Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik
Institut für Simulation und Grafik

Masterarbeit

Vergleich klassischer Maschineller Lernverfahren mit Hidden non-Markovian Models anhand ausgewählter Anwendungsbeispiele

Verfasser:

Sascha Bosse

Eingereicht bei:

Prof. Dr.-Ing. habil. Graham Horton,
Prof. Dr. rer. nat. habil. Rudolf Kruse

29. September 2011

Betreuer:

Dr.-Ing. Claudia Krull

Universität Magdeburg
Fakultät für Informatik
Postfach 4120, D–39016 Magdeburg
Germany

Bosse, Sascha:
Vergleich klassischer Maschineller Lernverfahren mit Hidden non-Markovian Models anhand ausgewählter Anwendungsbeispiele
Masterarbeit, Otto-von-Guericke-Universität Magdeburg, 2011.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Abkürzungen und Symbole

CRF	Conditional Random Field
DecTree	Entscheidungsbaum/Decision Tree
HMM	Hidden Markov Model
HnMM	Hidden non-Markovian Model
ML	Maschinelles Lernen/Machine Learning
MLP	Mehrschichtiges Perzeptron/Multi-Layer-Perceptron
NN	Neuronales Netz
$\mathbb{N}_+$	Menge der positiven natürlichen Zahlen
$\mathbb{N}_0$	Menge der positiven natürlichen Zahlen und Null
$\mathbf{x}$	Vektor von Werten
X	Menge

Zusammenfassung

Hidden non-Markovian Models wurden in den letzten Jahren entwickelt, um die Lösungseigenschaften von Hidden Markov Models und die Modellierungsmöglichkeiten von Petri-Netzen zu vereinen. Sie können daneben auch als Verfahren zur Mustererkennung interpretiert werden. So modellieren sie einen stochastischen Prozess, um derartige Probleme zu beschreiben und bieten Algorithmen an, um diese zu lösen. Ihre Entwicklung ist dabei noch nicht abgeschlossen und aktueller Gegenstand der Forschung.

In dieser Arbeit wird die Fragestellung bearbeitet, ob andere, etablierte Verfahren zur Mustererkennung ebenfalls im Stande sind, diese Probleme zu lösen. Dazu wird versucht, Maschinelle Lernverfahren anzuwenden, die das Verhalten der zugrunde liegenden stochastischen Prozesse erlernen können. Dabei wird, neben der Betrachtung der Genauigkeit, auch eine Betrachtung der Geschwindigkeit und Handhabbarkeit dieser Verfahren vorgenommen.

Zu diesem Zweck werden in dieser Arbeit zwei Beispielanwendungen von Hidden non-Markovian Models vorgestellt und der Versuch unternommen, diese mit anderen Verfahren zu lösen.

Die Ergebnisse dieser Experimente zeigen, dass klassische Verfahren zwar schneller und leichter zu handhaben sind als Hidden non-Markovian Models, jedoch deren Genauigkeit nicht erreichen können.

Somit scheinen Hidden non-Markovian Models einen berechtigten neuen Ansatz im Bereich der Mustererkennung darzustellen, auch wenn sie noch einige Nachteile vorweisen. Nach weiterer Forschungsarbeit könnten die Hidden non-Markovian Models als vollwertiges Maschinelles Lernverfahren angesehen werden.

Kapitel 1

Einleitung

Dieses Kapitel soll dem Leser verdeutlichen, welche Motivation hinter der Bearbeitung der Problemstellung steht. Des Weiteren werden Ziele und Aufgaben der Arbeit identifiziert, bevor deren Struktur und wissenschaftlicher Kontext erläutert wird.

1.1 Warum Maschinen lernen müssen

Eine der wichtigsten Voraussetzungen für den Fortschritt der Menschheit ist das Sammeln und die Weitergabe von Wissen. Dieser Fortschritt wurde immer dann in der Geschichte beschleunigt, wenn es neue Möglichkeiten gab, Wissen zu erfassen und festzuhalten. Bereits vor 30 000 Jahren wurde Wissen in Form von Höhlenmalerei konserviert, später ermöglichte die Schrift das Niederschreiben von Wissen. Heute nutzt die Menschheit Computer, um Informationen in vorher unvorstellbaren Mengen abzuspeichern. Doch Informationen allein stellen noch kein Wissen dar und daher bleibt die Frage, wie eine Maschine Wissen sammeln und repräsentieren kann.

Wie kann Wissen nun gesammelt werden? Eine Antwort gibt der Aufklärer *Immanuel Kant*: „Alles Wissen stammt aus der Erfahrung“. Eine weitere der *Deutsche Duden*, der die Aneignung von Wissen als *Lernen* bezeichnet. Beide Definitionen sehen Wissen als Ergebnis eines Lernprozesses. Damit ein Computer Wissen generieren kann, muss er also lernen. Das heißt, aus den gesammelten Informationen automatisch Gesetzmäßigkeiten ableiten.

Diese Gesetzmäßigkeiten lassen sich mathematisch als Beziehung zwischen gewissen Ein- und Ausgabewerten definieren. Um diese Abhängigkeiten erfassen zu können, muss ein Lernalgorithmus Regelmäßigkeiten oder Muster in diesen Daten erkennen. Die Fähigkeit, diese Muster zu finden und automatisch bei neuem Input anzupassen, kann als Fähigkeit zu Lernen angesehen werden. Ein Beispiel für die Anwendung dieser Fähigkeit ist in Abbildung 1.1 gegeben: Die bezeichnenden Eigenschaften einer Darstellung der Zahl Drei können durch einen Lernalgorithmus so identifiziert werden, dass auch bei verrauschten Daten diese Zahl erkannt werden kann.

Verfahren, die dies ermöglichen, werden unter dem Begriff *Maschinelles Lernen* oder *Machine Learning* gesammelt [RN03]. Und obwohl sich diese Verfahren sehr in ihren Definitionen und theoretischen Einsatzgebieten unterscheiden, bleibt in der Praxis oft die Frage, welches von ihnen für ein spezielles Problem am besten geeignet ist.

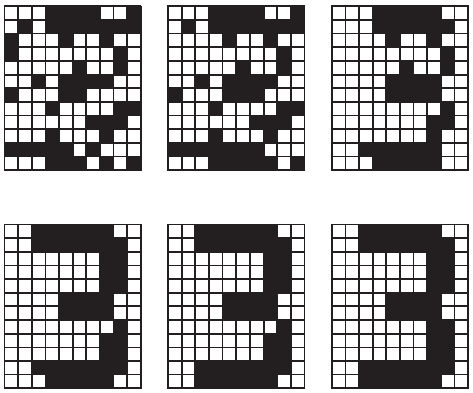

Abbildung 1.1: Anwendungsszenario Zeichenerkennung – Eine „Drei“, verschieden stark verrauscht

Hidden non-Markovian Models stellen einen relativ jungen Ansatz im Bereich der Mustererkennung dar. Sie lassen sich dabei als eine Erweiterung der populären *Hidden Markov Models* interpretieren und ermöglichen theoretisch die Analyse von Problemen eines breiteren Anwendungsbereiches. Damit erfüllen sie ähnliche Aufgaben wie Maschinelle Lernverfahren. Nachdem in den letzten Jahren dieses Verfahren formalisiert und erste Anwendungsbeispiele umgesetzt wurden, steht eine Beurteilung dieses neuen Ansatzes im Vergleich zu den verwandten Verfahren des Maschinellen Lernens noch aus.

1.2 Ziele und Nutzen – Sind Hidden non-Markovian Models in der Mustererkennung notwendig?

Ziel dieser Arbeit ist es, festzustellen, ob Hidden non-Markovian Models einen berechtigten Ansatz im Bereich der Mustererkennung darstellen. Dazu wird nicht nur untersucht, inwieweit die theoretische Definition ein einzigartiges Einsatzgebiet beschreibt, sondern auch ein Vergleich der verwandten Verfahren an konkreten Anwendungsbeispielen durchgeführt.

Dadurch soll diese Arbeit auch eine Entscheidungshilfe darstellen, also dem Leser die Vor- und Nachteile der einzelnen Verfahren und ihre Performanz in einem bestimmten Anwendungsgebiet darlegen. Dabei wird die Eignung eines Verfahrens über drei Kriterien bestimmt: Genauigkeit, Geschwindigkeit und Handhabbarkeit (vergleiche [ISO06]).

Ein Beispiel für ein System mit hoher Genauigkeitsanforderung ist die Überführung mutmaßlicher Straftäter durch DNA-Abgleich. Ein Verlust von Genauigkeit für dieses Problem ließe sich auch mit erhöhter Geschwindigkeit oder Handhabbarkeit nicht rechtfertigen.

Eine andere Problemklasse sind Echtzeitsysteme. Hier liegt der Fokus auf dem Geschwindigkeitsfaktor. Ein Spracherkennungssystem zum Beispiel sollte die Erfassung der gesprochenen Worte nicht unterbrechen, um die Genauigkeit der Zuordnung zu erhöhen.

Handhabbarkeit als subjektive Größe wird vor allem von Systemen mit Anwenderfokus verlangt. Eine Software zur Erkennung von Melodien durch Pfeifen sollte dem Nutzer

seine Komplexität möglichst verbergen, um keine potentiellen Anwender abzuschrecken.

Durch diese ganz unterschiedliche Fokussierung je nach Anwendungsbereich ist eine Einschätzung der Verfahren nach allen drei Kategorien erforderlich. Ein grundlegendes Entscheidungsmaß kann daher nicht angeboten werden. Vielmehr kann diese Arbeit die Basis bieten, für ein spezielles Problem nach Wichtung der Kriterien eine Abschätzung der Eignung vorzunehmen.

1.3 Aufgaben und Struktur der Arbeit

Um die Ziele dieser Arbeit zu erreichen, wird die folgende Vorgehensweise gewählt: Nachdem in diesem Kapitel eine Einleitung gegeben wurde, werden zunächst in Kapitel 2 die Prinzipien und einige Verfahren des Maschinellen Lernens theoretisch vorgestellt und eingeordnet. Darauf folgt die ausführliche Beschreibung der Hidden non-Markovian Models.

In Kapitel 3 werden Anwendungsbeispiele vorgestellt, in denen Hidden non-Markovian Models bisher eingesetzt wurden. Danach wird versucht, diese Anwendungen mit den vorgestellten Verfahren zu modellieren und umzusetzen. Darauffolgend werden die Ergebnisse dieser Umsetzungen vorgestellt und bewertet.

Das Schlusskapitel 4 fasst die Erkenntnisse dieser Arbeit zusammen und beurteilt diese. Zusätzlich werden offene aufgeworfene Fragen sowie die Bedeutung der Arbeit im Kontext des Forschungsgebietes in Form eines Ausblicks erörtert.

1.4 Wissenschaftlicher Kontext

Diese Arbeit wurde vom Lehrstuhl für Simulation an der Otto-von-Guericke-Universität Magdeburg in Auftrag gegeben. Ihr Zweck ist es, die Forschung im Bereich der Hidden non-Markovian Models weiter voran zu treiben. Zu den Veröffentlichungen in diesem Bereich vom Lehrstuhl für Simulation zählen [IWH06, KH09a, KH09b, KBH10, BKSH10] und [BKH11].

Kapitel 2

Überblick: Verfahren zur Mustererkennung

In diesem Kapitel werden die für das Verständnis der Arbeit notwendigen Grundlagen vermittelt. Zunächst wird zu diesem Zweck erörtert, inwiefern Maschinelle Lernverfahren mit Hidden non-Markovian Models konkurrieren können. Dazu wird der Kontext des Maschinellen Lernens beleuchtet und verschiedene Lernverfahren vorgestellt. Am Ende dieses Kapitels wird der Ansatz der Hidden non-Markovian Models ausführlich beschrieben. Somit wird das Verständnis der Umsetzungen der Verfahren in Kapitel 3 ermöglicht.

2.1 Einordnung: Hidden non-Markovian Models und Maschinelles Lernen

Hidden non-Markovian Models stellen eine Erweiterung der Hidden Markov Models – einem Maschinellen Lernverfahren – in dem Sinne dar, dass sie auch stochastische Prozesse mit der Markov-Eigenschaft modellieren können. Da sie jedoch nicht auf diese Prozesse beschränkt sind, sind sie theoretisch in einem größeren Einsatzgebiet anwendbar. Auch die vorgestellten Conditional Random Fields sind in der Modellierung der verborgenen stochastischen Prozesse auf Markovsche Prozesse beschränkt.

Die anderen vorgestellten Lernverfahren basieren nicht auf der Modellierung eines stochastischen Prozesses und sind damit in einer theoretischen Betrachtungsweise keine Konkurrenz für die Hidden non-Markovian Models. Auf einer anwendungsorientierten Ebene ist das Ziel der Anwendung dieser Verfahren jedoch in der Regel eine Entscheidungs- oder Zielfunktion zu finden. Diese Funktion kann auch von nichtprozessorientierten Verfahren approximiert werden.

Weiterhin wurde bisher keine allgemeine Methode entwickelt, Hidden non-Markovian Models zu trainieren. Daher wurden diese Modelle in der Vergangenheit manuell parametrisiert. Eine weitere Frage, die in dieser Arbeit zumindest aufgeworfen werden soll, ist deswegen, ob die Maschinellen Lernverfahren durch automatisiertes Training den Modellierungsaufwand für Hidden non-Markovian Models überflüssig machen.

Aufgrund dieser Tatsachen besteht die Möglichkeit, dass die vorgestellten Verfahren in bestimmten Anwendungen die Aufgaben der Hidden non-Markovian Models –

zumindest teilweise – erfolgreich bewältigen können, auch wenn diese Verfahren ganz verschiedene theoretische Einsatzgebiete definieren.

In den nächsten Abschnitten dieses Kapitels werden daher das Maschinelle Lernen und einige Verfahren desselben vorgestellt und formalisiert.

2.2 Maschinelles Lernen – Eine Definition

Als Maschinelles Lernen wird die Fähigkeit eines Computers bezeichnet, seine Aktionen so anzupassen, dass der Nutzen dieser maximiert wird [Mar09]. Mögliche Aktionen sind dabei zum Beispiel die Steuerung eines Roboters oder eine Vorhersage für die Zukunft. Diese Fähigkeit setzt voraus, dass ein solcher Computer Gesetzmäßigkeiten in der Beziehung zwischen bestimmten Aktionen und deren Nutzen erfasst. Aus mathematischer Sicht müssen dabei in den gesammelten Daten Muster und Abhängigkeiten gefunden werden. Damit ist dieser Sammelbegriff eng verwandt mit dem Begriff des *Data Mining*, der jedoch in der Regel einen größeren Prozess der Wissensgenerierung umfasst [HMS01]. Eine zweite Definition des Maschinellen Lernens fasst daher darunter Verfahren zusammen, die Muster in Daten erkennen und daraus Wissen generieren können [RN03].

Diese Verfahren werden zunächst abhängig von ihrer Form unterschieden (vergleiche [Mar09]): Den so genannten überwachten (engl. supervised) Verfahren stehen als „Erfahrung“ Eingabe- und Zielwerte zur Verfügung. Die Aufgabe dieser Verfahren ist es, diese gegebenen Beispiele so zu generalisieren, dass auch bei neuen Eingabewerten korrekte Zielwerte berechnet werden.

Im Gegensatz dazu stehen die unüberwachten (engl. unsupervised) Lernverfahren: Hier stehen nur Eingabewerte – in diesem Zusammenhang meist als *Features* bezeichnet [HTF01] – zur Verfügung, so dass diese Verfahren Ähnlichkeiten in den Daten finden, die eine gemeinsame Kategorisierung ermöglichen. Dieses Vorgehen ist vergleichbar mit der statistischen Dichteschätzung.

Die im weiteren Verlauf dieser Arbeit betrachteten Anwendungen, in denen Hidden non-Markovian Models eingesetzt wurden, entsprechen Problemen des überwachten Maschinellen Lernens. Dabei sind Beispiele gegeben, um den Zielwerten a priori eine Semantik zu verleihen. Eine Anwendung von unüberwachten Lernverfahren könnte erfolgreich sein, wird jedoch im Rahmen dieser Arbeit nicht durchgeführt. Daher werden im Folgenden überwachte Verfahren vorgestellt.

Formalisierung

Die folgende Definition überwachter Lernverfahren ist angelehnt an [HTF01]: Ein überwachtes Lernproblem lässt sich als Tupel (X, Y, T) beschreiben mit

- Eingabe $X = X_1 \times \cdots \times X_n$ als Menge der n-dimensionalen Vektoren $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ mit $n \in \mathbb{N}_+$,
- Ausgabe Y als Menge der m-dimensionalen Vektoren $\mathbf{y} = (y_1, y_2, \ldots, y_m)$ mit $m \in \mathbb{N}_+$ und
- Trainingssatz $T \subset X \times Y$ bestehend aus Trainingstupeln $(\mathbf{x}, \mathbf{y})$.

Die Aufgabe eines Lernverfahrens ist es dann, die explizit gegebene Relation T zu einer impliziten Relation F_ω, in der Regel eine Funktion $f_\omega : X \to Y$, zu verallgemeinern. Für beliebige Eingaben $\mathbf{x}$ kann so eine Ausgabe $\hat{\mathbf{y}}$ geschätzt werden. ω ist dabei das zugrunde liegende Modell, dessen Parameter angepasst werden, um das Trainingsziel zu erreichen. Dies wird durch eine Fehlerfunktion $e(\omega)$ bestimmt, meist die Minimierung des Fehlers $|\mathbf{y} - f(\mathbf{x})|$ über alle Beispiele [Alp10]. Um das so erhaltene Modell zu testen, wird ein Evaluationssatz $E \subseteq X \times Y$ mit $T \cap E = \emptyset$ angewandt [Bra07].

Die überwachten Lernverfahren werden traditionell in Regressions- und Klassifikationsverfahren eingeteilt [Mar09]. Diese Verfahren werden nach der Art ihrer Zielwerte unterschieden. Andere Verfahren basieren auf probabilistischen Modellen und können je nach Aufgabe als Regressions- oder Klassifikationsverfahren angesehen werden. Gemeinsam haben sie die Betrachtung der Eingabewerte als zeitlich geordnete Wertereihe und werden daher als Sequenzerkennungsverfahren bezeichnet [SG01]. Im weiteren Verlauf dieser Arbeit werden diese Verfahren als eigene Klasse betrachtet. In den nächsten Abschnitten werden diese Klassen und jeweils einige Verfahren vorgestellt.

2.3 Regression – Reelle Zielwerte

Regressionsverfahren werden eingesetzt, wenn angenommen werden kann, dass sich die gewünschte Ausgabe aus einer Kombination von Parameterfunktionen der Eingabewerte ergibt. Diese Funktionen werden so parametrisiert, dass eine Fehlerfunktion minimiert wird [BBHK10]. Die Ausgabewerte sind dabei reell oder anderweitig numerisch.

Einen ähnlichen Ansatz verfolgt die Interpolation, bei deren Einsatz eine Funktion gesucht wird, welche die Eingabedaten ohne Fehler beschreibt. Dies wird im Kontext des Maschinellen Lernens oft als „auswendig lernen“ bezeichnet und so kritisiert, dass reale Daten meist ungenau und fehlerhaft sein können. Um diesem Sachverhalt gerecht zu werden, streben Regressionsverfahren möglichst einfache funktionale Lösungen an – im Gegensatz zu Interpolationsverfahren, wo die Komplexität der Ergebnisfunktion von der Anzahl der Beispiele abhängt.

Im weiteren Verlauf dieses Abschnittes wird zunächst die Idee der linearen Regression vorgestellt, bevor künstliche Neuronalen Netze, die so genannten Mehrschichtigen Perzeptren, eingeführt werden. Diese stellen eine Möglichkeit dar, beliebige Funktionen zu approximieren. Nachfolgend wird vereinfacht davon ausgegangen, dass die Ausgabe aus einem einzelnen Ausgabewert besteht.

2.3.1 Lineare Regression

Die Hypothese der Linearen Regression lautet, dass sich die Ausgabe aus einer Linearkombination der Eingabewerte und einem inhomogenen Glied, dem so genannten Schwellenwert, ergibt (Formel 2.1). Die w_i werden dabei als Gewichte bezeichnet. Alternativ kann auch als Eingabevektor $(-1, x_1, \ldots, x_n)$ aufgefasst werden. Dann bildet der Parameter θ das Gewicht der konstanten Eingabe -1 ab [Mar09].

$$f(\mathbf{x}) = \hat{y} = -\theta + \sum_{i=1}^{n} w_i \cdot x_i \tag{2.1}$$

Zur Berechnung der Parameter dieses Verfahrens wird oft die Methode der kleinsten Quadrate angewandt. Diese garantiert eine optimale, algebraisch berechenbare Lösung für das Regressionsproblem, wenn die Fehlerfunktion die Summe der quadratischen Fehler darstellt [HTF01].

Anschaulich kann sich das Ergebnis im zweidimensionalen Fall vorgestellt werden: Es wird versucht, eine Gerade zu finden, welche die Abstände zu den Daten minimiert. Durch das Quadrieren des Fehlers wird nicht nur das Vorzeichen desselben egalisiert, sondern auch Differenzen kleiner Eins unter- sowie größer Eins überbewertet. Dies führt dazu, dass eine Vermeidung von größeren Abständen angestrebt wird. Dies kann zu unerwünschten Effekten führen, wenn in den Daten Ausreißer zu finden sind. Daher werden in der Praxis robustere Fehlerfunktionen verwendet [BBHK10].

Die Methode der kleinsten Quadrate lässt sich auf multilineare und polynomiale Funktionen erweitern. Auch ist es möglich, die Originalräume der gesuchten Funktion so zu transformieren, dass im transformierten Raum eine lineare beziehungsweise polynomiale Funktion gefunden werden kann.

2.3.2 Perzeptron

Motiviert von den biologischen Vorgängen im menschlichen Hirn, wo Neuronen die Speicherung und Verarbeitung von Informationen ermöglichen, entwickelten *McCulloch* und *Pitts* 1943 ein künstliches Neuron, das *McCulloch-Pitts-Neuron* [Mar09]. Ein solches Neuron besteht dabei aus gewichteten Eingabewerten $w_i \cdot x_i$, die aufsummiert mit einem Schwellenwert θ verglichen werden. Die Ausgabe ist dann eine Indikatorfunktion für den Vergleich, daher wird dieses Modell auch als Schwellenwertelement oder Perzeptron bezeichnet [BKKN03]. In Abbildung 2.1 ist ein solches Element grafisch dargestellt.

Ein so definiertes Perzeptron kann eingesetzt werden, um Daten zu klassifizieren, solang diese Klassen linear separierbar sind. Ein großer Vorteil ist dabei, dass die Parameter dieses Modells automatisch durch die so genannte *Delta-Regel* – auch als *Widrow-Hoff-Verfahren* bekannt – angepasst werden können, so dass sich der Fehler des Modells verringert [BKKN03].

Durch Anpassung der Ausgabefunktion kann ein Perzeptron auch Regressionsaufgaben lösen. Dazu wird die Indikatorfunktion für den Vergleich der Summe der gewichteten Eingänge mit dem Schwellenwert ersetzt durch die in Formel 2.1 dargestellte Funktion.

Nach diesen Anpassungen ist ein Perzeptron also ein Element, das eine (multi-)lineare Regression durchführt. Durch parallele Verknüpfung können linear separierbare Funktionen approximiert werden. Um jedoch komplexere Funktionen abbilden zu können, wurden weitere Entwicklungen vorgenommen.

2.3.3 Mehrschichtige Perzeptren – Neuronale Netze

Nach der Entwicklung einzelner Perzeptren schien es nur logisch, komplexe Sachverhalte ähnlich wie im Gehirn durch Netze von Neuronen zu lösen. Dabei sollte der Output eines Neurons wiederum Input eines anderen sein. Doch während die Ausgabewerte in diesen Netzen recht einfach berechnet werden können, konnte das Problem des Trainings der *Mehrschichtigen Perzeptren* (engl. *Multi-Layer-Perceptron* MLP) erst 1986 durch *Rumelhart*, *Hinton* und *McClelland* gelöst werden [Mar09].

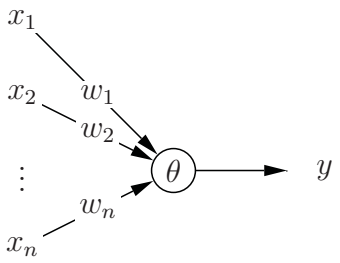

Abbildung 2.1: Grafische Darstellung eines allgemeinen Perzeptrons

Ein r-schichtiges Perzeptron besteht aus einer Eingangsebene mit n Neuronen, aus $r-2$ versteckten Ebenen und einer Ausgabeebene mit m Neuronen. Neuronen einer Schicht können dabei nur mit Knoten einer benachbarten Schicht verbunden sein, so entsteht eine Baumstruktur. In Abbildung 2.2 ist ein 3-schichtiges Perzeptron mit drei Eingangs- und zwei Ausgangsvariablen sowie vier versteckten Neuronen dargestellt.

Die Aktivierungsfunktionen der Neuronen, die den Einfluss des Schwellenwertes abhängig von den Eingaben festlegen, sind in diesem Netz nicht auf eine Schwellwertfunktion beschränkt, sondern können durch beliebige sigmoide Funktionen gebildet werden (vergleiche [BBHK10, BKKN03]).

Diese Änderung in der Definition der Neuronen ermöglicht, dass die Aktivierungsfunktionen differenzierbar sind und damit das entwickelte Trainingsverfahren anwendbar ist. Dieses besteht aus drei Schritten (nach [Mar09]):

(i) *Vorwärts-Propagation* - Hierbei werden die Ausgaben der einzelnen Neuronen Schicht für Schicht berechnet. Weicht die Ausgabe von den gewünschten Werten ab, muss der Fehler rückwärts durch das Netz propagiert werden

(ii) *Rückwärts-Propagation* - Ähnlich wie beim klassischen Perzeptron wird ein *Gradientenabstieg* durchgeführt, um die Fehlerfunktion zu minimieren. Dabei wird bei den Ausgabeneuronen angefangen und der Fehler rückwärts durch das Netz berechnet.

(iii) Nach Anpassung der Parameter wird geprüft, ob der Fehler unter die Zielschranke gefallen ist, ansonsten wird bei (i) fortgefahren

Ein weiterer Vorteil dieser Definition – neben dem möglichen Training – ist, dass ein MLP dazu in der Lage ist, jede Riemann-integrierbare Funktion beliebig genau zu approximieren. Damit sind Mehrschichtige Perzeptren sehr gut für komplexe Regressionsaufgaben geeignet.

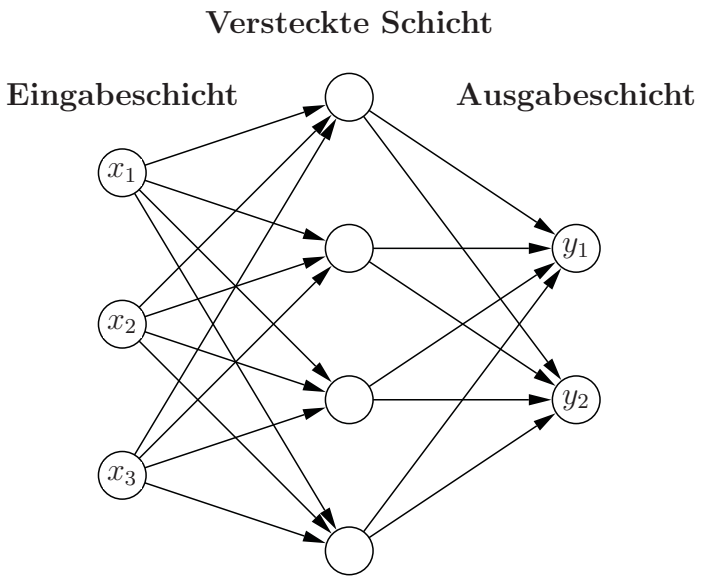

Abbildung 2.2: Beispielhaftes 3-schichtiges Perzeptron

2.4 Klassifikation – Symbolische Zielwerte

Die Bedingung für den Einsatz von Regressionsverfahren ist, dass die Zielwerte numerisch beziehungsweise reell sind. Maschinelle Lernverfahren, deren einziger Zielwert aus einer endlichen Menge stammt, werden als Klassifikatoren bezeichnet ($Y = C = \{c_1, c_2, \ldots, c_m\}$). Die einzelnen Elemente der Zielmenge werden als Klassen bezeichnet. Die Aufgabe eines Klassifikators ist es, für einen bestimmten Eingabevektor die korrekte Klasse zu bestimmen, also Entscheidungsregeln zu finden, um die Klassen voneinander abzugrenzen [Mar09].

Auch Neuronale Netze können als Klassifikatoren benutzt werden, zum Beispiel das im vorherigen Abschnitt eingeführte McCulloch-Pitts-Neuron. Allgemein ist es immer möglich, ein Regressionverfahren auf ein Klassifikationsproblem anzupassen, beispielsweise durch Diskretisierung der Zielwerte. Die im Folgenden vorgestellten Verfahren sind jedoch speziell für Klassifikationsprobleme entworfen worden.

2.4.1 Entscheidungsbäume

Ein einfacher Ansatz für Klassifikationsprobleme wäre ein Regelsatz, der angibt, wann ein Eingabevektor einer bestimmten Klasse zuzuordnen ist. Bei dieser Idee gibt es jedoch zwei Probleme: Überschneiden sich Regeln in ihren Prämissen, so ist unklar, welche Regel angewandt werden soll. Das zweite Problem ist, dass die sequentielle Überprüfung aller Regeln nicht sehr effizient ist.

Ein *Entscheidungsbaum* kann diese Probleme lösen. Dabei werden die Prämissen der Regeln hierarchisch angeordnet. So wird die Regelüberprüfung auf einen logarithmischen Aufwand reduziert. Ein (binärer) Entscheidungsbaum besteht dabei aus einer beliebigen Anzahl Entscheidungs- und Ergebnisknoten, wobei jeder Entscheidungsknoten zwei beliebige Nachfolgeknoten besitzt und jeder Ergebnisknoten ein Blatt dieses Baumes ist. Ein beispielhafter Entscheidungsbaum ist in Abbildung 2.3 dargestellt.

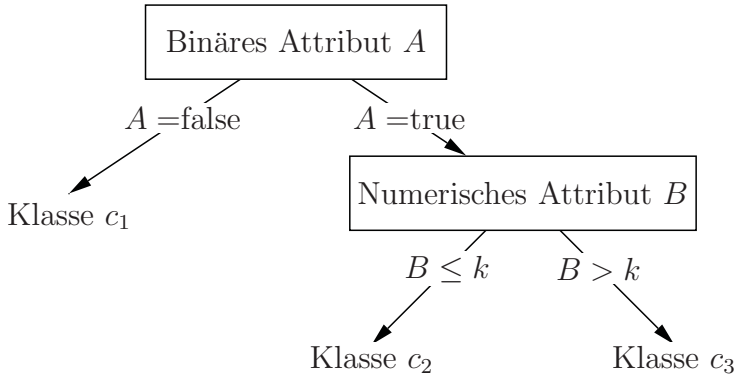

Abbildung 2.3: Grafische Darstellung eines beispielhaften Entscheidungsbaumes

Soll nun ein Eingabevektor klassifiziert werden, so wird zunächst das in der Wurzel des Baumes angegebene Attribut geprüft. Je nach konkreter Ausprägung dieses Attributs wird der Pfad zu einem der beiden Nachfolgeknoten eingeschlagen, für den die Überprüfung fortgesetzt wird. Ist ein Ergebnisknoten erreicht, wird die dort angegebene Klasse dem Eingabevektor zugeordnet. Diese Vorgehensweise ist auch für Laien leicht verständlich und stößt damit auf weniger Akzeptanzprobleme als beispielsweise Neuronale Netze, deren Semantik schwierig nachzuvollziehen ist [Mar09]. Daher ist es nicht verwunderlich, dass sich Entscheidungsbäume als ein Standardverfahren im Maschinellen Lernen und im Data Mining etabliert haben [BBHK10].

Mathematisch – angenommen, alle Eingabedimensionen sind reell – ist dies die Einteilung der Daten in Hyperquader. Diese werden durch Schnitte von (orthogonalen) Halbräumen geformt, die durch die Entscheidungsfunktion in jedem Entscheidungsknoten gebildet werden. Sind einige Eingabedimensionen symbolischer Natur, kann es sinnvoll sein, die Anzahl an Nachfolgeknoten beliebig zu setzen, um die Domäne komplett erfassen zu können.

Um einen Entscheidungsbaum „zu lernen“, wird er induktiv, also Ebene für Ebene, gebildet. Die Güte eines Entscheidungsbaumes bemisst sich dabei nicht nur an der Korrektheit der Klassenzuordnung, sondern auch an der Komplexität des Baumes. Beide Kriterien hängen im hohen Maße davon ab, welche Attribute wann geprüft werden. Daher muss eine Funktion definiert werden, die den Effekt der Auswahl eines Attributes abbildet; ein so genanntes Evaluationsmaß.

Der bekannteste Algorithmus für den induktiven Aufbau eines Entscheidungsbaumes ist der *ID3* [Mar09]. Evaluationsmaß ist dabei der *Information Gain*. Dieser gibt an, wie weit die *(Shannon-)Entropie* beziehungsweise die Unordnung der Daten verringert wird, wenn ein bestimmtes Attribut ausgewählt wird (vergleiche [BBHK10]). Der Datensatz wird dann entsprechend dem ausgewählten Vergleich getrennt und in den Nachfolgeknoten analog verfahren. Sinkt die Entropie dabei auf 0, ist der Datensatz „rein“, enthält also nur noch Vektoren mit gleicher Klassenzuordnung. Dann kann ein Ergebnisknoten erzeugt werden. Sind alle Attribute geprüft, aber die Entropie noch größer 0, wird die häufigste Klasse im übrig gebliebenen Datensatz zugeordnet.

Um die Daten auch mit diesem Verfahren nicht „auswendig zu lernen“ und gleichzeitig die Komplexität des Baumes zu reduzieren, wird das so genannte *Pruning* durchgeführt. Hierbei gibt es zwei Varianten: Bei der ersten wird bereits bei der Induktion des Baumes

ein Grenzwert gesetzt, der bestimmt, wann die Entropie des Datensatzes ausreicht, um einen Ergebnisknoten zu erzeugen. Dies wird als *Prepruning* bezeichnet. Ist der Baum auch nach dem Prepruning zu komplex, kann ein ungleich aufwändigeres *Postpruning* durchgeführt werden, das einzelne Knoten des Baumes löscht oder Teilbäume durch Ergebnisknoten ersetzt (vergleiche [BBHK10]).

Am Ende dieser Schritte steht ein Entscheidungsbaum, der einen Kompromiss aus der korrekten Klassifikation der Trainingsdaten sowie der Geschwindigkeit und Anschaulichkeit darstellt. Nachteile dieses Verfahrens sind die möglicherweise ausufernde Komplexität des Baumes sowie die komplexe Aufgabe, bei numerischen Attributen den besten binären Schnitt zu finden.

2.4.2 Bayes-Klassifikatoren

Ein weiter Ansatz, Daten zu klassifizieren, stammt aus der Wahrscheinlichkeitstheorie (vergleiche [BBHK10]). Dabei wird ein Datenpunkt der für ihn wahrscheinlichsten Klasse zugeordnet. Dazu muss theoretisch für jeden möglichen Datenpunkt $\mathbf{x}$ und jede Klasse c die Wahrscheinlichkeit der Zugehörigkeit $P(c|\mathbf{x})$ berechnet werden. Die Anzahl der zu speichernden Wahrscheinlichkeiten wächst jedoch selbst bei diskreten Eingabedimensionen exponentiell an, bei kontinuierlichen Dimensionen müssen theoretisch unendlich viele Werte gespeichert werden. Daher gehen *Bayes-Klassifikatoren* von bestimmten Annahmen aus, um die Anzahl zu speichernder Informationen zu minimieren. Je nach Art dieser Annahmen werden zwei Formen unterschieden: Der *Naive* und der *Volle Bayes-Klassifikator*. Gemeinsam haben sie – wie der Name suggeriert – die Verwendung des *Satzes von Bayes*.

Naiver Bayes-Klassifikator

Bei diesem Klassifikator gilt die Annahme, dass alle Eingabedimensionen, wenn nur einzelne Klassen betrachtet werden, unabhängig voneinander sind. Dies, kombiniert mit dem Baysschen Theorem, führt zu Formel 2.2. Da $P(\mathbf{x})$ dabei unabhängig von der Klasse ist, kann es als Normalisierungskonstante betrachtet werden und ist später für die Entscheidung der Klassifikation unerheblich.

$$P(c|\mathbf{x}) = \frac{P((x_1, x_2, \ldots, x_n)|c) \cdot P(c)}{P(\mathbf{x})} = \frac{\prod_{i=1}^{n} P(x_i|c) \cdot P(c)}{P(\mathbf{x})} \tag{2.2}$$

Statt nun für jeden Datenpunkt die Klassenwahrscheinlichkeiten zu speichern, müssen nur noch die $P(x_i|c)$ vorliegen. Diese werden je nach Datentyp der Dimension verschieden geschätzt: Besteht die Eingabedimension aus Symbolen, ist sie also diskret, so werden die Häufigkeiten in den Trainingsdaten als Schätzer gebraucht. Bei numerischen Attributen wird pro Dimension eine univariate Normalverteilung angenommen, Mittelwert und Varianz werden ebenfalls aus den Trainingsdaten geschätzt.

Sobald die Schätzwerte aus dem Trainingssatz berechnet wurden, können dem Klassifikator Eingabevektoren übergeben werden. Dieser bestimmt für jede Klasse und jedes Attribut die geschätzte unnormalisierte Wahrscheinlichkeit. Der Klasse mit dem höchsten Wert wird der Datenpunkt zugeordnet.

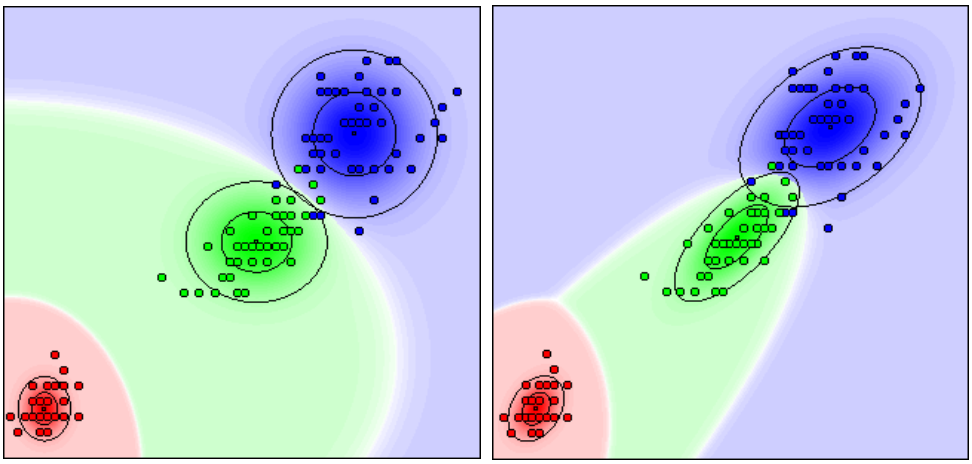

Abbildung 2.4: Beispielhafter Naiver (links) und Voller (rechts) Bayes-Klassifikator in zwei Dimensionen des IRIS-Datensatzes

Voller Bayes-Klassifikator

Wenn nicht-numerische Attribute ausgeschlossen werden können, kann die Verteilung der Klassen über die Eingabedimensionen auch als multivariate Normalverteilung angenommen werden. Dabei ermöglicht die Kovarianzmatrix dieser Verteilung die Erfassung von Korrelationen in den Eingabedaten. Dabei müssen für jede Klasse, ausgehend von den Trainingsdaten, der Mittelwertvektor und die Kovarianzmatrix geschätzt werden. Ist die Kovarianzmatrix eine Diagonalmatrix, so entspricht der Volle dem Naiven Bayes-Klassifikator.

In Abbildung 2.4 ist der Unterschied zwischen Vollem und Naivem Bayes-Klassifikator (als Screenshot aus dem Programm *BCView*[1]) dargestellt. Das zu lösende Problem ist die Zuordnung von Blüten zu den drei Arten der Schwertlilie/Iris [Fis36].

Dargestellt sind zwei Dimensionen der Klassifikatoren, die drei Farben Rot, Grün und Blau repräsentieren die verschiedenen Arten/Klassen. Deutlich ist zu sehen, dass die Normalverteilungen beim Naiven Bayes-Klassifikator orthogonal sind, während dies beim Vollen Bayes-Klassifikator nicht der Fall ist.

2.5 Sequenzerkennung – Partiell Beobachtbare Stochastische Prozesse

Die dritte Klasse der Maschinellen Lernverfahren ist nicht wie die beiden oben vorgestellten Klassen durch die Art der Ausgabewerte gekennzeichnet. Tatsächlich verbinden Sequenzerkennungsverfahren reelle sowie symbolische Ausgaben je nach Anwendung. Die Besonderheit dieser Verfahren liegt in der Tatsache begründet, dass sie Eigenschaften von stochastischen Prozessen nutzen, um ihre Berechnungen durchzuführen.

Sequenzen sind in diesem Zusammenhang als chronologisch sortierte Folgen von Symbolen zu verstehen. Ihre Entstehung lässt sich durch einen stochastischen Prozess beschreiben. Damit besteht eine Sequenz aus einer Folge von Zufallsvariablen $(X_{t_1}, \ldots, X_{t_T})$

[1] `http://www.borgelt.net/bcview.html`

mit $0 \leq t_1 \cdots \leq t_T$. Ein Sequenzerkennungsverfahren kennt dabei zwei Arten von Sequenzen: Die eine Art lässt sich als Eingabevektor $\mathbf{x}$ auffassen und besteht aus konkreten Beobachtungen des realen Systems. Die zweite Art sind Sequenzen $\mathbf{y}$, die nicht beobachtbar sind, aber von den Eingabesequenzen abhängen.

Ein Sequenzerkennungsverfahren modelliert diese beiden stochastischen Prozesse. Statt einer Funktion $f : X \to Y$ wird also eine Wahrscheinlichkeitsverteilung $p(\mathbf{x}, \mathbf{y})$ gesucht, welche die Abhängigkeiten zwischen den Dimensionen wiedergibt [Wal04]. Da diese Abhängigkeiten aufgrund der Autokorrelation in den Sequenzen sehr komplex werden können, treffen die hier vorgestellten Verfahren verschiedene Annahmen, welche die Berechnung dieser Verteilung vereinfachen sollen. Die beiden klassischen Sequenzerkennungsverfahren, *Hidden Markov Models* (HMMs) und *Conditional Random Fields* (CRFs), nehmen dabei an, dass der verborgene, nicht-beobachtbare Prozess ein *Markovscher Prozess* ist.

Markovsche Prozesse

Markovsche Prozesse sind nach *Andrej Markov* (1856-1922) benannt, der diese Theorie entwickelte [Fin08]. Sie sind stochastische Prozesse, in denen die *Markov-Eigenschaft* gilt. Diese Eigenschaft wird oft als „Gedächtnislosigkeit“ bezeichnet.

Im Folgenden wird die Klasse von Markov-Prozessen betrachtet, die als *Markov-Ketten* bezeichnet werden. Die Ausprägungen einer Markov-Kette sind dabei diskrete Zustände, gesammelt in der Menge $S = \{s_1, s_2, \ldots, s_m\}$. Es werden dabei zeitdiskrete und zeitkontinuierliche Markov-Ketten unterschieden. Bei ersteren Prozessen finden alle Ereignisse zu diskreten Zeitpunkten statt. Die Sequenz $(X_{t_1}, \ldots, X_{t_T})$ wird dann – da $t_k - t_{k-1} = 1$ – als $(X_1, \ldots, X_T)$ notiert. Ist die Größe der Zeit kontinuierlich, können Ereignisse zu allen nicht-negativen reellen Zeitpunkten stattfinden.

In Formel 2.3 ist die Markov-Eigenschaft erster Ordnung für eine zeitdiskrete Markov-Kette dargestellt [Fin08]. Die künftige Entwicklung des Prozesses hängt also nur vom aktuellen und dem letzten Zustand ab. Im zeitkontinuierlichen Fall hängt die Entwicklung zusätzlich von der seit dem letzten Ereignis vergangenen Zeit ab.

$$P(Y_k = s_k | Y_{k-1} = s_{k-1}, \ldots, Y_0 = s_0) = P(Y_k = s_k | Y_{k-1} = s_{k-1}) \tag{2.3}$$

In Abbildung 2.5 ist eine zeitdiskrete Markov-Kette dargestellt. Runde Knoten repräsentieren dabei die Zustände, die Kanten dazwischen die Möglichkeit des Zustandswechsels. Sie werden dazu mit Übergangswahrscheinlichkeiten $p_{ij} = P(X(t_{k+1}) = s_j | X(t_k) = s_i)$ mit $0 \leq i, j \leq m$ assoziiert.

Formal lässt sich eine zeitdiskrete Markov-Kette als Tupel (S, P, π) beschreiben. S ist dabei die oben erwähnte Menge an Zuständen, $P \in \mathbb{R}^{m \times m}$ die stochastische Matrix der Übergangswahrscheinlichkeiten. $\pi \in \mathbb{R}^m$ ist die Anfangsverteilung. Aufgrund der Markov-Eigenschaft lässt sich die Verteilung im k-ten Schritt durch $\pi(k) = \pi(k-1)^T \cdot P$ berechnen.

Im zeitkontinuierlichen Fall enthält die Matrix P keine Übergangswahrscheinlichkeiten, sondern Übergangsraten λ. Die Zeit von einem Zustand zum nächsten ist exponentialverteilt mit Parameter λ. Die Veränderung der Verteilung wird durch $\frac{d\pi}{dt} = \pi^T \cdot P$ berechnet.

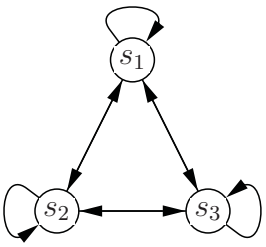

Abbildung 2.5: Grafische Darstellung des Zustandsraums einer Markov-Kette mit drei Zuständen

Aufgaben der Sequenzerkennung

Wie oben erwähnt, modelliert ein Sequenzerkennungsverfahren zwei stochastische Prozesse, die eine beobachtbare und eine nicht-beobachtbare Sequenz erzeugen. Dabei sind die Abhängigkeiten der beobachtbaren Symbole in der Regel zu komplex, um sie in einem einzelnen Modell zu erfassen. Daher werden die (vereinfachten) Abhängigkeiten zwischen diesen beiden stochastischen Prozessen ausgenutzt [SM06]. Durch diese Abstraktionen können die folgenden Aufgaben durch ein Sequenzerkennungsverfahren bearbeitet werden [Fin08]:

Die Aufgabe, von einer gegebenen Observationssequenz $\mathbf{x}$ auf die wahrscheinlichste nicht-beobachtbare Sequenz von Zuständen $\mathbf{y}$ zu schließen, wird als *Dekodierung* bezeichnet (Formel 2.4).

$$\text{decode}(\mathbf{x}) = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{2.4}$$

Eine weitere Aufgabe, die ein Sequenzerkennungsverfahren lösen kann, ist die Wahrscheinlichkeit, dass eine gegebene Sequenz x überhaupt vom Modell ω erzeugt wurde, zu bestimmen. Dieser Vorgang wird als *Evaluierung* bezeichnet (Formel 2.5).

$$\text{eval}(\mathbf{x}, \omega) = P(\mathbf{x}|\omega) \tag{2.5}$$

Während also die Aufgabe der Dekodierung nur innerhalb eines Modells sinnvoll anwendbar ist, ermöglicht die Evaluierung auch den Vergleich mehrerer Modelle. Damit kann die Aussage verbunden werden, welches Modell die Sequenz wahrscheinlich erzeugt hat, also eine Klassifizierungsaufgabe gelöst werden. Andererseits wird für ein einzelnes Modell bei dieser Aufgabe ein reeller Wert berechnet. Dies kann als Regressionsaufgabe aufgefasst werden. Sequenzerkennungsverfahren verbinden damit Aspekte der Klassifikation und der Regression.

Die dritte Aufgabe eines Sequenzerkennungsverfahrens ist das Lernen der Parameter, das *Training* (Formel 2.6). Dabei soll im Ergebnis die Wahrscheinlichkeit zur Erzeugung der Sequenz im angepassten Modell höher sein.

$$\text{train}(\mathbf{x}, \omega) = \omega' \text{ mit } P(\mathbf{x}|\omega) \leq P(\mathbf{x}|\omega') \tag{2.6}$$

Nachdem die Aufgaben eines Sequenzerkenners umschrieben wurden, werden im Folgenden zwei der bekanntesten Verfahren in dieser Klasse vorgestellt: Hidden Markov Models und Conditional Random Fields.

2.5.1 Hidden Markov Models

Hidden Markov Models stellen einen bewährten Ansatz zur Sequenzerkennung dar und wurden bereits in vielen Anwendungen eingesetzt [MZ97]. Beispielhaft sei hier die Spracherkennung zu erwähnen (z.B. [Eul06]). Dabei besteht die beobachtbare Sequenz aus aufgenommenen Lauten, die ein Sprecher von sich gegeben hat. Ein Hidden Markov Model hat in diesem System die Aufgabe, aus dieser Sequenz von Lauten auf eine Sequenz von Silben oder Wörtern zu schließen.

Dabei werden zwei Eigenschaften vorausgesetzt: Erstens ist der nächste Laut nur vom vorangegangenen abhängig. Zweitens hängt dieser Laut nur davon ab, welches Wort oder welche Silbe tatsächlich gesagt wurde. Diese beiden Eigenschaften kennzeichnen in abstrahierter Form jedes HMM: Der verborgene Prozess ist wertdiskret und Markovsch und die Verteilung der Observationen hängt nur vom aktuellen Zustand ab [Fin08].

Formell ist ein (zeitdiskretes) Hidden Markov Model ein Tupel (S, O, A, π, B) mit

- $S = (s_1, s_2, \ldots, s_m)$ als die Menge von Zuständen der verborgenen Markov-Kette,
- $O = (o_1, o_2, \ldots, o_n)$ als Menge an beobachtbaren Symbolen,
- $A = (a_{ij})_{1 \leq i,j \leq m}$ als Matrix der Übergangswahrscheinlichkeiten a_{ij} in der Markov-Kette,
- $\pi^T = (\pi_1, \pi_2, \ldots, \pi_m)$ als Anfangsverteilung und
- $B = (b_{jk})_{1 \leq i \leq m, 1 \leq j \leq n}$ als Matrix der Ausgabewahrscheinlichkeiten $b_{ij} = P(x_k = o_j | y_k = s_i)$ (vergleiche [Fin08]).

Hidden Markov Models werden oft grafisch dargestellt, entweder über ihren Zustandsraum oder als probabilistisches Netz der Zufallsvariablen. Dabei entspricht die erstere Darstellung der einer Markov-Kette. Zusätzlich wird jedem Zustand die diskrete Verteilung der Ausgabesymbole zugeordnet. Die letztere Darstellung repräsentiert über graphentheoretische Aspekte die Abhängigkeiten der Zufallsvariablen X_k und Y_k. Deutlich wird durch diese Darstellung, dass ein Hidden Markov Model ein dynamisches *Bayessches Netz* ist [Mar09].

Erweitert werden kann die obige Definition, um zeitkontinuierliche HMMs abzubilden. Dabei wird die Matrix A wie bei einer entsprechenden Markov-Kette mit Raten gefüllt. Eine andere Erweiterung ermöglicht die Verarbeitung numerischer Beobachtungssequenzen, indem in der Matrix B für jeden Zustand eine stetige Wahrscheinlichkeitsverteilung angegeben wird.

Für die Evaluation einer beobachteten Sequenz ist der *Vorwärts-Algorithmus* geeignet. Hierbei wird die Sequenz induktiv durchlaufen. Für jedes Symbol wird die Wahrscheinlichkeit der Erzeugung durch verschiedene Zustände addiert. Somit werden alle möglichen Zustandspfade des Modells durchlaufen.

Eine ähnliche Vorgehensweise verfolgt der *Viterbi-Algorithmus* zur Dekodierung. Statt jedoch die Wahrscheinlichkeiten der verschiedenen Pfade aufzusummieren, werden die Einzelwahrscheinlichkeiten in jedem Schritt gespeichert und so kann am Ende der wahrscheinlichste Pfad durch *Backtracking* rekonstruiert werden.

Zum Trainieren eines HMMs gibt es zwei grundlegende Ansätze. Die erste entspricht der oben definierten Lernaufgabe (Formel 2.6), implementiert durch den *Baum-Welch-Algorithmus*. Dieser ist ein *Expectation-Maximization(EM-)Algorithmus* und erhöht die Wahrscheinlichkeit des Modells, die gegebene Sequenz erzeugt zu haben, lokale Optima können dabei nicht ohne Eingriffe verlassen werden [Fin08].

Ein weiterer Trainingsansatz ist das *Viterbi-Training*. Hierbei wird nicht die Erzeugungswahrscheinlichkeit des gesamten Modells, sondern nur die des wahrscheinlichsten Pfades optimiert [Fin08].

2.5.2 Conditional Random Fields

Conditional Random Fields wurden in den letzten 20 Jahren entwickelt und sollen eine Alternative zu Hidden Markov Models darstellen. Während letztere *generative Modelle* sind, also die Erzeugung der Observationen durch eine Markov-Kette beschreiben, sind Conditional Random Fields *diskriminative Modelle*. Dies bedeutet, sie modellieren nicht die Verbundwahrscheinlichkeit $p(\mathbf{x}, \mathbf{y})$, sondern direkt die bedingte Wahrscheinlichkeitsverteilung $p(\mathbf{y}|\mathbf{x})$ [Wal04].

Dargestellt werden CRFs über einen so genannten Faktorgraphen, einem ungerichteten probabilistischem Netz mit Markov-Eigenschaft. Dies ist ein bipartiter Graph (V, F, E) mit $V = X \cup Y$ und $E \subseteq V \times F$. In Abbildung 2.6b ist ein solcher Graph dargestellt: Die Faktorknoten werden dabei durch ausgefüllte Quadrate dargestellt, Eingabeknoten durch Kreise und Ausgabeknoten durch unausgefüllte Rechtecke.

Jeder Faktorknoten F_A ($A \subset X \cup Y$) definiert ein so genanntes *(Markov) Random Field*. Dies ist eine Exponentialverteilung $\psi_A(\mathbf{x}_A, \mathbf{y}_A) = \exp(\sum_i \theta_{A_i} \cdot f_{A_i}(\mathbf{x}_A, \mathbf{y}_A))$. $\boldsymbol{\theta}_A$ ist dabei ein reellwertiger Parametervektor und $\{f_{A_i}\}$ eine Menge von so genannten *Feature-Funktionen*. x_A und y_A sind die Teilsequenzen, welche die mit dem Faktorknoten verbundenen Zufallsvariablen enthalten.

Mithilfe dieser Faktoren kann die Verbundwahrscheinlichkeit wieder zusammen gesetzt werden (Formel 2.7 mit C Normalisierungskonstante) und die bedingte Verteilung berechnet werden (Formel 2.8). $C(\mathbf{x})$ ist dabei eine instanzspezifische Funktion zur Normalisierung (vergleiche [SM06]).

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{C} \prod_A \psi_A(\mathbf{x}_A, \mathbf{y}_A) \tag{2.7}$$

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{p(\mathbf{y}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}', \mathbf{x})} \\ &= \frac{1}{C(\mathbf{x})} \prod_A \exp(\sum_i \theta_{A_i} \cdot f_{A_i}(\mathbf{x}_A, \mathbf{y}_A)) = \frac{1}{C(\mathbf{x})} \exp(\sum_A \sum_i \theta_{A_i} \cdot f_{A_i}(\mathbf{x}_A, \mathbf{y}_A)) \end{aligned} \tag{2.8}$$

Für Sequenzerkennungsaufgaben werden *lineare CRF-Ketten* (*Linear-Chain Conditional Random Fields*) benutzt. Dies sind CRFs, in denen eine Ausgabevariable nur mit maximal einer Vorgänger- und einer Nachfolgeausgabevariable verbunden sein darf. Die Random Fields der Verteilung können dann über Formel 2.9 beschrieben werden. $\mathbf{x}_t$ ist dabei der Vektor aller Eingabevariablen, die zum Zeitpunkt t zur Berechnung benötigt werden.

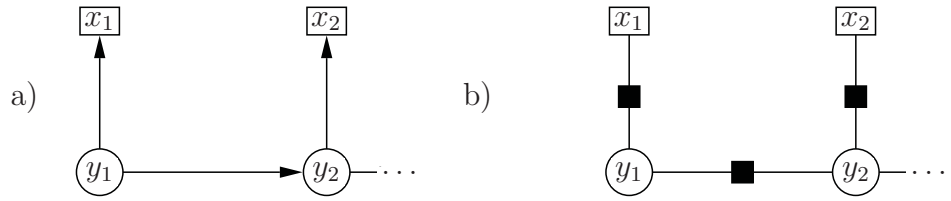

Abbildung 2.6: Probabilistisches Netz eines a) Hidden Markov Models und b) des korrespondierenden Conditional Random Fields

$$\psi_t(y_t, y_{t-1}, \mathbf{x}_t) = \exp(\sum_k \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)) \tag{2.9}$$

Lineare CRF-Ketten sind damit eng verwandt mit Hidden Markov Models. Sie bilden ein diskriminativ-generatives Paar [Wal04], woraus folgt, dass jedes HMM in eine CRF-Kette überführt werden kann. In Abbildung 2.6a ist ein Hidden Markov Model, in Abbildung 2.6b ist die entsprechende lineare CRF-Kette grafisch dargestellt. Umgekehrt gilt diese Aussage nicht, die CRF-Kette aus Abbildung 2.7 verdeutlicht dies: Hier sind erweiterte Abhängigkeiten zwischen Ausgabe- und Eingabeknoten modelliert. In diesem Beispiel hängt die Verteilung für y_2 nicht nur vom Vorgangszustand y_1, sondern zusätzlich von der Observation x_1 ab. Derartige Abhängigkeiten können von einem Hidden Markov Model nicht erfasst werden.

Trotz dieser erweiterten Abhängigkeiten lassen sich die Aufgaben der Sequenzerkennung in linearen CRF-Ketten durch Adaptionen der HMM-Algorithmen effizient lösen. Trainiert werden können diese Ketten, indem die Parameter θ_k so geschätzt werden, dass die Summe über die logarithmisierte bedingte Wahrscheinlichkeit $p(\mathbf{y}|\mathbf{x})$ für alle Trainingsdaten $(\mathbf{x}, \mathbf{y})$ maximiert wird. Dies wird in der Regel mit numerischen Optimierungsverfahren, wie zum Beispiel dem *Gradientenabstiegsverfahren*, durchgeführt.

Der größte Vorteil der Conditional Random Fields ist die Möglichkeit, sehr komplexe Abhängigkeiten in den Daten zu erfassen. Auch die linearen Ketten ermöglichen trotz ihrer Restriktionen unter anderem voneinander abhängige Observationsvariablen. Dies ermöglicht die Erfassung mehrerer abhängiger Features in einem Modell. Hidden Markov Models können solche Features nur unabhängig modellieren.

Ein großer Nachteil der CRFs ist ihre relative Inflexibilität aufgrund der nicht modellierten Verbundverteilung. Dies hat den Effekt, dass zum Beispiel bei Observationssequenzen, denen nur teilweise Ausgabewerte zugeordnet sind, generative Verfahren wie HMMs besser abschneiden. Sind die Sequenzen ohne gewünschte Ausgabe gegeben, muss ein CRF unüberwacht lernen. Dieses Problem ist jedoch noch Gegenstand der Forschung (vergleiche [SM06]).

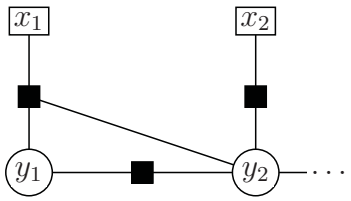

Abbildung 2.7: Lineare CRF-Kette mit erweiterten Abhängigkeiten

2.6 Hidden non-Markovian Models – Ein neuer Ansatz

Hidden non-Markovian Models wurden in den letzten Jahren entwickelt, um auch Sequenzen zu verarbeiten, die von nicht-Markovschen Prozessen erzeugt wurden. Sie wurden schrittweise aus zeitkontinuierlichen Hidden Markov Models entwickelt: Zunächst wurden die Observationen nicht mehr als Ergebnis eines Zustands, sondern eines Zustandsüberganges modelliert (*Extended HMMs* aus [KH07]). Jedes Element der beobachteten Sequenz wird daher um einen Zeitstempel erweitert. Die Verteilung der Observationen hängt dabei nur vom aktuellen Zustandsübergang ab. Des Weiteren ist die Verteilung der Zufallsgröße „Zeit bis zum Zustandswechsel“ nicht auf Exponentialverteilungen beschränkt. Somit kann der verborgene Prozess in Hidden non-Markovian Models (HnMMs) ein beliebiger, wertdiskreter stochastischer Prozess sein.

Definition

Hidden non-Markovian Models werden nach drei Aspekten kategorisiert: Emittieren alle Zustandsübergänge Observationen oder nur einige (*Eall* vs. *Esome*)? Gibt es nur eine Möglichkeit des Überganges zwischen zwei Zuständen oder mehrere (*SConeT* vs. *SCnT*)? Werden bei einem Zustandsübergang die Aktivierungszeiten der Zustandsübergänge gelöscht oder gespeichert (*Treset* vs. *Tkeep*) [KH09a]?

Je nach Kategorie unterscheiden sich die Modelle in gewissen theoretischen Aspekten. Im Folgenden wird eine Definition für den Fall (Eall, SConeT, Treset) gegeben: Ein solches Hidden non-Markovian Model ist ein Tupel (S, C, O, A, B, π) mit

- $S = (s_1, s_2, \ldots, s_m)$ als Menge der Zustände,
- $O = (o_1, o_2, \ldots, o_n)$ als Menge der Ausgabesymbole,
- $C \subseteq S^2$ als Menge der möglichen Zustandsübergänge,
- $A : \mathbb{R} \to \mathbb{R}^{m \times m}$ gibt für jeden Zeitpunkt $t \in \mathbb{R}$ die Matrix der Übergangswahrscheinlichkeiten an,
- $B : C \to (O \to \mathbb{R})$ gibt für jeden Zustandsübergang die Verteilung der Ausgabesymbole an und
- $\pi \in \mathbb{R}^n$ als Anfangsverteilung für die Zustände ($\pi = \pi(t_0)$).

Im Falle *Esome* ist die Funktion B nicht links-total. Bei der Eigenschaft *SCnT* kann die Menge C auch mehrere Paare von gleichen Zuständen enthalten. Gilt *Tkeep*, muss ein so genannter *Alters-Vektor*, der die Aktivierungszeiten enthält, gespeichert werden.

A lässt sich auch als Matrix von zeitabhängigen Funktionen beschreiben. Diese Funktionen werden als *instantaneous rate functions* (IRFs) bezeichnet und sind wie folgt definiert: $\mathrm{irf}(t) = \mu(t) = \frac{\mathrm{pdf}(t)}{1-\mathrm{cdf}(t)}$. *pdf* bezeichnet dabei die Dichtefunktion (englisch: probability density function) einer Zufallsvariable und *cdf* die entsprechende Verteilungsfunktion (englisch: cumulative distribution function).

Die IRF einer Transition gibt die Rate an, dass diese Transition zu einem Zeitpunkt t feuert, unter der Bedingung, dass bis dahin kein Übergang stattgefunden hat. Im Falle einer Exponentialverteilung wäre diese Rate konstant [Hor02].

Durch Integration der IRF ließe sich die Wahrscheinlichkeit eines Zustandsüberganges bis zu einem bestimmten Zeitpunkt bestimmen, für jeden einzelnen Zeitpunkt wäre diese Wahrscheinlichkeit jedoch theoretisch 0 [KH09a]. Da jedoch die Zeitpunkte für mögliche Zustandswechsel a priori durch die Observationssequenz vorliegen, lässt sich die bedingte Wahrscheinlichkeit dieses Wechsels durch Formel 2.10 berechnen [KH09b].

$$P(y_k = (c_{ij}, t_k) | x_k = (o, t_k)) = \pi_i(t_{k-1}) \cdot \frac{\mu_{ij}(t_k - t_{k-1})}{\sum_{s_k \in S} \mu_{ik}(t_k - t_{k-1})} \tag{2.10}$$

Diese Wahrscheinlichkeit kann – bei Modellen mit *Treset* – benutzt werden, um die Algorithmen der Hidden Markov Models für Evaluation, also den *Vorwärts-Algorithmus*, anzupassen. Der *Viterbi-Algorithmus* zur Dekodierung kann nur bei der zusätzlichen Eigenschaft *Eall* angepasst werden.

Speichert ein Modell die Aktivierungszeiten (*Tkeep*), so können die HMM-Algorithmen nicht angepasst werden. Berechnungen für diese Modelle werden durchgeführt, indem der Zustandsraum des Hidden non-Markovian Model modelliert wird. Der so entstehende Baum wird als *Proxel-Baum* bezeichnet [KH08]. Jeder mögliche Zustandspfad des Modells wird dabei durch einen *Proxel* (probabilistisches Element) repräsentiert. Dabei werden in dem Proxel wichtige Informationen, wie der bisherige Zustandspfad und dessen Wahrscheinlichkeit sowie zusätzliche Variablen wie die Aktivierungszeiten der Transitionen, gekapselt. Dieser Baum wird induktiv, also Ebene für Ebene beziehungsweise Zeitpunkt für Zeitpunkt, aufgebaut. Die Zeitdifferenz zwischen zwei Proxelebenen muss dabei so gewählt werden, dass nur ein Ereignis in diesem Schritt möglich ist.

Die Stärke dieser Methode ist die Traversierung aller möglichen Zustandspfade des Modells. Wie der Zustandsraum kann jedoch auch dieser Baum exponentielles Wachstum besitzen und damit extensive Berechnungen erfordern. Trotz Vereinfachungen und Verbesserungen ist daher dieser Algorithmus nur bei Modellen mit relativ wenigen Zuständen sinnvoll anwendbar [BKSH10].

Um ein Hidden non-Markovian Model zu trainieren, steht der *Baum-Welch-Algorithmus* nicht zur Verfügung. Dieser passt zur Optimierung die Übergangswahrscheinlichkeiten an, jedoch wurde noch keine Möglichkeit entwickelt, die IRF und damit die Verteilung der Aktivierungszeiten sinnvoll anzupassen [KH09a]. Eine Möglichkeit, diese Verteilungen anzupassen, ist sie als Markov-Kette mit erweitertem Zustandsraum, als eine so genannte *Phasentyp-Verteilung*, zu modellieren. Der Zeitraum, in der die

Wahrscheinlichkeitsmasse in den Zuständen der Phasentyp-Verteilung verbleibt, kann dabei jeder nicht-negativen Verteilung beliebig genau entsprechen, wenn entsprechend viele Zwischenzustände eingeführt werden. So ist es möglich, die Trainingsalgorithmen für HMMs zumindest für einzelne Verteilungen anzuwenden.

Kapitel 3

Experimente

In diesem Kapitel werden die vorgestellten Verfahren an zwei Anwendungsbeispielen getestet. Zunächst erfolgt jedoch eine Beschreibung der Voraussetzungen für die durchgeführten Experimente. Später werden die untersuchten Problemstellungen zunächst beschrieben, bevor die Umsetzung der verschiedenen Verfahren vorgestellt wird.

Die Ergebnisse dieser Anwendungen werden in den vorher erwähnten Kategorien Genauigkeit, Geschwindigkeit und Handhabbarkeit eingeteilt, wobei je nach speziellem Problem verschiedene Maße benutzt werden. Danach folgt eine Bewertung der verschiedenen Lösungen.

Abschließend werden die Ergebnisse der Experimente zusammengefasst und Schlussfolgerungen gezogen.

3.1 Rahmenbedingungen der Experimente

In diesem Abschnitt werden die Rahmenbedingungen für die Experimente erläutert. Dies schließt die Nennung aller genutzten Hilfsmittel und die Beschreibung der Experimentierumgebung ein.

Um die Verfahren vor allem hinsichtlich ihrer Laufzeit vergleichen zu können, wurden alle Experimente unter denselben Voraussetzungen durchgeführt. Dazu stand eine Maschine mit einem *Intel Pentium Dual Prozessor* (2.16 GHz) und 4 GiB Arbeitsspeicher zur Verfügung. Alle genutzten und entwickelten Implementierungen wurden dabei so gewählt, dass sie auf einem *Windows 7* Betriebssystem (64 Bit) in der virtuellen Java-Umgebung 1.6.0 ausgeführt werden konnten.

Zur Konfiguration der genutzten Bibliotheken und Ausführung der entwickelten Programme wurden die Entwicklungsumgebungen *AnyLogic 6.6.0 University Edition* (`http://www.xjtek.com/`) und *Eclipse Helios Service Release 2* (`http://www.eclipse.org/`) genutzt.

Während die Lösungen für die Hidden non-Markovian Models – basierend auf der gegebenen Definition – im Rahmen dieser Arbeit selbst entwickelt wurde, wurden für andere Verfahren bereits vorhandene Implementierungen genutzt. Diese sind

- für das Mehrschichtige Perzeptron *Christian Borgelts MLP* (`http://www.borgelt.net//mlp.html`),

- für die Bayes-Klassifikatoren *Christian Borgelts Bayes* (`http://www.borgelt.net//bayes.html`),
- für den Entscheidungsbaum *Christian Borgelts DTree* (`http://www.borgelt.net//dtree.html`),
- für Hidden Markov Models *Jahmm* (`http://code.google.com/p/jahmm/`) und
- für Conditional Random Fields *Sunita Sarawagis CRF Project* (`http://crf.sourceforge.net/`).

Diese Implementierungen wurden für die speziellen Probleme konfiguriert, deren Bearbeitung in den folgenden Abschnitten dokumentiert wurde.

3.2 Rekonstruktion eines Maschinenverhaltens

In diesem Anwendungsbeispiel ist das Ziel die Zuordnung von Produkten zu den sie erzeugenden Maschinen. Diese Zuordnung basiert auf den Zeitstempeln dieser Produkte, da diese ansonsten nicht unterscheidbar sind. Die Definition und Lösung dieses Problems mit Hidden non-Markovian Models werden in [BKSH10] beschrieben.

3.2.1 Problembeschreibung – Können zwei Maschinen unterschieden werden?

Es seien zwei Maschinen gegeben, die Teil eines Produktionsprozesses sind. Dieser Prozess erfordert die Verarbeitung von Rohlingen zu Produkten, die beide Maschinen so durchführen, dass eine spätere Unterscheidung aufgrund der Beschaffenheit der Produkte unmöglich ist. Die Maschinen unterscheiden sich jedoch in ihrer Verarbeitungszeit, möglicherweise aufgrund eines Technologieunterschieds.

Um Fehlproduktionen zu erkennen, befindet sich nach der Zusammenführung des Outputs beider Maschinen ein Prüfgerät. Dieses entscheidet, ob ein Produkt „OK“ oder „Defect“ ist und notiert den Zeitstempel der Prüfung. Um zum Beispiel abnorme Fehlerwahrscheinlichkeiten einer Maschine zu erkennen, kann es nötig sein, die Einträge im Prüfungsprotokoll wieder den einzelnen Maschinen zuzuordnen. Schematisch ist dieses Verhalten in Abbildung 3.1 dargestellt.

Die Verarbeitungszeit der beiden Maschinen ist dabei zufällig und normalverteilt. Die erste Maschine benötigt im Mittel 150 Sekunden mit einer Standardabweichung von 25 Sekunden, die zweite Maschine im Mittel 120 Sekunden mit einer Standardabweichung 20 Sekunden. Dabei liegt die Wahrscheinlichkeit beider Maschinen, defekte Produkte herzustellen, bei fünf Prozent.

Durch die Überschneidung der beiden Verteilungen kann aus einem gegebenen Protokoll keine eindeutige Zuordnung der Produkte zu den erzeugenden Maschinen erfolgen. Vielmehr sind mit zunehmender Dauer des Protokolls zunehmend mehr Kombinationen möglich. Ein Verfahren, das eine Zuordnung vornehmen will, müsste aus den möglichen Kombinationen den wahrscheinlichsten Verlauf identifizieren.

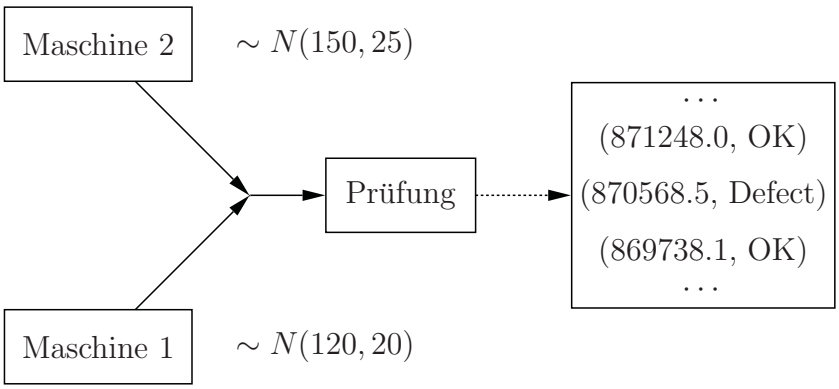

Abbildung 3.1: Schematische Darstellung der Protokollerzeugung für das Maschinenproblem

Um die angewandten Verfahren zu testen und gegebenenfalls zu trainieren, wurden – basierend auf dem oben definierten Modell – zufällige Protokolle erzeugt. Um den Effekt der stochastischen Einflüsse zu nivellieren, wurde bis zur Zeit von 10 000 Sekunden simuliert, so dass im Schnitt 148 Produkte protokolliert wurden. Für das Training und zum Vergleich der späteren Ergebnisse ist jedem Eintrag in den erzeugten Protokollen die verarbeitende Maschine zugeordnet. So wurden drei Trainings- und drei Testsequenzen erzeugt.

3.2.2 Entwickelte Lösungen

In den folgenden Abschnitten dieses Kapitels werden die Lösungen beschrieben, die mithilfe der verschiedenen vorgestellten Verfahren entwickelt wurden. Die Ergebnisse der einzelnen Lösungen werden dann nach dieser Beschreibung in Abschnitt 3.2.3 verglichen.

Hidden non-Markovian Model

Das Hidden non-Markovian Model, welches in [BKSH10] zur Lösung dieses Problems entwickelt wurde, hat die folgende Struktur: Es besteht aus einem einzigen Zustand, der immer wieder erreicht wird. In diesem Zustand sind zwei reflexive Zustandsübergänge möglich, welche die beiden Maschinen repräsentieren. Die Aktivierungszeit dieser Transitionen ist dabei verteilt wie die Bearbeitungszeiten der jeweiligen Maschine. Wenn eine der beiden Transitionen feuert, wird Zeitstempel und Symbol (*OK* oder *Defect*) ausgegeben. Die Aktivierungszeit der Transition, die nicht gefeuert hat, wird dabei jedoch gespeichert, so dass beim nächsten Produkt die Feuerwahrscheinlichkeit dieser Transition erhöht wird. Dieses Modell ist in Abbildung 3.2 dargestellt.

Aufgrund dieser Struktur hat dieses Hidden non-Markovian Model die Eigenschaften *Eall*, *Tkeep* und *SCnT*. Um nun die Dekodierung einer gegebenen Sequenz durchzuführen und so die wahrscheinlichste Zuordnung der Einträge zu den Maschinen vorzunehmen, muss das Proxel-Verfahren angewandt werden.

Um den Aufwand dieses Verfahrens so gering wie möglich zu halten, wurde eine untere Wahrscheinlichkeitsschranke eingeführt: Falls ein Proxel eine geringere Wahrscheinlich-

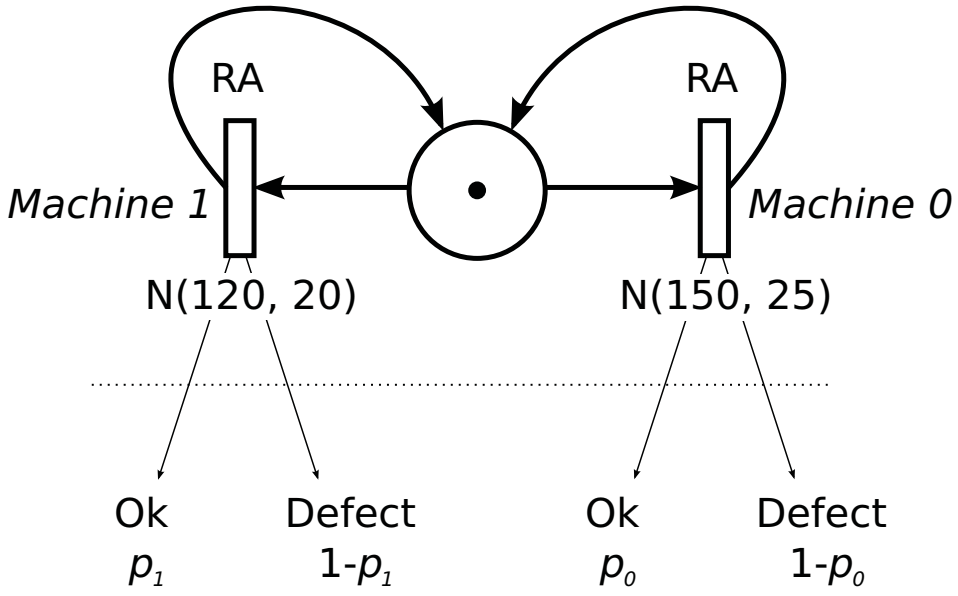

Abbildung 3.2: Maschinenverhalten: Hidden non-Markovian Model aus [BKSH10]

keit besitzt, wird er verworfen. Um jedoch bei extrem seltenen Ereignissen nicht alle Proxel zu löschen, wird ein Iterationsschritt mit leerer Ergebnismenge ohne untere Schranke wiederholt. Um dem Effekt der immer geringer werdenden Wahrscheinlichkeiten mit zunehmender Sequenzlänge entgegen zu wirken, werden weiterhin die Wahrscheinlichkeiten aller Proxel für eine Iterationsebene normiert.

Die Verteilungen der Feuerzeiten wurden dabei aus den Trainingsdaten geschätzt. Die Annahme einer Normalverteilung führte zu den Ergebnissen $N(121, 22.3)$ für Maschine 1 und $N(149, 27.1)$ für Maschine 2. Mit diesen Parametern konnte nun das Modell initialisiert werden und Testsequenzen übergeben werden.

Diese Sequenzen konnten dann mittels des Proxel-Algorithmus dekodiert werden. Dabei konnten in der Regel über 80 % der Produkte korrekt zugeordnet werden.

Regressionsverfahren

Um dieses Problem mit einem Regressionsverfahren zu lösen, musste zunächst ein funktionaler Zusammenhang zwischen Ein- und Ausgabe identifiziert werden. Da für diese Verfahren keine sinnvollen Initialwerte angegeben werden können, die bereits das Problem grob umschreiben, wurden mehrere Datenmodelle entwickelt. Diese sollten dazu dienen, dem Verfahren durch Vorverarbeitung der Daten Informationen über deren Abhängigkeiten zu übergeben.

Das erste benutzte Datenmodell besteht dabei aus den Rohdaten der erzeugten Protokolle. Eingabedimensionen sind der reelle Zeitstempel und ein Symbol für den Zustand des Produkts, Ausgabe das Symbol für die jeweilige Maschinenzuordnung. Die zu approximierende Funktion ist dann eine Entscheidungsfunktion, die bei Zielwerten kleiner 0.5

Dimesionen	Zeitdifferenz zum vorherigen Eintrag	Zustand des Produktes	Zielwert
			Maschine
Beispielwerte	58.1	*Defect*	*M1*
	15.4	*OK*	*M2*
			Maschinenpaar
Beispielwert	15.4	*OK*	*M1-M2*

Tabelle 3.1: Maschinenszenario: Alternative Datenmodelle für das MLP

Maschine 1 und bei Werten größer 0.5 Maschine 2 klassifiziert.

Da bei dieser Problemstellung nur die Zeitdifferenzen zwischen den Protokolleinträgen eine Entscheidung unterstützen können, wird für das zweite Datenmodell statt der absoluten Zeit die Differenz zum vorherigen Zeitstempel zusammen mit dem Zustand des Produkts übergeben. Ziel ist wiederum eine Klassifikation zu Maschine 1 oder 2.

Das dritte Datenmodell besteht aus denselben Eingabewerten wie das zweite Datenmodell, aber einer anderen Klassifizierung. Da dem Verfahren die Information fehlt, dass ein Wechsel der Maschinen wahrscheinlicher wird, je öfter eine Maschine klassifiziert wird, repräsentieren die Klassen hier Maschinenpaare. Klassen sind also „M1-M1“, „M1-M2“, „M2-M1“ und „M2-M2“.

Beide Datenmodelle sind beispielhaft in Tabelle 3.1 dargestellt. Weiterhin wird deutlich, wie aus den Daten des zweiten Modells die Eingabedaten des dritten berechnet werden. Dazu werden immer zwei aufeinanderfolgende Tupel betrachtet und der Wechsel der Zuordnung im Zielwert dokumentiert.

Für jedes dieser Datenmodelle wurden dreischichtige Perzeptren trainiert. Dabei wurde die Anzahl der versteckten Neuronen variiert, um optimale Ergebnisse zu erreichen. Dabei wurden mit vier versteckten Neuronen circa 58 % der Tupel korrekt klassifiziert.

Klassifikatoren

Für die Klassifikatoren wurden die oben beschriebenen Datenmodelle ebenfalls genutzt. Mit den vorverarbeiteten Daten wurden dann Naive Bayes-Klassifikatoren sowie Entscheidungsbäume trainiert.

Da der Volle Bayes-Klassifikator reelle Eingabedimensionen verlangt, musste die Dimension *Zustand des Produkts* aus den Daten entfernt werden. Damit war jedoch der Vorteil des Vollen gegenüber dem Naiven Klassifikator egalisiert, nämlich die Betrachtung von Korrelationen in den Eingabedimensionen. Aus diesem Grund wurde hier auf den Einsatz eines Vollen Bayes-Klassifikators verzichtet.

Für die Entscheidungsbäume wurden dabei zusätzliche Sequenzen erzeugt, um neben Training und Test auch das *Pruning* beziehungsweise das Stutzen der trainierten Bäume zu ermöglichen. Nach der Induktion bestand der Baum aus 89 Knoten auf 22 Ebenen, der nach dem Pruning auf 59 Knoten in 17 Ebenen reduziert werden konnte.

Während der Bayessche Klassifikator im besten Fall mehr als 55 % der Daten korrekt zuordnen konnte, gelang dies dem trainierten Entscheidungsbaum nur in circa 45 % der Fälle.

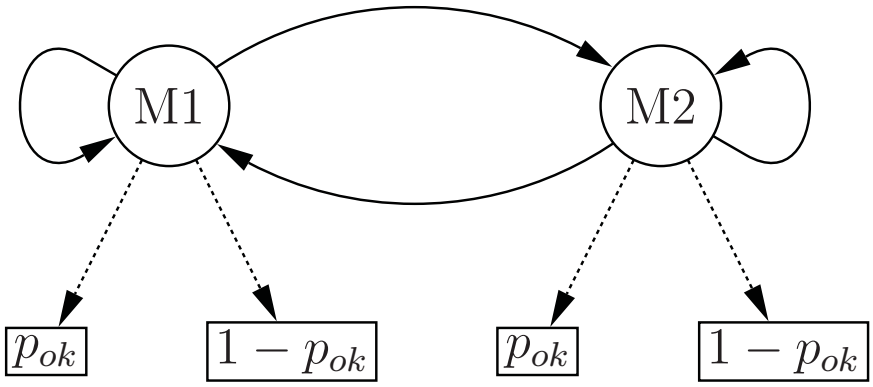

Abbildung 3.3: Maschinenszenario: Zustandsmodell der Sequenzerkennungsverfahren

Hidden Markov Model

Da im Hidden Markov Model die Symbole in den Zuständen emittiert werden, müssen hier zwei Zustände die jeweiligen Maschinen repräsentieren. Des Weiteren wurden alle Zeitstempel aus den Daten entfernt, da in jedem diskreten Zeitschritt ein Zustandswechsel möglich ist. Somit stellt die Anzahl der Symbole im Protokoll gleichzeitig die maximale Anzahl an Zeitschritten dar.

Die Initialwerte des Modells sind dabei neutral, dass heißt alle Transitionen und Observationen sind anfangs gleichverteilt. Auch andere Anfangsverteilungen wurden modelliert, um optimale Ergebnisse zu erzielen.

Trainiert wurde dieses Modell mit dem Baum-Welch-Algorithmus. Die Verwendung eines einzigen Modells hat hier jedoch den Nachteil, dass kein überwachtes Training möglich ist. Dies kann den Effekt haben, dass die Trainingsergebnisse nicht der Semantik des Problems entsprechen.

Die Ergebnisse des Hidden Markov Model zeigen dabei, dass nicht einmal die Hälfte der Observationen korrekt zugeordnet werden konnte.

Modellierung mit Phasenverteilungen

Wie im vorherigen Kapitel erwähnt, kann die Einführung von Zwischenzuständen die Modellierung verschiedener Verteilungsfunktionen ermöglichen. Da das Ergebnis dieses Vorgangs eine Markov-Kette darstellt, kann mit der Einführung eines weiteren Observationssymbols ohne Semantik daraus ein Hidden Markov Model gebaut werden. Dabei kann diese Scheinobservation nur in den Phasen, also den Zwischenzuständen dieses Hidden Markov Models, emittiert werden. Sinnvolle Ausgaben können nur vom absorbierenden Zustand der Phase erzeugt werden, also wenn die gesamte Phasenverteilung durchlaufen wurde.

Für diesen Zweck wurden Phasenverteilungen mit bis zu 16 Phasen im Zeitschritt 1 modelliert. Diese wurden vor dem Training dieses Hidden Markov Models auf Basis der Trainingsdaten optimiert, um die Verteilungen in den Daten anzunähern (vergleiche [IWH06]). Die Güte der Approximation durch diese Phasenverteilung lässt sich am Beispiel der Normalverteilung mit Mittelwert 120 und Standardabweichung 20 abschätzen,

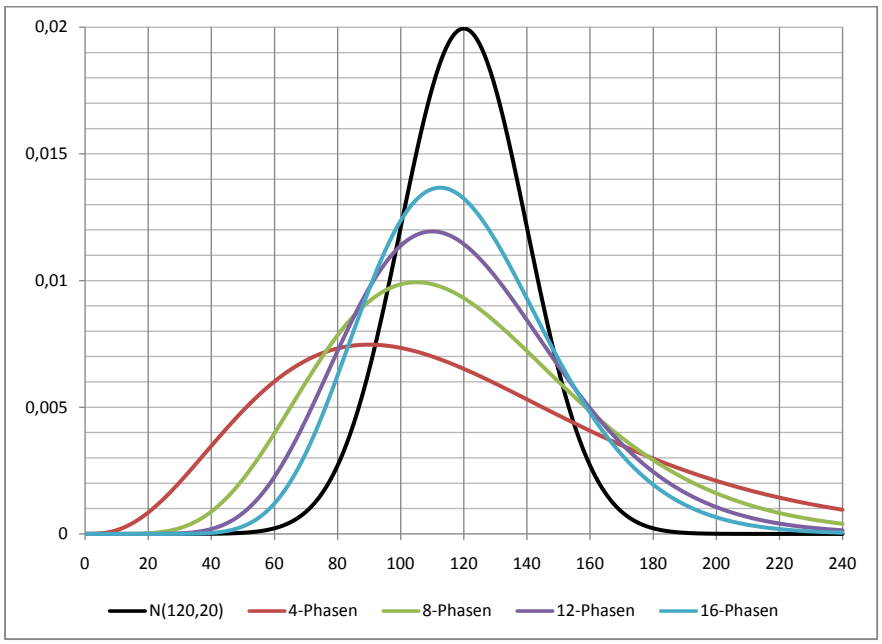

Abbildung 3.4: Approximation einer Normalverteilung mit verschiedenen Phasenverteilungen im Zeitschritt 1

was in Abbildung 3.4 für verschiedene Phasenanzahlen dargestellt ist.

Trotz der Einführung von bis zu 16 Phasen konnten die Ergebnisse des Hidden Markov Model nicht signifikant verbessert werden. Sechs Phasen pro Verteilung konnten dabei dieselben Ergebnisse erzielen wie Verteilungen mit mehr Phasen. Da sie jedoch sehr viel weniger Zeit für das Training und zur Dekodierung benötigen, wurde dieser Wert bevorzugt.

Conditional Random Field

Zur Lösung des Problems wurde eine lineare CRF-Kette modelliert. Der zugrunde liegende Zustandsraum besteht dabei aus zwei Zuständen, welche die jeweiligen Maschinen repräsentieren. Jedem Zustand wurden dabei verschiedene Feature-Funktionen übergeben:

- Start Feature $f(y_0)$ und End Feature $f(y_n)$,
- Edge Features $f(y_i, y_{i-1})$ mit $0 < i \leq n$ und
- Word Features $f(y_i, x_i)$.

Für jede mögliche Kombination von Argumenten werden diese Funktionen mit dem Wert 1 initialisiert. Durch Training werden dann die entsprechenden Parameter θ angepasst. So wird eine lineare CRF-Kette erzeugt, die einem Hidden Markov Model sehr ähnelt.

Die Observationen sind dabei ein Tupel aus Zeitstempeldifferenz und Zustand des Produkts. Diese Differenz musste aufgrund der eher komplexen Einbindung von reellen Observationen diskretisiert werden. Verschiedene Diskretisierungen wurden dabei in

Betracht gezogen, im Endeffekt stellte sich die Einteilung in die Bereiche „unter 50s“, „unter 100s“ und „über 100s“ als Kompromiss aus Genauigkeit und Geschwindigkeit heraus. Dennoch konnten weniger als 50 % der Daten korrekt klassifiziert werden.

3.2.3 Ergebnisse und Bewertung

Nachdem alle nötigen Modelle aufgestellt wurden, konnten diese trainiert und getestet werden. Um die Ergebnisse und die Performanz der einzelnen Verfahren vergleichen zu können, werden zunächst Maße für die einzelnen Kriterien Genauigkeit, Geschwindigkeit und Handhabbarkeit für dieses spezielle Problem ausgewählt.

Vergleichsmaße

Um die Genauigkeit einer Lösung für dieses Problem zu bestimmen, wird ein Maß für Klassifikationsaufgaben (aus [Bra07]) verwendet. Hierbei wird für jede Sequenz $\mathbf{x}$ die *Error Rate* bestimmt. Dies ist der Prozentsatz an Fehlklassifikationen in einer Sequenz und kann hier auch als *Hamming-Distanz* zur korrekten Sequenz (nach [Ham50]) interpretiert werden. Wird dieser Wert über alle Testsequenzen gemittelt, ist dies ein Schätzer für die Wahrscheinlichkeit einer falschen Maschinenzuordnung $P(f(\mathbf{x}) \neq c|(\mathbf{x}, c) \in E)$.

Für die Geschwindigkeit einer Lösung werden Trainings- und Testgeschwindigkeit unterschieden. Diese Unterscheidung ist notwendig, da in den meisten Anwendungen das Training eines Verfahrens einmalig, ein Test jedoch mehrmalig durchgeführt wird. Beide Geschwindigkeiten werden im inversen Maß *Millisekunden pro Datentupel* gemessen.

Das Maß für die Handhabbarkeit einer Lösung soll den Aufwand für den Anwender repräsentieren. Je mehr Möglichkeiten die Lösung dabei bietet, desto mehr Hintergrundwissen benötigt der Anwender und desto höher ist sein Aufwand, die Lösung anzuwenden. Daher wird als Maß die *Anzahl freier Parameter* in einem speziellen Modell genutzt. Da das Kriterium der Handhabbarkeit jedoch sehr subjektiv ist und sich nicht vollständig in Zahlen erfassen lässt, wird zusätzlich eine Einschätzung des Einarbeitungsaufwandes vom Autor dieser Arbeit angeboten.

Ergebnisse

Die Ergebnisse der einzelnen Lösungen in den oben beschriebenen Maßen ist in Tabelle 3.2 dargestellt.

Das entwickelte Hidden non-Markovian Model hat mit einer Fehlerrate von circa 16 % die besten Ergebnisse erzielt, ist jedoch auch mit mehr als sechs Millisekunden pro Tupel das mit Abstand langsamste Verfahren bei der Evaluierung. Die Güte der entwickelten Lösung ist stark abhängig von der Größe des Zeitschritts im Proxelbaum und der unteren Wahrscheinlichkeitsschranke. So kann es vorkommen, dass bei spezifischen Sequenzen beide Werte angepasst werden müssen, um optimale Ergebnisse zu erzielen. Eine Herabsetzung der Schranke garantiert dabei nur leicht bessere Ergebnisse mit unverhältnismäßig steigenden Ausführungszeiten.

Mehrschichtige Perzeptren, Bayes-Klassifikator und Entscheidungsbäume zur Lösung dieses Problems klassifizierten zwischen 44.2 % und 54.1 % der Tupel falsch. Trotz ihrer Schnelligkeit mit nur 0.04 bis 0.08 Millisekunden pro Tupel und ihrer geringen Anzahl

Lösung	Variante	Error Rate	Trainings-/Testdauer in $\frac{ms}{\text{Tupel}}$	Freie Parameter
HnMM		0.159	-/6.104	2
MLP	Datenmodell 1	0.442	0.05/0.08	1
	Datenmodell 2	0.425	0.05/0.08	1
	Datenmodell 3	0.456	0.05/0.08	1
Bayes	Datenmodell 1	0.449	0.05/0.04	0
	Datenmodell 2	0.452	0.06/0.05	0
	Datenmodell 3	0.473	0.05/0.05	0
DecTree		0.541	0.05/0.04	0
HMM	2 Zustände	0.551	10.28/0.02	0
	6-Phasen-Verteilungen	0.551	287.00/0.28	1
CRF		0.589	7.32/0.03	1

Tabelle 3.2: Ergebnisse Maschinenverhalten – Fehlerrate, Zeit pro Tupel, Freie Parameter

an freien Parametern (nur im Multi-Layer-Perzeptron kann die Anzahl der versteckten Neuronen die Ergebnisse beeinflussen) sind sie damit kaum geeignet, die hier gestellte Aufgabe zu erfüllen.

Diese Aussage gilt auch für die beiden klassischen Sequenzerkennungsverfahren: Hidden Markov Models und Conditional Random Fields (freier Parameter ist hier die genaue Einteilung der Differenzen in diskrete Symbole). Die Fehlerrate dieser Modelle liegt zwischen 55.1 % und 58.9 % und damit noch über den Raten der Regressions- und Klassifikationsverfahren. Bei den Hidden Markov Models ist es das unüberwachte Training, das dazu führt, dass einer Maschine immer korrekte und einer Maschine immer fehlerhafte Produkte zugeordnet werden. Dabei fällt auf, dass selbst die Einführung von Phasenverteilungen zwar eine extreme Erhöhung des Aufwands, aber keine Erhöhung der Genauigkeit nach sich zieht. Aber auch die Conditional Random Fields, die den Vorteil des überwachten Trainings und der zusätzlichen Observationen besitzen, können keine besseren Ergebnisse erzielen.

Das Diagramm in Abbildung 3.5 veranschaulicht die Performanz der einzelnen Lösungen auf einen Blick. Tabelle 3.3 zeigt dabei, wie die Normierung der Werte aus Tabelle 3.2 durchgeführt wurde. Deutlich wird, dass die Genauigkeit des Hidden non-Markovian Models unerreicht bleibt, jedoch die Geschwindigkeit kaum mit den anderen Verfahren vergleichbar ist. In den Punkten Handhabung und Geschwindigkeit wird deutlich, dass das Hidden Markov Model die besten Werte erzielt. In diesem Anwendungsszenario ist es jedoch nicht in der Lage akzeptable Ergebnisse zu produzieren.

Größe	Performanzwert 0	Performanzwert x	Performanzwert 1
Genauigkeit	Fehlerrate 1	1-Fehlerrate	Fehlerrate 0
Geschwindigkeit	Testdauer ∞	0.02/Testdauer	Testdauer 0.02 ms/Tupel
Handhabbarkeit	∞ freie Parameter	1/freie Parameter	0 freie Parameter

Tabelle 3.3: Machinenszenario: Normierung der Ergebnisse

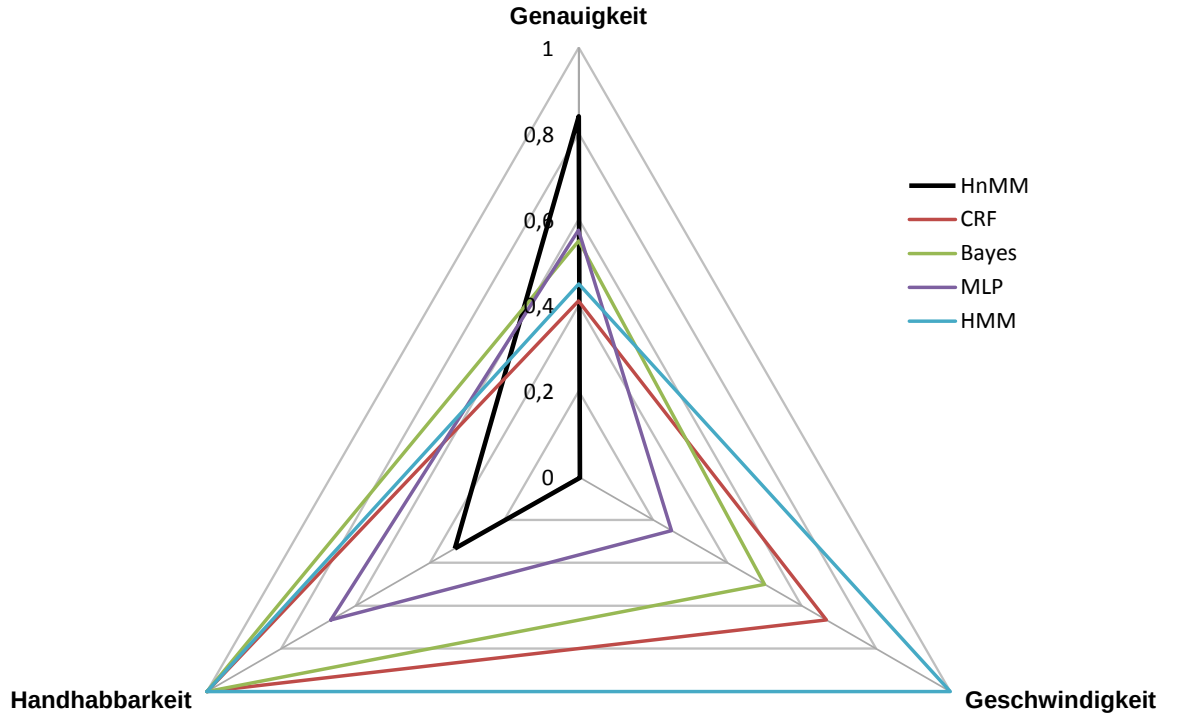

Abbildung 3.5: Performanz der Verfahren - Maschinenverhalten

Bewertung

Trotz relativ schlechter Werte bezüglich Geschwindigkeit und Handhabbarkeit hat die Lösung mit einem Hidden non-Markovian Model keine Konkurrenz. Die Genauigkeit der anderen hier präsentierten Lösungen ist dafür zu niedrig. Dies wird ganz besonders deutlich, wenn man die Resultate dieser Lösungen mit denen eines „Immer-M1-Klassifizierer“ vergleicht. Diese Lösung ordnet jedem Datenpunkt die Klasse *Maschine 1* zu, hat damit keinen Berechnungsaufwand und erzielt trotz dessen eine Fehlerrate von 44.9 %. Dieser Wert liegt unter den Raten für einige sehr viel aufwändigere Lösungen.

Die Gründe für die schlechte Genauigkeit aller Verfahren außer den Hidden non-Markovian Models sind verschieden. Da die Regressions- und Klassifikationsverfahren jedes Tupel der Sequenz als unabhängig ansehen und eine Analyse der Abhängigkeiten zwischen den einzelnen Tupeln nicht erfolgt, können die Trainingsalgorithmen zwar Entscheidungsregeln finden, welche die Trainingsdaten aufschlüsseln, jedoch bei unbekannten Daten versagen. Zwar führt die Vorverarbeitung der Zeitstempel zu Differenzen zu verbesserten Ergebnissen, aber die Tatsache, dass die Klassifikation aufgrund der Problemstellung wechselhaft sein sollte, kann durch diese Lösungen nicht abgebildet werden. Möglicherweise könnten jedoch weitere Anpassungen des Datenmodells zu besseren Ergebnissen führen.

Die Betrachtung der Eingabedaten als Sequenz mit untereinander abhängigen Elementen garantiert jedoch keine Verbesserung der Genauigkeit, wie die Ergebnisse der Sequenzerkennungsverfahren zeigen. Hier war der Effekt zu beobachten, dass eine Fehlklassifikation an einem bestimmten Punkt der Sequenz weitere Fehler nach sich zieht.

Dieser Effekt war bei Hidden non-Markovian Models nicht zu beobachten, die trotz Fehlklassifikationen nach wenigen weiteren Tupeln wieder korrekte Klassifikationen vornehmen konnten.

Diese Tatsache lässt sich mit der Markov-Eigenschaft erklären, die Hidden Markov Models und Conditional Random Fields annehmen. Wird ein Element der Sequenz fehlerhaft klassifiziert, so hängt die nächste Klassifikation nur von dieser falschen Information ab. So können diese Modelle die Information, dass ein Maschinenwechsel wahrscheinlicher werden sollte, je öfter eine Maschine klassifiziert wird, nicht abbilden. Somit wächst mit zunehmender Länge der Sequenz die Diskrepanz zwischen der zu erkennenden und der erkannten Sequenz.

Die subjektive Handhabbarkeit der Verfahren ist dabei sehr unterschiedlich. Während für die Verfahren wie Bayes-Klassifikatoren, Entscheidungsbäume und Mehrschichtiges Perzeptron nur eine Vorverarbeitung der Daten nötig war, mussten für Hidden Markov Models und Conditional Random Fields Bibliotheken genutzt und konfiguriert werden. Somit war der Arbeitsaufwand für letztere Verfahren ungleich größer als für zuerst genannte. Da für Hidden non-Markovian Models noch keine einheitliche Implementierung genutzt werden konnte und durch die Kombination der Eigenschaften verschiedene Lösungsstrategien gewählt werden mussten, hatte die Umsetzung dieses Ansatzes den mit Abstand größten Arbeitsaufwand. Hinzu kommt, dass eine geringe Herabsetzung der unteren Schranke – unter Umständen nötig, um überhaupt Ergebnisse zu erhalten – eine extreme Erhöhung des zeitlichen Aufwands nach sich zog. Im Gegensatz zu den klassischen Sequenzerkennungsverfahren war dieser erhöhte Aufwand aber mit sehr viel höherer Genauigkeit verbunden.

Neben den oben genannten Gründen für die geringe Genauigkeit einiger Lösungen, könnte auch die gleiche Fehlerwahrscheinlichkeit beider Maschinen bessere Ergebnisse verhindert haben. Dadurch wird die Information über den Zustand eines Produkts zu einer unnützen Information, denn sie kann kein Indikator dafür sein, welche Maschine zu klassifizieren ist. Eine Anpassung der Fehlerwahrscheinlichkeiten könnte also zu besseren Resultaten bei den entsprechend trainierten Lösungen führen.

3.2.4 Einführung einer fehlerhaften Maschine

Wie bereits oben in der Problemstellung erwähnt, könnte eine Anwendungsmöglichkeit einer Lösung das Erkennen einer fehlerhaften Maschine sein. Da jedoch nur das entwickelte Hidden non-Markovian Model in der Lage ist, die Originaldaten zufriedenstellend zu klassifizieren, wird auf die Analyse der anderen Lösungen in diesem Punkt verzichtet.

Da jedoch durch die Einführung einer höheren Fehlerwahrscheinlichkeit die Unterscheidung der Maschinen einfacher ist, wird versucht, die entwickelten Lösungen mit diesen veränderten Daten zu trainieren. Dies hat zwar keine Relevanz bezogen auf die Problemstellung, kann jedoch die Genauigkeit erhöhen und ermöglicht so eine Abschätzung des Effekts unterschiedlicher Fehlerwahrscheinlichkeiten. Zu diesem Zweck wurde die Fehlerwahrscheinlichkeit von Maschine 1 auf 30 % erhöht, während die Wahrscheinlichkeit von Maschine 2, Fehler zu produzieren, bei 5 % verbleibt. Mit diesen veränderten Parametern wurden wiederum Trainings- und Testdatensätze erzeugt.

Anwendung des Hidden non-Markovian Model

Dem für gleiche Fehlerwahrscheinlichkeiten parametrisiertem Modell wurden die neuen Testdaten übergeben. Neben der Fehlerrate der Klassifikationen wird in diesem Fall auch die Wahrscheinlichkeit für Fehlproduktionen, abhängig von der Maschine, $P(\text{Defect}|M)$ geschätzt.

Bei den Originaldaten lagen diese Wahrscheinlichkeiten um die tatsächlichen 5 %, die Differenz zwischen den Fehlerwahrscheinlichkeiten der einzelnen Maschinen lag zwischen 0.001 und 0.015. In diesem Bereich kann also keine Entscheidung getroffen werden, welche Maschine fehlerhaft ist.

Die Anwendung des Modells auf die veränderten Daten führt zu einer Fehlerrate von 15.28 %. Dieser Wert unterscheidet sich nicht signifikant von der Fehlerrate von 15.9 % bei den Originaldaten. Daraus lässt sich folgern, dass die Güte der Klassifikation in diesem Fall fast nur von der Ausprägung der Zeitstempel und weniger von der Zustandswahrscheinlichkeit abhängt. Die geschätzten Fehlerwahrscheinlichkeiten liegen bei 29.8 % für Maschine 1 und 11.0 % für Maschine 2.

Es wird deutlich, dass die Fehlerwahrscheinlichkeit der Maschine 1 sehr gut geschätzt wird, während die der Maschine 2 überschätzt wird. Dies lässt sich durch Fehlklassifikationen erklären, deren Häufigkeit gegenüber den Originaldaten konstant bleibt. Unter der Prämisse, dass diese Fehler unabhängig vom Zustand des Produkts auftreten, werden fehlerhafte Produkte von Maschine 1 teilweise Maschine 2 zugeordnet. Der Unterschied der Fehlerwahrscheinlichkeit beider Maschinen liegt hier zwischen 0.17 und 0.21.

Somit kann eine Entscheidung, welche Maschine wahrscheinlich Mängel aufweist, mit hoher Sicherheit getroffen werden. Also ist das entwickelte Hidden non-Markovian Model für die allgemeine Problemstellung, wie sie oben formuliert wurde, geeignet.

Verändertes Training

In diesem Abschnitt soll geklärt werden, inwieweit die Möglichkeit, Maschinen aufgrund ihrer Fehlerwahrscheinlichkeit zu unterscheiden, die Ergebnisse der entwickelten Lösungen beeinflusst. Diese Ergebnisse sind in Tabelle 3.4 den Ergebnissen aus Tabelle 3.2 gegenübergestellt. In der letzten Spalte ist die prozentuale Veränderung, farbig unterlegt, dargestellt. Dabei wird die Farbe Grün mit Verbesserung, die Farbe Gelb mit keiner Veränderung und die Farbe Rot mit Verschlechterung assoziiert.

Während Regressions- und Klassifikationsverfahren unter diesen veränderten Bedingungen signifikant bessere Ergebnisse erzielen, ist dies bei den Sequenzerkennungsverfahren nicht zu beobachten. Das Hidden Markov Model schneidet hier sogar schlechter ab.

Lösung	Fehlerrate original	Fehlerrate neu	Veränderung in %
MLP	0.425	0.385	-9.5
Bayes	0.449	0.439	-2.3
DecTree	0.541	0.385	-28.9
HMM	0.551	0.568	+3.0
CRF	0.589	0.584	-0.9

Tabelle 3.4: Vergleich der verschieden trainierten Lösungen für das Maschinenszenario

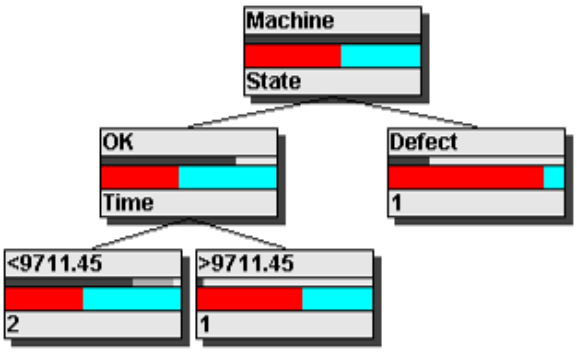

Abbildung 3.6: Machinenszenario – Experiment: Grafische Darstellung des Entscheidungsbaumes (Screenshot aus *DecTree*)

Dies liegt darin begründet, dass die Einbindung mehrdimensionaler, unter Umständen sogar abhängiger Observationen, in diesen Modellen schwierig ist. Weiterhin kann das Hidden Markov Model, wie bereits oben erwähnt, in diesem Fall nicht überwacht trainiert werden. Somit kann die Information, dass Maschine 1 mehr defekte Produkte herstellt, nicht in das Hidden Markov Model übergeben werden.

Die benutzen Entscheidungsbäume profitieren enorm von der neuen Bedeutung der Dimension *Zustand der Maschine*. Sie sind entwickelt worden, um maximal entscheidungsunterstützende Kriterien in den Daten zu finden. Die Information, dass defekte Produkte zu hoher Wahrscheinlichkeit von Maschine 1 stammen, kann also in hohem Maße in das Modell eingebracht werden. Dies wird auch durch die Struktur des trainierten und gestutzten Baumes deutlich: Er besteht aus gerade einmal fünf Knoten auf drei Ebenen (Abbildung 3.6). In diesem Baum wird also jedes defekte Produkt zu Maschine 1 zugeordnet, erst danach erfolgt die (fehlerbehaftete) Zuordnung nach der Zeitdifferenz.

Der entwickelte Bayessche Klassifikator schneidet von diesen Verfahren nun am schlechtesten ab. Da auch hier der Volle Klassifikator aufgrund der symbolischen Natur der Zustandsdimension ausfällt, kann der Klassifikator die Korrelationen zwischen den Dimensionen nicht ausnutzen.

Die Veränderung der Ergebnisse der einzelnen Lösungen ist nachvollziehbar und zeigt die Unterschiede in der Interpretation der Eingabedaten auf. Die klassischen, Markovschen Sequenzerkennungsverfahren interpretieren diese richtigerweise als Sequenz, jedoch ist ihre Annahme der Markov-Eigenschaft dafür verantwortlich, dass die Ergebnisse nicht zufriedenstellend sind. Die zusätzliche Information, die in diesem Szenario bereitgestellt wurde, kann von diesen Lösungen nicht in dem Maße verarbeitet werden, wie es in den Regressions- und Klassifikationslösungen der Fall ist. Diese Verfahren interpretieren die sequentiellen Eingabedaten als unabhängige Tupel. Ihre Stärke liegt gerade darin begründet, dass sie hochdimensionale Daten verarbeiten und deren Abhängigkeiten ausnutzen können. Die zusätzliche Information in diesem Szenario wird von ihnen maximal genutzt. Trotz teilweise signifikanter Verbesserungen in den Ergebnissen bleiben jedoch die Resultate des Hidden non-Markovian Models in puncto Genauigkeit unerreicht. Kei-

nes der Verfahren scheint, zumindest mit den hier genutzten Datenmodellen, in der Lage zu sein, die definierte Problemstellung zufriedenstellend zu bearbeiten.

3.3 Erkennung von Krankheiten

Als zweites Anwendungsbeispiel wird ein Szenario gewählt, in dem Krankheiten erkannt werden sollen. Hier ist es das Ziel, die Krankheit eines Patienten aufgrund der Entwicklung seiner Symptome zu diagnostizieren. Die maschinelle Unterstützung bei der Diagnostik ist ein recht altes Forschungsgebiet. Zu erwähnen ist hierbei besonders das System *MYCIN*, das 1970 als regelbasiertes Expertensystem zur Diagnostik von bakteriellen Infektionen entwickelt wurde (siehe [SDA+75]). Im Gegensatz zu diesem System, das eine einmalige Analyse des derzeitigen Zustands eines Patienten durchführt, werden hier Lösungen diskutiert, die aus den gesammelten Testergebnissen des Krankheitsverlaufs die wahrscheinlichste Krankheit bestimmt. Dieser Ansatz wurde in [KBH10] beschrieben und mithilfe von Hidden non-Markovian Models gelöst.

3.3.1 Problembeschreibung – Diagnose einer Krankheit nach Symptomverlauf

In diesem Szenario sollen Patienten drei verschiedenen, abstrakten Krankheiten zugeordnet werden. Dazu werden ihnen vier Mal täglich Fieber-, Urin- und Blutwerte notiert. Der Patient hat dabei Fieber (F_j) oder nicht (F_n), Urin- und Bluttest können drei verschiedene Werte liefern (U_a, U_b, U_c bzw. B_a, B_b, B_c).

Jede dieser drei Krankheiten ist dabei durch einen 3-Phasen-Verlauf gekennzeichnet. In der ersten Phase, der Inkubation, sind die Symptome in der Regel weit weniger ausgeprägt als in der darauffolgenden Phase, der Verschlechterung. In der letzten Phase einer Krankheit sind dann die stärksten Symptome zu beobachten. Unabhängig von der Phase, in der ein Patient sich befindet, kann er jederzeit geheilt werden und die Symptome verschwinden. Dieser Zustandsraum ist in Abbildung 3.7 dargestellt.

Eine Unterscheidung der verschiedenen Krankheiten kann dabei durch das Zusammenspiel von Symptomen und unterschiedlicher Phasendauer in den einzelnen Krankheiten ermöglicht werden. Die exakten Parameter der einzelnen Krankheiten sind in Tabelle 3.5 dargestellt.

Es wird deutlich, dass die einzelnen Krankheiten sich nicht signifikant über die Symptome unterscheiden. Krankheit 2 und Krankheit 3 haben dabei sogar die exakt gleichen Verteilungen für die Symptome. Wesentlich unterscheiden sich die Krankheiten nur in der Dauer ihrer Phasen, die durch eine Gleich-, Normal- oder Exponentialverteilung angegeben ist. Die Dauer der finalen Phase bezieht sich dabei aber nicht auf die Verweildauer in derselben, sondern auf die Gesamtzeit bis zur Heilung, die in jeder Krankheitsphase erfolgen kann. Zeiten werden dabei in Tagen angegeben, also ist die Zeitdifferenz zwischen zwei Observationen 0.25 Tage.

Basierend auf diesen Daten konnte ein Zufallsgenerator implementiert werden, der Krankheitsprotokolle erzeugt. Für jede Krankheit wurden dabei fünf Trainings- und fünf Testsequenzen erzeugt. In den Trainingssequenzen sind als zusätzliche Informationen die Zeitpunkte der Phasenübergänge gekennzeichnet.

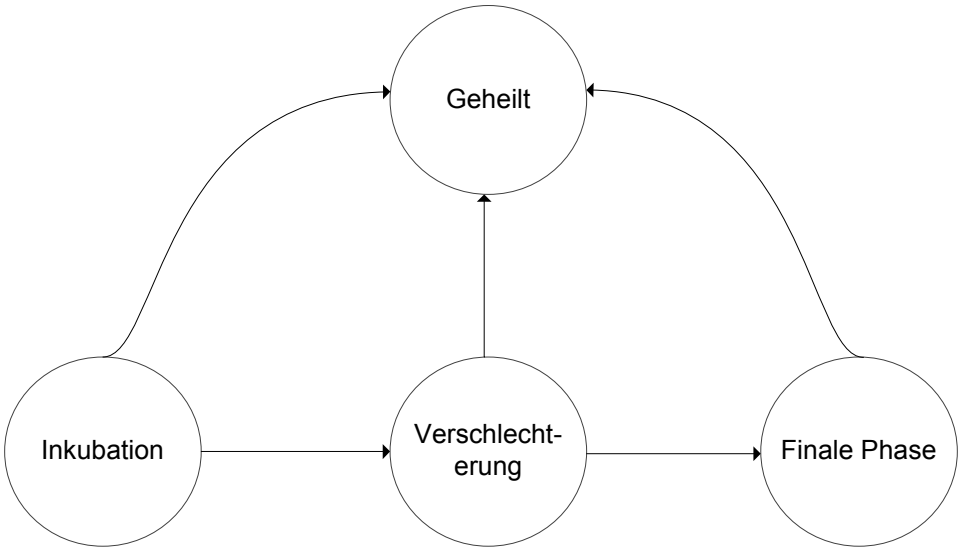

Abbildung 3.7: Schematische Darstellung des Zustandsraums für das Krankheitsszenario

Krankheit	Phase	Dauer	Fieberwert-verteilung	Urinwert-verteilung	Blutwert-verteilung
1	Inkubation	$U(2,4)$	(0.1,0.9)	(0.8,0.1,0.1)	(0.7,0.2,0.1)
	Verschlechterung	Exp(0.1)	(0.5,0.5)	(0.6,0.3,0.1)	(0.5,0.3,0.2)
	Finale Phase	$U(7,14)$	(0.8,0.2)	(0.5,0.3,0.2)	(0.2,0.3,0.5)
2	Inkubation	$N(3,0.5)$	(0.3,0.7)	(0.8,0.2,0.0)	(0.7,0.2,0.1)
	Verschlechterung	Exp(0.3)	(0.7,0.3)	(0.7,0.2,0.1)	(0.5,0.3,0.2)
	Finale Phase	$N(10,2)$	(0.9,0.1)	(0.7,0.2,0.1)	(0.2,0.3,0.5)
3	Inkubation	$N(6,1)$	(0.3,0.7)	(0.8,0.2,0.0)	(0.7,0.2,0.1)
	Verschlechterung	Exp(0.3)	(0.7,0.3)	(0.7,0.2,0.1)	(0.5,0.3,0.2)
	Finale Phase	$N(12,2)$	(0.9,0.1)	(0.7,0.2,0.1)	(0.2,0.3,0.5)

Tabelle 3.5: Parameter der einzelnen Krankheitsmodelle – Phasendauer- und Symptomverteilung

3.3.2 Entwickelte Lösungen

In den folgenden Abschnitten werden die Lösungen, die mithilfe der einzelnen Verfahren entwickelt wurden, vorgestellt. Danach erfolgt eine Zusammenfassung und der Vergleich der Ergebnisse.

Hidden non-Markovian Model

Im Unterschied zu der originalen Definition für Hidden non-Markovian Models (vergleiche Kapitel 2), wurde für die Lösung dieses Problems die Ausgabe von Observationen verändert. Vergleichbar mit einem Hidden Markov Model werden diese nun im Takt von 0.25 Tagen abhängig vom aktuellen Zustand des Modells emittiert. Tabelle 3.6 zeigt eine beispielhafte Sequenz, wie sie vom Hidden non-Markovian Model verarbeitet wird.

Zeit in Tagen	Symptome
0.25	(F_n, U_a, B_a)
0.5	(F_j, U_a, B_a)
0.75	(F_n, U_b, B_b)
...	...
11.5	(F_j, U_c, B_b)
11.72	Heilung

Tabelle 3.6: Krankheitsszenario: Beispiel für eine Symptomsequenz

Ausgehend vom oben erläuterten Zustandsmodell besteht das HnMM aus vier Zuständen, welche die einzelnen Krankheitsphasen und den Zustand *Geheilt* repräsentieren. Zwischen Phase 1 und 2 sowie zwischen Phase 2 und 3 ist ein Zustandsübergang möglich. Ferner wurde eine Transition zum Zustand *Geheilt* erstellt. Der Proxel-Lösungsalgorithmus wurde dabei dahingehend angepasst, dass diese Transition im Initialzustand aktiviert wurde und zu jedem (durch die angegebene Verteilung möglichen) Zeitpunkt feuern kann. Dieses Modell ließe sich also als (*Tkeep*, *SConeT*) kategorisieren. Eine Zuordnung von *Eall* beziehungsweise *Esome* kann hier nicht erfolgen, da die Observationen hier nicht von den Transitionen emittiert werden.

Die Wahrscheinlichkeit einer Symbolausgabe wurde dabei durch das Produkt der Wahrscheinlichkeiten der Einzelsymbole berechnet. Somit wurden alle drei Testkategorien in einer Dimension zusammengefasst, damit das Hidden non-Markovian Model diese Vektoren als einzelne Symbole interpretieren konnte.

Das Schätzen der Modellparameter konnte aufgrund der geringen Datenmenge hier nicht durchgeführt werden, stattdessen wurden für jede Krankheit die oben angegebenen Parameter benutzt. Nachdem so für jede Krankheit ein Modell entwickelt wurde, konnten die Testsequenzen den drei Modellen übergeben werden. Die Krankheit, deren entsprechendes Modell die höchste Erzeugungswahrscheinlichkeit für die Sequenz vorweist, wird klassifiziert.

Mehrschichtiges Perzeptron

Für eine Lösung mit einem Mehrschichtigen Perzeptron wurden drei verschiedene Ansätze verfolgt. Der erste ist ein einzelnes Mehrschichtiges Perzeptron, dessen Datenmodell aus 59 Eingabedimensionen und einer Ausgabedimension besteht. Jede Eingabedimension repräsentiert dabei einen Testzeitpunkt und kann jeden möglichen Symptomvektor als Wert annehmen. Die Anzahl wurde so gewählt, da in keinem Datensatz eine Krankheit mit einer Dauer länger als 15 Tage auftrat und dies bei Krankheit 1 unmöglich, bei Krankheit 2 und 3 zumindest unwahrscheinlich ist. Ausgabe ist ein symbolischer Wert, welcher der zu diagnostizierenden Krankheit entspricht. Dieses Datenmodell ist in Tabelle 3.7 dargestellt. Das Symbol N repräsentiert dabei einen geheilten Patienten, der keine Krankheitssymptome mehr vorweist.

Dimensionen	$t = 0.25$	$t = 0.5$	...	$t = 14.75$	Krankheit
Beispiel	(F_n, U_a, B_b)	(F_j, U_a, B_a)	...	N	$K2$

Tabelle 3.7: Krankheitsszenario: Hochdimensionales Datenmodell

Im zweiten Ansatz wurde statt einem kombinierten Netz für jede Krankheit ein einzelnes Mehrschichtiges Perzeptron trainiert. Dabei entsprechen die Eingabedimensionen denen des ersten Ansatzes. Die Ausgabe ist hier jedoch ein Konfidenzwert zwischen 0 und 1. Das Perzeptron mit der höchsten Ausgabe bei einer bestimmten Eingabe repräsentiert die zu diagnostizierende Krankheit. Jedes Perzeptron wird dabei mit fünf Positiv- (Zielwert 1) und zehn Negativbeispielen (Zielwert 0) trainiert.

Der dritte Ansatz besteht ebenfalls aus einzelnen Perzeptren für jede Krankheit, sein Datenmodell besteht jedoch nur aus vier Eingabedimensionen: Dem reellen Zeitstempel und einzelnen Dimensionen für Fieber-, Urin- und Blutwerte. Ausgabe ist hier ein diskreter Wert, der der derzeitigen Krankheitsphase entspricht. Während die vorherigen Ansätze für jedes Protokoll ein Datentupel verlangen, muss hier für jeden Testzeitpunkt ein Tupel gebildet werden. Nachdem ein Datensatz bearbeitet wurde, wird der Mittelwert der Konfidenz für jede einzelne Klassifikation als Maß für die *Likelihood* dieses Modells gebildet.

Klassifikatoren

Mit demselben Datenmodell, das im ersten Ansatz für ein Mehrschichtiges Perzeptron benutzt wurde, wurde ein Naiver Bayes-Klassifikator und ein Entscheidungsbaum trainiert. Aufgrund der symbolischen Natur der Observationen ist auch hier die Anwendung eines Vollen Bayes-Klassifikator nicht sinnvoll.

Sequenzerkennungsverfahren

Um Hidden Markov Models und Conditional Random Fields für dieses Problem anzuwenden, wurden die Zeitstempel aus den Daten entfernt. In diesen Modellen wird in jedem Zeitschritt ein Symptomvektor ausgegeben. Beide Lösungen interpretieren diesen dabei als einzelnes Symbol.

Nach dieser Vorverarbeitung können die einzelnen Modelle trainiert werden. Dazu wird für jede Krankheit ein Modell mit gleichverteilten Initialparametern entwickelt, unmögliche Transitionen werden dabei mit der Wahrscheinlichkeit 0 initialisiert. Eine Trainingssequenz wird dann dem, der Krankheit entsprechenden, Modell übergeben. Das Conditional Random Field hat dabei den Vorteil, dass es auch innerhalb des Modells überwacht trainiert werden kann, das heißt jedem Datentupel ist der korrekte Zustand zugeordnet. Das entsprechende Hidden Markov Model wird mit dem Baum-Welch-Algorithmus unüberwacht trainiert.

Die Modelle bestehen aus jeweils vier Zuständen, welche die vier Krankheitszustände repräsentieren. Der Aufbau der linearen CRF-Kette und die Feature-Funktionen sind dabei analog zu der Lösung im ersten Anwendungsszenario.

Zum Testen wird jedem Modell die zu testende Sequenz übergeben, daraufhin werden Erzeugungswahrscheinlichkeiten berechnet. Das Modell mit der höchsten *Likelihood* repräsentiert die klassifizierte Krankheit.

3.3.3 Ergebnisse und Bewertung

Analog zur Vorgehensweise im vorherigen Anwendungsszenario werden in diesem Abschnitt zunächst Vergleichsmaße vorgestellt, bevor die Ergebnisse der einzelnen Lösungen präsentiert und bewertet werden.

Vergleichsmaße

Die Genauigkeit einer Lösung lässt sich in diesem Anwendungsszenario an der korrekten Klassifizierung der Krankheit messen. Dabei wird die – gegebenenfalls erfolgte – modellinterne Zuordnung der Krankheitsphasen ignoriert, da sie nicht das Ziel der Anwendung ist.

Um eine Lösung in einem Wert zu beurteilen, wird das Maß der *Accuracy* gewählt. Dies ist der Anteil an korrekten Klassifikationen an allen gegebenen Testsätzen und damit ein Maß für die Wahrscheinlichkeit einer korrekten Klassenzuordnung $P(f(\mathbf{x}) = c|(\mathbf{x}, c) \in R)$. Zusätzlich dazu werden für jede Lösung drei Werte angegeben, die ein Maß für die korrekte Zuordnung der einzelnen Krankheiten bilden. Dieses Maß wird als *F1-Measure* bezeichnet und wird aus den Maßen *Recall* und *Precision* gebildet.

Recall R für eine Klasse ist dabei der Anteil korrekt klassifizierter Beispiele an allen Beispielen für diese Klasse. Precision P für eine Klasse ist der Anteil korrekt klassifizierter Beispiele an allen zu dieser Klasse klassifizierten Beispielen. Das F1-Maß wird dann wie folgt berechnet: $\frac{2 \cdot P \cdot R}{P+R}$ (vergleiche Maße für Klassifikationen in [Bra07])

Maße für Geschwindigkeit und Handhabung sind analog zum ersten Anwendungsszenario, also die Zeit pro Trainings- beziehungsweise Testbeispiel in Sekunden als inverses Maß für Geschwindigkeit und Anzahl freier Parameter in der Lösung für die Handhabbarkeit. Die im vorherigen Szenario gegebene Aufwandseinschätzung der Verfahren gilt hier auch, soweit keine Ergänzungen oder Gegeneinschätzungen gegeben werden.

Ergebnisse

Die Ergebnisse der entwickelten Lösungen sind in Tabelle 3.8 dargestellt. Das Hidden non-Markovian Model liefert dabei die höchste Genauigkeit, da fast drei von vier Klassifikationen korrekt sind. Auch die konstanten Werte für alle drei F1-Maße zeigen, dass dieses Modell eine ausgewogen gute Diagnose erstellen kann. Da der Parameter der unteren Wahrscheinlichkeitsschranke hier – aufgrund der im Vergleich zum Maschinen-Szenario (Abschnitt 3.2) geringen Sequenzlänge – geringe Auswirkungen auf das Verhältnis von Laufzeit und Genauigkeit hat, ist dieser Wert in dieser Lösung auf $1 \cdot 10^{-10}$ fixiert.

Die Lösungen des einzelnen Mehrschichtigen Perzeptrons und des Entscheidungsbaumes diagnostizieren in fast der Hälfte der Testfälle die korrekte Krankheit, dabei sind jedoch größere Schwankungen zwischen den einzelnen Krankheiten bemerkbar. So klassifiziert das Perzeptron oft in Zusammenhang mit Krankheit 2 falsch, während der Entscheidungsbaum nicht ein einziges Mal Krankheit 3 klassifiziert. Die Genauigkeit der separaten Modelle liegt dabei deutlich unter der des kombinierten Modells. Im Gegensatz zum Entscheidungsbaum kann der Bayes-Klassifikator gerade einmal ein gutes Viertel der Krankheiten korrekt klassifizieren.

Während der Bayes-Klassifikator jedoch jede Krankheit zumindest einmal klassifiziert, schneiden die klassischen Sequenzerkennungsverfahren weitaus schlechter ab. Auch

Lösung	Variante	Accuracy	F1-Maß			Trainings-/	Freie
			K1	K2	K3	Testdauer in s	Parameter
HnMM		0.73	0.75	0.71	0.75	-/14.44	1
MLP	1 Modell	0.47	0.5	0.36	0.57	0.002/5E-4	1
	3 Modelle	0.40	0.18	0.55	0.50	0.017/1.6E-3	3
	Ziel:Phase	0.33	0	0	0.50	0.003/1.3E-3	3
Bayes		0.27	0.31	0.25	0.22	5E-4/7E-4	0
DecTree		0.47	0.55	0.57	0	2E-3/2.4E-3	0
HMM		0.27	0.31	0	0.57	9.2E-3/2E-4	0
CRF		0.27	0.83	0	0	0.09/0.06	0

Tabelle 3.8: Ergebnisse Krankheitsszenario – Accuracy, F1-Maße, Zeit pro Tupel, Anzahl freier Parameter

wenn ihre Genauigkeit auf demselben Niveau liegt, zeigen die F1-Maße, dass das Hidden Markov Model nie Krankheit 2, das Conditional Random Field sogar nur Krankheit 1 klassifiziert.

Abbildung 3.8 veranschaulicht die Performanz der verschiedenen Lösungen. Die Normierungsberechnungen sind dazu in Tabelle 3.9 dargestellt. Wie schon im ersten Szenario ist das Hidden non-Markovian Model mit Abstand das zeitaufwändigste Verfahren, bieten aber auch die genauesten Ergebnisse. Hidden Markov Models zeichnen sich in den Kriterien Handhabbarkeit und Geschwindigkeit aus, versagen jedoch bei der Klassifikation.

Bewertung

Auch in diesem Anwendungsszenario scheinen Hidden non-Markovian Models die einzige akzeptable Lösung des Problems darzustellen. Dies wird auch dadurch deutlich, dass die Klassifikationsergebnisse für die einzelnen Krankheiten konstant sind und damit die Lösung als robust angesehen werden kann. Der Zeitaufwand des Verfahrens ist zwar im Vergleich zu den anderen Lösungen unverhältnismäßig hoch, jedoch im Vergleich zu den Laufzeiten im Maschinenszenario gering. Aufgrund dieser Tatsache ist eine Abwägung von Genauigkeit und Laufzeit in diesem Beispiel nicht so wichtig wie im vorherigen Experiment.

Die klassischen Sequenzerkennungsverfahren, Hidden Markov Models und Conditional Random Fields, sind auch in diesem Beispiel nicht in der Lage, zufriedenstellende Ergebnisse zu produzieren. Bei beiden Lösungen zeigte ein Test mit den Trainingsdaten, dass diese komplett korrekt klassifiziert wurden. Somit konnte das Training zwar die gegebenen Daten approximieren, jedoch nicht die den Daten zugrunde liegenden Zusam-

Größe	Performanzwert 0	Performanzwert x	Performanzwert 1
Genauigkeit	Accuracy 0	Accuracy	Accuracy 1
Geschwindigkeit	Testdauer ∞	2E-4/Testdauer	Testdauer 2E-4 s/Tupel
Handhabbarkeit	∞ freie Parameter	1/freie Parameter	0 freie Parameter

Tabelle 3.9: Krankheitsszenario: Normierung der Ergebnisse

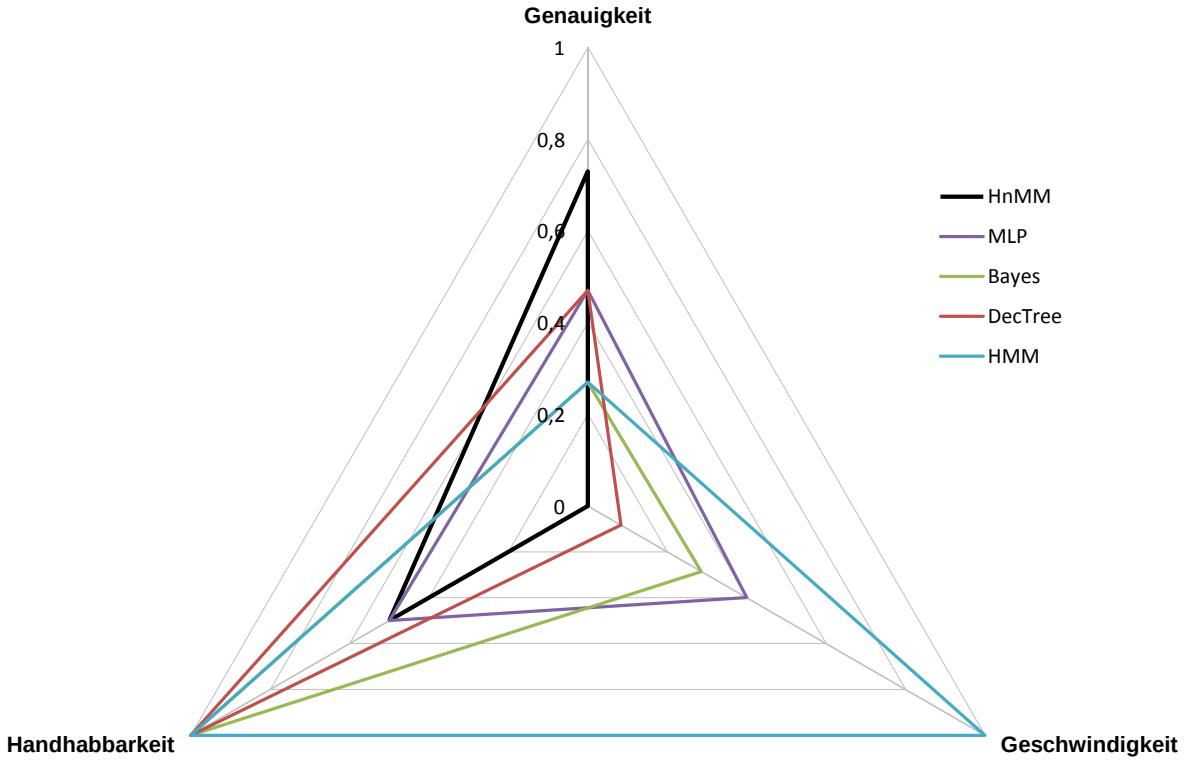

Abbildung 3.8: Performanz der Verfahren - Diagnoseanwendung

menhänge in das Modell integrieren.

Der Grund für das schlechte Abschneiden dieser Verfahren liegt wiederum in der Markov-Eigenschaft ihrer verborgenen Prozesse begründet. Dies wird besonders an folgendem Beispiel deutlich: Leidet ein Patient an Krankheit 1, so ist die Dauer des Verbleibs in der ersten Phase gleichverteilt zwischen zwei und vier Tagen. Dies bedeutet für das Modell, dass ein Übergang in den ersten sieben Zeitschritten der Modelle unmöglich sein sollte. Ohne Zwischenzustände kann jedoch ein solches Verhalten durch keine mögliche Übergangswahrscheinlichkeit abgebildet werden.

Die Modellierung von Phasenverteilungen scheitert an mehreren Punkten. Zuallererst lässt sich die Eigenschaft der jederzeit möglichen Heilung schwierig modellieren. Weiterhin sind für die angegebenen Verteilungen im Zeitschritt 0.25 *Tage* sehr viele Zwischenzustände nötig, um diese akzeptabel approximieren zu können. Besonders schwierig gestaltet sich dabei die Modellierung von Gleichverteilungen: Die Eliminierung von Übergangswahrscheinlichkeiten nach kurzer Zeit lässt sich nur durch die Einführung weiterer Zwischenzustände garantieren. Durch die so entstehende Explosion des Zustandsraums werden Training und Testen des Modells erschwert bis unmöglich gemacht.
Klassifikations- und Regressionsverfahren zeigen in der Regel eine durchschnittliche Performanz. Nur der Bayessche Klassifikator kann die entsprechenden Genauigkeitswerte nicht erreichen. Dies lässt sich über die hohe Dimensionalität der Eingabedaten erklären. Während der Entscheidungsbaum zur Klassifikation wenig hilfreiche Dimensionen ausklammert (Baum enthält nur 12 Knoten bei 59 Eingabedimensionen), muss der Naive Bayes-Klassifikator alle Dimensionen berücksichtigen, da Korrelationen in den Daten

nicht erfasst werden können.

Bei den Mehrschichtigen Perzeptren fällt auf, dass das einzelne, kombinierte Modell besser abschneidet als die Lösungen mit separaten Modellen. Letzerer Ansatz scheitert daran, dass zwar Negativbeispiele übergeben werden, diese jedoch offenbar nicht ausreichen, um die verschiedenen Krankheiten einzeln abzugrenzen. Dieser Effekt verstärkt sich in der dritten Lösungsvariante, in der nur eine indirekte Klassifikation im Perzeptron erfolgt. So wird nicht nur die *Accuracy* immer niedriger, auch die Schwankungen der einzelnen F1-Maße wird immer größer. Die letzte Lösungsvariante ist dabei de facto sogar ein „Immer-K3-Klassifizierer“, also eine Lösung, die jeden Testsatz Krankheit 3 zuordnet.

Diese Ergebnisse ließen sich auch durch die Verdoppelung der Größe des Trainingssatzes nicht signifikant steigern. Eine weitere Vergrößerung wurde im Rahmen der Arbeit nicht durchgeführt, obwohl davon ausgegangen werden kann, dass durch eine hinreichend große Anzahl von Trainingsbeispielen die Genauigkeit, zumindest der Regressions- und Klassifikationsverfahren, steigen sollte.

Insgesamt lässt sich sagen, dass die hohe Überschneidung der einzelnen Krankheitsmodelle eine Klassifizierung erschwert. Selbst die genauste Lösung mit Hidden non-Markovian Models klassifiziert von vier Beispielen eines falsch. Es ist jedoch zu erkennen, dass die Klassifikations- und Regressionslösungen in der Lage sind, diese Überscheidungen zumindest teilweise aufzulösen. So erkennen diese bestenfalls zwei von vier Beispielen. Absolut ungeeignet sind die Sequenzerkennungsverfahren, die offensichtlich nicht in der Lage sind, dieses Problem erfolgreich abzubilden.

3.4 Zusammenfassung – Hidden non-Markovian Models langsam, aber genau

In diesem Abschnitt sollen die Ergebnisse der durchgeführten Experimente noch einmal zusammengefasst werden, bevor im nächsten Kapitel Schlussfolgerungen daraus abgeleitet werden.

Beide Experimente hatten eine unterschiedliche Zielsetzung. Während der Fokus bei der Rekonstruktion des Maschinenverhaltens auf der korrekten Zuordnung der einzelnen Protokolleinträge zu den Maschinen lag, lag dieser bei der Krankheitsdiagnose auf der Zuordnung eines gesamten Protokolls zu der korrekten Krankheit.

Der Hauptunterschied beider Szenarien liegt dabei darin, dass im Maschinenbeispiel das zugrunde liegende Modell derart von den benutzten Verfahren abgebildet werden musste, dass die einzelnen Datentupel korrekt klassifiziert wurden. In der Diagnoseanwendung war die Abbildung des zugrunde liegenden Modells nicht notwendigerweise erforderlich, da für die Güte der Lösung nur die Krankheitszuordnung herangezogen wurde. Somit ist letzteres Anwendungsszenario eine Zuordnung weniger, aber hochdimensionaler Datentupel, während im ersteren viele, aber niedrigdimensionale Datentupel verarbeitetet werden mussten. Diese fehlende Ähnlichkeit kann es erlauben, Schlüsse aus den Experimenten zu ziehen.

Denn trotz dieser Unterschiede zeigen die Ergebnisse beider Szenarien gewisse Muster. Eine Übersicht über die Eignung der einzelnen Lösungen in den drei Kriterien ist

Verfahren	Genauigkeit	Geschwindigkeit	Handhabbarkeit
HnMM			
MLP/DecTree/Bayes			
HMM/CRF			

Tabelle 3.10: Grobe Einschätzung der Eignung der vorgestellten Lösungen

in Tabelle 3.10 in einem Ampelsystem (Grün – vergleichsweise akzeptabel, Gelb – vergleichsweise durchschnittlich, Rot – vergleichsweise inakzeptabel) farbig dargestellt.

So ist bezüglich der Genauigkeit bei beiden Beispielen eine Reihenfolge erkennbar. Während Hidden non-Markovian Models die mit Abstand genausten Ergebnisse erzielen, liegen die Ergebnisse der Regressions- und Klassifikationsverfahren noch weit über den Genauigkeitswerten, die mit Sequenzerkennungsverfahren erreicht werden konnten.

Wird hingegen die Geschwindigkeit der Verfahren betrachtet, ist diese Reihenfolge umgekehrt. Hier sind Hidden Markov Models und Conditional Random Fields deutlich schneller als Mehrschichtige Perzeptren, Bayes-Klassifikatoren und Entscheidungsbäume. In keinem Verhältnis dazu stehen die Laufzeiten der nicht-Markovschen Lösungen, die bei beiden Beispielen um den Faktor 100 bis 10 000 darüber liegen.

Auch bezüglich der Handhabbarkeit zeichnet sich – zumindest aus einer subjektiven Perspektive – eine 3-Klassen-Einteilung ab. Während die Lösungen für Hidden non-Markovian Models sich teilweise in grundlegenden Implementierungsaspekten unterscheiden, nutzen die Lösungen für HMMs und CRFs dieselben Bibliotheken, mussten aber unterschiedlich konfiguriert werden. Die leichteste Handhabung boten jedoch die genutzten Klassifikations- und Regressionsverfahren, die auf gleichem Datenbestand operieren konnten und nur wenige Konfigurationsmaßnahmen nötig machten.

3.5 Schlussfolgerungen

Nachdem die Experimente durchgeführt und deren Ergebnisse in den vorherigen Abschnitten beschrieben und zusammengefasst wurden, werden in diesem Abschnitt Schlüsse aus diesen Ergebnissen gezogen Dabei wird eine Einschätzung über die Eignung der Verfahren, die in dieser Arbeit bearbeiteten Problemstellungen zu lösen, gegeben. Weiterhin werden Gründe für die Performanz der Verfahren diskutiert und der Versuch unternommen, die Schlüsse aus den Experimenten zu verallgemeinern.

Die hier verwendeten klassischen Regressions- und Klassifikationsverfahren, also Mehrschichtige Perzeptren, Bayes-Klassifikatoren und Entscheidungsbäume, sind bewährte und häufig eingesetzte Verfahren. Warum dies so ist, offenbaren auch die Ergebnisse der Experimente. So bieten diese Verfahren schlanke Lösungen ohne großen Aufwand und erzielen dabei zumindest bessere Ergebnisse als die Sequenzerkennungsverfahren. Trotz des allgemeinen Ansatzes dieser Verfahren und der wenigen Information über die zugrunde liegenden Prozesse, konnten diese Lösungen zumindest teilweise Zusammenhänge in den Daten erkennen und diese bei der Klassifikation anwenden.

Dabei fällt auf, dass Entscheidungsbäume mehr als die beiden anderen Verfahren von vielen Eingabedimensionen und Korrelationen darin profitieren. Zumindest bei sym-

bolischen Attributen ist dieses Verfahren offenbar dem Naiven Bayes-Klassifikator vorzuziehen. Die Eignung eines Vollen Bayes-Klassifikator wäre hier ein interessanter Vergleichswert, aufgrund der diskreten Natur der Eingabedimensionen in den definierten Problemstellungen war dieser jedoch nicht anwendbar.

Das Mehrschichtige Perzeptron ist dabei als Regressionsverfahren noch mächtiger als die Klassfikatoren, kann jedoch diesen Vorteil zumindest bei den hier bearbeiteten Beispielen – mit in der Regel symbolischen Zielwerten – nicht ausspielen. Auffällig sind hier jedoch die vielen verschiedenen Möglichkeiten der Modellierung, die aufgrund der hohen Flexibilität dieses Verfahrens ausgenutzt werden können.

Insgesamt gesehen sind diese Verfahren auch ohne Kenntnis des modelltheoretischen Hintergrunds eines Problems geeignet, Entscheidungsregeln zu finden. Ihre Trainingsalgorithmen sind schnell und können auch ohne großen Datenbestand arbeiten. Dem Nutzer bleibt dabei das trainierte Modell verborgen, weswegen sich diese Verfahren auch ohne vertiefte Kenntnis des Verfahrens bedienen lassen. Dieser allgemeine Ansatz ist zwar in vielen Einsatzgebieten anwendbar, kann jedoch die Genauigkeit eines Verfahrens wie Hidden non-Markovian Models in dessen speziellem Einsatzgebiet nicht erreichen.

Im Gegensatz dazu stehen die Sequenzerkennungsverfahren, die im Rahmen dieser Arbeit eingesetzt wurden: Hidden Markov Models und Conditional Random Fields. Beide Verfahren erzielten in beiden Anwendungsszenarios schlechtere Ergebnisse als die Regressions- und Klassifikationsverfahren. Die erwartet guten Werte in den Kategorien Geschwindigkeit und Handhabbarkeit können dabei die mangelnde Genauigkeit zu keinem Zeitpunkt ausgleichen.

Dies geschieht trotz der Mehrinformation, die in diesen Modellen initial gekapselt ist, und trotz der Erzeugung der Daten durch einen stochastischen Prozess. Offensichtlich birgt die Annahme eines stochastischen Erzeugungsprozesses weniger Vorteile, als durch die Annahme der Markov-Eigenschaft Beschränkungen erwachsen. Dies lässt den Schluss zu, dass diese Modelle zwar in der Nische der partiell beobachtbaren Markov-Prozesse sehr gut geeignet sind, jedoch nicht die Flexibilität aufweisen, Prozesse ohne die Markov-Eigenschaft modellieren zu können.

Die Einführung von Zwischenzuständen zur Modellierung so genannter Phasentyp-Verteilungen ermöglicht zwar theoretisch die Abbildung jedes stochastischen Prozesses. Jedoch zeigte die Anwendung dieses Prinzips bei den gewählten Beispielen auch Grenzen auf. So kann zwar eine Phasenverteilung jede Verteilung beliebig genau annähern, jedoch steigt die Anzahl der dafür nötigen Zustände rapide an. Damit wird wiederum das Training der so modellierten Hidden Markov Modelle sehr aufwändig. Dies kann im ersten Anwendungsbeispiel, der Rekonstruktion des Maschinenverhaltens, beobachtet werden: Dort ist die Geschwindigkeit des Modells mit Phasenverteilungen deutlich langsamer als die des einfachen Hidden Markov Models, ohne dabei die Genauigkeit der Ergebnisse zu erhöhen. Der Ansatz, Phasenverteilungen in ein Hidden Markov Model zu integrieren, scheiterte sogar beim zweiten Anwendungsbeispiel und kann daher ohne weitere Betrachtung nicht empfohlen werden.

Conditional Random Fields konnten sich in den hier untersuchten Anwendungsbeispielen nicht als Alternative zu Hidden Markov Models etablieren. Neben den Genauigkeitswerten, die nicht höher als die der Hidden Markov Models waren, liegt dies auch an der geringeren Geschwindigkeit dieser Modelle, die in dieser Form nicht erwartet werden

konnte. Warum der in der Literatur beschriebene Geschwindigkeitsvorteil dieser Modelle sich hier nicht zeigt, kann an dieser Stelle nicht abschließend begründet werden.

Die Lösungen, die in dieser Arbeit mit Conditional Random Fields entwickelt wurden, schöpfen dabei aber nicht alle theoretischen Möglichkeiten aus. Die Definition komplexerer Feature-Funktionen für die einzelnen Modellzustände könnte hier eine Verbesserung der Performanz nach sich ziehen. Dennoch erfordert diese Vorgehensweise tiefgehende Kenntnisse der Conditional Random Fields und deren zugrunde liegenden Theorie, womit die Handhabbarkeit dieser Modelle gegenüber Hidden Markov Models stark abfällt.

Die guten Ergebnisse der Hidden non-Markovian Models bei der Lösung der beschriebenen Probleme können nicht überraschen, da diese bereits in den vorangegangen Forschungsarbeiten ([BKSH10] und [KBH10]) erzielt werden konnten. Jedoch scheint das Hidden non-Markovian Model, zumindest bei den hier betrachteten Verfahren, den einzigen Lösungsansatz mit annehmbarer Genauigkeit darzustellen. Dieses Verfahren zeichnet sich dabei in den Beispielen aus, mit Kenntnis des Modells trotz hohen Potentials für Mehrdeutigkeiten gute Klassifizierungen vornehmen zu können.

Die hohen Genauigkeitswerte dieses Ansatzes haben jedoch auch ihren Preis: Die Geschwindigkeit des Verfahrens ist in Relation zu den anderen Verfahren unverhältnismäßig gering und die Handhabung zu diesem Zeitpunkt noch schwierig. Aufgrund der verschiedenen Klassen, in denen diese Modelle eingeordnet werden, konnte bisher kein einheitlicher Lösungsansatz entwickelt werden. So konnten Hidden non-Markovian Models mit gewissen Annahmen auch in Echtzeitsystemen eingesetzt werden ([BKH11]), hier musste jedoch der Proxel-Algorithmus angewandt werden.

Dieser Algorithmus verfolgt alle möglichen Modellpfade und ist daher äußerst aufwändig. Das exponentielle Wachstum des Proxelbaums kann dabei mit bestimmten Methoden eingedämmt werden. Falls aber – wie im ersten Anwendungsbeispiel – keine Zusammenfassung der Proxel erfolgen kann, steigen Zeit- und Speicherverbrauch dieses Verfahrens schnell exponentiell an. Die Optimierung dieses Algorithmus ist daher Gegenstand der aktuellen Forschung (zum Beispiel [BKH10]).

Ein weiterer Nachteil des Verfahrens ist ein fehlender Trainingsalgorithmus. Mit Kenntnis des Modells und festgelegter Verteilungen ist zwar eine Parameterschätzung aus den Trainingsdaten möglich, ein Erlernen des Modells wie in anderen überwachten Maschinellen Lernverfahren ist jedoch nicht möglich.

Kapitel 4

Schluss

In diesem letzten Kapitel werden die Inhalte der Arbeit noch einmal zusammengefasst, bevor ein Fazit gezogen wird. Am Ende dieses Kapitels werden dann Aspekte diskutiert, die im Rahmen dieser Arbeit nicht beleuchtet wurden, und aufgeworfene und offene Fragen benannt.

4.1 Zusammenfassung

Hidden non-Markovian Models wurden in den letzten Jahren entwickelt, um die Einschränkungen der Hidden Markov Models aufzuheben. Hidden Markov Models sind dabei ein populäres Verfahren in der Analyse partiell beobachtbarer stochastischer Prozesse. Diese Prozesse müssen jedoch Markovsch sein, um sie mit diesem Verfahren zu bearbeiten. Hidden non-Markovian Models ermöglichen die Analyse nicht-Markovscher Prozesse und sind Thema aktueller Forschungen.

Im Rahmen dieser Forschung wurden verschiedene Anwendungsbeispiele mit Hidden non-Markovian Models erfolgreich bearbeitet. Da diese Modelle jedoch aufgrund ihrer vergleichsweise geringen Entwicklungszeit noch viele Nachteile bei der Anwendung aufweisen, kam die Frage auf, ob andere Verfahren diese Problemstellungen vergleichbar lösen können.

Diese Fragestellung wird im Rahmen dieser Arbeit bearbeitet. Dazu werden Maschinelle Lernverfahren eingesetzt, um das Verhalten der zugrunde liegenden stochastischen Prozesse durch Training mit Ein- und Ausgabetupeln zu erlernen. Es werden Verfahren aus den Bereichen *Regression* (Mehrschichtiges Perzeptron), *Klassifikation* (Bayes-Klassifikatoren und Entscheidungsbäume) und *Sequenzerkennung* (Hidden Markov Models und Conditional Random Fields) vorgestellt und angewandt. Dabei wird, neben der Betrachtung der Genauigkeit, auch eine Betrachtung der Geschwindigkeit und Handhabbarkeit dieser Verfahren, bezogen auf die ausgewählten Anwendungsbeispiele, vorgenommen.

Die dabei bearbeiteten Problemstellungen sind einerseits die Rekonstruktion des Verhaltens einer Produktionskette aus Prüfungsprotokollen und andererseits die Diagnostik von Krankheiten basierend auf Symptomverläufen von Patienten. Für beide Probleme wurden Lösungen mit den zu betrachtenden Verfahren entwickelt und umgesetzt.

Die Ergebnisse dieser Experimente zeigen, dass klassische Verfahren zwar schneller

und leichter zu handhaben sind als Hidden non-Markovian Models, jedoch deren Genauigkeit nicht erreichen können. Besonders fällt dabei auf, dass die Eignung der Hidden Markov Models, aus denen die Hidden non-Markovian Models entwickelt wurden, weit geringer als die der klassischen Regressions- und Klassifkationsverfahren ist. Dies zeigt deutlich, dass die nicht-Markovschen Modelle ein eigenes Einsatzgebiet beschreiben.

4.2 Fazit – Hidden non-Markovian Models berechtigt in der Mustererkennung

Trotz aller Nachteile, welche die Anwendung eines Hidden non-Markovian Models mit sich bringt, bleiben die Genauigkeitswerte dieser Lösungen unerreicht. Damit kann zur Lösung der hier bearbeiteten Probleme nur der Einsatz von Hidden non-Markovian Models empfohlen werden. Auch andere Problemstellungen, in denen ein verborgener stochastischer Prozess bekannt ist, sollten von Hidden non-Markovian Models mit akzeptablen Ergebnissen gelöst werden können.

Bei realweltlichen Problemen ist dafür jedoch sehr viel Modellierungsarbeit vonnöten, bei Echtzeit- beziehungsweise geschwindigkeitssensitiven Systemen müssen sogar Einschränkungen bei der Modellierung des verborgenen Prozesses gemacht werden, um die schnellen Hidden Markov Model Algorithmen anwenden zu können. Ist dies nicht möglich, können nur die Regressions- und Klassifikationsverfahren in hoher Geschwindigkeit zumindest teilweise akzeptable Ergebnisse produzieren.

Da der Einsatz der Sequenzerkennungsverfahren als gescheitert betrachtet werden muss, zeigt sich eine scheinbar triviale Schlussfolgerung. Falls ein Problem nicht in ein spezielles Einsatzgebiet fällt, gilt: Je weniger Annahmen durch das anzuwendende Verfahren getroffen werden, desto besser ist dessen Eignung. Dies wird vor allem durch das Mehrschichtige Perzeptron und durch Entscheidungsbäume gezeigt, die keine impliziten Annahmen über die Daten treffen. Hidden Markov Models und Conditional Random Fields sind zwar in ihren definierten Einsatzgebieten genau, schnell und gut handhabbar, deren Ansätze können jedoch nicht ohne Weiteres in andere Einsatzgebiete importiert werden.

Hidden non-Markovian Models können dabei noch nicht als vollwertiger Ansatz im Bereich des Maschinellen Lernens angesehen werden. Sollten jedoch einheitliche Lösungs- und Trainingsalgorithmen bereitgestellt werden, könnte sich dieses Verfahren als unverzichtbares Sequenzerkennungsverfahren für beliebige wertdiskrete und partiell beobachtbare stochastische Prozesse herausstellen. Dann könnten sie die Fähigkeit, diese Probleme zu bearbeiten, mit der Fähigkeit zum automatischen Erlernen von Wissen so kombinieren, dass detaillierte Kenntnisse über die theoretischen Grundlagen von Hidden non-Markovian Models nicht mehr nötig sind.

Ob klassische Maschinelle Lernverfahren grundsätzlich dazu geeignet sind, solche Prozesse ganz oder teilweise zu beschreiben, kann im Rahmen dieser Arbeit nicht vollständig geklärt werden. Dennoch haben die Experimente die Flexibilität dieser Verfahren deutlich aufgezeigt, da sie ohne große Datenmengen oder Modellinformationen Entscheidungsfunktionen finden können. Die Güte dieser Ergebnisse ist zwar verbesserungswürdig, könnte jedoch durch intensivere Modellierungsarbeit (mehr Trainingsdaten, andere Datenmodelle) erhöht werden.

4.3 Ausblick – HnMMs als Maschinelles Lernverfahren?

In diesem Abschnitt sollen Punkte angesprochen werden, die aus verschiedensten Gründen in dieser Arbeit nicht untersucht oder diskutiert wurden. Weiterhin werden Möglichkeiten zur Fortführung dieser Forschung benannt und die Bedeutung der Arbeit in einem größeren Zusammenhang erörtert.

Sollten die Hidden non-Markovian Models in Richtung eines Maschinellen Lernverfahrens weiterentwickelt werden, so müssen bestimmte Probleme gelöst werden. Das erste betrifft die Entwicklung eines solchen Modells. Dies ist bisher nur mit Kenntnis des zugrunde liegenden stochastischen Prozesses möglich. Auch wenn diese Arbeit gezeigt hat, dass eine Parametrisierung dieser Modelle aus Trainingsdaten möglich ist, wäre ein automatisches Erlernen des Modells wünschenswert.

Das zweite Problem ist die recht komplexe Anpassung der Lösungsalgorithmen auf ein spezielles Problem. Auch wenn in den vergangenen Jahren in diesem Bereich und auch bezüglich der Geschwindigkeit der Lösung gute Fortschritte erzielt wurden, scheint die Zusammenfassung der verschiedenen Bestandteile in einem einheitlichen Verfahrensbegriff, auch anwendbar in Echtzeitanwendungen, noch einige Jahre der Forschung zu beanspruchen. Hidden non-Markovian Models können sich in jedem Fall jedoch als wichtiger Ansatz herausstellen, zeitabhängige Muster zu beschreiben und zu erkennen.

Möglichkeiten, die in dieser Arbeit gewonnenen Erkenntnisse zu vertiefen, gibt es viele. Zunächst einmal sei da auf weitere Anwendungsbeispiele von Hidden non-Markovian Models verwiesen, beispielsweise im Bereich der Gestenerkennung. Interessant wäre dabei auch die Betrachtung von Anwendungsbeispielen, die mit anderen Verfahren zurzeit noch nicht umfassend bearbeitet werden können.

Auch wurden in dieser Arbeit grundsätzliche Versionen der vorgestellten Verfahren verwendet. Dabei gibt es sicherlich sehr viele Möglichkeiten zur speziellen Anpassung oder auch zur Kombination dieser Modelle, um die beschriebenen Probleme besser lösen zu können. Hier seien beispielhaft die Conditional Random Fields genannt, in denen sehr viel komplexere Modellelemente eingeführt werden können als dies hier getan wurde.

Weiterhin ist die Aufzählung der hier untersuchten Verfahren nicht komplett. Als Beispiele für nicht betrachtete Verfahren können regelbasierte Verfahren oder Evolutionäre Algorithmen genannt werden. Auch wurden unüberwachte Maschinelle Lernverfahren in dieser Arbeit nicht betrachtet. Aber auch aus diesem Bereich könnten Verfahren wie zum Beispiel Clustering oder Selbstorganisierende Karten für die hier beschriebenen Problemen anwendbar sein.

Trotz dieser mannigfaltigen Möglichkeiten hat diese Arbeit gezeigt, dass Hidden non-Markovian Models zumindest einen berechtigten Ansatz im Bereich der Mustererkennung darstellen. In einem breiteren Kontext wurde dabei die Fragestellung bearbeitet, ob die in dieser Arbeit vorgestellten Verfahren des Maschinellen Lernens beliebige stochastische Prozesse abbilden können. Auch wenn diese Frage nicht grundsätzlich geklärt werden konnte, zeigen doch die Ergebnisse, dass klassische Verfahren zumindest in eng gesteckten

Grenzen spezielle Problemstellungen, basierend auf stochastischen Prozessen, bearbeiten können.

Abseits von dieser Betrachtungsweise enthält diese Arbeit einen anwendungsbezogenen Vergleich mehrerer Maschineller Lernverfahren und ermöglicht Rückschlüsse auf deren Eignung in ähnlichen Fällen. Damit kann diese Arbeit dazu dienen, dem Leser einen Überblick über essentielle Verfahren des Machine Learning und deren grundsätzliche Eigenschaften in den Bereichen Genauigkeit, Geschwindigkeit und Handhabbarkeit bei der Lösung spezieller Problematiken zu bieten.

Literaturverzeichnis

[Alp10] Alpaydin, E.: *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2. Auflage, 2010.

[BBHK10] Berthold, M. R.; Borgelt, C.; Höppner, F.; Klawonn, F.: *Guide to Intelligent Data Analysis*. Springer Verlag, London, 1. Auflage, 2010.

[BKH10] Buchholz, R.; Krull, C.; Horton, G.: Efficient event-driven proxel simulation of a subclass of hidden non-markovian models. In *EUROSIM*. Prag, Czech Republic, September 2010.

[BKH11] Bosse, S.; Krull, C.; Horton, G.: Modeling of gestures with different exectution speeds: Are hidden non-markovian models applicable for gesture recognition. In *The 10th International Conference on Modeling and Applied Simulation*, S. 189 – 194. Rome, Italy, 2011.

[BKKN03] Borgelt, C.; Klawonn, F.; Kruse, R.; Nauck, D.: *Neuro-Fuzzy-Systeme*. Friedr. Vieweg & Sohn Verlag — GWV Fachverlage GmbH, Wiesbaden, 3. Auflage, 2003.

[BKSH10] Buchholz, R.; Krull, C.; Strigl, T.; Horton, G.: Using hidden non-markovian models to reconstruct system behavior in partially-observable systems. In *3rd International ICST Conference on Simulation Tools and Techniques*, 2010.

[Bra07] Bramer, M. A.: *Principles of Data Mining*. Undergraduate Topics in Computer Science. Springer Verlag, London, 2007.

[Eul06] Euler, S.: *Grundkurs Spracherkennung*. Friedr. Vieweg & Sohn Verlag — GWV Fachverlage GmbH, Wiesbaden, 1. Auflage, 2006.

[Fin08] Fink, G. A.: *Markov Models for Pattern Recognition - From Theory to Applications*. Springer Verlag, Berlin, Heidelberg, 2008.

[Fis36] Fischer, R.: The use of multiple measurements in axonomic problems. In *Annals of Eugenics 7*. Cambridge University Press, 1936.

[Ham50] Hamming, R.: Error detecting and error correcting codes. *Bell System Technical Journal*, Band 29, Nr. 2, S. 147–160, 1950.

[HMS01] Hand, D.; Mannila, H.; Smyth, P.: *Principles of Data Mining*. The MIT press, 2001.

[Hor02] Horton, G.: A new paradigm for the numerical simulation of stochastic petri nets with general firing times. In *Proceedings of the European Simulation Symposium 2002*, S. 129–136, 2002.

[HTF01] Hastie, T.; Tibshirani, R.; Friedman, J.: *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Verlag, New York, NY, 2001.

[ISO06] ISO: 9241-11 – Anforderungen an die Gebrauchstauglichkeit. Technischer Bericht, International Organization for Standardization, 2006.

[IWH06] Isensee, C.; Wickborn, F.; Horton, G.: Training hidden non-markov models. In *Proceedings of 13th International Conference on ANALYTICAL and STOCHASTIC MODELLING TECHNIQUES and APPLICATIONS*, S. 105–110. Bonn, 2006.

[KBH10] Krull, C.; Buchholz, R.; Horton, G.: Matching hidden non-markovian models: Diagnosing illnesses based on recorded symptoms. In *24th European Simulation and Modelling Conference*, S. 133 – 138, 2010.

[KH07] Krull, C.; Horton, G.: Expanded hidden markov models: Allowing symbol emissions at state changes. In *Proceedings of 14th International Conference on ANALYTICAL and STOCHASTIC MODELLING TECHNIQUES and APPLICATIONS*, S. 185–190. Prague, Czech Republic, 2007.

[KH08] Krull, C.; Horton, G.: The effect of rare events on the evaluation and decoding of hidden non-markovian models. In *Proceedings of the 7th International Workshop on Rare Event Simulation*. Rennes, France, 2008.

[KH09a] Krull, C.; Horton, G.: Hidden non-markovian models: Formalization and solution approaches. In *6th Vienna International Conference on Mathematical Modelling*, S. 682 – 693, 2009.

[KH09b] Krull, C.; Horton, G.: Solving hidden non-markovian models: How to compute conditional state change probabilities. In *21st European Modeling and Simulation Symposium*. Santa Cruz de Teneriffe, Spain, 2009.

[Mar09] Marsland, S.: *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press, Boca Raton, FL, 1. Auflage, 2009.

[MZ97] MacDonald, I. L.; Zucchini, W.: *Hidden Markov and Other Models for Discrete-valued Time Series*. Monographs on statistics and applied probability. Chapman & Hall, London, 1. Auflage, 1997.

[RN03] Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ, 2. Auflage, 2003.

[SDA+75] Shortliffe, E. H.; Davis, R.; Axline, S. G.; Buchanan, B. G.; Green, C.; Cohen, S. N.: Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the mycin system. *Computers and Biomedical Research*, Band 8, Nr. 4, S. 303 – 320, 1975.

[SG01] Sun, R.; Giles, L.: *Sequence Learning: Paradigms, Algorithms and Applications*. Springer Verlag, Heidelberg, 2001.

[SM06] Sutton, C.; Mccallum, A.: Introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[Wal04] Wallach, H. M.: Conditional random fields: An introduction. Technischer Bericht, University of Pennsylvania CIS, 2004.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, den 29. September 2011

Sascha Bosse