

Bioinformatik

HMM Algorithmen

Viterbi

Forward-Backward

Baum-Welch

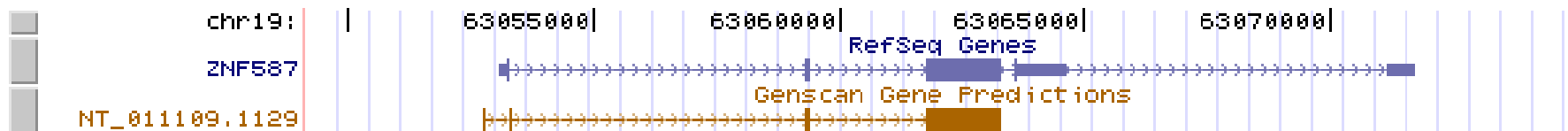
Ulf Leser

Wissensmanagement in der
Bioinformatik



Gene Prediction

- Kann man **Gene vorhersagen**?
 - Ist an der Sequenz eines Gens irgendwas anders als an „Nicht-Gen“ Sequenzen?
 - Wenn ja: Kann man die Unterschiede aus bekannten Gen-/Nichtgen-Sequenzen lernen?
 - Kann man das Gelernte zur Vorhersage eines Gens auf einer neuen Sequenz benutzen?
- **Gene Prediction**
 - Nach wie vor sehr aktives Forschungsgebiet
 - Aktuelle Verfahren benutzen alle verfügbaren Informationen
 - GRAIL, GeneWise, Gene-ID, GeneScan, ...
 - Vorhergesagte Gene werden typischerweise als „putative“ in die aktuellen **Genomannotationen** übernommen



Prokaryoten versus Eukaryoten

(B) PROCARYOTES

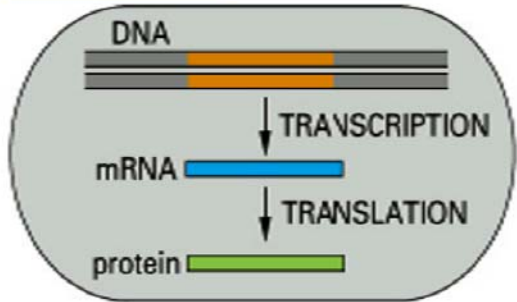


Figure 6-21 part 2 of 2. Molecular Biology of the Cell, 4th Edition.

(A) EUKARYOTES

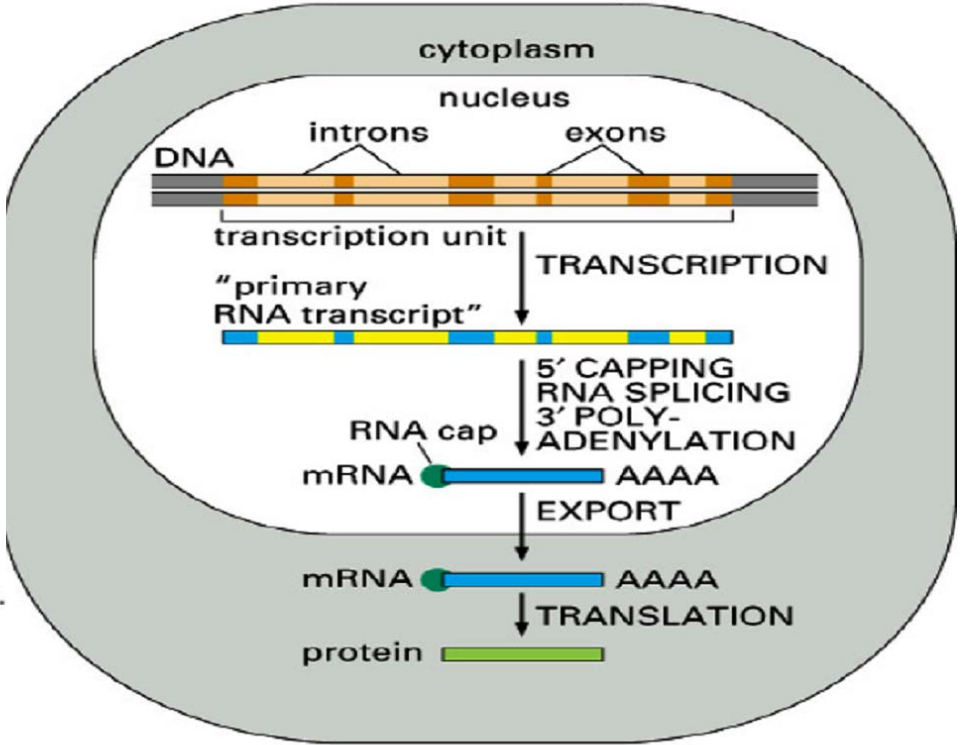
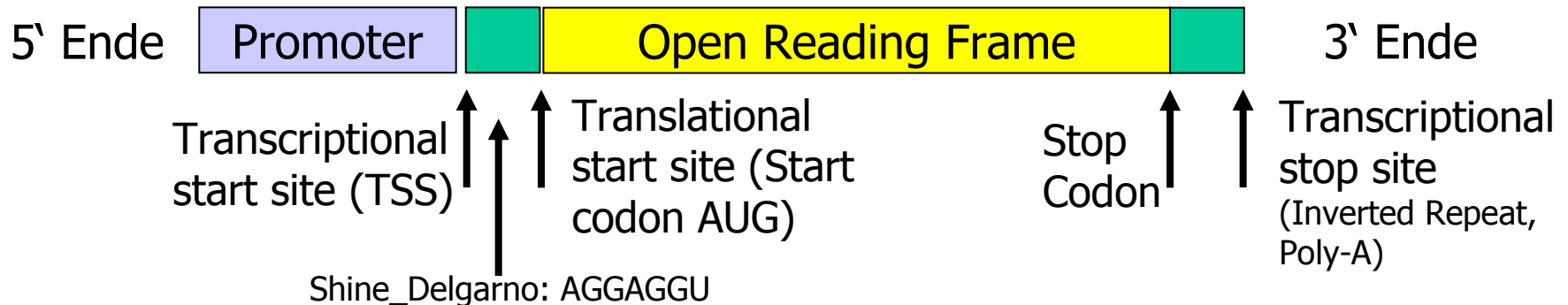


Figure 6-21 part 1 of 2. Molecular Biology of the Cell, 4th Edition.

Quelle: William Stafford Noble

Gene in Prokaryoten

- Haben eine vergleichsweise einfache Struktur
 - Relativ feste **Start- und Stopcodons**
 - **Open Reading Frame (ORF)**: Sequenz zwischen Start- und Stopcodon von mindestens 100 Basen Länge; Länge durch 3 teilbar
 - Signale für **Anfang und Ende der Transkription**
 - **Promoterregion**: Konservierte Motive im Abstand von -35 bzw. -10 Basen von der Transcriptional Start Site (TSS)



Sigma-Faktoren

Faktor	Erkennungssequenz -35	Erkennungssequenz -10	Bedingungen
σ^{70}	TTGACA	TTGACA	Normal (~70% aller Gene)
σ^{32}	CTTGAA	CTTGAAA	Hitzestress
σ^{54}	CTGGCAC	CTGGCAC	Stickstoffmangel
σ^{28}	TAAA	CTAAA	...
...

- Verschiedene Sigma-Faktoren binden an verschiedene **Sequenzmotive**
 - E.Coli hat 7 Faktoren; andere haben mehr/weniger
- Motive müssen nicht perfekt erhalten sein
 - Dargestellt sind **Consensus-Sequenzen**
 - Je größer die Abweichung, desto geringer die Expression des regulierten Gens

Gene Prediction in Prokaryoten

- Verfügbare **Evidenzen sammeln**
 - ORFs finden
 - Konservierte Promotor-Sequenzen eines Sigma-Faktors prüfen
 - In einem ORF ist die dritte Base jedes Codons häufiger gleich als statisch erwartet
 - Grund: Spezies favorisieren spezifische Codons für Aminosäuren, bei denen es mehrere Möglichkeiten gibt
 - Weitere Signale: Transcriptional Stop Site, Shine-Delgado-Sequenz, ...
- Wenn man die (fast) alle gefunden hat, hat man mit hoher Wahrscheinlichkeit ein Gen
 - Wahrscheinlichkeit eines **Falsch-Positiven Hits** für ein beliebiges ORF der Länge 60 Codons
 - 60-mal kein Stop-Codon sehen: $(61/64)^{60} \sim 4\%$

Eukaryoten – Alles viel schwieriger

(A) EUKARYOTES

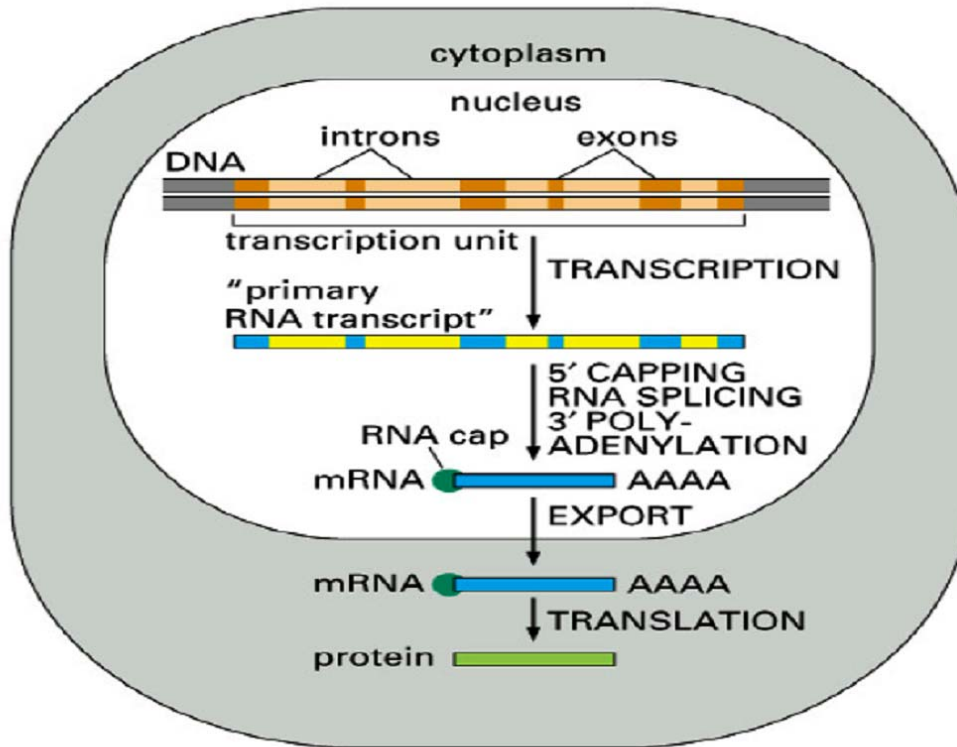


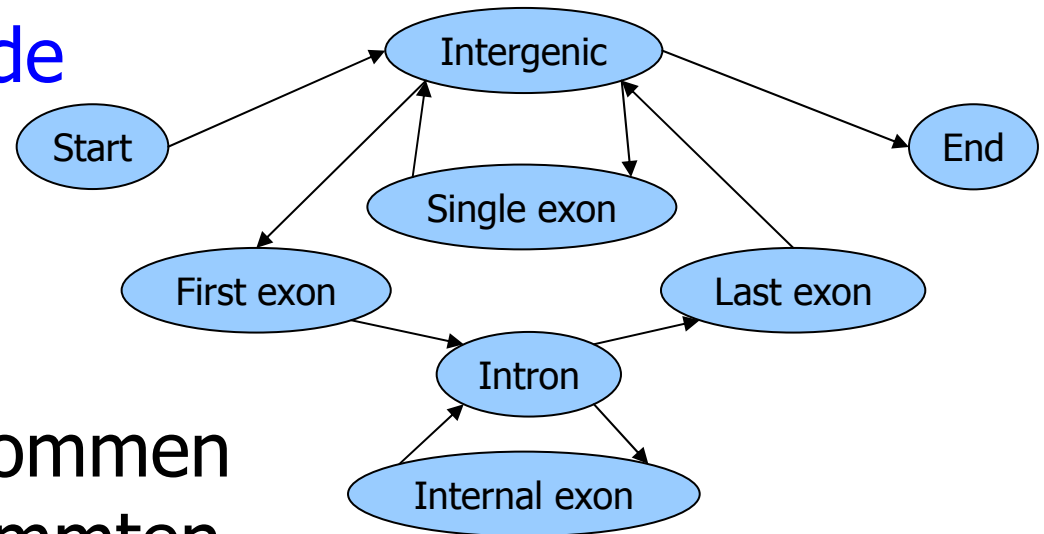
Figure 6–21 part 1 of 2. Molecular Biology of the Cell, 4th Edition.

Quelle: William Stafford Noble

- **Introns** variabler Zahl und Länge
 - können bis zu MB lang sein
- Differentielles Splicing
- 3 RNA Polymerasen
- **Promoterbereiche können mehrere MB** vom Gen entfernt sein
- Polymerase bindet nicht direkt, sondern nur bei Vorhandensein mehrerer **Transcription Factors (TF)**
 - Mensch: Ca. verschiedene 2000 TF
 - Expression benötigt im Schnitt >5 gebundene TFs
- Sehr großer Anteil nicht kodierender DNA
- ...

Module und Zustände

- Module sind **Zustände** eines Modells
- **Zustände** emittieren **Basen**
- In einem Zustand kommen Basen in einer bestimmten Frequenz, Reihenfolge und Anzahl vor
- Pfeile sind Zustandsübergänge
- Übergänge haben eine bestimmte Wsk
- Das ist ein **Hidden Markov Model (HMM)**



Probleme (informell)

- Einer gegebenen Sequenz kann man erst mal nicht ansehen, aus welchen Zuständen in welcher Reihenfolge sie am wahrscheinlichsten generiert wird
 - Alle emittieren A,C,G,T, nur mit (geringfügig) unterschiedlicher Wsk
- Problem 1: Gegeben eine Sequenz und ein Modell: **Finde die Modulgrenzen** (also die Zustandsübergänge)

ACTG	ACT	ACTAAATTGCCGCTCGT	GACGACGATCTACTAG	GGGCGCGACCTATGCG
	SSS	EEEEEEEEEEEEEEEEEEEESS	IIIIIIIIIIIIIIIIIIII	SSEEEEEEEEEEEEEEEE...

- Problem 2: Gegeben viele Gene: **Finde die Übergangs- und Emissionswahrscheinlichkeiten** des Modells
 - Und womöglich das Modell selber

CpG Inseln

- CpG-Inseln
 - Sequenzabschnitte, in denen **mehr CpG als erwartet** (bezogen auf absolute Häufigkeit im Genom) vorkommen
 - Die meisten CpG Inseln liegen vor Genen
 - Die meisten Gene liegen hinter einer CpG Insel
- Wie kann man für eine Sequenz entscheiden, ob sie eine CpG Insel ist?
 - Wir wissen, dass bestimmte Dinukleotide häufiger sind als sonst
 - Nach C kommt häufiger ein G als ein A oder T
 - Richtig fest ist aber nichts
 - Erster Versuch: **Markov-Modelle** erster Ordnung

Markov-Modell (oder Markov-Kette)

- Definition

*Gegeben ein Alphabet Σ . Ein **Markov-Modell** erster Ordnung ist ein sequentieller stochastischer Prozess (Zustandsfolge) über $|\Sigma|$ Zuständen s_1, \dots, s_n mit*

- *Jeder **Zustand s_i** emittiert genau ein Zeichen aus Σ*
- *Keine zwei Zustände emittieren das selbe Zeichen*
- *Für eine Folge z_1, z_2, \dots von Zuständen gilt:*

$$p(z_t=s_t | z_{t-1}=s_{t-1}, z_{t-2}=s_{t-2}, \dots, z_1=s_1) = p(z_t=s_t | z_{t-1}=s_{t-1})$$

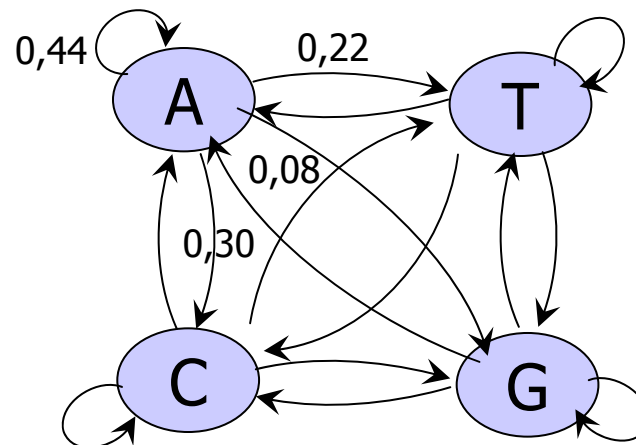
- *Die $a_{0,i} = p(z_1=s_i)$ heißen **Startwahrscheinlichkeiten***
- *Die $a_{s_i,s_j} = p(z_t=s_j | z_{t-1}=s_i)$ heißen **Übergangswahrscheinlichkeiten***

- Bemerkung

- Die Wahrscheinlichkeit des Auftretens eines Zustands hängt also **nur vom Vorgängerzustand** ab

Visualisierung

- Jeder Zustand einer Markov-Kette emittiert genau ein eindeutiges Zeichen des Alphabets
 - Daher können wir **Zustände und Zeichen verschmelzen**
 - Bei HMM geht das nicht, daher trennen wir in der Definition
- Damit können wir ein Markov-Modell als **Zustandsgraph** visualisieren
 - Knoten sind die Zeichen des Alphabets (Zustände)
 - Kanten sind mit Übergangswahrscheinlichkeiten beschriftet



Hier sind alle Zustände mit allen verbunden; das muss nicht so sein ($a_{ij}=0$)

CpG Inseln erkennen

- Erster Versuch: Wir bilden **zwei Markov-Modelle**
 - Modell M+ für die Übergangshäufigkeiten in CpG Inseln
 - Modell M- für die Übergangshäufigkeiten in normaler Sequenz
 - Berechnung des Log-Odds-Score

$$s = \log\left(\frac{p(S | M+)}{p(S | M-)}\right) = \sum_{i=1}^n \frac{a_{i-1,i}^+}{a_{i-1,i}^-}$$

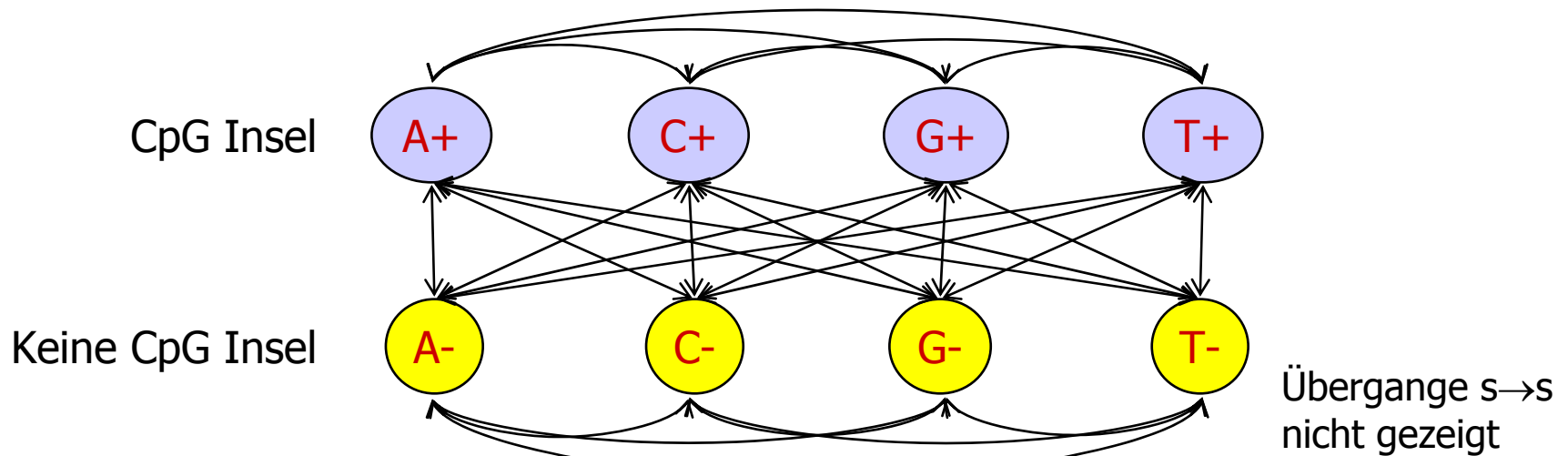
- $s > 0$: Die Sequenz ist **wahrscheinlich eine CpG Insel**
 - Je größer s , desto wahrscheinlicher
- $s < 0$: Die Sequenz ist wahrscheinlich keine CpG Insel

Lösung 2: Hidden Markov-Modelle (HMM)

- Baum, L. E. and Petrie, T. (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains." *Annals of Mathematical Statistics* **37**: 1554-1563.
- Folgende Idee
 - Wir denken uns einen Automaten mit zwei „Meta-Zuständen“: Einer für CpG, einer für nicht-CpG
 - Jeder Zustand kann alle Zeichen des Alphabets mit einer bestimmten Emissions-Wsk emittieren
 - Das ist neu: Zustände \neq Zeichen
 - Damit kann man die Zustandsfolge nicht mehr einfach aus der Zeichenfolge ablesen
 - Das ist neu: Zustände sind verborgen (hidden)
 - Die Emissions-Wsk und Übergangs-Wsk in den beiden Metazuständen sind unterschiedlich und modellieren die Unterschiede zwischen normaler DNA und CpG-Inseln

Vorgehen

- Wir verdoppeln die Anzahl der Zustände: $A \rightarrow A+, A- \dots$
- Die Zustände $s+$ und $s-$ emittieren jeweils **dasselbe Zeichen**
- Die Übergangswahrscheinlichkeiten zwischen allen $s+$ entnehmen wir dem $M+$ Modell
- Die Übergangswahrscheinlichkeiten zwischen allen $s-$ Zuständen entnehmen wir dem $M-$ Modell
- Übergang von jedem $s+$ Zustand zu jedem $s-$ Zustand und umgekehrt ist mit einer bestimmten (kleinen) Wahrscheinlichkeit erlaubt



Formale Definition von HMMs

- Definition

Gegeben Σ . Ein Hidden Markov Modell ist ein sequentieller stochastischer Prozess über k Zuständen s_1, \dots, s_k mit

- *Zustand s emittiert Zeichen $x \in \Sigma$ mit Wahrscheinlichkeit $p(x/s)$*
- *Die Folge der Zustände ist eine Markov-Kette, d.h.:*
$$p(z_t=s_t/z_{t-1}=s_{t-1}, z_{t-2}=s_{t-2}, \dots, z_0=s_0) = p(z_t=s_t/z_{t-1}=s_{t-1})=a_{t-1,t}$$
- *Die $a_{0,1}$ heißen Startwahrscheinlichkeiten*
- *Die $a_{t-1,t}$ heißen Übergangswahrscheinlichkeiten*
- *Die $e_s(x)=p(x/s)$ heißen Emissionswahrscheinlichkeiten*

- Bemerkung

- Jetzt ist die Unterscheidung zwischen Zustand und Zeichen wichtig
- Eine Sequenz kann durch viele Zustandssequenzen emittiert werden

Problemfelder

- Die drei klassischen HMM Probleme
 - **Dekodierung/Parse**: Gegeben eine Sequenz S und ein HMM M ; durch welche Zustandsfolge wurde S wahrscheinlich erzeugt?
 - Lösung: Viterbi Algorithmus
 - **Evaluation**: Gegeben eine Sequenz S und HMM M ; mit welcher Wahrscheinlichkeit wurde S durch M erzeugt?
 - Muss alle Zustandssequenzen berücksichtigen, die S erzeugen
 - Lösung: Forward/Backward Algorithmus
 - **Lernen/Trainieren**: Gegeben eine Sequenz S ; welche HMM M (mit gegebener Zustandsmenge) erzeugt S mit der größten Wahrscheinlichkeit?
 - Lernen der Übergangs- und Emmissionswahrscheinlichkeiten aus S
 - Lösung: Baum-Welch Algorithmus

Inhalt der Vorlesung

- Decodierung: Viterbi-Algorithmus
- Evaluation: Forward-Backward Algorithmus
- Parameterschätzung: Baum-Welch
- HMM für Sequenzalignment

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

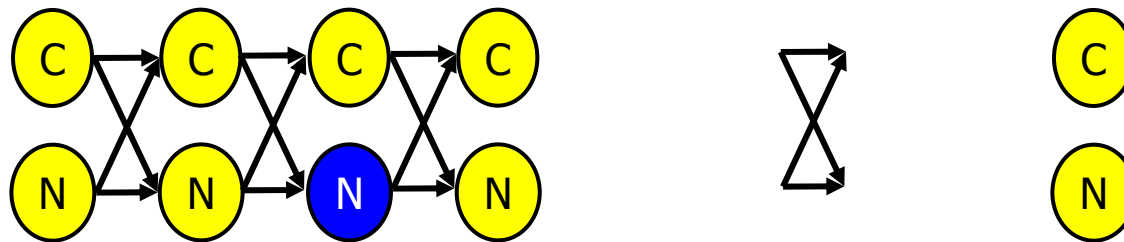
- Bessere Lösung: **Viterbi-Algorithmus**

– 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 Algorithmus: Grundidee

- Beobachtung

- Es gibt viele Pfade, um einen Zustand s an Position i der Sequenz zu erreichen
- Einer davon muss der wahrscheinlichste sein
- Alle Fortsetzungen von Pfaden ab „ s an i “ brauchen nur diese Wsk
- Also: Sukzessive alle Wsken „ s an i “ berechnen
 - Sehr ähnlich dem Algorithmus von Dijkstra



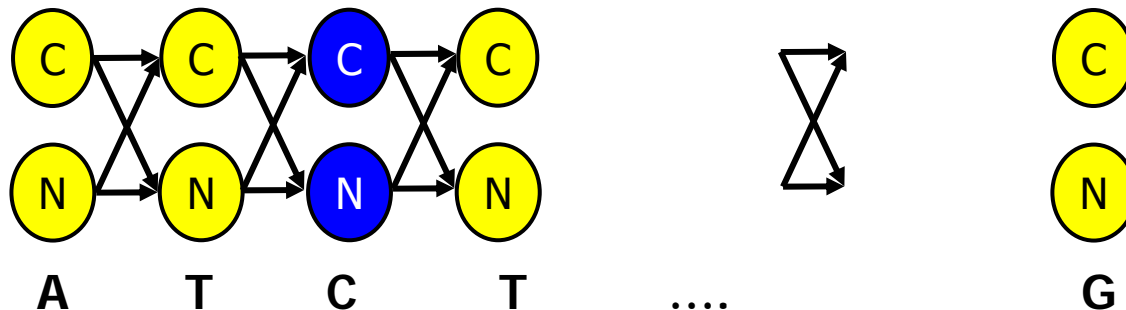
Sequenz:

A T C T

G

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	...
S_0	1	0	0	
A+	0	0,2	0	
T+	0	0	0	
G+	0	0	0	
C+	0	0	0,054	
A-	0	0,25	0	
T-	0	0	0	
G-	0	0	0	
C-	0	0	0,051	

- Startzustand s_0 hat Wsk 1, alle anderen Zustände haben als Start Wsk 0
- Wir berechnen die Tabelle spaltenweise (und nur so)
- Jedes Feld kann von **jedem Feld der Vorgängerspalte** erreicht werden
 - Also aus jedem Endzustand für das um 1 Zeichen kürzere Präfix
 - Gilt nur, wenn alle Übergangs-Wsk positiv sind
- In jeder Spalte i ist die Wsk aller Zustände, die nicht $S[i]$ emittieren, 0 (weil $e(S[i])=0$)

Tabellarische Darstellung

		A	C	T	G	...
S ₀	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?



Komplexität

- Sei $|S|=n$, HMM habe $|M|=k$ Zustände
- Allgemein
 - Tabelle hat $n*k$ Zellen
 - Für jede Zelle greifen wir auf k Vorgängerzellen zu
 - Zusammen: $O(n*k^2)$
- Unser konkretes CpG-Insel Modell
 - In unserem aktuellen Modell sind in jedem Zustand **alle Emissionswsk bis auf eine 0**
 - Damit müssen wir pro Spalte nur zwei Zellen betrachten, und die haben nur 2 Vorläufer mit $Wsk \neq 0$
 - Damit ist **der Algorithmus linear: $O(n)$**

Numerische Schwierigkeiten

- Multiplikation vieler Zahlen $\ll 1$ erreicht schnell die **Rechengenauigkeitsgrenze**
- Besser: Logarithmieren

– Statt:

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

– Berechnet man:

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

Inhalt der Vorlesung

- Decodierung: Viterbi
- Evaluation: Forward-Backward Algorithmus
- Parameterschätzung: Baum-Welch
- HMM für Sequenzalignment

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 dafür, dass S von (einer Folge von Zuständen von) M erzeugt wurde.*

Forward-Algorithmus

- Anders herum ausgedrückt
 - Gegeben ein HMM, wie wahrscheinlich generiert es eine feste Sequenz S ?
 - Einfachster Versuch
 - Mit Viterbi die wahrscheinlichste Zustandsfolge berechnen und deren Wsk ausgeben
 - Aber: S kann durch **verschiedene Zustandsfolgen** erzeugt werden
 - Zur Evaluation müssen wir über diese alle aggregieren

$$p(S | M) = \sum_{p \in \text{paths}} p(S, p)$$

- $P(S|M)$ kann durch eine **kleine Variation** des Viterbi-Algorithmus berechnet werden
 - Welche?

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

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

Komplexität

- Ändert sich nicht (im Vergleich zu Viterbi)
- Sei $|S|=n$, HMM habe k Zustände
- Tabelle hat $n*k$ Zellen
- Für jede Zelle greifen wir auf k Vorgängerkzellen zu
- Zusammen: $O(n*k^2)$

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

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

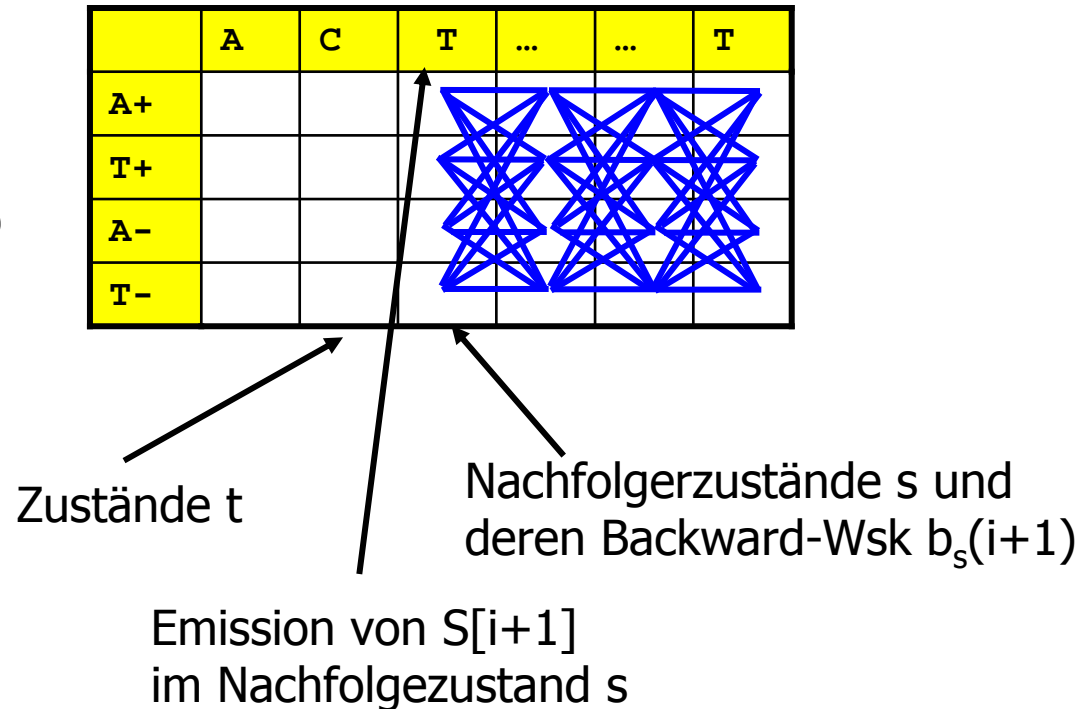
- $f_s(i)$ kennen wir: Forward-Algorithmus
- $b_s(i)$ heißt die **Backward-Wsk**
 - Wsk für die Restsequenz $S[i+1..n]$, gegeben den Startzustand s
- Man berechnet sie durch eine **Rekursion von rückwärts**

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

- 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)

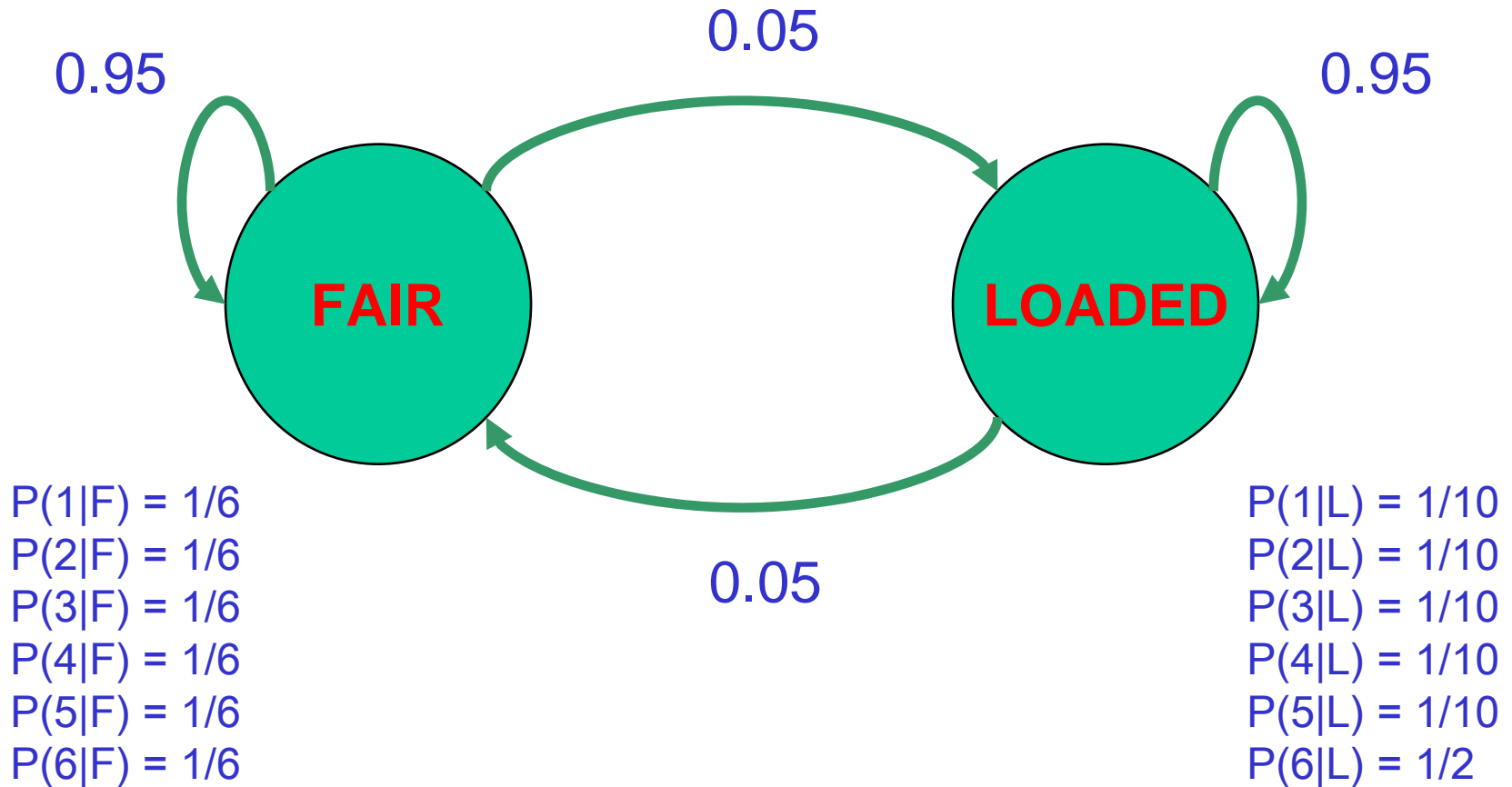
Veranschaulichung

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



- Achtung: $e(\dots)$ ist jetzt in der Klammer (anders als bei Forward-Algorithmus), da das Zeichen nach dem Übergang emittiert wird

Beispiel



Beispiel (geschätzt, nicht gerechnet)

- Sequenz 126416235626156366123413122654155241
- Der wahrscheinlichste Einzelpfad besteht nur aus F
- Aber es gibt viele Pfade, die dem blauen Bereich den Zustand L mit **höherer Wahrscheinlichkeit** zuordnen als der optimale Pfad ihm den Zustand F zuordnet
- Passiert oft, wenn
 - Sehr kleinen Übergangs-Wsken: Jeder Übergang wird so stark bestraft, dass sich Wechsel erst **bei sehr langen L-Phasen** „lohnt“
 - Die **Unterschiede zwischen den Pfad-WSK sehr klein** sind
 - Dann bietet Viterbi keine sehr gute Entscheidungshilfe

Posterior-Pfade

- Kann man aus den Zuständen einer Posterior-Dekodierung einen kompletten Pfad bauen? Wie wahrscheinlich ist der?
 - Man kann einen Pfad bauen
 - Der kann aber Wahrscheinlichkeit 0 haben
 - Tritt auf, wenn er einen Übergang mit Wsk 0 enthält
- Posterior-Dekodierung ist eine rein **lokale Betrachtung**

Inhalt der Vorlesung

- Decodierung: Viterbi
- Evaluation: Forward-Backward Algorithmus
- Parameterschätzung: Baum-Welch
- HMM für Sequenzalignment

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

Einfacher Fall: Zustandssequenz bekannt

- 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.*

- Bemerkung

- S sollte möglichst lang sein (oder es gibt viele S)

- Erster Fall

- Wir kennen zusätzlich zu **jedem Zeichen von S den Zustand**, der das Zeichen emittiert hat
- Dann reicht ein einfacher **Maximum Likelihood Estimator**
- Kann bewiesen werden (machen wir nicht)

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

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

Erweiterungen

- **Overfitting**

- Vorsicht vor Nullen: Seltene Übergänge/Emissionen werden bei **zu kleinen Trainingsmengen** nicht gesehen
 - Beachte: Wir müssen u.U. **sehr viele Parameter** lernen
- Nuller setzen aber die Wsk von Pfaden sofort auf 0, egal, wie wahrscheinlich der Restpfad ist
- Gefahr von Overfitting: Wir sind **(zu) nahe an den Trainingsdaten**
 - Trainingsdaten sind immer nur ein Ausschnitt der Realität
- Lösung: Kleine „**Pseudo-Counts**“ zu allen Häufigkeiten addieren

- **Hintergrundwissen**

- Manchmal weiß man mehr als in den Trainingsdaten steckt
 - Z.B. Gesamthäufigkeiten von Basen in einem Genom
 - Z.B. Wissen über die Häufigkeit von Phasenübergängen (CpG) im Genom; diese Übergänge sind häufig unter/überrepräsentiert in Trainingsdaten
- Dieses Wissen kann man einbeziehen
- Stichwort: „Bayes'sche Prior Distribution“

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**

Baum-Welch Algorithmus

- Baum, L. E. and Petrie, T. (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains." *Annals of Mathematical Statistics* **37**: 1554-1563.
- Iterative lokale Suchheuristik
 - Genereller Aufbau
 - Sei $S = \{S_1, \dots, S_m\}$ die Menge aller Trainingssequenzen
 - Rate beliebige Startkonfiguration a_{st} und $e_s(x)$
 - Berechne $k = P(S|M)$
 - Evaluation: Wsk aller Trainingsdaten gegeben das Modell
 - While (true)
 - Berechne neue erwartete Übergangs-Wsk a_{st}
 - Berechne neue erwartete Emissions-Wsk $e_s(x)$
 - Berechne $k' = P(S|M)$
 - » Wsk aller Trainingsdaten gegeben das neue Modell
 - Wenn $(k' - k < t)$: stop
 - $k := k'$
 - end while

Eigenschaften

- Der Baum-Welch Algorithmus **verbessert k garantiert** in jeder Iteration
 - Beweis führen wir nicht
- BW läuft in lokale Optima
 - Lösungsmöglichkeiten: Mit verschiedenen Startkonfigurationen starten, Simulated Annealing, ...
- Wenn mit Real-Zahlen gerechnet wird, terminiert BW ohne Schwellwert ($k^i - k = 0$) i.A. nicht
 - Ewige, infinitesimal kleine Verbesserungen
 - Daher Abbruch bei Unterschreitung eines Schwellwerts
- Für Machine Learner: BW ist ein **Spezialfall des Expectation Maximization Algorithmus (EM)**

Neue erwartete Übergangs-Wsk

- Berechnung der neuen a_{st}
 - Wsk des Übergangs an Position i in einer einzelnen Sequenz S_k

$$p(z_i = s, z_{i+1} = t | S_k) = \frac{f_s(i) * a_{st} * e_t(S_k[i+1]) * b_t(i+1)}{P(S_k)}$$

- Alle Pfade bis s , Übergang zu t , t emittiert $S_k[i+1]$, alle Pfade bis zum Ende
- Aggregation über alle Positionen und alle Sequenzen

$$A_{st} = \sum_k \frac{1}{P(S_k)} \sum_{i=1..|S_k|-1} f_s^k(i) * a_{st} * e_t(S_k[i+1]) * b_t^j(i+1)$$

Neue erwartete Emissions-Wsk

- Berechnung der neuen **erwarteten** $e_s(x)$

$$E_s(x) = \sum_k \frac{1}{P(S_k)} \sum_{\{i | S_k[i]=b\}} f_s^k(i) * b_t^j(i+1)$$

Komplexität

- Sei l die Anzahl an Iterationen
- Sei k die Anzahl der Zustände
- Sei m die Zahl der Sequenzen und n deren jeweilige Länge
- In jeder Iteration benötigen wir alle Forward- und Backward-Variablen aller Sequenzen
 - $O(m*n*k^2)$
- Zusammen: $O(l*m*n*k^2)$

Viterbi-Training

- BW versucht, die Wsk aller Emissionen aller Zustände aller Pfade zu maximieren
- Alternative: Maximiere nur die Wsk des wahrscheinlichsten Pfades
 - „Viterbi-Training“
 - Vorgehen
 - Benutzung der Viterbi-Variablen $v_s(i)$ statt $f_s(i)$ in der Schätzung der neuen Parameter
 - Benutzung der Viterbi-Wsk als Abbruchkriterium
 - Vorteil
 - Schnellerer, einfacherer Algorithmus
 - Konvergiert immer (gegen ein lokales Maximum)
 - Da genau ein Pfad gewinnt

Was wir nicht machen

- Wie kommt man zu einer **guten Struktur**?
 - Trade-Off: Komplexe Modelle passen sich besser an, aber man muss auch mehr Parameter raten
 - Lösung ist „... more of an art [than a science]“
- Modellierung **zeitlicher Abhängigkeiten**
 - Bisher sind die Wsk im Modell alle gleich
 - Es kann aber sinnvoll sein, die Wsk z.B. mit der Länge einer Phase ab-/zunehmen zu lassen
 - Bsp: Wir wissen, dass CpG Islands nie länger als X Basen sind. Also sollte die Wsk CpG→Non-CpG nach mehr als X Übergängen in einer CpG-Phase erheblich steigen

Markov-Ketten höherer Ordnung

- Markov-Modell der Ordnung k

- Wsk von Zustand z_i hängt von den k Vorgängern ab

$$p(z_t=s_t/z_{t-1}=s_{t-1}, z_{t-2}=s_{t-2}, \dots, z_1=s_1) = p(z_t=s_t/z_{t-1}=s_{t-1}, \dots, z_{t-k}=s_{t-k})$$

- Nicht ausdrucksstärker als Markov-Ketten der Ordnung 1

- Warum?
- Jedes Markov-Modell der Ordnung k mit n Zuständen kann in ein Markov-Modell der Ordnung 1 mit n^k Zuständen überführt werden
- Beispiel

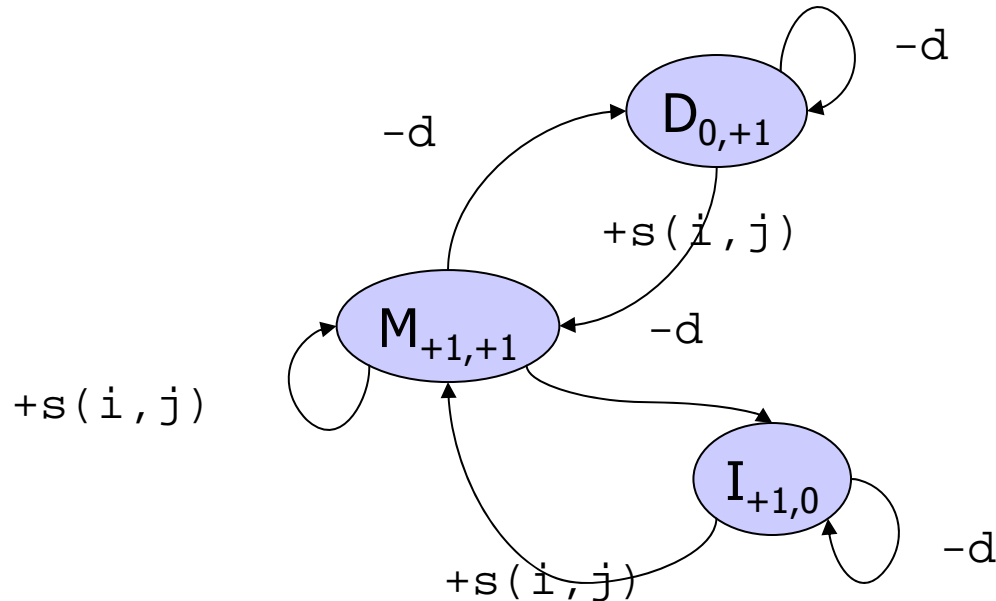
- Zustände eines Markov-Modells der Ordnung 3 für DNA-Sequenzen:
AAA, AAC, AAT, AAG, ACA, ...

- Kann aber intuitivere Modellierung ermöglichen

Inhalt der Vorlesung

- Decodierung: Viterbi
- Evaluation: Forward-Backward Algorithmus
- Parameterschätzung: Baum-Welch
- HMM für Sequenzalignment (sketched)

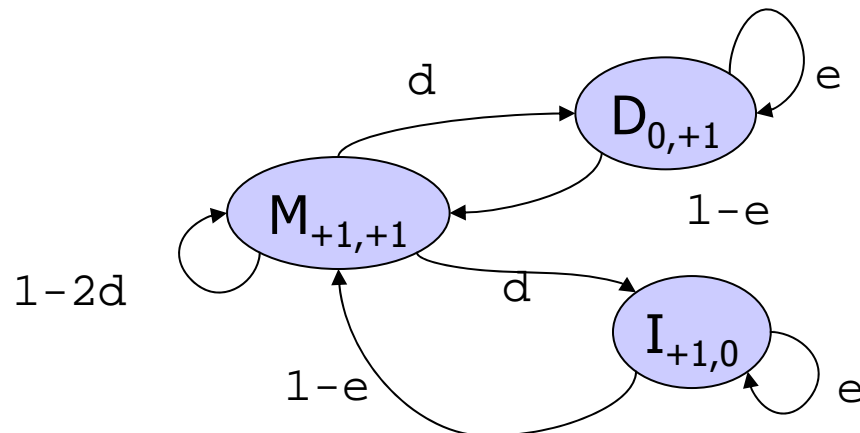
Recall: Automaten und Alignments



- Lineares (affines) Gapkostenmodell
 - Wir schenken uns den Übergang $I \leftrightarrow D$
- Ein Pfad (=Zustandssequenz) entspricht einem Alignment
- Der **beste Pfad entspricht dem optimalen Alignment**

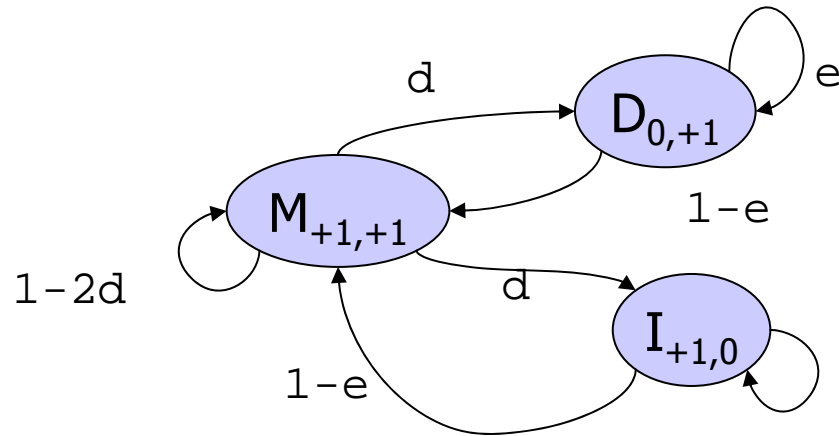
Anwendung von HMMs für das Sequenzalignment

- 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

Verdeutlichung



**-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC-GGTCGATTGCCCCGACC
IMMDMMMMMMMMDDMMMMMMMDMMMMMMMMMIIMMMMMMI II**

Training

- Wo kommen die Parameter her?
 - Müssen aus Trainingsdaten (=alignierte Sequenzen) gelernt werden
 - Dazu kann man z.B. Sequenzpaare wie für PAM / BLOSUM benutzen

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)

Literatur

- Gusfield behandelt das Thema nicht
- Literatur
 - Rabiner, L. R. (1988). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* 77(2): 257-286.
 - Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998). "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids". Cambridge, UK, Cambridge University Press.
 - Anders Krogh (1998). „An Introduction to Hidden Markov Models for Biological Sequences“.
<http://www.binf.ku.dk/~krogh/publications/ps/Krogh98a.pdf>