Bachelorarbeit

Informatik

Entwicklung einer Veranschaulichung von Hidden Markov Modellen zur Unterstützung der Lehre

Eingereicht von Chris Jacobs

Matrikel Nr.: 184239

Datum: 8. Mai 2012

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Irxleben, 8. Mai 2012 ….....

Inhaltsverzeichnis

Abstract:

Hidden Markov Modelle können schwierig zu verstehen sein. Daher ist es hilfreich wenn Material zur Veranschaulichung von Hidden Markov Modellen verfügbar ist, um sie besser verstehen zu können. Die Arbeit zeigt, wie Hidden Markov Modelle veranschaulicht werden können. Dazu wurde ein einfaches Hidden Markov Modell in einem Java Applet implementiert. Weiterhin wurde der Forward Algorithmus und der Viterbi Algorithmus implementiert. Beide Algorithmen dienen dazu wichtige Probleme bei Hidden Markov Modellen zu lösen.

1. Einleitung

1.1. Motivation

Die Lehrveranstaltung Introduction to Simulation[1] behandelt verschiedene grundlegende Themen der Simulation. In diesem Zusammenhang gibt es eine Vorlesung über Markov Ketten. In dieser Vorlesung werden auch kurz Hidden Markov Modelle behandelt. Da Hidden Markov Modelle nur kurz angesprochen werden und sie sich stark von den anderen behandelten Themen der Lehrveranstaltung unterscheiden, kann es für die Studenten schwierig sein, sie zu verstehen. Für Studenten denen die Vorlesung nicht ausreicht um Hidden Markov Modelle zu verstehen oder die daran interessiert sind sich weiter in das Thema zu vertiefen, wäre es hilfreich zusätzliches Material zur Verfügung zu haben. Dabei ist es hilfreich wenn das Material interaktiv ist. Auf der Webseite von Introduction to Simulation werden bereits zu verschiedenen Themen Java Applets angeboten. Daher bietet es sich an, ein Applet zu erstellen, um Hidden Markov Modelle zu veranschaulichen.

1.2. Ziel der Arbeit

Das Ziel der Arbeit ist es, Hidden Markov Modelle interaktiv zu veranschaulichen. Die Veranschaulichung soll in Form eines Java Applets erstellt werden. Um verschiedene Aspekte des Modells näher betrachten zu können, soll das Applet verschiedene Interaktionsmöglichkeiten haben. Laut *„an Introduction to Hidden Markov Models“* von L.R. Rabiner und B.H. Juang[2] gibt es drei Schlüsselprobleme die für ein Hidden Markov Modell gelöst werden müssen. Zwei dieser Probleme lassen sich analytisch lösen. Die dafür verwendeten Algorithmen, der Forward Algorithmus und der Viterbi Algorithmus, sollen ebenfalls veranschaulicht werden. Als Vergleich zur rechnerischen Lösung der zwei Probleme soll eine Monte Carlo Simulation implementiert werden.

1.3. Aufgaben

Das Applet wird mit AnyLogic erstellt. Als Beispiel eines Hidden Markov Modells wird ein Modell aus der Vorlesung implementiert. Das Modell ist ein einfaches Krankheitsmodel. Es hat nur fünf Zustände und nur vier Beobachtungen. Für das implementierte Beispiel soll der Nutzer sich einen

Trace von Beobachtungen generieren lassen können. Um den Trace auszuwerten, kann man den Forward Algorithmus und den Viterbi Algorithmus auf ihn anwenden. Weiterhin soll eine Monte Carlo Simulation implementiert werden. Die Ergebnisse der Algorithmen sollen mit den Ergebnissen der Simulation vergleichbar sein. Außerdem soll der Nutzer die Möglichkeit haben sich Erläuterungen, über Hidden Markov Modelle und über die beiden implementierten Algorithmen, anzeigen zu lassen.

2. Hintergrund

Hidden Markov Modelle werden dazu verwendet, um Rückschlüsse auf Daten, die im Modell als Zustände modelliert werden, zu ziehen die nur durch andere Daten, im Modell die Beobachtungen, impliziert werden. Sie finden Anwendung in der Handschriften-, Sprach- oder Gestenerkennung. Auch in der Bioinformatik werden sie verwendet. In der Bioinformatik können sie zum Beispiel dazu verwendet werden, Ähnlichkeiten zwischen Proteinen zu erkennen.[3] Weiterhin werden sie in Spamfiltern, speziell Markow-Filter, verwendet. Der Filter errechnet dabei mit welcher Wahrscheinlichkeit eine überprüfte Wortkette zu bekannten Spamtexten passt.[4]

Im folgenden Kapitel erläutere ich kurz die Definition von Hidden Markov Modellen. Des Weiteren werde ich kurz darauf eingehen wie der Forward und der Viterbi Algorithmus funktionieren.

2.1. Hidden Markov Modell

Formal wird ein Hidden Markov Modell als ein Quintupel (X, A, Y, B, π) definiert. Beziehungsweise kann ein Hidden Markov Modell kompakter als $\lambda=(A, B, \pi)$ beschrieben werden.[2]

- X ist die Menge aller möglichen Zustände im Modell
- A ist die Zustandsübergangsmatrix die angibt mit welcher Wahrscheinlichkeit das Modell von einem Zustand in einen anderen übergeht.
- Y ist die Menge aller möglichen Beobachtungen.
- B ist die Beobachtungsmatrix die angibt mit welcher Wahrscheinlichkeit welche Beobachtung in welchem Zustand gemacht wird.
- π ist die Anfangswahrscheinlichkeitsverteilung die angibt mit welcher Wahrscheinlichkeit das Modell mit welchem Zustand startet.

Der Unterschied zwischen einem Hidden Markov Modell und einer Markov Kette besteht vor allem darin, dass der aktuelle Zustand des Modells von außen nicht erkennbar ist und jeder Zustand eine Beobachtung erzeugt.[6]

Wie schon in 1.2. erwähnt gibt es drei zu lösende Schlüsselprobleme bei Hidden Markov Modellen, um sie sinnvoll anwenden zu können.

Das erste Problem ist das Evaluierungsproblem. Zur Lösung des Evaluierungsproblems muss die Wahrscheinlichkeit $P(O|\lambda)$, wobei O eine Sequenz von Beobachtungen darstellt, bestimmt werden. Es ist also zu berechnen wie wahrscheinlich eine bestimmte Sequenz von Beobachtungen in einem gegebenen Modell auftreten kann. Das Evaluierungsproblem lässt sich mit dem Forward Algorithmus lösen. Das Ergebnis dient dazu, um verschiedene Modelle zu bewerten und zu vergleichen. So kann man sich das für einen bestimmten Zweck am besten geeignete Modell aussuchen, wenn mehrere zur Auswahl stehen.

Das zweite Problem ist das Dekodierungsproblem. Beim Dekodierungsproblem soll eine Folge von Zuständen, also einen Pfad, bestimmt werden. Gesucht ist dabei der Pfad der am besten zu einer Sequenz von Beobachtungen passt. Den gesuchten Pfad kann man mit dem Viterbi Algorithmus bestimmen. Das Dekodierungsproblem entsteht weil man bei einem Hidden Markov Modell die internen Zustände die das Modell durchläuft nicht erkennen kann, man kann nur versuchen anhand

der Beobachtungen Rückschlüsse auf die Zustände zu ziehen.

Das dritte und schwerste Problem ist das Training des Modells. Dabei sollen die Parameter so angepasst werden, dass $P(O|\lambda)$ maximiert wird. Dieses Problem lässt sich nicht analytisch lösen. Man kann es iterativ mit dem Baum-Welch Algorithmus lösen. Dieses Problem wird hier nicht näher behandelt und ist auch nicht im Applet implementiert da es zu umfangreich ist.

2.2. Forward Algorithmus

Der Forward Algorithmus ist ein Algorithmus mit dem das erste Problem bei Hidden Markov Modellen gelöst werden kann[2]. Es ist möglich mit ihm zu berechnen mit welcher Wahrscheinlichkeit eine bestimmte Sequenz von Beobachtungen in einem gegebenen Hidden Markov Modell gemacht wird. Im Forward Algorithmus wird die Forward Variable $\alpha_t(i)$ berechnet, $\alpha_t(i)$ gibt die Wahrscheinlichkeit an zum Zeitpunkt t im Zustand $x_i \in X$ die Beobachtungen o_1 bis o_t gemacht zu haben.

Der Algorithmus wird initialisiert indem die Wahrscheinlichkeit $\alpha_1(i)$ berechnet wird. Dazu wird $\pi_i * b_i(o_1)$ für $1 \le i \le |X|$ berechnet.

Nach der Initialisierung wird als Rekursionsschritt $1 < t \le T \; ist \; \alpha_t(j) = \sum_{i=1}^{S} (\alpha_{t-1}(i) * a_{ij} * b_i(o_t))$ berechnet.

Um den Algorithmus zu terminieren und die Wahrscheinlichkeit der Sequenz O zu erhalten muss man $\sum_{i=1}^{S} \alpha_T(i)$ berechnen. Die berechnete Summe ist die Lösung des Evaluierungsproblems.

2.3. Viterbi Algorithmus

Der Viterbi Algorithmus dient dazu, das zweite Problem bei Hidden Markov Modellen zu lösen[2]. Mit dem Algorithmus kann man die wahrscheinlichste Folge von Zuständen für eine gegebene Sequenz von Beobachtungen ermitteln. Zu diesem Zweck berechnet der Algorithmus eine Matrix V.

Um den Algorithmus zu initialisieren, werden die Zellen $V[i,1]$ der Matrix mit $\pi_i * b_i(o_1)$ gefüllt.

Der Rest der Matrix $V[j,t]$ wird mit $b_j(o_t) * max(a_{ij} * V[i,t-1])$ gefüllt. Damit man den Pfad bestimmen kann, muss für jedes Element in V vermerkt werden, welches Element an dem Maximum beteiligt war. Das notierte Element stellt den Vorgänger in einem Pfad dar.

Den wahrscheinlichsten Pfad bestimmt man zum Schluss indem man das Maximum $V[i,T]$ bestimmt und von diesem Element aus die notierten Vorgänger zurückverfolgt bis zu einem Element ohne Vorgänger. Mit dem Pfad und der Wahrscheinlichkeit des Pfades hat man eine Lösung für das Dekodierungsproblem.

Der Baum-Welch Algorithmus zum Training des Modells wird hier nicht näher erläutert, da er im Gegensatz zu den restlichen Algorithmen nicht im Applet implementiert wurde.

3. Umsetzung des Applets

In diesem Kapitel gehe ich näher darauf ein wie das Applet umgesetzt wurde. Dabei gehe ich näher auf den Aufbau der Benutzeroberfläche ein. Des Weiteren erläutere ich kurz wie die Algorithmen und die Monte Carlo Simulation zur Lösung des Evaluierungs- und Dekodierungsproblem implementiert wurden.

Die Veranschaulichung wurde in Form eines Java Applets umgesetzt. Das Applet wurde dabei mit AnyLogic erstellt. AnyLogic ist eine Simulationssoftware von XJ Technologies.[5] Die Software bietet einem die Möglichkeit, mit verschiedenen Elementen Simulationsmodelle zu erstellen. Die Modelle werden erstellt indem man die benötigten Elemente auf der Benutzeroberfläche nach Bedarf anordnet. AnyLogic generiert automatisch für die verwendeten Elemente Java Code. Aufgrund dessen kann man die Elemente mithilfe von Java Code nach Bedarf modifizieren.

Einer der Gründe aus dem ich mich für AnyLogic entschieden habe ist, dass man mit AnyLogic leicht eine grafische Benutzeroberfläche erstellen kann. Weiterhin habe ich bereits Erfahrung mit AnyLogic und Java, daher war es unnötig mich in eine Software oder Programmiersprache einzuarbeiten. Außerdem musste ich keine Lizenz für die Software beschaffen, da ich bereits von der Teilnahme bei Introduction to Simulation eine Educational Lizenz habe. Ein weiterer wichtiger Grund für die Nutzung von AnyLogic war die Tatsache, dass man die Modelle, die in AnyLogic erstellt wurden, als Java Applet exportieren kann.

3.1. Aufbau des Applets

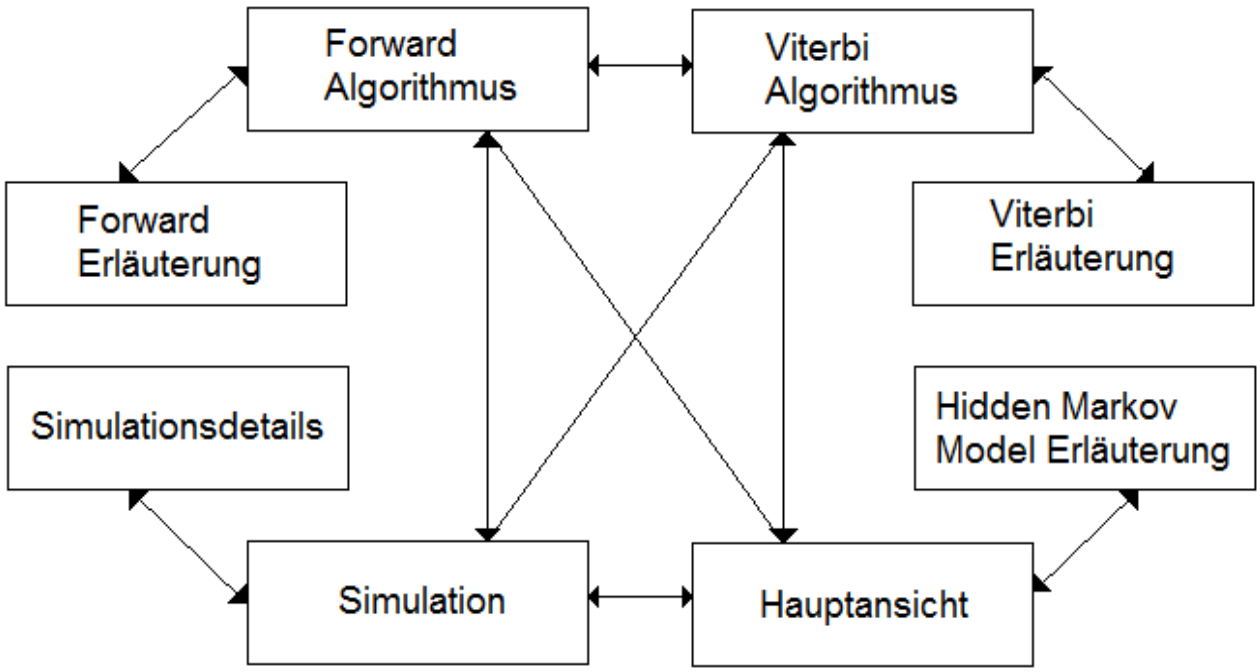


Abbildung 1. Anordnung der Ansichten des Applets

Abbildung 1 zeigt den groben Aufbau des Applets. Das Applet ist in mehrere Views aufgeteilt. Die Aufteilung ist notwendig um das gesamte Material zu organisieren. In Abbildung 1 ist erkennbar das man zwischen Hauptansicht, Forward Algorithmus und Viterbi Algorithmus frei hin und herwechseln kann. Zusätzlich hat jeder dieser Views ein zusätzlichen View mit Erläuterungen oder

zusätzlichen Details. Um zwischen den Views zu wechseln sind in jedem View verschiedene Buttons. Wenn das Applet gestartet wird startet der Nutzer in der Hauptansicht.

3.1.1. Aufbau der Hauptansicht

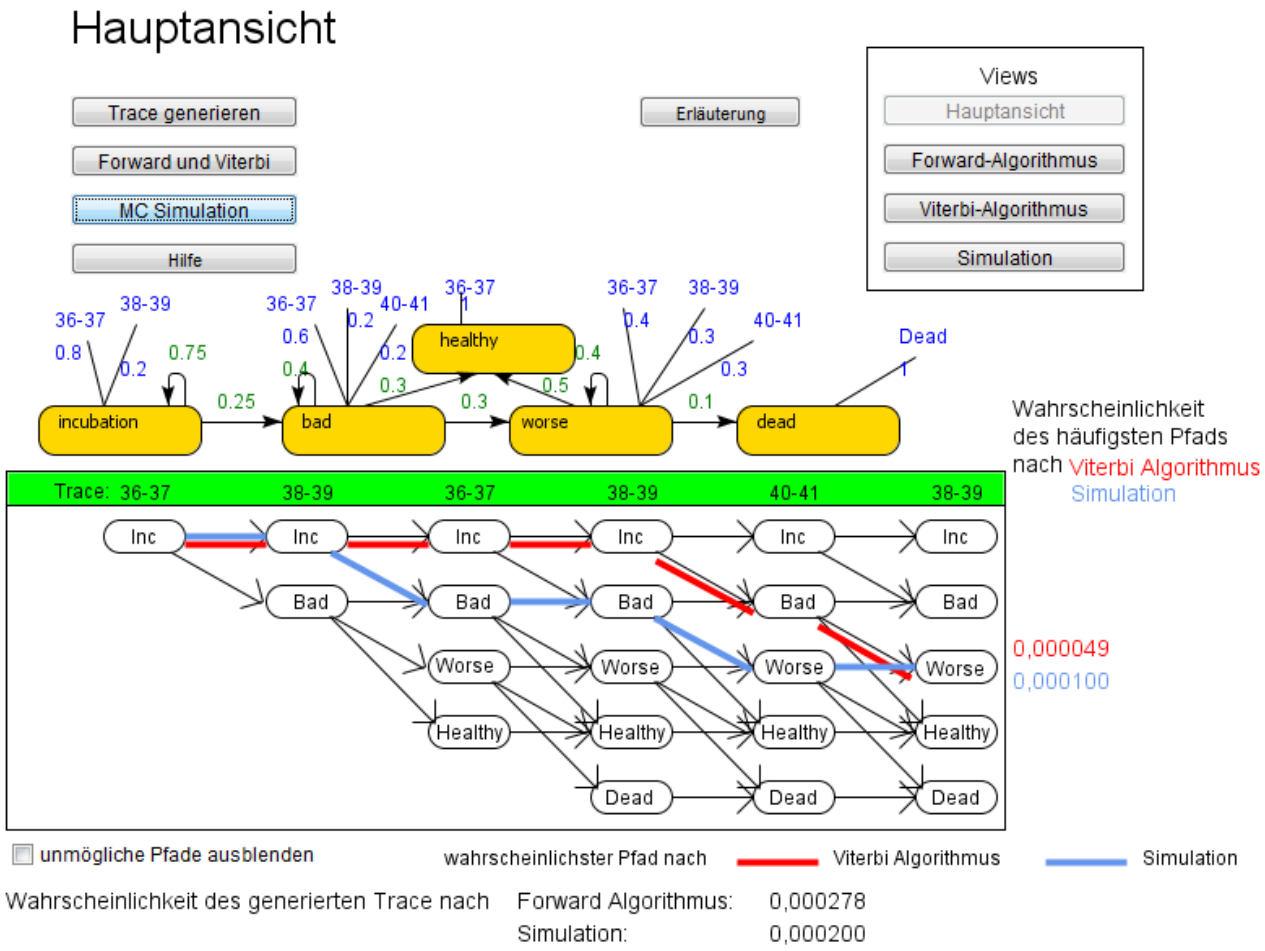


Abbildung 2. Screenshot der Hauptansicht

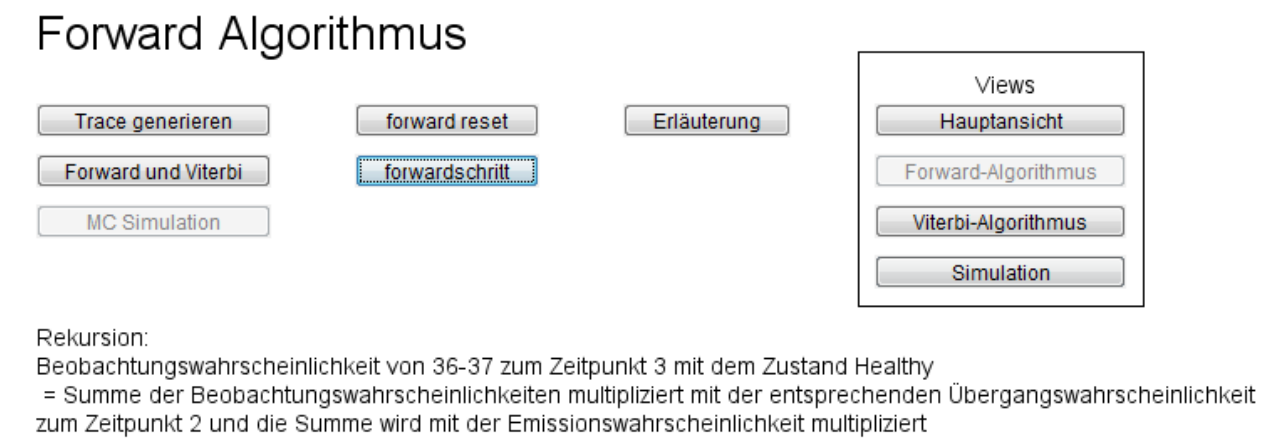
Die Hauptansicht bietet dem Nutzer eine Übersicht über das Modell. Dazu ist in der Hauptansicht eine Abbildung des implementierten Hidden Markov Modells. An dem Modell sind auch die Übergangswahrscheinlichkeiten zwischen den Zuständen und die möglichen Beobachtungen mit den Beobachtungswahrscheinlichkeiten zu sehen. Weiterhin ist eine Ansicht aller möglichen Pfade in dem Modell vorhanden. Ein Pfad ist eine Folge von Zuständen in dem Modell. Der Nutzer hat auch die Möglichkeit Pfade auszublenden die für einen erstellten Trace unmöglich sind. Der generierte Trace wird über den möglichen Pfaden angezeigt. Wenn der Viterbi Algorithmus für den Trace angewandt wurde, wird der wahrscheinlichste Pfad in der Ansicht rot hervorgehoben. Der bei der Monte Carlo Simulation am häufigsten aufgetretene Pfad wird blau markiert. Unter der Ansicht wird das Ergebnis des Forward Algorithmus und das Vergleichsergebnis der Simulation angezeigt. Rechts von der Ansicht wird das Ergebnis des Viterbi Algorithmus und das Vergleichsergebnis der Simulation angezeigt.

In Abbildung 2 sind am oberen Rand mehrere Buttons zur Steuerung zu erkennen. Der Button *„Trace generieren“* generiert einen Trace für das Implementierte Modell. Der Button *„Forward und Viterbi“* ist deaktiviert bis ein Trace generiert wurde. Wenn ein Trace generiert wurde, wendet der Button den Viterbi und den Forward Algorithmus auf den Trace an. Der Button *„MC Simulation“* dient dazu eine Monte Carlo Simulation zu starten. Der Button *„Hilfe“* blendet ein paar Hinweise über die Hauptansicht ein. Mit dem Button *„Erläuterung“* wechselt man zu einer

Ansicht mit einer allgemeinen Beschreibung von Hidden Markov Modellen.

Oben rechts sind die Buttons um zu den verschiedenen Views zu wechseln. Der Button für den View den man aktuell betrachtet ist deaktiviert.

3.1.2. Forward/Viterbi View



	Incubation	Bad	Worse	Healthy	Dead
36-37	0,800000	0,000000	0,000000	0,000000	0,000000
38-39	0,120000	0,040000	0,000000	0,000000	0,000000
36-37	0,072000	0,027600	0,004800	---	---
38-39	---	---	---	---	---
40-41	---	---	---	---	---
38-39	---	---	---	---	---

Abbildung 3. Screenshot des Forward Views

Für den Forward Algorithmus und dem Viterbi Algorithmus ist jeweils ein View vorhanden. Beide Views sind identisch aufgebaut. Da diese Views dem Nutzer helfen sollen die Algorithmen zu verstehen, bieten sie dem Nutzer detailliertere Informationen über die Berechnung der Algorithmen.

Am oberen Rand des Views findet man wie schon bei der Hauptansicht verschiedene Steuerbuttons. Unter anderem sind die Buttons *„Trace generieren“*, *„Forward und Viterbi“*, *„MC Simulation“* und die Buttons zum wechseln der Views, auch in diesem View zu finden. Zusätzlich zu diesen Buttons sind noch drei weitere Buttons vorhanden. Der Button *„forward reset“* bzw. *„viterbi reset“* setzt die Berechnung des entsprechenden Algorithmus zum Anfang zurück. Mit dem Button *„forwardschritt/viterbischritt“* kann der Nutzer einen einzelnen Berechnungsschritt des Algorithmus durchführen lassen. Der letzte Button, *„Erläuterung“*, führt zu einem View mit einer allgemeinen Beschreibung des Algorithmus.

Damit der Nutzer die Berechnung besser nachvollziehen kann, befindet sich unter den Buttons eine kurze Beschreibung des nächsten Berechnungsschrittes. Unter der Beschreibung wird der nächste Berechnungsschritt mit den zu verwendenden Werten angezeigt. Zu guter Letzt befindet sich am unteren Rand des Views eine Tabelle mit den Zwischenergebnissen des Algorithmus.

Trace	Zustände				
	Incubation	Bad	Worse	Healthy	Dead
36-37	0,800000	0,000000	0,000000	0,000000	0,000000
38-39	0,120000	0,040000	0,000000	0,000000	0,000000
36-37	0,072000	0,027600	0,004800	---	---
38-39	---	---	---	---	---
40-41	---	---	---	---	---
38-39	---	---	---	---	---

Abbildung 4. teilweise gefüllte Tabelle mit Zwischenergebnissen eines Algorithmus

Im Tabellenkopf stehen alle Zustände des Modells, am linken Rand wird der aktuelle Trace angezeigt. In der Tabelle werden die Zwischenergebnisse, beim Forward Algorithmus die Forward Variable und beim Viterbi Algorithmus der Inhalt der Matrix V , bis zu dem letzten durchgeführten Rechenschritt angezeigt.

3.1.3. Simulation

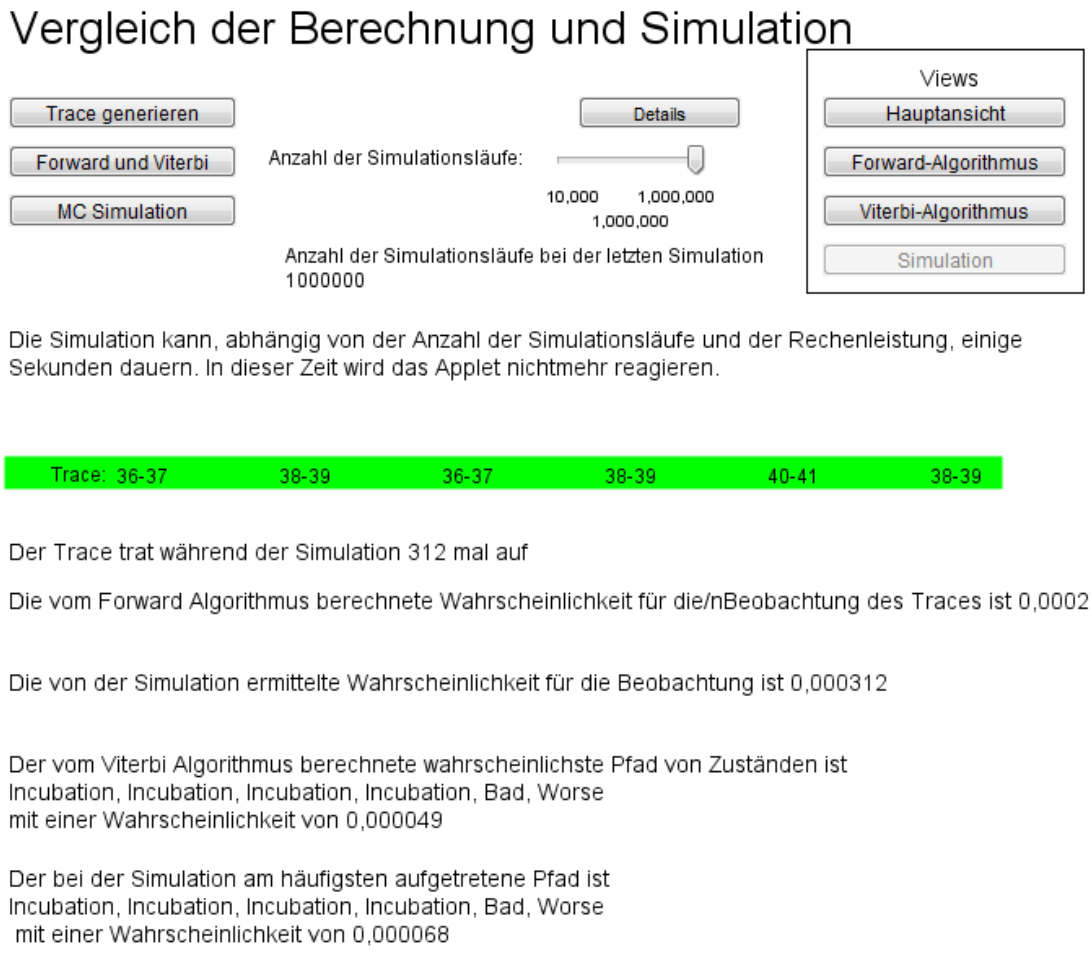


Abbildung 5. Screenshot des Simulation View

Der Simulation View dient dazu, die Ergebnisse der Algorithmen mit den Ergebnissen einer Monte

Carlo Simulation zu vergleichen. Zu diesem Zweck werden in diesem View die Endergebnisse des Algorithmus und die entsprechenden Simulationsergebnisse angezeigt. Weiterhin wird der aktuelle Trace, für den die Algorithmen und die Simulation durchgeführt wurde, angezeigt. Am oberen Rand des Views befinden sich, wie schon in den anderen Views, verschiedene Steuerelemente. Wie schon in der Hauptansicht und den Views für die beiden Algorithmen sind die Buttons zum wechseln der Views und die Buttons zum generieren eines Trace, zum anwenden der Algorithmen und zur Durchführung der Simulation zu finden. Zusätzlich findet man ein Slider Element mit dem man die Anzahl der durchzuführenden Simulationsläufe verändern kann. Es werden mindestens 10.000 und maximal 1.000.000 Simulationsläufe durchgeführt.

Außerdem ist der Button *„Details“* vorhanden. Durch diesem Button kann man zu einem View wechseln in dem man die Zwischenergebnisse der Algorithmen und die entsprechenden Ergebnisse der Simulation vergleichen kann. Die Zwischenergebnisse und die Ergebnisse der Simulation werden in dem Detailview in den gleichen Tabellen wie im Forward und Viterbi View angezeigt.

3.2. Wichtige Funktionen

In diesem Abschnitt werden die wichtigsten Funktionen des Applets erläutert. Hierbei beschränke ich mich auf Funktionen die zur Berechnung der Algorithmen und der Durchführung der Monte Carlo Simulation dienen. Andere Funktionen werden nicht näher betrachtet da sie zum Großteil von AnyLogic automatisch generiert wurden oder ihre Implementierung eher trivial ist.

3.2.1. Umsetzung des Viterbi und Forward Algorithmus

Für die Ausführung der beiden Algorithmen sind drei Funktionen implementiert. Die Funktionen führen nur die Initialisierung und die Rekursion durch, sie berechnen $\alpha_t(i)$ für den Forward Algorithmus und die Matrix $V[i,t]$.

Die erste Funktion ist *„forward_viterbi“*. Wenn diese Funktion aufgerufen wird, dann führt sie alle Initialisierungs- und Rekursionsschritte durch.

Die Initialisierung besteht aus einer for Schleife. In der Schleife wird für jeden möglichen Zustand die Startwahrscheinlichkeit mit der Beobachtungswahrscheinlichkeit für die erste Beobachtung im Trace multipliziert und in einem Array für jeden Algorithmus gespeichert.

Die Rekursion ist mit drei for Schleifen umgesetzt. Die erste Schleife steht für den Zeitpunkt. Die zweite Schleife steht für den Zustand für den der Rekursionsschritt durchgeführt wird. Mit der dritten Schleife wird das Maximum für den Viterbi Algorithmus und die Summe für den Forward Algorithmus bestimmt.

Das Applet hat zwei weitere Funktionen mit denen die Algorithmen berechnet werden können. Die Funktionen *„forwardschritt“* und *„viterbischritt“* ermöglichen es, die Algorithmen schrittweise zu berechnen.

3.2.2. Simulation

Die Simulation wurde mit der Funktion *„MCSim“* umgesetzt. Die Funktion generiert sich einen

Pfad von Zuständen und einen dazugehörigen Trace. Der Simulationstrace wird mit dem Trace den man auswerten möchte verglichen und ausgewertet.

Um das Evaluierungsproblem zu lösen muss gezählt werden wie oft die Simulation den gleichen Trace erzeugt. Zusätzlich wird auch gezählt wie oft die einzelnen Zustände zu welchem Zeitpunkt auftreten. Dabei wird ein Zustand nur gezählt wenn die Traces bis zu diesem Zeitpunkt übereinstimmen. Dadurch erhält der Nutzer Werte die er mit den Zwischenergebnissen des Forward Algorithmus vergleichen kann.

Für das Dekodierungsproblem wird der generierte Pfad gespeichert wenn beide Traces übereinstimmen. Außerdem wird gespeichert wie oft ein Pfad aufgetreten ist. Weiterhin werden Werte gesammelt die sich mit den Zwischenergebnissen des Viterbi Algorithmus vergleichen lassen.

Wenn man die gezählten Werte durch die Anzahl der durchgeführten Simulationsläufe teilt, erhält man Werte die man mit den berechneten Werten vergleichen kann.

Die restlichen Funktionen im Applet werden nicht weiter erläutert. Sie dienen hauptsächlich dazu die Ergebnisse anzuzeigen, auszuwerten und zu formatieren.

4. Funktionstest

In diesem Kapitel wird das Vorgehen bei einem Funktionstest des Applets erläutert. Dabei wird überprüft ob alle Elemente richtig angezeigt werden und ob die Ergebnisse der Algorithmen korrekt sind. Für einen Funktionstest wurde ein Beispieltrace aus der Vorlesung verwendet. Der Trace ist (36-37, 36-37, 36-37, 40-41, 38-39, 36-37). Dieser Trace wurde verwendet da so einige der Ergebnisse mit den Vorlesungsfolien verglichen werden können. Die Ergebnisse der Simulation werden mit den Ergebnissen der Algorithmen verglichen. Wenn die Algorithmen korrekt ausgeführt werden, kann davon ausgegangen werden, dass sich die Ergebnisse der Algorithmen und der Simulation ähnlich sind.

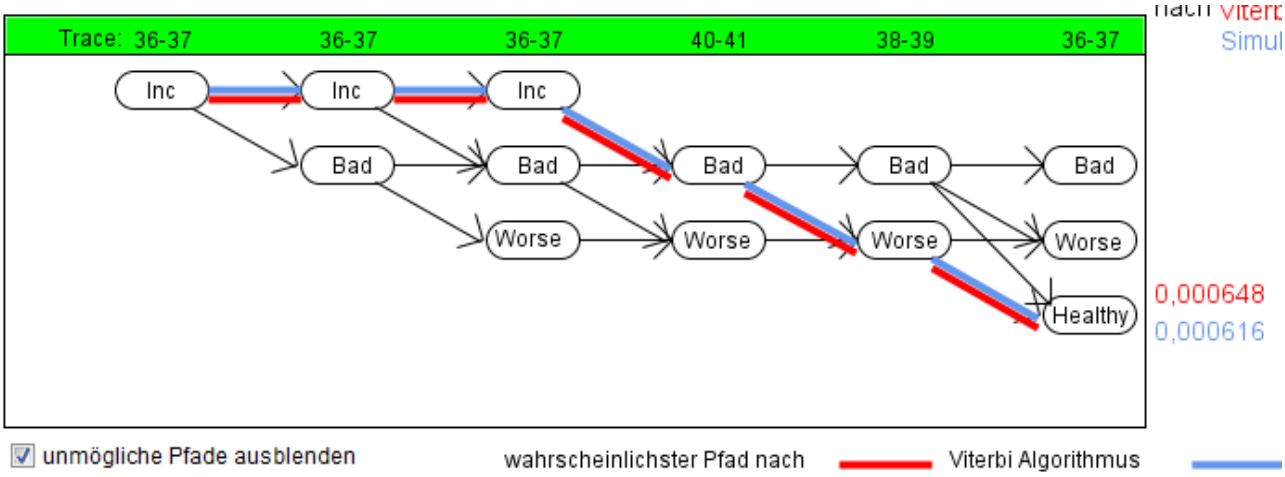


Abbildung 6. Pfadübersicht im Main View

Wie in Abbildung 6 zu sehen wird der Trace korrekt angezeigt. Durch aktivieren der Checkbox unter der Übersicht werden Pfade, die mit dem gegebenen Trace nicht erreicht werden können, ausgeblendet. Ein Vergleich mit den Vorlesungsfolien zeigt, dass die richtigen Pfade ausgeblendet werden. Weiterhin werden die Endergebnisse des Forward Algorithmus und des Viterbi Algorithmus richtig berechnet. Auch die entsprechenden Vergleichsergebnisse der Simulation erscheinen plausibel. Außerdem werden die vom Viterbi Algorithmus und von der Simulation ermittelten wahrscheinlichsten Pfade in der Übersicht farblich hervorgehoben, rot für den Algorithmus und blau für die Simulation.

Im Forward und im Viterbi View wird die Berechnung wie gewünscht zurückgesetzt wenn der entsprechende Button gedrückt wird. Bei der schrittweisen Berechnung wird die kurze Beschreibung des nächsten Schritts, sowie die Werte die für den nächsten Schritt verwendet werden korrekt angezeigt. Die berechneten Zwischenergebnisse in den Tabellen sind ebenfalls richtig.

Im Simulationsview werden die Ergebnisse korrekt dargestellt. Außerdem wird die Zahl der Simulationsläufe die mit dem Slider Element eingegeben wird übernommen und richtig angezeigt. Im Detailview der Simulation sind die simulierten Vergleichswerte für die Zwischenergebnisse der Algorithmen plausibel.

Nach diesem kurzen Funktionstest gehe ich davon aus, das die Algorithmen wie beabsichtigt funktionieren.

Um zu überprüfen ob das Applet dabei helfen würde Hidden Markov Modelle zu verstehen, wäre es notwendig das Applet von verschiedenen Personen testen zu lassen. Solche Tests wären jedoch sehr zeitaufwändig und konnten daher nicht durchgeführt werden.

5. Fazit

Es wurde ein Java Applet mit AnyLogic erstellt. In diesem Applet ist ein einfaches Krankheitsmodell implementiert. Nutzer des Applets können sich für das Krankheitsmodell einen Trace generieren lassen. Es ist möglich das Evaluierungs- und das Dekodierungsproblem für einen generierten Trace mit dem Forward und Viterbi Algorithmus zu lösen. Außerdem können die beiden Probleme mit einer Monte Carlo Simulation gelöst werden. Die Ergebnisse der Algorithmen können mit den Ergebnissen der Simulation verglichen werden. Des Weiteren sind Ansichten mit Erläuterungen zu Hidden Markov Modellen und den Algorithmen enthalten. Damit sind die definierten Aufgaben erfüllt.

Die gesetzten Ziele wurden ebenfalls erfüllt. Es wurde ein Applet erstellt, mit dem Hidden Markov Modelle veranschaulicht werden. Nutzer haben verschiedene Interaktionsmöglichkeiten mit dem Applet. Weiterhin kann mithilfe des Forward Algorithmus, des Viterbi Algorithmus und einer Monte Carlo Simulation das Evaluierungsproblem und das Dekodierungsproblem gelöst werden.

Das Applet kann auf der Webseite von Introduction to Simulation online gestellt werden. Studenten können das Applet dazu nutzen, um Hidden Markov Modelle besser zu verstehen.

Da für diese Arbeit keine neuen Algorithmen entworfen werden mussten, sondern nur welche implementiert werden mussten, war die Umsetzung recht einfach. Die größte Schwierigkeit an der Arbeit bestand darin, das Applet übersichtlich zu gestalten. Außerdem ist es nicht einfach die Modelle und die Algorithmen verständlich darzustellen.

Referenzen

[1] Graham Horton: Introduction to Simulation
Vorlesung an der Fakultät für Informatik der Otto-von Guericke Universität, WS 2011
http://www.sim.ovgu.de/Lehrstuhl+für+Simulation.html

[2] L.R. Rabiner und B.H. Juang: an Introduction to Hidden Markov Models
IEEE ASSP Magazine Januar 1986, Seite 4-16
http://luthuli.cs.uiuc.edu/~daf/courses/Signals%20AI/Papers/HMMs/01165342.pdf

[3] Gabrielle A Reeves**,** David Talavera and Janet M Thornton: Genome and proteome annotation: organization, interpretation and integration
Journal of the Royal Society Interface, Februar 2009, Seite 129-147
http://rsif.royalsocietypublishing.org/content/6/31/129.full

[4] Shalendra Chhabra, William Yerazunis, and Christian Siefkes: Spam filtering using a Markov random field model with variable weighting schemas
November 2004 IEEE International Conference on Data Mining

[5] XJ Technologies: AnyLogic
http://www.xjtek.com/

[6] W. Stewart: Introduction to the Numerical Solution of Markov Chains
November 1994, Princeton University Press
ISBN 978-0691036991