup \{B\}} t_{z',z} \delta_{z',0},
\end{aligned}$$

da $\delta_{B,0} = 1$ und $\delta_{z',0} = 0$ für $z \in Z \backslash Z'$.
Sei $z \in Z$. Dann gilt

$$\begin{aligned}
& \delta_{z,i} \\
= \; & \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} P\big(X_j = z, X[0,j) = x[0,j), S_j = s[1,i]\big) \\
= \; & \max \Big\{ t_{B,z} e_{z,s[i,i+d_z)} \cdot P\big(X_0 = x_0, S_0 = s[1, i - d_z]\big) \,, \max_{j \geq 2} \max_{x[0,j-1) \in Z_B^*} \\
& \max_{z' \in Z} t_{z',z} e_{z,s[i,i+d_z)} \cdot P\big(X_{j-1} = z', X[0,j-1) = x[0,j-1), S_{j-1} = s[1, i - d_z]\big) \Big\} \\
= \; & e_{z,s[i,i+d_z)} \cdot \max \Big\{ t_{B,z} \delta_{B,i-d_z} \,, \\
& \max_{z' \in Z} t_{z',z} \cdot \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} P\big(X_j = z', X[0,j) = x[0,j), S_j = s[1, i - d_z]\big) \Big\} \\
= \; & e_{z,s[i,i+d_z)} \cdot \max \Big\{ t_{B,z} \delta_{B,i-d_z} \,, \max_{z' \in Z} t_{z',z} \delta_{z',i-d_z} \Big\} \\
= \; & e_{z,s[i,i+d_z)} \cdot \max_{z' \in Z \cup \{B\}} t_{z',z} \delta_{z',i-d_z}
\end{aligned}$$

□

Definition 1.27 (Viterbi-Wahrscheinlichkeit) *Sei $s[1,l]$ eine Beobachtung in Σ^l. Für einen Pfad $x \in Z^*$ sei l_x die Länge von x.*
Dann definieren wir die Viterbi-Wahrscheinlichkeit

$$\delta_{*,l} := \max_{x[1,l_x] \in Z^*} P\big(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_{l_x} = s[1,l]\big) \tag{1.39}$$

Lemma 1.28 *Es gilt*

$$\delta_{*,l} = \max_{z \in Z} \delta_{z,l} \cdot t_{z,E} \tag{1.40}$$

Beweis
Sei Z_B^* die Menge aller Pfade beliebiger Länge mit Start im Zustand B. Sei $x_0 := B$. Es gilt

$$\begin{aligned}
\delta_{*,l} &= \max_{x \in Z^*} P\big(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_{l_x} = s[1,l]\big) \\
&= \max_{x \in Z^*} P\big(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_{l_x} = s[1,l] \,\big|\, X_0 = B\big) \cdot P(X_0 = B) \\
&= \max_{x \in Z_B^*} P\big(X[0,l_x] = x[0,l_x], X_{l_x+1} = E, S_{l_x} = s[1,l]\big) \\
&= \max_{j \geq 1} \max_{x[0,j] \in Z_B^*} P\big(X[0,j] = x[0,j], X_{j+1} = E, S_j = s[1,l]\big) \\
&= \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} \max_{x_j \in Z} t_{x_j,E} \cdot P\big(X[0,j) = x[0,j), S_j = s[1,l]\big) \\
&= \max_{z \in Z} t_{z,E} \cdot \max_{j \geq 1} \max_{x[0,j) \in Z_B^*} P\big(X[0,j) = x[0,j), S_j = s[1,l]\big) \\
&= \max_{z \in Z} t_{z,E} \cdot \delta_{z,l},
\end{aligned}$$

da $P(X_1 = x_1 \,|\, X_0 = B) = P(X_1 = x_1)$ und $P(X_0 = B) = 1$.

Satz 1.29 *Sei $s[1,l]$ eine Beobachtung in Σ^+.*
Jeder Pfad $v := v[1,r] \in Z^r$ für den gilt

$$\delta_{v_r,l} t_{v_r,E} = \delta_{*,l} \qquad \textit{und} \tag{1.41}$$

$$\delta_{v_j,i} = e_{v_j,s[i,i+d_{v_j})} \delta_{v_{j-1},i-d_{v_j}} t_{v_{j-1},v_j}, \quad \forall j = 1,\ldots,r, i = 1,\ldots,l \tag{1.42}$$

ist ein Viterbi-Pfad.

Beweis

$$\begin{aligned}
& \text{Ein Pfad } v \in Z^* \text{ ist ein Viterbi-Pfad} \\
\Leftrightarrow\ & v \in \arg\max_{x \in Z^*} P(X[1,l_x] = x[1,l_x], X_{l_x+1} = E \,|\, S_r = s[1,l]) \\
\Leftrightarrow\ & v \in \arg\max_{x \in Z^*} \frac{P(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_r = s[1,l])}{P(S_r = s[1,l])} \\
\Leftrightarrow\ & v \in \arg\max_{x \in Z^*} P(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_r = s[1,l])
\end{aligned}$$

Sei nun $v := v[1,r] \in Z^r$ ein Pfad der Länge r, für den die Bedingungen (1.41) und (1.42) erfüllt sind. Sei $v_0 := B$, $s_0 := \epsilon$. Dann gilt

$$\begin{aligned}
& \max_{x \in Z^*} P\big(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_r = s[1,l]\big) \\
=\ & \delta_{*,l}
\end{aligned}$$

$$
\begin{aligned}
&= \delta_{v_r,l} t_{v_r,E} \\
&= e_{v_r,s[l,l+d_{v_r})} \delta_{v_{r-1},l-d_{v_{r-1}}} t_{v_{r-1},v_r} t_{v_r,E} \\
&= e_{v_r,s[l,l+d_{v_r})} \cdot \big(e_{v_{r-1},s[l-d_{v_r},l-d_{v_r}+d_{v_{r-1}})} \delta_{v_{r-2},(l-d_{v_r}-d_{v_{r-1}})} t_{v_{r-2},v_{r-1}} \big) \\
&\quad \cdot t_{v_{r-1},v_r} t_{v_r,E} \\
&\vdots \\
&= \prod_{j=r}^{0} e_{v_j,s[l-D_{j+1,r},l-D_{j+1,r}+d_{v_j})} \prod_{j=0}^{r-1} t_{v_j,v_{j+1}} t_{v_r,E} \\
&= e_{v_0,s[l-D_{1,r},l-D_{1,r}+d_{v_0})} \prod_{j=1}^{r} e_{v_j,s[l-D_{j+1,r},l-D_{j+1,r}+d_{v_j})} t_{v_{j-1},v_j} t_{v_r,E} \\
&= e_{v_0,s[l-D_{1,r},l-D_{1,r}+d_{v_0})} P\big(X[1,r] = v[1,r], S_r = s[1,l]\big) t_{v_r,E} \quad \text{nach (1.23)} \\
&= \begin{cases} e_{B,\epsilon} \prod_{i=1}^{r} t_{x_{i-1},x_i} e_{x_i,s[D_{1,i},D_{1,i}+d_{x_i})} t_{v_r,E}, & \text{falls } D_{1,r} = l \\ 0, & \text{falls } D_{1,r} \neq l. \end{cases} \quad \text{nach (1.21)} \\
&= P\big(X_{r+1} = E \,\big|\, X_r = v_r\big) P\big(X[1,r] = v[1,r], S_r = s[1,l]\big) \\
&= P\big(X[1,r] = v[1,r], X_{r+1} = E, S_r = s[1,l]\big).
\end{aligned}
$$

$\Rightarrow v \in \arg\max_{x \in Z^*} P(X[1,l_x] = x[1,l_x], X_{l_x+1} = E, S_r = s[1,l])$ □

Verfolgen wir diesen Pfad v zurück (engl. *backtracking*), erhalten wir also einen Viterbi-Pfad der Beobachtung $s[1,l]$:

Folgerung 1.30 (backtracking) *Sei $s[1,l]$ eine Beobachtung in Σ^l. Sei $v[1,r]$ ein Viterbi-Pfad und $D_{i,r} = \sum_{j=i}^{r} d_{v_j}$, $i = 1, \ldots, r$.*
Dann folgt aus Satz 1.29

$$v_r \in \arg\max_{z \in Z} \delta_{z,l} t_{z,E} \tag{1.43}$$

und

$$
\begin{aligned}
v_i &= \arg\max_{z \in Z} e_{v_{i+1},s[l-D_{i+1,r},\, l-D_{i+1,r}+d_{v_{j+1}})} \delta_{z,l-D_{i+1,r}} t_{z,v_{i+1}} \\
&= \arg\max_{z \in Z} \delta_{z,l-D_{i+1,r}} t_{z,v_{i+1}}
\end{aligned}
$$

für $i = r-1, \ldots, 1$.

Algorithmus 1.31 (Viterbi-Algorithmus)
Sei der Zustandsraum $Z = \{z_1, \ldots, z_m\}$ entsprechend der Bedingung 1.33 sortiert. Sei $z_0 := B$. Dann lässt sich die Viterbi-Wahrscheinlichkeit mithilfe des folgenden *Viterbi-Algorithmus* berechnen:

Erzeuge das Feld $\delta_{z,i}, z \in Z \cup \{B\}, i = 0, \ldots, l$.
Vorwärtsteil:

1. Initialisierung:

$$\delta_{B,0} = 1,$$
$$\delta_{z,0} = 0, \quad \forall z \in Z \backslash Z',$$

für $j = 1, \ldots, m, z_j \in Z'$ berechne

$$\delta_{z_j,0} = \max_{z_k \in Z' \cup \{B\}, k<j} \delta_{z_k,0} t_{z_k,z_j},$$

2. Rekursion:

für $i = 1, \ldots, l$,

für $j = 1, \ldots, m$ berechne

$$\delta_{z_j,i} = e_{z_j,s[i,i+d_{z_j})} \max_{z_k \in Z \cup \{B\}, k<j} \delta_{z_k,i-d_{z_j}} t_{z_k,z_j}$$

3. Termination:

berechne

$$\delta_{*,l} = \max_{z \in Z} \delta_{z,l} t_{z,E}$$

Rückwärtsteil: (backtracking)

1. Beginn:

berechne

$$v_r = \arg\max_{z \in Z} \delta_{z,l} t_{z,E}$$

2. Rekursion:

$$\text{für } i = r-1, \ldots, 1, D_{i+1,r} = \sum_{j=i+1}^{r} d_{v_j}, \text{ berechne}$$
$$v_i = \arg\max_{z \in Z} \delta_{z,l-D_{i+1,r}} t_{z,v_{i+1}}$$

Bemerkung 1.32 Im Viterbi-Algorithmus wird, wie im Vorwärts-Algorithmus, aufgrund der Sortierung der Zustände des Zustandsraums Z entsprechend ihrer Reihenfolge im topologisch sortierten Sub-Zustandsgraphen G', für alle $i \geq 0$ nur auf bereits berechnete Variablen $\delta_{z,i}$ zugegriffen. Näheres dazu siehe Bemerkung 1.22.

Folgerung 1.33 (Laufzeit) *Sei $s[1,l]$ eine Beobachtung der Länge l und m die Anzahl aller Zustände in Z. Dann lassen sich sämtliche Viterbi-Variablen, die Viterbi-Wahrscheinlichkeit und ein Viterbi-Pfad der Beobachtung $s[1,l]$ mithilfe des Viterbi-Algorithmus in Zeit $O(m^2 l)$ berechnen.*

Kapitel 2

Profil-Hidden-Markov-Modelle

In Kapitel 1 wurde der Begriff einer Beobachtung eingeführt. Dies ist eine Folge von Elementen im Emissionsalphabet Σ. Betrachten wir nun als Emissionsalphabet das Aminosäure-Alphabet, das als Elemente alle 20 Aminosäuren hat, oder das Nukleotid-Alphabet, dessen Elemente die vier Nukleotide A (Adenin), C (Cytosin), G (Guanin) und T (Thymin) sind, so können wir jede Protein- bzw. DNA-Sequenz als eine Beobachtung $s[1,l]$ in Σ^l auffassen. l sei dabei die Länge der Sequenz.
Wir nehmen an, wir haben eine Familie von Sequenzen $\mathcal{S} := \{S_1, \ldots, S_K\}$ gegeben, z.B eine Proteinfamilie oder ein Subtyp einer bestimmten Virusgruppe, und möchten für eine Sequenz $s[1,l]$ (*Anfragesequenz*) wissen, wie ähnlich sie zu dieser Familie ist bzw. ob sie Mitglied der Sequenzfamilie ist. Eine einfache Methode dafür ist, die Ähnlichkeit der Sequenz s zu jeder einzelnen Sequenz $S_i \in S$ mithilfe eines *paarweisen Alignments* (siehe Anhang) zu bestimmen. Dadurch können jedoch nur selten entfernte Verwandtschaften zwischen der Anfragesequenz und der Sequenzfamilie festgestellt werden. Auch werden in einem paarweisen Alignment bestimmte konservierte Muster bzw. Merkmale innerhalb der Familie nicht oder nur kaum berücksichtigt. Dies ist jedoch mithilfe eines *multiplen Sequenzalignments* (*MSA*) der Familie möglich. Gehen wir davon aus, dass wir ein *korrektes* MSA der Sequenzfamilie gegeben haben, dann lassen sich anhand der Spalten des MSA konservierte Positionen bzw. Bereiche der Sequenzen sehr gut von Regionen höherer Variabilität unterscheiden.

Notation 2.1 Sei ' $-$ ' das Zeichen für eine Lücke (engl. *gap*) in einem multiplen Sequenzalignment. Dann beschreiben wir ein gegebenes multiples Sequenzalignment der Familie $\mathcal{S} = \{S_1, \ldots, S_K\}$ mit

$$A := (a_{k,j})_{1 \leq k \leq K, 1 \leq j \leq N}, \quad a_{k,j} \in \Sigma \cup \{-\}. \tag{2.1}$$

Alignieren wir die Anfragesequenz zu diesem MSA, können wir erkennen, welche der konservierten Merkmale der Sequenzfamilie in der Anfragesequenz enthalten sind,

und dadurch ihre Ähnlichkeit zur Sequenzfamilie bestimmen.
Ähnlich zu einem paarweisen Alignment, muss für dieses Alignment die Möglichkeit gegeben sein, ein Symbol der Anfragesequenz zu einer Spalte des MSA zu alignieren (*Match*) und eine Lücke in das MSA (*Insertion* eines Symbols in s) bzw. in die Anfragesequenz einzufügen (*Deletion* einer Spalte des MSA).
Im MSA kann es jedoch Spalten mit sehr vielen Lücken geben. Die in diesen Spalten auftretenden Symbole betrachten wir selbst als Insertionen und schliessen die Möglichkeit eines Matches der Anfragesequenz zu diesen Spalten aus. D.h. ein Match ist jeweils nur zu einer Spalte des MSA möglich, deren Anzahl an *Nicht-Gaps* eine bestimmte Schranke überschreitet. Solch eine Spalte nennen wir *Konsensusspalte.*

Definition 2.2 *Sei $\vec{a}_j$ der j-te Spaltenvektor in A. Dann definiert*

$$|\vec{a}_j| := \#\{a_{k,j} \,|\, a_{k,j} \neq \text{'}-\text{'}, 1 \leq k \leq K\}, \quad \forall j = 1, \ldots, N. \tag{2.2}$$

die Anzahl aller Symbole in der j-ten Spalte von A, d.h. die Anzahl aller Sequenzen, die in dieser Spalte keine Lücke aufweisen.

Definition 2.3 (Konsensusspalte) *Eine Spalte $j, j = 1, \ldots, N$, des Alignments A wird* Konsensusspalte *genannt, wenn die Bedingung*

$$|\vec{a}_j| \geq c \cdot K \tag{2.3}$$

für eine bestimmte Schranke $0 \leq c \leq 1$ erfüllt ist.
Die Schranke c nennen wir das Konsensusspaltenkriterium.

Ein geeignetes Modell für solch ein Alignment einer Anfragesequenz zu einem MSA ist ein spezieller Typ eines HMMs (Kapitel 1), ein sog. *Profil-Hidden-Markov-Modell* (*Profil-HMM*) [11]. Ein Profil-HMM ist ein HMM mit einer sich wiederholenden Struktur von Zuständen, die jeweils einer bestimmten Spalte im MSA zugeordnet sind. Die Übergangs- (Abschnitt 2.3.2) und Emissionswahrscheinlichkeiten (Abschnitt 2.3.1) der Zustände werden abhängig von der Häufigkeitsverteilung der Symbole in den Spalten, dem sog. *Profil* des MSA, geschätzt, woraus sich der Name *Profil*-HMM ableitet. Dadurch sind, im Gegensatz zu einem paarweisen Alignment, positionsspezifische Lückenkosten gegeben.

2.1 Definition eines Profil-HMMs

Sei ein multiples Sequenzalignment A einer Sequenzfamilie $\mathcal{S}$ und eine Anfragesequenz $s[1, l]$ gegeben.
Jeder Konsensusspalte j von A werden drei Zustände zugeordnet: Ein *Match*-Zustand M_j, ein *Insert*-Zustand I_j und ein *Delete*-Zustand D_j.
Ein Match-Zustand M_j emittiert jeweils ein Symbol der Anfragesequenz, wobei die

Emissionswahrscheinlichkeit von der Häufigkeit der in der Konsensusspalte j auftretenden Symbole abhängt. Er spiegelt somit die Verteilung der Symbole in der Spalte j wieder. Von jedem Match-Zustand M_j ist ein Übergang in den Match-Zustand M_k möglich, der der nächsten auf j folgenden Konsensusspalte $k > j$ in A zugeordnet ist.
Ein Insert-Zustand I_j emittiert ebenfalls mit einer bestimmten Wahrscheinlichkeit ein Symbol. Diese ist abhängig von der Häufigkeit der Symbole in den Spalten zwischen der Konsensusspalte j und der nächsten auf j folgenden Konsensusspalte k, da wir die Symbole in diesen Spalten als Insertionen in den entsprechenden Sequenzen des MSA A interpretieren. Zusätzlich zum Übergang von M_j nach M_k, $k > j$ sei dabei die nächste auf j folgende Konsensusspalte in A, erlauben wir den Übergang von M_j in den Insert-Zustand I_j und von I_j nach M_k, wodurch die Möglichkeit gegeben ist, zwischen je zwei aufeinanderfolgenden Konsensusspalten j und k des MSA ein Symbol in der Anfragesequenz einzufügen. Mehrere Insertionen zwischen den Spalten j und k sind durch den Übergang von I_j zu sich selbst möglich.
Zusätzlich zu den genannten Insert-Zuständen führen wir einen Insert-Zustand I_B ein, der das Einfügen eines oder mehrerer Symbole vor der ersten Konsensusspalte des MSA erlaubt. Für diesen Zustand sind wie für alle anderen Insert-Zustände der Übergang zu sich selbst und zu dem Match-Zustand möglich, der der ersten Konsensusspalte in A zugeordnet ist.
Ein Delete-Zustand D_j erlaubt das Löschen der Spalte j. Er ist im Gegensatz zu einem Match- bzw. Insert-Zustand ein stummer Zustand (1.6), d.h. er emittiert kein Symbol des Emissionsalphabets.
Sei $i < j$ die der Konsensusspalte j vorhergehende Konsensusspalte in A und k die nächste auf j folgende Konsensusspalte. Dann lassen wir für jeden Delete-Zustand D_j den Übergang von M_i nach D_j und von D_j nach M_k zu, so dass im MSA die Spalte j übersprungen werden kann ohne ein Symbol zu emittieren. Dies ist gleichbedeutend damit, an dieser Position in der Anfragesequenz eine Lücke einzufügen. Zusätzlich dazu sind Übergänge zwischen zwei jeweils aufeinanderfolgenden Delete-Zuständen D_j und D_k möglich, so dass mithilfe einer langen Folge von Delete-Zuständen sogar grössere Bereiche des MSA übersprungen werden können. Dies ist notwendig falls eine bestimmte Region, beispielsweise eine Proteindomäne, in der Anfragesequenz fehlt, die in der Sequenzfamilie vorhanden ist.
Wie in HMMER Plan7 [7], einem der am häufigsten verwendeten Programme für Profil-HMMs, schliessen wir Übergänge zwischen Insert- und Delete-Zuständen aus.

Definition 2.4 (Zustandsraum) *Sei das MSA A (2.1) gegeben. Sei I_B der Insert-Zustand der Insertionen vor der ersten Konsensusspalte des MSA erlaubt.*
Wir definieren den Zustandsraum

$$Z := \left\{ M_j, I_j, D_j \,\middle|\, |\vec{a}_j| \geq c \cdot K, j = 1, \ldots, N \right\} \cup \left\{ I_B \right\}, \tag{2.4}$$

als die Menge aller Match-, Insert- und Delete-Zustände, die einer Konsensusspalte im Alignment A zugeordnet sind, und des Zustands I_B.

Folgerung 2.5 *Die Menge Z' aller stummen Zustände in Z ist die Menge aller Delete-Zustände in Z, da diese, im Gegensatz zu den Match- und Insert-Zuständen, kein Symbol aus Σ emittieren:*

$$Z' = \left\{ D_j \,\middle|\, |a_j| \geq c \cdot K, j = 1, \ldots, N \right\}. \tag{2.5}$$

Zusätzlich zu den Match-, Insert- und Delete-Zuständen führen wir wie für ein HMM einen stummen Start- (B) und einen stummen End-Zustand (E) ein. Vom Start-Zustand B sind nur Übergänge zum Insert-Zustand I_B und zum Match- und Delete-Zustand der ersten Konsensusspalte des MSA zugelassen. Übergänge in den End-Zustand E sind nur von den drei der letzten Konsensusspalte des MSA zugeordneten Zuständen möglich. In Abbildung 2.1 ist als Beispiel der gerichtete Zustandsgraph eines MSA mit n Konsensusspalten gegeben.

Definition 2.6 (Spalte eines Zustands) *Sei $Z^+ := Z \cup \{B, E\}$. Dann definiert*

$$\begin{aligned} \mathrm{col} : Z^+ &\rightarrow \{0, \ldots, N+1\} \\ z &\mapsto \mathrm{col}(z). \end{aligned} \tag{2.6}$$

die Konsensusspalte in A, der der Zustand z zugeordnet ist.
Wir definieren $\mathrm{col}(B) := 0$, $\mathrm{col}(I_B) := 0$ *und* $\mathrm{col}(E) := N + 1$.

Definition 2.7 (Typ eines Zustandes) *Sei $Z^+ := Z \cup \{B, E\}$. Dann definiert*

$$\begin{aligned} \mathrm{typ} : Z^+ &\rightarrow \{B, M, I, D, E\} \\ z &\mapsto \mathrm{typ}(z). \end{aligned} \tag{2.7}$$

den Typ *eines Zustands.*

Definition 2.8 *Sei das Alignment A gegeben. Sei* $\mathrm{col}(z)$ *die Spalte in A, der der Zustand $z \in Z^+$ zugeordnet ist. Dann definieren wir folgende Ordnungsrelation $\prec_A$ auf der Menge aller Zustände Z^+:*
Für $z_k, z_l \in Z^+$ gelte

$$z_k \prec_A z_l \quad :\Longleftrightarrow \quad \mathrm{col}(z_k) < \mathrm{col}(z_l) \tag{2.8}$$

Lemma 2.9 *Die Relation $\prec_A$ definiert eine* strenge Halbordnung *auf Z^+.*

Beweis
Irreflexivität: $\mathrm{col}(z_k) < \mathrm{col}(z_k)$ ist für kein $z_k \in Z^+$ erfüllt.
Transitivität: Sei $z_k, z_i, z_l \in Z^+$ mit $z_k \prec_A z_i, z_i \prec_A z_l$
$\Rightarrow \mathrm{col}(z_k) < \mathrm{col}(z_i), \mathrm{col}(z_i) < \mathrm{col}(z_l) \quad \Rightarrow \quad \mathrm{col}(z_k) < \mathrm{col}(z_l) \quad \Rightarrow \quad z_k \prec_A z_l.$ □

Definition 2.10 (Profil-Hidden-Markov-Modell)
Sei $Z^+ := Z \cup \{B, E\}$. *Sei die Übergangsmatrix* $\mathcal{T} = (t_{z_k,z_l})_{z_k,z_l \in Z^+}$ *definiert wie in (1.4) und die Emissionsmatrix* $\mathcal{E} = (e_{z_k,\sigma_j})_{z_k \in Z^+, \sigma_j \in \Sigma^+}$ *wie in (1.5).*
Ein Profil-Hidden-Markov-Modell *(*Profil-HMM*) ist ein HMM mit Parametern* $M = (\Sigma^+, Z^+, \mathcal{T}, \mathcal{E})$, *mit folgenden zusätzlichen Bedingungen*

$$t_{z_k,E} = 0, \text{ falls } \exists z_i \in Z \text{ mit } (z_k \prec_A z_i) \wedge (\text{typ}(z_i) = \text{typ}(z_k)), \tag{2.9}$$

$$t_{z_k,z_l} = 0, \text{ falls } \begin{cases} & (z_l \prec_A z_k) \\ \textit{oder} & (z_k \prec_A z_l) \wedge (\text{typ}(z_l) = I) \\ \textit{oder} & (\text{typ}(z_k) = I) \wedge (\text{typ}(z_l) = D) \\ \textit{oder} & (\text{typ}(z_k) = D) \wedge (\text{typ}(z_l) = I) \\ \textit{oder} & (\text{col}(z_k) = \text{col}(z_l)) \wedge (\text{typ}(z_l) \in \{M, D\}) \\ \textit{oder} & \exists z_i \in Z : (z_k \prec_A z_i \prec_A z_l) \wedge (\text{typ}(z_i) = \text{typ}(z_l)), \end{cases} \tag{2.10}$$

für alle $z_k, z_l \in Z$.

Ein Profil-HMM erzeugt also wie ein HMM eine beobachtbare Sequenz $S_r := Y_1 Y_2 \ldots Y_r$ mit Elementen in Σ, und einen ihr zugrundeliegenden, verborgenen Pfad $X[1, r]$.

Bemerkung 2.11 Betrachten wir den Zustandsgraphen G eines Profil-HMMs (Abb. 2.1), dann sehen wir, dass ein Profil-HMM ein streng vorwärts gerichteter Prozess von links nach rechts durch G ist. Dadurch, dass Übergänge in einen stummen Zustand von Z, d.h. in einen Delete-Zustand, nur aus Zuständen heraus erlaubt sind, die der jeweils direkt vorhergehenden Konsensusspalte des Alignments zugeordnet sind, können wir die Zustände in Z entsprechend der Konsensusspalten sortieren, und erhalten so eine topologische Sortierung des Sub-Zustandsgraphen.

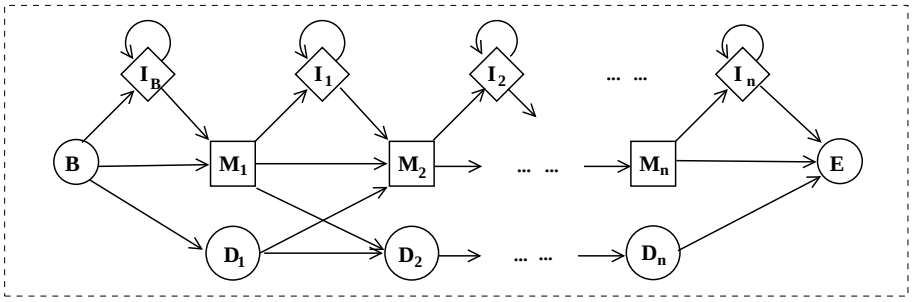


Abbildung 2.1: Der Zustandsgraph eines *Profil-HMMs* für ein multiples Sequenzalignment mit n Konsensusspalten. Die Architektur enspricht der von [11] eingeführten Architektur eines Profil-HMMs. Quadrate kennzeichnen Match-Zustände M, Rauten Insert-Zustände I und Kreise *stumme* Zustände, d.h. den Start-Zustand B, den End-Zustand E und Delete-Zustände D. Mögliche Übergänge zwischen den Zuständen sind durch Pfeile markiert.

2.2 Vorwärts- und Viterbi-Algorithmus

Da ein Profil-HMM ein HMM ist, das einige zusätzliche Bedingungen bezüglich der Emissions- und Übergangswahrscheinlichkeiten erfüllen muss, lassen sich alle in Kapitel 1 für HMMs eingeführten Bezeichnungen, Definitionen und Sätze auf ein Profil-HMM übertragen. Wir können also mithilfe des Vorwärts-Algorithmus (Algo. 1.21) die Wahrscheinlichkeit berechnen, eine Anfragesequenz $s[1,l]$ durch ein Profil-HMM zu erzeugen, und mit dem Viterbi-Algorithmus (Algo. 1.31) einen der ihr am wahrscheinlichsten zugrundeliegenden Pfade, d.h. einen *Viterbi-Pfad*, bestimmen. Dabei muss allerdings folgendes Problem berücksichtigt werden: Ist die Sequenz $s[1,l]$ sehr lang, was bei DNA- bzw. Aminosäure-Sequenzen sehr häufig der Fall ist, müssen im Vorwärts- und Viterbi-Algorithmus entsprechend viele Iterationen durchlaufen werden. Da die Emissions- und Übergangswahrscheinlichkeiten eines Profil-HMMs, abhängig vom gegebenen MSA A, sehr klein sein können, kann es durch die Vielzahl von Multiplikationen in der Rekursion passieren, dass die Vorwärts- bzw. Viterbi-Variablen so kleine Werte annehmen, dass sie mit dem im Programm zur Speicherung dieser Werte verwendeten Datentyp nicht mehr darstellbar sind und gleich 0 gesetzt werden.
Dieses Problem kann vermieden werden, indem der Vorwärts- und der Viterbi-Algorithmus jeweils für die log-transformierten Werte durchgeführt wird. Der Logarithmus eines Produkts zweier Werte ist gleich der Summe der Logarithmen dieser beiden Werte, wodurch in der entsprechenden Rekursion Additionen statt Multiplikationen auftreten, und somit das oben beschriebene Problem für den Viterbi-Algorithmus gelöst ist. Wie man anhand der Rekursion für die Vorwärts-Variablen in Lemma 2.13 sehen kann, ist dies jedoch für den Vorwärts-Algorithmus dadurch noch nicht gelöst. Dies wird direkt nach dem Beweis des Lemmas 2.13 genauer erklärt.

Definition 2.12 *Sei $s[1,l]$ eine Sequenz in Σ^l. Seien $\alpha_{z,i}$ bzw. $\delta_{z,i}$, $z \in Z$, $i = 0,\ldots,l$ die in (1.16) und (1.25) definierten Vorwärts- bzw. Viterbi-Variablen.*
Wir definieren für alle $z \in Z$ und für alle $i = 0,\ldots,l$

$$\begin{aligned} \tilde{\alpha}_{z,i} &:= \log(\alpha_{z,i}), \\ \tilde{\alpha}_{*,l} &:= \log(\alpha_{*,l}), \\ \tilde{\delta}_{z,i} &:= \log(\delta_{z,i}), \\ \tilde{\delta}_{*,l} &:= \log(\delta_{*,l}). \end{aligned}$$

$\tilde{\alpha}_{*,l}$ bzw. $\tilde{\delta}_{*,l}$ nennen wir den log-Vorwärts-Score *bzw. den* log-Viterbi-Score.

Lemma 2.13 *Es gilt*

$$\tilde{\alpha}_{z,0} = \begin{cases} -\infty & ,\forall z \in Z \backslash Z', \\ \log\Big(\sum\limits_{z' \in Z' \cup \{B\}} \exp\big(\tilde{\alpha}_{z',0} + \log(t_{z',z})\big)\Big) & ,\forall z' \in Z', \end{cases} \tag{2.11}$$

$$\tilde{\alpha}_{z,i} = \log(e_{z,s[i,i+d_z)} + \log\Big(\sum_{z' \in Z \cup \{B\}} \exp\big(\tilde{\alpha}_{z',i-d_z} + \log(t_{z',z})\big)\Big), \tag{2.12}$$

$$\tilde{\delta}_{z,0} = \begin{cases} -\infty & ,\forall z \in Z \backslash Z', \\ \max\limits_{z' \in Z' \cup \{B\}} \big(\tilde{\delta}_{z',0} + \log(t_{z',z})\big) & ,\forall z' \in Z', \end{cases} \tag{2.13}$$

$$\tilde{\delta}_{z,i} = \log(e_{z,s[i,i+d_z)} + \max_{z' \in Z \cup \{B\}} \big(\tilde{\delta}_{z',i-d_z}) + \log(t_{z',z})\big), \tag{2.14}$$

$i = 1, \ldots, l$.

Beweis

$$\begin{aligned}
\tilde{\alpha}_{z,0} &= \log(\alpha_{z,0}) \\
&= \begin{cases} 0 & ,\forall z \in Z \backslash Z', \\ \log\Big(\sum\limits_{z' \in Z' \cup \{B\}} \alpha_{z',0} t_{z',z}\Big) & ,\text{falls } z \in Z', \end{cases} \\
&= \begin{cases} -\infty & ,\forall z \in Z \backslash Z', \\ \log\Big(\sum\limits_{z' \in Z' \cup \{B\}} \exp\big(\log(\alpha_{z',0} t_{z',z})\big)\Big) & ,\forall z' \in Z', \end{cases} \\
&= \begin{cases} -\infty & ,\forall z \in Z \backslash Z', \\ \log\Big(\sum\limits_{z' \in Z' \cup \{B\}} \exp\big(\tilde{\alpha}_{z',0} + \log(t_{z',z})\big)\Big) & ,\forall z' \in Z', \end{cases}
\end{aligned}$$

$$\begin{aligned}
\tilde{\delta}_{z,0} &= \log(\delta_{z,0}) \\
&= \begin{cases} 0 & ,\forall z \in Z \backslash Z', \\ \log\Big(\max\limits_{z' \in Z' \cup \{B\}} \delta_{z',0} t_{z',z}\Big) & ,\text{falls } z \in Z', \end{cases} \\
&= \begin{cases} -\infty & ,\forall z \in Z \backslash Z', \\ \max\limits_{z' \in Z' \cup \{B\}} \log(\delta_{z',0} t_{z',z}) & ,\text{falls } z \in Z', \end{cases} \\
&= \begin{cases} -\infty & ,\forall z \in Z \backslash Z', \\ \max\limits_{z' \in Z' \cup \{B\}} \big(\tilde{\delta}_{z',0} + \log(t_{z',z})\big) & ,\forall z' \in Z', \end{cases}
\end{aligned}$$

Für $\tilde{\alpha}_{z,i}$ und $\tilde{\delta}_{z,i}$, $i = 1, \ldots, l$ wird dies analog bewiesen. □

Durch das Logarithmieren von Wahrscheinlichkeiten erhalten wir negative Werte für die Variablen $\tilde{\alpha}_{z,i}$ und $\tilde{\delta}_{z,i}$. Betrachten wir nun die Vorwärts-Rekursion (2.12), so sieht man, dass durch die Terme

$$\exp\big(\tilde{\alpha}_{z',i-d_z} + \log(t_{z',z})\big) \quad \text{bzw.} \quad \log\Big(\sum_{z' \in Z \cup \{B\}} \exp\big(\tilde{\alpha}_{z',i-d_z} + \log(t_{z',z})\big)\Big)$$

ebenfalls Werte erhalten werden können, die so klein sind, dass sie mit dem im Programm zur Speicherung dieser Werte verwendeten Datentyp nicht mehr darstellbar sind. Zur Lösung dieses Problems betrachten wir folgende Umformung:
Seien $\tilde{p} = \log p$ und $\tilde{q} = \log q$ die Logarithmen zweier Wahrscheinlichkeiten p und q. Dann gilt für $\tilde{r} = \log(p+q)$

$$\begin{aligned}\tilde{r} &= \log\big(\exp(\tilde{p}) + \exp(\tilde{q})\big) \\ &= \log\Big(\exp(\tilde{p})\big(1 + \frac{\exp(\tilde{q})}{\exp(\tilde{p})}\big)\Big) \\ &= \tilde{p} + \log\big(1 + \exp(\tilde{q} - \tilde{p})\big)\end{aligned}$$

Beachten wir dabei, dass $\tilde{p} \geq \tilde{q}$, so ist sichergestellt, dass $\tilde{r} \leq 1$. Wenden wir nun diese Umformung in der Vorwärts-Rekursion an (Algo. 2.14), dann ist das oben beschriebene Problem auch für den Vorwärts-Algorithmus gelöst.

Algorithmus 2.14 (log-Vorwärts-Algorithmus)
Sei Z' die Menge aller stummen Zustände in Z und T' die Menge aller Übergänge, die in einem stummen Zustand enden.
Da der Sub-Zustandsgraph $G' = (Z, T')$ azyklisch ist, können wir den Zustandsraum $Z := \{z_1, \ldots, z_m\}$ entsprechend des topologisch sortierten Sub-Zustandsgraphen sortieren (siehe Kapitel 1, 1.33):
Für je zwei Zustände $z_i, z_j \in Z$ gelte

$$i < j, \quad \text{falls } z_i \prec_A z_j, \tag{2.15}$$

Sei $z_0 := B$. Dann lässt sich der log-Vorwärts-Score mithilfe des folgenden *log-Vorwärts-Algorithmus* berechnen:
Erzeuge das Feld $\tilde{\alpha}_{z,i}, z \in Z \cup \{B\}, i = 0, \ldots, l$.

1. Initialisierung:

$\tilde{\alpha}_{B,0} = 0,$
$\tilde{\alpha}_{z,0} = -\infty, \quad \forall z \in Z \backslash Z',$
$\tilde{\alpha}_{z,i} = -\infty, \quad \forall z \in Z,\ i = 1, \ldots, l,$
$\tilde{\alpha}_{*,l} = -\infty,$
für $j = 1, \ldots, m,\ z_j \in Z',$
  für $k = 0, \ldots, j-1,\ z_k \in Z' \cup \{B\},$ berechne
    falls $\big(\tilde{\alpha}_{z_j,0} \geq (\tilde{\alpha}_{z_k,0} + \log(t_{z_k,z_j}))\big)$
      $\tilde{\alpha}_{z_j,0} = \tilde{\alpha}_{z_j,0} + \log\Big(1 + \exp\big((\tilde{\alpha}_{z_k,0} + \log(t_{z_k,z_j})) - \tilde{\alpha}_{z_j,0}\big)\Big)$
    sonst

$$\tilde{\alpha}_{z_j,0} = \left(\tilde{\alpha}_{z_k,0} + \log(t_{z_k,z_j})\right) + \log\left(1 + \exp\left(\tilde{\alpha}_{z_j,0} - (\tilde{\alpha}_{z_k,0} + \log(t_{z_k,z_j}))\right)\right)$$

2. Rekursion:

für $i = 1, \ldots, l$,
  für $j = 1, \ldots, m$
    für $k = 0, \ldots, j-1$, $z_k \in Z \cup \{B\}$, berechne
      falls $\left(\tilde{\alpha}_{z_j,i} \geq \tilde{\alpha}_{z_k,i-d_{z_j}} + \log(t_{z_k,z_j})\right)$
        $\tilde{\alpha}_{z_j,i} = \tilde{\alpha}_{z_j,i} + \log\left(1 + \exp(\tilde{\alpha}_{z_k,i-d_{z_j}} + \log t_{z_k,z_j} - \tilde{\alpha}_{z_j,i})\right)$
      sonst
        $\tilde{\alpha}_{z_j,i} = \tilde{\alpha}_{z_k,i-d_{z_j}} + \log t_{z_k,z_j} + \log\left(1 + \exp(\tilde{\alpha}_{z_j,i} - (\tilde{\alpha}_{z_k,i-d_{z_j}} + \log t_{z_k,z_j}))\right)$
    $\tilde{\alpha}_{z_j,i} = \tilde{\alpha}_{z_j,i} + \log(e_{z_j,s[i,i+d_{z_j}]})$

3. Termination:

für $k = 1, \ldots, m$, $z_k \in Z$, berechne
  falls $\left(\tilde{\alpha}_{*,l} \geq \tilde{\alpha}_{z_k,l} + \log(t_{z_k,E})\right)$
    $\tilde{\alpha}_{*,l} = \tilde{\alpha}_{*,l} + \log\left(1 + \exp\left(\tilde{\alpha}_{z,l} + \log(t_{z,E}) - \tilde{\alpha}_{*,l}\right)\right)$
  sonst
    $\tilde{\alpha}_{*,l} = \tilde{\alpha}_{z,l} + \log(t_{z,E}) + \log\left(1 + \exp\left(\tilde{\alpha}_{*,l} - (\tilde{\alpha}_{z,l} + \log(t_{z,E}))\right)\right)$

Folgerung 2.15 *Die Wahrscheinlichkeit, eine Sequenz $s[1,l] \in \Sigma^l$ durch das Profil-HMM zu erzeugen, ist gegeben durch*

$$\alpha_{*,l} = \exp(\tilde{\alpha}_{*,l}). \tag{2.16}$$

Algorithmus 2.16 (log-Viterbi-Algorithmus)
Sei Z' die Menge aller stummen Zustände in Z und T' die Menge aller Übergänge, die in einem stummen Zustand enden. Sei der Zustandsraum $Z := \{z_1, \ldots, z_m\}$ entsprechend der Bedingung 2.15 sortiert. Sei $z_0 := B$. Dann lässt sich der log-Viterbi-Score mithilfe des folgenden *log-Viterbi-Algorithmus* berechnen:
Erzeuge das Feld $\delta_{z,i}, z \in Z \cup \{B\}, i = 0, \ldots, l$.
Vorwärtsteil:

1. Initialisierung:

$$\tilde{\delta}_{B,0} = 0,$$
$$\tilde{\delta}_{z,0} = -\infty, \quad \forall z \in Z \backslash Z',$$

für $j = 1, \ldots, m, z_j \in Z'$ berechne

$$\tilde{\delta}_{z_j,0} = \max_{z_k \in Z' \cup \{B\}, k<j} \left(\tilde{\delta}_{z_k,0} + \log(t_{z_k,z_j})\right),$$

2. Rekursion:

für $i = 1, \ldots, l$,

für $j = 1, \ldots, m$ berechne

$$\tilde{\delta}_{z_j,i} = \log(e_{z_j,s[i,i+d_{z_j})}) + \max_{z_k \in Z \cup \{B\}, k<j} \left((\tilde{\delta}_{z_k,i-d_{z_j}}) + \log(t_{z_k,z_j})\right)$$

3. Termination:

berechne

$$\tilde{\delta}_{*,l} = \max_{z \in Z} \left(\tilde{\delta}_{z,l} + \log(t_{z,E})\right)$$

Rückwärtsteil: (backtracking)

1. Beginn:

berechne

$$v_r = \arg\max_{z \in Z} \left(\tilde{\delta}_{z,l} + \log(t_{z,E})\right)$$

2. Rekursion:

$$\text{für } i = r-1, \ldots, 1, D_{i+1,r} = \sum_{j=i+1}^{r} d_{v_j}, \text{ berechne}$$

$$v_i = \arg\max_{z \in Z} \left(\tilde{\delta}_{z,l-D_{i+1,r}} + \log(t_{z,v_{i+1}})\right)$$

Der Viterbi-Pfad $v[1,r]$ ist der wahrscheinlichste der Sequenz $s[1,l]$ zugrundeliegende Pfad. Er definiert also das beste Alignment, d.h. das Alignment mit dem höchsten Score, der Anfragesequenz $s[1,l]$ zum multiplen Sequenzalignment A der Familie $\mathcal{S}$. Dieses Alignment erhalten wir indem wir den Pfad von Position 1 bis Position r durchlaufen und für jeden Match-Zustand ein bestimmtes Symbol der Anfragesequenz (siehe Algo. 2.17) mit der Spalte des Alignments A, der der Match-Zustand zugeordnet ist, alignieren bzw. für jeden Insert- und Delete-Zustand eine Lücke an der entsprechenden Position im Alignment bzw. in der Sequenz einfügen:

Algorithmus 2.17 (Alignment-Algorithmus)
Sei $s[1,l]$ die Anfragesequenz und $v[1,r]$ der Viterbi-Pfad dieser Sequenz. Dann erhalten wir das Alignment der Sequenz s zum MSA A mithilfe des folgenden *Alignment-Algorithmus*:

Sei $i = 1, \ldots, l$ die Position in der Anfragesequenz s und $j = 1 \ldots, r$ die Position im Viterbi-Pfad v.

Initialisierung: $i = 1, j = 1$
while $(i < l)$ do {
  if $(\text{typ}(v_j) == M)$ {
    aligniere das Symbol an Pos. i in s mit der Spalte $\text{col}(v_j)$ in A;
    $i = i + 1$;
  }
  else if $(\text{typ}(v_j) == I)$ {
    if $(\text{col}(v_j) + 1$ ist eine Konsensusspalte in $A)$
      füge eine Spalte mit Lücken in A ein;
      aligniere das Symbol an Pos. i in s mit dieser Spalte;
    else if $(\text{col}(v_j) + 1$ ist keine Konsensusspalte in $A)$
      aligniere das Symbol an Pos. i in s mit dieser Spalte;
  $i = i + 1$;
  }
  else if $(\text{typ}(v_j) == D)$ {
    füge an Position i in s eine Lücke ein;
  }
  $j = j + 1$;
}

2.3 Schätzung der Parameter eines Profil-HMMs

Nehmen wir an, dass die Emissions- und Übergangswahrscheinlichkeiten eines Profil-HMMs grösser sind als 0, dann kann das Profil-HMM jede beliebige Sequenz mit Symbolen aus dem gegebenen Emissionsalphabet Σ erzeugen. Es definiert also eine Wahrscheinlichkeitsverteilung auf der Menge aller Sequenzen mit Buchstaben in Σ. Wir wollen nun die Parameter des Profil-HMMs, d.h. die Emissions- und Übergangswahrscheinlichkeiten, so schätzen, dass diese Verteilung für Mitglieder der Sequenzfamilie $\mathcal{S}$ besonders hohe Werte annimmt. Dazu nutzen wir die im MSA gegebene Information über die Verteilung der Nukleotid- bzw. Aminosäure-Häufigkeiten in den Spalten des MSA.

2.3.1 Emissionswahrscheinlichkeiten

Jede Spalte eines MSA kann durch einen Vektor von Nukleotid- bzw. Aminosäure-Häufigkeiten repräsentiert werden:

Definition 2.18 *Sei die Spalte j des Alignment A gegeben.*
Dann repräsentiert der Vektor

$$\begin{aligned} \vec{n}_j &:= (n_{j,1}, \ldots, n_{j,n}) \\ mit\ n_{j,i} &:= \#\{a_{k,j} \,|\, a_{k,j} = \sigma_i, k = 1, \ldots, K\}, \quad i = 1, \ldots, n \end{aligned} \tag{2.17}$$

die Häufigkeiten der in der Spalte j beobachteten Symbole.
Wir definieren $|\vec{n}_j| := \sum_{i=1}^{n} n_{j,i}$ als die Anzahl aller Symbole aus Σ, d.h. aller Nicht-Gaps, in der Spalte j von A.

Um nun die Emissionswahrscheinlichkeiten der Zustände des Profil-HMMs aus dem gegebenen MSA A zu schätzen, betrachten wir für jeden Match-Zustand die Häufigkeiten der Symbole aus Σ in der Konsensusspalte der dieser Zustand zugeordnet ist, und für jeden Insert-Zustand die Häufigkeiten in den Spalten zwischen der entsprechenden Konsensusspalte und der nächsten, da wir die Symbole in diesen Spalten als Insertionen interpretieren.

Definition 2.19 *Seien zwei Vektoren $\vec{n}_i$ und $\vec{n}_j$ in $\mathbb{R}^n$ gegeben. Dann definieren wir*

$$\vec{n}_i + \vec{n}_j := (n_{i,1} + n_{j,1}, \ldots, n_{i,n} + n_{j,n}). \tag{2.18}$$

Definition 2.20 *Sei $z \in Z$ ein Match- oder Insert-Zustand und $j := \mathrm{col}(z)$ die Spalte im Alignment A, der dieser Zustand zugeordnet ist. Sei k die nächste auf j folgende Konsensusspalte in A.*
Dann repräsentiert der Vektor

$$\vec{n}_z := \begin{cases} \vec{n}_j & , falls\ \mathrm{typ}(z) = M, \\ \sum_{i=j+1}^{k-1} \vec{n}_i & , falls\ \mathrm{typ}(z) = I, \end{cases} \tag{2.19}$$

die dem Zustand z zugeordneten Häufigkeiten der Symbole aus Σ in A, d.h. für einen Match-Zustand die Häufigkeiten der in der Spalte j vorkommenden Symbole, und für einen Insert-Zustand die Häufigkeiten der zwischen den Konsensusspalten j und k vorkommenden Symbole. D.h. die dem Insert-Zustand I_B zugeordneten Häufigkeiten $\vec{n}_{I_B}$ sind die Häufigkeiten der in den Spalten vor der ersten Konsensusspalte im Alignment beobachteten Symbole.

Nehmen wir an, dass die in einer Konsensusspalte (für einen Match-Zustand) bzw. in den Spalten zwischen zwei Konsensusspalten (für einen Insert-Zustand) beobachteten Symbole σ_i jeweils entsprechend einer dieser Konsensusspalte bzw. einer diesen

Spalten zugrundeliegenden Wahrscheinlichkeitsverteilung $\vec{p} := (p_1, \ldots, p_n)$ erzeugt werden, p_i sei dabei die Wahrscheinlichkeit das Symbol σ_i zu erzeugen, $p_i \geq 0$, $\sum_{i=1}^{n} p_i = 1$, dann sind die für einen Zustand z beobachteten Häufigkeiten $\vec{n}_z$ jeweils entsprechend der Multinomialverteilung mit Parametern $\vec{p}$ verteilt. Diese Parameter $\vec{p}$ entsprechen den zu schätzenden Emissionswahrscheinlichkeiten.

Relative Häufigkeiten

Eine Methode die Parameter $\vec{p}$ für einen Zustand z aus den beobachteten Häufigkeiten $\vec{n}_z$ zu schätzen, sind die relativen Häufigkeiten der beobachteten Symbole, d.h.

$$\hat{p}_i = \frac{n_{z,i}}{|\vec{n}_z|}, \qquad i = 1, \ldots, n. \tag{2.20}$$

Ein Nachteil dieser Methode ist jedoch, dass die so für einen Zustand z geschätzte Wahrscheinlichkeit $\hat{p}_i$ eines Symbols σ_i den Wert 0 annimmt, falls dieses Symbol für den Zustand z nicht beobachtet wird. Ein Grund für das Fehlen eines Symbols kann z.B. die Konserviertheit der entsprechenden Konsensusspalte (für einen Match-Zustand) sein, oder aber auch eine zu kleine Datenmenge, d.h. eine zu geringe Anzahl an Sequenzen im MSA A. Besonders im zuletzt genannten Fall wollen wir jedoch nicht ausschliessen, dass andere Mitglieder der Sequenzfamilie an dieser Position dieses Symbol enthalten.

Pseudo-Counts

Addieren wir für jedes Symbol $\sigma_i \in \Sigma$ zu der Häufigkeit $n_{z,i}$ mit der es in der Konsensusspalte $\mathrm{col}(z)$ (Match) bzw. zwischen dieser und der nächsten Konsensusspalte (Insert) beobachtet wird, jeweils eine Konstante $\alpha_i > 0$, einen sog. *Pseudo-Count*, und normieren diesen Wert, d.h.

$$\hat{p}_i := \frac{n_{z,i} + \alpha_i}{|\vec{n}_z| + |\vec{\alpha}|}, \qquad \text{mit } |\vec{\alpha}| := \sum_{i=1}^{n} \alpha_i, \qquad i = 1, \ldots, n, \tag{2.21}$$

dann ist in jedem Match- und Insert-Zustand die geschätzte Emissionswahrscheinlichkeit $\hat{p}_i$ für alle σ_i grösser 0. Wie man anhand dieser Gleichung jedoch sehen kann, wird in der Schätzung der Emissionswahrscheinlichkeit des Symbols σ_i für einen Zustand z nicht berücksichtigt welche anderen Aminosäuren für z im MSA beobachtet werden. Es gibt jedoch z.B. sog. *neutrale Substitutionen* zwischen Aminosäuren, d.h. eine Aminosäure wird durch eine andere ersetzt, ohne dass die Funktion des Proteins verändert wird. Solch ein Austausch ist jedoch nicht zwischen allen Aminosäuren möglich und tritt in bestimmten Bereichen eines Proteins häufiger auf als in anderen.
Es ist somit sinnvoll, dieses sog. *a-priori-Wissen* über bestimmte Zusammenhänge

bzw. Eigenschaften der Aminosäuren bei der Schätzung der Emissionswahrscheinlichkeiten zu berücksichtigen. Eine Methode mit der dies möglich ist, und die auch in anderen Programmen für Profil-HMMs, wie z.B HMMER [7] und SAM [9], zur Schätzung der Emissionswahrscheinlichkeiten genutzt wird, ist eine sog. *Mischung von Dirichlet-Verteilungen* [4, 22].

Mischung von Dirichlet-Verteilungen

Sei $D_n := \{(p_1, \ldots, p_n) \,|\, p_1 + \ldots + p_n = 1\}$ die Menge aller möglichen Verteilungen der Symbole σ_i aus Σ. Wir nehmen an, wir haben eine Wahrscheinlichkeitsverteilung über der Menge D_n gegeben, die das a-priori-Wissen über die Verteilung der Symbole in den Konsensusspalten bzw. in den Spalten zwischen je zwei Konsensusspalten des MSA wiederspiegelt. Diese Verteilung nennen wir die *a-priori-Verteilung*.
Statt wie bisher die Parameter $\vec{p}$ der Multinomialverteilung für jeden beobachteten Häufigkeitsvektor $\vec{n}_z, z \in Z$, separat zu schätzen, wollen wir nun aus allen für einen bestimmten Zustandstyp, d.h Match oder Insert, beobachteten Häufigkeitsvektoren direkt die zugrundeliegende Wahrscheinlichkeitsverteilung bzw. ihre Dichte schätzen. Als a-priori-Verteilung über der Menge D_n wählen wir die Dirichlet-Verteilung:

Definition 2.21 (Dirichlet-Verteilung)
Eine Dirichlet-Verteilung ist eine Wahrscheinlichkeitsverteilung über D_n und ist definiert durch die Dichte

$$\rho(\vec{p}) = \frac{\prod_{i=1}^{n} p_i^{\alpha_i - 1}}{\int_{\vec{p}} \prod_{i=1}^{n} p_i^{\alpha_i - 1}}, \quad \vec{p} \in D_n, \tag{2.22}$$

mit Parametern $\vec{\alpha} := (\alpha_1, \ldots, \alpha_n)$, $\alpha_i > 0$.

Um also die Dichte ρ der Dirichlet-Verteilung zu schätzen müssen ihre Parameter $\vec{\alpha}$ geschätzt werden.
Nehmen wir an, dass die Parameter einer Dirichlet-Verteilung gegeben sind, dann lassen sich die für einen bestimmten Zustand beobachteten Häufigkeiten $\vec{n}_z$ der Symbole und die Verteilung $\vec{p}$ durch folgendes Zufallsexperiment erzeugen:

1. wähle $\vec{p} \in D_n$ zufällig, entsprechend der a-priori-Verteilung gegeben durch die Dichte ρ
2. erzeuge die beobachteten Häufigkeiten $\vec{n}_z$ der Symbole zufällig, entsprechend der Verteilung $\vec{p}$; somit ist $\vec{n}_z$ entsprechend der Multinomialverteilung mit Parametern $\vec{p}$ verteilt;

Die geschätzte Wahrscheinlichkeit $\hat{p}_i$ des Symbols σ_i, gegeben eine Dirichlet-Dichte mit Parametern $\vec{\alpha}$ und einen Häufigkeitsvektor $\vec{n}_z$, ist nun definiert durch

$$\hat{p}_i := P_{\vec{\alpha}}(\sigma_i \,|\, \vec{n}_z) \tag{2.23}$$

Jedoch ist es auch mit einer einzigen Dirichlet-Verteilung nicht möglich, komplexe Zusammenhänge zwischen Aminosäuren zu berücksichtigen, da diese durch die Parameter $\vec{\alpha}$ nicht ausreichend wiedergespiegelt werden können. Kombinieren wir allerdings mehrere Dirichlet-Verteilungen, jeweils definiert durch die Dichte $\rho_j, j = 1, \ldots, l$, miteinander, so können wir mithilfe der verschiedenen Parameter $\vec{\alpha}_j$ eine Vielzahl der dem Alignment zugrundeliegenden prototypischen Aminosäureverteilungen berücksichtigen. Damit kommt eine Mischung von Dirichlet-Verteilungen der (gedachten) natürlichen Verteilung von Aminosäuren in den Spalten eines MSA sehr viel näher als eine einzige Dirichlet-Verteilung.

Definition 2.22 (Dichte einer Mischung von Dirichlet-Verteilungen)
Seien die Dichten $\rho_j, j = 1, \ldots, l$, mit Parametern $\vec{\alpha}_j := (\alpha_{j,1}, \ldots, \alpha_{j,n})$ von l Dirichlet-Verteilungen gegeben. Sei $q_j \geq 0, j = 1, \ldots, l$, und $q_1 + \ldots + q_l = 1$.
Dann ist die Mischung *dieser l Dirichlet-Verteilungen definiert durch die Dichte*

$$\rho = q_1\rho_1 + \ldots + q_l\rho_l. \tag{2.24}$$

Die q_j nennen wir die Koeffizienten *und die Dichten ρ_j die* Komponenten *der Dichte der der Dirichlet-Mischung.*
Die Menge aller Parameter Θ einer Mischung von Dirichlet-Dichten ist gegeben durch $\Theta = (\vec{\alpha}_1, \ldots \vec{\alpha}_l, q_1, \ldots, q_l)$. Wir schreiben deshalb auch ρ_Θ statt ρ.

Nehmen wir an, wir haben die Parameter Θ der Dichte ρ_Θ und einen Häufigkeitsvektor $\vec{n}_z$ gegeben, dann ist die geschätzte Wahrscheinlichkeit $\hat{p}_i$ des Symbols σ_i gegeben durch

$$\begin{aligned}
\hat{p}_i &:= P_\Theta(\sigma_i \,|\, \vec{n}_z) \\
&= \int_{\vec{p}} P(\sigma_i \,|\, \vec{p}) \cdot \rho_\Theta(\vec{p} \,|\, \vec{n}_z) d\vec{p} && (2.25) \\
&= \int_{\vec{p}} p_i \cdot \rho_\Theta(\vec{p} \,|\, \vec{n}_z) d\vec{p} && (2.26) \\
&= E(p_i \,|\, \vec{n}_z). && (2.27)
\end{aligned}$$

Dabei ist

$$\rho_\Theta(\vec{p} \,|\, \vec{n}_z) = \frac{\rho(\vec{p}) \cdot P(\vec{n}_z \,|\, \vec{p})}{P(\vec{n}_z)} \tag{2.28}$$

die bedingte Dichte der Verteilung $\vec{p}$, gegeben den Häufigkeitsvektor $\vec{n}_z$.
$P(\vec{n}_z \,|\, \vec{p})$ ist dabei die Verteilung von $\vec{n}_z$ entsprechend der Multinomialverteilung mit Parametern $\vec{p}$ und

$$P(\vec{n}_z) = \int_{\vec{p}} P(\vec{n}_z \,|\, \vec{p}) \cdot \rho(\vec{p}) d\vec{p} \tag{2.29}$$

Lemma 2.23 *Seien die Parameter* $\Theta = (\vec{\alpha}_1, \dots \vec{\alpha}_l, q_1, \dots, q_l)$ *und der Häufigkeitsvektor* $\vec{n}_z$ *eines Zustandes* z *gegeben.*
Sei $B(\vec{x}) := \prod_{i=1}^{n} \frac{\Gamma(x_i)}{\Gamma(|\vec{x}|)}$ *die Beta-Funktion. Dann gilt*

$$\hat{p}_i = \begin{cases} \dfrac{n_{z,i} + \alpha_i}{|\vec{n}_z| + |\vec{\alpha}|} & , \textit{falls } l = 1, \\ \dfrac{\sum_{j=1}^{l} q_j \frac{B(\vec{n}_z + \vec{\alpha}_j)}{B(\vec{\alpha}_j)} \frac{n_{z,i} + \alpha_{j,i}}{|\vec{n}_z| + |\vec{\alpha}_j|}}{\sum_{k=1}^{l} q_k \frac{B(\vec{n}_z + \vec{\alpha}_k)}{B(\vec{\alpha}_k)}} & , \textit{sonst.} \end{cases} \tag{2.30}$$

Beweis siehe [22]. □

Für $l = 1$, d.h. für eine einzelne Dirichlet-Verteilung, entspricht dies also der Addition von Pseudo-Counts (2.21) zu den beobachteten Häufigkeiten.

Sei die Menge $\{\vec{n}_1, \dots, \vec{n}_t\}$ aller einem bestimmten Zustandstyp zugeordneten Häufigkeitsvektoren gegeben. Dann schätzen wir die Parameter Θ der Dichte ρ_Θ mithilfe der *Maximum-Likelihood*-Methode (ML). D.h. wir suchen die Parameter Θ, die die gemeinsame Wahrscheinlichkeit des Auftretens der beobachteten Häufigkeitsvektoren $\vec{n}_j, j = 1, \dots, t$, maximieren.
Da wir annehmen, dass die Vektoren $\vec{n}_j$ unabhängig voneinander erzeugt werden, gilt

$$P_\Theta\big(\vec{n}_1, \dots, \vec{n}_m \,\big|\, |\vec{n}_1|, \dots, |\vec{n}_m|\big) = \prod_{j=1}^{m} P_\Theta\big(\vec{n}_j \,\big|\, |\vec{n}_j|\big). \tag{2.31}$$

Wir suchen also die Parameter Θ^* für die gilt

$$\Theta^* = \arg\max_\Theta \prod_{j=1}^{m} P_\Theta\big(\vec{n}_j \,\big|\, |\vec{n}_j|\big). \tag{2.32}$$

Da gilt

$$\Theta^* = \arg\max_\Theta \prod_{j=1}^{m} P_\Theta\big(\vec{n}_j \,\big|\, |\vec{n}_j|\big) \iff \Theta^* = \arg\min_\Theta \Big(-\sum_{j=1}^{m} \log P_\Theta\big(\vec{n}_j \,\big|\, |\vec{n}_j|\big)\Big) \tag{2.33}$$

können wir die Parameter Θ^* mithilfe des Gradientenabstiegsverfahrens in zwei Schritten schätzen; im ersten Schritt werden die $\vec{\alpha}_j$ für feste q_j geschätzt, im zweiten Schritt die q_j für feste $\vec{\alpha}_j$. Dieses Verfahren bzw. der Algorithmus dazu wird ausführlich in [22] beschrieben.

Die Wahl der Anzahl der Komponenten q_j der Dirichlet-Mischung hat einen entscheidenden Einfluss darauf, welche und vor allem wieviele Zusammenhänge zwischen den Elementen aus Σ berücksichtigt werden. Dabei muss allerdings beachtet

werden, dass einerseits möglichst viele der bekannten Zusammenhänge berücksichtigt werden, andererseits aber auch die Anzahl der zu schätzenden Parameter Θ nicht zu gross ist. [22] wählen deshalb eine Dirichlet-Mischung mit neun Komponenten, deren Parameter auf der BLOCKS-Datenbank [8] geschätzt werden. Diese Dirichlet-Mischung wird ebenfalls in den Programmen HMMER und SAM zur Schätzung der Emissionswahrscheinlichkeiten für Proteinsequenzen verwendet. Die Emissionswahrscheinlichkeiten eines Profil-HMMs für ein MSA von DNA-Sequenzen können mit einer einfachen Dirichlet-Verteilung, d.h. mit der Pseudo-Count-Methode, geschätzt werden, da die für Aminosäuren genannten Zusammenhänge nicht für Nukleotide berücksichtigt werden müssen.

2.3.2 Übergangswahrscheinlichkeiten

Auch die Übergangswahrscheinlichkeiten eines Profil-HMMs können anhand des MSA geschätzt werden: Zu jeder Sequenz des MSA A gibt es einen eindeutigen Pfad durch den Zustandsgraphen des Profil-HMMs. Fassen wir die in diesen Pfaden beobachteten Übergänge zusammen, so erhalten wir für jeden Zustand $z \in Z$ einen Vektor $\vec{n}_z$, der die Häufigkeiten der im MSA beobachteten Übergänge von z in die nachfolgenden Zustände enthält:

Definition 2.24 *Sei $z \in Z$ ein Zustand und $j := \mathrm{col}(z)$ die Konsensusspalte im Alignment A, der dieser Zustand zugeordnet ist. Seien M_j, I_j und D_j die drei der Spalte j zugeordneten Zustände. Seien M_k, I_k und D_k die drei der nächsten auf j folgenden Konsensusspalte k zugeordneten Zustände.*
Dann sei

$$\vec{n}_z := \begin{cases} (n_{z,M_k}, n_{z,I_j}, n_{z,D_k}) & ,\textit{falls } \mathrm{typ}(z) = M, \\ (n_{z,M_k}, n_{z,I_j}) & ,\textit{falls } \mathrm{typ}(z) = I, \\ (n_{z,M_k}, n_{z,D_k}) & ,\textit{falls } \mathrm{typ}(z) = D, \end{cases} \tag{2.34}$$

der Vektor der Häufigkeiten, der vom Zustand z aus möglichen Übergänge, und

$$\vec{t}_z := \begin{cases} (t_{z,M_k}, t_{z,I_j}, t_{z,D_k}) & ,\textit{falls } \mathrm{typ}(z) = M, \\ (t_{z,M_k}, t_{z,I_j}) & ,\textit{falls } \mathrm{typ}(z) = I, \\ (t_{z,M_k}, t_{z,D_k}) & ,\textit{falls } \mathrm{typ}(z) = D. \end{cases} \tag{2.35}$$

der Vektor der entsprechenden Übergangswahrscheinlichkeiten.

Wollen wir nun die Übergangswahrscheinlichkeiten aus den beobachteten Übergängen heraus schätzen, müssen wir ein ähnliches Problem zu dem in Abschnitt 2.3.1 beschriebenen beachten. Sind in zwei direkt aufeinanderfolgenden Konsensusspalten j und k, d.h. $k = j + 1$, des MSA keine Lücken zu finden, beobachten wir für jede Sequenz des MSA einen Übergang von M_j zu M_k. Die Häufigkeit für alle anderen

Übergänge an dieser Position ist somit 0. Da wir jedoch an jeder Position Insertionen und Deletionen zulassen wollen, können wir die Übergangswahrscheinlichkeiten nicht entsprechend der beobachteten relativen Häufigkeiten der Übergänge wählen. Wie die Emissionswahrscheinlichkeiten eines Profil-HMMs für DNA-Sequenzen, schätzen wir die Übergangswahrscheinlichkeiten mit einer einfachen Dirichlet-Verteilung. Dabei betrachten wir die verschieden Arten von Übergängen, d.h Übergänge aus Match-, aus Insert- und aus Delete-Zuständen heraus, unabhängig voneinander und schätzen für jede der drei Arten von Übergängen eine eigene Dirichlet-Dichte. Die Parameter $\vec{\alpha}_M$,$\vec{\alpha}_I$ und $\vec{\alpha}_D$, dieser drei Dichten sind gegeben durch

$$\vec{\alpha}_M := (\alpha_{M,M}, \alpha_{M,I}\alpha_{M,D}) \quad \text{für Übergänge aus einem Match-Zustand,} \tag{2.36}$$

$$\vec{\alpha}_I := (\alpha_{I,M}, \alpha_{I,I}) \quad \text{für Übergänge aus einem Insert-Zustand,} \tag{2.37}$$

$$\vec{\alpha}_D := (\alpha_{D,M}, \alpha_{D,D}) \quad \text{für Übergänge aus einem Delete-Zustand.} \tag{2.38}$$

Lemma 2.25 *Seien zwei Zustände z_i und $z_j \in Z$ gegeben. Dann gilt für die geschätzte Übergangswahrscheinlichkeit $\hat{t}_{z_i,z_j}$ von z_i nach z_j*

$$\hat{t}_{z_i,z_j} = \frac{n_{z_i,z_j} + \alpha_{\text{typ}(z_i),\text{typ}(z_j)}}{|\vec{n}_{z_i}| + |\vec{\alpha}_{\text{typ}(z_i)}|}. \tag{2.39}$$

Beweis nach Lemma 2.23. □

Die Parameter $\vec{\alpha}_M, \vec{\alpha}_I$ bzw. $\vec{\alpha}_D$ lassen sich wie die Parameter der Dirichlet-Dichte für die Emissionswahrscheinlichkeiten mithilfe der Maximum-Likelihood (ML) -Methode schätzen.
Wistrand & Sonnhammer [29] zeigen jedoch, dass die mit der ML-Methode für Alignments der Pfam-Datenbank [2] geschätzten Parameter schlechtere Ergebnisse bezüglich der Klassifikation von Proteinsequenzen, d.h. eine höhere Anzahl an Missklassifikationen, liefern als die von ihnen empirisch ermittelten Parameter. Die Werte dieser empirisch ermittelten Parameter sind

$$\vec{\alpha}_M = (0.794, 0.095, 0.005),\ \vec{\alpha}_I = (0.333, 0.667) \text{ und } \vec{\alpha}_D = (0.278, 0.222). \tag{2.40}$$

Kapitel 3

Jumping Alignments

Jumping Alignments sind eine von Rainer Spang, Marc Rehmsmeier und Jens Stoye [24, 25] entwickelte Methode zur Proteinklassifizierung und Entdeckung neuer, möglicherweise entfernt verwandter Mitglieder einer Proteinfamilie in einer gegebenen Datenbank. Der entsprechende *Jumping-Alignment-Algorithmus* (JALI) bedient sich dem Prinzip der Dynamischen Programmierung und kann als eine Erweiterung des Smith-Waterman-Algorithmus (siehe Anhang) gesehen werden.
Für jede einzelne Sequenz (*Anfragesequenz*) der Datenbank wird die Ähnlichkeit zur Proteinfamilie durch ein lokales Alignment zum multiplen Sequenzalignment (MSA) der Familie bestimmt. Dabei wird jedoch im Gegensatz zu Profil-HMMs nicht die Verteilung der Aminosäuren in den einzelnen Spalten des MSA berücksichtigt. Stattdessen wird jede Position der Anfragesequenz zu jeweils einer Sequenz, der sog. *Referenzsequenz* des MSA lokal aligniert, wobei die Referenzsequenz innerhalb des Alignments wechseln kann. Solch ein Wechsel wird als *Sprung* (engl. *jump*) bezeichnet und bei der Berechnung des Alignment-Scores bestraft. Aufgrund der möglichen Sprünge innerhalb des Alignments wird dieses lokale Alignment der Anfragesequenz zum MSA *Jumping Alignment* genannt.
Der Score dieses Alignments, der sog. *Jumping-Alignment-Score*, ist eine direkte Verallgmeinerung des lokalen Alignment-Scores des Smith-Waterman-Algorithmus. Er entspricht dem Score des paarweisen Alignments der Anfragesequenz mit der jeweiligen Referenzsequenz, wobei für jeden Sprung innerhalb des Alignments, d.h. für jedes Wechseln der Referenzsequenz, eine bestimmte Konstante, die sog. *jump-cost* abgezogen wird. Gesucht wird der maximale Jumping-Alignment-Score, d.h. der Score des besten Jumping Alignments der Anfragesquenz mit dem MSA der Proteinfamilie. Dieser gibt an, wie ähnlich die Anfragesequenz zur Proteinfamilie ist, und ob sie möglicherweise Mitglied dieser Familie ist. Der Jumping-Alignment-Score wird mithilfe des folgenden Algorithmus bestimmt.

3.1 Jumping-Alignment-Algorithmus

Sei eine Proteinfamilie $\mathcal{S} := \{S_1, \ldots, S_K\}$ und eine Anfragesequenz $s[1,l]$ in Σ^l gegeben. Sei

$$A = (a_{k,j})_{1 \leq k \leq K, 1 \leq j \leq N}, \quad a_{k,j} \in \Sigma \cup \{-\} \tag{3.1}$$

das multiple Sequenzalignment dieser Familie.
Sei a'_k die k-te Zeile in A, aus der wir durch Streichen aller Lücken '$-$' die Sequenz $S_k \in \mathcal{S}$ erhalten.
Zu jeder Zeile a'_k in A definieren wir eine Matrix

$$D_k := ((d_{t,j})_{1 \leq t \leq l, 1 \leq j \leq N})_k, \quad k = 1, \ldots, K,$$

wobei $(d_{t,j})_k$ der maximale Score aller Jumping Alignments des Präfixes $s[1,t]$ von $s[1,l]$ mit den ersten j Spalten des Alignments A, endend an Position j der Zeile a'_k, sei.
Sei $gapcost > 0$ der Wert, mit dem das Einfügen einer Lücke in einer Sequenz bestraft wird, und $jumpcost > 0$ der entsprechende Wert für den Wechsel der Referenzsequenz im MSA.
Sei

$$G_k(j) := \begin{cases} 0 & \text{, falls} \quad a_{k,j} = \text{'}-\text{'} \\ gapcost & \text{, sonst} \end{cases} \tag{3.2}$$

der Wert, mit dem das Einfügen einer Lücke in der Anfragesequenz bestraft wird. Falls im Alignment an der entsprechenden Position bereits eine Lücke eingefügt ist, soll dies nicht bestraft werden.
Sei $w := (w_{i,j})_{1 \leq i,j \leq n}$ eine Scoringmatrix, die jedem Paar von Symbolen $(\sigma_i, \sigma_j) \in \Sigma \times \Sigma$ einen bestimmten Wert zuordnet.
Dann lassen sich die Matrizen $D_k(t,j), k = 1, \ldots, K$ wie folgt rekursiv, jeweils von links oben nach rechts unten, berechnen:

$$D_k(t,j) = \max \begin{cases} 0 & \\ D_k(t-1, j-1) + w(s_t, a_{k,j}) & \\ D_k(t-1, j) - gapcost & \\ D_k(t, j-1) - G_k(j) & \\ D_{k'}(t-1, j-1) + w(s_t, a_{k,j}) - jumpcost & , \forall k' \neq k \\ D_{k'}(t-1, j) - gapcost - jumpcost & , \forall k' \neq k \\ D_{k'}(t-1, j) - G_k(j) - jumpcost & , \forall k' \neq k. \end{cases} \tag{3.3}$$

Der maximale Eintrag in allen Matrizen $D_k, k = 1, \ldots, K$ an Position (l, N) gibt den Jumping-Alignment-Score der Anfragesequenz s mit dem multiplen Sequenzalignment A wieder. Überschreitet er eine geeignet zu wählende Schranke, geht man davon aus, dass die Anfragesequenz Mitglied der Proteinfamilie ist.

3.2 Vergleich von Jumping Alignments und Profil-HMMs

Im Gegensatz zu einem Profil-HMM berücksichtigt ein Jumping Alignment die in einem MSA gegebene *horizontale Information*: Durch das paarweise Alignment der Anfragesequenz zu jeweils einer Referenzsequenz des MSA werden sowohl die Beziehung benachbarter Symbole innerhalb einer Sequenz, als auch konservierte Muster innerhalb einzelner Subfamilien bei der Proteinklassifizierung berücksichtigt. Durch die Sprünge innerhalb des Alignments ist zusätzlich die Möglichkeit gegeben, *vertikale Information* zu berücksichtigen, indem beachtet wird welche Symbole an der entsprechenden Position in den anderen Sequenzen vorkommen. Dadurch wird jedoch im Gegensatz zu einem Profil-HMM nicht auf die Konserviertheit der Spalten des Alignments eingegangen, denn *mismatches* der Anfragesequenz mit Symbolen einer konservierten Spalte werden in gleichem Maße bestraft, wie *mismatches* mit einer nicht so stark konservierten Spalte des Alignments.
Ein Profil-HMM berücksichtigt die im MSA gegebene vertikale Information sehr viel stärker als ein Jumping Alignment. Dadurch, dass jeweils die Häufigkeit aller in einer Spalte des MSA vorkommenden Symbole berücksichtigt wird, können stark konservierte Bereiche sehr gut von Regionen mit hoher Variabilität unterschieden werden.
Im folgenden Kapitel stellen wir ein Modell vor, dass die Vorteile der beiden genannten Methoden vereint, indem es sowohl die horizontale als auch die vertikale Information eines MSA für das Alignment einer Anfragesequenz zum MSA nutzt. Dieses Modell nennen wir ein *springendes Profil-HMM*.

Kapitel 4

Ein Springendes Profil-Hidden-Markov-Modell

Sequenzen von HI- und Hepatitis-C-Viren lassen sich in verschiedene Typen, Subtypen und Sub-Subtypen klassifizieren, die durch gleichzeitige Infektion eines Individuums mit verschiedenen (Sub)Typen rekombiniert werden können. Die genomischen Sequenzen der dadurch entstehenden neuen Virus-(Sub)Typen setzen sich somit aus Sequenzen verschiedener Subtypen zusammen. Ziel ist es, für eine unbestimmte, möglicherweise rekombinante Sequenz (*Anfragesequenz*) die Subtypen zu bestimmen, aus denen sie sich zusammensetzt, und die entsprechenden Bereiche in der Sequenz zu identifizieren.
Wir nehmen an, wir haben eine Sequenzfamilie $\mathcal{S} := \{S_1, \ldots, S_K\}$ gegeben die sich in K' Subtypen $\mathcal{S}_1, \ldots, \mathcal{S}_{K'}$ unterteilen lässt, so dass jede Sequenz S_i zu genau einem Subtyp $\mathcal{S}_j$ gehört. Sei A das multiple Sequenzalignment dieser Familie, das sich entsprechend der Subtypen in K' *Subalignments* unterteilen lässt. Sei $s[1, l]$ eine Anfragesequenz, deren Rekombination aus den gegebenen Subtypen identifiziert werden soll.
Die Ähnlichkeit der Anfragesequenz zu jedem *einzelnen* Subtyp der Familie können wir bestimmen, indem wir, wie in Kapitel 2 beschrieben, ein Profil-HMM des entsprechenden Subalignments entwickeln, und den Viterbi-Pfad der Sequenz durch dieses Profil-HMM bestimmen. Um nun verschiedene Bereiche der Sequenz *unterschiedlichen* Subtypen zuordnen zu können, nutzen wir eine Verallgemeinerung der Idee des Jumping-Alignment-Algorithmus (JALI) (Kap. 3): Wir lassen Übergänge zwischen den Profil-HMMs der einzelnen Subalignments von A zu, so dass jede Position der Anfragesequenz zu jeweils einem Subalignment von A lokal aligniert werden kann. Solch einen Übergang zwischen zwei verschiedenen Profil-HMMs nennen wir in Anlehnung an JALI einen *Sprung* (engl. *jump*).
Das Modell, das wir durch Verbinden der Profil-HMMs aller Subalignments von A durch diese Übergänge erhalten, ist ein HMM, das wir ein *springendes Profil-HMM* (engl. *jumping profile HMM*) (*jpHMM*) nennen. Eine Vorhersage über die

Rekombination der Anfragesequenz aus den gegebenen Subtypen können wir mithilfe des Viterbi-Pfades der Sequenz durch das jpHMM treffen: Da jeder Zustand des Viterbi-Pfades zu einem bestimmten Profil-HMM gehört, können wir jeder Position der Sequenz den Subtyp zuordnen, zu dessen Profil-HMM der Zustand im Viterbi-Pfad gehört, der das Symbol an dieser Position emittiert hat, und erhalten so die Rekombination der Sequenz. Das Alignment der Sequenz $s[1,l]$ zum MSA A, das durch den Viterbi-Pfad der Sequenz durch das jpHMM definiert ist, nennen wir wie in JALI das *Jumping Alignment*.

4.1 Definition eines jpHMMs

Sei $\mathcal{S} := \{S_1, \ldots, S_K\}$ eine Sequenzfamilie, die sich in K' Subtypen $\mathcal{S}_1, \ldots, \mathcal{S}_{K'}$ unterteilen lässt, so dass jede Sequenz S_i zu genau einem Subtyp $\mathcal{S}_j$ gehört. Es gilt also

$$\mathcal{S}_i \cap \mathcal{S}_j = \emptyset \quad \forall i \neq j, \mathcal{S}_i, \mathcal{S}_j \subset \mathcal{S} \quad \text{und} \quad \bigcup_{i=1}^{K'} \mathcal{S}_i = \mathcal{S}. \tag{4.1}$$

Definition 4.1 *Sei*

$$A = (a_{k,j})_{1 \leq k \leq K, 1 \leq j \leq N}, \qquad a_{k,j} \in \Sigma \cup \{-\}$$

das multiple Sequenzalignment der Familie $\mathcal{S}$.
Sei $S_k, k = 1, \ldots, K$, die Sequenz in $\mathcal{S}$, die man durch Streichen aller Lücken in der k-ten Zeile des Alignments A erhält.
Dann ist das Subalignment A_i in A der Sequenzen des Subtyps $\mathcal{S}_i \subset \mathcal{S}$ in A definiert durch

$$A_i := (a_{k,j})_{S_k \in \mathcal{S}_i, 1 \leq j \leq N} \tag{4.2}$$

Ein Beispiel für ein Alignment mit zwei Subalignments ist in Abbildung 4.1 gegeben.

Definition 4.2 *Sei S_k die Sequenz in $\mathcal{S}$, die man durch Streichen aller Lücken in der k-ten Zeile des Alignments A erhält.*
Sei A_i ein Subalignment in A und $\vec{a}_j^{\,i}$ der j-te Spaltenvektor in A_i. Dann definiert

$$|\vec{a}_j^{\,i}| := \#\{a_{k,j} \,|\, a_{k,j} \neq \text{'}-\text{'}, S_k \in \mathcal{S}_i\}, \quad \forall j = 1, \ldots, N. \tag{4.3}$$

die Anzahl aller Symbole in der j-ten Spalte von A_i.

Zum Konsensusspaltenkriterium c (siehe Kap. 2, 2.3) für eine Spalte j eines Subalignments definieren wir ein zusätzliches Kriterium:
Eine Spalte j eines Subalignments A_i ist genau dann eine Konsensusspalte in A_i, wenn sie das Konsensusspaltenkriterium c (2.3) erfüllt, oder wenn die Anzahl der Symbole in dieser Spalte mindestens gleich einer bestimmten Schranke $t \geq 1$ ist.

Definition 4.3 *Sei A_i ein Subalignment in A und K_i die Anzahl der Zeilen in A_i. Eine Spalte j in A_i ist genau dann eine* Konsensusspalte *in A_i, wenn gilt*

$$(|\vec{a}_j^i| \geq c \cdot K_i) \vee (|\vec{a}_j^i| \geq t). \tag{4.4}$$

Dieses Kriterium führen wir ein, da die Anzahl der Sequenzen der verschiedenen Subtypen sehr stark variieren kann (siehe Kap. 6). Wählen wir z.B. als Konsensusspaltenkriterium $c = 0.5$, so ist eine Spalte eines Subtyps mit zwei Sequenzen bereits eine Konsensusspalte, wenn eine der beiden Sequenzen an dieser Position ein Symbol enthält. In einem Subtyp bestehend aus 100 Sequenzen müssen dagegen mindestens 50 Sequenzen ein Symbol an dieser Position enthalten, damit das Konsensusspaltenkriterium erfüllt ist. Durch das Kriterium t ist die Möglichkeit gegeben, das Konsensusspaltenkriterium c abzuschwächen.
Zusätzlich dazu muss folgendes Problem beachtet werden: Im Alignment können Subtypen enthalten sein, die in den ersten bzw. letzten Spalten hauptsächlich Lücken enthalten. Dies ist bei den von uns betrachteten Daten (Kap. 6) in einigen Subtypen aufgrund fehlender Annotation von Sequenzen im Anfangs- und Endbereich der Fall. Da diese Spalten keine Konsensusspalten sind, werden diesen Spalten auch keine Zustände zugeordnet. Ist nun eine Anfragesequenz gegeben, die einem bestimmten Subtyp $\mathcal{S}_i$ ähnelt, jedoch beispielsweise am Anfang einen langen Insertionsbereich hat, so kann dieser Bereich durch den ersten, $\mathcal{S}_i$ zugeordneten, Insert-Zustand I_B (siehe Kap. 2) emittiert werden. Da es im MSA jedoch Subtypen geben kann, die in diesen Bereichen viele Konsensusspalten haben, kann es sein, dass aufgrund der möglicherweise sehr grossen Unterschiede zwischen den Emissionswahrscheinlichkeiten der Match- und Insert-Zustände zu Beginn des Jumping Alignments ein Sprung in einen Match-Zustand eines Subtyps $\mathcal{S}_j$ mit vielen Konsensusspalten bevorzugt wird, und die Sequenz im Anfangsbereich zu diesem Subtyp $\mathcal{S}_j$ aligniert wird. Aufgrund der Bestrafung von Sprüngen zwischen den veschiedenen Profil-HMMs kann es dadurch trotz möglicherweise sehr viel niedrigerer Match-Emissionswahrscheinlichkeiten in $\mathcal{S}_j$ als im Subtyp $\mathcal{S}_i$, dem die Sequenz ähnelt, passieren, dass kein Sprung zu $\mathcal{S}_i$ stattfindet und die folgenden Positionen der Sequenz dem Subtyp $\mathcal{S}_j$ zugeordnet werden. Das bedeutet, dass für diese Sequenz in diesem Bereich ein falscher Subtyp und somit eine falsche Rekombination vorhergesagt wird. Um dies zu verhindern, fordern wir zusätzlich zu den oben genannten Konsensusspaltenkriterien, dass die erste und die letzte Konsensusspalte aller Subalignments, denen ein Zustand zugeordnet wird, in allen Subalignments gleich ist.

Definition 4.4 *Sei A_i das Subalignment des Subtyps $\mathcal{S}_i, i = 1, \ldots, K'$, in A, und K_i die Anzahl der Zeilen in A_i. Sei $\vec{a}_j^i$ der j-te Spaltenvektor in A_i und $|\vec{a}_j^i|$ die Anzahl aller Symbole in $\vec{a}_j^i$.*
Dann definiert

$$fc \;:=\; \min\left\{ j \in \{1, \ldots, N\} \,|\, (|\vec{a}_j^i| \geq c \cdot K_i) \vee (|\vec{a}_j^i| \geq t), \forall\, i = 1, \ldots, K' \right\} \tag{4.5}$$

die erste gemeinsame *Konsensusspalte (first consensus column) aller Subalignments A_i in A und*

$$lc := \max\left\{j \in \{1,\ldots,N\} \mid (|\vec{a}_j^i| \geq c \cdot K_i) \vee (|\vec{a}_j^i| \geq t), \forall\, i = 1,\ldots,K'\right\} \quad (4.6)$$

die letzte gemeinsame *Konsensusspalte (last consensus column) aller Subalignments A_i in A.*

Definition 4.5 *Sei $A_i, i = 1,\ldots,K'$ ein Subalignment in A und K_i die Anzahl der Zeilen in A_i. Seien fc und lc die erste bzw. die letzte gemeinsame Konsensusspalte aller Subalignments A_i. Sei c das Konsensusspaltenkriterium für alle Subalignments und t die für eine Konsensusspalte geforderte Mindestanzahl an Symbolen. Seien $M_{i,j}, I_{i,j}$ und $D_{i,j}$ die drei der Konsensusspalte j des Subalignments A_i zugeordneten Match-, Insert- und Delete-Zustände. Dann definieren wir*

$$Z_i := \left\{M_{i,j}, I_{i,j}, D_{i,j} \mid fc \leq j \leq lc, (|\vec{a}_j^i| \geq c \cdot K_i) \vee (|\vec{a}_j^i| \geq t), j = 1,\ldots,N\right\}. \quad (4.7)$$

Dies ist die Menge aller, dem Subalignment A_i zugeordneten Zustände.

Bemerkung 4.6 Bezeichnen wir den Insert-Zustand, der Insertionen vor der ersten Konsensusspalten im Subalignment A_i erlaubt, mit $I_{i,B}$, so ist $Z_i \cup \{I_{i,B}\}$ der Zustandsraum des Profil-HMMs $\mathcal{P}_i$ für das Subalignment A_i.

Die Konsensusspaltenkriterien c und t müssen also für die jeweilige Spalte eines Subalignments erfüllt sein, nicht aber für die entsprechende Spalte des Alignments A. Somit kann eine Spalte j des Alignments A in einem bestimmten Subalignment eine Konsensusspalte sein, während sie es in einem anderen Subalignment nicht ist. Jede Spalte j eines Subalignments A_i ist Teil der Spalte j im MSA A. Ist nun ein Zustand z der Spalte j in einem Subalignment A_i zugeordnet, so ist er ebenfalls der Spalte j in A zugeordnet. Sei $j := \mathrm{col}_{A_i}(z)$ die Spalte in A_i der der Zustand z zugeordnet ist. Dann gilt also (siehe Def. (2.6))

$$\mathrm{col}(z) := \mathrm{col}_{A_i}(z) = j$$

4.1.1 Zustandsraum eines jpHMMs

Ein jpHMM eines MSA mit K' Subalignments kann als die Vereinigung der entsprechenden K' Profil-HMMs gesehen werden, die durch bestimmte Übergänge miteinander verbunden sind. Der Zustandsraum eines jpHMMs setzt sich also aus den Zustandsräumen der Profil-HMMs $\mathcal{P}_i, i = 1,\ldots,K'$, zusammen. Dabei müssen wir allerdings folgendes berücksichtigen:
Der Zustandsraum des Profil-HMMs $\mathcal{P}_i$ enthält einen Insert-Zustand $I_{i,B}$ (siehe Bem. 4.6), der Insertionen vor der ersten Konsensusspalte des Subalignments A_i erlaubt, und einen Insert-Zustand, hier $I_{i,lc}$ genannt, der der letzten Konsensusspalte lc in

A_i zugeordnet ist und Insertionen nach dieser Spalte erlaubt. Wir führen nun zwei Insert-Zustände I_B und I_E ein, die die Funktion aller Zustände $I_{i,B}$ bzw. $I_{i,lc}$ beinhalten. I_B ist ein Insert-Zustand, dessen Nachfolger alle Match-Zustände der Z_i sind, die der ersten Konsensusspalte eines Subalignments A_i zugeordnet sind, und I_B selbst. Er erlaubt also Insertionen vor der ersten Konsensusspalte jedes Subalignments A_i. Der Insert-Zustand I_E ist Nachfolger von sich selbst und von allen Match-Zuständen, die der letzten Konsensusspalte eines Subalignments in A zugeordnet sind, und erlaubt somit Insertionen nach der jeweils letzten Konsensusspalte aller Subalignments. Die Insert-Zustände $I_{i,B}$ bzw. $I_{i,lc}$ aller Profil-HMMs $\mathcal{P}_i$, $i = 1, \ldots, K'$, sind deshalb nicht im Zustandsraum Z des jpHMMs enthalten.
Zusätzlich zu diesen beiden Insert-Zuständen führen wir zwei Delete-Zustände D_B und D_E ein, die ein *lokales Alignment* der Anfragesequenz zum MSA erlauben. Der Delete-Zustand D_B ist Nachfolger des Start-Zustandes B und es sind Übergänge in alle Match-Zustände aller Profil-HMMs $\mathcal{P}_i$ möglich, die nicht der ersten Konsensusspalte eines Subalignments zugeordnet sind. Von allen Match-Zuständen aller Profil-HMMs, die nicht der letzten Konsensusspalte eines Subalignments zugeordnet sind, ist ein Übergang in den Delete-Zustand D_E möglich, dessen einziger Nachfolger der End-Zustand E ist. Dadurch ist die Möglichkeit gegeben, vom Start-Zustand in einen beliebigen Match-Zustand und von jedem Match-Zustand in den End-Zustand zu gelangen, ohne eine lange Kette von Delete-Zuständen durchlaufen zu müssen. Dies ist beispielsweise sinnvoll, falls die Anfragesequenz sehr viel kürzer ist als das MSA und nur zu einem Teil davon aligniert wird.

Definition 4.7 (Zustandsraum eines jpHMMs) *Sei lc die letzte gemeinsame Konsensusspalte aller Subalignments A_i, $i = 1, \ldots, K'$, von A. Sei Z_i (4.7) die Menge aller, dem Subalignment A_i zugeordneten Zustände, und $I_{i,lc}$ der Insert-Zustand, der der letzten Konsensusspalte in A_i zugeordnet ist.*
Sei I_B der Insert-Zustand, der Insertionen vor der jeweils ersten, und I_E der Insert-Zustand, der Insertionen nach der jeweils letzten Konsensusspalten aller Subalignments erlaubt. Seien D_B und D_E zwei Delete-Zustände. Dann definiert

$$Z := \Big(\bigcup_{i=1}^{K'} Z_i \backslash I_{i,lc} \Big) \cup \{I_B, I_E, D_B, D_E\} \tag{4.8}$$

den Zustandsraum eines jpHMMs *für das Alignment A mit K' Subalignments.*

Definition 4.8 *Für die Zustände I_B, I_E, D_B und D_E definieren wir*

$$\begin{aligned} \mathrm{col}(D_B) &:= 0, \\ \mathrm{col}(I_B) &:= 0, \\ \mathrm{col}(I_E) &:= N + 1, \\ \mathrm{col}(D_E) &:= N + 1. \end{aligned}$$

4.1.2 Übergänge innerhalb eines jpHMMs

Die Zustände eines jpHMMs sind, wie für ein HMM, durch Übergänge verbunden, denen Übergangswahrscheinlichkeiten zugeordnet sind. Übergänge sind zwischen bestimmten Zuständen *eines* Profil-HMMs $\mathcal{P}_i$, zwischen bestimmten Zuständen *verschiedener* Profil-HMMs und von bzw. zu den Zuständen B, I_B, I_E, D_B, D_E und E möglich.
Für die Übergänge *innerhalb* eines jeden Profil-HMMs $\mathcal{P}_i$ gelten die in Kap. 2, Def. 2.10, genannten Bedingungen. D.h. für ein Profil-HMM $\mathcal{P}_i$ sind Übergänge von einem Match-Zustand in Z_i in den derselben Konsensusspalte in A_i zugeordneten Insert-Zustand und in den der nächsten Konsensusspalte in A_i zugeordneten Match- und Delete-Zustand, von einem Delete-Zustand in den nächsten Match- und Delete-Zustand, und von einem Insert-Zustand zu sich selbst und in den Match-Zustand der nächsten Konsensusspalte in A_i möglich.
Zwischen den einzelnen Profil-HMMs des jpHMMs sind ebenfalls nur bestimmte Übergänge bzw. *Sprünge* möglich. Es gilt:
Ein Sprung von einem Profil-HMM $\mathcal{P}_i$ in ein Profil-HMM $\mathcal{P}_j$ ist nur möglich

r1 von einem Zustand $z_k \in Z_i$ in einen Zustand $z_l \in Z_j$, der der *nächsten auf* $\mathrm{col}(z_k)$ *folgenden* Konsensusspalte in A_j zugeordnet ist

r2 von einem Zustand $z_k \in Z_i$ in einen Zustand $z_l \in Z_j$, der einer Konsensusspalte in A_j zugeordnet ist, die *nicht grösser* ist als die nächste auf $\mathrm{col}(z_k)$ folgende Konsensusspalte in A_i

r3 von einem Match-Zustand in einen Match- und einen Delete-Zustand $(M \rightarrow \{M, D\})$, und von einem Insert- und einem Delete-Zustand in einen Match-Zustand $(\{I, D\} \rightarrow M)$

Die Regeln **r1** und **r2** verhindern, dass grosse Bereiche eines Subtyps 'übersprungen' werden können, d.h. dass Deletionen in der Anfragesequenz eingefügt werden, ohne dass dafür Delete-Zustände angenommen werden.
Durch Regel **r3** wird die Anzahl an möglichen Übergängen und somit die Anzahl an möglichen Pfaden durch das jpHMM verringert. Dies ist notwendig damit die Laufzeit und der benötigte Arbeitsspeicher des Programms (Folg. 4.11) für Sequenzfamilien mit vielen Subtypen nicht zu gross wird. Bei der Auswahl der möglichen Übergänge spielten folgende Überlegungen eine Rolle:
Für jeden Zustand eines Profil-HMMs $\mathcal{P}_i$ muss es möglich sein, in einen Match-Zustand eines anderen Profil-HMMs $\mathcal{P}_j$ zu springen (falls die Bedingungen $r1$ und $r2$ erfüllt sind) und somit die nächste Position der Anfragesequenz zu dem Subalignment A_j zu alignieren. Wie auch innerhalb eines Profil-HMMs sind Sprünge zwischen Insert- und Delete-Zuständen verschiedener Profil-HMMs ausgeschlossen. Ebenfalls ausgeschlossen ist ein Sprung innerhalb eines Deletionsbereiches, d.h. Sprünge zwischen zwei Delete-Zuständen verschiedener Profil-HMMs sind nicht erlaubt. Zusätz-

lich dazu schliessen wir einen Sprung von einem Match- bzw. Insert-Zustand eines Profil-HMMs $\mathcal{P}_i$ in einen Insert-Zustand eines Profil-HMMs $\mathcal{P}_j$ aus. Dies stellt jedoch keine grosse Einschränkung dar, da durch die Möglichkeit eines Sprungs von einem Insert- in einen Match-Zustand für jede Position die Möglichkeit einer Insertion gegeben ist. Der Ausschluss dieser Übergänge hat somit lediglich Einfluss auf die Wahrscheinlichkeit einer Insertion an der entsprechenden Position.

Für die Zustände B, I_B, I_E, D_B, D_E und E haben wir bereits in Abschnitt 4.1.1 einige der möglichen Übergänge genannt. Zusätzlich zum Übergang von B nach D_B, sind vom Start-Zustand B Übergänge in alle Match- und Delete-Zustände der Profil-HMMs $\mathcal{P}_i, i = 1, \ldots, K'$, möglich, die der ersten Konsensusspalte des entsprechenden Subalignments A_i in A zugeordnet sind, und in den Insert-Zustand I_B. Übergänge in den End-Zustand E sind, zusätzlich zu den Übergängen von I_E nach E und von D_E nach E, von allen Match-Zuständen, die der letzten Konsensusspalte eines Subalignments A_i in A zugeordnet sind, möglich.
Ein Beispiel für ein jpHMM für ein multiples Sequenzalignment mit zwei Subtypen ist in Abb. 4.1 gegeben. Abb. 4.2 zeigt die allgmeine Architektur eines jpHMMs, wobei der Übersicht halber nur ein Teil der möglichen Übergänge und Sprünge eingezeichnet ist. Die Pfeile an den Delete-Zuständen D_B und D_E deuten die Vielzahl der für diese Zustände möglichen Übergänge an.

Definition 4.9 (Springendes Profil-Hidden-Markov-Modell)
Sei $Z^+ := Z \cup \{B, E\}$, $Z^- := Z \backslash \{I_B, D_B, I_E, D_E\}$ *und*
$Z' := \{z_k \in Z \,|\, \mathrm{typ}(z_k) = D\}$ *die Menge aller stummen Zustände in* Z. *Sei für einen Zustand* $z \in Z^-$ $\mathcal{S}(z)$ *der Subtyp in* $\mathcal{S}$, *zu dessen Zustandsraum der Zustand* z *gehört, und* $A_{\mathcal{S}(z)}$ *das entsprechende Subalignment in* A.
Ein springendes Profil-Hidden-Markov-Modell (jpHMM) *ist ein HMM mit Parametern* $M = (\Sigma^+, Z^+, \mathcal{T}, \mathcal{E})$, *für das folgende Bedingungen erfüllt sind:*

1.) $\forall z_k, z_l \in Z^-$ *mit* $\mathcal{S}(z_k) = \mathcal{S}(z_l)$ *erfülle* t_{z_k, z_l} *die Bedingung (2.10)*

2.) sei für alle $z_k \in Z$ j_k *die nächste auf* $\mathrm{col}(z_k)$ *folgende Konsensusspalte in* $A_{\mathcal{S}(z_k)}$, *d.h.* $j_k := \min\limits_{k > \mathrm{col}(z_k)} \left\{k \,\middle|\, |\vec{a}_k^i| \geq c \cdot K_i, \mathcal{S}_i = \mathcal{S}(z_k)\right\}$.

Dann gelte $\forall z_k, z_l \in Z^-$ *mit* $\mathcal{S}(z_k) \neq \mathcal{S}(z_k)$

$$t_{z_k,z_l} = 0 \textit{ falls } \begin{cases} & z_l \prec_A z_k, \\ \textit{oder} & (\mathrm{typ}(z_k) = M) \wedge (\mathrm{typ}(z_l) = I) \\ \textit{oder} & (\mathrm{typ}(z_k) = I) \wedge (\mathrm{typ}(z_l) = D) \\ \textit{oder} & (\mathrm{typ}(z_k) = D) \wedge (\mathrm{typ}(z_l) = I) \\ \textit{oder} & \mathrm{typ}(z_k) = \mathrm{typ}(z_l) \in \{I, D\} \\ \textit{oder} & \mathrm{col}(z_l) > j_k, \\ \textit{oder} & \exists z_i \in Z^- : \begin{cases} & (\mathcal{S}(z_i) = \mathcal{S}(z_l)) \\ \wedge & (z_k \prec_A z_i \prec_A z_l) \\ \wedge & (\mathrm{col}(z_i) < j_k) \\ \wedge & (\mathrm{typ}(z_i) = \mathrm{typ}(z_l)). \end{cases} \end{cases}$$

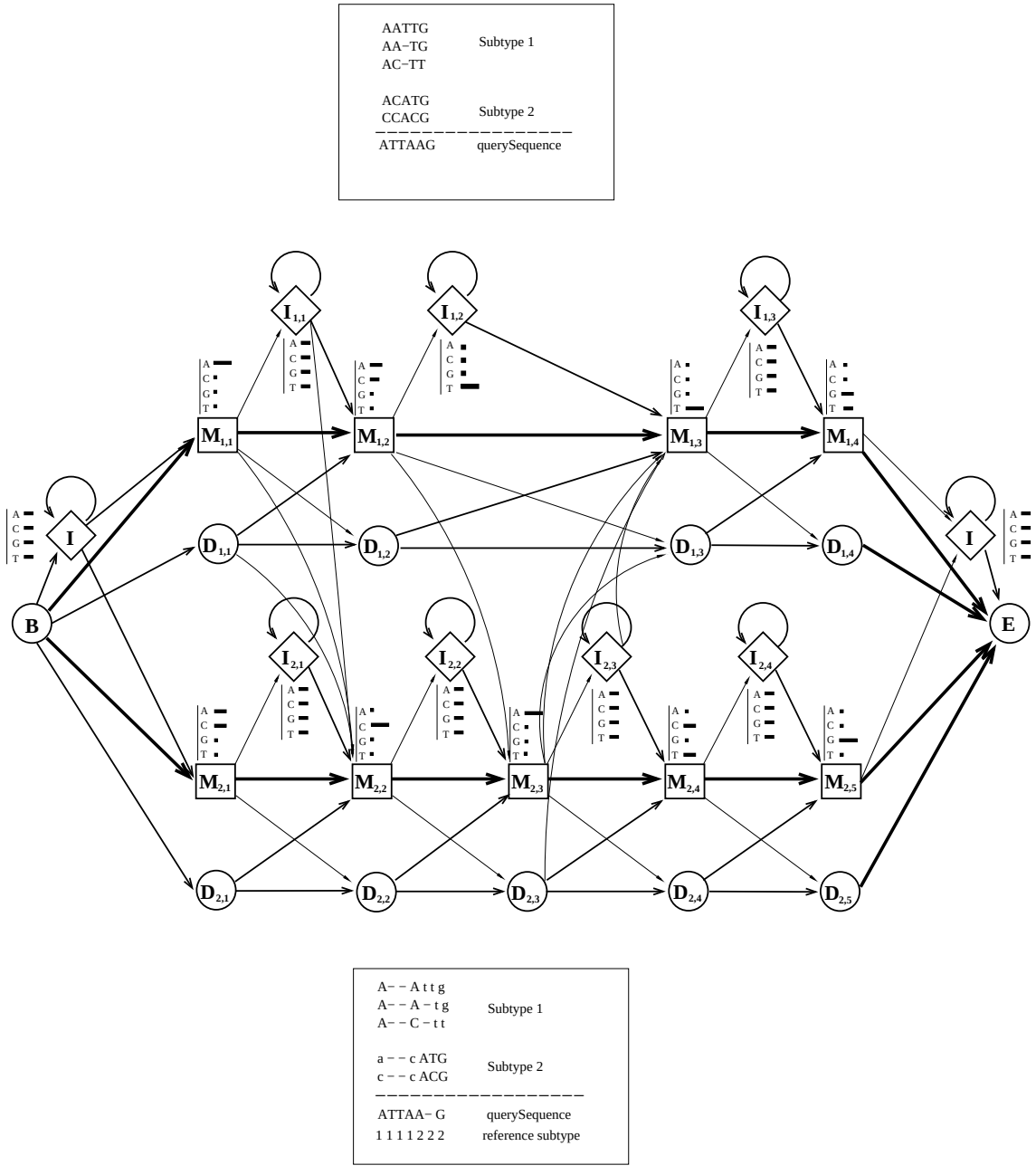
3.) für alle $z_k \in Z^-$ *gelte*

$$t_{B,z_k} = 0, \textit{ falls } \exists z_i \in Z^- : \begin{cases} & (\mathcal{S}(z_i) = \mathcal{S}(z_k)) \\ \wedge & (z_i \prec_A z_k) \\ \wedge & (\mathrm{typ}(z_i) = \mathrm{typ}(z_k)), \end{cases}$$

$$t_{z_k,E} = 0, \textit{ falls } \exists z_i \in Z^- : \begin{cases} & (\mathcal{S}(z_i) = \mathcal{S}(z_k)) \\ \wedge & (z_k \prec_A z_i) \\ \wedge & (\mathrm{typ}(z_i) = \mathrm{typ}(z_k)), \end{cases}$$

4.) für alle $z_k \in Z^+$ *gelte*

$$\begin{aligned} t_{z_k,B} &= 0, && \forall z_k \in Z^+, \\ t_{B,I_E} &= t_{B,D_E} = 0, \\ t_{I_B,z_k} &= 0, && \forall z_k \in \{D_B, I_E, D_E, E\}, \\ t_{I_E,z_k} &= 0, && \forall z_k \in \{D_B, I_B, D_E\}, \\ t_{D_B,z_k} &= 0, && \forall z_k \in \{D_B, I_B, I_E, D_E, E\}, \\ \textit{und} \quad t_{D_E,z_k} &= \begin{cases} 0, & \forall z_k \in \{D_B, I_B, I_E, D_E\}. \\ 1, & \textit{falls} \quad z_k = E. \end{cases} \end{aligned}$$



viterbi path (gives us the alignment of the query to the alignment):

Abbildung 4.1: Ein Beispiel für ein jpHMM für ein MSA von zwei Subtypen. Der erste Subtyp besteht aus drei DNA-Sequenzen mit vier Konsensusspalten, der zweite Subtyp aus zwei Sequenzen mit fünf Konsensusspalten. Jeder Match- und Insert-Zustand trägt einen Vektor von vier Emissionswahrscheinlichkeiten für die Nukleotide A, C, G und T. Die Pfeile stellen mögliche Übergänge zwischen den Zuständen dar. Der Übersicht halber sind nur einige der möglichen Übergänge angegeben. Auch die Delete-Zustände D_B und D_E sind nicht angegeben. Die unterschiedliche Dicke der Pfeile deutet auf die Grössenverhältnisse der Übergangswahrscheinlichkeiten hin.

4.2 Viterbi-Algorithmus

Sei eine Anfragesequenz $s[1,l]$ gegeben. Dann kann mithilfe des Viterbi-Pfades der Sequenz durch das jpHMM eine Vorhersage über die Rekombination dieser Sequenz aus den in der Familie $\mathcal{S}$ gegebenen Subtypen getroffen werden:
Sei $v[1,r]$ der Viterbi-Pfad der Sequenz $s[1,l]$. Jeder Zustand des Pfades v ist einer Spalte eines bestimmten Subalignments A_i in A und somit einem bestimmten Subtyp $\mathcal{S}_i$ der Familie $\mathcal{S}$ zugeordnet. Ordnen wir jeder Position der Anfragesequenz den Subtyp zu, dem der Zustand in v, der das Symbol an dieser Position erzeugt, zugeordnet ist, so erhalten das *Jumping Alignment* der Anfragesequenz zum MSA A und ihre Rekombination aus den Subtypen der Familie $\mathcal{S}$.
Den Viterbi-Pfad der Sequenz $s[1,l]$ können wir mithilfe des in Kap. 1, Algo. 1.31, angegebenen Viterbi-Algorithmus berechnen, da ein jpHMM ein HMM ist, das bestimmte zusätzliche Bedingungen erfüllt. Um dabei das in Abschnitt 2.2 beschriebene Problem der möglicherweise zu grossen Anzahl an Multiplikationen in der Viterbi-Rekursion zu vermeiden, führen wir den Viterbi-Algorithmus wie für ein Profil-HMM für die log-transformierten Werte der Viterbi-Variablen durch. D.h. wir nutzen den in Algo. 2.16 beschriebenen log-Viterbi-Algorithmus um den Viterbi-Pfad der Sequenz $s[1,l]$ zu berechnen. Dabei muss ebenfalls die Reihenfolge berücksichtigt werden, in der die Viterbi-Variablen $\tilde{\delta}_{z,i}$ für einen bestimmten Zeitpunkt i berechnet werden, so dass für einen stummen Zustand z nicht auf noch nicht berechnete Werte zugegriffen werden kann (siehe Bem. 1.22):
Jeder Zustand $z \in Z\backslash\{I_B, D_B, I_E, D_E\}$ ist genau einer Spalte j eines Subalignments A_i in A und somit der Spalte j in A zugeordnet. Jeder Zustand z des Zustandsraums Z^+ des jpHMMs ist also genau einer Spalte j im Alignment A zugeordnet, und wir können die in (2.8) für den Zustandsraum Z^+ eines Profil-HMMs definierte Ordnungsrelation $\prec_A$ auf den Zustandsraum Z^+ eines jpHMMs übertragen.
Da zwischen Zuständen verschiedener Profil-HMMs des jpHMMs, die der gleichen Spalte in A zugeordnet sind, keine Sprünge möglich sind (siehe Def. 4.9), können wir die Zustände in Z entsprechend der Spalten in A denen sie zugoerdnet sind sortieren, und erhalten so eine topologische Sortierung des Sub-Zustandsgraphen $G = (Z, T')$. Seien die Zustände in Z von 1 bis m indiziert, d.h. sei $Z := \{z_1, \ldots, z_m\}$. Für je zwei Zustände $z_i, z_j \in Z$ gelte

$$i < j, \quad \text{falls} \quad z_i \prec_A z_j \tag{4.9}$$

Dann können wir den Viterbi-Pfad der Sequenz $s[1,l]$ mithilfe des *log-Viterbi-Algorithmus* (Kap. 2, Algo. 2.16) berechnen.

Bemerkung 4.10 Für einen Zustand z berechnen wir die *log-Viterbi-Variable* $\tilde{\delta}_{z,i}$ zu einem bestimmten Zeitpunkt i, indem wir das Maximum (siehe Algo. 2.16) über alle Vorgänger von z bilden. Da ein Match-Zustand höchstens drei Zustände in jedem Profil-HMM des jpHMMs und den Delete-Zustand D_B als Vorgänger hat, ein Delete-Zustand eines Profil-HMMs $\mathcal{P}_i$ zwei Vorgänger in $\mathcal{P}_i$ und höchstens $K'-1$ Vorgänger

in den Profil-HMMs $\mathcal{P}_j, j \neq i$, und ein Insert-Zustand eines Profil-HMMs $\mathcal{P}_i$ nur zwei Vorgänger in $\mathcal{P}_i$, müssen für einen Zustand $z \in Z \backslash \{I_B, D_B, I_E, D_E\}$ höchstens $3K' + 1$ Werte berechnet werden, um das Maximum zu erhalten. Lediglich für den Delete-Zustand D_E muss das Maximum über alle Match-Zustände des jpHMMs gebildet werden. Der Insert-Zustand I_B hat nur B und sich selbst als Vorgänger, und der Insert-Zustand I_E sich selbst und K' Match-Zustände des jpHMMs.

Folgerung 4.11 (Laufzeit und Speicherbedarf) *Sei $s[1, l]$ eine Beobachtung der Länge l. Sei K' die Anzahl aller Subtypen in $\mathcal{S}$ und m die Anzahl aller Zustände in Z. Dann lassen sich sämtliche Viterbi-Variablen, der Viterbi-Score und ein Viterbi-Pfad der Beobachtung $s[1, l]$ mithilfe des log-Viterbi-Algorithmus in Zeit $O(mK'l)$ berechnen.*
Der zur Berechnung des Viterbi-Pfades benötigte Speicherplatz hat eine Komplexität von $O(ml)$, da für jede Position i in $s[1, l]$ die Viterbi-Variable für jeden Zustand $z \in Z$ gespeichert werden muss.

Die Rekombination der Sequenz aus den Subtypen der Familie $\mathcal{S}$ lässt sich aus dem Viterbi-Pfad der Sequenz mithilfe des folgenden Algorithmus ermitteln.

Algorithmus 4.12 (Algorithmus zur Identifikation der Rekombination) Sei $s[1, l]$ die Anfragesequenz und $v[1, r]$ der Viterbi-Pfad dieser Sequenz. Sei $\mathcal{S}(v_i), i = 1, \ldots, r$, der Subtyp in $\mathcal{S}$, zu dessen Zustandsraum der Zustand v_i gehört. Dann erhalten wir die Rekombination $R := R[1, l], R_i \in \{\mathcal{S}_1, \ldots, \mathcal{S}'_K\}$, $i = 1, \ldots, l$, der Sequenz s aus den Subtypen der Familie $\mathcal{S}$ mithilfe des folgenden Algorithmus:

Sei $i = 1, \ldots, l$ die Position in der Anfragesequenz s und $j = 1 \ldots, r$ die Position im Viterbi-Pfad v.

Initialisierung: $i = 1, j = 1$
while $(i < l)$ do {
  if (typ$(v_j) ==$ M oder typ$(v_j) == I$) {
    $R_i = \mathcal{S}(v_j)$;
    $i = i + 1$;
  }
  $j = j + 1$;
}

4.2.1 Laufzeitverbesserung und Speicherreduzierung

Wie man anhand der Laufzeit- und Speicherkomplexität des Viterbi-Algorithmus sieht, kann der Algorithmus für sehr lange Alignments sehr viel Zeit und vor allem

zu viel Speicherplatz benötigen, um den Viterbi-Pfad zu bestimmen, und somit eine Vorhersage über die Rekombination einer Anfragesequenz zu treffen. Dieses Problem wird in Kap. 5, in dem die Implementation des jpHMM-Programms beschrieben und eine Analyse (Abschnitt 5.3) der Laufzeit und des Speicherbedarfs des Programms, insbesondere des Viterbi-Algorithmus, gegeben ist, genauer beschrieben. Als Beispiel betrachten wir dort das von uns zur Auswertung der Genauigkeit der Vorhersage eines jpHMMs (Kap. 6) verwendete multiple Sequenzalignment des HIV-Typs 1, und zeigen unter anderem, dass für dieses Alignment allein der Viterbi-Algorithmus einen Speicherplatz von 15.7 GB benötigt.
Um die Laufzeit des Algorithmus zu verkürzen und den benötigten Speicherplatz einzuschränken, nutzen wir den *Beam-Search-Algorithmus* [13, 17]. Dieser Algorithmus wurde ursprünglich in der Spracherkennung verwendet und basiert auf der Idee, den Suchraum für den Viterbi-Pfad auf 'erfolgversprechende' Pfade des jpHMMs zu beschränken.
Bereits nach nur wenigen Iterationen in der Viterbi-Rekursion liegt eine grosse Menge an zu betrachtenden Pfaden vor. Unter all diesen Pfaden sind jedoch auch solche dabei, für die die Wahrscheinlichkeit, das Präfix der Anfragesequenz bis zu der im aktuellen Rekursionsschritt betrachteten Position zu erzeugen, sehr viel kleiner ist als die Wahrscheinlichkeit des lokal besten Pfades. Diese Pfade betrachten wir als 'irrelevant' und verfolgen sie nicht weiter. Dies können wir erreichen, indem wir für jede Position i der Anfragesequenz nur für bestimmte Zustände z eine modifizierte Viterbi-Variable $\delta'_{z,i} \leq \delta_{z,i}$ berechnen und speichern. Und zwar betrachten wir nur die Zustände z als mögliche, dieser Position zugrundeliegende Zustände, deren modifizierte Viterbi-Variable $\delta'_{z,i}$ nicht sehr viel kleiner ist als die der optimalen Lösung

$$\delta^*_i := \max_{z \in Z} \delta'_{z,i} \tag{4.10}$$

für diese Position. Diese Zustände nennen wir *aktive* Zustände. Die Menge aller aktiven Zustände zu einem bestimmten Zeitpunkt i ist gegeben durch

$$\mathcal{A}_i := \{z \in Z \mid \delta'_{z,i} \geq \mathcal{B}\delta^*_i\}, \qquad 0 < \mathcal{B} \ll 1. \tag{4.11}$$

Der Parameter $\mathcal{B}$ wird die *Beam-Weite* genannt.
Die modifizierte Viterbi-Variable $\delta'_{z,i+1}$ eines Zustandes z an der Position $i+1$ der Sequenz s wird nun nur aus den Vorgängern z' von z berechnet, die zum Zeitpunkt i, bzw. $i+1$ für einen stummen Zustand z, aktiv waren. D.h. es gilt

$$\delta'_{z,i+1} = \begin{cases} e_{z,s_{i+1}} \max\limits_{z' \in \mathcal{A}_i} \{\delta'_{z',i} t_{z',z}\} & \text{, falls typ}(z) \in \{M, I\} \\ \max\limits_{z' \in \mathcal{A}_{i+1}} \{\delta'_{z',i+1} t_{z',z}\} & \text{, falls typ}(z) = D. \end{cases} \tag{4.12}$$

Die modifizierte Viterbi-Variable eines 'inaktiven' Zustandes wird auf 0 gesetzt und muss nicht gespeichert werden. Daraus folgt auch, dass wir die modifizierte Viterbi-Variable $\delta'_{z,i+1}$ nur für die Zustände berechnen, die Nachfolger eines Zustandes in

$\mathcal{A}_i$, bzw. in $\mathcal{A}_{i+1}$ für einen stummen Zustand z, sind.
Die *Beam-Weite* $\mathcal{B}$ wird so bestimmt, dass weder zuviel Speicherplatz benötigt wird, noch zu viele, der Anfragesequenz zugrundeliegende Pfade durch den *Beam-Search*-Algorithmus ausgeschlossen werden.

4.3 Schätzung der Parameter eines jpHMMs

4.3.1 Emissionswahrscheinlichkeiten

Die Emissionswahrscheinlichkeiten eines jpHMMs schätzen wir, wie für ein Profil-HMM beschrieben, für DNA-Sequenzen mit einer Dirichlet-Verteilung (2.21) und für Proteinsequenzen mit einer Mischung von neun Dirichlet-Verteilungen (2.22):

Es sei $n = |\Sigma|$ die Anzahl der Symbole im Emissionsalphabet $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ (Kap. 1).

Definition 4.13 *Sei A_l, $l = 1, \ldots, K'$, das Subalignment der Sequenzen des Subtyps $\mathcal{S}_l$ in A. Sei $S_k \in \mathcal{S}$ die Sequenz der Sequenzfamilie $\mathcal{S}$, die man durch Streichen aller Lücken der k-ten Zeile in A erhält.*
Sei nun l fest gewählt. Dann definiert

$$\vec{n}_j := (n_{j,1}, \ldots, n_{j,n}), \quad \textit{mit } n_{j,i} := \#\{a_{k,j} \,|\, a_{k,j} = \sigma_i, S_k \in \mathcal{S}_l\}, \tag{4.13}$$

den Häufigkeitsvektor der in der Spalte j des Subalignments A_l beobachteten Symbole, *und* $|\vec{n}_j| := \sum_{i=1}^{n} n_{j,i}$ *die* Anzahl *aller Sequenzen des Subalignments A_l, die in der Spalte j ein Symbol in Σ, d.h. keine Lücke, enthalten.*

Definition 4.14

1.) *Sei $z \in Z \backslash \{I_B, I_E\}$, $\mathrm{typ}(z) \in \{M, I\}$. Sei $A(z)$ das Subalignment in A zu dessen Profil-HMM der Zustand z gehört.*
Sei $j := \mathrm{col}(z)$ die Spalte in $A(z)$ der der Zustand z zugeordnet ist und j' die nächste auf j folgende Konsensusspalte in $A(z)$.
Sei $\vec{n}_j$ der Häufigkeitsvektor der in der Spalte j des Subalignments $A(z)$ beobachteten Symbole. Dann repräsentiert der Vektor

$$\vec{n}_z := \begin{cases} \vec{n}_j & ,\textit{falls } \mathrm{typ}(z) = M, \\ \vec{n}_{j+1} + \ldots + \vec{n}_{j'-1} & ,\textit{falls } \mathrm{typ}(z) = I, \end{cases} \tag{4.14}$$

die dem Zustand z zugeordneten Häufigkeiten *der Symbole aus Σ in $A(z)$.*

2.) *Sei fc die erste und lc die letzte gemeinsame Konsensusspalte aller Subalignments $A_l, l = 1, \ldots, K'$, von A. Dann repräsentieren die Vektoren*

$$\vec{n}_{I_B} := (n_{I_B,1}, \ldots, n_{I_B,n}) \tag{4.15}$$

$$\textit{mit } n_{I_B,i} := \#\{a_{k,j} \mid a_{k,j} = \sigma_i,\ k = 1,\ldots,K,\ j = 1,\ldots,fc-1\},$$

und

$$\vec{n}_{I_E} := (n_{I_E,1},\ldots,n_{I_E,n}) \tag{4.16}$$
$$\textit{mit } n_{I_E,i} := \#\{a_{k,j} \mid a_{k,j} = \sigma_i,\ k = 1,\ldots,K,\ j = lc+1,\ldots,N\},$$

die dem Zustand I_B *bzw.* I_E zugeordneten Häufigkeiten *der Symbole aus* Σ *in* A.

Die den Insert-Zuständen I_B und I_E zugeordneten Häufigkeiten $\vec{n}_{I_B}$ bzw. $\vec{n}_{I_E}$ entsprechen also den Häufigkeiten der in allen Subalignments in den Spalten vor der ersten (fc) bzw. nach der letzten (lc) gemeinsamen Konsensusspalte beobachteten Symbole.

Emissionswahrscheinlichkeiten für DNA-Sequenzen

Da für Nukleotidsequenzen nicht solch komplexe Zusammenhänge wie für Aminosäuresequenzen berücksichtigt werden müssen, schätzen wir die Emissionswahrscheinlichkeiten der Match- und die Insert-Zustände eines jpHMMs für ein Alignment von DNA-Sequenzen jeweils mithilfe einer einzelnen Dirichlet-Verteilung. Dies entspricht wie wir in Kap. 2 gesehen haben der Pseudo-Count-Methode.
Sei $\Sigma = \{A, C, G, T\}$. Seien $\vec{\alpha}_M := (\alpha_{M,1},\ldots,\alpha_{M,4})$ die Parameter der Dirichlet-Dichte für einen Match-Zustand und $\vec{\alpha}_I := (\alpha_{I,1},\ldots,\alpha_{I,4})$ die Parameter für einen Insert-Zustand. Dann gilt also für die Emissionswahrscheinlichkeiten eines Match- bzw. Insert-Zustandes z:

$$e_{z,\sigma_i} = \frac{n_{z,i} + \alpha_{\mathrm{typ}(z),i}}{|\vec{n}_z| + |\vec{\alpha}_{\mathrm{typ}(z)}|}, \quad \sigma_i \in \Sigma. \tag{4.17}$$

Die Parameter $\vec{\alpha}_M$ bzw. $\vec{\alpha}_I$ schätzen wir unabhängig voneinander anhand des gegebenen MSAs A. Sei $N_M := \{\vec{n}_z \mid z \in Z, \mathrm{typ}(z) = M\}$ die Menge aller Häufigkeitsvektoren die den Match-Zuständen des jpHMMs zugeordnet sind, und $N_I := \{\vec{n}_z \mid z \in Z, \mathrm{typ}(z) = I\}$ die Menge aller den Insert-Zuständen zugeordneten Häufigkeitsvektoren. Dann schätzen wir die Parameter $\vec{\alpha}_M$ für die Match- bzw. $\vec{\alpha}_I$ für die Insert-Emissionswahrscheinlichkeiten jeweils mithilfe der ML-Methode (2.32). Dies ist nach (2.33) äquivalent dazu, die Parameter $\vec{\alpha}_M^*$ bzw. $\vec{\alpha}_I^*$ zu bestimmen, für die gilt :

$$\vec{\alpha}_M^* = \arg\min_{\vec{\alpha}_M} \Big(- \sum_{\vec{n}_z \in N_M} \log P_{\vec{\alpha}_M}\big(\vec{n}_z \,\big|\, |\vec{n}_z|\big)\Big) \tag{4.18}$$

bzw.

$$\vec{\alpha}_I^* = \arg\min_{\vec{\alpha}_I} \Big(- \sum_{\vec{n}_z \in N_I} \log P_{\vec{\alpha}_I}\big(\vec{n}_z \,\big|\, |\vec{n}_z|\big)\Big). \tag{4.19}$$

Die Parameter $\vec{\alpha}_M^*$ bzw. $\vec{\alpha}_I^*$ berechnen wir mit dem in [22] ausführlich dargestellten Gradientenabstiegsverfahren.

Emissionswahrscheinlichkeiten für Proteinsequenzen

Zur Schätzung der Emissionswahrscheinlichkeiten eines jpHMMs für ein Alignment von Proteinsequenzen nutzen wir, wie die Programme HMMER und SAM, die von [22] auf der BLOCKS-Datenbank geschätzte Mischung von neun Dirichlet-Verteilungen. Die Parameter dieser Dirichlet-Mischung sind ebenfalls in [22] gegeben.

4.3.2 Übergangswahrscheinlichkeiten

Übergangswahrscheinlichkeiten

Die Wahrscheinlichkeiten der Übergänge *innerhalb* der einzelnen Profil-HMMs $\mathcal{P}_i, i = 1, \ldots, K'$, eines jpHMMs schätzen wir, wie in Abschnitt 2.3.2 beschrieben, für die drei verschiedenen Arten von Übergängen (aus einem Match-, aus einem Insert- und aus einem Delete-Zustand heraus) mit jeweils einer Dirichlet-Verteilung als a-priori-Verteilung.
Zu jeder Sequenz eines Subtyps $\mathcal{S}_i, i = 1, \ldots, K'$, bzw. zu jeder Zeile des entsprechenden Subalignments A_i gibt es einen eindeutigen Pfad durch das Profil-HMM $\mathcal{P}_i$. Der einem Zustand $z \in Z \backslash \{I_B, D_B, I_E, D_E\}$ zugeordnete Häufigkeitsvektor entspricht also dem Vektor der für diesen Zustand beobachteten Übergangshäufigkeiten in $\mathcal{P}_i$ (2.34).
Als Parameter $\vec{\alpha}_M$, $\vec{\alpha}_I$ bzw. $\vec{\alpha}_D$ der Dichte der entsprechenden Dirichlet-Verteilung wählen wir die in (2.40) angegebenen, von [29] empirisch ermittelten Parameter.

Sprungwahrscheinlichkeiten

Die Sprungwahrscheinlichkeiten, d.h. die Wahrscheinlichkeiten der Übergänge *zwischen* den Profil-HMMs $\mathcal{P}_i, i = 1, \ldots, K'$, des jpHMMs, können, im Gegensatz zu den Übergangswahrscheinlichkeiten *innerhalb* der Profil-HMMs, nicht anhand von beobachteten Häufigkeiten geschätzt werden, da Sprünge nicht für ein MSA beobachtbar sind. Stattdessen nutzen wir für alle Sprünge eine feste, empirisch ermittelte Wahrscheinlichkeit, die wir die *Sprungwahrscheinlichkeit* P_{jump} nennen. In der Anwendung eines jpHMMs zur Identifizierung rekombinanter HIV- und HCV-Sequenzen (Kap. 6) nutzen wir eine Sprungwahrscheinlichkeit von $P_{\text{jump}} = 10^{-9}$. Diese wurde anhand der gegebenen Trainingssequenzen und multiplen Sequenzalignments empirisch ermittelt (Abschnitt 6.4).
Die Sprungwahrscheinlichkeit ist die Wahrscheinlichkeit, in einem Zustand $z \in Z$ eines Profil-HMMs $\mathcal{P}_i$ einen Sprung zu machen. Sind Sprünge in verschiedene Profil-HMMs $\mathcal{P}_j, j \neq i$, möglich, wird die Sprungwahrscheinlichkeit auf all diese Übergänge gleich verteilt. D.h. die Wahrscheinlichkeit eines Sprungs von einem Zustand $z_k \in Z_i$

eines Profil-HMMs $\mathcal{P}_i$ in einen beliebigen Zustand des Profil-HMMs $\mathcal{P}_j$, $i \neq j$, ist

$$\frac{P_{\text{jump}}}{\#\{\mathcal{P}_x, x \neq i \,|\, \exists z_n \in Z_x, \text{ s. d. ein Übergang von } z_k \text{ nach } z_n \text{ möglich ist}\}}. \quad (4.20)$$

Diese Sprungwahrscheinlichkeit von z_k in irgendeinen Zustand des Profil-HMMs $\mathcal{P}_j$ wird für einen Sprung aus einem Match-Zustand nochmals unterteilt in die Wahrscheinlichkeit für einen Sprung in einen Match-Zustand und in die Wahrscheinlichkeit für einen Sprung in einen Delete-Zustand. Das Verhältnis dieser beiden Wahrscheinlichkeiten zueinander entspricht dem Verhältnis der entsprechenden Übergangswahrscheinlichkeiten *innerhalb* des Profil-HMMs $\mathcal{P}_j$, d.h dem Verhältnis der Übergangswahrscheinlichkeiten vom vorhergehenden Match-Zustand in $\mathcal{P}_j$ in diese beiden Zustände.
Damit für alle Zustände die Summe der Übergangswahrscheinlichkeiten gleich 1 ist, berechnen wir für einen Zustand z_k eines Profil-HMMs $\mathcal{P}_i$ zuerst alle Wahrscheinlichkeiten der Übergänge in Zustände desselben Profil-HMMs $\mathcal{P}_i$, diese nennen wir

$$t^*_{z_k,z_l} \quad (4.21)$$

und multiplizieren diese jeweils mit $(1 - P_{\text{jump}})$, falls vom Zustand z_k Sprünge in ein anderes Profil-HMM möglich sind. D.h. für die Wahrscheinlichkeit eines Übergangs von einem Zustand $z_k \in Z_i$ eines Profil-HMMs $\mathcal{P}_i$ in einen Zustand $z_l \in Z_i$ von $\mathcal{P}_i$ gilt dann

$$t_{z_k,z_l} = t^*_{z_k,z_l} \cdot (1 - P_{\text{jump}}). \quad (4.22)$$

Übergangswahrscheinlichkeiten von B, I_B, D_B, I_E und D_E

Für die Übergangswahrscheinlichkeiten der Zustände B, I_B, D_B, I_E und D_E bzw. für die Übergangswahrscheinlichkeiten in diese Zustände führen wir drei Konstanten $P_{\text{Insert}}, P_{\text{D,init}}, P_{\text{D,ext}} \in (0,1)$, $P_{\text{D,init}} < P_{\text{Insert}}$, ein. Diese sind in Anlehnung an das Einfügen (engl. *insert*ion) eines Symbols (P_{Insert}), der Einführung (engl. *init*iation) einer Lücke ($P_{\text{D,init}}$) und der Verlängerung (engl. *ext*ension) einer Lücke ($P_{\text{D,ext}}$) in der Anfragesequenz benannt. Ihre Bedeutung wird im Folgenden erläutert.
P_{Insert} ist die Wahrscheinlichkeit, eine Insertion in der Anfragesequenz um eine Position zu verlängern:
Für die Übergangswahrscheinlichkeit von I_B nach I_B bzw. von I_E nach I_E gilt

$$t_{I_B,I_B} = P_{\text{Insert}} \quad \text{und} \quad t_{I_E,I_E} = P_{\text{Insert}}.$$

Daraus folgt

$$t_{I_E,E} = 1 - P_{\text{Insert}}.$$

Die Wahrscheinlichkeit eines Übergangs von I_B in einen Match-Zustand, der der ersten Konsensusspalte eines Subalignments A_i in A zugeordnet ist, ist für all diese

Match-Zustände gleich. D.h. für alle Zustände $z_k \in Z$, $\text{typ}(z_k) = M$, die der ersten Konsensusspalte eines Subalignments A_i in A zugeordnet sind, gilt

$$t_{I_B,z_k} = \frac{(1 - P_{\text{Insert}})}{K'}.$$

$P_{\text{D,init}}$ bestimmt die Wahrscheinlichkeit des Einführens einer Lücke am Anfang und am Ende der Anfragesequenz: Für die Übergangswahrscheinlichkeiten des Start-Zustandes B gilt

$$\begin{aligned} t_{B,D_B} &= P_{\text{D,init}}, \\ t_{B,I_B} &= P_{\text{Insert}} - P_{\text{D,init}} \\ \text{und} \quad t_{B,z_k} &= \frac{1 - P_{\text{Insert}}}{2 \cdot K'} \end{aligned}$$

für alle Zustände $z_k \in Z$, $\text{typ}(z_k) \in \{M, D\}$, die der ersten Konsensusspalte eines Subalignments A_i in A zugeordnet sind. Daraus folgt $\sum_{z_k \in Z} t_{B,z_k} = 1$.
$P_{\text{D,ext}}$ bestimmt die Wahrscheinlichkeit des Fortführens bzw. des Beendens einer Lücke in die Anfragesequenz: Ein Pfad von B über D_B in einen Match-Zustand z_k, der der Spalte $\text{col}(z_k)$ in A zugeordnet ist, entspricht dem Einfügen von $\text{col}(z_k) - 1$ Lücken in der Anfragesequenz. Da durch den Übergang von B zu D_B bereits eine Lücke in der Anfragesequenz eingefügt wird, entspricht also ein Sprung von D_B in einen Match-Zustand z_k, der der Spalte $\text{col}(z_k)$ in A zugeordnet ist, dem Einfügen von $\text{col}(z_k) - 2$ Lücken, und dem Beenden dieser Lücke in der Anfragesequenz. Für alle Match-Zustände $z_k \in Z$, $\text{typ}(z_k) = M$, die nicht der ersten Konsensusspalte eines Subalignments in A zugeordnet sind, gilt also

$$t_{D_B,z_k} = \frac{(P_{\text{D,ext}}) \cdot (P_{\text{D,ext}})^{\text{col}(z_k)-2}}{K'} = \frac{(P_{\text{D,ext}})^{\text{col}(z_k)-1}}{K'}.$$

Um die Forderung $\sum_{z_k \in Z} t_{D_B,z_k} = 1$ zu erfüllen, normieren wir die berechneten Wahrscheinlichkeiten.
Der Übergang von einem Match-Zustand z_k in den Delete-Zustand D_E entspricht dem Einfügen von $N - \text{col}(z_k)$ Lücken in der Anfragesequenz. D.h. er entspricht dem Einführen und dem $(N - \text{col}(z_k) - 1)$-maligen Fortführen einer Lücke in der Anfragesequenz. Für alle Match-Zustände $z_k \in Z$, $\text{typ}(z_k) = M$, die *nicht* der letzten Konsensusspalte eines Subalignments in A zugeordnet sind, gilt also

$$t_{z_k,D_E} = P_{\text{D,init}} \cdot (P_{\text{D,ext}})^{N-\text{col}(z_k)-1}.$$

Sei $z_k \in Z_i \subset Z$. Sind im Subalignment A_i nicht alle Spalten vor bzw. nach der Spalte $\text{col}(z_k)$ Konsensusspalten, müssen natürlich weniger als $\text{col}(z_k) - 1$ bzw. $N - \text{col}(z_k) - 1$ Konsensusspalten in A übersprungen werden Der Übersicht halber schreiben wir hier dennoch $\text{col}(z_k) - 1$ bzw. $N - \text{col}(z_k) - 1$ und meinen damit die Anzahl der Konsensusspalten in A_i vor bzw. nach der Spalte der der Zustand z_k

zugeordnet ist.
Die Übergangswahrscheinlichkeit von einem Match-Zustand z_k, der der letzten Konsensusspalte eines beliebigen Subalignments A_i in A zugeordnet ist, in den Insert-Zustand I_E ist gegeben durch

$$t_{z_k,I_E} = P_{\text{Insert}}.$$

Daraus folgt

$$t_{z_k,E} = 1 - P_{\text{Insert}}.$$

Für alle Delete-Zustände z_k die der letzten Konsensusspalte eines Subalignments A_i in A zugeordnet sind, gilt

$$t_{z_k,E} = 1,$$

da diese Zustände keine weiteren Nachfolger haben.

Die Wahrscheinlichkeiten P_{Insert}, $P_{\text{D,init}}$ und $P_{\text{D,ext}}$ werden empirisch anhand der Trainingsdaten (Abschnitt 6.2) ermittelt (Abschnitt 6.4).

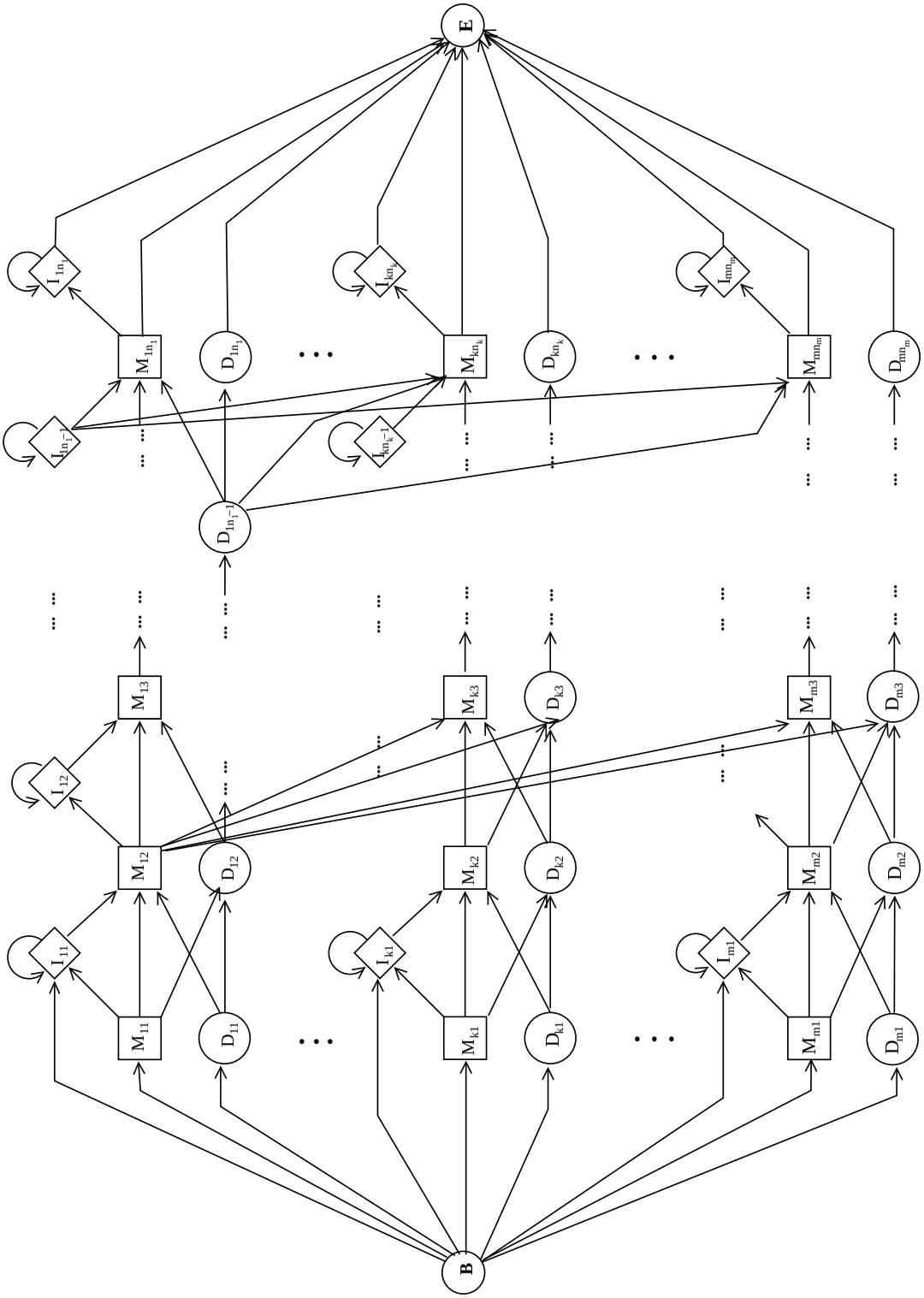


Abbildung 4.2: Architektur eines springenden Profil-Hidden-Markov-Modells

Kapitel 5

Implementation

Zur Implementation des jpHMM-Programms wurde die Programmiersprache C++ [26] verwendet. Kompiliert wurde das Programm unter Linux mit dem Compiler g++ (gcc, Version 3.3.5).

5.1 Eingabe

Als Eingabe verlangt das Progamm eine Datei, die ein multiples Sequenzalignment, bestehend aus mehreren (Sub-)Subtypen, enthält, und eine Datei mit einer oder mehreren Anfragesequenzen. Diese müssen in einem bestimmten, zum FASTA-Format [15] ähnlichen, Format vorliegen: Der Name einer Sequenz ist durch das Zeichen '>' vor dem entsprechenden Namen gekennzeichnet, die Sequenz selbst steht in der darauffolgenden Zeile. Der Unterschied zum FASTA-Format ist dabei, dass innerhalb jeder Sequenz keine Zeilenumbrüche erlaubt sind. In der Eingabedatei des multiplen Sequenzalignments müssen zusätzlich alle Sequenzen entsprechend der Subtypen sortiert sein. Jeder Subtyp wird durch '>>', gefolgt durch den Namen des Subtyps, gekennzeichnet. Die ihm zugeordneten Sequenzen finden sich in den darauffolgenden Zeilen, mit Name und Sequenz im oben beschriebenen Format. Kommentare innerhalb der Dateien sind nicht erlaubt.
Zusätzlich dazu wird eine Datei mit den zur Schätzung der Emissionswahrscheinlichkeiten (Abschnitt 4.3.1) benötigten Parametern verlangt, da diese für jedes gegebene Alignment veschieden sind. Diese Parameter schätzten wir anhand eines Trainingsprogramms, das als Eingabe ein multiples Sequenzalignment und als Ausgabe die Parameter hat. Die Datei, die diese Parameter enthält, wurde manuell erstellt.

Als optionale Parameter können dem Programm die Sprungwahrscheinlichkeit P_{jump}, die *Beam-Weite* $\mathcal{B}$ für den *Beam-Search*- bzw. den Viterbi-Algorithmus, und die Konsensusspaltenkriterien c und t übergeben werden.

Aus dem multiplen Sequenzalignment erzeugt das Programm ein jpHMM. Die Parameter zur Schätzung der Übergangswahrscheinlichkeiten des jpHMMs sind fest gewählt und im Programm-Code enthalten. Für jede Anfragesequenz wird der Viterbi-Pfad durch das jpHMM und die Rekombination aus den im Alignment gegebenen (Sub-)Subtypen bestimmt.

5.2 Ausgabe

Die Ausgabe des Programms ist eine Datei, die die Rekombination einer oder mehrerer Anfragesequenzen aus den im multiplen Sequenzalignment gegebenen (Sub-)Subtypen enthält. Diese Ausgabedatei liegt in zwei verschiedenen Formaten vor:
- als Auflistung aller in der Anfragesequenz gefundenen (Sub-)Subtypen mit Angabe einiger Parameter des jpHMMs; als Beispiel ist die Ausgabe einer Beispielsequenz '*example*' angegeben:

```
# Output of the jpHMM.
#
#   format:
#   sequence_name    chosen_parameters   predicted_subtypes
#
#   (bw=beam-width, jp=jump probability)
#
# predicted recombination of the following sequence(s):

example bw=1e-20, jp=1e-09  01_AEA1G
```

- als Ausgabe der Rekombination mit den entsprechenden Bruchstellen in der Sequenz; als Beispiel ist ebenfalls die Ausgabe der Beispielsequenz '*example*' angegeben:

```
# Output of the jpHMM.
#
#   chosen parameters:
#   jump probability:   1e-09
#   beam-width:         1e-20
#
#   format:
#   sequence_name
#   start_position   end_position    predicted_subtype
#
# predicted recombination with breakpoints of the following sequence(s):

example
1   1392    A1
```

```
1393    2408    G
2409    4800    A1
4801    5272    01_AE
5273    5395    G
5396    8425    A1
```

Zusätzlich zu diesen beiden Ausgabe-Dateien wird eine Datei mit der Rekombination der Anfragesequenz im General-Feature-Format (GFF) (`http://www.sanger.ac.uk/Software/formats/GFF`) erstellt. Als Beispiel ist die Ausgabe der Sequenz '*example*' angegeben:

```
# Output of the jpHMM.
#
# predicted recombination of sequence 1 ( name = example, length = 8425 )
#
# predicted subtypes for sequence 'example':
example jpHMM   A1  1   1392    .   +   .
example jpHMM   G   1393    2408    .   +   .
example jpHMM   A1  2409    4800    .   +   .
example jpHMM   01_AE   4801    5272    .   +   .
example jpHMM   G   5273    5395    .   +   .
example jpHMM   A1  5396    8425    .   +   .
```

Zeilen die mit dem Zeichen '#' beginnen, kennzeichnen Kommentarzeilen. Alle weiteren Zeilen geben die vorhergesagte Rekombination der Anfragesequenz aus den (Sub-)Subtypen der gegebenen Sequenzfamilie mit den entsprechenden Bruchstellen wieder. Jede Zeile entspricht einem (Sub-)Subtyp. Der erste Eintrag einer Zeile enthält den Namen der Sequenz (*example*), der zweite Eintrag den Namen des Programms (*jpHMM*), das die Vorhersage für die Rekombination der Sequenz erstellt hat, und der dritte Eintrag den vorhergesagten (Sub-)Subtyp. Die Anfangs- und End-Position dieses (Sub-)Subtypen in der Sequenz ist im vierten bzw. fünften Eintrag angegeben. Das Zeichen '+' im siebenten Eintrag steht für den Vorwärtsstrang der Anfragesequenz.
Die vorhergesagte Rekombination einer Anfragesequenz kann mithilfe des Programms *gff2ps* [1] visualisiert werden. Als Beispiel ist in Abb. 5.1 die Visualisierung der Rekombination der Sequenz '*example*' gegeben.

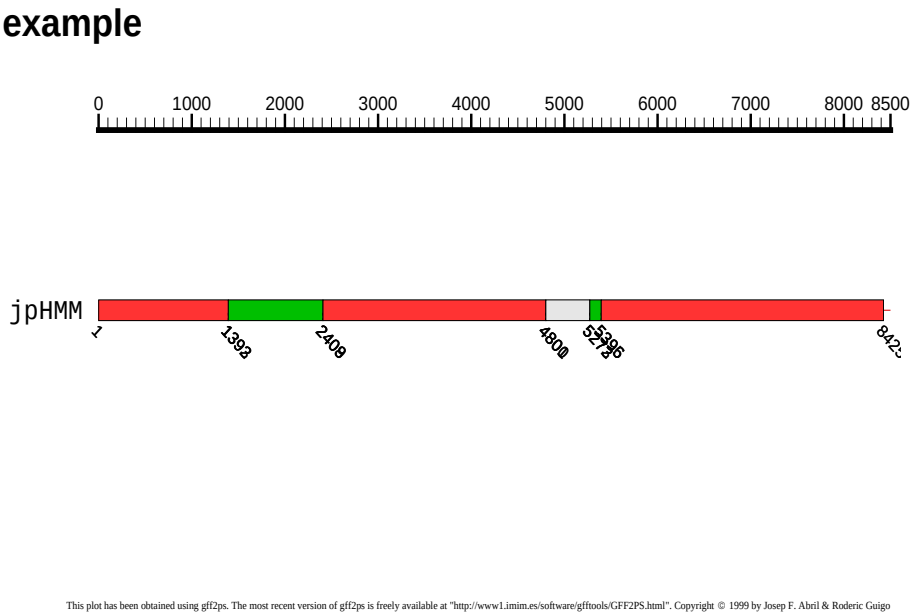


Abbildung 5.1: gff2ps-Visualisierung der Rekombination einer Beispielsequenz

5.3 Laufzeit und Speicherbedarf

Den Hauptteil des benötigten Speicherbedarfs und der Laufzeit des jpHMMs nimmt der Viterbi-Algorithmus ein. Sei eine Anfragesequenz der Länge l gegeben. Sei $\mathcal{S} = \{S_1, \ldots, S_K\}$ eine Sequenzfamilie und $K' \leq K$ die Anzahl aller Subtypen in $\mathcal{S}$. Sei m die Anzahl aller Zustände im Zustandsraum Z des jpHMMs. Nehmen wir an, dass der Viterbi-Algorithmus (Abschnitt 4.2) ohne Integration des *Beam-Search*-Algorithmus durchgeführt wird, dann benötigen wir zur Speicherung aller zu berechnenden Viterbi-Variablen eine Matrix der Grösse $l \times m$. Als Beispiel betrachten wir das von uns zur Auswertung der Genauigkeit der Vorhersage eines jpHMMs (Kap. 6) verwendete multiple Sequenzalignment des HIV-Typs 1. Dieses Alignment besteht aus 14 (Sub-)Subtypen, deren Sequenzen jeweils eine Länge von ca. $10000nt$ haben. Nehmen wir an, dass jeder Subtyp jeweils 10000 Konsensusspalten enthält, dann beträgt die Anzahl der dem Alignment zugeordneten Zustände $m = 3 \cdot 14 \cdot 10000 = 4.2 \cdot 10^5$, da jeder Konsensusspalte 3 Zustände zugeordnet werden. Für eine Anfragesequenz der Länge $l = 10000$ müssen also $l \cdot m = 10000 \cdot 4.2 \cdot 10^5 = 4.2 \cdot 10^9$ Viterbi-Variablen gespeichert werden. Da wir zur Speicherung der berechneten Wahrscheinlichkeiten im Viterbi-Algorithmus Gleitkommazahlen einfacher Genauigkeit (*float*, 4 Byte) in C++ verwenden, benötigt also allein der Viterbi-Algorithmus des jpHMMs einen Speicherplatz von $4 \cdot 4.2 \cdot 10^9 = 15.7$ GB. Durch die Integration des *Beam-Search*-Algorithmus in

den Viterbi-ALgorithmus lässt sich dieser Speicherplatz jedoch enorm einschränken. Für eine *Beam-Weite* $\mathcal{B} = 10^{-20}$ lag die für jede Anfragesequenz berechnete durchschnittliche Anzahl aktiver Zustände pro Position für die Sequenzen des Trainingsdatensatzes zwischen 1620 (für eine CRF des Typs CRF12 und einer Länge von $8760nt$) und 2862 (für eine CRF des Typs CRF11 der Länge $9768nt$). Für eine Anfragesequenz der Länge $10000nt$ mit durchschnittlich 2862 aktiven Zuständen pro Position, erhalten wir somit einen benötigten Speicherplatz von $4 \cdot 10000 \cdot 2862 = 109.2$ MB, was eine deutliche Verringerung gegenüber 15.7 GB darstellt. In Abb. 5.2 ist ein Beispiel für die Reduzierung der aktiven Zustände mit einer *Beam-Weite* $\mathcal{B} = 10^{-20}$ für eine HIV-Sequenz (CRF02) der Länge 8428 angegeben. Für jede Position der Anfragesequenz sind zwei Spalten des Alignments eingezeichnet: die minimale und die maximale Spalte, denen für diese Position mindestens ein aktiver Zustand zugeordnet ist. D.h. allen Spalten im Alignment die *ausserhalb* des von diesen beiden Spalten eingeschlossenen Bereichs liegen, ist kein aktiver Zustand für diese Position der Anfragesequenz zugeordnet. Den Spalten im Alignment, die *zwischen* diesen beiden Spalten liegen, können für diese Position aktive Zustände zugeordnet sein, dies muss aber nicht der Fall sein. Die durchschnittliche Anzahl der aktiven Zustände für diese Anfragesequenz beträgt 1690.
Die Laufzeit des Viterbi-Algorithmus setzt sich zusammen aus der Berechnung aller benötigten Viterbi-Variablen und des Zurückverfolgens des Viterbi-Pfades. Für die Berechnung der Viterbi-Variablen $\delta_{z,i}$ einer Position i der Anfragesequenz und eines Zustandes z muss eine Liste aller Vorgänger von z durchlaufen werden, und die Emissionswahrscheinlichkeit des Symbols an Position i der Sequenz im Zustand z berechnet werden, falls z kein stummer Zustand ist. Die Emissionswahrscheinlichkeit wird in konstanter Zeit berechnet, so dass die Laufzeit zur Berechnung einer Viterbi-Variablen $\delta_{z,i}$ proportional ist zur Anzahl aller Vorgänger von z. Die Anzahl aller Vorgänger eines Zustandes $z \in Z \backslash D_E$ beträgt höchstens $3K' + 1$ (Bem. 4.10), und für den Delete-Zustand D_E ca. $m/3$, da D_E als Vorgänger alle Match-Zustände des Zustandsraumes Z hat. Diese Anzahl kann sich jedoch aufgrund der Anwendung des *Beam-Search*-Algorithmus ebenfalls noch verringern.
Die Laufzeit für das Zurückverfolgen des Viterbi-Pfades kann vernachlässigt werden, da wir bereits in der Berechnung der Viterbi-Variablen für jede Position i für jeden Zustand z den besten Vorgänger speichern. Sei m_d die durchschnittliche Anzahl aktiver Zustände der Anfragesequenz. Dann hat die für den Viterbi-Algorithmus benötigte Laufzeit eine Komplexität von $O(m_d K' l)$.
Für die beiden oben genannten CRFs (CRF12 der Länge 8760, CRF11 der Länge 9768) beträgt die CPU-Laufzeit des jpHMM-Programms, unter Verwendung des *Beam-Search*-Algorithmus mit der *Beam-Weite* $\mathcal{B} = 10^{-20}$, auf einem Linux-PC mit 3 MB RAM und 3.2 GHz 434 bzw. 819 Sekunden. Dies schliesst sowohl die Bildung des jpHMMs als auch die Suche des Viterbi-Pfades der Anfragesequenz ein, wobei die CPU-Laufzeit für die Bildung des jpHMMs für das zugrundeliegende MSA von 14 HIV-(Sub-)Subtypen (Kap. 6) nur 12 Sekunden beträgt.

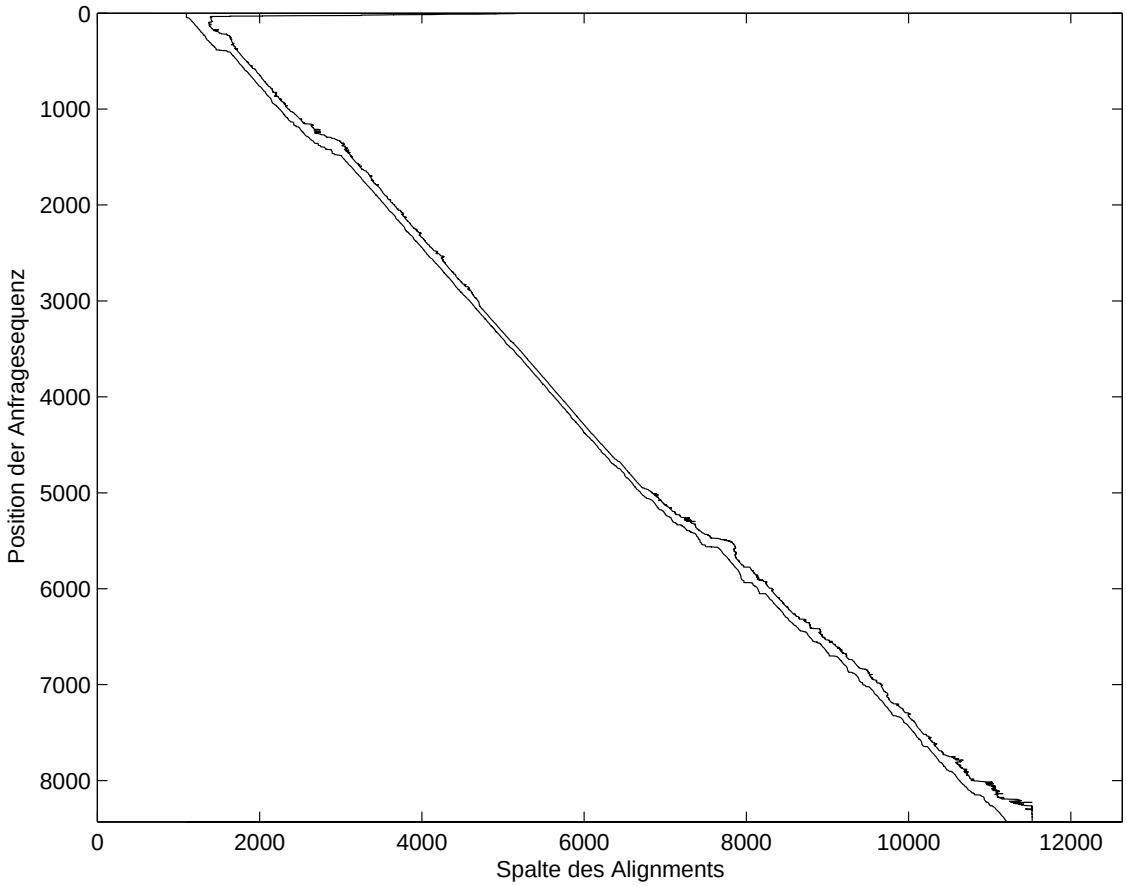


Abbildung 5.2: Reduzierung der Anzahl aktiver Zustände im Viterbi-Algorithmus durch Integration des *Beam-Search*-Algorithmus mit der *Beam-Weite* $\mathcal{B} = 10^{-20}$ für eine Beispielsequenz von HIV. Das hier betrachtete Alignment bestand aus 14 (Sub-)Subtypen. Für jede Position der Anfragesequenz sind zwei Spalten des Alignments eingezeichnet: die minimale und die maximale Spalte, denen für diese Position mindestens ein aktiver Zustand zugeordnet ist. D.h. allen Spalten im Alignment die *ausserhalb* des von diesen beiden Spalten eingeschlossenen Bereichs liegen, ist kein aktiver Zustand für diese Position der Anfragesequenz zugeordnet. Einer Spalte im Alignment, die *zwischen* diesen beiden Spalten liegt, können, aber müssen nicht notwendigerweise, für diese Position aktive Zustände zugeordnet sein. Jede Spalte entspricht dabei $3 \cdot 14 = 42$ Zuständen, da jeder Spalte eines Subtyps 3 Zustände zugeordnet werden. In der Viterbi-Rekursion muss für jede Position der Anfragesequenz, statt aller Zustände des jpHMMs, nur ein Teil der Zustände durchlaufen werden, die einer Spalte zwischen den beiden eingezeichneten Spalten im Alignment zugeordnet sind.

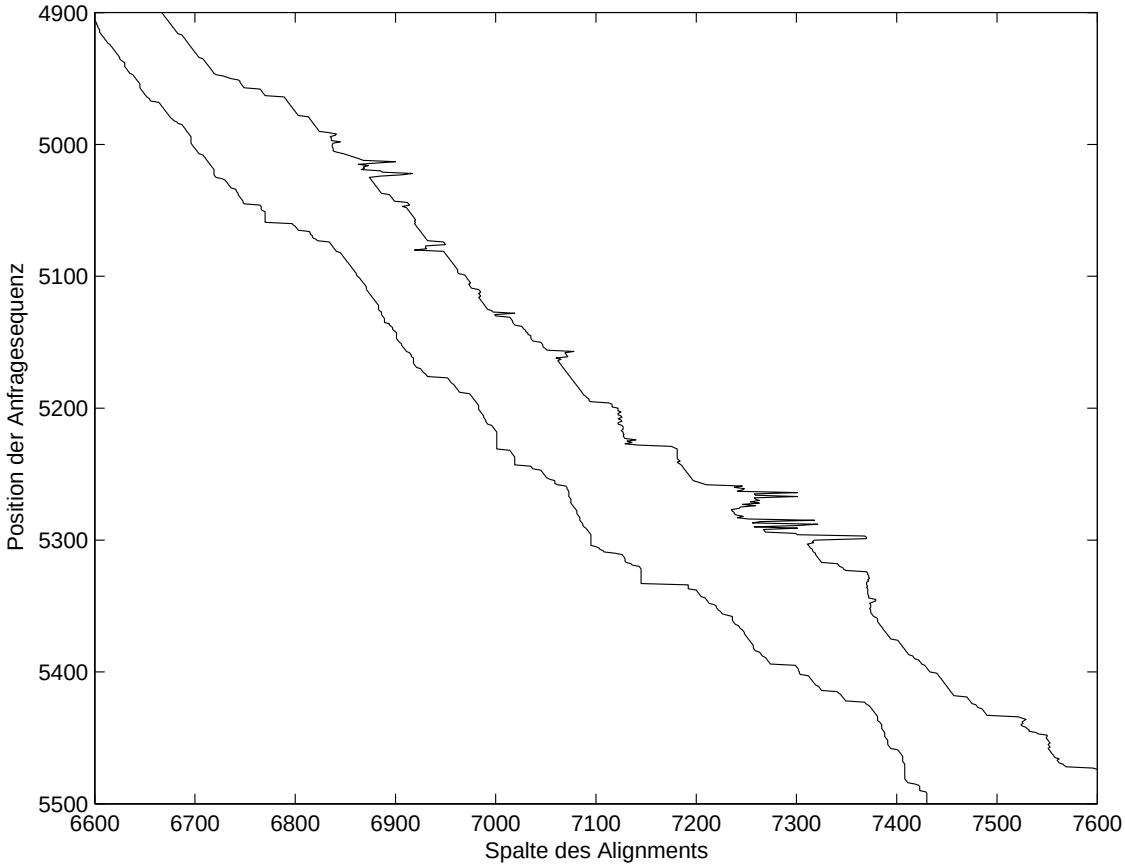


Abbildung 5.3: Ein vergrösserter Ausschnitt der Abbildung 5.2 zur Reduzierung der Anzahl aktiver Zustände im Viterbi-Algorithmus durch den *Beam-Search*-Algorithmus

Kapitel 6

Ergebnisse

Das von uns entwickelte Programm eines jpHMMs wurde für DNA-Sequenzen von HIV und HCV getestet. Für beide Virustypen lagen jeweils ein MSA mehrerer Subtypen und ein (für HCV) bzw. mehrere (für HIV) Datensätze von Anfragesequenzen vor, deren Rekombination aus den im MSA gegebenen Subtypen bestimmt werden sollte.
Anhand des entsprechenden MSA schätzten wir die Parameter $\vec{\alpha}$ zur Schätzung der Emissionswahrscheinlichkeiten der Match- und Insert-Zustände des jpHMMs (Abschnitt 4.3.1) für HIV und HCV unabhängig voneinander jeweils mithilfe der Maximum-Likelihood-Methode (2.32) und des Gradientenabstiegsverfahrens, siehe Abschnitt 2.3.1 und [22].
Die Sprungwahrscheinlichkeit P_{jump} (Abschnitt 4.3.2) und die Parameter der Übergangswahrscheinlichkeiten von bzw. zu den Zuständen B, I_B, D_B, I_E und D_E (Abschnitt 4.3.2) ermittelten wir empirisch anhand der Anfragesequenzen des Trainingsdatensatzes (Abschnitt 6.2). Da uns nur zwei rekombinante Sequenzen von HCV vorlagen, nutzten wir dazu einen Datensatz von HIV-Sequenzen. Die für diesen Datensatz ermittelten Parameter nutzten wir sowohl zur Vorhersage der Rekombination der vorliegenden HIV-Testsequenzen mit dem jpHMM als auch für die der HCV-Testsequenzen. Die Auswertung dieser Vorhersagen ist in Abschnitt 6.5 beschrieben.

6.1 Multiple Sequenzalignments

Das MSA der HIV-Sequenzen bestand aus 14 Subtypen, Sub-Subtypen bzw. CRFs A1, A2, B - D, G, H, J, K, O, CPZ und 01_AE. O entspricht der Gruppe O des HIV-Typs 1, CPZ ist ein in Schimpansen gefundenes Virus [21], und 01_AE eine CRF. Alle anderen sind Subtypen bzw. Sub-Subtypen der Gruppe M. Insgesamt enthielt das Alignment 320 Sequenzen, die aligniert eine Länge von $12632nt$ hatten. Für HCV

lag uns ein MSA von 6 Subtypen vor, die wiederum in insgesamt 18 verschiedene Sub-Subtypen 1a-c, 2a-c,k, 3a,b,k, 4a, 5a und 6a,b,d,g,h,k unterteilt waren. Die Anzahl der Sequenzen in diesem MSA betrug 144, die aligniert jeweils eine Länge von $9820nt$ hatten. In beiden Alignments variierte die Anzahl der Sequenzen der vorliegenden (Sub-)Subtypen sehr stark. Beispielsweise lagen für den Subtyp C von HIV 109 Sequenzen vor, während für die Subtypen J und K jeweils nur 2 Sequenzen betrachtet wurden. 10 der 18 HCV-Sub-Subtypen enthielten sogar jeweils nur eine Sequenz. Von diesen Subtypen wurden bisher jedoch keine weiteren Sequenzen identifiziert.
Als Konsensusspaltenkriterium für das jeweilige MSA wählten wir $c = 0.5$ und $t = 5$. D.h. eine Spalte eines Subalignments des MSA ist nur dann eine Konsensusspalte, wenn mindestens die Hälfte aller Zeilen oder mindestens 5 Zeilen des Subalignments an dieser Position ein Symbol enthalten.

6.2 Trainingsdaten

Als Trainingssequenzen nutzten wir einen Datensatz von annotierten HIV-Sequenzen 12 verschiedener CRFs (CRF02 - CRF08 und CRF10 - CRF14). Die Rekombination dieser CRFs und die entsprechenden Bruchstellen in den Sequenzen sind in der aktuellen Literatur veröffentlicht. Eine Übersicht der Strukturen der CRFs findet man auf der Homepage des Los Alamos National Laboratory (LANL) unter
`http://hiv-web.lanl.gov/content/hiv-db/CRFs/CRFs.html`.

6.3 Testdaten

HIV

Für HIV betrachteten wir drei verschiedene Testdatensätze, **ACRFs**, **500nts** und **7000nts**.
ACRFs ist ein Datensatz künstlich erzeugter Sequenzen mit bekannten Bruchstellen. Die Sequenzen des Datensatzes wurden so erzeugt, dass sie aus jeweils zwei verschiedenen Subtypen bzw. zwei verschiedenen Sub-Subtypen des gleichen Subtyps zusammengesetzt sind, und an jeder x-ten ($x \in \{500, 1000, 1500\}$) Position eine Bruchstelle haben. Rekombinante Sequenzen aus zwei Subtypen bezeichen wir als *Inter-Subtyp-rekombinante* Sequenzen, Sequenzen die aus zwei verschiedenen Sub-Subtypen des gleichen Subtyps zusammengesetzt sind als *Inter-Sub-Subtyp-rekombinante* Sequenzen.
Für die *Inter-Subtyp-rekombinanten* Sequenzen verwendeten wir folgende Subtyp-Rekombinationen (in Klammern ist jeweils die in GenBank [3] verwendete Bezeichnung der für den entsprechenden Subtyp verwendeten Sequenz angegeben): A1(AF193275) und C(AY463217), A1 (AF193275) und D (AF133821),

A1 (AF193275) und G (AF450098), B (AF042101) und C (AY463217),
B (AF042101) und F1 (AY173958), B (AF042101) und 01_AE (AB032741).
Die *Inter-Sub-Subtyp-rekombinanten* Sequenzen setzen sich jeweils zusammen aus den Sub-Subtypen
A1 (AF413987) und A2 (AF286240), A2 (AF286241) und A1 (AF539405),
B (AF538302) und D (AJ320484), D (AJ488926) und B (AY352275),
F1 (AY173957) und F2 (AF377956), F2 (AY371158) und F1 (AY173958).
Die Unterscheidung von z.B. F1 und F2 und F2 und F1 wird gemacht, da in beiden Fällen unterschiedliche Sequenzen der Sub-Subtypen betrachtet werden und die ersten x Nukleotide der künstlich rekombinierten Sequenz jeweils dem zuerst genannten Subtyp angehören.
500nts ist ein Datensatz von 1108 Fragmenten existierender Subtypen und CRFs mit einer Länge von ca. $500nt$.
7000nts ist ein Datensatz von 311 vollständigen Genomen existierender Subtypen und CRFs die jeweils eine Länge von ca. $7000nt$ haben.

HCV

Für HCV stand uns ein Testdatensatz von zwei rekombinanten Sequenzen (RF) zur Verfügung. Die St.Petersburg-RF des Typs 1b/2k [10] und eine künstlich erzeugte RF des Typs 1a/2a. Beide Sequenzen enthalten jeweils nur eine Bruchstelle.

6.4 Schätzung der Parameter des jpHMMs

Zur Schätzung der Parameter des jpHMMs anhand des Trainingsdatensatzes nutzten wir für den *Beam-Search-* bzw. den Viterbi-Algorithmus eine *Beam-Weite* von $\mathcal{B} = 10^{-20}$, da dadurch weder zuviel Speicherplatz verbraucht und eine zu lange Rechenzeit benötigt wird, noch zu viele der jeweiligen Anfragesequenz zugrundeliegende Pfade ausgeschlossen werden. Für eine grössere *Beam-Weite* ist zwar die Rechenzeit kürzer, da die Anzahl der aktiven Zustände für eine Position einer Anfragesequenz stärker eingeschränkt wird, jedoch ergaben die für die Trainingsdaten vorhergesagten Rekombinationen eine schlechtere Übereinstimmung mit den entsprechenden veröffentlichten Daten. Mit kleineren Werten als $\mathcal{B} = 10^{-20}$ vergrössert sich die benötigte Laufzeit des Programms immens. Für Werte kleiner als 10^{-35} wird sogar für einige Sequenzen die Speicherkapazität des von uns genutzten Linux-PCs überschritten. Ein weiterer Punkt der bei der Wahl der *Beam-Weite* beachtet wurde, ist die Anzahl möglicher Deletionen an einer bestimmten Position in der Anfragesequenz. Nehmen wir beispielsweise an, dass die Übergangswahrscheinlichkeit von einem Delete-Zustand in einen Delete-Zustand desselben Profil-HMMs des jpHMMs gleich 0.5 ist, dann gilt für die Anzahl x der für eine bestimmte Position i

der Sequenz aktiven, im selben Profil-HMM aufeinanderfolgenden Delete-Zustände $0.5^x \leq 10^{-20}$. Durch die *Beam-Weite* $\mathcal{B} = 10^{-20}$ ist also in diesem Fall noch die Möglichkeit gegeben, für die Position i der Sequenz $x = 66$ Delete-Zustände desselben Profil-HMMs zu durchlaufen, d.h. 66 aufeinanderfolgende Konsensusspalten des entsprechenden Subtyps zu löschen.
Die Parameter $P_{\text{Insert}}, P_{\text{D,init}}$ und $P_{\text{D,ext}}$ des jpHMMs wurden so gewählt, dass im Jumping Alignment einer Anfragesequenz zum MSA, das durch den Viterbi-Pfad der Sequenz bestimmt wird, nicht nur ein sehr kurzer Bereich der Sequenz direkt zu den Spalten des Alignments aligniert wird, und das Alignment ansonsten aus beispielsweise einem sehr grossen Deletionsbereich am Anfang und einem sehr grossen Insertionsbereich am Ende besteht. Dies ist z.B. der Fall, wenn der Viterbi-Pfad der Sequenz sich zuerst im Delete-Zustand D_B befindet, dann einen Sprung zu einem Match-Zustand macht, der einer sehr hohen Spalte im MSA zugeordnet ist, einige weitere Match-Zustände durchläuft, denen sich eine lange Folge des Insert-Zustandes I_E anschliesst. D.h. die Parameter wurden so gewählt, dass die Zustände I_B, D_B, I_E und D_E nur für ein lokales Alignment der Anfragesequenz zum MSA angenommen werden. Parameter, die dies für alle Trainingsdaten erfüllen und die wir verwendet haben, sind:

$$P_{\text{Insert}} = 0.99, \quad P_{\text{D,init}} = 0.01 \quad \text{und} \quad P_{\text{D,ext}} = 0.99. \tag{6.1}$$

Die Parameter $\vec{\alpha}$ der Dirichlet-Verteilung zur Schätzung der Emissionswahrscheinlichkeiten wurden auf dem für HIV bzw. HCV gegebenen MSA mit dem Gradientenabstiegsverfahren (Abschnitt 2.3.1) geschätzt. Sei α_{M,σ_i} bzw. α_{I,σ_i}, $\sigma_i \in \Sigma$, $\Sigma = \{A, C, G, T\}$, der Parameter der Dirichlet-Verteilung zur Schätzung der Emissionswahrscheinlichkeit des Symbols σ_i in einem Match- bzw. einem Insert-Zustand. Die von uns geschätzten Parameter $\vec{\alpha}_M := (\alpha_{M,A}, \alpha_{M,C}, \alpha_{M,G}, \alpha_{M,T})$ für die Emissionswahrscheinlichkeiten der Match-Zustände bzw.
$\vec{\alpha}_I := (\alpha_{I,A}, \alpha_{I,C}, \alpha_{I,G}, \alpha_{I,T})$ für die Insert-Emissionswahrscheinlichkeiten sind
für HIV

$$\begin{aligned} \vec{\alpha}_M &= (0.0895, 0.0474, 0.062, 0.053) \\ \vec{\alpha}_I &= (1.0106, 1.0058, 1.0089, 1.0057) \end{aligned} \tag{6.2}$$

und für HCV

$$\begin{aligned} \vec{\alpha}_M &= (0.0318, 0.0427, 0.0402, 0.0340) \\ \vec{\alpha}_I &= (1.0006, 1.0006, 1.0006, 1.0006) \end{aligned} \tag{6.3}$$

Für die in (6.1) angegebenen Parameter $P_{\text{Insert}}, P_{\text{D,init}}$ und $P_{\text{D,ext}}$ verglichen wir für verschiedene Sprungwahrscheinlichkeiten P_{jump} die vom jpHMM-Programm vorhergesagte Rekombination aller CRFs des Trainingsdatensatzes mit der entsprechenden, in der aktuellen Literatur veröffentlichten Rekombination.
Berücksichtigt wurden dabei die in der Rekombination auftretenden (Sub-)Subtypen, die Anzahl der Bruchstellen und die Position der Bruchstellen.

Die Sprungwahrscheinlichkeit, die die höchste Zahl an Übereinstimmungen zwischen den vom jpHMM-Programm ermittelten Rekombinationen und den veröffentlichten Daten liefert, ist

$$P_{\text{jump}} = 10^{-9} \tag{6.4}$$

Für diese Sprungwahrscheinlichkeit liegen für 70% der betrachteten CRFs alle vorhergesagten Bruchstellen innerhalb eines Bereichs von $150nt$ der entsprechenden veröffentlichten Bruchstellen. Die durchschnittliche Abweichung der Bruchstellen von den veröffentlichten Bruchstellen für diese CRFs beträgt $27nt$. Die Auswertung hierzu wurde von Ming Zhang am Los Alamos National Laboratory (LANL) vorgenommen.

6.5 Auswertung

Die Rekombination der Sequenzen der vier Testdatensätze wurde mithilfe des jpHMMs bestimmt, dessen Parameter auf dem gegebenen MSA und den Sequenzen des Trainingsdatensatzes geschätzt bzw. empirisch ermittelt wurden. Die vom jpHMM vorhergesagten Subtypen und die entsprechenden Bruchstellen wurden mit den bekannten, künstlich erzeugten bzw. veröffentlichten Daten verglichen. Die Ergebnisse für die künstlich erzeugten Sequenzen (**ACRFs**) von HIV verglichen wir zusätzlich mit den für dieselben Sequenzen durch das Programm-Paket *Simplot* [12], eines der am häufigsten verwendeten Werkzeuge zur Vorhersage von Rekombinationen, erhaltenen Ergebnisse.
Simplot beinhaltet zwei verschiedene Methoden zur Identifikation von Rekombinationen, die jeweils die paarweise Ähnlichkeit der Anfragesequenz zu jeder Referenzsequenz mittels eines *sliding windows* auf dem multiplen Alignment aller betrachteten Sequenzen bestimmen. Zum einen wird innerhalb des betrachteten Fensters die prozentuale Identität der Anfragesequenz zu jeder der Referenzsequenzen berechnet und der mittleren Position der Anfragesequenz in diesem Fenster zugeordnet [12]. Durch Verschieben des Fensters entlang dieses Alignments (*sliding window*) können so jeder Position der Anfragesequenz die für das entsprechende Fenster berechneten Werte zugeordnet werden. Diese Werte werden mit einem sog. *similarity plot* graphisch dargestellt, so dass für jede Referenzsequenz eine Kurve ensteht, die die Ähnlichkeit zur Anfragesequenz beschreibt. Die Referenzsequenz, der für eine bestimmte Position der Anfragesequenz die höchste Identität zugeordnet wird, gibt den Subtyp an, der für diese Position vorhergesagt wird. Die zweite in Simplot verwendete Methode ist *bootscanning* [20]. Hier wird für die jeweils betrachteten Abschnitte der Sequenzen ein phylogenetischer Baum erzeugt. Positionen, an denen sich das Verzweigungsmuster des Baumes ändert, markieren Rekombinations-Bruchstellen.
Die Ergebnisse für die Testdatensätze wurden für HIV von Ming Zhang und für HCV von Carla Kuiken (LANL) ausgewertet.

HIV

Für den Testdatensatz **ACRFs** stimmten die vom jpHMM vorhergesagten Subtypen der jeweiligen Rekombination für alle Sequenzen mit den gegebenen Daten überein, wogegen es Abweichungen für die Positionen der vorhergesagten Bruchstellen gab. Zur Bestimmung der Genauigkeit der vom jpHMM vorhergesagten Bruchstellen wurden von Ming Zhang sowohl für das jpHMM als auch für Simplot für jede Sequenz des Datensatzes die Abstände der vorhergesagten Bruchstellen zu den entsprechenden wirklichen Bruchstellen bestimmt. Für die künstlichen Sequenzen mit Bruchstellen an jeder 1000-sten Position sind diese in der, ebenfalls von Ming Zhang angefertigten, Abb. 6.1 dargestellt. Für die beiden verschiedenen Arten von Sequenzen, *Inter-Subtyp-rekombinante* und *Inter-Sub-Subtyp-rekombinante* Sequenzen, berechneten wir jeweils den Median der Abstände und den Interquartilsabstand. Diese sind in Tabelle 6.1 angegeben.

	jpHMM		Simplot	
	Inter-Subtyp	*Inter-Sub-Subtyp*	*Inter-Subtyp*	*Inter-Sub-Subtyp*
Median	10	9	53.5	84
IQA	$4-15$	$3.5-19$	$19-72$	$19.5-122$

Tabelle 6.1: Median und Interquartilsabstand (IQA) der Abstände der vom jpHMM und von Simplot vorhergesagten Bruchstellen zu den wirklichen Bruchstellen für die *Inter-Subtyp-* und *Inter-Sub-Subtyp-rekombinanten* Sequenzen

Um eine Aussage über möglicherweise signifikante Unterschiede in der Vorhersage der beiden Programme treffen zu können, wendeten wir den Vorzeichenrangtest nach Wilcoxon [28] an. Mithilfe dieses nicht-parametrischen Tests liess sich feststellen, dass die Mediane der Abstände, der vom jpHMM und von Simplot vorhergesagten Bruchstellen zu den wirklichen Bruchstellen, für die *Inter-Subtyp-rekombinanten* Sequenzen für $p < 10^{-9}$ und für die *Inter-Sub-Subtyp*-rekombinanten Sequenzen für $p < 10^{-7}$ signifikant verschieden sind. Die Vorhersage des jpHMMs ist für diese Sequenzen also genauer als die Vorhersage mit Simplot.

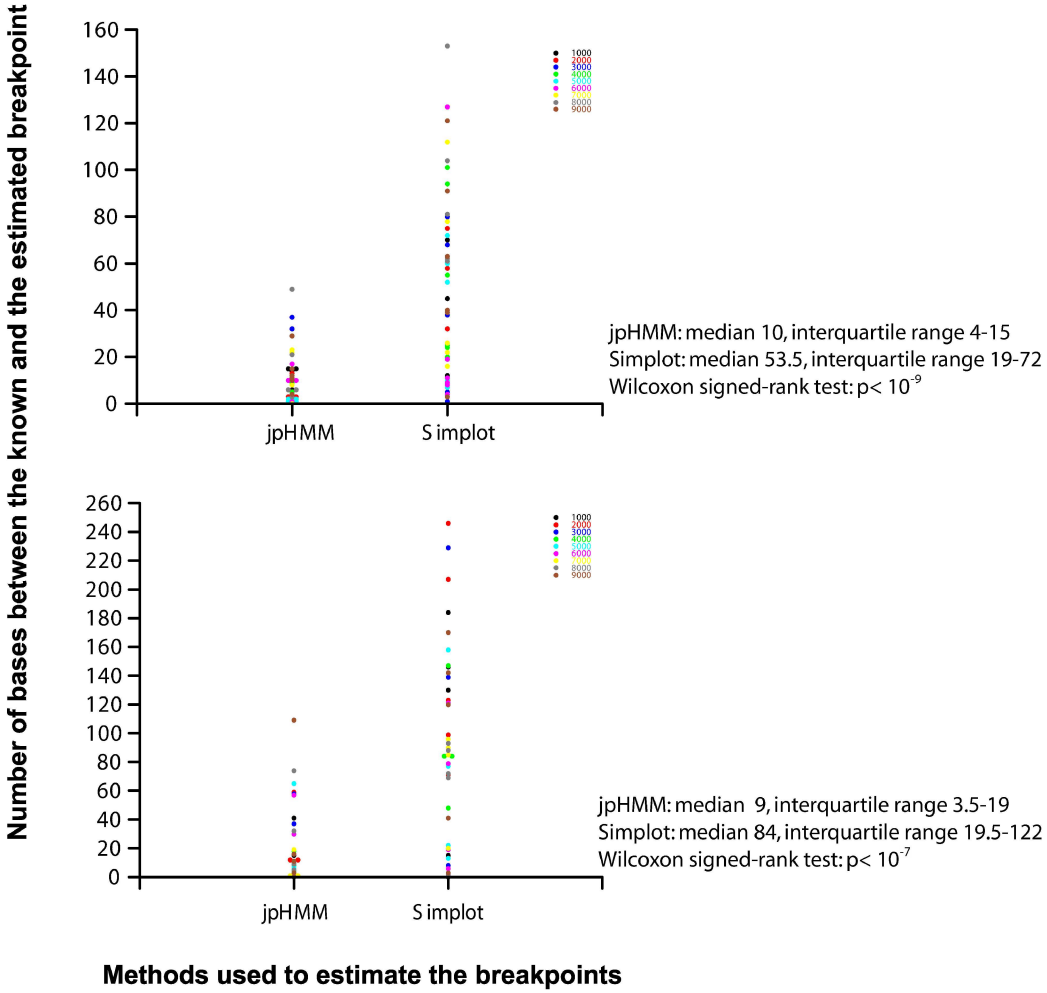


Abbildung 6.1: Vergleich der Ergebnisse des Datensatzes **ACRFs** für das jpHMM und Simplot. In der Abbildung sind die Ergebnisse für die Sequenzen des Datensatzes mit einer Bruchstelle an jeder 1000-sten Position angegeben. Für jede Sequenz ist für jede vom entsprechenden Programm vorhergesagte Bruchstelle der Abstand (in *nt*) zur wirklichen Bruchstelle angegeben. Jeder Bruchstelle ist dabei eine andere Farbe zugeteilt. **a)** Ergebnisse für die *Inter-Subtyp*-rekombinanten Sequenzen. **b)** Ergebnisse für die *Inter-Sub-Subtyp*-rekombinanten Sequenzen. Diese Abb. wurde von Ming Zhang (LANL) erstellt.

Für die Testdatensätze **500nts** und **7000nts** liegen bisher keine genauen Auswertungen vor. Jedoch ergaben sich für diese beiden Datensätze, wie auch für Sequenzen des Trainingsdatensatzes, häufig Schwierigkeiten in der Vorhersage der Subtypen H, J und K. Diese Subtypen wurden in den meisten Fällen nicht erkannt oder die ent-

sprechenden Bruchstellen an falschen Positionen bestimmt, wodurch für CRFs, die diese Subtypen enthalten, oft eine Rekombination vorhergesagt wurde, die nicht mit den veröffentlichten Daten übereinstimmt. Für Sequenzen der Länge $500nt$ nutzten wir eine *Beam-Weite* von $\mathcal{B} = 10^{-45}$ statt $\mathcal{B} = 10^{-20}$, da diese Sequenzen zum Teil Endfragmente existierender CRFs bzw. Subtypen sind, und somit den letzten Spalten des gegebenen MSA ähnlich sind. Für eine *Beam-Weite* von $\mathcal{B} = 10^{-20}$ werden die Pfade im jpHMM, die ein Alignment der entsprechenden Anfragesequenz mit den letzten Spalten des MSA ermöglichen, bereits im Beam-Algorithmus als 'irrelevante' Pfade ausgeschlossen, wodurch möglicherweise eine falsche Rekombination vorhergesagt wird.

HCV

Nach heutigem Wissensstand enthalten rekombinante HCV-Sequenzen nur eine einzige Bruchstelle und setzen sich somit nur aus zwei verschiedenen (Sub-)Subtypen zusammen. Für die beiden getesten Sequenzen sagt das jpHMM jedoch jeweils zwei Bruchstellen vorher.
Für die St.Petersburg-RF des Typs 1b/2k wird eine Rekombination der Typen 2a, 2k und 1b, in der genannten Reihenfolge, beginnend an Position 1 der Sequenz, vorhergesagt, mit Bruchstellen an den Positionen 238/239 für den Übergang 2a/1b, und 3186/3187 für 1b/2k. Die für diese Sequenz veröffentlichte, und auf Simplot basierende Vorhersage der Bruchstelle des Übergangs 1b/2k stimmt mit der genannten Position überein.
Für die künstlich erzeugte RF des Typs 1a/2a wird eine Rekombination der Form 1a/2a/1a vorhergesagt, mit Bruchstellen an den Positionen 349/350 und 2759/2760. Die für den zuletzt genannten Übergang vorhergesagte Bruchstelle weicht von der exakten Bruchstelle um 6 Nukleotide ab, da diese sich an Position 2765/2766 befindet. Alle Position sind in HCV-H-Nummerierung angegeben, d.h. die jeweils angegebene Position ist die Position auf dem Referenzstrang HCV-H (GenBank: M67463), die der berechneten Position entspricht (http://hcv.lanl.gov/content/hcv-db/LOCATE/locate.html).

6.6 Diskussion

Eine der am häufigsten verwendeten Methoden zur Identifikation rekombinanter Sequenzen ist Simplot. Die im vorhergehenden Abschnitt (6.5) beschriebenen Ergebnisse zeigen, dass das jpHMM-Programm Rekombinationen in genomischen Sequenzen von HIV und HCV sehr gut aufdecken kann und in den meisten Fällen eine genauere Vorhersage der Bruchstellen in den betrachteten Sequenzen liefert als Simplot. Jedoch wird anhand der Ergebnisse auch sichtbar, dass dies grösstenteils dann der Fall ist, wenn viele Referenzsequenzen der Subtypen vorliegen. Im Fall von nur

sehr wenigen Referenzsequenzen im MSA, wie für die HIV-Subtypen H, J und K, ist die in diesen Subtypen gegebene Information über die Verteilung der Nukleotide in den Spalten des Subtyps zu gering, um diese Subtypen in einer Anfragesequenz zu identifizieren. Ein weiterer Grund dafür kann sein, dass die für diese drei Subtypen vorliegenden Sequenzen den entsprechenden Subtyp nicht sehr gut repräsentieren.
Der Grund für eine zusätzliche Rekombinationsvorhersage des jpHMMs an Position 238/239 für die St-Petersburg-RF bzw. an Position 349/350 für die künstlich erzeugte RF kann die sehr hohe Konserviertheit des HCV-Genoms im Bereich der ersten 350 Nukleotide sein, so dass sich die unterschiedlichen Genotypen in diesem Bereich nur kaum voneinander unterscheiden. Selbst eine phylogenetische Klassifizierung der Genotypen ist in diesem Bereich oft nicht möglich (Carla Kuiken, unveröffentlichte Ergebnisse).
Um also eine gute Vorhersage für rekombinante Sequenzen zu erhalten, die sich aus Subtypen zusammensetzen, für die nur wenige repräsentative Sequenzen vorliegen, sollte das jpHMM mit anderen Methoden, wie zum Beispiel Simplot, kombiniert werden. Im Fall einer grossen vorliegenden Datenmenge zeigen die Ergebnisse jedoch, dass ein jpHMM eine sehr gute Methode ist, um genaue Vorhersagen über die Subtypzusammensetzung und die entsprechenden Bruchstellen in rekombinanten Sequenzen zu treffen. In den meisten von uns betrachteten Fällen ist diese Vorhersage sogar sehr viel besser als die von Simplot, einer der zur Zeit am häufigsten genutzten Methoden. Ein Vorteil des jpHMMs gegenüber Simplot ist, dass, statt einiger Referenzsequenzen oder einer einzigen Konsensussequenz, alle, zu einem Subtyp gehörenden Sequenzen, und davon, statt einiger Spalten ('sliding window'), das gesamte Alignment betrachtet werden. Ein jpHMM kann dadurch bei der Identifikation von Rekombinationen sehr viel mehr Informationen der bereits bekannten Subtypen berücksichtigen als Simplot.

Kapitel 7

Ausblick

7.1 A-posteriori-Wahrscheinlichkeiten

Mithilfe des Viterbi-Algorithmus können wir den Viterbi-Pfad einer Anfragesequenz $s[1,l]$ durch ein jpHMM mit Zustandsraum Z^+ bestimmen, und somit die wahrscheinlichste, dieser Sequenz zugrundeliegende Rekombination aus den gegebenen Subtypen identifizieren. Es ist jedoch möglich, dass für eine bestimmte Position der Sequenz ein anderer Subtyp ähnlich wahrscheinlich wie der im Viterbi-Pfad bestimmte Subtyp ist. Deshalb macht es Sinn, für jede Position der Anfragesequenz die bedingte Wahrscheinlichkeit anzugeben, mit der jeder der in der Sequenzfamilie enthaltenen Subtypen ihr zugrundeliegt, die sog. *a-posteriori-Wahrscheinlichkeiten* der Subtypen.
Die a-posteriori-Wahrscheinlichkeit eines Subtyps $\mathcal{S}_j$ für eine bestimmte Position i der Anfragesequenz können wir berechnen, indem wir für jeden Zustand des Profil-HMMs $\mathcal{P}_j$ die a-posteriori-Wahrscheinlichkeit für diese Position berechnen, und diese Wahrscheinlichkeiten dann aufsummieren. Um die a-posteriori-Wahrscheinlichkeiten eines Zustandes für eine bestimmte Position zu berechnen, benötigen wir den *Rückwärts-Algorithmus*. Dieser berechnet für jede Position i in s und für jeden Zustand $z \in Z$ die Wahrscheinlichkeit, mit einem beliebigen Pfad durch das jpHMM, beginnend im Zustand z, das Suffix $s[i+1,l]$ der Sequenz zu erzeugen. Diese Wahrscheinlichkeit nennen wir die *Rückwärts-Variable* $\beta_{z,i}$.

Rückwärts-Variable

Definition 7.1 *Sei $Y_1, \ldots, Y_j, Y_{j+1}, \ldots, Y_r$, $r \geq 1$, eine Folge von Emissionen. Dann definieren wir*

$$S_{j+1,r} := Y_{j+1} \ldots Y_r \tag{7.1}$$

$$\textit{und} \quad S_r := Y_1 \ldots Y_r \quad (\textit{siehe } 1.17). \tag{7.2}$$

Definition 7.2 *Für eine Markov-Kette $X_1, X_2, \ldots$ mit Werten in Z^+ sei $|X|$ die Anzahl der Elemente der Kette bevor zum ersten Mal der End-Zustand E angenommen wird. D.h. für $X_1, \ldots, X_{r+1}$, mit $X_i \neq E, i \leq r$, und $X_{r+1} = E$, gilt $|X| = r$.*

Definition 7.3 (Rückwärts-Variable) *Sei $s[1,l]$ ein Beobachtung in Σ^l und z ein Zustand in Z. Sei $j > 0$, so dass gilt $P(X_j = z) > 0$.*
Dann definieren wir die Rückwärts-Variablen

$$\begin{aligned} \beta_{z,i} &:= P\big(S_{j+1,|X|} = s[i+1,l] \,\big|\, X_j = z\big),\ i = 0, \ldots, l-1, \\ \beta_{z,l} &:= P\big(S_{j+1,|X|} = \epsilon \,\big|\, X_j = z\big). \end{aligned} \tag{7.3}$$

Die Wahl von j ist dabei beliebig, da $X_1, X_2, \ldots$ eine homogene Markov-Kette 1. Ordnung ist.

Lemma 7.4 *Sei $s[1,l]$ eine Beobachtung in Σ^l.*
Dann gilt für alle $z \in Z$

$$\beta_{z,l} = \sum_{z' \in Z} t_{z,z'} e_{z',\epsilon} \beta_{z',l}, \tag{7.4}$$

$$\beta_{z,i} = \sum_{z' \in Z} t_{z,z'} e_{z',s[i+1,i+1+d_{z'})} \beta_{z',i+d_{z'}}, \qquad i = 0, \ldots, l-1. \tag{7.5}$$

Bemerkung 7.5 Da dieses Kapitel nur zum Ausblick auf mögliche Erweiterungen eines *springenden Profil-Hidden-Markov-Modells* bzw. des dafür implementierten Programms dient, sind alle in diesem Kapitel verwendeten Lemmata ohne Beweis gegeben.

A-posteriori-Wahrscheinlichkeit eines Subtyps

Definition 7.6 *Sei $s[1,l]$ eine Anfragesequenz.*
Sei für eine Markov-Kette 1.Ordnung $X_1, X_2, \ldots$ mit Werten in Z^+

$$d_{X_j} := \begin{cases} 1, & \textit{falls} \quad X_j = z \textit{ mit } z \in Z \backslash Z', \\ 0, & \textit{falls} \quad X_j = z \textit{ mit } z \in Z' \cup \{B, E\}, \end{cases} \tag{7.6}$$

$$\textit{und} \qquad D_j := \sum_{i=1}^{j} d_{X_i} \quad \forall j = 1, \ldots, r. \tag{7.7}$$

Es sei

$$J(i) := \arg\min_{j \geq 1} \{D_j \geq i\} \tag{7.8}$$

der erste Zeitpunkt, zu dem das i-te Zeichen der Anfragesequenz, s_i, emittiert wird.

Definition 7.7 (A-posteriori-Wahrscheinlichkeit eines Zustands)
Sei $s[1,l]$ eine Anfragesequenz. Dann ist

$$P\big(X_{J(i)} = z \,|\, S_{|X|} = s[1,l]\big) \tag{7.9}$$

die a-posteriori-Wahrscheinlichkeit *des Zustandes $z \in Z$ an Position i der Anfragesequenz s.*

Lemma 7.8 *Sei $s[1,l]$ eine Anfragesequenz. Für die a-posteriori-Wahrscheinlichkeit des Zustands z an Position i der Sequenz s gilt*

$$P\big(X_{J(i)} = z \,|\, S_{|X|} = s[1,l]\big) = \begin{cases} \dfrac{\alpha_{z,i} \cdot \beta_{z,i}}{\alpha_{*,l}} & , falls\ z \in Z \backslash Z' \\ 0 & , falls\ z \in Z' \end{cases} \tag{7.10}$$

Mithilfe der Vorwärts- und der Rückwärts-Variablen können wir also die *a-posteriori*-Wahrscheinlichkeit eines Zustandes $z \in Z$ berechnen.
Die Wahrscheinlichkeit, mit der ein bestimmter Subtyp $\mathcal{S}_k$ der Sequenzfamilie $\mathcal{S} = \{S_1, \ldots, S_K\}$ einer Position i der Anfragesequenz $s[1,l]$ zugrundeliegt, nennen wir die *a-posteriori-Wahrscheinlichkeit $P_{\text{post},i}(\mathcal{S}_k)$ des Subtyps $\mathcal{S}_k$.*

Lemma 7.9 *Sei A_k das Subalignment des Subtyps $\mathcal{S}_k$ im gegebenen multiplen Sequenzalignment A. Sei Z_k der Zustandsraum des Profil-HMMs $\mathcal{P}_k$ für das Subalignment A_k. Dann gilt für die a-posteriori-Wahrscheinlichkeit $P_{\text{post},i}(\mathcal{S}_k)$ des Subtyps $\mathcal{S}_k$*

$$\begin{aligned} P_{\text{post},i}(\mathcal{S}_k) &= \sum_{z \in Z_k} P\big(X_{J(i)} = z \,|\, S_{|X|} = s[1,l]\big) \\ &= \frac{\sum_{z \in Z_k} \alpha_{z,i} \cdot \beta_{z,i}}{\alpha_{*,l}} \end{aligned}$$

Implementation des Rückwärts-Algorithmus

Der Rückwärts-Algorithmus und die Berechnung der a-posteriori-Wahrscheinlichkeiten wurden im Rahmen dieser Arbeit bereits implementiert. Die zur Berechnung der a-posteriori-Wahrscheinlichkeiten benötigten Vorwärts-Variablen wurden parallel zu den Viterbi-Variablen berechnet, d.h. für jede Position der Anfragesequenz wurden ebenfalls nur die Vorwärts-Variablen der für diese Position im Viterbi-Algorithmus aktiven Zustände berechnet. Der Rückwärts-Algorithmus wurde so implementiert, dass für jede Position der Anfragesequenz nur die Rückwärts-Variablen der Zustände berechnet wurden, die für diese Position im Vorwärts-Algorithmus aktiv waren. Da jedoch in der Berechnung der Rückwärts-Variablen für eine Position i und einen

Zustand z statt der Vorgänger von z die Nachfolger von z betrachtet werden, wurde in einigen Fällen die Rückwärts-Variable eines Zustandes z berechnet, der keine aktiven Nachfolger hatte. Dieses Problem liess sich vermeiden, indem wir diese Zustände an dieser Position als nicht-aktive Zustände behandelten. Dadurch ergab sich jedoch in einigen Fällen das Problem, dass die Summe aller für diese Position der Anfragesequenz berechneten a-posteriori-Wahrscheinlichkeiten ungleich 1 war. Nur für eine *Beam-Weite* $\mathcal{B}$ sehr nahe bei 0, $\mathcal{B} \leq 10^{-45}$, trat dieses Problem nicht auf, da in diesem Fall für jede Position der Anfragesequenz nahezu alle Zustände des jpHMMs aktiv waren.
Eine Idee zur Vermeidung dieses Problem ist die direkte Integration des *Beam-Search*-Algorithmus in den Rückwärts-Algorithmus. Das bedeutet, dass in der Berechnung der Rückwärts-Variablen für jede Position der Anfragesequenz die aktiven Zustände unabhängig von den im Vorwärts-Algorithmus aktivierten Zuständen bestimmt werden. Dies kann allerdings zu dem Problem führen, dass für eine Position der Anfragesequenz im Vorwärts- und Rückwärts-Algorithmus unterschiedliche Zustände des Zustandsraums Z aktiv sind.

7.2 Ein jpHMM zur Proteinklassifizierung

Ein springendes Profil-HMM kann, wie ein Profil-HMM oder der Jumping-Alignment-Algorithmus (JALI), zur Klassifizierung von Proteinen genutzt werden. Sei $\mathcal{S} = \{S_1, \ldots, S_K\}$ eine gegebene Proteinfamilie von K Proteinsequenzen, und $s[1, l]$ eine Anfragesequenz, deren Ähnlichkeit zur Proteinfamilie bestimmt werden soll.
Dann entwickeln wir für die Sequenzfamilie ein jpHMM, indem wir für jede Sequenz $S_i \in \mathcal{S}$ ein Profil-HMM $\mathcal{P}_i$ entwickeln, und diese Profil-HMMs $\mathcal{P}_i$, wie in Abschnitt 4.1 beschrieben, durch Übergänge miteinander verbinden. So erhalten wir, wie in JALI, die Möglichkeit, jede Position der Anfragesequenz s zu jeweils einer Referenzsequenz des MSA zu alignieren, wobei die Referenzsequenz, aufgrund der zwischen den verschiedenen Profil-HMMs erlaubten Übergänge, innerhalb des Alignments wechseln kann. Wir nennen diese Profil-HMMs hier *Zeilen-Profil-HMMs*.
Entwickeln wir zusätzlich zu den Profil-HMMs der einzelnen Sequenzen der Familie ein Profil-HMM für das gesamte MSA, das sog. *Spalten-Profil-HMM*, und lassen ebenfalls Sprünge zwischen diesem und den Zeilen-Profil-HMMs zu, dann erhalten wir zusätzlich die Möglichkeit, jede Position der Anfragesequenz zu einer Spalte des MSA zu alignieren. Durch die erlaubten Sprünge zwischen dem Spalten- und den Zeilen-Profil-HMMs ist die Möglichkeit gegeben, sowohl die Konserviertheit der Spalten als auch Muster in den einzelnen Sequenzen des MSA bei der Proteinklassifizierung zu berücksichtigen. Dadurch werden die jeweiligen Vorteile (Abschnitt 3.2) der in Kap. 2 und Kap. 3 vorgestellten Methoden zur Proteinklassifizierung, ein Profil-HMM und der Jumping-Alignment-Algorithmus, in einem jpHMM vereint.
Eine weitere Möglichkeit besteht darin, die in der Proteinfamilie enthaltenen Sequen-

zen zu Unterfamilien zusammenzufassen, und, statt für jede Sequenz der Proteinfamilie, für jede dieser Unterfamilien ein Profil-HMM $\mathcal{P}_i$ zu entwickeln. Dies macht Sinn, falls einige der in der Proteinfamilie enthaltenen Sequenzen sehr ähnlich sind oder dieselben Muster enthalten. Dadurch ist es möglich, konservierte Bereiche, die nur in einigen Sequenzen der Familie enthalten sind, stärker zu berücksichtigen.

Zusammenfassung

In dieser Arbeit wurde ein neues Modell, ein sog. *springendes Profil-Hidden-Markov-Modell* (*jpHMM*), zum Vergleich von DNA- bzw. Proteinsequenzen mit einer gegebenen Sequenzfamilie entwickelt. Dabei nehmen wir an, dass sich die Sequenzfamilie in verschiedene Subtypen, jeweils bestehend aus einer oder mehreren Sequenzen, einteilen lässt. Ein jpHMM ist spezielles Hidden-Markov-Modell, das die Idee eines Jumping Alignments mit der eines Profil-Hidden-Markov-Modells auf folgende Weise verknüpft: Für jeden Subtyp der Sequenzfamilie wird für das entsprechende Subalignment des multiplen Alignments der Sequenzfamilie ein Profil-HMM entwickelt. Diese Profil-HMMs werden durch Übergänge, sog. Sprünge, miteinander verbunden. Um nun die Ähnlichkeit einer Anfragesequenz zu der Sequenzfamilie bzw. zu den einzelnen Subtypen zu bestimmen, wird diese zu der Sequenzfamilie aligniert, indem jede Position der Sequenz zu einem Referenzsubtyp aligniert wird. Aufgrund der möglichen Sprünge zwischen den verschiedenen Profil-HMMs kann dieser innerhalb des Alignments wechseln. Mithilfe des Viterbi-Pfades der Anfragesequenz lässt sich dann feststellen, welche Bereiche der Sequenz welchen Subtypen der Sequenzfamilie am ähnlichsten sind, und wo die Sprungstellen, die sog. Bruchstellen, sind.
Angewandt wurde das jpHMM zur Identifizierung rekombinanter HIV- und HCV-Sequenzen. Für diese Viren lag uns jeweils ein multiples Sequenzalignment einer Vielzahl von Subtypen bzw. Sub-Subtypen vor, für das wir in der beschriebenen Weise ein jpHMM entwickelten. Für mehrere Datensätze von Anfragesequenzen wurde mithilfe des Viterbi-Pfades der jeweiligen Sequenz durch das jpHMM eine Rekombination aus den gegebenen Subtypen und die entsprechenden Bruchstellen bestimmt. Zur Auswertung der Genauigkeit dieser Vorhersage verglichen wir unsere Ergebnisse mit denen von Simplot, einem der zur Zeit am häufigsten verwendeten Programme zur Identifizierung von Rekombinationen in HIV- und HCV-Sequenzen, und mit einigen in der aktuellen Literatur veröffentlichten Rekombinationen, die zum Teil ebenfalls auf Simplot beruhen. Die Auswertung zeigt, dass das jpHMM Rekombinationen in HIV- und HCV-Sequenzen sehr gut aufdecken kann, und in den meisten Fällen eine genauere Vorhersage der Bruchstellen in den betrachteten Sequenzen liefert als Simplot. Jedoch wird anhand der Ergebnisse für die HIV-Sequenzen auch sichtbar, dass dies grösstenteils dann der Fall ist, wenn viele Referenzsequenzen der Subtypen vorliegen. Im Fall von nur sehr wenigen Referenzsequenzen im MSA ist die in diesen Subtypen gegebene Information über die Verteilung der Nukleotide in

den Spalten der entsprechenden Subalignments im MSA möglicherweise zu gering, um diese Subtypen in einer Anfragesequenz zu identifizieren. Ein weiterer Grund dafür kann jedoch auch sein, dass die für diese Subtypen vorliegenden Sequenzen den entsprechenden Subtyp nicht sehr gut repräsentieren.
Rekombinierte HCV-Sequenzen besitzen im Gegensatz zu HIV-Sequenzen jeweils nur eine Bruchstelle. Diese wird, in Bezug auf die veröffentlichten Daten, für die beiden betrachteten Sequenzen vom jpHMM richtig bzw. bis auf wenige Positionen genau vorhergesagt. Zusätzlich dazu wird jedoch jeweils noch eine zweite Bruchstelle im Bereich der ersten 350 Nukleotide vorhergesagt. Ein Grund für diese Vorhersage kann die sehr hohe Konserviertheit des HCV-Genoms in diesem Bereich sein. Selbst eine phylogenetische Klassifizierung der Genotypen ist in diesem Bereich oft nicht möglich.
Um also eine gute Vorhersage für rekombinante Sequenzen zu erhalten, die sich aus Subtypen zusammensetzen, für die nur wenige repräsentative Sequenzen vorliegen, sollte das jpHMM mit anderen Methoden, wie zum Beispiel Simplot, kombiniert werden. Im Fall einer grossen vorliegenden Datenmenge zeigen die Ergebnisse jedoch, dass ein jpHMM eine sehr gute Methode ist, um genaue Vorhersagen über die Rekombination von HIV- und HCV-Sequenzen zu treffen. Für die meisten von uns betrachteten Sequenzen ist diese Vorhersage sogar sehr viel besser als die von Simplot. Ein Vorteil des jpHMMs gegenüber Simplot ist dabei, dass, statt eines 'sliding windows' über einem Alignment einiger Referenz- oder Konsensussequenzen der Subtypen, das gesamte Alignment der Sequenzfamilie betrachtet wird. Ein jpHMM kann dadurch bei der Identifikation von Rekombinationen sehr viel mehr Informationen der bereits bekannten Subtypen berücksichtigen als Simplot.

Anhang

Paarweise Alignments

Eine Möglichkeit die Ähnlichkeit einer Anfragesequenz zu einer gegebenen Sequenz zu bestimmen ist es, das optimale *paarweise Alignment* dieser beiden Sequenzen zu bilden. Ein paarweises Alignment zweier Sequenzen besteht darin die beiden Sequenzen symbolweise zueinander auszurichten, so dass an jeder Position entweder zwei Symbole miteinander gepaart werden (*Match* bzw. *Mismatch*), gegenüber eines Symbols in der gegebenen Sequenz eine Lücke (engl. *gap*) in die Anfragesequenz eingefügt wird (*Deletion*), oder eine Lücke in die gegebene Sequenz eingefügt wird, was dem Einfügen eines Symbols (*Insertion*) in der Anfragesequenz entspricht.
Grundlage eines paarweisen Alignments ist meistens eine *Scoringmatrix*, die jedem Paar von Symbolen (Aminosäuren bzw. Nukleotiden) einen *Score* zuordnet. Dieser Score spiegelt die Ähnlichkeit des betrachteten Symbol-Paares wieder. Das Einfügen einer Lücke wird mit einem bestimmten Wert ($gapcost > 0$) bestraft.
Indem man die Summe aller Scores der im Alignment erzeugten Symbol-Paare bildet, und davon für jede eingefügte Lücke die *gapcost* abzieht, erhält man einen Score für das gesamte Alignment der beiden Sequenzen, den *Alignment-Score*.
Das optimale paarweise Alignment ist das paarweise Alignment, dem der grösste Alignment-Score zugeordnet ist.
Zwei bekannte Algorithmen zur Berechnung paarweiser Alignments sind der Needleman-Wunsch-Algorithmus [14] und der Smith-Waterman-Algorithmus [23]. Beide beruhen auf dem selben Prinzip und bedienen sich der *Dynamischen Programmierung*.

Needleman-Wunsch-Algorithmus

Mithilfe des Needleman-Wunsch-Algorithmus wird das optimale *globale* Alignment zweier Sequenzen gebildet.
Seien zwei Sequenzen $x = x_1 \dots x_m$ und $y = y_1 \dots y_n$ gegeben. Sei s eine Scoring-Matrix die jedem Paar von Symbolen (x_i, y_j) einen bestimmten Score $s(x_i, y_j)$ zuordnet. Sei F eine Matrix mit $m{+}1$ Zeilen und $n{+}1$ Spalten. Der Eintrag $F(i, j)$ definiert

den maximalen Score des Alignments der beiden Präfixe $x_1 \dots x_i$ und $y_1 \dots y_j$. $F(0,0)$ wird mit 0 initialisiert. Um das Einfügen von Lücken am Anfang beider Sequenzen zu ermöglichen, entspricht die 0-te Zeile bzw. 0-te Spalte der Matrix F der Position vor dem ersten Symbol der Sequenz x bzw. der Sequenz y. In der Zeile $F(0,j)$ steht der Score für das Alignment des Präfixes $y_1 \dots y_j$ mit j Lücken. Das bedeutet, dass in die Sequenz x vor dem ersten Symbol x_1 j Lücken eingefügt werden, wodurch sich $F(0,j) = -j \cdot \mathit{gapcost}$ ergibt. Analog dazu ist die Spalte $F(i,0)$ definiert und es gilt $F(i,0) = -i \cdot \mathit{gapcost}$. Mit diesen Voraussetzungen können wir die Matrix F rekursiv, von links oben nach rechts unten, berechnen. Es gilt dann:

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_i), \\ F(i-1,j) - \mathit{gapcost}, \\ F(i,j-1) - \mathit{gapcost}. \end{cases} \tag{7.11}$$

Nach dieser Definition enthält $F(m,n)$ den maximalen Score des globalen Alignments der beiden Sequenzen x und y, d.h. den Score des besten globalen Alignments von x und y. Durch Zurückverfolgen des Pfades bis zum Eintrag $F(0,0)$ wird das optimale globale Alignment bestimmt.

Smith-Waterman-Algorithmus

Der Smith-Waterman-Algorithmus ist eine Methode zur Berechnung des optimalen *lokalen* Alignments von zwei Sequenzen, d.h. des besten Alignments zweier Subsequenzen von x und y. Dieser Algorithmus funktioniert in ähnlicher Weise wie der Needleman-Wunsch-Algorithmus, mit folgenden Unterschieden:

$$F(i,j) = max \begin{cases} 0, \\ F(i-1,j-1) + s(x_i, y_i), \\ F(i-1,j) - \mathit{gapcost}, \\ F(i,j-1) - \mathit{gapcost}. \end{cases} \tag{7.12}$$

Durch Einführen des konstanten Wertes 0 ist die Möglichkeit gegeben, dass jede Position (i,j) in F der Anfang eines neuen lokalen Alignments ist. Ebenso kann jede Position das Ende des lokalen Alignments sein. Diese Positionen werden durch den maximalen Score in F, der in diesem Fall nicht notwendigerweise in $F(m,n)$ zu finden ist, bestimmt. Durch Zurückverfolgen des Pfades in F bis zu einem Eintrag 0 wird das beste lokale Alignment bestimmt.

Zustandsgraphen

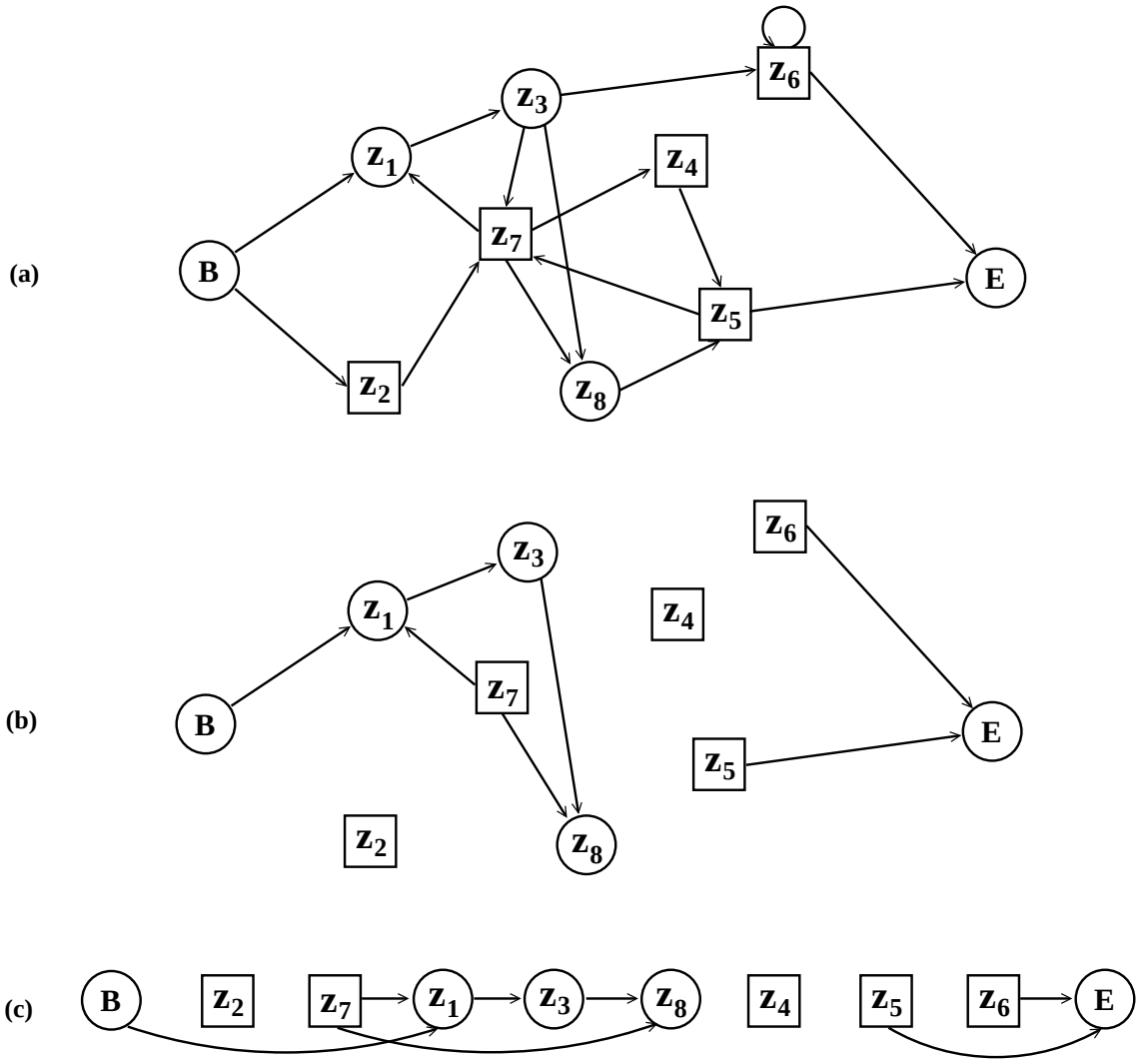


Abbildung 7.1: **(a)** Beispiel für einen Zustandsgraphen $G = (Z, T)$ eines HMMs, wobei $Z = \{z_1, \ldots, z_8\}$ und T die Menge aller möglichen Übergänge zwischen den Zuständen aus Z ist. Zusätzlich dazu sind Übergänge vom Start-Zustand B und zum End-Zustand E eingezeichnet. Kreise kennzeichnen stumme Zustände. **(b)** Der Sub-Zustandsgraph $G' = (Z^+, T')$ des Graphen $G = (Z^+, T)$. **(c)** Der Sub-Zustandsgraph $G' = (Z^+, T')$ topologisch sortiert.

Literaturverzeichnis

[1] J. F. Abril and R. Guigó. gff2ps: visualizing genomic annotations. *Bioinformatics*, 16(8):743–744, 2000.

[2] A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy. The Pfam protein families database. *Nucleic Acids Res.*, 32:D138–D141, 2004.

[3] H. S. Bilofsy, C. Burks, J. W. Fickett, W. Goad, F. I. Lewitter, W. P. Rindone, C. D. Swindell, and C. S. Tung. The GenBank genetic sequence databank. *Nucleic Acids Res.*, 14(1):1–4, 1986.

[4] M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet Mixture Priors to Derive Hidden Markov Models for Protein Families. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology*, volume 12, pages 47–55, Menlo Park, CA, 1993. AAAI/MIT Press.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 1990.

[6] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, UK, 1998.

[7] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.

[8] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Res.*, 19(23):6565–6572, 1991.

[9] R. Hughey and A. Krogh. SAM: Sequence alignment and modeling software system. Technical report, UCSC-CRL-95-7, University of California, Santa Cruz, CA, 1995.

[10] A. Kalinina, H. Norder, S. Mukomolov, and L. O. Magnius. A natural intergenotypic recombinant of hepatitis C virus identified in St. Petersburg. *J. Virology*, 76:4034–4043, 2002.

[11] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[12] K. S. Lole, R. C. Bollinger, R. S. Paranjape, D. Gadkari, S. S. Kulkarni, N. G. Novak, R. Ingersoll, H. W. Sheppard, and S. C. Ray. Full-length human immunodeficiency virus type 1 genomes from subtype C-infected seroconverters in India, with evidence of intersubtype recombination. *J. Virology*, 73:152–160, 1999.

[13] B. Lowerre. The Harpy Speech Recognition System. Technical report, 1976.

[14] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[15] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85(8):2444–2448, 1988.

[16] M. Peeters. Recombinant HIV Sequences: Their Role in the Global Epidemic. In C. Kuiken, B. Foley, B. Hahn, B. Korber, F. McCutchan, P. Marx, J. Mellors, J. Mullins, J. Sodroski, and S. Wolinksy, editors, *HIV Sequence Compendium 2000*, pages I–39–54. Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, Los Alamos, NM, 2000.

[17] T. Plötz and G. A. Fink. Accelerating the Evaluation of Profile HMMs by Pruning Techniques. Technical report, Bielefeld University, Faculty of Technology. Report 2004-03.

[18] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[19] D. L. Robertson, J. P. Anderson, J. A. Bradac, J. K. Carr, B. Foley, R. K. Funkhouser, F. Gao, B. H. Hahn, C. Kuiken, G. H. Learn, T. Leitner, F. McCutchan, S. Osmanov, M. Peeters, D. Pieniazek, M. L. Kalish, M. Salminen, P. M. Sharp, S. Wolinsky, and B. Korber. HIV-1 Nomenclature Proposal. *Science*, 288:55–57, 2000.

[20] M. O. Salminen, J. K. Carr, D. S. Burke, and F. E. McCutchan. Identification of breakpoints in intergenotypic recombinants of HIV type-1 by bootscanning. *AIDS Res. Hum. Retrovir*, 11:1423–1425, 1995.

[21] P. M. Sharp, G. M. Shaw, and B. H. Hahn. Simian Immunodeficiency Virus Infection of Chimpanzees. *J. Virology*, 79(7):3891–3902, 2005.

[22] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet Mixtures: A Method for Improved Detection of Weak but Significant Protein Sequence Homology. *Comput. Applic. Biosci.*, 12:327–345, 1996.

[23] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[24] R. Spang, M. Rehmsmeier, and J. Stoye. Sequence Database Search Using Jumping Alignments. In *Proceedings of ISMB 2000*, pages 367–375, 2000.

[25] R. Spang, M. Rehmsmeier, and J. Stoye. A Novel Approach to Remote Homology Detection: Jumping Alignments. *J. Comp. Biol.*, 9(5):747–760, 2002.

[26] B. Stroustrup. *The C++ programming language.* Addison-Wesley Series in Computer Science, 1991.

[27] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, IT-13:260–269, 1967.

[28] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics*, 1:80–83, 1945.

[29] M. Wistrand and E. L. Sonnhammer. Transition Priors for Protein Hidden Markov Models: An Empirical Study towards Maximum Discrimination. *J. Comp. Biol.*, 11(1):181–193, 2004.

Danksagung

An dieser Stelle möchte ich mich bei allen herzlich bedanken, die mich während dieser Arbeit und meines gesamten Studiums unterstützt und begleitet haben.
Mein besonderer Dank gilt Dr. Mario Stanke, der diese Arbeit betreut, Korrektur gelesen, und mit vielen Anregungen und Diskussionen unterstützt hat.
Für die Betreuung dieser Arbeit danke ich ebenfalls besonders herzlich Prof. Burkhard Morgenstern, der stets ein offenes Ohr für Fragen hatte, und es mir ausserdem ermöglicht hat, während dieser Zeit an Tagungen teilzunehmen.
Herzlich danke ich Prof. Stephan Waack, der die Idee hatte, 'Jumping Alignments' mit 'Profil-HMMs' zu verknüpfen.
Ein grosser Dank geht an Rasmus Steinkamp und Maike Tech für die technische Unterstützung während dieser Arbeit, und an Isabelle Schneider, die während dieser Zeit im selben Zimmer sass und immer Zeit für Probleme und Fragen hatte.
Besonderer Dank gilt Ming Zhang und Carla Kuiken vom Los Alamos National Laboratory für die Bereitstellung der Daten und die Auswertung der Ergebnisse dieser Arbeit.
Allen 'Göttingern' und 'Ex-Göttingern', im Besonderen Birte (für unser ehemaliges Zuhause), Frederike (dafür, dass 'es doch noch mit uns beiden geklappt hat'), Melanie, Annette, Jörg (KJ), Christian, Jens, Max, Nikolai, Jörg (JJ), Janina und Robert, danke ich für die Zeit im Übungssaal, in den Mensen und den Kneipen Göttingens.
Ein riesengrosses Dankeschön geht dabei an Annette für alles (naja, 'que de bétises...' :-)), für unsere (gelebten) Träume, und dafür, dass sie die ganze Zeit da war und ist.
Von ganzem Herzen danke ich Tilman für Perl-Skripte, Korrekturlesen, das Zuhören bei Wutanfällen, und vor allem für das Leben neben der Arbeit.
Von allen die nicht 'hier' waren, danke ich ganz besonders Fränzi, Susi und Haydar für ihre Freundschaft, und dafür, dass sie während dieser Zeit trotzdem 'da' waren.
Ausserdem möchte ich mich bei meinen Geschwistern Julia und Yannick bedanken, die das Leben um einiges schöner und lustiger machen.
Mein herzlichster Dank gilt meinem Großvater für seine finanzielle Unterstützung während meines Studiums.
Ganz besonders und vor allem danke ich meinen Eltern, die mich während meines gesamten Studiums in jeglicher Hinsicht unterstützt haben, und ohne die das alles gar nicht möglich gewesen wäre.