TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Artificial Intelligence

TUD-FI19-02 April 2019

Wissensrepräsentation und diagnostische Inferenz mittels Bayesscher Netze im medizinischen Diskursbereich

Sebastian Flügge, Sandra Zimmer and Uwe Petersohn

Technische Berichte
Technical Reports
ISSN 1430-211X

Technische Universität Dresden
Faculty of Computer Science
D-01062 Dresden
Germany

Wissensrepräsentation und diagnostische Inferenz mittels Bayesscher Netze im medizinischen Diskursbereich

Sebastian Flügge, Sandra Zimmer und Uwe Petersohn

2. Mai 2019

Abstract

Für die diagnostische Inferenz unter Unsicherheit werden Bayessche Netze untersucht. Grundlage dafür bildet eine adäquate einheitliche Repräsentation des notwendigen Wissens. Dies ist sowohl generisches als auch auf Erfahrungen beruhendes spezifisches Wissen, welches in einer Wissensbasis gespeichert wird. Concept Based Reasoning wird für die Diagnose-, Therapie- und Medikationsempfehlung und -evaluierung über generisches Wissen eingesetzt. Sonderfälle in Form von spezifischen Patientenfällen werden durch das Case Based Reasoning verarbeitet (vgl. [PGIB09]). Darüber hinaus erlaubt der Einsatz von Bayesschen Netze den Umgang mit Unsicherheit, Unschärfe und Unvollständigkeit. Es können so die gültigen allgemeinen Konzepte nach deren Wahrscheinlichkeit berechnet werden. Dazu werden verschiedene Inferenzmechanismen vorgestellt und anschließend im Rahmen der Entwicklung eines Prototypen evaluiert. Mit Hilfe von Tests wird die Klassifizierung von Diagnosen durch das Netz bewertet.

1 Einleitung

1.1 Medizinischer Hintergrund

Die Diagnostik in der Medizin ist die Zuordnung eines realen Krankheitsfalles zu einem idealen Abbild oder Modell eines Krankheitsfalles. Leitgrößen dieser Zuordnung bilden Anamnese, Symptome und Befunde. Der Mediziner entscheidet nach dem Maß der Ähnlichkeit, ob bei gegebenem Anamnesebild ein realer Fall dem idealen Abbild (generisches medizinisches Wissen) entspricht oder nicht.

Die Erkennung der Ähnlichkeit stellt sich relativ einfach dar, wenn

- es sich um ein invariantes Krankheitsbild handelt
- die Krankheit eine kurze Dauer aufweist
- die Leitsymptome sehr spezifisch sind oder
- die Diagnose eher spät bzw. immer zum gleichen Zeitpunkt innerhalb des Krankheitsverlaufes gestellt wird

Für die idealen Abbilder einer Krankheit gilt, dass

- in Fachbüchern häufig alle der Krankheit zuschreibbaren Merkmale das Idealbild dieser Krankheit beschreiben, obwohl in der Praxis dieses ideale Bild selten vorgefunden wird
- andere Krankheiten oder Merkmale (z. B. Alter, subjektives Empfinden), die Auswirkungen auf Krankheitsverläufe haben können, unbeachtet bleiben
- biologische Eigenschaften wie bspw. genetische Variabilität oder ontogenetische Modifikabilität unberücksichtigt bleiben und
- unter Umständen gleiche Krankheiten sehr unterschiedliche Verläufe und Konsequenzen haben können.

Während demnach Idealbilder die Variabilität von Krankheiten reduzieren, sind in der Realität Krankheiten häufig durch Heterogenität ausgezeichnet. Hinzu kommt, dass das in medizinischen Fachbüchern enthaltene oft strenge deterministische Wissen häufig nicht der ärztlichen Praxis entspricht und Mediziner dazu neigen, eigene individuelle medizinische Konzepte zu entwickeln. Dabei hat im Allgemeinen ein älterer und spezialisierterer Mediziner auch die größere Erfahrungskompetenz. Der Arzt besitzt mehr Kenntnis hinsichtlich der Variabilität von Krankheitsbildern, ihrer korrekten Zuordnung, ihrem Verlauf bzw. der Modifikation des Verlaufes unter Berücksichtigung therapeutischer Interventionen und in Abhängigkeit von Krankheitsstadien sowie der Prädiktion des künftigen Verlaufes. Diese Problematik gewinnt im Zusammenhang mit neuen medizinischen Paradigmen an Bedeutung. Beispielsweise können durch zeitigere Intervention mittels Früherkennung und Frühtherapie Krankheitsverläufe nachdrücklich modifiziert werden. Solche Überlegungen spielen vor allem dann eine Rolle, wenn es sich um nicht kausal heilbare Krankheiten handelt, die jedoch modifizierbar sind oder lang dauernden Behandlungsbedarf erfordern.

Unter Berücksichtigung dieser Rahmenbedingungen soll ein medizinisches System in der Lage sein, den Mediziner bei seinen komplexen Entscheidungen maßgeblich zu unterstützen.

1.2 Integration Case Based und Concept Based Reasoning

Für eine ganzheitliche computerunterstützte Behandlung von Patienten muss sowohl generisches als auch auf Erfahrungen beruhendes Wissen in einer Wissensbasis abgebildet werden. Für medizinische Standardfälle[1] ist häufig konzeptbasiertes Schließen, das auf generischem Wissen beruht, ausreichend. Da jedoch auch medizinisches Erfahrungswissen in der Praxis von großer Bedeutung ist, stellt das fallbasierte Schließen eine unerlässliche Ergänzung dar. Im System gespeicherte Erfahrungen medizinischer Fälle werden bei der Diagnose- und Therapiebestimmung berücksichtigt, gemäß der Annahme, dass ähnliche Probleme auch ähnliche Lösungen haben (vgl. [BKI06, S. 157]). Aufgrund der Verpflichtung zur Dokumentation, die sich aus § 10 Abs. 1 Musterberufsordnung für Ärzte (MBO-Ä) 1997 und aus dem Behandlungsvertrag ergibt, müssen medizinische Patienteninformationen gesetzlich gespeichert werden (vgl. [MBO]). Werden diese Patientendaten detailliert aufgenommen, bilden sie eine gute Basis für die Problemlösung mit fallbasiertem Schließen. Zudem sind diese Daten ein notwendiger Ausgleich für fehlendes generisches Wissen oder zur Aufklärung von generischen Abhängigkeiten (vgl. [PGIB09, S. 4]; [OP98, S. 2]).

Concept Based Reasoning wird für die Diagnose-, Therapie- und Medikationsempfehlung sowie -evaluierung eingesetzt. Auf der Basis repräsentierten generischen Wissens ist das Ergebnis

[1] Der Begriff der medizinischen Standardfälle wird häufig im Zusammenhang mit der evidenzbasierten Medizin verwendet und beschreibt „Normalfälle“ zur Behandlung von Patienten. In Einzelfällen wird der Mediziner hiervon abweichen.

des Concept Based Reasoning das Auffinden der spezifischsten Konzepte in der Konzeptbasis (vgl. [PGIB09, S. 7]). Grundlage dafür bilden Konzepte, die im Allgemeinen Generalisierungen von Fällen repräsentieren. Basierend auf Informationen über Anamnesen, Befunde, Symptome und Risikofaktoren sowie anwendbare therapeutische Maßnahmen können mittels Concept Based Reasoning Diagnosen, Befunde und Therapieempfehlungen abgeleitet und dem Mediziner vorgeschlagen werden.

Für die medizinische Problemlösung auf der Grundlage von spezifischem instanziiertem Wissen kann Case Based Reasoning eingesetzt werden. Es geht von der Annahme aus, dass ähnliche Problemsituationen auch ähnliche Problemlösungen erfordern. Es wird dabei menschliches Verhalten nachgeahmt. Tritt ein neues Problem auf, wird nach vergleichbaren in der Vergangenheit liegenden Situationen gesucht. Die bereits gemachte Erfahrung wird auf das aktuelle Problem komplett, teilweise oder adaptiert angewendet. Voraussetzung für das Case Based Reasoning bildet das Vorhandensein von spezifischen Erfahrungen in Form von Fällen, die in einer Fallbasis organisiert sind.

Darüber hinaus ist das medizinische Wissen verstärkt durch Unschärfe, Unsicherheit und Unvollständigkeit geprägt. Hauptgründe dafür sind die Komplexität in der Medizin und die Individualität von Patienten. Auch im klinischen Arbeitsalltag müssen Mediziner häufig mit Unsicherheiten umgehen. Befunde und klinische Daten liegen zum Zeitpunkt einer medizinisch notwendigen Entscheidung unter Umständen nicht vollständig vor. Weiterhin wird medizinisches Wissen ständig aktualisiert und Mediziner haben Schwierigkeiten, ihren persönlichen Wissensstand systematisch aktuell zu halten (vgl. [SS08, S. 153f]). Durch den Einsatz von Case Based Reasoning kann unscharfes Wissen in Form bereits gelöster Patientenfälle eine Ergänzung bieten.

Zur vollständigen Problemlösung werden Case Based Reasoning für die Verarbeitung von fallbasiertem spezifischen Wissen und Concept Based Reasoning für die generische Wissensverarbeitung eingesetzt. Integraler Bestandteil des Concept Based Reasoning bilden hier die Bayesschen Netze, die auch unscharfe Konzepte verarbeiten können. Der Umgang mit dieser Unsicherheit mittels Bayesscher Netze bildet den Gegenstand dieser Publikation.

1.3 State of the Art

Es existieren verschiedene Methoden, um mit Entscheidungen unter Unsicherheit umzugehen. Die wichtigsten Ansätze sind:

- **Bayessche Netze:**
  Die Werte der Verbundverteilung werden situationsbezogen in Abhängigkeit von Zufallsvariablen in einem Fachgebiet berechnet. Es wird das Konzept der bedingten Unabhängigkeiten von Ereignissen verfolgt.

- **Dempster-Shafer-Theorie (Evidenztheorie):**
  Basierend auf subjektiven Einschätzungen hinsichtlich der Zuverlässigkeit von Expertenaussagen (Glaubensmaße) erfolgt die quantitative Bewertung der Unsicherheiten.

- **Fuzzy-Logik:**
  Jedem Element wird mittels einer Zugehörigkeitsfunktion ein Wert zugeordnet, der den Grad der Zugehörigkeit zu einer unscharfen Menge repräsentiert. Prädikaten werden so Fuzzy-Mengen bzw. -Relationen als sog. Possibilitätsverteilungen zugeordnet, welche die Möglichkeit ihres Zutreffens subjektiv einschätzen.

Im Zusammenhang mit der medizinischen Entscheidungsunterstützung werden Bayesschen Netze bevorzugt, weil das medizinische Wissen zum Einen strukturiert abgebildet werden kann und zum Anderen die in medizinischen Systemen häufig vorliegenden A-Priori-Wahrscheinlichkeiten effizient für die Problemlösung genutzt werden können.

Allgemein kann unsicheres Wissen durch eine n-dimensionale Zufallsvariable $X_1 \dots X_n$ repräsentiert werden, deren Werte voneinander abhängig sein können. Dies bildet die Grundlage zur Konstruktion hochdimensionaler Wahrscheinlichkeitsverteilungen mittels Bayesscher Netze. Diese basieren auf lokalen probabilistischen Regeln, die statistische Abhängigkeiten zwischen Merkmalen berücksichtigen und sich damit eng an einer kausalen Modellierung der Welt orientieren. Deshalb lassen sich mit Bayesschen Netzen komplexe Problemstellungen mit einer verhältnismäßig geringen Zahl bedingter Wahrscheinlichkeiten modellieren (vgl. [SS05, S. 498ff]).

Ziel ist es nun, Methoden zu entwickeln, wie Bayessche Netze in diagnostische Systeme integriert werden können. Folgende Anforderungen soll der dafür implementierte prototypische Agent erfüllen:

1. Die Definition der Wissensbasis soll extern in Form eines abstrakten Dateiformates erfolgen, da sich eine Repräsentation in Form von Quellcode für die Modellierung größerer Netze als schwer handhabbar erwiesen hat.

2. Es sollen verschiedene Inferenzpakete getestet werden, wobei bei der Implementierung eine leichte Integration in bestehende Systeme sichergestellt werden soll.

3. Die Zeitkomplexität für die Berechnung soll beherrschbar sein.[2]

Für die Umsetzung wurden verschiedene Inferenzalgorithmen für Bayessche Netze untersucht. Das Inferenzpaket **ebay/bayesian_networks** ist eine Implementation des Junction-Tree-Algorithmus. Es wurde von EBay entwickelt, ist in Python implementiert und unter einer Open-Source-Lizenz für die Allgemeinheit verfügbar.[3] Eine genaue Beschreibung der Funktionsweise findet sich in [FP15] und [eBa14].

Infer.NET ist eine Entwicklung von Microsoft Research, welches für die Inferenz auf verschiedenen Probabilistischen Graphischen Modellen genutzt werden kann. So ist es damit auch möglich, Bayessche Netze zu modellieren. Die Repräsentation erfolgt mittels C#-Code, welcher die Klassen der Infer.NET-Bibliothek[4] nutzt. Dieses Modell wird von Infer.NET in optimierten C#-Code übersetzt, auf dem dann wiederholt Abfragen gestellt werden können (Abschnitt 4.3). Infer.NET implementiert drei Inferenzalgorithmen (vgl. [MWG$^+$18]):

1. **Expectation Propagation** (EP) ist eine Erweiterung von **Loopy Belief Propagation** (LBP, Abschnitt 3.2.5), welche auch kontinuierliche normalverteilte Variablen unterstützt. EP wird hier wie eine Implementation von LBP behandelt, da für den besprochenen Diskursbereich nur diskrete Netze relevant sind.

2. Der Referenzalgorithmus von Infer.NET ist **Variational Message Passing** (VMP, Abschnitt 3.4), welcher zur Inferenz auf komplexen gemischten Netzen aus kontinuierlichen und

[2] Für eine statistisch Bayes-optimale Entscheidung müssen die Wahrscheinlichkeiten bekannt sein. Es kann die Anzahl der benötigten Wahrscheinlichkeiten jedoch so weit steigen, dass bei einer großen Anzahl von Werten Rechnungen aufgrund der Zeitkomplexität nicht beherrschbar sind.

[3] Es gibt darüber hinaus eine Reihe weiterer Implementierungen für Bayessche Netze wie bspw. Analytica [Ana17], BayesiaLab [S.A17], Bayes Server [Ltd17], JavaBayes [Coz17] oder GeNIe/SMILE [Bay17].

[4] Seit 2018 steht neben der schon bisher nutzbaren kompilierten Bibliothek auch der Quellcode zur Verfügung und ist unter der MIT-Lizenz auf Github abrufbar.

diskreten Variablen entwickelt wurde. Die Inferenzleistung der beiden Algorithmen auf dem Kopfschmerznetz (Abbildung 8) wird in Abschnitt 5 untersucht.

3. Als dritter Algorithmus steht **Gibbs-Sampling** zur Verfügung, welches aufgrund des hohen Rechenaufwands für hinreichende Genauigkeit nicht näher untersucht wurde.

Die nachfolgende Tabelle gibt einen Überblick über die Eigenschaften der untersuchten Inferenzalgorithmen.

	Deterministisch?	Exaktes Ergebnis?	Konvergiert immer?	Effizient?	In Infer.NET verfügbar?
Loopy Belief Propagation	Ja	Nur in einfach verbundenen Netzen	Nein, aber für viele Anwendungen ausreichend	Ja (für diskrete Netze)	Ja, mit Expectation Propagation
Join Tree Belief Propagation	Ja	Ja	Ja	Ja (für diskrete nicht zu komplexe Netze)	Nein
Variational Message Passing	Ja	Nein	Ja	Ja	Ja

Tabelle 1: Übersicht der in dieser Arbeit betrachteten Inferenzalgorithmen für Bayessche Netze [MWG+18]

2 Repräsentation und Inferenz

Bayessche Netze konstruieren, basierend auf lokalen probabilistischen Regeln, eine hochdimensionale Wahrscheinlichkeitsverteilung. Dabei wird ein Bayessches Netz als gerichteter azyklischer Graph repräsentiert, in dem Knoten Zufallsvariablen sind und Kanten bedingte Abhängigkeiten zwischen den Variablen beschreiben. Die (gerichteten) Kanten repräsentieren den kausalen Einfluss eines Knotens (Elternknoten) auf einen anderen Knoten. Der Einfluss des Elternknotens auf die entsprechende lokale Zufallsvariable wird durch eine Tabelle der bedingten Wahrscheinlichkeiten jedem Knoten im Netz zugeordnet.

2.1 Definition eines Bayesschen Netzes

2.1.1 Definition der Struktur

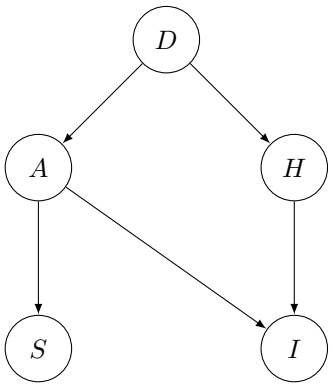
Ein Bayessches Netz ist ein Probabilistisches Graphisches Modell (PGM), das der Abbildung unscharfen Wissens in einer geschlossenen Domäne dient. Es besteht aus einem Graphen und lokalen Wahrscheinlichkeitsverteilungen (Abbildung 1). Es gelten folgende Eigenschaften:

- Jeder Knoten repräsentiert eine Zufallsvariable X mit einem Wertebereich, der diskret oder stetig sein kann.[5] Mit $\mathbf{\Pi}_X$ bezeichnen wir die Menge der Elternknoten, mit π_X ihre Instanziierungen.

[5] In der medizinischen Diagnostik sind die meisten Knoten nur binär.

- Die Beziehungen zwischen den Knoten des Graphen sind **gerichtet** und der Graph ist **zyklenfrei** (DAG).
- Jede Kante beschreibt eine Eltern-Kind-Beziehung zwischen zwei Knoten. Der Ausgangsknoten heißt **Elternknoten**, der Zielknoten heißt **Kindknoten**.
- Jeder Knoten enthält eine **lokale Wahrscheinlichkeitsverteilung**. Für Kindknoten ist diese Wahrscheinlichkeitsverteilung eine bedingte Wahrscheinlichkeitsverteilung $\mathbf{P}(X|\mathbf{\Pi}_X)$, die den Effekt der Elternknoten $\mathbf{\Pi}_X$ auf den Knoten X quantifiziert. Für Knoten mit $\mathbf{\Pi}_X = \emptyset$ gelten unbedingte A-priori-Verteilungen $\mathbf{P}(X)$.

Die nachfolgende Abbildung 1 zeigt ein Beispiel eines Bayesschen Netzes in der Medizin zur Berechnung der Wahrscheinlichkeit eines Schlaganfalls und Herzinfarkts. Diabetes Mellitus D hat möglicherweise Arteriosklerose (Gefäßverkalkung) A und Hypertonie (Bluthochdruck) H zur Folge, welche wiederum bewirken können, dass der Patient einen Herzinfarkt I hat. Außerdem kann Arteriosklerose zu einem Schlaganfall S führen. Die in der Abbildung angegebenen Wahrscheinlichkeitsbewertungen bzw. Verbundverteilungen sind fiktiv.



Diabetes mellitus D		
$P(D)$	d	$\neg d$
	0,2	0,8

Arteriosklerose A		
$P(A\|D)$	a	$\neg a$
d	0,2	0,8
$\neg d$	0,05	0,95

Hypertonie H		
$P(H\|D)$	h	$\neg h$
d	0,8	0,2
$\neg d$	0,2	0,8

Schlaganfall S		
$P(S\|A)$	s	$\neg s$
a	0,8	0,2
$\neg a$	0,6	0,4

Herzinfarkt I			
$P(I\|A,H)$	i	$\neg i$	
a,	h	0,8	0,2
a,	$\neg h$	0,8	0,2
$\neg a$,	h	0,8	0,2
$\neg a$,	$\neg h$	0,05	0,95

Abbildung 1: Beispiel eines einfachen Bayesschen Netzes in der Medizin

Je nach Gegebenheit des Netzes kann die Wahrscheinlichkeit a priori oder a posteriori ermittelt werden. Bei Knoten ohne Eltern müssen die Wahrscheinlichkeiten, die die Zufallsvariablen annehmen können, a priori, wie z. B. $P(A = i)\forall i$, spezifiziert werden. In Abbildung 1 ist dies der Fall bei dem Knoten *Diabetes mellitus*. Hier werden typischerweise Wahrscheinlichkeiten mit den booleschen Werten *true* oder *false* genutzt. Für Knoten mit Elternknoten werden bedingte Wahrscheinlichkeiten, wie bspw. $P(A = i|B = j, C = k)\forall i, j, k$ spezifiziert. Die bedingten Wahrscheinlichkeiten werden tabellarisch erfasst.[6] Sie ergeben sich aus der Kombination aller Ausprägungen der Elternknoten für alle Werte des Zielknotens (vgl. [SS05, S. 499]; vgl. [SS08, S. 181]).

[6]Die Definition und Quantifizierung der kausalen Abhängigkeiten wird im Allgemeinen von einem Experten der betrachteten Domäne übernommen, der die Tabellen mit den bedingten Wahrscheinlichkeitswerten füllt. Jedoch kann dies, vorallem wenn eine Variable viele Eltern besitzt, schwierig bis unmöglich sein. In diesem Fall werden häufig vereinfachende Strukturen wie bspw. *Noisy-OR* eingesetzt (Abschnitt 2.4.3)

2.1.2 Darstellung einer vollständigen gemeinsamen Verteilung

Die vollständige gemeinsame Verteilung, auch **Verbunddichte** genannt, enthält das gesamte unsichere Wissen über eine Domäne. Ein Bayessches Netz lässt sich als Darstellung einer solchen Verteilung auffassen. Sei $P(x_1, \ldots, x_n)$ ein generischer Eintrag in der Wahrscheinlichkeitstabelle, wobei x_1 bis x_n konkrete Wertzuweisungen zu den einzelnen Knoten sind. Die Wahrscheinlichkeit dieser Wertzuweisung lässt sich als Produkt der bedingten Wahrscheinlichkeiten der einzelnen Knoten wie folgt ausdrücken:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\pi_i). \tag{1}$$

π_i sind die in den Verteilungen $x_1, \ldots, x_n$ erscheinenden konkreten Instanziierungen der Elternknoten $\mathbf{\Pi}_i$.

2.2 Methode zur Konstruktion eines Bayesschen Netzes

Hier soll nun ein Verfahren zum Erstellen eines Bayesschen Netzes gezeigt werden, so dass das resultierende Netz die Domäne möglichst gut repräsentiert. Dafür werden bedingte Unabhängigkeiten benötigt, welche von (1) impliziert werden. Mit der Produktregel lässt sich (1) auch wie folgt schreiben:

$$P(x_1, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}, \ldots, x_1).$$

Ein wiederholtes Anwenden der Produktregel (auch Kettenregel genannt) ergibt folgendes Produkt:

$$\begin{aligned} P(x_1, \ldots, x_n) &= P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \ldots P(x_2|x_1)P(x_1) \\ &= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1). \end{aligned}$$

Verglichen mit (1) gilt somit für jede Variable X_i des Netzes und ihre Elternknoten $\mathbf{\Pi}_i$:

$$\mathbf{P}(X_i|X_{i-1}, \ldots, X_1) = \mathbf{P}(X_i|\mathbf{\Pi}_i) \tag{2}$$

unter der Voraussetzung $\mathbf{\Pi}_i \subseteq \{X_{i-1}, \ldots, X_1\}$. Aus Gleichung (2) geht hervor, dass ein Bayessches Netz eine Domäne nur dann korrekt repräsentiert, wenn ein Knoten X_i bei bekannten Eltern $\mathbf{\Pi}_i$ bedingt unabhängig von seinen restlichen Vorgängern in der Knotenreihenfolge ist (Abschnitt 2.3.2).

Folgender Algorithmus erzeugt ein Netz, welches diese Bedingungen erfüllt.

Algorithmus 1 (Konstruktion eines Bayesschen Netzes)
Knoten und Kanten eines Bayesschen Netzes lassen sich wie folgt bestimmen.

1. *Knoten: Bestimme die Menge von Zufallsvariablen X_i, welche für die Domäne relevant sind. Um eine möglichst kompakte Netzstruktur zu erreichen, sortiere sie in der Reihenfolge, dass Ursachen vor Wirkungen erscheinen.*[7] *Es ergibt sich $\{X_1, \ldots, X_n\}$.*

2. *Kanten: Führe für $i = 1$ bis n aus:*

[7] Damit ist unter Anderem sichergestellt, dass es keine Zyklen im resultierenden gerichteten Graphen gibt.

- *Wähle aus der Menge $\{X_1, \ldots, X_{i-1}\}$ eine minimale Teilmenge von Eltern $\mathbf{\Pi}_i$, welche die Gleichung der Verbunddichte erfüllt.*
- *Füge für jedes Element aus $\mathbf{\Pi}_i$ eine gerichtete Kante vom Elternknoten zum Knoten X_i ein.*
- *Definiere die bedingte Wahrscheinlichkeitsverteilung $\mathbf{P}(X_i|\mathbf{\Pi}_i)$.*

Mit anderen Worten: $\mathbf{\Pi}_i$ soll alle Knoten aus $\{X_1, \ldots, X_{i-1}\}$ enthalten, die X_i direkt beeinflussen.

Der beschriebene Algorithmus garantiert Azyklizität, da jeder Knoten nur mit Knoten verbunden ist, die in der nach Kausalität sortierten Knotenmenge vor ihm liegen.

2.3 Struktureigenschaften

2.3.1 Lokale Strukturierung und Kompaktheit

Bayessche Netze gehören zur Klasse der lokal strukturierten Systeme. Ein lokal strukturiertes System besteht aus Unterkomponenten, wobei jede Komponente nur mit einer begrenzten Anzahl anderer Komponenten interagiert, unabhängig von der Gesamtzahl der Komponenten. Ein solches System weist deshalb meist ein lineares statt ein exponentielles Komplexitätswachstum auf.

Übertragen auf Bayessche Netze heißt das, dass jede Zufallsvariable nur von einer begrenzten Anzahl anderer Zufallsvariablen direkt beeinflusst wird. Sei diese Anzahl mit der Konstante k bezeichnet und es wird ein Netz mit n booleschen Variablen betrachtet. Da für die Angabe der bedingten Wahrscheinlichkeiten pro Variable maximal 2^k Zahlen benötigt werden, sind nur $n2^k$ Zahlen für die Spezifikation des vollständigen Netzes nötig, während die vollständige gemeinsame Verteilung jedoch 2^n Zahlen enthält. Je größer also die Anzahl der Knoten ist, desto drastischer ist dieser Unterschied. Somit sind Bayessche Netze eine besonders kompakte Repräsentation einer vollständigen gemeinsamen Verteilung.

Diese Annahmen gelten für die meisten Domänen, aber es gibt auch Problemstellungen, in denen jede Variable von allen anderen beeinflusst wird. Modelliert als Bayessches Netz ergibt das ein *vollständig verbundenes* Netz, in dem jeder Knoten durch eine Kante mit jedem anderen Knoten verbunden ist. In diesem Fall übersteigt die erforderliche Informationsmenge zur Angabe der bedingten Wahrscheinlichkeiten des Netzes sogar die notwendige Informationsmenge zur Spezifikation einer entsprechenden vollständigen gemeinsamen Verteilung.[8]

In manchen Domänen sind Abhängigkeiten zwischen zwei Variablen sehr schwach ausgeprägt. Da die benötigten Wahrscheinlichkeiten, welche vom Netz berechnet werden sollen, selten sehr genau sein müssen, können diese Abhängigkeiten bei der Modellierung des Netzes außen vor gelassen werden, um ein kompakteres Netz zu erhalten. Eine besonders kompakte Struktur haben *einfach verbundene* Netze, sogenannte Polybäume[9]. Hier gibt es bei n Knoten nur $n - 1$ Kanten.

[8] Für ein vollständig verbundenes Netz gilt $k = n - 1$ und es werden $n2^{n-1}$ Zahlen benötigt, um die bedingten Wahrscheinlichkeiten für ein solches Netz anzugeben.

[9] Ein Polybaum ist ein gerichteter Graph, bei dem ein Knoten mehrere Wurzelknoten haben kann, aber bei dem kein Knoten von einem anderen über unterschiedliche Pfade erreichbar ist. Werden die Richtungen von den Kanten entfernt, so ergibt sich ein azyklischer Graph, der sich auch als Baum darstellen lässt.

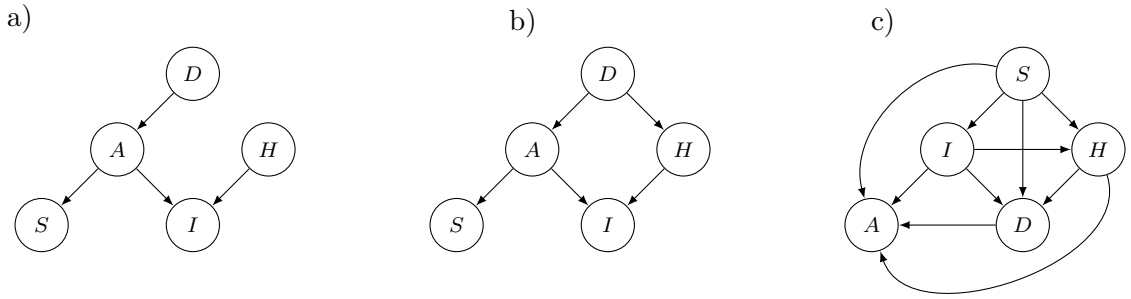


Abbildung 2: Unterschiedliche Verbundenheit von Bayesschen Netzen am Beispiel des Netzes aus Abbildung 1

In Abbildung 2 wird unter a) ein *einfach verbundenes* Netz gezeigt.[10]b) zeigt das originale Netz aus Abbildung 1 als *mehrfach verbundenes* Netz. Wird, wie in c) abgebildet, eine schlechte Knotenreihenfolge gewählt, ergibt sich im ungünstigsten Fall ein *vollständig verbundenes* Netz.

In einer lokal strukturierten Domäne hängt die Kompaktheit stark von der Knotenreihenfolge ab. Eine ungünstige Reihenfolge führt zu zusätzlichen Verbindungen im Netz, deren Wahrscheinlichkeitsverteilungen zudem sehr unnatürlich und daher schwierig zu bestimmen sind. Ein solches Netz codiert die bedingten Unabhängigkeitsaussagen zwischen den Variablen nur schlecht. Allgemein kann festgehalten werden, dass eine kausale Reihenfolge der Knoten, also von Ursache zu Wirkung, zu deutlich kompakteren Netzen führt als eine diagnostische Reihenfolge. In der medizinischen Domäne sollten also Diagnosen für Krankheiten vor Symptomen von Krankheiten gestellt werden. Auch erfahrene Ärzte geben lieber Wahrscheinlichkeiten für kausale statt für diagnostische Regeln an, was die Angabe der bedingten Wahrscheinlichkeiten zusätzlich vereinfacht.

2.3.2 Bedingte Unabhängigkeit

Wie bereits im letzten Abschnitt erwähnt, codiert eine Graphstruktur eine Menge von bedingten Unabhängigkeitsaussagen, welche nicht von der Quantifizierung des Netzes abhängig sind.

Eine Unabhängigkeitsaussage der Form *X und Y sind unabhängig bei bekanntem Z* heißt, dass für alle Kombinationen der Werte x, y und z folgende Gleichung gilt:

$$\mathbf{P}(\mathbf{x}|\mathbf{z}) = \mathbf{P}(\mathbf{x}|\mathbf{yz})$$

Anders formuliert: Ist z bekannt, wird die Kenntnis von y die Vorstellung von x nicht beeinflussen.

[10]Vom originalen Netz aus Abbildung 1 wurde der kausale Zusammenhang zwischen D und H weggelassen

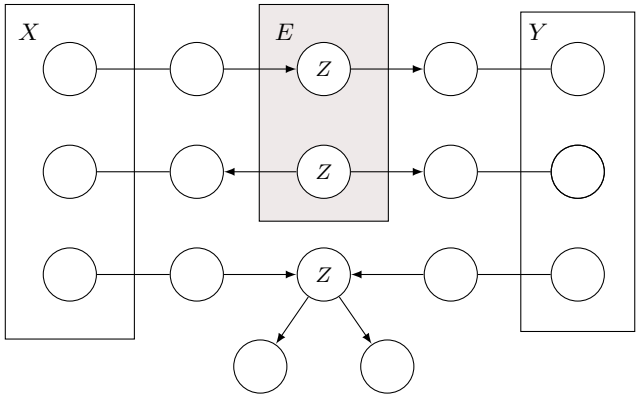


Abbildung 3: d-Separierbarkeit [RN95, S. 445]

Die Graphstruktur des Netzes codiert bedingte Unabhängigkeitsaussagen. Zwei und mehr Unabhängigkeitsaussagen können eine weitere Unabhängigkeitsaussage implizieren. Die hierbei geltenden Regeln sind als Graphoid-Axiome definiert.[11] d-Separation (Abbildung 3) ist eine graphentheoretische Relation, die alle diese ableitbaren Unabhängigkeiten abdeckt. Z d-separiert X und Y in einem DAG, wenn innerhalb des Netzes X und Y bei bekanntem Z unabhängig bezüglich der Graphoid-Axiome sind. Daraus lassen sich folgende Beziehungen ableiten:

- Jeder Knoten ist unabhängig von seinen Nichtnachfolgern bei gegebenen Eltern.
- Jeder Knoten ist unabhängig von allen anderen Knoten im Netz, wenn seine Markov-Decke (Abbildung 4) bekannt ist.

In Abbildung 3 sind Knoten aus der Menge X von Knoten aus der Menge Y unabhängig, wenn für alle Pfade zwischen X und Y eines der drei Kriterien erfüllt ist. Für Kriterium (1) und (2) muss Z bekannt sein (in der Evidenzmenge E enthalten). Für Kriterium (3) darf weder Z noch einer seiner Nachfolger in der Evidenzmenge E enthalten sein. Ungerichtete Kanten bezeichnen Kanten, bei denen die Richtung für die Erfüllung des Kriteriums keine Rolle spielt [RN95, S. 445].

Wie in Abschnitt 3 noch erläutert werden wird, sind die Unabhängigkeitsaussagen in Bayesschen Netzen wichtig, um die Komplexität von Inferenzalgorithmen zu reduzieren.

[11] Das Prädikat *Independence* $I(X, Z, Y)$ sei definiert mit X ist unabhängig von Y bei beobachtetem Z (Z ist Teil der Evidenzmenge E). Die Graphoid-Axiome sind dann:

1. Symmetrie: $I(X, Z, Y) \Rightarrow I(Y, Z, X)$.
2. Dekomposition: $I(X, Z, Y \cup W) \Rightarrow I(X, Z, Y) \wedge I(X, Z, W)$.
3. Schwache Vereinigung: $I(X, Z, Y \cup W) \Rightarrow I(X, Z \cup Y, W)$.
4. Kontraktion: $I(X, Z, Y) \wedge I(X, Z \cup Y, W) \Rightarrow I(X, Z, Y \cup W)$.
5. Schnitt: $I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W) \Rightarrow I(X, Z, Y \cup W)$.

Ein Graphoid ist ein Abhängigkeitsmodel M, welches unter den Regeln 1-5 abgeschlossen ist, ein Semi-Graphoid ist unter den Regeln 1-4 abgeschlossen.

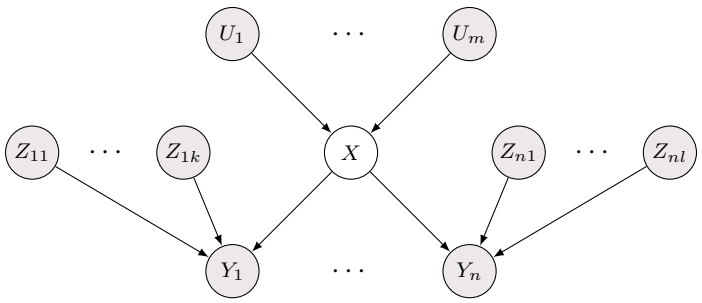


Abbildung 4: Markov-Decke des Knotens X, bestehend aus seinen Elternknoten $\{U_1, \ldots, U_n\}$, seinen Kindknoten $\{Y_1, \ldots Y_n\}$ und den anderen Elternknoten seiner Kindknoten $\{Z_{11}, \ldots, Z_{1k}, Z_{21}, \ldots, Z_{n1}, \ldots, Z_{nl}\}$ [RN09, S. 518].

2.4 Repräsentation bedingter Verteilungen

Die bedingten Verteilungen können als Wahrheitswertetabelle notiert werden. Dies wird aber schon bei wenigen Parametern ineffizient, da die Kombinationsmöglichkeiten aus Zuständen von Eltern- und Kindknoten und damit die Anzahl der Zeilen einer Wahrheitswertetabelle exponentiell steigen. Gefragt sind kompaktere Möglichkeiten zur Darstellung lokaler Beziehungen zwischen den Zufallsvariablen. Etabliert haben sich kanonische Wahrscheinlichkeitsmodelle, welche mit wenigen Parametern größere bedingte Verteilungen modellieren können [DD06]. Im Folgenden werden die Beziehungen erläutert, welche zur Modellierung des prototypischen Netzes verwendet wurden und vom implementierten Agenten verarbeitet werden können. Eine Erweiterung des Agenten um andere kanonische Modelle ist mit überschaubarem Aufwand möglich.

2.4.1 Vollständige bedingte Wahrscheinlichkeitstabelle

Der einfachste Fall ist eine vollständig ausgefüllte bedingte Wahrscheinlichkeitstabelle (BWT). Für jede Kombination von möglichen Zuständen eines Kindknotens und seiner Elternknoten muss eine Wahrscheinlichkeit gefunden werden. Diese Zustandskombination bildet zusammen mit der Wahrscheinlichkeit eine Zeile der Tabelle. Somit wird die direkte Notation einer solchen Tabelle schon bei wenigen Elternknoten unpraktikabel.

Kanonische Wahrscheinlichkeitsmodelle stellen eine Möglichkeit dar, eine bedingte Wahrscheinlichkeitsverteilung mit wenigen Parametern anzugeben. Mittels der jeweiligen Berechnungsformel der kanonischen Verteilung wird aus den Parametern eine vollständige bedingte Wahrscheinlichkeitstabelle konstruiert.

2.4.2 Deterministischer Knoten

Der einfachste Fall einer Beziehung zwischen Eltern- und Kindknoten ist eine deterministische, also keinerlei Unsicherheit enthaltende Relation. Der Wert des Kindknotens wird durch eine Funktion bestimmt, die die Werte der Elternknoten als Parameter entgegennimmt.

Definition 1 (Bedingte Wahrscheinlichkeitsverteilung)
Die bedingte Wahrscheinlichkeit ergibt sich wie folgt:

$$P(y|\mathbf{x}) = \begin{cases} 1 & \textit{wenn } y = f(x) \\ 0 & \textit{sonst.} \end{cases}$$

Eine Liste gängiger Funktionen ist in Tabelle 2 aufgeführt.

Funktionstyp	Variablentyp	Name	Definition
logisch	boolean	NOT	$y \Longleftrightarrow \neg x$
logisch	boolean	OR	$y \Longleftrightarrow x_1 \vee \ldots \vee x_n$
logisch	boolean	AND	$y \Longleftrightarrow x_1 \wedge \ldots \wedge x_n$
algebraisch	geordnet	MINUS	$y \Longleftrightarrow -x$
algebraisch	geordnet	INV	$y \Longleftrightarrow x_{\text{max}} - x$
algebraisch	geordnet	MAX	$y \Longleftrightarrow \max(x_1, \ldots, x_n)$
algebraisch	geordnet	MIN	$y \Longleftrightarrow \min(x_1, \ldots, x_n)$

Tabelle 2: Definition einiger deterministischer Funktionen

2.4.3 Stochastische Verknüpfungen (ICI-Modelle)

In der Praxis sind deterministische Beziehungen nicht besonders häufig, da das Zusammenspiel verschiedener Einflussfaktoren in der realen Welt in der Regel von Unsicherheit geprägt ist. Wird für die Elternknoten einer Beziehung in einem Bayesschen Netz aber angenommen, dass sie unabhängig in ihrem kausalen Einfluss auf einen Kindknoten sind, so lassen sich eine Reihe von Modellen für probabilistische Beziehungen formulieren, welche mit wenigen Parametern auskommen. Aus diesen Parametern kann mit entsprechenden Berechnungsformeln eine vollständige bedingte Wahrscheinlichkeitstabelle konstruiert werden. Diese **ICI-Modelle**[12] lassen sich in zwei Klassen einteilen: Noisy und Leaky. Beide funktionieren nach demselben Prinzip, wobei im Fall eines Noisy-Modells angenommen wird, dass der betrachtete Sachverhalt vom Modell vollständig erfasst wird. Beim Leaky-Modell hingegen wird ein zusätzlicher Leak-Parameter abgeschätzt, welcher die nicht explizit modellierten Fälle abdeckt. Im Folgenden werden zunächst die beiden grundsätzlichen ICI-Klassen erklärt und anschließend drei konkrete Noisy-ICI-Modelle abgeleitet.

Noisy-ICI-Modelle

Noisy-ICI-Modelle lassen sich in der theoretischen Betrachtung aus deterministischen Modellen entwickeln, indem Hilfsvariablen $Z_1, \ldots, Z_n$ einführt werden, wobei n die Anzahl der Elternknoten ist. Diese Hilfsvariablen dienen nur zur Erklärung des Modells, sind aber nicht Bestandteil der Modellierung. Y ist eine deterministische Funktion der Z_is, während jedes Z_i probabilistisch von X_i abhängt, dargestellt durch jeweils eine BWT $P(z_i|x_i)$. Diese BWTs sind die Parameter des Modells. Die bedingte Wahrscheinlichkeit $P(y|\mathbf{x})$ berechnet sich mit:

$$P(y|\mathbf{x}) = \sum_{\mathbf{z}} P(y|\mathbf{z}) \cdot P(\mathbf{z}|\mathbf{x}),$$

wobei $P(\mathbf{z}|\mathbf{x})$ das Produkt aller $P(z_i|x_i)$ ist:

$$P(\mathbf{z}|\mathbf{x}) = \prod_i P(z_i|x_i).$$

[12] Diez und Druzdzel sprechen von *„independence of causal influence“*, daher der Ausdruck *„ICI“* [DD06].

Für alle Noisy-ICI-Modelle gilt:

$$P(y|\mathbf{x}) = \sum_{\mathbf{z}|f(\mathbf{z})=y} \prod_i P(z_i|x_i). \tag{3}$$

Leaky-ICI-Modelle

Beim Noisy-ICI-Modell wird angenommen, dass alle Variablen, die Einfluss auf den Knoten Y haben, im Modell explizit enthalten sind. Diese Modellierung ist aber für Probleme der realen Welt nicht immer realisierbar oder wünschenswert. In einem reduzierten Modell sind nur die Variablen des Systems explizit enthalten, für die sich die Parameter aus den zur Verfügung stehenden Daten gut ermitteln lassen oder die für den modellierten Sachverhalt besonders relevant sind. Der Einfluss implizierter Variablen $\mathbf{V_I}$ auf Y kann durch den Leak-Parameter $P(z_L)$ abgeschätzt werden, wobei Z_L eine virtuelle Variable ist, welche den Effekt von $\mathbf{V_I}$ zusammenfasst.

$$P(y|\mathbf{x}) = \sum_{\mathbf{z}|f(\mathbf{z})=y} \prod_{i|X_i \in \mathbf{X}} P(z_i|x_i) \sum_{z_L|f(\mathbf{z},z_L)=y} P(z_L) \tag{4}$$

Noisy-OR

Beim Noisy-OR ist Y eine unscharfe OR-Verknüpfung über seinen Elternknoten X_i. Wird eine Noisy-OR-Verknüpfung kausal interpretiert, so ist jedes X_i eine mögliche Ursache von Y. Die bedingten Wahrscheinlichkeiten Z_i deuten an, ob Y von X_i erzeugt wurde. Die Bezeichnung „noisy“ bezieht sich also auf die Möglichkeit, dass eine präsente Ursache x_i keine Wirkung y erzeugt. Für das hier beschriebene Modell bedeutet ein $\neg z_i$, dass X_i nicht zu Y geführt hat, egal, ob es mit $X_i = \neg x_i$ nicht präsent war oder ob ein Inhibitor I_i verhindert hat, dass es $Y = y$ erzeugt. Mit q_i wird die Wahrscheinlichkeit angegeben, dass ein solcher Inhibitor I_i aktiv ist.[13]

Beispiel 1 (Noisy-OR)
Sei ein Herzinfarkt (I) die Folge von Arteriosklerose (Gefäßverkalkung) (A), Hypertonie (Bluthochdruck) (H) oder Endokorditis (Entzündung der Herzinnenhaut) (E), dann bedeutet

$$q_A = P(\neg i|a, \neg h, \neg e) = 0,1,$$

dass die Erzeugung eines Herzinfarktes durch Gefäßverkalkung kausal unabhängig von den anderen Ursachen verhindert wurde. Für die beiden anderen Ursachen gilt

$$q_H = P(\neg i|\neg a, h, \neg e) = 0,2 \textit{ und}$$
$$q_E = P(\neg i|\neg a, \neg h, e) = 0,6.$$

Die Wahrscheinlichkeit, dass Y durch X_i erzeugt wurde, ist

$$c_i = P(+z_i| + x_i) = 1 - q_i. \tag{5}$$

[13] Ereignisse werden immer wahrscheinlicher, wenn mehrere Bedingungen erfüllt sind. Wenn bspw. Krankheit A einen Herzinfarkt mit einer Wahrscheinlichkeit von $c_A = 1 - q_A$ erzeugt und Krankheit B mit $c_B = 1 - q_B$, dann ergibt sich eine Wahrscheinlichkeit für einen Herzinfarkt mit $1 - q_A q_B$ für das Vorliegen beider Krankheiten. Dieser Ausdruck ist größer als c_A und c_B.

Wenn X_i nicht auftritt, kann es Y auch nicht verursachen, weshalb gilt:

$$P(+z_i|\neg x_i) = 0. \tag{6}$$

Aus 5 und 6 folgt die Parameter-Tabelle für Noisy-OR:

$P(z_i\|x_i)$	$+x_i$	$\neg x_i$
$+z_i$	c_i	0
$\neg z_i$	$1 - c_i$	1

Tabelle 3: Parameter in BWT-Form für Noisy-OR für $X_i \rightarrow Y$

Um die vollständige BWT mit Gleichung 3 zu berechnen, seien zunächst I_+ und $I_\neg$ als Mengen der Indizes der Variablen definiert, die in einer Konfiguration binärer Elternknoten *true* bzw. *false* annehmen. Formal wird dies wie folgt ausgedrückt:

$$I_+(\mathbf{v}) = \{i|V_i \text{ nimmt Wert } +v_i \text{ in } \mathbf{v} \text{ an}\} \tag{7}$$
$$I_\neg(\mathbf{v}) = \{i|V_i \text{ nimmt Wert } \neg v_i \text{ in } \mathbf{v} \text{ an}\}. \tag{8}$$

Unter Berücksichtigung der Tatsache, dass $f_{\text{OR}}(\mathbf{z}) = \neg y$ nur für die Konfiguration $(\neg z_1, \ldots, \neg z_n)$ gilt, ergibt sich aus Gleichung 3:

$$P(\neg y|\mathbf{x}) = \prod_{i=1}^{n} P(\neg z_i|x_i) = \prod_{i \in I_+(\mathbf{x})} P(\neg z_i| + x_i) \cdot \prod_{i \in I_\neg(\mathbf{x})} P(\neg z_i|\neg x_i).$$

Mit Einsetzen der Parameter aus Tabelle 3 lautet die Berechnungsvorschrift für die vollständige bedingte Wahrscheinlichkeitsverteilung einer Noisy-OR-Verknüpfung

$$P(\neg y|\mathbf{x}) = \prod_{i \in I_+(x)} q_i = \prod_{i \in I_+(x)} (1 - c_i). \tag{9}$$

Über das Gegenereignis wird $P(+y|\mathbf{x}) = 1 - P(\neg y|\mathbf{x})$ berechnet. Die komplette Berechnung einer Tabelle für das Beispiel Herzinfarkt zeigt Tabelle 4.

Endokorditis (E)	Hypertonie (H)	Arteriosklerose (A)	$P(Herzinfarkt)$ ($P(I)$)	$P(\neg Herzinfarkt)$ ($P(\neg I)$)
false	*false*	*false*	**0,0**	**1,0**
false	*false*	*true*	$\mathbf{0,3} = 1 - 0,7$	**0,7**
false	*true*	*false*	$\mathbf{0,4} = 1 - 0,6$	**0,6**
false	*true*	*true*	$0,58 = 1 - 0,42$	$0,42 = 0,6 \cdot 0,7$
true	*false*	*false*	$\mathbf{0,6} = 1 - 0,4$	**0,4**
true	*false*	*true*	$0,72 = 1 - 0,28$	$0,28 = 0,4 \cdot 0,7$
true	*true*	*false*	$0,76 = 1 - 0,24$	$0,24 = 0,4 \cdot 0,6$
true	*true*	*true*	$0,832 = 1 - 0,168$	$0,168 = 0,4 \cdot 0,6 \cdot 0,7$

Tabelle 4: Noisy-OR am Beispiel von Ursachen für Herzinfarkt

Ein Herzinfarkt (I) kann kausal unabhängig von Endokorditis (E), Hypertonie (H) und Arteriosklerose (A) verursacht werden. Die Wahrscheinlichkeiten, dass ein Herzinfarkt unter einer einzelnen dieser Ursachen nicht eintritt, seien $q_E = 0,4$, $q_H = 0,6$ und $q_A = 0,7$. Mithilfe dieser Inhibitorwahrscheinlichkeiten lässt sich die komplette BWT für $P(I|A,H,E)$ berechnen. Die Werte, die in einer Parameter-BWT eines Noisy-OR auftreten, sind in der Tabelle 4 fett markiert.

Noisy-MAX

Das Noisy-MAX-Modell stellt eine Verallgemeinerung des Noisy-OR für eine Variable Y mit mehr als zwei Zuständen dar. Z_i repräsentiert hier den Wert von Y, der durch X_i hervorgerufen wird. Den resultierenden Wert von Y liefert das größte z_i, es gilt also $y = f_{\text{MAX}}(\mathbf{z})$. Alle Z_is müssen dieselbe Domäne haben wie Y. Die Noisy-MAX-Parameter für die Verbindung $X_i \to Y$ sind

$$c_{z_i}^{x_i} = P(z_i|x_i), \tag{10}$$

wobei jedes $c_y^{x_i}$ als Wahrscheinlichkeit verstanden werden kann, dass X_i mit dem Wert x_i den Wert von Y auf y setzt. Die vollständige BWT für ein Noisy-MAX-Modell errechnet sich, indem zuerst $P(Y \leq y|\mathbf{x})$ für alle Werte von y und alle Konfigurationen $\mathbf{x}$ errechnet wird:

$$P(y \leq y|\mathbf{x}) = \sum_{\mathbf{z}|f_{\text{MAX}}(\mathbf{z})\leq y} \prod_i c_{z_i}^{x_i} = \sum_{z_1 \leq y} \cdots \sum_{z_n \leq y} \prod_i c_{z_i}^{x_i} = \prod_i \left(\sum_{z_i \leq y} c_{z_i}^{x_i} \right).$$

Daraus resultiert folgende Berechnung für die akkumulierten Parameter:

$$C_y^{x_i} = \sum_{z_i \leq y} c_{z_i}^{x_i} \tag{11}$$

und damit

$$P(Y \leq y|\mathbf{x}) = \prod_i C_y^{x_i}. \tag{12}$$

Die Werte der BWT können nun wie folgt errechnet werden:

$$P(y|\mathbf{x}) = \begin{cases} P(Y \leq y|\mathbf{x}) - P(Y \leq y-1|\mathbf{x}), & \text{wenn } y \neq y_{\min} \\ P(Y \leq y|\mathbf{x}), & \text{wenn } y = y_{\min} \end{cases} \tag{13}$$

Noisy-AND

Die Elternknoten einer Noisy-AND-Verknüpfung lassen sich als Bedingungen interpretieren, die wahr sein müssen, damit Y wahr wird, wobei die Unschärfe dadurch erzeugt wird, dass jede Bedingung unterdrückt oder ersetzt werden kann. Inhibitor-Wahrscheinlichkeiten funktionieren wie beim Noisy-OR: q_i ist die Wahrscheinlichkeit, dass I_i aktiv ist, und wieder gilt $c_i = 1 - q_i$. Ist kein Inhibitor aktiv, ist $c_i = 1$. Die Wahrscheinlichkeit, dass das i-te Substitut X_i ersetzt, wenn die Bedingung nicht erfüllt ist, wird mit s_i bezeichnet, und es gilt $s_i = 0$, wenn es kein Substitut für X_i gibt. Allgemein gilt $c_i \cong 1$ und $s_i \cong 0$. Es ergibt sich folgende BWT eines Noisy-AND-Parameters $P(z_i|x_i)$:

$P(z_i\|x_i)$	$+x_i$	$\neg x_i$
$+z_i$	c_i	s_i
$\neg z_i$	$1-c_i$	$1-s_i$

Tabelle 5: Parameter in BWT-Form für Noisy-AND für die Verbindung $X_i \rightarrow Y$

Die Berechnung der gesamten BWT erfolgt nach folgender Formel:

$$P(+y|\mathbf{x}) = \prod_i P(+z_i|x_i) = \prod_{i \in I_+(\mathbf{x})} c_i \prod_{j \in I_\neg(\mathbf{x})} s_j.$$

3 Inferenzmechanismen

Die wichtigste Operation in Bayesschen Netzen stellt die Inferenz dar. Sie widmet sich der Frage, welche Wahrscheinlichkeitsverteilung eine Variable X unter Beobachtung bestimmter Werte bei anderen Variablen annimmt. Gegeben ist dabei die Evidenz e als Menge der mit beobachteten Werten instanziierten Variablen. Die Variable X repräsentiert die Abfragevariable. Gesucht ist die Posteriori-Wahrscheinlichkeit mit $P(X = x|e)$[14].

Zunächst wird zur Veranschaulichung mit der Methode der Aufsummierung eine einfache und exakte Inferenzmethode vorgestellt (Abschnitt 3.1). Es wird sich jedoch zeigen, dass der Berechnungsaufwand mit dieser Methode mit der Anzahl der Variablen exponentiell steigt. Für die Praxis sind folglich effizientere Verfahren interessant, welche im Weiteren vorgestellt werden:

- Der Algorithmus der Glaubenspropagation (Abschnitt 3.2) ist für einfach verbundene Netze effizient und exakt und es gibt mit der Inferenz in Clusterbäumen (Abschnitt 3.2.4) und Loopy Belief Propagation (Abschnitt 3.2.5) Ansätze, die ihn auch für mehrfach verbundene Netze verwendbar machen (vgl. [Pea82], vgl. [KP83]).
- Gibbs Sampling ist eine Anwendung von Monte-Carlo-Markov-Chain-Methoden (MCMC) auf die Inferenz in Bayesschen Netzen (Abschnitt 3.3) (vgl. [GG84]).
- Variational Message Passing ist ein moderner Algorithmus, um auch komplexe Verteilungen effizient zu approximieren (Abschnitt 3.4) (vgl. [JGJS99]).

3.1 Inferenz durch Aufsummierung

Jede bedingte Wahrscheinlichkeit kann durch Aufsummierung der Terme aus der vollständigen gemeinsamen Verteilung berechnet werden. Für eine einzelne Abfragevariable X erfolgt das mit der Gleichung

$$\mathbf{P}(X|\mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e}) = \alpha \sum_y \mathbf{P}(X, \mathbf{e}, \mathbf{y}),$$

wobei α ein Normierungsfaktor ist und $\mathbf{y}$ Instanzen aller nicht beobachteten Variablen $\mathbf{Y}$ (ohne X) sind.

[14] Die Berechnung der Posteriori-Wahrscheinlichkeiten ist komplexitätstheoretisch sehr schwierig. Die Effizienz der Algorithmen richtet sich dabei nach den Eigenschaften des Netzes. Bei optimaler Netzmodellierung ist für reale Problemstellungen jedoch eine effiziente Berechnung möglich.

Algorithmus 2 (Inferenz durch Aufsummierung)
Der Algorithmus läuft in zwei Schritten ab [RN09, S. 525]:

1. *Mit der Gleichung der Verbunddichte und Anwendung der Produktregel werden die Terme $P(x, \mathbf{e}, \mathbf{y})$ als Produkte aus den bedingten Wahrscheinlichkeiten des Netzes für jede mögliche Belegung der nicht beobachteten Variablen berechnet.*

2. *Anschließend werden die Einzelwahrscheinlichkeiten aufsummiert.*

Der Algorithmus zur Inferenz durch Aufsummierung kann wie folgt in Pseudocode abgebildet werden.

Algorithmus 3 (Pseudocode für Inferenz durch Aufsummierung)

```
1:  function ENUMERATION-ASK(X, e, bn)
2:      return eine Verteilung über X
3:      Input X, die Abfragevariable
4:      Input e, beobachtete Werte für die Variablen E
5:      Input Y, alle nicht beobachteten Variablen
6:      Input bn, ein Bayessches Netz mit Variablen {X} ∪ E ∪ Y
7:      P(X|e) ← eine Verteilung über X, anfänglich leer
8:      for all Wert x von X do
9:          Erweitere e mit Wert x für X
10:         P(x) ← ENUMERATION-ALL(VARS(bn), e)
11:     end for
12:     return NORMALIZE(P(X|e))
13: end function
```

```
1:  function ENUMERATION-ALL(vars, e)
2:      return eine reelle Zahl
3:      if ISEMPTY(vars) then
4:          return 1.0
5:      end if
6:      Y ← FIRST(vars)
7:      if Y hat den Wert y in e then
8:          return P(y|parents(Y)) × ENUMERATION-ALL(REST(vars), e)
9:      else
10:         return Σ_y P(y|parents(Y)) × ENUMERATION-ALL(REST(vars), e_y)
11:         wobei e_y gleich e erweitert mit Y = y ist
12:     end if
13: end function
```

Das Verfahren ist äquivalent zur Tiefensuche in einem Baum und hat somit eine lineare Speicherkomplexität, aber eine exponentielle Zeitkomplexität.

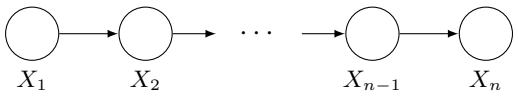
Wird die Berechnung in einem Ausdrucksbaum visualisiert, wird deutlich, dass viele Teilberechnungen mehrfach durchgeführt werden. Ein Zwischenspeichern der Ergebnisse dieser Berechnungen, um sie später wiederzuverwenden, senkt die benötigte Rechenzeit deutlich. Ebenso sind sämtliche Variablen, die keine Vorfahren einer Abfrage- oder Evidenzvariablen sind, für die Abfrage irrelevant und können vor der Berechnung entfernt werden. Ein Algorithmus, der dies leistet, ist die sogenannte Variableneliminierung, die in [RN09] beschrieben wird.

3.2 Inferenz durch Glaubenspropagation

Glaubenspropagation ist ein Inferenzverfahren, welches es ermöglicht, die Posteriori-Wahrscheinlichkeiten für alle Knoten des Netzes effizient zu berechnen. Jedoch ist der Basisalgorithmus insofern eingeschränkt, dass er nur auf einfach verbundenen, also polybaumförmigen Netzen angewendet werden kann. Zunächst soll die Kernidee des Algorithmus, das sogenannte Message Passing gezeigt werden. Danach wird beschrieben, wie dieses Prinzip auf komplexere Netze übertragen werden kann. Zum Schluss werden Ansätze diskutiert, mit denen die Einschränkungen des Basisalgorithmus teilweise aufgehoben werden.

3.2.1 Message Passing

Die Grundidee der Glaubenspropagation ist das iterative Aktualisieren der bedingten Wahrscheinlichkeitsverteilung eines Knotens anhand seiner Nachbarknoten. Das Prinzip soll zunächst an einem ganz einfachen Netz demonstriert werden. Dieses Netz habe eine kettenförmige Struktur, die Knoten $X_1, X_2, \ldots, X_{n-1}, X_n$ seien linear aneinandergereiht.



Faktorisierung der Verbunddichte
Die Randverteilung[15] einer beliebigen Variable X_i lässt sich durch Aufsummierung über alle möglichen Werte aller anderen Variablen errechnen:

$$P(X_i) = \sum_{X_1} \sum_{X_2} \cdots \sum_{X_{i-1}} \sum_{X_{i+1}} \cdots \sum_{X_n} P(\mathbf{X}).$$

Definition der Nachrichten
Es kann nun μ_β als die zwischen den Knoten zu übermittelnde Nachricht iterativ auf der Knotenkette zurück bis zum gewünschten Knoten X_i berechnet werden:

$$\begin{aligned} \mu_\beta(X_{n-2}) &= \sum_{X_{n-1}} P(X_{n-1}|X_{n-2})\mu_\beta(X_{n-1}) \\ &\cdots \\ \mu_\beta(X_i) &= \sum_{X_{i+1}} P(X_{i+1}|X_i)\mu_\beta(X_{i+1}). \end{aligned}$$

Ähnlich kann auch vom anderen Ende der Kette aus verfahren werden:

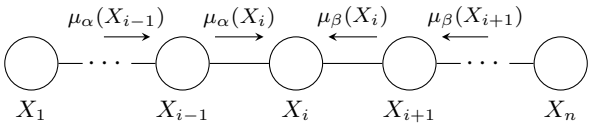
$$\begin{aligned} \mu_\alpha(X_2) &= \sum_{X_1} P(X_1)P(X_2|X_1) \\ &\cdots \\ \mu_\alpha(X_i) &= \sum_{X_{i-1}} P(X_i|X_{i-1})\mu_\alpha(X_{i-1}). \end{aligned}$$

[15] Die Randverteilung, auch Marginalverteilung, wird durch die Randwahrscheinlichkeiten (Marginale) gebildet. Der Name kommt von der Randhäufigkeit, die als Summe von Häufigkeiten eines Merkmals am Rand einer Kontingenztafel steht. Randwahrscheinlichkeiten werden nach demselben Prinzip durch Aufsummierung der bedingten Wahrscheinlichkeiten einer Variable berechnet. Dieser Vorgang wird auch Marginalisieren genannt.

Die Randverteilung $P(X_i)$ kann nun als Produkt der von beiden Enden kommenden Beiträge berechnet werden:

$$P(X_i) = \mu_\alpha(X_i)\mu_\beta(X_i).$$

Somit kann $\mu_\alpha(X_i)$ als Nachricht aufgefasst werden, die von Knoten X_{i-1} zu Knoten X_i übermittelt wird, und $\mu_\beta(X_i)$ als Nachricht von Knoten X_{i+1} zu Knoten X_i.



Jede ausgehende Nachricht wird durch Multiplizieren der eingehenden Nachricht mit der lokalen bedingten Wahrscheinlichkeit und Aufsummieren über alle möglichen Werte des sendenden Knotens erhalten.

Vollständiges Message Passing
Um die Randwahrscheinlichkeiten aller Knoten X_i der Kette zu berechnen, wird nach folgendem Schema verfahren:

1. Senden der Nachrichten μ_α, mit $\mu_\alpha(X_1)$ beginnend bis $\mu_\alpha(X_n)$
2. Zurücksenden der Nachrichten μ_β, mit $\mu_\beta(X_n)$ beginnend bis $\mu_\beta(X_1)$

Berücksichtigung von Evidenz
Soll nun die Posteriori-Wahrscheinlichkeit unter gegebenen Beobachtungen $\mathbf{e}$ berechnet werden, wird beim Berechnen der Nachrichten des beobachteten Knotens einfach der beobachtete Wert verwendet, statt über alle möglichen Werte des Knotens aufzusummieren. Anschließend muss normalisiert werden:

$$\mathbf{P}(X_i|\mathbf{e}) = \frac{\mathbf{P}(X_i, \mathbf{e})}{\sum_{X_i} P(X_i, \mathbf{e})}. \tag{14}$$

Im Abschnitt 3.2.3 wird das eben beschriebene Message-Passing-Schema auf beliebige Polybaumnetze verallgemeinert. Dafür wird jedoch eine geeignetere graphische Struktur benötigt, die im nächsten Abschnitt eingeführt wird.

3.2.2 Faktorgraphen

Wie schon im letzten Abschnitt gezeigt, macht sich Glaubenspropagation die Faktorisierung von Bayesschen Netzen für eine iterative Berechnung zunutze. Daher lässt sie sich auf einer graphischen Struktur, die diese Faktorisierung besonders hervorhebt, leichter implementieren und erklären.

Definition 2 (Faktorgraph)
Ein Faktorgraph ist ein bipartiter Graph[16] *mit folgenden Eigenschaften:*

[16] Bipartite Graphen beschreiben die Beziehungen zwischen den Elementen zweier Mengen.

- *Er besteht aus zwei verschiedenen Knotentypen, Variablen und Faktoren, sowie ungerichteten Kanten zwischen jeweils einer Variable und einem Faktor.*[17]
- *Ein Knoten der Variablenmenge repräsentiert eine Zufallsvariable X_i und wird durch einen Kreis dargestellt.*
- *Ein Knoten der Faktormenge repräsentiert einen Faktor f_j, dargestellt durch ein Quadrat. Dieser beschreibt eine Wahrscheinlichkeitsverteilung über die angrenzenden Zufallsvariablen. Dies kann eine bedingte Wahrscheinlichkeitsverteilung, aber auch das Produkt von Wahrscheinlichkeitsverteilungen bis hin zu einer Verbundwahrscheinlichkeit sein (Abbildung 5 b) und c)).*

Das Bayessche Netz in Abbildung 5 a) kann in äquivalente Faktorgraphen umgewandelt werden. Die A-Priori-Wahrscheinlichkeiten der Knoten X_1 und X_2 können b) direkt in einem Faktor zusammengefasst oder c) in separaten Faktoren modelliert werden.

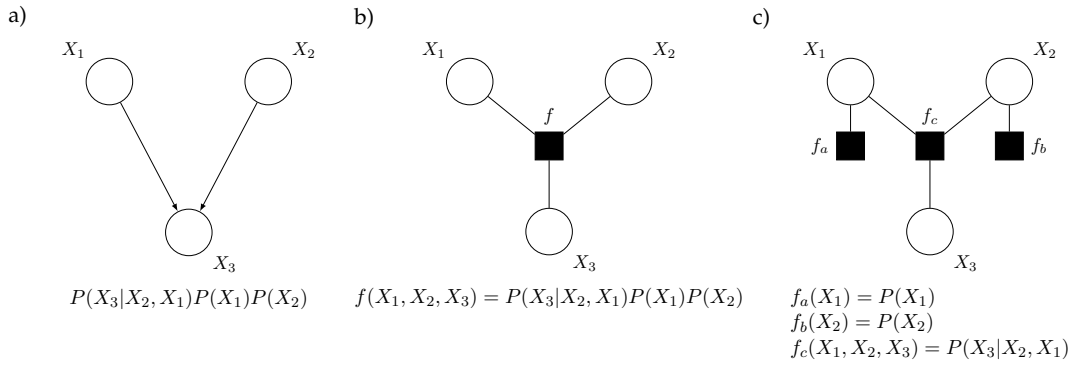


Abbildung 5: Bayessches Netz und äquivalente Faktorgraphen

Jedes Bayessche Netz lässt sich durch einen Faktorgraphen darstellen.[18] Die Vorgehensweise zur Bildung eines solchen äquivalenten Faktorgraphen ist wie folgt:

1. Für jeden Knoten des Bayesschen Netzes wird ein Variablenknoten erzeugt.
2. Für jeden Kindknoten des Bayesschen Netzes wird ein Faktorknoten mit der bedingten Wahrscheinlichkeitsverteilung des Kindknotens erzeugt. Anschließend wird der Faktorknoten mittels ungerichteter Kanten mit den äquivalenten Variablenknoten der Elternknoten des Kindknotens und dem Kindknoten selbst verbunden.
3. Für jeden Wurzelknoten wird ein Faktor mit der A-Priori-Wahrscheinlichkeit des jeweiligen Wurzelknotens erzeugt und verbindet ihn durch eine ungerichtete Kante mit der äquivalenten Variable des Wurzelknotens (Abbildung 5 c)).
   Alternativ: Es wird die A-Priori-Wahrscheinlichkeit jedem Faktorknoten hinzugefügt, der an den zum Wurzelknoten äquivalenten Variablenknoten angrenzt, indem sie mit diesem Faktor multipliziert wird (Abbildung 5 b)).

[17] Wie beim Bayesschen Netz die Begriffe *Knoten* und *Variablen* werden beim Faktorgraphen die Begriffe *Faktor* und *Faktorknoten* synonym verwendet, wobei der erste Begriff eher im statistischen und der zweite eher im graphisch-strukturellen Kontext verwendet wird. Für die Knoten der Variablenmenge wird der Begriff *Variable* oder *Variablenknoten* verwendet.

[18] Der Beweis dafür basiert auf dem sogenannten Hammersley-Clifford-Theorem [AKN12].

3.2.3 Summe-Produkt-Algorithmus auf Faktorgraphen

Der Summe-Produkt-Algorithmus ist ein effizientes Verfahren für exakte Inferenz auf baumartig strukturierten Graphen. Er ist eine Verallgemeinerung des zuvor auf dem kettenartigen Netz gezeigten Message-Passing-Algorithmus.

Gesucht ist die Randverteilung von X:

$$\mathbf{P}(X) = \sum_{\mathbf{X} \setminus X} \mathbf{P}(\mathbf{X}).$$

Wird das Message-Passing-Schema für kettenförmige Graphen verallgemeinert, so berechnet sich die Randwahrscheinlichkeit als das Produkt der Nachrichten, die von allen benachbarten Faktoren $f_i \in \mathcal{N}(X)$ kommen (Abbildung 6):

$$P(X) = \prod_{f_i \in \mathcal{N}(X)} \mu_{f_i \to x}(X).$$

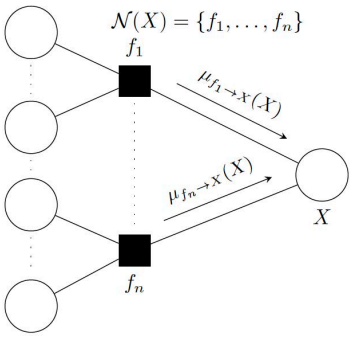


Abbildung 6: Die Randverteilung $P(X)$ eines Knotens X berechnet sich durch Multiplikation der eingehenden Knotennachrichten $\mu_{f_1 \to X}(X), \ldots \mu_{f_1 \to X}(X)$ von allen benachbarten Faktoren $f_1, \ldots, f_n$.

In einem Faktorgraphen werden zwei Typen von Nachrichten unterschieden: die Faktornachricht von einem Faktor f_s zu einem Knoten X und die Knotennachricht von einem Knoten X_j an einen Faktor f_s.

Faktornachricht
Eine Faktornachricht an Knoten X ist das Produkt der in den Faktorknoten f_s eingehenden Nachrichten von allen benachbarten Variablenknoten außer von X, multipliziert mit dem Faktor, welcher über allen möglichen Werten aller im Faktor vorkommenden Variablen außer X aufsummiert wird:

$$\mu_{f_s \to X}(X) = \sum_{X_1} \cdots \sum_{X_m} f_s(X, X_1, \ldots, X_m) \prod_{X_j \in \mathcal{N}(f_s) \setminus X} \mu_{X_j \to f_s}(X_j). \tag{15}$$

Knotennachricht
Eine Knotennachricht an einen Faktor f_s ist das Produkt der in den Knoten X eingehenden

Faktornachrichten von allen anderen Faktoren, außer f_s:

$$\mu_{X \to f_s}(X) = \prod_{f_i \in \mathcal{N}(X) \setminus f_s} \mu_{f_i \to X}(X). \tag{16}$$

Initialisierung
Bevor der eigentliche Message-Passing-Algorithmus startet, müssen die Nachrichten der Blätter des Graphen initialisiert werden. Ein Blatt kann sowohl ein Faktor, als auch eine Variable sein.

Vollständiges Message Passing
Um die Randwahrscheinlichkeiten für alle Knoten des Graphen zu berechnen, müssen Nachrichten über alle Kanten des Graphen jeweils in beide Richtungen geschickt werden. Ähnlich wie beim Message Passing auf einer Knotenkette (Abschnitt 3.2.1) läuft der Algorithmus in zwei Phasen ab. Nach Initialisierung der Blätter wird zunächst ein beliebiger Knoten des Graphen als Wurzelknoten gewählt.

1. In Phase 1 werden, von den Blattknoten ausgehend, die Nachrichten in Richtung des Wurzelknoten geschickt.
2. In Phase 2 werden, vom Wurzelknoten ausgehend, die Nachrichten zurück in die Blattknoten geschickt.

Werden für jeden Variablenknoten die Nachrichten zwischengespeichert, so kann anschließend für jede Variable die Randwahrscheinlichkeit durch Multiplikation dieser Nachrichten bestimmt werden. Für das Berechnen aller Randwahrscheinlichkeiten wird auch hier wieder nur die doppelte Anzahl an Nachrichtenübermittlungen benötigt, wie für das Berechnen der Randwahrscheinlichkeit für eine einzelne Variable.

Berücksichtigung von Evidenz
Sollen die Posteriori-Wahrscheinlichkeiten unter gegebenen Beobachtungen berechnet werden, werden die beobachteten Werte der Evidenzvariablen[19] verwendet, um Faktornachrichten zu berechnen, statt über alle ihre Werte aufzusummieren (analog zu Abschnitt 3.2.1). Gleichung 15 ändert sich unter Beobachtung des Knotens X_i mit dem Wert e_i wie folgt:

$$\mu_{f_s \to X}(X) = \sum_{X_1} \cdots \sum_{X_{i-1}} \sum_{X_{i+1}} \cdots \sum_{X_m} f_s(X, X_1, \ldots, e_i, \ldots, X_m) \prod_{X_j \in \mathcal{N}(f_s) \setminus X} \mu_{X_j \to f_s}(X_j). \tag{17}$$

Die Berechnung der Posteriori-Wahrscheinlichkeiten für die einzelnen Knoten X_i unter gegebener Evidenz $\mathbf{e}$ erfolgt, wie die Berechnung der Randwahrscheinlichkeiten ohne Evidenz, durch Multiplikation der eingehenden Faktornachrichten, wobei mit dem Faktor $\frac{1}{\sum_X P(X,\mathbf{e})}$ normalisiert werden muss.

$$\mathbf{P}(X|\mathbf{e}) = \frac{\prod_i \mu_{f_i \to X}(X)}{\sum_X \prod_i \mu_{f_i \to X}(X)}. \tag{18}$$

[19] Evidenzveriablen sind Variablenknoten des Graphen, für die ein bestimmter Wert beobachtet wird.

3.2.4 Exakte Inferenz auf verbundenen Netzen mit Junction Trees

Ein verbundenes Bayessches Netz liefert einen Faktorgraphen mit Schleifen. Auf einem solchen Graphen ist mit dem Summe-Produkt-Algorithmus keine exakte Inferenz mehr möglich (Abschnitt 3.2.5). Es ist jedoch möglich, das originale Bayessche Netz durch Clustering in eine graphische Struktur zu überführen, auf der Glaubenspropagation mit exakten Ergebnissen möglich ist. Der Transformationsalgorithmus fasst Knoten, die Teil einer Schleife sind, zu Clustern zusammen. Das Ergebnis ist ein baumförmiger Cluster-Graph, der sogenannte Junction Tree, auf welchem anschließend das Message Passing durchgeführt wird.

Der Junction-Tree-Algorithmus liefert für kleinere Netze sehr gute Ergebnisse, hat aber das Problem, dass seine Komplexität mit der maximalen Anzahl von Variablen in einem Cluster exponentiell ansteigt, was ihn für große komplexe Netze ungeeignet macht. Für die in diesem Report behandelte Problemstellung ist er geeignet, was bereits in [FP15] gezeigt wurde. Da dieser jedoch nicht in Infer.NET implementiert ist, wird im praktischen Teil eine Variante der Glaubenspropagation verwendet, die im folgenden Abschnitt beschrieben wird.

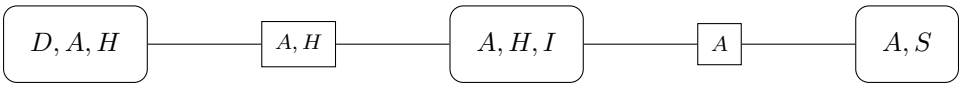


Abbildung 7: Cluster-Graph für das Netz aus Abbildung 1.

3.2.5 Approximative Inferenz mit Loopy Belief Propagation

Für Faktorgraphen mit Schleifen kann der sogenannte Loopy Belief Propagation (LBP) Algorithmus angewandt werden. Es handelt sich um ein iteratives Verfahren, bei dem die Glaubenspropagation bis zum Erreichen eines Konvergenzkriteriums wiederholt wird (Algorithmus 4). Die Ergebnisse sind jetzt jedoch nicht mehr exakt, sondern eine Näherung.

Algorithmus 4 (Loopy Belief Propagation)
Iteratives Verfahren des Loopy Belief Propagation Algorithmus:

1: ***function*** LBP
2: ***for all*** *Kanten zwischen Faktoren f und Variablen X* ***do***
3: *Initialisiere $\mu^{(0)}_{f\to X}$ und $\mu^{(0)}_{X\to f}$ mit Gleichverteilung*
4: ***end for***
5: ***while*** $\mu^{(t+1)} \neq \mu^{(t)}$ ***do*** ▷ *Muss für jede Nachricht μ überprüft werden*
6: ***for all*** *Kanten zwischen Faktoren f und Variablen X* ***do***
7: $\mu^{(t+1)}_{f\to X} \leftarrow$ CalcFactorMessage$(\mu^{(t)}_{X_j\in\mathcal{N}(f)\setminus X\to f})$ ▷ *Gleichung 19*
8: $\mu^{(t+1)}_{X\to f} \leftarrow$ CalcNodeMessage$(\mu^{(t)}_{f_i\in\mathcal{N}(X)\setminus f\to X})$ ▷ *Gleichung 20*
9: $t \leftarrow t + 1$
10: ***end for***
11: ***end while***
12: ***end function***

Das Prinzip eines Iterationsschrittes ist das gleiche wie beim vollständigen Message-Passing-Schema aus Abschnitt 3.2.3: Durch jede Kante des Graphen wird in beide Richtungen je eine Nachricht geschickt. Der Unterschied ist, dass bei Loopy Belief Propagation zur Berechnung einer Nachricht die Nachrichten aus dem vorhergehenden Iterationsschritt verwendet werden, während bei der normalen Glaubenspropagation die Berechnung einer Nachricht auf

Nachrichten desselben Iterationsschrittes basiert (es gibt ja nur den einen). Sei t der vorhergehende Iterationsschritt und $t+1$ der aktuelle Iterationsschritt, so berechnen sich die Nachrichten in Schritt $t+1$ wie folgt:

$$\mu_{f\to X}^{(t+1)}(X) = \sum_{X_1}\cdots\sum_{X_m} f_s(X, X_1, \ldots, X_m) \prod_{X_j\in\mathcal{N}(f)\setminus X} \mu_{X_j\to f}^{(t)}(X_j) \tag{19}$$

$$\mu_{X\to f}^{(t+1)}(X) = \prod_{f_i\in\mathcal{N}(X)\setminus f} \mu_{f_i\to X}^{(t)}(X) \tag{20}$$

wobei $\mu_{f\to X}^{(t+1)}(X)$ die Faktornachrichten und $\mu_{X\to f}^{(t+1)}(X)$ die Knotennachrichten sind.

Damit die Nachrichten für Schritt $t=1$ berechnet werden können, müssen sie für alle Kanten für Schritt $t=0$ initialisiert werden. Da die Nachrichten selbst Wahrscheinlichkeitsverteilungen sind, werden sie typischerweise mit einer Gleichverteilung initialisiert. Danach werden mit jedem Iterationsschritt alle Nachrichten mit den Gleichungen 19 bzw. 20 aktualisiert. Es wird solange iteriert, bis die Nachrichten konvergieren, d.h. für alle Nachrichten μ

$$\mu^{(t+1)} = \mu^{(t)}$$

gilt oder ein anderes Abbruchkriterium (z.B. eine maximale Zahl von Iterationen) erreicht ist. Denn es ist nicht garantiert, dass die Nachrichten konvergieren. In bestimmten Fällen ist es möglich, dass die Nachrichten ab einer gewissen Iteration oszillieren. Es hat sich gezeigt, dass die Reihenfolge, in der die Nachrichten in einem Iterationsschritt neu berechnet werden, die Konvergenz beeinflusst. Ein gleichzeitiges Aktualisieren der Nachrichten führt deutlich häufiger zum Oszillieren als das Nacheinander-Berechnen. Ein geschicktes Planen der Reihenfolge, in der die Nachrichten berechnet werden, verbessert die Ergebnisse auch bezüglich ihrer Genauigkeit. Für dieses Scheduling der Aktualisierungsreihenfolge gibt es verschiedene Verfahren, wovon einige in [SM12] beschrieben werden. Eine Analyse, unter welchen Bedingungen LBP konvergiert, findet sich in [MK07].

Obwohl der Algorithmus keine Konvergenz garantiert, so hat er sich doch in sehr vielen Anwendungsfällen bewährt[20] und liefert auch für die in dieser Arbeit beschriebene Art von Bayesschen Netzen sehr gute Ergebnisse. LBP ist in Infer.NET in Form von Expectation Propagation [Min01] implementiert und wird für den praktischen Teil dieses Reports verwendet.

3.3 Approximative Inferenz durch Sampling

Ein weiteres Prinzip approximativer Inferenz ist das Erzeugen von Stichproben anhand der im Netz modellierten lokalen Wahrscheinlichkeitsverteilungen. Die relative Häufigkeit der erzeugten Werte ergibt eine Annäherung an die Wahrscheinlichkeit. Je größer die Stichprobe, desto genauer ist der berechnete Wert.

- **Direktes Sampling** erzeugt die Stichproben anhand der Wahrscheinlichkeitsverteilungen in topologischer Reihenfolge, also von Eltern- zu Kindknoten. Sei N die Anzahl aller erzeugten Stichproben und $N(\mathbf{x})$ die Anzahl der Stichproben, welche mit dem gesuchten spezifischen Ereignis $\mathbf{x}$ konsistent sind, so ergibt sich die approximierte Wahrschein-

[20] Einer der bekanntesten Anwendungsfälle von Loopy Belief Propagation sind die Turbo-Codes, welche erstmals Informationskodierung nahe der Shannongrenze ermöglichten. Sie waren gleichzeitig der Beweis der praktischen Anwendbarkeit des Verfahrens [BGT93].

lichkeit dieses Ereignisses mit:

$$\hat{P}(\mathbf{x}) = \frac{N(\mathbf{x})}{N} \approx P(\mathbf{x}).$$

Dieser naive Ansatz kann keine Evidenz behandeln.

- **Ablehnungssampling** kann Verteilungsapproximationen der Form $\hat{\mathbf{P}}(x|\mathbf{e})$ liefern, indem es nur die Stichproben verwendet, die auch mit der beobachteten Evidenz **e** konsistent sind. Es werden aber viele Stichproben zurückgewiesen, so dass es ziemlich ineffizient ist. Die Wahrscheinlichkeitsverteilung einer Variablen wird wie folgt approximiert:

$$\hat{\mathbf{P}}(X|\mathbf{e}) = \frac{\mathbf{N}(X, \mathbf{e})}{N(e)} \approx \mathbf{P}(X|\mathbf{e}).$$

- **Wahrscheinlichkeitsgewichtung** ist effizienter als Ablehnungssampling, weil es keine nicht verwendeten Stichproben erzeugt und nur Variablen sampelt, die keine Evidenzvariablen sind. Jedes Sample wird nach der Wahrscheinlichkeit gewichtet, mit dem es der Evidenz entspricht. Höher gewichtete Samples fließen stärker in die Berechnung ein als niedrig gewichtete. Bei steigender Anzahl von Evidenzvariablen verschlechtert sich jedoch die Leistung.

- **MCMC-Algorithmen**[21] erzeugen nicht jede einzelne Stichprobe von Grund auf neu, sondern durch zufällige Änderungen an der vorherigen Stichprobe. Der *aktuelle Zustand* spezifiziert für jede Variable einen Wert, der *nächste Zustand* wird vom MCMC-Algorithmus generiert, indem er den aktuellen Zustand zufällig ändert. Ein viel verwendeter MCMC-Algorithmus ist das **Gibbs Sampling**, welches sich besonders gut für Bayessche Netze eignet. Der Anfangszustand der Nicht-Evidenz-Variablen wird zufällig festgelegt, anschließend werden diese gesampelt. Das Sampling einer Variablen erfolgt auf Basis der bekannten aktuellen Werte seiner Markov-Decken-Variablen und das Ändern einer Variable entspricht einem Schritt im Zustandsraum und somit einer Stichprobe. Gibbs Sampling ist in Infer.NET implementiert und kann als Inferenz-Algorithmus verwendet werden. Ein Nachteil ist, dass es sehr berechnungsintensiv ist, wenn hinreichende Genauigkeit erreicht werden soll. Außerdem ist es schwer, festzustellen, wann der Algorithmus konvergiert – jede zusätzliche Iteration kann das Ergebnis potentiell wieder verschlechtern.

3.4 Approximative Inferenz mit Variational Message Passing

In Modellen mit vielen kontinuierlichen Variablen oder diskreten Variablen mit vielen Zuständen werden die bedingten Wahrscheinlichkeitsverteilungen zwischen den einzelnen Variablen sehr komplex. Die zur Ermittlung der Wahrscheinlichkeiten nötigen hochdimensionalen Summen und Integrale sind analytisch nicht zu bestimmen und eine numerische Lösung oder Abschätzung durch MCMC-Sampling ist sehr berechnungsintensiv. Für solche Problemstellungen wurde die Methode der Variationsinferenz entwickelt. Dabei wird iterativ sowohl eine Approximation der Modellevidenz $P(\mathbf{V})$ als auch die Posteriori-Verteilung $P(\mathbf{H}|\mathbf{V})$ optimiert, wobei $\mathbf{V}$ die beobachteten (*visible*) Variablen des Modells und $\mathbf{H}$ die verborgenen (*hidden*) Variablen sind.

[21] MCMC steht für *Markov Chain Monte Carlo*. Ein solches Verfahren wird auch **Markov-Ketten-Simulation** genannt. Die Bezeichnung „Monte Carlo“ geht auf John von Neumann zurück, der diesen Namen in Anspielung auf die Spielbank im gleichnamigen Stadtteil von Monaco vorschlug (vgl. [And86]).

Die Kernidee ist eine Dekomposition der logarithmierten Modellevidenz:

$$\ln P(\mathbf{V}) = \mathcal{L}(Q) + \mathrm{KL}(Q(\mathbf{H}) \parallel P(\mathbf{H}|\mathbf{V})).$$

$\mathrm{KL}(Q \parallel P)$ ist dabei die Kullback-Leibler-Divergenz, ein Maß für den Unterschied zwischen wahrer Verteilung P und approximierter Verteilung Q. $\mathcal{L}(Q)$ ist eine untere Schranke bezüglich $\ln P(\mathbf{V})$, auch als *Freie Energie* bezeichnet, und ist definiert durch:

$$\mathcal{L}(Q) = \langle \ln P(\mathbf{V}) \rangle_Q - \mathrm{KL}(Q(\mathbf{H}) \parallel P(\mathbf{H}|\mathbf{V})),$$

wobei die Schätzung $\langle \ln P(\mathbf{V}) \rangle_Q$ mittels Q erfolgt. Durch Ermittlung des Qs, welches die Funktion $\mathcal{L}(Q)$ maximiert, wird indirekt die Kullback-Leibler-Divergenz zwischen $Q(\mathbf{H})$ und der exakten Posteriori-Verteilung $P(\mathbf{H}|\mathbf{V})$ minimiert. Das heißt, wenn $Q(\mathbf{H}) = P(\mathbf{H}|\mathbf{V})$ ist, dann ist $\mathcal{L}(Q) = \ln P(\mathbf{V})$. Die Optimierung von $\mathcal{L}(Q)$ erfolgt nun, indem vereinfachende Beschränkungen bezüglich der funktionalen Form von Q vorgenommen werden. Zunächst sollte Q eine faktorisierbare Wahrscheinlichkeitsverteilung mit

$$Q(\mathbf{H}) = \prod_i Q_i(H_i)$$

sein, wobei die einzelnen Q_is eine Posterior-Verteilung von unabhängigen Gruppen von Variablen $\mathbf{H}_i$ approximieren, während $\mathcal{L}(Q)$ die Log-Evidenz $\ln P(\mathbf{V})$ approximiert. Des Weiteren sind die einzelnen Q_is aus einer bestimmten Verteilungsfamilie, einer sogenannten Exponentialfamilie, zu wählen. Mitglieder einer Exponentialfamilie lassen sich mittels eines natürlichen Parametervektors ϕ, eines suffizienten Statistikvektors $\mathbf{u}$[22] und eines Normierungsfaktors g bilden:

$$P(\mathbf{X}|\mathbf{Y}) = \exp\left[\phi(\mathbf{Y})^{\mathrm{T}}\mathbf{u}(\mathbf{X}) + f(\mathbf{X}) + g(\mathbf{Y})\right].$$

Innerhalb einer Exponentialfamilie haben die Parameter- und Statistik-Vektoren jeweils die gleiche Form. Eine effiziente Optimierung der einzelnen Faktoren erfolgt durch Austausch der Schätzungen dieser Parameter- und Statistikvektoren zwischen den Knoten in Form eines Message-Passing-Schemas.

Der Vorteil von VMP besteht darin, dass sich auch sehr komplexe Modelle aus diskreten und kontinuierlichen Verteilungen effizient approximieren lassen. Der Nachteil besteht darin, dass die Approximation nicht beliebig genau werden kann. Bei beobachteten diskreten Variablen mit mehr als zwei Zuständen entspricht die Approximation aufgrund der funktionalen Beschränkung der Q_is nicht der exakten Verteilung. Dies kann bei kleinen, rein diskreten Modellen dazu führen, dass sich dieser Fehler so weit fortpflanzt, dass der Algorithmus schlechtere Ergebnisse liefert als andere Inferenzalgorithmen, die bei geringer Modellgröße und hauptsächlich binären Variablen berechnungstechnisch noch gut handhabbar sind und genauer approximieren können. Wie sich das in der Praxis auswirkt, wird in Abschnitt 5.3.2 beschrieben.

4 Prototypische Softwarearchitektur

In diesem Kapitel sollen nun die bisher untersuchten Ansätze für die Entwicklung eines praxistauglichen Agenten vorgestellt und bewertet werden. Als Erstes wird das dafür entwickelte Bayessche Netz zur Diagnose von Kopfschmerzen vorgestellt. Danach wird der implementierte Agent und die Vorgehensweise beim Übersetzen des externen Formates in die Infer.NET-Form

[22] Eine suffiziente Statistik ist eine Stichprobe, aus der sich die zugrundeliegende Verteilung vollständig ableiten lässt.

und die Inferenz mit Infer.NET erläutert.

4.1 Bayessches Netz zur Kopfschmerzdiagnostik

Um die Inferenzverfahren testen zu können, wurde zunächst beispielhaft ein Bayessches Netz zur Schmerzdiagnostik modelliert. Dabei wurde ausgehend von den Diagnosen modelliert und die Diagnosehierarchie wurde in der Netzstruktur so abgebildet, dass speziellere Diagnosen als Elternknoten von allgemeineren Diagnosen modelliert wurden. Der Prototyp beschränkt sich auf ein Beispielnetz zur Diagnose eines Hirntumors sowie drei verschiedene Kopfschmerzformen (Migräne, Spannungskopfschmerz und Clusterkopfschmerz) mit den entsprechenden Symptomen (Abbildung 8). Der Migräne wurden noch zwei speziellere Diagnosen vorangestellt: Migräne mit Aura und Migräne ohne Aura.

Abbildung 8: Bayessches Netz für Kopfschmerzdiagnostik

4.2 Repräsentationsformen der Wissensbasis

Damit mittels eines Inferenzpaketes Abfragen an ein Netz gerichtet werden können, muss die Übergabe der Abfrage in einer geeigneten Repräsentationsform erfolgen. Mit den genannten

Bausteinen lässt sich jedes beliebige diskrete Bayessche Netz konstruieren. Die Repräsentation eines Bayesschen Netzes in Quellcodeform ist jedoch für den Aufbau einer Wissensbasis durch einen Domänexperten eher ungeeignet. Sie ist nicht portabel, unübersichtlich und schwer durch externe Werkzeuge zu bearbeiten. Gewünscht ist ein Format, welches übersichtlich strukturiert, leicht zu parsen und auch von Fachexperten in einem Editor leicht zu bearbeiten ist. Zusätzlich sollte es eine kompakte Darstellungsform für komplexere Knotenfunktionen wie die in Kapitel 2.4.3 vorgestellten kanonischen probabilistischen Modelle bieten. Es gibt bereits eine Reihe von Textformaten für Bayessche Netze, so dass die Eigenentwicklung eines Formates hier nicht zwingend notwendig ist. PropmodelXML wurde bspw. für die Darstellung verschiedener Probabilistischer Graphischer Modelle entwickelt und eignet sich zum Einsatz im Diskursbereich. Eine genaue Beschreibung des Formates findet sich in [DD06]. Wichtigstes Kriterium ist die vollständige Unterstützung für ICI-Modelle, die von keinem anderen Format geboten wurde. Darüber hinaus handelt es sich um ein XML-Format, das flexibel und erweiterbar ist.

4.3 Implementierung eines Prototyps mithilfe von Infer.NET

Um die vorgestellten Methoden auf ihre Praxistauglichkeit zu überprüfen, wurde ein prototypischer Agent in C# implementiert. Das Programm ist in der Lage, ein Netz im Pgmx-Format zu parsen und in eine Infer.NET-Modelldefinition zu überführen (Abschnitt 4.3.1). Wie auf dem eingelesenen Netz mithilfe von Infer.NET Abfragen durchgeführt werden, wird in Abschnitt 4.3.2 gezeigt. Außerdem ist es möglich, ein Netz tiefergehend zu analysieren, indem Beispiel-Datensätze für dieses Netz erzeugt und mit demselben Netz verarbeitet werden können.

4.3.1 Überführung der externen Wissensrepräsentation

Die Überführung der externen Netzrepräsentation in ein Modell, auf dem das Infer.NET-Paket Inferenzen durchführen kann, erfolgt schrittweise[23]:

1. Parsen
Zunächst werden die Elemente eines Bayesschen Netzes aus der XML-Datei geparst und in äquivalente Datenstrukturen in C# überführt und zu einer Netzstruktur verknüpft, in der jeder Kindknoten seine Elternknoten kennt. Der wichtigste Baustein ist hier `BayesianNode`, der einen Netzknoten repräsentiert.

2. Konstruktion von bedingten Wahrscheinlichkeitsbäumen
Soll nun mithilfe dieser Struktur aus den Wahrscheinlichkeiten eines Knotens (im Pgmx-Format durch sogenannte *Potentials* repräsentiert) Infer.NET-kompatibler C#-Code erzeugt werden, so zeigt sich, dass sich die verschachtelten Fallunterscheidungen einer bedingten Wahrscheinlichkeitsverteilung am besten rekursiv aus einem Baum erzeugen lassen. Ein Baum ist gleichzeitig eine sehr effiziente Darstellungsform für BWTs. Abbildung 9 zeigt die BWT aus Tabelle 4 in Baumform. Jede Verzweigungstiefe unterhalb der Baumwurzel repräsentiert eine Variablen-Spalte der BWT und somit jede Knotenebene (außer der Wurzel) eine Variable in der bedingten Wahrscheinlichkeitsverteilung. Die letzte Verzweigung repräsentiert die bedingte Vari-

[23] Bei dem von Microsoft Research entwickelten Infer.NET handelt es sich um eine Bibliothek zur Inferenz Graphischer Probabilistischer Modelle wie bspw. Bayessche Netze oder Markov Random Fields. Expectation Propagation, Variational Message Pasing und Gibbs Sampling sind implementierte Inferenzalgorithmen.

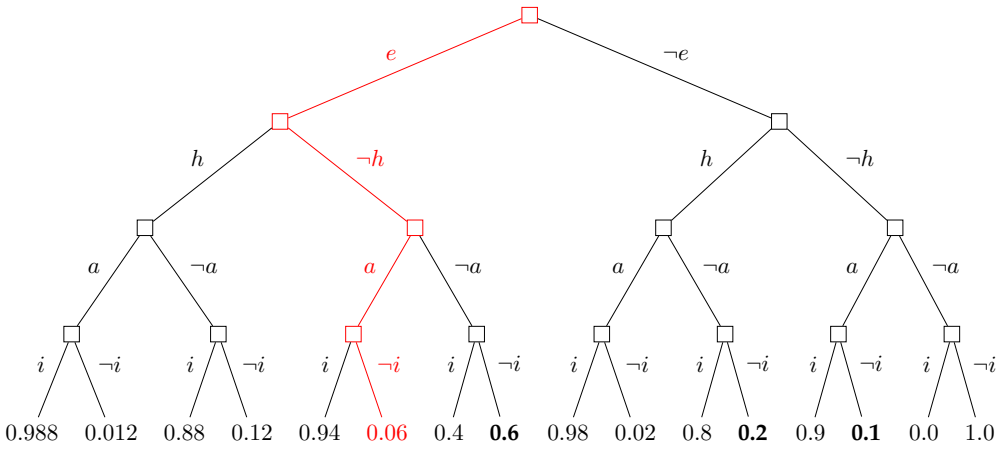


Abbildung 9: Bedingte Wahrscheinlichkeitstabelle in Baumstruktur für das Noisy-Or-Beispiel „Herzinfarkt (i) mit möglichen Ursachen Endokorditis (e), Hypertonie (h) oder Arteriosklerose (a)“. Der rot markierte Pfad durch den Baum liefert die Wahrscheinlichkeit $P(\neg i | e \wedge \neg h \wedge a) = 0.06$. Die fett markierten Wahrscheinlichkeiten sind die Inhibitor-Parameter des Noisy-Or.

able (also den Kindknoten). Die Kanten des Baumes sind mit den jeweiligen Wertzuweisungen der darunterliegenden Variable beschriftet, so dass jeder Pfad durch den Baum – von der Wurzel zu einem Blatt – eine eindeutige Kombination von Wertzuweisungen ergibt und somit genau eine Zeile in der BWT repräsentiert. Es werden die Blätter mit den zugehörigen bedingten Wahrscheinlichkeiten beschriftet und es wird so eine Datenstruktur erhalten, in der sehr einfach navigiert werden kann.

Es werden die Wahrscheinlichkeiten aus der externen Repräsentation in solche bedingte Wahrscheinlichkeitsbäume überführt und ihren Netzknoten zugeordnet, wobei bei ICI-Modellen (Noisy-OR o.ä.) der größte Teil der bedingten Wahrscheinlichkeiten in den Blättern anhand der zugehörigen Formeln aus Abschnitt 2.4.3 berechnet wird. Damit jeder Baumknoten die Variable kennt, zu deren Spalte er gehört, bekommt er bei der Erzeugung des Baumes eine Referenz auf den zugehörigen Netzknoten.

3. Erzeugung des Infer.NET-Model
Zum Schluss werden für alle Netzknoten korrespondierende Infer.NET-Variablen erstellt, wobei jeder Netzknoten eine Referenz auf seine zugehörige Infer.NET-Variable erhält. Nun können die Infer.NET-Variablen mithilfe der bedingten Wahrscheinlichkeitsbäume verknüpft werden. Dabei wird rekursiv in den BWT-Baum abgestiegen und innerhalb eines `using`-Statements für die dem aktuellen Baumlevel zugehörige Infer.NET-Variable eine Fallunterscheidung erzeugt. Ist das Blattlevel erreicht, wird der zugehörigen Infer.NET-Variable die in den Blättern definierte Wahrscheinlichkeitsverteilung zugewiesen.

4.3.2 Inferenz mit dem Agenten

Im Folgenden wird erläutert, wie der implementierte Prototyp Infer.NET nutzt, um Abfragen auf der konstruierten Modelldefinition durchzuführen. Dazu wird die Arbeitsweise der Inferenz-Engine von Infer.NET vorgestellt.

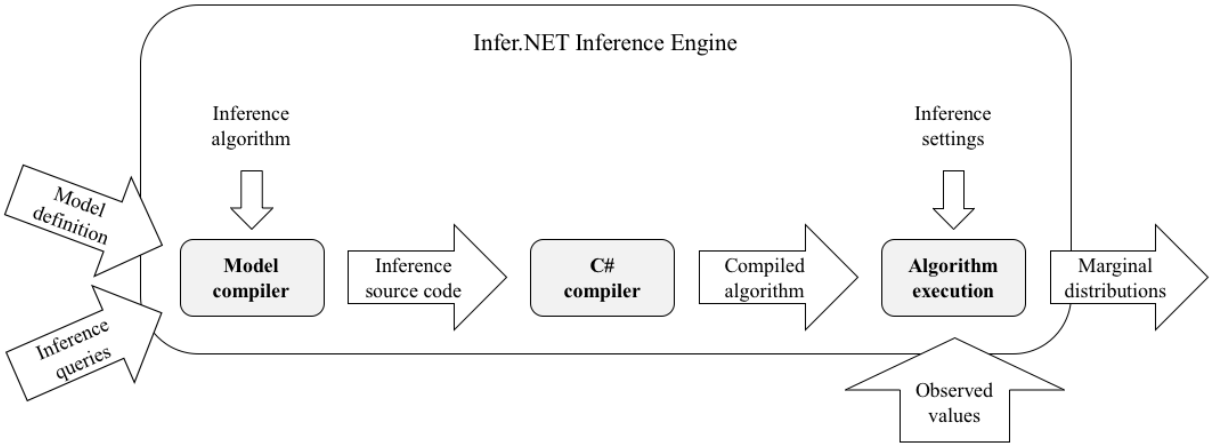


Abbildung 10: Arbeitsweise von Infer.NET [MWG+18]

1. Zunächst wird eine Modell-Definition erstellt, welche mithilfe der Modeling-API beschrieben wird. Dies übernimmt der im Abschnitt 4.3.1 vorgestellte Transformationsalgorithmus. Zudem wird der Inferenzalgorithmus festgelegt und es werden Inferenzabfragen an das Modell spezifiziert, indem die beobachteten Variablen und die Abfragevariablen festgelegt werden.
2. Das Modell wird an den Model-Compiler übergeben, welcher optimierten C#-Quellcode generiert, der es ermöglicht, auf Basis des spezifizierten Inferenzalgorithmus Abfragen auf dem Modell durchzuführen. Der generierte Quellcode wird in eine Datei geschrieben und kann später auch direkt verwendet werden.
3. Der Quellcode wird mithilfe des C#-Compilers kompiliert. Es ist möglich, diesen Vorgang manuell auszuführen, um genauer zu kontrollieren, wie die Inferenz durchgeführt wird. Am einfachsten ist das automatische Ausführen mit der `Infer`-Methode.
4. Mit einem Satz von konkreten Werten für die beobachteten Variablen führt die Inferenz-Engine den kompilierten Algorithmus aus. Dies kann für unterschiedliche Werte der beobachteten Variablen wiederholt werden, ohne den Algorithmus neu kompilieren zu müssen. Welche Variablen beobachtet werden, ist allerdings für ein Kompilat festgelegt.

Die Schritte 2 bis 4 werden durch Aufruf der `Infer`-Methode initiiert. Infer.NET entscheidet bei einem erneuten Aufruf der `Infer`-Methode, ob der Algorithmus neu kompiliert werden muss.

5 Evaluation

5.1 Bewertung der Netzqualität mittels realer Daten

Um zu testen, wie gut das Netz die modellierten Diagnosen klassifiziert, wurden in der vorrangegangenen Arbeit [FP15] zunächst einige Tests auf Basis realer Anamnesedaten durchgeführt. Das Verfahren wird hier noch einmal vorgestellt, um es mit dem verbesserten Verfahren in Abschnitt 5.2 vergleichen zu können. Es wurden einige Datensätze, bestehend aus Diagnosen und Anamnesen, ausgewählt und daraus manuell Abfragen erzeugt. Mit dem implementierten Netz wurden aus diesen Abfragen Diagnosewahrscheinlichkeiten berechnet und

Merkmale	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
Anorexia	True	False		False				boolean
AuraSymptoms	False		False	True	True			False
Headache	True	True	True	True	True	True	True	True
IpsilateralConjunctivalInjection	False	False	False	False			True	
IpsilateralLacrimination	True		False	False			True	
IpsilateralPtosis	False	False		False				
Nausea	True	False	True	False				True
PainDuration			Days	Days	Years	Years	Hours	Hours
PainIntensityAvrg	3	2	8	3	7	7	7	5
PainLocationUnilateral	False		True	False			True	
PainQuality	Pulsating	Stabbing	Stabbing		Other	Stabbing	Other	Stabbing
Phonophobia	True	True	True	True				
Photophobia	True	True		False				
Restlessness	False	False	True					
Sex	Female	Male	Female	Female	Male	Female	Male	Male
TriggerAlcohol	True			True	True			False
TriggerAnxietyDepression			True		False			False
TriggerBodyStress	True	True	False		True	True		False
TriggerMenstruation			True	True				
TriggerMentalStress	True		True	True	True	True		False
TriggerSmoking								False
TriggerUpset	True			True		True		False
TriggerWeather	True	True		True		True		False
Vomiting	False	False	True	False				

Tabelle 6: Testabfragen für acht Patienten

diese mit den ärztlichen Diagnosen aus den Datensätzen verglichen. Erwartet wurden hohe Wahrscheinlichkeiten für die gestellten Diagnosen und signifikant niedrigere für die nicht gestellten.

Eine Abfrage besteht aus einer Menge von Evidenzvariablen und den zugehörigen beobachteten Werten. Die Anamnesedaten jedes Datensatzes wurden manuell ausgewertet. War ein im Netz modelliertes Merkmal in der Anamnese hinreichend beschrieben, so wurde der Abfrage eine Evidenzvariable hinzugefügt und ihr ein Wert zugewiesen, welcher zur Anamnesebeschreibung passte (Tabelle 6).

Für jede Abfrage wurden mit dem Netz die bedingten Wahrscheinlichkeitsverteilungen sämtlicher Knoten berechnet. Von Interesse sind die Wahrscheinlichkeiten der Diagnosen, welche in Tabelle 7 aufgeführt sind.

Patienten	Diagnosen	CHA	Migräne	MA	MO	TTHA
Patient 1	MO	0,003	0,9997	0,0172	0,09847	0,9907
Patient 2	MO	0,0086	0,9914	0,5904	0,4073	0,3892
Patient 3	MO	0,0011	1	0,018	0,992	0,0342
Patient 4	MA	0,0003	0,9997	0,9934	0,02	0,9289
Patient 5	TTHA	0,354	0,9768	0,9679	0,817	0,5799
Patient 6	TTHA	0,0006	0,1953	0,068	0,1376	0,9948
Patient 7	CHA	0,9979	0,0399	0,242	0,0161	0,008
Patient 8	CHA	0,1022	0,8951	0,0936	0,8022	0,0048

Tabelle 7: Testergebnisse: Diagnosen und erwartete Diagnosen; CHA: Cluster Headache; MA: Migräne mit Aura; MO: Migräne ohne Aura; TTHA: Tensiontype Headache

Die Ergebnisse der Tests zeigen, dass die vom Arzt gestellte Diagnose vom Netz in sechs von acht Fällen als sehr wahrscheinlich vorhergesagt wird, obwohl nur ein kleiner Teil der für die Domäne relevanten Variablen im Netz modelliert wurde. Auch ist die Diagnosewahrscheinlichkeit für die anderen Diagnosen meist sehr gering. Die „falschen“ Klassifikationen sind außerdem alle erklärbar:

- Patient 1 und 5: Die Queries enthalten viele Symptome, die auf beide Diagnosen hindeuten. Diverse Merkmale in den Anamnesedaten, die den Spannungskopfschmerz von der Migräne unterscheidbar machen, wie Schmerzlokalisation und Schmerzempfinden, aber auch der zeitliche Verlauf, sind im Netz noch nicht detailliert genug modelliert. Zudem treten Symptome von Migräne und Spannungskopfschmerz häufig parallel auf. Dies wird als Kombinationskopfschmerz beschrieben. Patient 5 hat zudem Symptome angegeben, die auf eine Aura hindeuten, weswegen die Migräne stärker gewichtet wird als der Spannungskopfschmerz. Er wird in der Datenbank auch als Migräniker geführt.
- Patient 2: Hier war es beim Auswerten der Anamnese schwierig, anhand der Daten zu entscheiden, ob Aurasymptome vorhanden sind oder nicht. Das Netz klassifiziert daher korrekt eine Migräne, kann aber nicht genau entscheiden, um welche speziellere Migräne es sich handelt.
- Patient 8: Es ist auch für einen Experten nicht leicht, Clusterkopfschmerz korrekt von Migräne zu unterscheiden. In der Literatur wird deswegen meist explizit auf eine Reihe von Kriterien zur Differentialdiagnose hingewiesen. Hier wurde keines der für den Clusterkopfschmerz typischen ipsilateralen Symptome wie Bindehautentzündung, Tränenfluss und Ptosis bei der Anamnese angegeben. Trotzdem klassifiziert das Netz den Clusterkopfschmerz mit einer fast dreimal höheren Wahrscheinlichkeit, als in den anderen Fällen, wo kein Clusterkopfschmerz vorlag.

Beim Auswerten der Anamnesedaten war noch das allgemeine Phänomen zu beobachten, dass viele Patienten schon mit einer festen Diagnosevorstellung zum Arzt kommen. So hat Patient 1 in der Anamnese immer wieder seine Migräne erwähnt, so dass eine gewisse Voreingenommenheit des Arztes bei der Diagnose nicht ausgeschlossen werden kann und vielleicht deshalb ein eventuell vorhandener Spannungskopfschmerz nicht diagnostiziert wird.

Die Tests liefern interessante Ergebnisse, jedoch ist die Stichprobengröße zur Netzbewertung mit acht Patienten sehr klein. Darüber hinaus ist auch die Qualität der zur Verfügung stehenden Patientendaten unzureichend. Zum Einen sind die Anamnesemerkmale stark überspezifiziert, zum Anderen nur unzureichend ausgefüllt. Außerdem gab es für bestimmte Diagnosen sehr wenige oder keine Beispielpatienten.

5.2 Bewertung der Netzqualität mittels generierter Beispieldaten

Um die Klassifizierungsqualität des Prototypen zu testen (Abschnitt 4.3) und auch mit dem Prototypen der vorangegangenen Arbeit [FP15] vergleichen zu können, wurde ein neuer Ansatz verfolgt. Mithilfe des Netzes selbst wurden A-Priori-Verteilungen für die bei einer Anamnese beobachteten Merkmale ermittelt und auf Basis dieser Verteilungen Beispieldatensätze gesampelt. Diese Form der Qualitätsbestimmung eines Netzes wurde bereits in der Fehlerdiagnose von Kraftfahrzeugen erfolgreich eingesetzt [PDT03].

Zunächst werden, wie im Folgenden beschrieben, für jede Diagnose passende Beispiele generiert.

1. Es wird eine Abfrage mit folgenden Eigenschaften erzeugt:
   - Die gewünschte Diagnose ist eine beobachtete Variable mit dem Wert *true*.
   - Alle anderen Diagnosen sind beobachtete Variablen mit dem Wert *false*.
   - Alle bei der Diagnostik beobachteten Merkmale (z.B. Symptome) sind Abfragevariablen.
2. Für alle zu beobachtenden Merkmale werden die bedingten Wahrscheinlichkeiten unter der gegebenen Diagnose durch Inferenz auf der eben erzeugten Abfrage ermittelt.
3. Es wird ein Beispiel generiert, indem für jedes zu beobachtende Merkmal zufällig der Wert unter der eben ermittelten Wahrscheinlichkeitsverteilung erzeugt wird. Dieser Schritt wird wiederholt, bis die gewünschte Anzahl an Beispielen generiert wurde.
4. Es werden Abfragen mit den Beispielwerten als beobachtete Variablen und sämtlichen Diagnosen als Abfragevariablen konstruiert und die Diagnosewahrscheinlichkeiten durch Inferenz ermittelt.
5. Es wird die für die Erzeugung des Beispiels genutzte, erwartete Diagnose mit der Diagnose verglichen, für die im vorangegangenen Schritt die höchste Wahrscheinlichkeit ermittelt wurde.

Das Sampling wurde für beide Prototypen implementiert, da das alte Netz teilweise andere Merkmalsverteilungen liefert als das neue überarbeitete Netz. Somit konnte auch geprüft werden, wie robust der neue Prototyp und sein in der Merkmalsaufteilung spezifischeres Bayessche Netz mit Beispielen umgeht, die unter einer weniger spezifischen Merkmalsaufteilung durch den alten Prototyp erzeugt wurden.

Mit jedem Algorithmus (Junction Tree Belief Propagation (JTBP) auf Basis des alten Netzes, Expectation Propagation (EP) und Variational Message Passing (VMP) auf Basis des neuen Netzes) wurden pro Diagnose 200 Beispiele erzeugt und in einer Datenbank zwischengespeichert. Anschließend wurde mit jedem dieser Algorithmen für alle Beispiele die Diagnosewahrscheinlichkeiten berechnet und zur Auswertung in die Datenbank geschrieben.

Um die Qualität der Inferenz zu beurteilen, wurden für jede Diagnose und jeden Algorithmus der Anteil der korrekt klassifizierten Beispiele ermittelt (Tabelle 8).

Diagnose (je 200 Beispiele)	Sampling-Algorithmus	Inferenzalgorithmen JTBP	EP	VMP
BrainTumor	JTBP	100	99	99
ClusterHeadache	JTBP	99	98	99
MigraineWithAura	JTBP	97	97	33
MigraineWithoutAura	JTBP	100	100	96
TensionHeadache	JTBP	97	98	96
BrainTumor	EP	100	100	100
ClusterHeadache	EP	99	100	100
MigraineWithAura	EP	93	93	38
MigraineWithoutAura	EP	99	99	94
TensionHeadache	EP	98	99	99
BrainTumor	VMP	100	100	100
ClusterHeadache	VMP	98	98	100
MigraineWithAura	VMP	97	97	39
MigraineWithoutAura	VMP	100	100	91
TensionHeadache	VMP	95	96	95

Tabelle 8: Anteil korrekt klassifizierter Beispiele in Prozent

Es wurden in drei Versuchen mit Join Tree Belief Propagation (JTBP), Expectation Propagation (EP) und Variational Message Passing (VMP) 200 Beispiele pro Diagnose aus den Kopfschmerz-Netzen ([FP15] und Abbildung 8) erzeugt und jeweils mit allen drei Algorithmen klassifiziert. In Tabelle 8 ist der Anteil der korrekt klassifizierten Beispiele dargestellt. Abbildungen 11 bis 13 stellen dies grafisch dar.

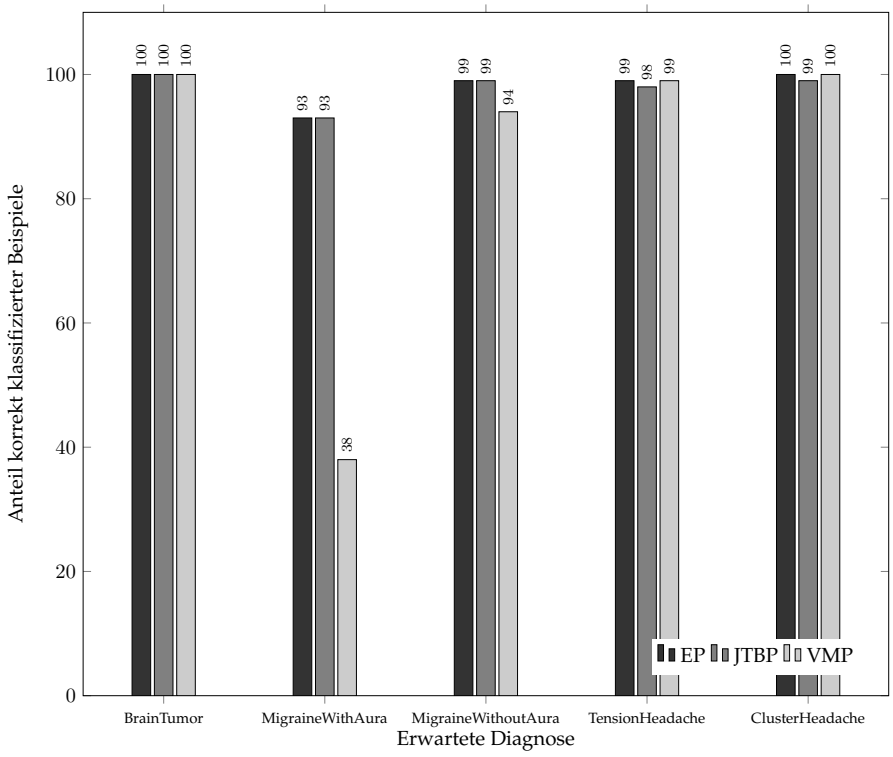


Abbildung 11: Anteile korrekt klassifizierter Beispiele durch Inferenz mit den drei Algorithmen für die mit EP erzeugten Beispiele (Tabelle 8).

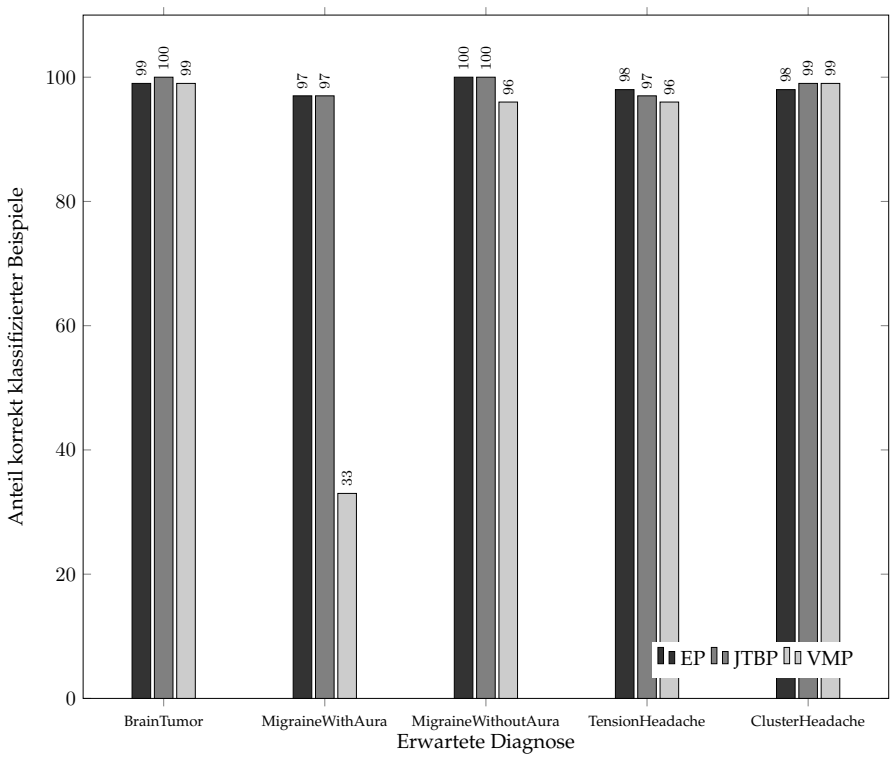


Abbildung 12: Anteile korrekt klassifizierter Beispiele durch Inferenz mit den drei Algorithmen für die mit JTBP erzeugten Beispiele (Tabelle 8).

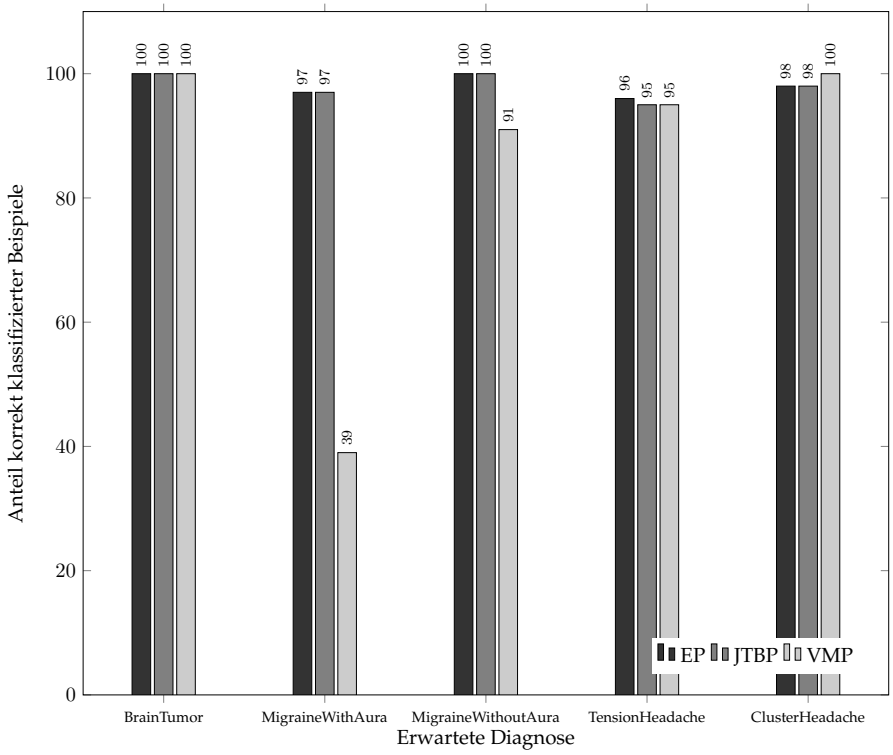


Abbildung 13: Anteile korrekt klassifizierter Beispiele durch Inferenz mit den drei Algorithmen für die mit VMP erzeugten Beispiele (Tabelle 8).

Eine vollständige Darstellung der für die Beispiele ermittelten Diagnosewahrscheinlichkeiten zeigen die Abbildungen 14 (erzeugt mit EP, Inferenz durch EP), 15 (erzeugt mit EP, Inferenz mit JTBP) und 16 (erzeugt mit EP, Inferenz mit VMP).

Die Wahrscheinlichkeit für das Auftreten der erwarteten Diagnose gibt der Marker im positiven Bereich der y-Achse von 0 bis 1 an. Für die nicht erwarteten Diagnosen wird die Wahrscheinlichkeit für das Nicht-Eintreten dieser durch die Marker im negativen Bereich der y-Achse von -1 bis 0 angegeben. Welche Diagnose erwartet wurde, zeigt die Markierung an der X-Achse. Die negativen Werte wurden um bis zu 0,04 verschoben, um die Marker für jede Diagnose sichtbar zu machen.

Außerdem wurden die Falschklassifikationen einer näheren Untersuchung unterzogen. Es wurde ermittelt, wie stark sich die Wahrscheinlichkeit der erwarteten Diagnose von der Wahrscheinlichkeit der klassifizierten Diagnose unterscheidet. Dazu wurden die Diagnosewahrscheinlichkeiten für falsch klassifizierte Beispiele, die mit Expectation Propagation erzeugt wurden, berechnet. Die Wahrscheinlichkeitsverteilungen der Falschklassifikationen für die Beispiele, die mit den anderen beiden Algorithmen erzeugt wurden, zeigen keine statistisch signifikanten Unterschiede.

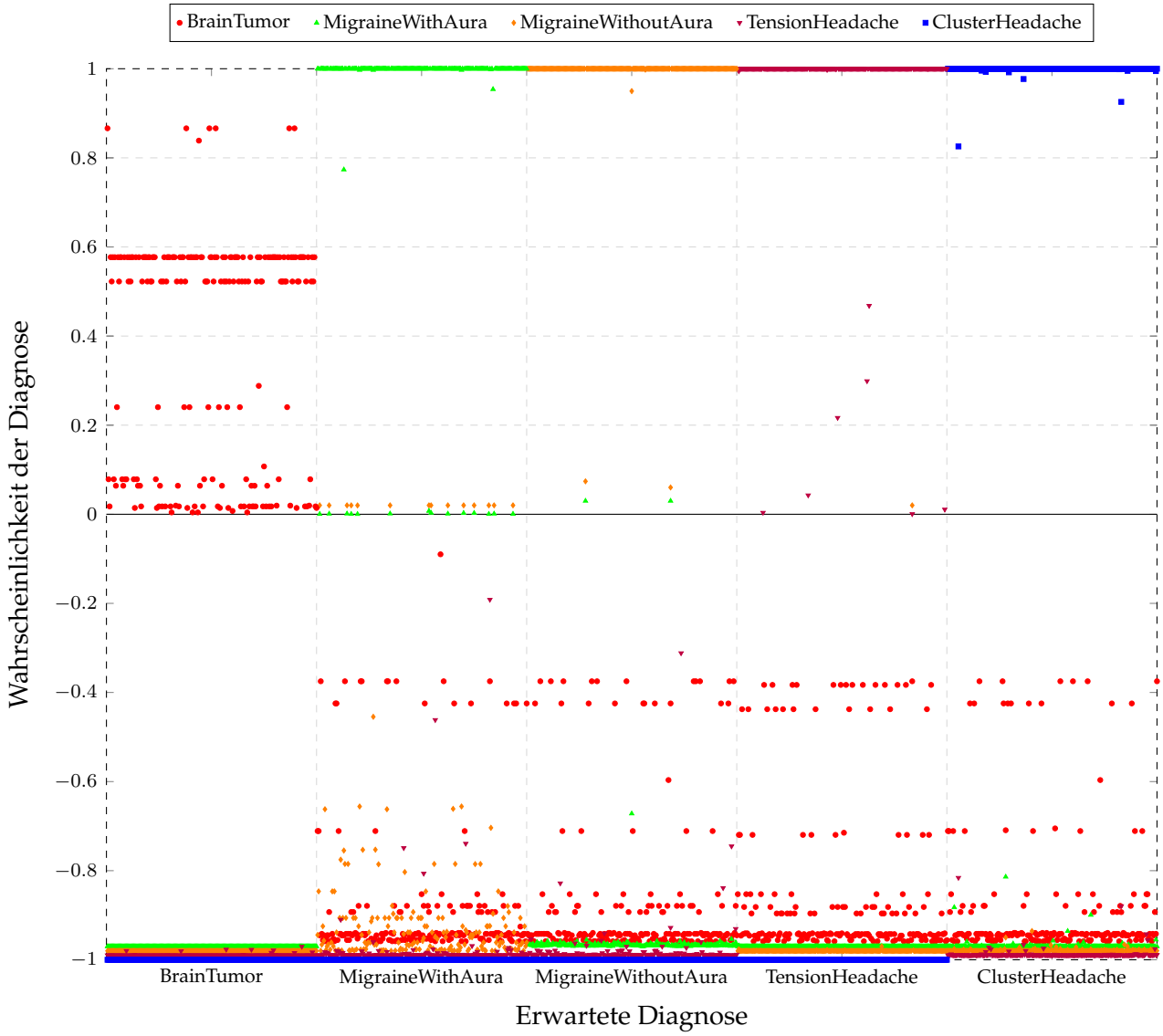


Abbildung 14: Ergebnisse der Inferenz mit Expectation Propagation für die mit EP erzeugten Beispiele.

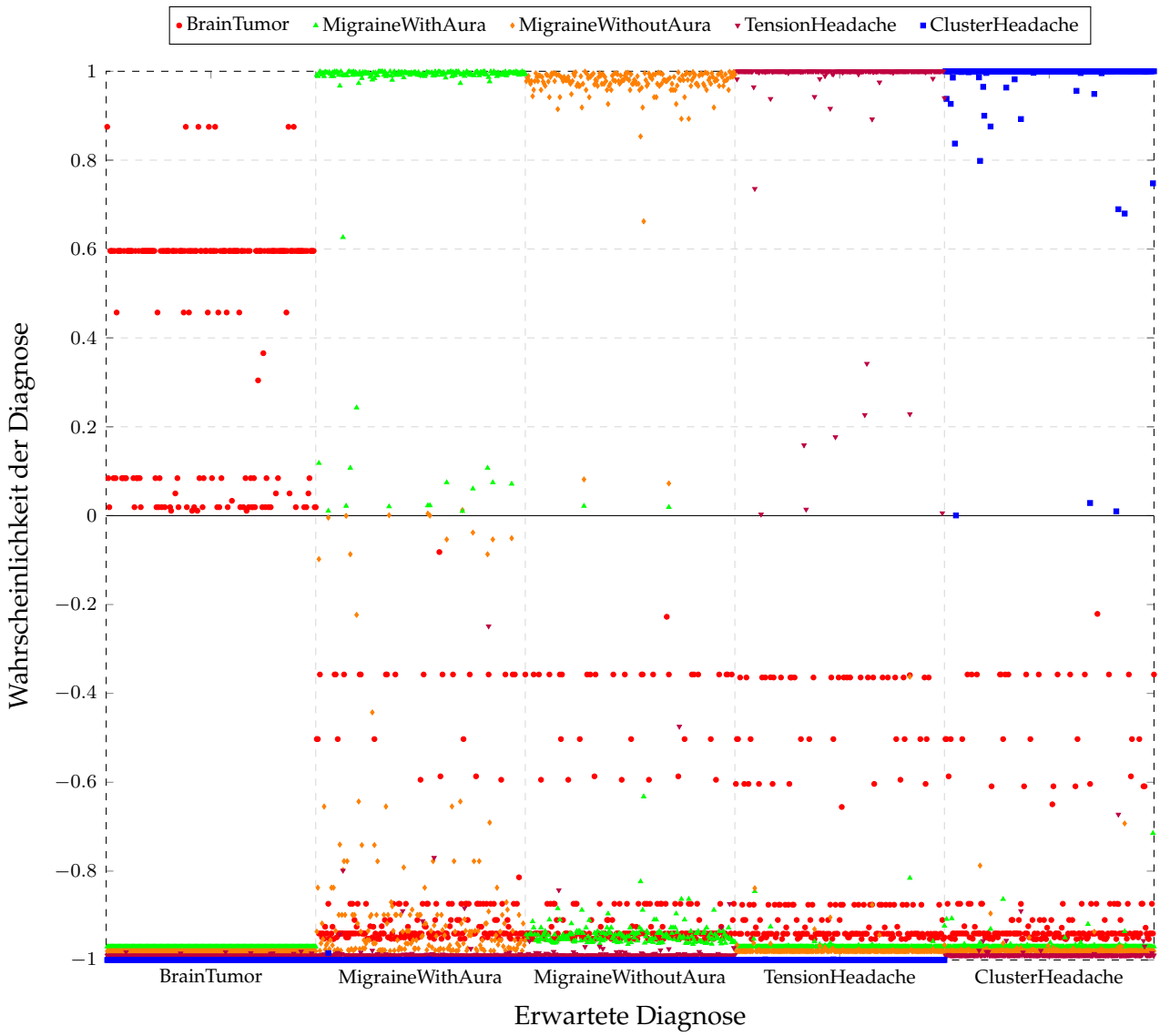


Abbildung 15: Ergebnisse der Inferenz mit Join Tree Belief Propagation für die mit EP erzeugten Beispiele.

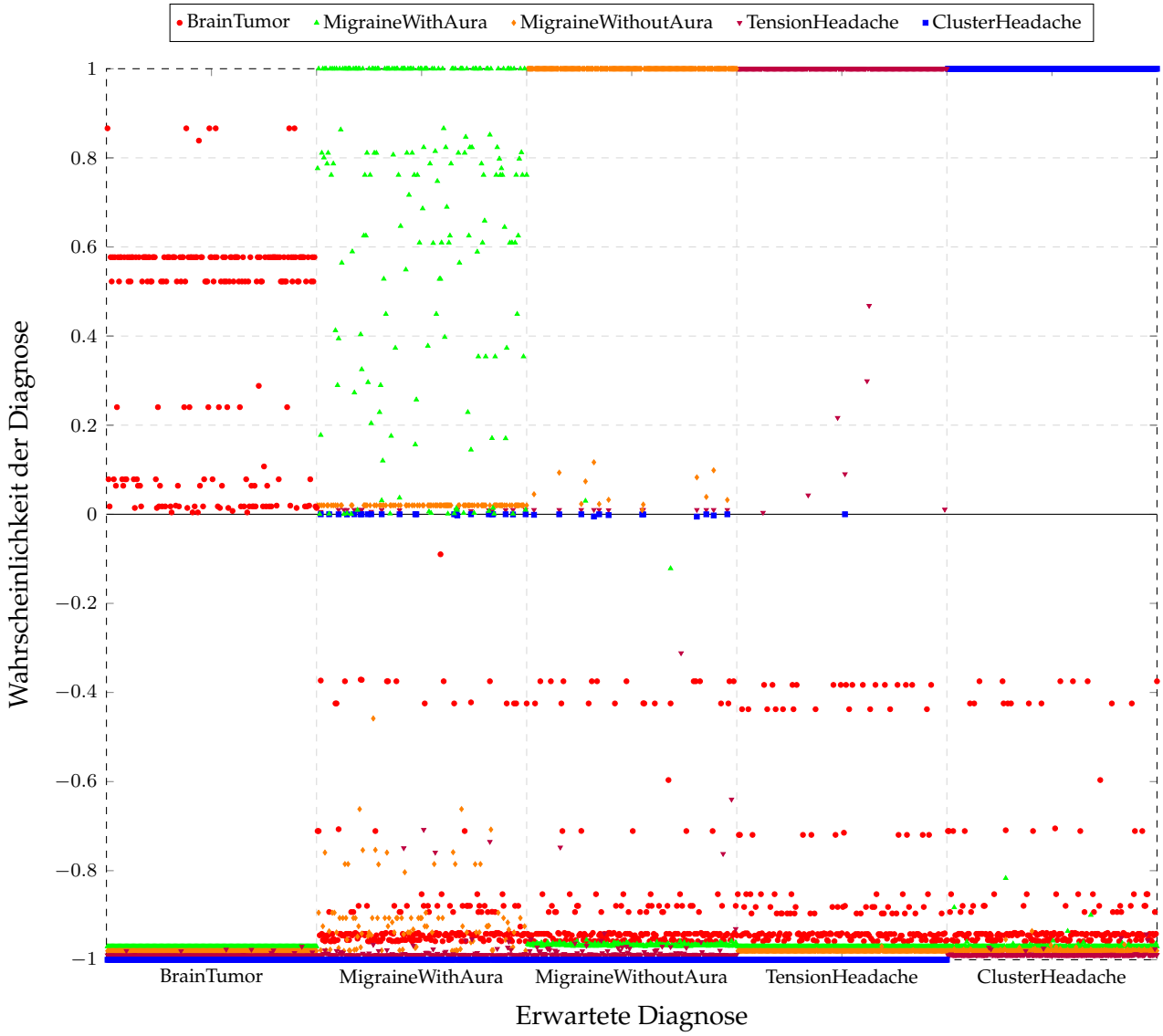


Abbildung 16: Ergebnisse der Inferenz mit Variational Message Passing für die mit EP erzeugten Beispiele.

5.3 Bewertung der Ergebnisse

5.3.1 Ergebnisse der Inferenz mit Junction Tree Belief Propagation und Expectation Propagation

Tabelle 8 und Abbildung 14 bis 16 zeigen, dass die Inferenz mit JTBP mit dem Netz aus [FP15] und EP mit dem Netz von Abbildung 8 sehr gute Ergebnisse für alle Beispieldatensätze liefert, unabhängig von Algorithmus und Netz, mit denen sie erzeugt wurden. Zudem unterscheiden sich die Ergebnisse von JTBP und EP kaum voneinander. Abbildungen 14 bis 16 zeigen weiterhin, dass lediglich die Streuung der Wahrscheinlichkeiten bei JTBP etwas höher ist, als bei EP. Dies kann teilweise auch durch das andere Netz bedingt sein, welches mit diesem Algorithmus verwendet wurde, ändert den Anteil der korrekt klassifizierten Beispiele aber nicht (Abbildung 14). Die wenigen falsch klassifizierten Beispiele sind meist auf den Zufall bei der Beispielerzeugung zurückzuführen, wobei ein differentialdiagnostisch kritisches Merk-

mal mit einem Wert erzeugt wurde, der die erwartete Diagnose nicht stützt. Zum Beispiel wurde bei falsch klassifizierten Beispielen, bei denen Migräne ohne Aura erwartet wurde, das Merkmal *Aurasymptome* mit dem Wert *true* erzeugt und bei erwarteter Migräne mit Aura mit dem Wert *false*. Zudem kann erwartet werden, dass Falschklassifizierungen besonders bei Beispielen auftreten, bei denen für zwei Diagnosen ähnliche Wahrscheinlichkeiten errechnet werden. Dieser Fall tritt sehr selten auf und nur dann, wenn ohnehin sämtliche Diagnosen sehr geringe Wahrscheinlichkeiten haben. Es finden sich ein paar Beispiele, bei denen Spannungskopfschmerz oder Clusterkopfschmerz erwartet wird und Hirntumor mit sehr geringer absoluter Wahrscheinlichkeit die (relativ) wahrscheinlichste Diagnose ist. Auch hier wurde ein Hirntumor-typisches Symptom bei der Beispielerzeugung mit einem positiven Wert erzeugt, so dass dieser die falsche Diagnose stützt.

5.3.2 Ergebnisse der Inferenz mit Variational Message Passing

Variational Message Passing zeigt für vier der fünf Diagnosen ähnlich gute Ergebnisse wie EP und JTBP, klassifiziert allerdings bei Beispielen, bei denen *Migräne mit Aura* erwartet wird, häufiger falsch als richtig (Tabelle 8). Ein Blick auf die Symptome zeigt, dass die Ursache nicht in den Beispielen liegt, denn es werden Beispiele falsch klassifiziert, bei denen die Merkmalswerte die erwartete Diagnose zum großen Teil stützen. Zudem konnten die meisten dieser Beispiele von EP und JTBP korrekt klassifiziert werden.

5.3.3 Fazit

Expectation Propagation und Join Tree Belief Propagation liefern für Netze in der medizinischen Diagnostik sehr gute Ergebnisse. Wenngleich VMP für die meisten Diagnosen vergleichbare Ergebnisse liefert, so ist ein partieller Qualitätseinbruch inakzeptabel. Nach jetzigem Stand der Erkenntnis sind sowohl EP und als auch JTBP geeignet, um Anfragen auf einem mehrfach verbundenen diskreten Bayesschen Netz durchzuführen. Von den in Infer.NET implementierten Algorithmen ist also Expectation Propagation für den angestrebten Anwendungszweck die Methode der Wahl und gegenüber Variational Message Passing klar vorzuziehen.

6 Zusammenfassung

Um die Qualität eines Bayesschen Netzes und eines Inferenzalgorithmus zu bewerten, wurde ein prototypisches Softwaresystem geschaffen, um Beispielabfragen für ein Netz zu generieren und diese mithilfe des Netzes auszuwerten. Das Werkzeug wurde genutzt, um die von Infer.NET zur Verfügung gestellten Algorithmen *Expectation Propagation* und *Variational Message Passing* zu analysieren und mit *Join Tree Belief Propagation* zu vergleichen. Ebenso wurde das Bayessche Netz aus Abbildung 8 modelliert und festgestellt, dass es für die modellierte Domäne der Kopfschmerzdiagnostik sehr gute Ergebnisse liefert. Zudem wurde gezeigt, dass EP als Inferenzalgorithmus für diese Domäne geeigneter ist als VMP und eine ebenso hohe Inferenzqualität liefert wie JTBP.

Um bspw. die Domäne der Schmerzdiagnostik vollständig abzudecken, ist die Konstruktion weiterer Bayesscher Netze notwendig. In [Nik00] werden geeignete Konstruktionsmethoden beschrieben, die diese Probleme adressieren und in einem iterativen Entwurfsprozess angewendet werden können.

Um die konstruierten Bayesschen Netze in diagnostischen Systemen nutzen zu können, muss

der entwickelte Agent in das jeweilige System integriert und mit der bestehenden Schnittstelle zur Abfrage von Anamnesedaten verbunden werden. Dabei können die Informationen Bayesscher Netze auf eine Weise genutzt werden, die über die reine Ermittlung von Diagnosewahrscheinlichkeiten zu gegebenen Anamnesen hinausgehen.

References

[AKN12] ABBEEL, Pieter ; KOLLER, Daphne ; NG, Andrew Y.: Learning Factor Graphs in Polynomial Time & Sample Complexity. In: *CoRR* abs/1207.1366 (2012). http://arxiv.org/abs/1207.1366;http://dblp.uni-trier.de/rec/bib/journals/corr/abs-1207-1366

[Ana17] *Analytica by Lumina Decision Systems*. 2017. – unter http://www.lumina.com (abgerufen am 29.01.2017)

[And86] ANDERSON, Herbert L.: Metropolis, Monte Carlo, and the MANIAC. In: *Los Alamos Science* (1986), Feb, S. 96–107

[Bay17] BAYESFUSION, LLC: *GeNIe/SMILE*. 2017. – unter http://www.bayesfusion.com/products (abgerufen am 29.01.2017)

[BGT93] BERROU, C. ; GLAVIEUX, A. ; THITIMAJSHIMA, P.: Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. In: *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on* Bd. 2, 1993, S. 1064–1070 vol.2

[BKI06] BEIERLE, C. ; KERN-ISBERNER, G.: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. 3., erweiterte Auflage. Wiesbaden : Vieweg Verlag, 2006

[Coz17] COZMAN, Fabio G.: *JavaBayes - Bayesian Networks in Javs of Escola Politécnica, University of São Paulo*. 2017. – unter http://www.cs.cmu.edu/~javabayes (abgerufen am 29.01.2017)

[DD06] DÍEZ, Francisco J. ; DRUZDZEL, Marek J.: Canonical probabilistic models for knowledge engineering. (2006)

[eBa14] EBAY: *Bayesian Belief Networks*. https://github.com/eBay/bayesian-belief-networks, August 2014

[FP15] FLÜGGE, Sebastian ; PETERSOHN, Uwe: *Repräsentation einer Wissensbasis zu Kopfschmerzen und diagnostische Inferenz mittels Bayesscher Netze*. 2015

[GG84] GEMAN, S. ; GEMAN, D.: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6 (1984), Nov, Nr. 6, S. 721–741. – ISSN 0162–8828

[JGJS99] JORDAN, Michael I. ; GHAHRAMANI, Zoubin ; JAAKKOLA, Tommi S. ; SAUL, Lawrence K.: An Introduction to Variational Methods for Graphical Models. In: HECKERMAN, David (Hrsg.): *Machine Learning* Bd. 37, Kluwer Academic Publishers, 1999, 183-233

[KP83] KIM, Jin H. ; PEARL, Judea: A computational model for combined causal and diagnostic reasoning in inference systems. In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* Bd. IJCAI-83. Karlsruhe, 1983, S. 190–193

[Ltd17] LTD., Bayes S.: *Bayes Server*. 2017. – unter `http://www.bayesserver.com` (abgerufen am 29.01.2017)

[MBO] *(Muster-)Berufsordnung für die in Deutschland tätigen üArztinnen und Ärzte (MBO-Ä 1997)*. – unter `https://www.bundesaerztekammer.de/fileadmin/user_upload/downloads/pdf-Ordner/MBO/MBO-AE.pdf` (abgerufen am 20.03.2019)

[Min01] MINKA, Thomas P.: Expectation Propagation for Approximate Bayesian Inference. In: *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001 (UAI '01). – ISBN 1–55860–800–1, 362–369

[MK07] MOOIJ, J.M. ; KAPPEN, H.J.: Sufficient Conditions for Convergence of the Sum-Product Algorithm. In: *Information Theory, IEEE Transactions on* 53 (2007), Dec, Nr. 12, S. 4422–4437. `http://dx.doi.org/10.1109/TIT.2007.909166`. – DOI 10.1109/TIT.2007.909166. – ISSN 0018–9448

[MWG+18] MINKA, T. ; WINN, J.M. ; GUIVER, J.P. ; ZAYKOV, Y. ; FABIAN, D. ; BRONSKILL, J.: */Infer.NET 0.3*. 2018. – Microsoft Research Cambridge. http://dotnet.github.io/infer

[Nik00] NIKOVSKI, Daniel: Constructing Bayesian networks for medical diagnosis from incomplete and partially correct statistics. In: *EEE Transactions on Knowledge and Data Engineering* 12 (2000), July, Nr. 4, S. 509–516

[OP98] OERTEL, W. ; PETERSOHN, U.: Managing Episodes, Cases, Concepts, and Rules - an Integration Approachfor Medical Problem Solving. In: AHA, D. (Hrsg.) ; DANIELS, J. (Hrsg.): *Case-Based Reasoning Integrations: Papers from the 1998 Workshop*. Madison, Wisconsin, USA, Menlo Park : AAAI Press, 1998

[PDT03] PRZYTULA, K. W. ; DASH, Denver ; THOMPSON, Don: Evaluation of Bayesian Networks Used for Diagnostics. (2003), March

[Pea82] PEARL, Judea: Reverend Bayes on Inference Engines - A Distributed Hierarchical Approach. In: *Proceedings of the Second National Conference on Artificial Intelligence* Bd. AAAI-82: Pittsburgh, PA. Menlo Park, California : AAAI Press, 1982, S. 133–136

[PGIB09] PETERSOHN, U. ; GUHLEMANN, S. ; IWER, L. ; BALZER, R.: Problem solving with Concept- and Case-based Reasoning. In: *Proceedings of the Conference Developments in Artifical Intelligence Methods*, 2009

[RN95] RUSSELL, Stuart ; NORVIG, Peter: *Artificial Intelligence - A Modern Approach*. Prentice-Hall, 1995

[RN09] RUSSELL, S. J. ; NORVIG, P.: *Artificial Intelligence: A Modern Approach*. 3rd. Prentice Hall, 2009

[S.A17] S.A.S., Bayesia: *BayesiaLab*. 2017. – unter `http://www.bayesia.com` (abgerufen am 29.01.2017)

[SM12] SUTTON, Charles A. ; MCCALLUM, Andrew: Improved Dynamic Schedules for Belief Propagation. In: *CoRR* abs/1206.5291 (2012). http://arxiv.org/abs/1206.5291;http://dblp.uni-trier.de/rec/bib/journals/corr/abs-1206-5291

[SS05] *Kapitel* Entscheidungsunterstützende Systeme und wissensbasierte Methoden in der Medizin. In: SPRECKELSEN, C. ; SPITZER, K.: *Handbuch der Medizinischen Informatik*. 2., vollständig neu bearbeitete Auflage. München, Wien : Carl Hanser Verlag, 2005, S. 483–548

[SS08] SPRECKELSEN, C. ; SPITZER, K.: *Wissensbasen und Expertensysteme in der Medizin: KI-Ansätze zwischen klinischer Entscheidungsunterstützung und medizinischem Wissensmanagement*. 1. Auflage. Vieweg+Teubner Verlag, 2008