

# Bioinformatik

Multiple Sequence Alignment

Sum-of-pairs Score

Center-Star Score

Ulf Leser

Wissensmanagement in der  
Bioinformatik



# Viterbi Algorithmus

---

- Definition

*Gegeben ein HMM  $M$  und eine Sequenz von Zeichen  $S$ . Das **Dekodierproblem** sucht nach der Folge von Zuständen, die mit der höchsten Wahrscheinlichkeit (unter allen Zustandsfolgen von  $M$ )  $S$  erzeugt hat.*

$$p^*(S | M) = \max_{p \in \text{paths}} p(S, p)$$

– Eine konkrete Zustandsfolge bezeichnen wir meistens als **Pfad**

- Naive Lösung

– Nehmen wir an, dass  $a_{ij} > 0$  und  $e_i(x) > 0$  für alle  $x, i, j$

– Dann gibt es **wie viele Pfade?**

- Es gibt  $k^n$  Pfade

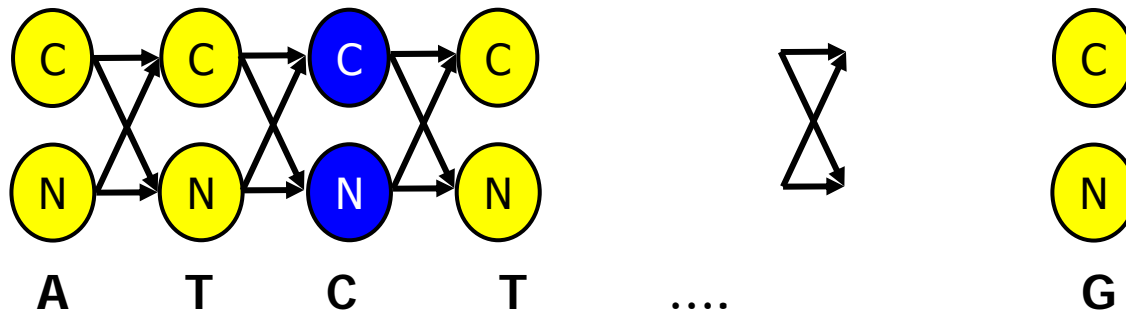
– Alle aufzählen und vergleichen ist also eine schlechte Idee

- Bessere Lösung: **Viterbi-Algorithmus**

– Viterbi, A. J. (1967). "Error bounds for convolution codes and an asymptotically optimal decoding algorithm." *IEEE Transactions on Information Theory* **IT-13**: 260-269.

# Viterbi: Dynamische Programmierung

- Dynamische Programmierung
  - Wir berechnen **optimale Pfade** für länger werdende Präfixe von  $S$ , die in einem der Zustände von  $M$  enden
  - Annahme: Sei  $v_s(i)$  die Wahrscheinlichkeit des optimalen Pfad für  $S[..i]$ , der in Zustand  $s$  endet
  - Wir brauchen eine **Rekursionsformel** für  $v_s(i+1)$  (für alle  $s \in M$ )
    - ... und die Randbedingungen
  - Die Rekursion wird dann wieder bottom-up, per Tabelle, berechnet



Sequenz:

A

T

C

T

....

G

# Rekursion

---

- Annahme: Sei  $v_s(i)$  die Wahrscheinlichkeit des optimalen Pfad für  $S[..i]$ , der in Zustand  $s$  endet
- Wenn **wir zu Zustand  $t$  übergehen**
  - Von Zustand  $s$  gehen wir mit Wahrscheinlichkeit  $a_{st}=v(t|s)$  nach  $t$
  - Dann emittieren wir das Zeichen  $S[i+1]$  mit Emissionswahrscheinlichkeit  $e_t(S[i+1])$
- Zusammen

$$v_t(i+1) = e_t(S[i+1]) * \max_{s \in M} (v_s(i) * a_{st})$$

# Tabellarische Darstellung

		A	C	T	G	...
S <sub>0</sub>	1	0	0	0	0	0
A+	0	0,20	0	0	0	
T+	0	0	0	0,010	0	0
G+	0	0	0	0	0,003	0
C+	0	0	0,054	0	0	0
A-	0	0,25	0	0	0	
T-	0	0	0	0,015	0	0
G-	0	0	0	0	0,003	0
C-	0	0	0,051	0	0	0

- Gesuchte Wahrscheinlichkeit ist die **höchste Zahl am rechten Rand**
- Optimaler **Pfad über Traceback**
  - Wieder Pointer während der Berechnung merken
- Haben alle Tabellen für HMM so viele Nullen?



# Evaluation

---

- Viterbi berechnet die wahrscheinlichste Zustandsfolge für ein gegebenes HMM
- Aber wie sicher kann man sein, dass man **das richtige HMM hat?**
- Definition  
*Gegeben ein HMM  $M$  und eine Sequenz von Zeichen  $S$ . Das **Evaluationsproblem** sucht nach der Gesamtwahrscheinlichkeit, dass  $S$  (von einer Folge von Zuständen) von  $M$  erzeugt wurde.*

# Änderung

- Viterbi

- Sei  $v_s(i)$  die Wsk des optimalen Pfad für  $S[..i]$ , der in Zustand  $s$  endet
- Gesucht:  $v_t(i+1)$  für alle  $t$ 
  - Von  $s$  gehen wir mit  $a_{st}$  nach  $t$
  - Dann Emission von  $S[i+1]$  mit  $e_t(S[i+1])$

- Wir suchen den Pfad mit der höchsten Wsk

$$v_t(i+1) = e_t(S[i+1]) * \max_{s \in M} (v_s(i) * a_{st})$$

- Forward-Algorithmus

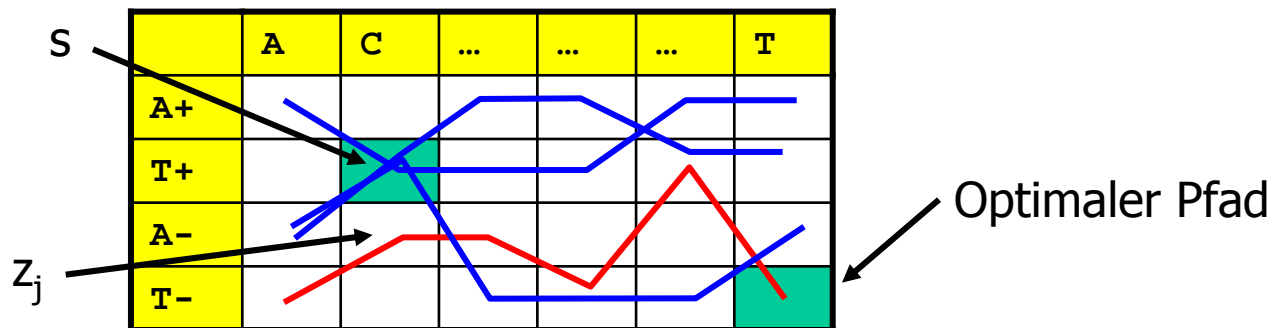
- Sei  $f_s(i)$  die Gesamtwsk, dass nach  $i$  Schritten der Zustand  $s$  erreicht ist
  - Egal, über welchen Pfad
- Gesucht:  $f_t(i+1)$  für alle  $t$ 
  - Von  $s$  gehen wir mit  $a_{st}$  nach  $t$
  - Dann Emission von  $S[i+1]$  mit  $e_t(S[i+1])$

- Gesamtwahrscheinlichkeit

$$f_t(i+1) = e_t(S[i+1]) * \sum_{s \in M} (f_s(i) * a_{st})$$

# Posterior Betrachtungen

- Betrachten wir eine einzelne Emission  $S[i]$
- Welcher Zust.  $s$  hat am wahrscheinlichsten  $S[i]$  emittiert?
  - Das ist nicht einfach das höchste  $v_s(i)$ 
    - Das würde den Teil der Sequenz nach  $i$  ignorieren
  - $s$  muss nicht auf dem wahrscheinlichsten Einzelpfad liegen
    - Nehmen wir an, der optimale Pfad laufe durch ein  $z_j \neq s$
    - Viele andere Pfade laufen durch  $s$
    - Deren Gesamtwsk kann höher sein als die des besten Einzelpfads



# Berechnung

---

- Wir benötigen  $p'(S[i] | M) = \max_{s \in M} (p(z_i = s | S))$ 
  - Diesen Term nennt man **Posteriori-Wsk**, da man ihn erst nach Betrachtung der gesamten Sequenz berechnen kann
  - In gewissem Sinne ist das der beste Tipp für  $S[i]$
- Die Berechnung erfolgt indirekt
  - Es gilt (für alle  $s$ ) 
$$p(z_i = s | S) = \frac{p(z_i = s, S)}{p(S)}$$
  - $p(S)$  kann, wie gehabt, mit Vorwärtsalgorithmus berechnet werden
  - Für den Zähler gilt
$$\begin{aligned} p(z_i = s, S) &= p(z_i = s, S[1..i]) * p(S[i+1..] | S[1..i], z_i = s) \\ &= p(z_i = s, S[1..i]) * p(S[i+1..] | z_i = s) \end{aligned}$$
- Letzteres folgt weil wir nur Markov-Ketten erster Ordnung betrachten

# Backward Algorithmus

---

$$\begin{aligned} p(z_i = s, S) &= p(z_i = s, S[1..i]) * p(S[i+1..n] | z_i = s) \\ &= f_s(i) * b_s(i) \end{aligned}$$

- $f_s(i)$  kennen wir: Forward-Algorithmus
- $b_s(i)$  heißt die **Backward-Wsk**
  - Wsk für die Restsequenz  $S[i+1..n]$ , gegeben den Startzustand  $s$
- Man berechnet sie durch eine **Rekursion von rückwärts**

$$b_t(i) = \sum_{s \in M} a_{ts} * e_s(S[i]) * b_s(i+1)$$

- Erklärung: In Schritt  $i+1$  kennen wir  $b_s(i+1)$ , die Wahrscheinlichkeit der Restsequenz  $S[i+1..n]$  für alle Startzustände  $s$  ( $=z_{i+1}$ )
- Berechnung  $b_t[i]$ : Über alle mgl. Nachfolgerzustände  $s$ ; gehe von  $t$  nach  $s$ , emittiere das Zeichen  $S[i+1]$  und fahre fort mit  $b_s(i+1)$
- Zur **Initialisierung** definiert man einen virtuellen Endzustand und gibt ihm Wahrscheinlichkeit 1 (d.h.  $a_{s,n+1}=1$  für alle  $s$ )

# Lernen eines HMM

---

- Wir können nun
  - Berechnen, wie wahrscheinlich eine Sequenz ist
    - Forward oder Backward Algorithmus
  - Berechnen, was der wahrscheinlichste Pfad für eine Sequenz ist
    - Viterbi Algorithmus
  - Berechnen, welches der wahrscheinlichste Zustand für jedes einzelne Zeichen war
    - Posterior Forward/Backward Algorithmus
  - ... **gegeben ein festes Hidden Markov Model**
- Jetzt: **Lernen der Parameter** eines HMM aus Beispieldaten
  - Komponente 1: Lernen der Struktur: Wird nicht behandelt
  - **Komponente 2: Lernen der Emission-/ Übergangs-Wsk**, gegeben eine feste Struktur

# Maximum Likelihood Schätzer für HMM

---

- Man zählt einfach

- Die relative **Häufigkeit aller Zustandsübergänge** für alle Paare von Zuständen  $s$  und  $t$

- Sei  $A_{st}$  die Zahl der Übergänge  $s \rightarrow t$
- Dann

$$a_{st} = p(t | s) = \frac{A_{st}}{\sum_{t' \in M} A_{st'}}$$

- Die relative **Häufigkeit aller Emission**  $e_s(x)$  für alle Zustände  $s$  und Zeichen  $x$

- Sei  $E_s(x)$  die Häufigkeit, mit der Zustand  $s$  Zeichen  $x$  emittiert
- Dann

$$e_s(x) = \frac{E_s(x)}{\sum_{x' \in \Sigma} E_s(x')}$$

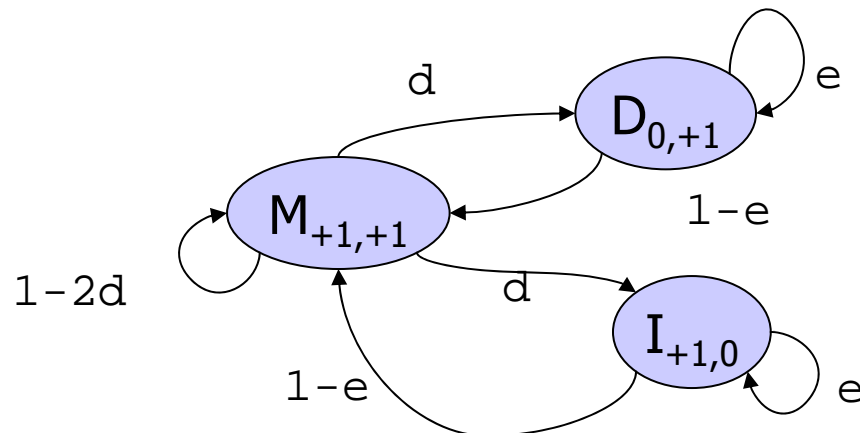
# Lernen bei unbekannter Zustandssequenz

---

- Definition  
*Gegeben ein HMM  $M$  mit fester Struktur und eine Sequenz  $S$ . Das **Lernproblem** findet die Übergangs-Wsk  $a_{st}$  und Emissions-Wsk  $e_s(x)$  von  $M$  so, dass  $p(S|M)$  maximal ist.*
- Zweiter Fall: Wir **kennen die Zustandssequenz nicht**
  - Das ist **viel schwieriger**
  - Es ist keine geschlossene Lösung bekannt
  - Prinzipiell kann man **viele Suchheuristiken** anwenden
    - Wähle eine Startkonfiguration
    - Bringe viele kleine/große Veränderungen an und betrachte die Veränderung von  $p(S|M)$
    - Übernimm die Veränderungen, die sich positiv auswirken
    - Wiederhole solange, bis ... (keine/ nur noch sehr kleine Verbesserung erzielt wird)
  - Findet immer nur **lokale Optima**

# Anwendung von HMMs für das Sequenzalignment

- Wir alignieren Sequenzen S und T
- Wir machen aus dem Automaten ein **Paar-HMM**
  - Die Zustände sind M, I, D
  - Zustand M **emittiert ein Basenpaar**  $S[i]T[i]$  an Position i des Alignments mit Wahrscheinlichkeit  $e_M(S[i]T[i])$
  - Zustand I, D emittiert eine Insertion/Deletion an Position i mit Wsk  $q(S[i])$  bzw.  $q(T[i])$
  - Übergangs-Wsk:  $M \rightarrow I: d$ ,  $M \rightarrow D: d$ ,  $I \rightarrow I: e$ ,  $D \rightarrow D: e$



Emissionswahrscheinlichkeiten  
nicht gezeigt

# Vorteile

---

- **Probabilistisches Modell**
  - Scores erhalten eine klare Interpretation
  - Wir finden das **wahrscheinlichste Alignment**
  - Wir haben ein direktes **Maß für seine Güte**
    - Nach geeigneter Normierung über Sequenzlänge
- Das optimale Alignment entspricht der optimalen Zustandsfolge durch das Paar-HMM
  - Also Viterbi
- Wir können über den **Forward-Algorithmus auch die Gesamtwsk berechnen**, dass sich die Sequenzen S und T „ähnlich“ unter M sind
  - Damit berechnen wir implizit einen Score über **alle möglichen Alignments**
  - Gerade wenn zwei Sequenzen nicht allzu gut alignieren, gibt es meistens viele ähnlich gute (schlechte) Alignments
  - Dann können wir nun immer noch eine Aussage darüber machen, mit welcher Wsk die beiden Sequenzen „ähnlich“ sind (gegeben M)

# Inhalt dieser Vorlesung

---

- Multiples Sequenzalignment
- Motivation
- Sum-Of-Pair Zielfunktion
- Center-Star Zielfunktion

# Definition

---

- Bisher
  - Vergleich zweier Strings (bzw. String und Datenbank)
- Jetzt
  - Multipler Stringvergleich: Vergleich von  $k > 2$  Strings
- Definition
  - *Ein **multiple Sequenzalignment (MSA)** von  $k$  Strings  $S_i$ ,  $1 \leq i \leq k$ , ist eine Tabelle mit  $k$  Zeilen und  $l$  Spalten, so dass*
    - *In Zeile  $i$  steht String  $S_i$ , mit beliebig eingefügten Leerzeichen*
    - *Jedes Zeichen jedes  $S_i$  steht in exakt einer Spalte*
    - *In keiner Spalte stehen nur Leerzeichen*
- Bemerkungen
  - Direkte Generalisierung des Alignment zweier Strings
  - Es folgt, dass  $l = |\text{MSA}| \leq \sum(|S_i|)$ 
    - Warum?



# Motivation

---

- Alignment sucht **ähnliche Sequenzen** unter vielen Sequenzen
  - Weil: ähnliche Sequenz – ähnliche Struktur – ähnliche Funktion
- MSA sucht „**das Ähnliche**“ in vielen Sequenzen
  - Argumentationsrichtung ist umgekehrt
  - Auch beim paarweisen Alignment findet man Regionen guter Übereinstimmung – aber nur für diese zwei Sequenzen
  - MSA startet mit vielen Sequenzen, bei denen man ähnliche Funktion / Struktur vermutet
  - MSA stellt fest, was das Gemeinsame dieser Sequenzen ist – Domänen, Motive, Signaturen, Profile, ...
  - These: dieses Gemeinsame ist **biologisch relevant**

# Motivation II

---

- Proteine (und damit auch DNA) setzen sich aus **funktionalen Blöcken** und „Zwischenraum“ zusammen
  - Die Blöcke findet man nicht, wenn man Sequenzen nur paarweise vergleicht
    - Bzw. man kann sie nicht vom Rauschen unterscheiden
- Trennung des eventuell zufällig Gemeinsamen (Alignment) vom bedeutungsvoll Gemeinsamen (MSA)

```
AAC_ GTG_ AT_ T_ GAC_  
_TCGAGTGC_ TTTACA_ GT
```

```
AAC_ GTG_ AT_ T_ GAC_  
_TCGAGTGC_ TTTACA_ GT  
GCCG_ TGC_ TA_ GTCG_  
TTC_ AGTGGACGTG_ GTA  
G_ GTGCA_ TGACC_
```

# Konservierte Domänen

---

- Gedankengang
  - Gegeben: Proteine  $S_1, \dots, S_k$  mit ähnlicher Funktion
    - Z.B.: Können durch die Zellwand tunneln, an DNA binden, bestimmte Proteine aktivieren, etc.
  - Annahme: identischer evolutionärer Ursprung
    - Es gab einmal das „Mutterprotein“  $S$
  - $S$  unterliegt Evolution
    - Mutation, Rekombination und Selektion
  - Abschnitte in  $S_i$ , die trotz Evolution gleich blieben (konserviert sind), müssen für die gemeinsame Funktion wichtig sein
    - Andere Abschnitte dagegen sind nicht oder weniger wichtig
- Also: Um funktionale Blöcke zu finden, muss man viele Sequenzen vergleichen
  - Sequenzieren bleibt wichtig

# Blöcke, Domänen, Sites

---

- Proteine
  - Bindungsstellen für andere Proteine / Liganden / Moleküle
  - Bindungsstellen an DNA
  - Signale zur Phosphorylierung / Dephosphorylierung
  - Signale zum Transport des Proteins
  - Signale zum Abbau des Proteins
  - ...
- DNA
  - Bindungsstellen für Proteine
  - Promotoren und Inhibitoren
  - Start- und Stoppcodons
  - Signal für differenzielles Splicen
  - ...

# Proteinfamilien

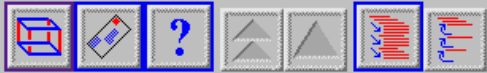
---

- Man unterteilt Proteine in Familien, Superfamilien, ...
  - Diverse Klassifikationen vorhanden (CATH, SCOP, ...)
- Idee
  - X00.000 Proteine zerfallen in X.000 Klassen ähnlicher Funktion?, Struktur?, Substruktur?
  - Untersuchung von **Vertretern von Familien** statt aller Proteine
  - Finden familienspezifischer Domänen
  - Benutzung familienspezifischer Substitutionsmatrizen
- Finden / Definieren von Proteinfamilien
  - Starte mit Proteinen gleicher/ähnlicher Funktion
  - **Finde das Gemeinsame durch MSA**
  - Suche nur mit konservierten Blöcken nach weiteren Vertretern
  - Modifiziere Familie entsprechend
  - Iteriere, bis Zufriedenheit eintritt

# Beispiel: SCOP

- Structural Classification of Proteins
- Hierarchische Anordnung
  - *Fold*: Major structural similarity
    - All Alpha, All Beta, Membrane proteins, ...
  - *Superfamily*: Probable common evolutionary origin
    - Nucleotide-binding domain, Neurotransmitter-gated ion-channel transmembrane pore, ...
  - *Family*: Clear evolutionarily relationship
    - Globins, Death Domain, 4 families of Immunoglobulin ...
  - Protein
  - Spezies

Structural Classification of Proteins



**Root: scop**

**Classes:**

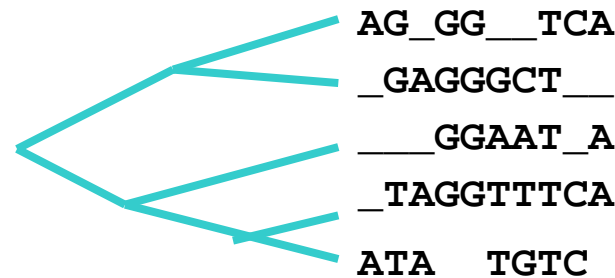
1. [All alpha proteins](#) [46456] (179)
2. [All beta proteins](#) [48724] (126)
3. [Alpha and beta proteins \(a/b\)](#) [51349] (121)   
*Mainly parallel beta sheets (beta-alpha-beta units)*
4. [Alpha and beta proteins \(a+b\)](#) [53931] (234)   
*Mainly antiparallel beta sheets (segregated alpha and beta regions)*
5. [Multi-domain proteins \(alpha and beta\)](#) [56572] (38)   
*Folds consisting of two or more domains belonging to different classes*
6. [Membrane and cell surface proteins and peptides](#) [56835] (36)   
*Does not include proteins in the immune system*
7. [Small proteins](#) [56992] (66)   
*Usually dominated by metal ligand, heme, and/or disulfide bridges*
8. [Coiled coil proteins](#) [57942] (6)   
*Not a true class*
9. [Low resolution protein structures](#) [58117] (18)   
*Not a true class*
10. [Peptides](#) [58231] (105)   
*Peptides and fragments. Not a true class*
11. [Designed proteins](#) [58788] (39)   
*Experimental structures of proteins with essentially non-natural sequences. Not a true class*

Enter [search](#) key:

# MSA Zielfunktion

---

- **Zielfunktion** beim einfachen Alignment war klar
  - Möglichst wenig I,R,D
  - Eventuell mit Substitutionsmatrix
  - Eventuell mit spezieller Behandlung von Gaps
- Zielfunktion für MSA ist nicht so klar
  - Score einer Spalte mit 2 T, zwei G und einem Leerzeichen?
  - Angabe einer Substitutionsmatrix für k Sequenzen über Alphabet  $\Sigma$  würde  $O(|\Sigma|^{k+1})$  Werte erfordern
  - Nicht machbar und biologisch nicht begründbar



# MSA Überblick

---

- Weg über Substitutionsmatrizen nicht gangbar
- Verschiedene alternative Vorschläge für Zielfunktionen existieren
  - Maximiere die Summe aller paarweisen Alignments
  - Maximiere die Summe der Alignments jeder Sequenz zu einer Consensussequenz (Center-Star)
  - Maximiere die Summe der Alignments folgend dem phylogenetischen Baum der Sequenzen

# Inhalt dieser Vorlesung

---

- Multiples Sequenzalignment
- Motivation
- Sum-Of-Pair Zielfunktion
- Center-Star Zielfunktion

# Definitionen

---

- Definition

- Gegeben ein MSA  $M$  für Sequenzen  $S_1, \dots, S_k$ . Das *durch  $M$  induziertes Alignment für zwei Sequenzen  $S_i$  und  $S_j$*  ist das folgende:
  - Entferne aus  $M$  alle Zeilen außer  $i$  und  $j$
  - Entferne alle Spalten, die in  $i$  und  $j$  ein Leerzeichen enthalten
- Gegeben ein MSA  $M$  für Sequenzen  $S_1, \dots, S_k$ . Der *Sum-Of-Pairs Score für  $M$  (SP-Score)* ist die Summe aller Alignmentsscores der durch  $M$  induzierten paarweisen Alignments
- Das *SP-Alignment Problem für Sequenzen  $S_1, \dots, S_k$*  sucht das MSA  $M$  mit minimalem SP-Score

- Bemerkung

- Vergleich aller Sequenzen mit allen anderen Sequenzen – aber entsprechend dem vorgegebenen MSA

# Beispiel

d/i	=	1
r	=	1
m	=	0

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{AT\_AATG} \\ \text{CTG\_G\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 5 \quad \} 14$$

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{\_ATAATG} \\ \text{C\_TGG\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 6 \quad \} 15$$

- Die Berechnung des SP-Scores für ein gegebenes MSA über k Sequenzen ist einfach
  - Komplexität?

# Beispiel

d/i	=	1
r	=	1
m	=	0

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{AT\_AATG} \\ \text{CTG\_G\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 5 \quad \} 14$$

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{\_ATAATG} \\ \text{C\_TGG\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 7 \quad \} 16$$

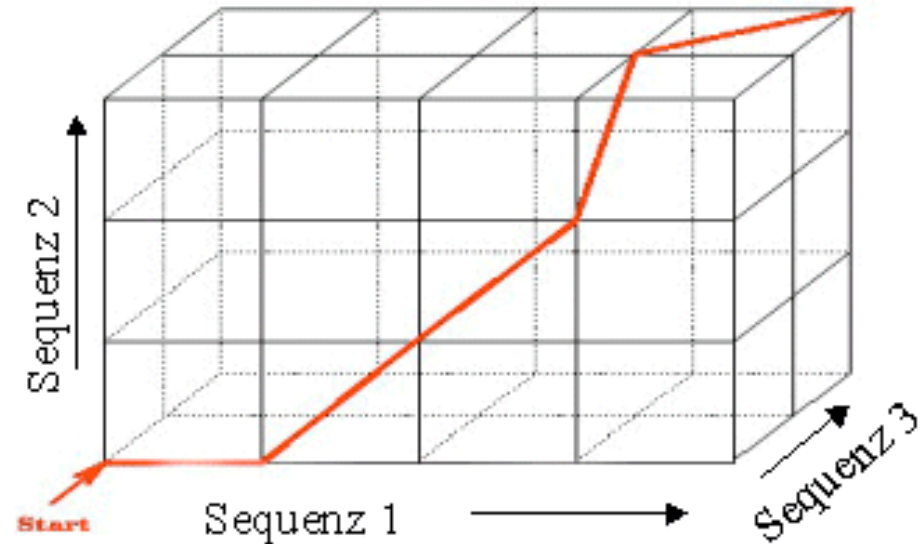
- Die Berechnung des SP-Scores für ein gegebenes MSA über  $k$  Sequenzen ist einfach
  - Komplexität  $O(k^2 \cdot l)$
- Aber wie findet man das MSA mit **minimalem SP-Score**?

# Dynamische Programmierung in k Dimensionen

- $k = 2$ 
  - 2-dimensionale Matrix

	0	1	2	3	4	5	6	7
		w	r	i	t	e	r	s
0	0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6
2	i	2	2	2	2	3	4	5
3	n	3	3	3	3	3	4	5
4	t	4	4	4	4	3	4	5
5	n	5	5	5	5	4	4	5
6	e	6	6	6	6	5	4	5
7	r	7	7	6	7	6	5	4

- $k = 3$ 
  - 3-dimensionale Matrix



# Erinnerung

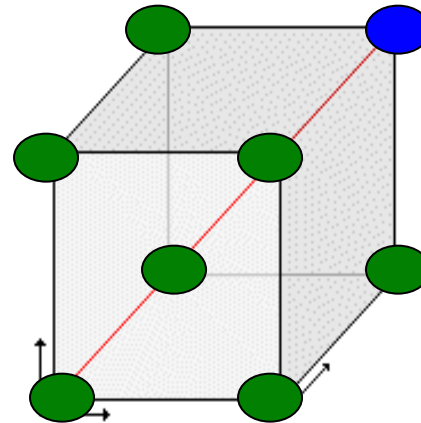
---

- Grundidee der dynamischen Programmierung für zwei Sequenzen  $S_1, S_2$ 
  - Berechnung des Alignment  $d(i,j)$  von  $S_1[1..i]$  und  $S_2[1..j]$  für steigende Werte  $(i, j)$  bis  $i=|S_1|$  und  $j=|S_2|$
  - Berechnung von  $d(i,j)$  aus  $d(i-1,j-1), d(i,j-1), d(i-1,j)$ 
    - Man verlängert  $d(i-1,j-1)$  um Match oder Mismatch
    - ... oder man verlängert  $d(i,j-1)$  um ein Insert
    - ... oder man verlängert  $d(i-1,j)$  um eine Deletion
  - Statische Initialisierung der Werte  $d(i,0)$  und  $d(0,j)$
- Wir betrachten im Folgenden nur den Fall  $k=3$

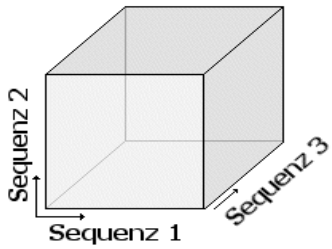
# Dynamische Programmierung für SP-MSA

---

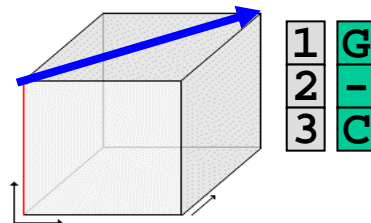
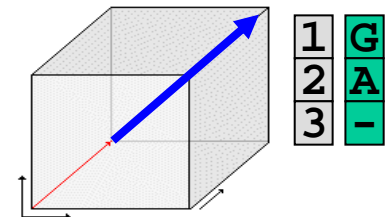
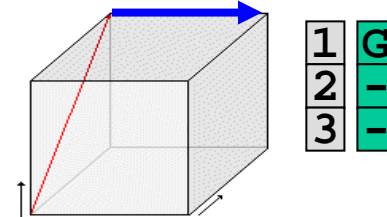
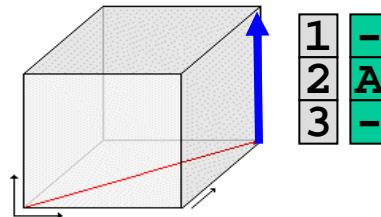
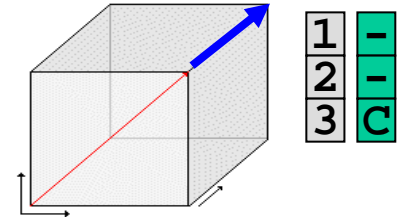
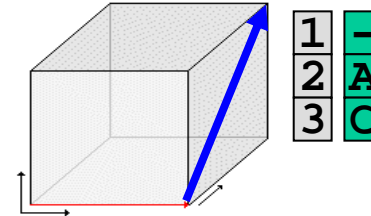
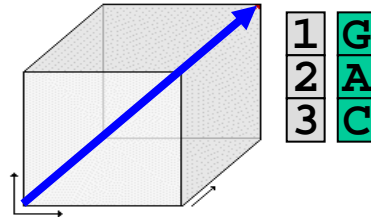
- Übertragung auf MSA
- Berechnung von  $d(i,j,k)$  aus
  - $d(i-1,j-1,k-1)$
  - $d(i,j-1,k-1)$
  - $d(i,j,k-1)$
  - $d(i,j-1,k)$
  - $d(i-1,j,k)$
  - $d(i-1,j-1,k)$
  - $d(i-1,j,k-1)$

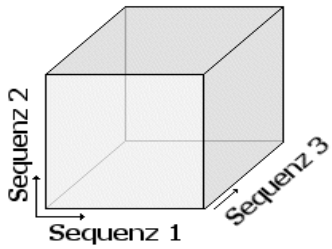


# Dyn Prog. für SP-MSA



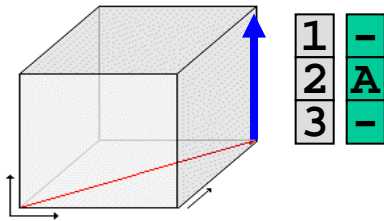
- $d(i-1, j-1, k-1)$
- $d(i, j-1, k-1)$
- $d(i, j, k-1)$
- $d(i, j-1, k)$
- $d(i-1, j, k)$
- $d(i-1, j-1, k)$
- $d(i-1, j, k-1)$



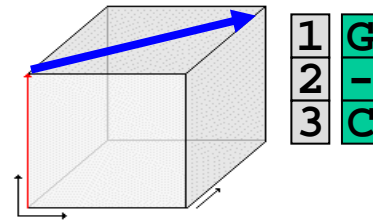


# Analogie

$d(i, j-1, k)$



$d(i-1, j, k-1)$



- SP-Alignment von  $d(i, j-1, k)$  ist bekannt
- Wir erweitern dieses zu  $d(i, j, k)$
- Dazu alignieren wir  $S_2[j]$  zweimal mit Leerzeichen (Inserts)

- SP-Alignment von  $d(i-1, j, k-1)$  ist bekannt
- Wir erweitern dieses zu  $d(i, j, k)$
- Dazu alignieren wir ein Leerzeichen mit  $S_1[i-1]$  und mit  $S_3[k-1]$

# Formal

- Wir nehmen ein einfaches Kostenmodell ( $I/D/R=1, M=0$ )
- Theorem
  - Gegeben Sequenzen  $S_1, S_2, S_3$ .
    - Sei  $d(i,j,k)$  der Score des SP-optimalen Alignments der Strings  $S_1[1..i], S_2[1..j], S_3[1..k]$
    - Sei  $c_{ij} = 0$ , wenn  $S_1[i] = S_2[j]$ , sonst 1
    - Sei  $c_{ik} = 0$ , wenn  $S_1[i] = S_3[k]$ , sonst 1
    - Sei  $c_{jk} = 0$ , wenn  $S_2[j] = S_3[k]$ , sonst 1
  - Dann berechnet sich  $d(i,j,k)$  als

$$d(i, j, k) = \min \left\{ \begin{array}{llll} d(i-1, j-1, k-1) + c_{ij} & + c_{ik} & & + c_{jk} \\ d(i-1, j-1, k) & + c_{ij} & + 2 & \\ d(i-1, j, k-1) & + c_{ik} & + 2 & \\ d(i, j-1, k-1) & + c_{jk} & + 2 & \\ d(i-1, j, k) & & + 2 & \\ d(i, j-1, k) & & + 2 & \\ d(i, j, k-1) & & + 2 & \end{array} \right.$$

# Randbedingungen

---

- Theorem Fortsetzung

- ...

- mit Initialisierung

- Sei  $D_{a,b}(i,j)$  der optimale Alignment score von  $S_a[1..i]$  mit  $S_b[1..j]$
    - $D(0, 0, 0) = 0$
    - $D(i, j, 0) = D_{1,2}(i, j) + (i+j)$
    - $D(i, 0, k) = D_{1,3}(i, k) + (i+k)$
    - $D(0, j, k) = D_{2,3}(j, k) + (j+k)$

- Bemerkung

- Beweis analog zum paarweisen Alignment
  - Alignment eines Leerzeichen mit einem Leerzeichen ist im induzierten paarweisen Alignment nicht enthalten

# Komplexität

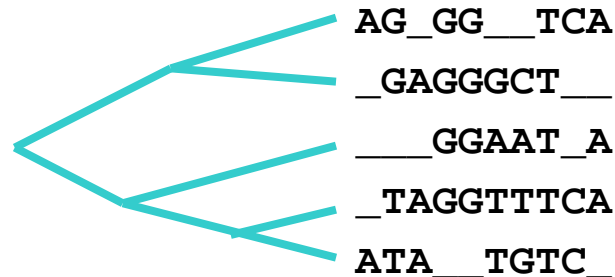
---

- ?
- Für drei Sequenzen der Länge  $n$ 
  - Würfel hat  $n^3$  Zellen
  - Für jede Zelle sind 7 Berechnungen notwendig
  - Zusammen  $O(7*n^3)$
- Allgemeiner Fall:  $k$  Sequenzen der Länge  $n$ 
  - Hyperwürfel hat  $n^k$  Zellen
  - Für jede Zelle sind  $2^k-1$  Berechnungen notwendig
    - Alle Ecken eines  $k$ -dimensionalen Würfels minus eins (Das ist die Ecke die gerade berechnet wird)
  - Zusammen  $O(2^k * n^k)$
- Tatsächlich gilt
  - *Das SP-Alignment Problem ist NP vollständig*

# MSA also praktisch unlösbar?

---

- SP-Score für mehr als eine Handvoll Sequenzen nennenswerter Länge nicht berechenbar
- Aber
  - SP berechnet **überhaupt nicht die minimale Menge an Evolution**



- SP ist nur eine mögliche Zielfunktion
- Andere: Center-Star, MSA entlang des phylogenetischen Baums
- Viele Heuristiken (Branch&Bound, iterative, lokal-Greedy, ...)

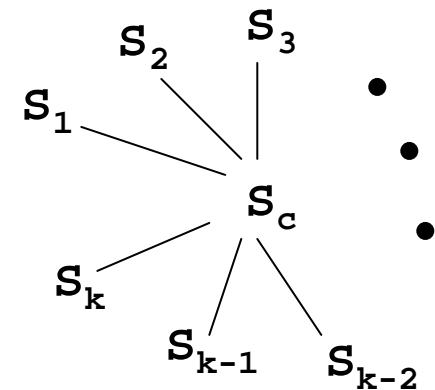
# Inhalt dieser Vorlesung

---

- Multiples Sequenzalignment
- Motivation
- Sum-Of-Pair Zielfunktion
- Center-Star Zielfunktion

# MSA mit Konsensussequenz

- SP minimiert die Summe aller paarweisen Alignments
- Alternativ: Minimierung der **Summe der Alignments aller Sequenzen  $S_1, \dots, S_k$  mit einer Konsensussequenz  $S_c$** 
  - $S_c$  kann eine der  $S_i$  sein, muss aber nicht
  - Konstruktion von  $S_c$  z.B. durch Untereinanderschreiben der  $S_i$  ohne Gaps und Wahl des häufigsten Buchstaben
  - MSA wird aus dem „Star“ abgeleitet
- Bei geeigneter Wahl von  $S_c$  und dem MSA – Ableitungsverfahren gilt:
  - Der SP Score des berechneten MSA ist **höchstens doppelt so hoch wie der SP-optimale**
  - Center-Star-Score approximiert den SP-Score bis auf Faktor 2



# Center-Star Verfahren

---

- Gegeben  $k$  Sequenzen der Länge  $n$
- Wähle als Konsensus die Sequenz  $S_i$ , die den **kleinsten durchschnittlichen Abstand** zu allen Sequenzen hat
  - Wie komplex ist das?
- Benutze  $S_i$  als Kern eines MSA  $M$ 
  - Bzw. als Zentrum des Stars
- Iteriere
  - Wähle eine noch nicht alignierte Sequenz  $T$
  - **Aligniere  $M$  und  $T$** 
    - Mit einer einfachen Methode, die wir nicht ausführen (Gusfield, p. 348)
  - Bis alle Sequenzen in  $M$  enthalten sind
- Beispiel für ein **progressives MSA** Verfahren
  - Sukzessive Sequenzen zu einem wachsenden MSA hinzufügen

# Beispiel

---

1. ATGGC
2. AGCC
3. TGCGAT
4. GCATG
5. TGCCTA
6. CAACTA

	S1	S2	S3	S4	S5	S6
S1	0	2	4	4	4	5
S2	2	0	4	4	3	4
S3	4	4	0	3	3	5
S4	4	4	3	0	3	4
S5	4	3	3	3	0	3
S6	5	4	5	4	3	0
Durchschnitt	3,8	3,4	3,8	3,6	3,2	4,2

Quelle: Martin Filip, Proseminar, 2005

# Beispiel 2

---

- Kern des „MSA“: **TGCCTA**
- Wähle Sequenz:  $S_3 =$  **TGCGAT**
- Alignment: 

```
TGCC_TA
TGCGAT_
```
- Wähle Sequenz:  $S_2 =$  **AGCC**
- Alignment: 

```
TGCC_TA
TGCGAT_
AGCC_____
```
- Wähle Sequenz:  $S_1 =$  **ATGGC**
- Alignment: 

```
_TGCC_TA
_TGCGAT_
_AGCC_____
ATGGC_____
```

# Beispiel 3

---

- Wähle Sequenz: S4= **GCATG**

- Alignment:

```
_TGCC_TA
_TGCGAT_
_AGCC___
ATGGC___
__GC_ATG
```

- Wähle Sequenz: S6= **CAACTA**

- Alignment:

```
_TGCC_TA
_TGCGAT_
_AGCC___
ATGGC___
__GC_ATG
CAAC__TA
```

# Fehlergrenze

---

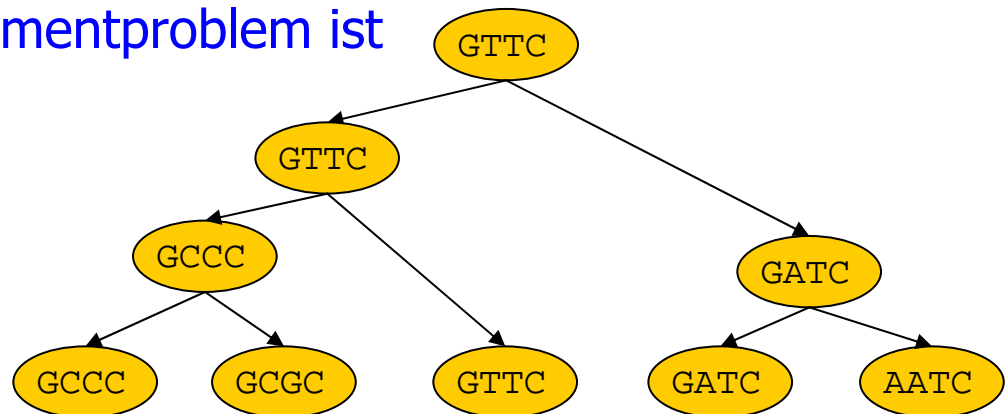
- Wenn für Zeichen  $i, j, k$  gilt:  $d(i, j) + d(j, k) \geq d(i, k)$ 
  - Dreiecksungleichung
  - Gilt leider **nicht für biologisch relevante Substitutionsmatrizen**
- Theorem
  - *Gegeben  $k$  Sequenzen,  $k \geq 3$*
  - *Sei  $d$  der SP Score des durch den Center-Star Algorithmus berechneten MSA*
  - *Sei  $d^*$  der optimale SP Score*
  - *Dann gilt*

$$\frac{d}{d^*} \leq 2 - \frac{1}{k} < 2$$

- Beweis
  - Siehe Gusfield

# MSA mit phylogenetischen Bäumen

- Grundidee
  - Annahme: **Sequenzen durch Evolution** aus Ursequenz entstanden
  - Wir ordnen Sequenzen Knoten im Baum zu
  - Sequenzen haben sich entlang der Pfade entwickelt
- Zielfunktion
  - Suche den Baum T so, dass die **Summe aller Alignmentsscores von benachbarten Sequenzen in T minimiert** wird
  - Aus T kann man ein MSA ableiten (gleich)
- Leider
  - Das **phylogenetische Alignmentproblem ist NP vollständig**
  - Ähnlich zu Phylogenie mit Maximum Parsimony (später)



# Lokales MSA-Problem

---

- Wir haben nur das globale MSA-Problem untersucht
- Auch das lokale MSA-Problem ist definiert und relevant
  - Aber nicht so populär wie lokales Alignment von Sequenzen
  - Warum? Weil man MSA i.d.R. nur mit ähnlichen Sequenzen macht
- Definition
  - Gegeben  $k$  Sequenzen  $S_1, \dots, S_k$ . Ein *lokales multiples Sequenzalignment* erhält man, in dem man
    - Für jedes  $S_i$  wähle eine Teilzeichenkette  $s_i \in S_i$
    - Aligniere alle  $s_i$  durch globales MSA
- Ziel ist natürlich wieder solche  $s_i$  zu finden, die möglichst hohe MSA Scores erzeugen
- Beispiel: DALIGN

# Suche mit MSA

---

- Erinnerung: Erzeugung von Proteinfamilien
  - Starte mit Proteinen gleicher/ähnlicher Funktion
  - Finde das Gemeinsame durch MSA
  - Suche mit dem MSA nach weiteren Vertretern
  - Modifiziere Familie entsprechend
  - Iteriere, bis Zufriedenheit eintritt
- Wie sucht man mit einem MSA?
  - Wir müssen entscheiden, wie gut eine (neue) Sequenz  $S$  zu einem MSA  $M$  passt
  - Versch. Möglichkeiten: Profile, RegExp, Profile-HMM