Proseminar: Machine-Learning

Hidden-Markov-Modelle

Benjamin Martin

Zusammenfassung

1953 stellten Watson und Crick ihr DNA-Modell vor. Damit öffnete sich für Genforscher ein riesiges Gebiet, das bisher so nicht bekannt war. Nun galt es, diese neue Struktur zu verstehen und zu analysieren, codierende Bereiche zu finden und von Nonsense-Sequenzen zu trennen, Gene zu lokalisieren und ihre Proteine und deren Strukturen kennen zu lernen. Doch die vorhandenen Methoden waren bestenfalls ungenau. Mit der Entwicklung neuer Methoden (speziell in den 60ern mit der Entwicklung der Hidden-Markov-Modelle) wurden bessere Ergebnisse erzielt.

Hier soll ein kleiner Einblick in die Theorie dieser Modelle gegeben werden, ausgehend von ihrer Definition über ein kleines Beispiel, hin zu den wichtigsten Algorithmen, die benötigt werden um ein solches Modell zu konstruieren, zu optimieren und neu anzupassen.

Inhaltsverzeichnis:

1. Einleitung

Auf der Suche nach einer Möglichkeit, effizienter die riesigen Datenmengen zu verarbeiten, die sich in den 1990ern im Bereich der Gensequenzierung angesammelt haben, waren einheitliche Modelle aus multiplen Alignments von Proteinfamilien ein viel versprechender Fortschritt. Mit diesen Modellen war es auch möglich, zusätzliche Informationen über vererbte Sequenzen, Insertionen und Deletionen zu erlangen.
Ein weiterer Fortschritt waren Hidden-Markov-Modelle (HMMs). Erste theoretische Entwicklungen für HMMs wurden bereits in den 60er Jahren gemacht. Seitdem wurden sie vor allem in der Spracherkennung, aber auch in vielen anderen Bereichen eingesetzt. Dazu gehörten auch Bereiche aus der Biologie, z. B. die Modellierung von kodierenden/nicht-kodierenden Sequenzen der DNA und Proteinbindungsstellen der DNA.
Seit Mitte der 1990er wurden HMMs aber auch, in Verbindung mit Machine-Learning, zum systematischen Modellieren, Vergleichen und Analysieren ganzer Proteinfamilien und DNA-Bereiche verwendet.
HMMs haben einen engen Bezug zu Neuronalen und Bayes`schen Netzen, sowie stochastischen Grammatiken. Hier soll ein kleiner Einblick in ihre Theorie und ihre Algorithmen gegeben werden. Spezielle Anwendungen folgen in Kapitel 2.

1.1 Definition eines HMMs

Ein diskretes HMM erster Ordnung ist ein deterministischer stochastischer Automat, der definiert ist durch:

- eine endliche Anzahl von Zuständen **S**.
- ein diskretes Alphabet **A** von Symbolen.
- eine Wahrscheinlichkeit-Übergangsmatrix $\mathbf{T} = (t_{ij})$.
- eine Wahrscheinlichkeit-Emissionsmatrix $\mathbf{E} = (e_{iX})$.
- eine Wahrscheinlichkeitsverteilung, Startzustand zu sein, über alle Zustände; $\boldsymbol{\pi}$.

Somit ergeben sich für ein HMM folgende Parameter: $\mathbf{w} = (\boldsymbol{\pi}, \mathbf{T}, \mathbf{E})$. Ein HMM ist also abhängig von seinem Startzustand, seiner Übergangsmatrix **T** mit $t_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$ und Emissionsmatrix **E**.
Ein HMM wechselt zufällig von einem gegebenen Zustand **i** mit der Wahrscheinlichkeit t_{ij} in den Zustand **j** und emittiert mit Wahrscheinlichkeit e_{iX} das Symbol **X**. Eine wichtige Annahme hierbei ist, dass Emissionen und Übergänge nur vom aktuellen, und vom neuen Zustand abhängt, nicht von vorherigen. Allerdings gibt es auch Varianten von HMMs, deren Zustandsübergänge von einem oder mehreren vorherigen Zuständen abhängen. Diese werden HMMs x-ter Ordnung genannt, wobei x die Anzahl der Zustände aus vorangegangenen Schritten ist, von denen der nächste Schritt abhängt.
Zu beobachten sind nur die emittierten Symbole, nicht der zu Grunde liegende Weg, auf dem das System die Zustände wechselt. Daher der Begriff „hidden“. Die versteckten, zufälligen Wege können als versteckte oder latente Variablen gesehen werden, die den Beobachtungen zu Grunde liegen.

1.1.1 Kleines Beispiel

Ein einfaches Beispiel ist in Abbildung 1.1 dargestellt. Zwei imaginäre „DNA-Würfel“ mit Wahrscheinlichkeitsvektoren für Emission: ($e_{1A} = 0{,}25$, $e_{1C} = 0{,}25$, $e_{1G} = 0{,}25$, $e_{1T} = 0{,}25$) und ($e_{2A} = 0{,}1$, $e_{2C} = 0{,}1$, $e_{2G} = 0{,}1$, $e_{2T} = 0{,}7$). Übergangswahrscheinlichkeiten siehe Abbildung.
Angenommen, es würde nun eine DNA-Sequenz beobachtet, zum Beispiel: ATCCTTTTTTTCA, werden sich schnell 3 Fragen ergeben:

1) Wie wahrscheinlich ist diese Ausgabe für dieses HMM? Und wie lässt sie sich berechnen?
2) Welche ist die wahrscheinlichste Folge von Übergängen und Emissionen für diese spezielle Ausgabe? Wie kann die Zustandsabfolge berechnet werden?
3) Wie sind die t_{ij} und e_{iX} zu berechnen, bzw. deren Werte zu verbessern mit der beobachteten Ausgabe? Wie kann das HMM lernen?

Nachfolgend werden diese drei Fragen behandelt und Algorithmen zu ihrer Lösung vorgestellt.

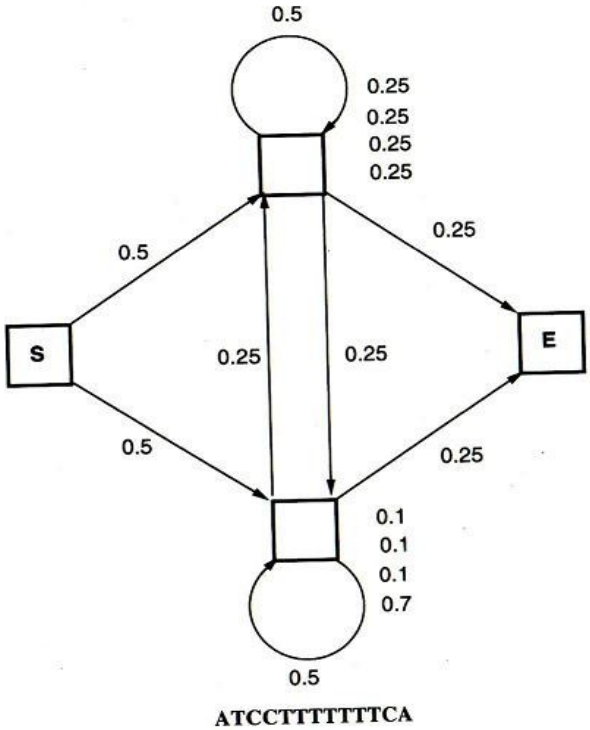

Abbildung 1.1: Einfaches Modell eines HMMs für Gensequenzen. ([Baldi98] S.146)

1.2 Wahrscheinlichkeit und grundlegende Algorithmen

In diesem Abschnitt werden die grundlegenden Algorithmen zur Lösung der ersten beiden Fragen diskutiert. Insbesondere wird es um die Berechnung der wahrscheinlichsten Sequenz von Zustandsübergängen und Emissionen bei einer beobachteten Sequenz gehen. Auch sind diese Algorithmen wichtig für die in Kapitel 1.3 beschriebenen Lernalgorithmen.
Ein erster Ansatz ist es, die Wahrscheinlichkeit **P(O| w)** berechnen, wobei $\mathbf{O = X^1 ... X^t ... X^T}$, die beobachtete Sequenz bei gegebenem HMM M = M(w) mit Parameter w ist. Dazu wird der Weg in **M** als $\mathbf{Q = q_1,q_2, ... q_T}$ definiert, eine Abfolge von Zuständen in M, beginnend mit *start* und endend mit *end* und mit einem Emissionssymbol für jeden emittierenden Zustand auf diesem Weg. Stimmt die Sequenz der emittierten Symbole auf dem Weg mit O überein, so erhalten wir:

$$\mathbf{P(O, Q \mid w)} = \prod_{start}^{end} t_{ij} \prod_{t=1}^{T} e_{iX^t} \quad ,$$

wobei das erste Produkt über alle Übergänge in **Q** und das zweite Produkt über alle zugehörigen emittierenden Zustände **i** in **Q** genommen wird. Stimmt **O** nicht mit der emittierten Sequenz überein, so ist **P(O, Q | w) = 0**. Die Wahrscheinlichkeit für eine Abfolge von Ausgaben kann dann wie folgt angegeben werden:

$$\mathbf{P(O \mid w)} = \sum_{Q} P(O, Q|w) \quad .$$

Diese Art der Berechnung hat allerdings auf Grund der Anzahl aller möglichen Pfade exponentielle Laufzeit. Allerdings gibt es einen effizienteren Weg, die Wahrscheinlichkeit zu berechnen, den sogenannten „Forward-Algorithmus“. Hierbei wird versucht, zu vermeiden, auf alle versteckten Pfade zu schauen.

1.2.1 Der *Forward*-Algorithmus

Sei

$$\mathbf{\alpha_i(t) = P(q_t = S_i, X^1 \ldots X^t \mid w)}$$

die Wahrscheinlichkeit, in Zustand **i** zur Zeit **t** zu sein und $\mathbf{X^1}$ bis $\mathbf{X^t}$ im Modell **M** beobachtet zu haben, beginnend mit

$$\mathbf{\alpha_{start}(0) = 1.}$$

Es soll $\mathbf{P(O \mid w) = \alpha_{end}(T)}$ berechnet werden**.** Die $\mathbf{\alpha_i(t)}$ können rekursiv berechnet werden durch

$$\mathbf{\alpha_i(t+1)} = \sum_{j \in S} \alpha_j(t) t_{ji} e_{iX^{t+1}} \quad .$$

Dies gilt für alle emittierenden Zustände. Für HMMs für multiple Alignments (2.1) gilt für *delete*-Zustände:

$$\mathbf{\alpha_i(t+1)} = \sum_{j \in S} \alpha_j(t) t_{ji} \quad .$$

Für wachsende **t** wird also die Wahrscheinlichkeit berechnet, dass $\mathbf{O = X^1 \ldots X^T}$ eintritt. Dies verläuft ähnlich der Berechnung eines Neuronalen Netzes mit **T** Schichten und **|S|** Zellen pro Schicht. Daraus folgt eine Laufzeit von **O(MT).** Eine weitere Abschätzung der Laufzeit des Forward-Algorithmus ergibt eine Laufzeit von $\mathbf{O(N^2T)}$, wobei **N** als die Länge des HMMs definiert ist, ist also in angemessener Zeit berechenbar.

1.2.2 Der *Backward*-Algorithmus

Dieser Algorithmus ist die Umkehrung des Forward-Algorithmus. (Ähnlich dem Lernen bei Neuronalen Netzen, müssen wir uns hier vom letzten Zustand aus rückwärts durch unser Modell bewegen, um die Wahrscheinlichkeiten zu berechnen)
So wird β als Variable dieses Algorithmus viel folgt definiert:

$$\mathbf{\beta_i(t) = P(X^{t+1} \ldots X^T \mid q_t = S_i, w)}$$

Also die Wahrscheinlichkeit, in Zustand **i** zur Zeit **t** zu sein und $\mathbf{X^{t+1}}$ bis $\mathbf{X^T}$ beobachtet zu haben. Analog zu den α_i wird

$$\beta_i(T) = 1, \quad \forall i : i \in \{1, \ldots, N\}$$

$$\beta_i(t) = \sum_{j \in S} \beta_j(t+1) t_{ij} e_{iX^{t+1}}$$

$$\beta_i(t) = \sum_{j \in S} \beta_j(t+1) t_{ij}$$

für Initialisierung, Fortschritt und für multiple Alignments (2.1), *delete*-Zustände. Auch dieser Algorithmus hat die Laufzeit **O(N²T)**.

Forward- und Backward-Algorithmus zusammen ermöglichen es, die Wahrscheinlichkeit zu berechnen zum Zeitpunkt **t** im Zustand **i** zu sein, bei gegebener Beobachtungssequenz und HMM **w**. Dazu wird $\gamma_i(t)$ derart definiert, dass

$$\gamma_i(t) = P(S_t = i \mid O, w) = \frac{\alpha_i(t) \beta_i(t)}{P(O|w)}$$

also

$$\gamma_i(t) = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j \in S} \alpha_j(t) \beta_j(t)}$$

Um den wahrscheinlichsten Zustand zum Zeitpunkt **t** zu finden, wird versucht $\gamma_i(t)$ zu maximieren. Allerdings wird meistens der wahrscheinlichste Pfad gesucht, da dieser auch eine wichtige Rolle beim Lernen und Anpassen eines HMMs spielt.

1.2.3 Der Viterbi-Algorithmus (Von A.J. Viterbi 1967 entworfen)

Ähnlich den $\alpha_i(t)$ und $\beta_i(t)$ in 1.2.1 und 1.2.2 wird hier eine Variable $\delta_i(t)$ definiert**.** Diese steht für die höchste Wahrscheinlichkeit der Pfade mit Länge **t** (und den ersten **t** beobachteten Symbolen)**,** die zu diesem Zeitpunkt in Zustand **i** enden. Also:

$$\delta_i(t) = \max_{j=1,\ldots,N} P(q_{t-1} = S_j, q_t = S_i, X^1 X^2 \ldots X^t | w)$$

Ähnlich wie beim Vorwärts-Algorithmus gilt auch hier:

$$\delta_i(1) = \pi_i \cdot e_{iX^1}$$

$$\delta_i(t+1) = [\max_j \delta_j(t) t_{ji}] e_{iX^{t+1}}$$

für emittierende Zustände. Auch hier gilt für multiple Alignments (2.1) für *delete*-Zustände:

$$\delta_i(t+1) = [\max_j \delta_j(t) t_{ji}]$$

1.3 Lernalgorithmen

Es gibt eine Vielzahl von Lernalgorithmen, um HMMs zu trainieren. Dazu gehören auch die sogenannten EM-Algorithmen. Diese laufen alle nach ähnlichem Schema ab:
- Erwartete Wahrscheinlichkeiten berechnen (**E**)
- Diese errechneten Werte maximieren (**M**)
Ein für HMM-Training wichtiger Algorithmus aus der Klasse der EM-Algorithmen ist der Baum-Welch-Algorithmus (auch *forward-backward*-Algorithmus genannt).

1.3.1 Der Baum-Welch-Algorithmus

Der Baum-Welch-Algorithmus beantwortet die dritte der obigen Fragen und dient sowohl der Maximierung der Wahrscheinlichkeit für eine bestimmte Beobachtungssequenz bei gegebenem HMM, als auch der Neubestimmung und Optimierung der Modellparameter des HMMs. Dazu werden alle Übergänge von Zustand $\mathbf{S_i}$ nach $\mathbf{S_j}$ betrachtet und jeweils maximiert. Hierfür wird $\boldsymbol{\xi_{ij}(t)}$ definiert als die Wahrscheinlichkeit, zum Zeitpunkt **t** in Zustand $\mathbf{S_i}$ und zum Zeitpunkt **t+1** in Zustand $\mathbf{S_j}$ zu sein, bei gegebenem HMM und **O**. Also

$$\boldsymbol{\xi_{ij}(t) = P(q_t = S_i, q_{t+1} = S_j \mid O, w)}$$

Mittels des Forward- und Backward-Algorithmus und deren $\boldsymbol{\alpha_i(t)}$ und $\boldsymbol{\beta_i(t)}$, sowie dem Theorem von Bayes können wir $\boldsymbol{\xi_{ij}(t)}$ schreiben als:

$$\boldsymbol{\xi_{ij}(t)} = \frac{\alpha_i(t) t_{ij} e_{iX^{t+1}} \beta_j(t+1)}{P(O|w)}$$

Also die Wahrscheinlichkeit, zum Zeitpunkt **t** von Zustand **i** nach Zustand **j** zu wechseln. $\boldsymbol{\xi_{ij}(t)}$ kann mit oben definiertem $\boldsymbol{\gamma_i(t)}$ in Beziehung gebracht werden durch:

$$\boldsymbol{\gamma_i(t)} = \sum_{j=1}^{N} \xi_{ij}(t)$$

Mit den nachfolgenden zwei Summen erhalten wir die erwartete Anzahl von Übergängen aus dem Zustand $\mathbf{S_i}$ und die erwartete Anzahl der Zustandswechsel von $\mathbf{S_i}$ nach $\mathbf{S_j}$:

$$\sum_{t=1}^{T-1} \gamma_i(t)$$

$$\sum_{t=1}^{T-1} \xi_{ij}(t)$$

Mittels dieser Formeln können wir nun die Parameter $\boldsymbol{\pi}$, **T** und **E** neu abschätzen durch:

$$\hat{\pi}_i = \gamma_i(1) \quad ,$$

wobei $\hat{\pi}_i$ die erwartet Häufigkeit angibt, zum Zeitpunkt $\mathbf{t = 1}$ in Zustand $\mathbf{S_i}$ zu sein.

$$\hat{t}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

und

$$\hat{e}_{jX^k} = \frac{\sum_{\forall t | X^t = X^k} \gamma_j(t)}{\sum_{t=1}^{T} \gamma_j(t)}$$

So erhalten wir neue Parameter für w und $\hat{w} = (\hat{\pi}_i, \hat{T}, \hat{E})$. Mit diesen neuen Parametern kann eine erneute Iteration der obigen Schritte (E und M) erfolgen, bis sich die geschätzten Werte nicht mehr ändern.

1.4 Weiteres

Trotz der hier genannten Algorithmen zum Optimieren und Lernen für HMMs bleiben dennoch einige Schwierigkeiten und Fragen zu klären. So sind zum Beispiel die Wahrscheinlichkeiten P(O, Q | w) sehr klein, da sie das Produkt vieler Übergangs- und Emissionswahrscheinlichkeiten sind, die allesamt kleiner als 1 sind. Gerade bei größeren Modellen werden die Werte dadurch beliebig klein. Daher wird versucht, über das sogenannte *Scaling* die Werte der β_i und α_i anzugleichen (zu erhöhen), um zu kleine Ergebnisse zu vermeiden.
Eine weiter Frage ist, ob anhand vorhandener Daten HMMs aufgebaut werden können. Auch hier gibt es Versuche, gerade im biologischen Bereich für die Sequenzanalyse. Dabei wird von einem komplexen Modell ausgegangen, mit so vielen Zuständen wie in der zu analysierenden Sequenz vorhanden sind. Dann wird solange versucht, Zustände zu verschmelzen, bis die Auswertung und der Vergleich mit einem früheren Ergebnis zufriedenstellend sind. Andersherum ist es möglich, von einem kleineren, aber vollständigen Modell auszugehen und in jedem Schritt die Übergänge mit der kleinsten Wahrscheinlichkeit zu löschen und die Zustände der Übergänge mit hoher Wahrscheinlichkeit zu duplizieren. Mit diesen beiden Methoden können im Allgemeinen auf HMMs mit weniger als 50 Zuständen erzeugt werden. Allerdings ist diese Methode sehr langwierig.
Ein Versuch, dies zu optimieren, ist, speziell für multiple Alignments (2.1), ein Standard-HMM der Länge *N* zu trainieren. *N* sei auch die Länge der zu untersuchenden Sequenz. Die Grundidee des Trainings ist, durch das HMM zu laufen und an geeigneten Stellen Zustände einzufügen oder zu löschen, derart, dass für *insert*-Zustände die in mehr als 50 % aller Viterbi-Pfade vorkommen ein *main*-Zustand eingefügt wird, zusammen mit *delete*- und *insert*-Zuständen. Analog wird ein zu mehr als 50 % genutzter *delete*-Zustand zusammen mit seinem *insert*- und *main*-Zustand gelöscht.

Allerdings wurde hierfür nie ein Beweis gefunden, dass dieses Verfahren terminiert.
Eine weitere wichtige Rolle für effizientere Ergebnisse spielt auch die Zusammenarbeit und die gezielte Verschaltung mehrerer HMMs untereinander.
Auch können Symbole mehrere Bedeutungen besitzen (z. B. kann ein X für A, T, C oder G in einer Sequenz stehen). In diesem Fall wird häufig das Symbol mit der höchsten Wahrscheinlichkeit eingesetzt, was aber gerade in Sequenzen mit einer hohen Anzahl solcher mehrdeutigen Symbole zu Problemen führen kann.

2. Anwendungsbeispiele von HMMs

Unabhängig von Struktur und Lernmechanismus, kann ein HMM in verschiedenen Bereichen angewendet werden. Dazu gehören:

1. Multiple Alignments
2. Data mining und Sequenz- und Fragmentklassifikation
3. Strukturanalyse und Mustererkennung

In diesem Abschnitt werden kurz die Anwendungsgebiete der multiplen Alignments und DNA/RNA-Bereiche angesprochen.

2.1 Multiple Alignments

Die Berechnung des Viterbi-Pfades wird auch das „Anpassen einer Sequenz an ein HMM“ genannt. So können durch das Anpassen verschiedener Viterbi-Pfade aneinander multiple Alignments effizient bestimmt werden. Da Trainingsphasen teilweise viel Zeit in Anspruch nehmen, kann dies auch *offline* gemacht werden. Eine solche Trainingsphase mit K Viterbi-Pfaden beansprucht $O(NK)$ Laufzeit, also lineare Laufzeit, und ist daher sehr effektiv. Auch kann durch ein HMM mehr an Information gewonnen werden als bei gewöhnlichen Alignments, Zum Beispiel beim Anpassen zweier Sequenzen, wobei diese an einer Stelle eine Lücke aufweisen. Diese Lücke kann sowohl von einer Deletion in der ersten, als auch einer Insertion in der der zweiten Sequenz stammen. Währen nun das konventionelle Verfahren diesen Unterschied nicht berücksichtigt, wird dieser Unterschied in einem HMM durch 2 verschiedene Viterbi-Pfade dargestellt.
Beim Training wird meistens von einem Standard-HMM ausgegangen (Abbildung 2.1). Danach gibt es viele Möglichkeiten, das HMM zu trainieren (siehe auch 1.4).
Eine wichtige Frage ist, ob und wie es möglich ist, die Operationen von *delete*- und *insert*-Zuständen in Verbindung mit der Evolution zu bringen. Allerdings sind HMMs nicht geeignet, klare Vererbungsabläufe darzustellen, da ihnen unter anderem die Möglichkeit fehlt, eine Baumstruktur zu erzeugen.

2.2 DNA und RNA Anwendungen

Im Gegensatz zu den Verfahren, die zur Bestimmung der Proteinsequenzen verwendet werden, ist es wesentlich schwieriger, DNA-Sequenzen zu bestimmen, gerade wenn es um Mutationen und Deletionen geht. Daraus resultiert auch oftmals eine zweifelhafte Signifikanz beim Sequenz-Alignment. Eine evolutionär stabilere Form von DNA-Sequenzen ist daher ein wichtiger Bestandteil.
HMMs benötigen im Voraus keine explizite Definition von möglichen Veränderungen innerhalb der DNA-Sequenz. Dies wird in der Praxis durch eine individuelle Verteilung des Aufwands einer Substitution an verschiedene Positionen im HMM realisiert. Bei HMMs wird daher versucht, das schwierige Problem der „many-to-many multiple sequence alignment“ auf ein einfacheres „many-to-one sequence-to-HMM alignment“ zu reduzieren. So gelang es mittels HMMs schon des öfteren, genauere Ergebnisse zu liefern, als es mit bisherigen Methoden möglich war.

2.2.1 Genanalyse in Pro- und Eukaryoten

Um Gene lokalisieren zu können, müssen viele verschiedene Sequenzen bekannt sein, unter anderem: Promotor Regionen, Start- und Stopp-Sequenzen der Translation, reading frames, und speziell für Eukaryoten die Sequenzen die für das Intronsplicing verantwortlich sind. Bei Prokaryoten ist im Gegensatz zu Eukaryoten die Dichte der Gene wesentlich höher. Allerdings gibt es bei Prokaryoten keine störenden Sequenzen innerhalb einer Gensequenz. Auf diese und mehr Unterschiede muss bei der Konstruktion eines HMMs geachtet werden.
Ein HMM wurde entwickelt, um Gene in *E. coli*-DNA zu finden, die für Proteine kodieren. Dieses HMM besaß Zustände zum Modellieren von Codons und deren Basensequenz in *E. coli*-Genen, sowie Muster innerhalb von Gensequenzen und repetitiven Sequenzen außerhalb von Genen sowie die Shine-Dalgarno Sequenz. Das HMM wurde so ausgelegt, dass es für auftretende Fehler innerhalb einer Sequenz für individuelle Nukleotide Insertionen und Deletionen zuließ. Die Parameter des HMM wurden unter Verwendung von circa einer Million Nukleotiden einer bekannten DNA abgeschätzt und das HMM auf unabhängigen *contigs* mit circa 325 000 Nukleotiden getestet. Das HMM fand die exakten Positionen von 80 % der bekannten *E. coli*-Gene und annähernd richtige Positionen von etwa 10 %. Aber es fand auch einige mögliche neue Gene und lokalisierte mehrere Stellen, an denen Insertionen und Deletionen aufgetreten waren.
Für Eukaryoten wurden gegen Ende 1997 einige viel versprechende HMMs entworfen. Bei Eukaryoten wird meist mit Submodellen, gerade für Exons, Introns und Splicingsides, gearbeitet. Diese Submodelle müssen allerdings klein gehalten werden, gerade wenn es das Ziel ist, das gesamte Genom zu analysieren. Auch kann entscheidend sein, für jede der drei bekannten Möglichkeiten, mit denen Introns den reading frame unterbrechen, je ein Submodell zu benutzen. So wurden gerade gegen Ende 1997 durch ein solches, etwas modifiziertes HMM, Ergebnisse bei der Suche nach Genen von 75-80 % erzielt.

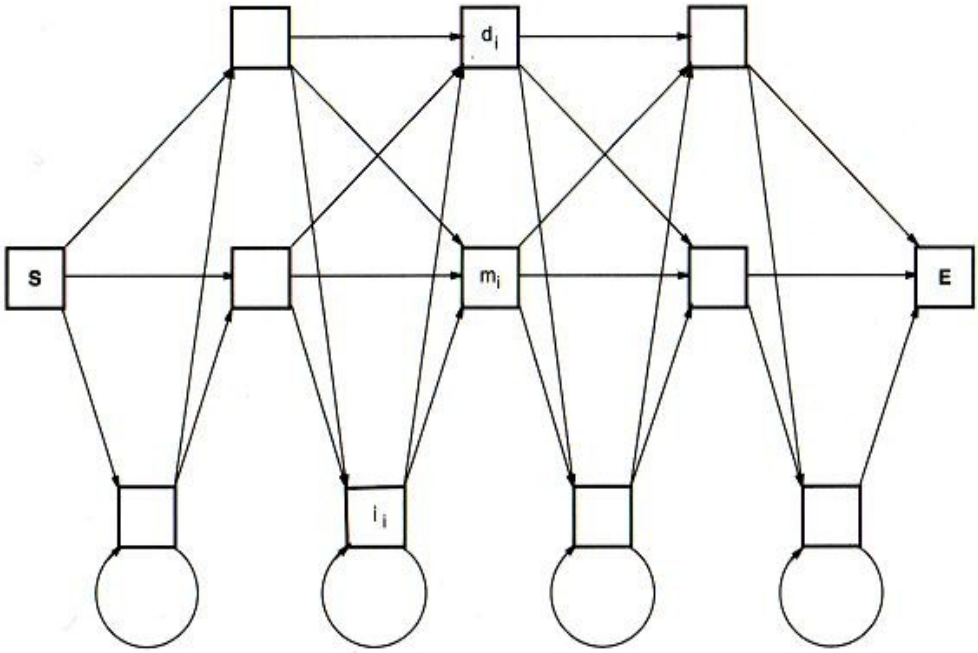

Abbildung 2.1: Standard-HMM für multiple Alignments. Die Länge beträgt $N = 5$, also von S (Startzustand) der direkte Weg über m_i (*main*-Zustand) nach E (Endzustand). Die d_i und i_i sind die jeweils zu m_i zugehörigen *delete*- und *insert*-Zustände.([Baldi98] S.147)

3. Vor- und Nachteile von HMMs

3.1 Vorteile

HMMs besitzen eine solide statistische Grundlage und sehr effiziente Lernalgorithmen. HMMs können direkt von Rohdaten lernen. Auch brauchen sie nicht wie Neuronale Netze eine Eingabe fester Länge und auch keinen Lehrer. HMMs sind ungemein flexibel. Ihr Anwendungsgebiet erstreckt sich über eine große Zahl unterschiedlicher Bereiche. Auch lassen sie sich leicht in Anwendungen einbinden, sei es modular oder hierarchisch.

3.2 Nachteile

Trotz ihres Erfolges besitzen HMMs zwei Nachteile.
Erstens bestehen sie aus einer Vielzahl unstrukturierter Parameter, was gerade im Bereich der Proteinanalyse zu einem Problem werden kann, wenn nur wenige der zu untersuchenden Sequenzen vorhanden sind.
Zweitens können gerade HMMs erster Ordnung keine Abhängigkeiten zwischen weiter entfernten versteckten Zuständen berechnen und gerade dies ist nötig, um die komplexe Proteinfaltung zu simulieren.

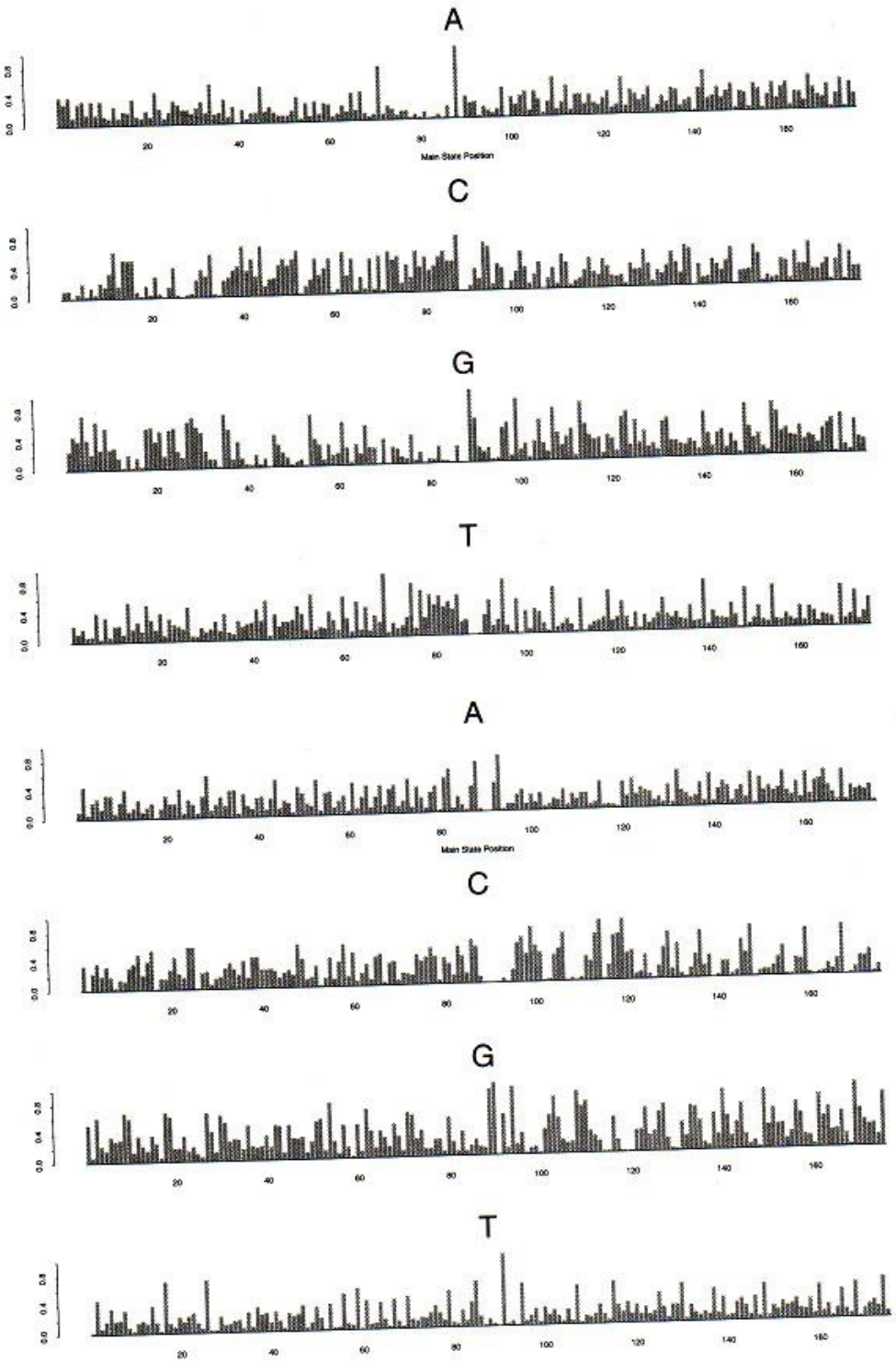

Abbildung 2.2: Ausgabe der *main*-Zustände eines HMMs, das an 1000 Acceptot-Sites (oben) und 1000 Donor-Sites (unten) trainiert wurde. Die Länge des HMMs ist 175. Gut zu sehen ist, dass Acceptor- und Donor-Site in etwa komplementär sind. ([Baldi98] S.188)

4. Literaturverzeichnis

[Baldi98] P. Baldi, S. Brunak, Bioinformatics: The Machine Learning Approach, 2. Edition, Kapitel 7 und 8, S. 165-223, 2001

[Rab89] L. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, IEEE 1989

5. Zusammenfassung

Es wurde ein kleiner Einblick in die Theorie der Hidden-Markov-Modelle gegeben, beginnend mit ihrer Definition und einem kleinen Beispiel. Anschließen wurden die grundlegenden Algorithmen und die Berechnungen der Wahrscheinlichkeiten vorgestellt. Danach wurde die Suche nach einem optimalen Pfad durch das HMM und die Neubestimmung der Modellparameter behandelt.
Aus dem Bereich der praktischen Anwendungen wurden Beispiele aus der Genetik vorgestellt, sowie die Vor- und Nachteile von HMMs und ihre Grenzen.