Technische Universität Dresden
Fakultät Informatik
Diplomarbeit zum Thema

Inkrementelle Fuzzy-Entscheidungsbäume

zur Erlangung des akademischen Grades
Diplom-Informatikerin

von

Marina Guetova
Matrikelnummer 2452240

Verantwortlicher Hochschullehrer: Prof. Dr. Steffen Hölldobler
Fachgebiet Wissensverarbeitung
Institut Künstliche Intelligenz
Betreuer: Dipl.-Inf. Hans-Peter Störr

begonnen am 15.6.2001
abgegeben am 15.12.2001

Erklärung

Hiermit versichere ich, daß ich die vorliegende schriftliche Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Dresden, den 15.12.2001

Marina Guetova

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Aufgaben des Gebiets „Künstliche Intelligenz“ ([RN95]) beinhalten u.a. die Analyse und Klassifikation von Datenmengen aus unterschiedlichen Gegenstandsbereichen unseres Lebens. Ziel der Analyse und der Klassifikation ist es, Zusammenhänge zwischen den Gegenständen eines Bereichs zu finden und anhand dieser Ähnlichkeiten die Gegenstände zu gruppieren. Mit Hilfe der festgestellten Eigenschaften der einzelnen Gruppen können neue, unbekannte Gegenstände zu einer bereits existierenden Gruppe zugeordnet werden. Den Prozeß des Gewinnens solcher Regeln über einen Bereich nennt man Lernprozeß, die verwendeten Verfahren Lernalgorithmen.

In diesem Kapitel wird auf verschiedene Lernalgorithmen verwiesen, die zur Klassifikation von Daten eingesetzt werden können. Als Motivation zum Einsatz von Lernverfahren wird das Aufgabenfeld des Projekts ISeC analysiert. Das Kapitel schließt mit einem Überblick über den weiteren Inhalt der Arbeit und mit einer Tabelle der verwendeten Symbole ab.

Aus der Literatur sind zahlreiche Ansätze für die Klassifikation von Daten bekannt. Wir unterscheiden einerseits zwischen Verfahren, die ein inkrementelles Lernen und solchen, die ein nicht-inkrementelles Lernen durchführen, und anderseits zwischen Algorithmen, die Fuzzy-Logik unterstützen und solchen, die keine Fuzzy-Logik einsetzen können.

Algorithmen, die inkrementelles Lernen unterstützen, sind in der Lage, das erzeugte Datenmodell an neue Beispiele anzupassen. Dagegen müssen die Verfahren, die ein nicht-inkrementelles Lernen ausüben, bei der Beobachtung neuer Beispiele ein neues Modell anhand der alten und der neuen Gegenstände erstellen.

Eine zentrale Rolle im Gebiet der „Künstlichen Intelligenz“ spielen die Neuronalen Netze ([NKK94, RN95]), die als „Nachbildung“ des menschlichen Gehirns entwickelt wurden. Ein Neuronales Netz besteht aus Einheiten (*Units*), die durch Verbindungen (*Links*) miteinander gekoppelt sind. Jede Verbindung ist mit einem Gewicht versehen. Ein wesentlicher Bestandteil des Modells Neuronaler Netze sind Algorithmen, die eine adaptive Veränderung der Gewichte ermöglichen. Das Ziel dieses Lernvorgangs ist es, die Gewichte so zu bestimmen, daß das Netz auf bestimmte Eingaben mit bestimmten Ausgaben reagiert. Somit kann ein Neuronales Netz für neue, unbekannte Eingaben korrekte Ausgaben berechnen. Die Neuronalen Netze besitzen eine hohe Lernfähigkeit, und sind in der Lage, sich an neue Aufgaben durch Gewichtsveränderungen anzupassen. Ihre Nachteile bestehen darin, daß sie schwerer zu interpretieren sind und eine lange Trainingsphase für die Anpassung der Gewichte benötigen.

Dagegen bieten die Entscheidungsbäume eine sehr gute Visualisierung des Lernergebnisses und sind für den Menschen leicht zu verstehen. Es wurden mehrere Verfahren zur Erstellung (Induktion) von Entscheidungsbäumen entwickelt, die auf den Algorithmen ID3 von J.R. Quinlan (siehe [Qui86]), CART (siehe [BFOS84]) und C4.5 (siehe [Qui93]) beruhen.

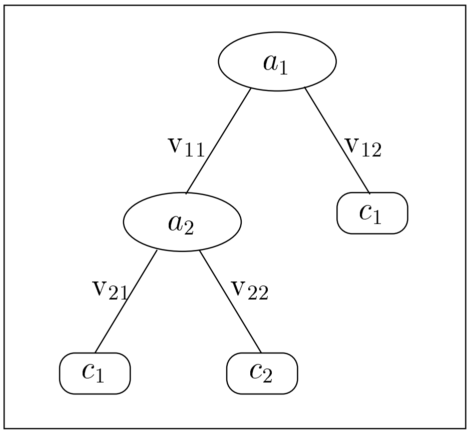

Abbildung 1.1: Entscheidungsbaum

Jeder innere Knoten eines Entscheidungsbaumes dient zur Verzweigung nach einem bestimmten Kriterium, jeder Blattknoten repräsentiert eine Klasse, die durch die Kriterien auf dem Weg von der Wurzel bis zum Blattknoten beschrieben werden kann.

Der Entscheidungsbaum aus Abbildung 1.1 wird wie folgt interpretiert: wenn ein unbekannter Gegenstand für Attribut a_1 den Wert v_{12} hat, dann wird er als Klasse c_1 klassifiziert, wenn er dagegen den Wert v_{11} hat, wird er in Abhängigkeit von seinem Wert für das Attribut a_2 klassifiziert.

Einen Versuch, die Ideen des inkrementellen Lernens und der Entscheidungsbäume zu verbinden, stellt der Algorithmus LDT (*Lazy Decision Tree*) [FKY96] dar, der keinen vollständigen Entscheidungsbaum erzeugt, sondern für jedes zu klassifizierende Beispiel genau den einen Pfad des Baumes erstellt, der dieses Beispiel klassifiziert. Bei einer geringen Anzahl von Beispielen führt dieses Verfahren zu einer schnelleren Klassifikation als bei ID3, wo zuerst der Entscheidungsbaum generiert werden muß und danach anhand dieses Baumes alle unbekannten Beispiele klassifiziert werden können. Haben wir eine große Lernmenge, so ist es effizienter, zuerst den Entscheidungsbaum aufzustellen und ihn danach zur Klassifikation zu nutzen.

Ausgehend vom Algorithmus zur Induktion von Entscheidungsbäumen ID3 wurden Verfahren zum inkrementellen Aufbau von Entscheidungsbäumen entwickelt, bekannt als ID4 (siehe [SF86]) und ID5 (siehe [Utg94]). In [Utg88] wird ein theoretischer und experimenteller Vergleich der Algorithmen ID3, ID4 und ID5 vorgenommen. Der Algorithmus ID5 wurde zum Algorithmus ITI (*Incremental Tree Induction*) ([UBC97], den wir in Unterkapitel 4.1 vorstellen) weiterentwickelt. Dieser Algorithmus ist in der Lage, bei der Beobachtung eines neuen Beispiels den existierenden Entscheidungsbaum, der die Zusammenhänge in einer gegebenen Menge von Beispielen wiedergibt, und dieses neue Beispiel auf einen neuen Entscheidungsbaum abzubilden. Der neue Entscheidungsbaum repräsentiert dann genau die Verhältnisse innerhalb der alten Menge zusätzlich des neuen Beispiels.

Verweise auf einen inkrementellen Algorithmus zum Verfahren CART (siehe [Cra89]) und weitere relevante Arbeiten sind in [UBC97] gegeben.

Der Algorithmus CDL4 (*Complementary Discrimination Learning*) [She97] baut inkrementell eine Entscheidungsliste auf, die aus einer Menge von Paaren besteht. Jedes Paar besteht aus einer Bedingung (Prämisse) und einer Entscheidung (Klasse). Die Vorbedingung ist entweder als Konjunktion von einfachen Tests – ein Attribut nimmt einen bestimmten Attributwert an oder nicht – , oder als *default* definiert. Die Entscheidung repräsentiert eine Klasse. Beim Eintreffen eines neuen Beispiels durchläuft man die Liste und wählt das erste Paar, dessen Vorbedingung durch das Beispiel erfüllt wird. Ist das Beispiel korrekt klassifiziert, so wird kein Lernprozeß gestartet. Ist dies nicht der Fall, wird das gefundene Paar durch ein oder mehrere neue Paare mit zusätzlichen Tests in der Konjunktion so ersetzt, daß das neue Beispiel die Vorbedingungen in den neu entstandenen Paaren nicht mehr erfüllt, sowie das neue Beispiel wird als Paar in die Liste aufgenommen.

Alle bis jetzt erwähnten Lernalgorithmen basieren auf der so genannten scharfen Logik. Vor allem für Gegenstandsbereiche, deren Beispielmengen nicht künstlich aufgestellt wurden, ist es oft schwierig, scharfe Grenzen für die Eigenschaften eines Objekts zu definieren. Zur Abhilfe wird die unscharfe Logik, bekannt als Fuzzy-Logik (siehe [KKG93]), benutzt.

In der Welt der Fuzzy-Logik können wir die Aussage, ob ein Objekt eine Eigenschaft besitzt oder nicht, beliebig abstufen. Dafür wird der Begriff Zugehörigkeitsgrad eingeführt. Der Zugehörigkeitsgrad nimmt Werte aus dem Intervall $[0, 1]$ an, wobei einem Zugehörigkeitsgrad von Null entspricht, daß das Objekt die Eigenschaft nicht hat, und einem Zugehörigkeitsgrad von Eins, daß das Objekt diese Eigenschaft besitzt.

Die Vorteile der Klassifikation mit Fuzzy-Logik führten zur Entwicklung von Algorithmen, die unter Benutzung der Theorie der Fuzzy-Logik eine flexible Klassifikation ermöglichen. Weit verbreitet sind die Ideen zur Kombination von Fuzzy-Logik mit Neuronalen Netze zu einem Neuro-Fuzzy-System (siehe [NKK94, Nau94, NK95, NK96, NK97, LFM95]) und mit Entscheidungsbäumen, den so genannten Fuzzy-Entscheidungsbäumen.

Nicht-inkrementelle Algorithmen zum Aufbau von Fuzzy-Entscheidungsbäumen sind unter anderem in [Zei99] (dort findet man Verweise auf weitere Ansätze) und [Jan98] vorgestellt. Eine Kombination der Vorteile des inkrementellen Lernens von Entscheidungsbäumen mit den Vorteilen der Fuzzy-Logik ist dagegen nicht bekannt. Ziel dieser Arbeit ist es, eine solche Verbindung zu schaffen. Als Motivation dient das gemeinsame Projekt ISeC (*Intelligente Systeme im eCommerce*) zwischen dem Unternehmen HYBRIS AG [hA01] und der Technischen Universität Dresden, Institut Künstliche Intelligenz [Dre01].

In den letzten Jahren, mit der Entwicklung und Verbreitung des World Wide Web und daraus resultierender Popularität des elektronischen Handels, wurde eine neue Anwendungsmöglichkeit der Lernalgorithmen erschlossen. Die Gestaltung der Produktpräsentation, d.h. der Internet-Seiten, spielt eine zunehmende Rolle und trägt zu einem attraktiveren und bequemeren Einkaufen für die Kunden bei. Deswegen wird die Darstellung der Produkte und die Anpassung dieser Präsentation an die individuellen Interessen und Vorlieben der Kunden, die so genannte Benutzermodellierung, ein zunehmend wichtiger Aspekt bei der Entwicklung eines Internet-Shop. Eine solche Anpassung ist nur dann möglich, wenn Kenntnisse über die Interessen und Wünsche des Kunden gewonnen werden. Ein Lernalgorithmus ist in der Lage, anhand des beobachteten Nutzerverhaltens ein solches Benutzerprofil zu erstellen, das als Vorlage zum Aufbau der Seiten dienen kann.

Im Rahmen des Projekts ISeC wurden unter anderem einige der oben beschriebenen Lernverfahren in die Web-Shop Lösung der Firma HYBRIS integriert. Eine detaillierte Beschreibung des Systems findet man in [HSMB01].

Um eine geeignete Lernmenge zu erstellen, müssen Beispiele durch die Beobachtung des Benutzerverhaltens generiert werden, die positiv oder negativ klassifiziert sind. Zur Beschreibung der Beispiele werden Attribute mit bestimmten Wertebereichen verwendet, wobei jedes Beispiel durch genau einen Wert aus dem Wertebereich jedes Attributs spezifiziert ist. Die Medienelemente, die zur Präsentation eines Produkts auf einer Seite benutzt werden, sind in 5 Gruppen unterteilt:

1. *SimpleText* : einfacher Text, der keine Alternativen zur Einbindung bietet
2. *ScrollText* : einfacher Text im Scrollfenster; die Sinneinheit wird als Link, der durch den Benutzer aufgeklappt werden kann oder als Fenster, das zugeklappt werden kann, dargestellt
3. *StretchText* : einfacher Text, der ebenfalls als Link oder einfach aufgeklappt auf der Seite eingebunden wird
4. *Bild* : das Bild kann groß oder klein eingebunden sein und durch den Benutzer verkleinert bzw. vergrößert werden
5. *Video* : das Video kann zum Abspielen bereit eingebunden sein oder als Link

Diese Sinneinheiten sind mit bestimmten Eigenschaften versehen, die die Rolle der Attribute übernehmen. So können durch eine Sinneinheit beispielsweise allgemeine (Technik) oder spezifische (Motor) technische Daten eines Produkts beschrieben werden, oder sie sagt etwas über die Gestaltung (Gestaltung) aus. Solche Attribute nehmen die Attributwerte true oder false an. Die Art der Präsentationselemente (ob der Benutzer sich lieber Bilder anschaut oder Text liest) sagt uns ebenfalls viel über die Vorlieben des Kunden. Daher wird ein zusätzliches Attribut Typ mit den Werten SimpleText, ScrollText, StretchText, Bild, Video eingeführt. Das Interesse der Kunden an den Sinneinheiten hängt ebenfalls von der Größe der Elemente ab, z.B. kann Interesse an einem kurzen, kompakten Text und Desinteresse an langen Texten bestehen. Um diesen Zusammenhang in der Lernmenge berücksichtigen zu können, definieren wir das Attribut Größe mit den Werten klein, mittel, groß.

Jede Benutzerinteraktion (der Benutzer kann die Darstellung der Sinneinheiten verändern und dadurch die Präsentation eines Produktes an seine individuellen Vorlieben anpassen) wird als Interesse/Desinteresse interpretiert: ist z.B. ein Bild, das die Eigenschaften Technik und Motor und eine mittlere Größe hat, klein eingebunden und wird durch den Benutzer vergrößert, so registrieren wir das Beispiel: (Technik=true, Motor=true, Gestaltung=false, Typ=Bild, Größe=mittel, Interesse=positiv).

Wird Fuzzy-Logik unterstützt, so können wir beispielsweise die Größe des Bildes nicht einfach nur als mittel, sondern mit verschiedenen Zugehörigkeitsgraden als klein, mittel und groß definieren. Damit ist eine natürlichere und anpassungsfähige Darstellung möglich.

Mit Verlassen einer Seite wird das Verhalten des Kunden ausgewertet, die Lernmenge erstellt oder erweitert, und ein Lernalgorithmus ausgeführt, der bestimmt, wie die nächste Seite aufgebaut wird, d.h. welche Sinneinheiten wie eingebunden werden (siehe Abbildung 1.2 aus [HSMB01]).

Im Rahmen des Projekts ISeC sind unterschiedliche Lernalgorithmen verglichen und auf ihre Eignung für die Benutzermodellierung bewertet worden (siehe [Gue01]), wobei folgende Anforderungen gestellt sind:

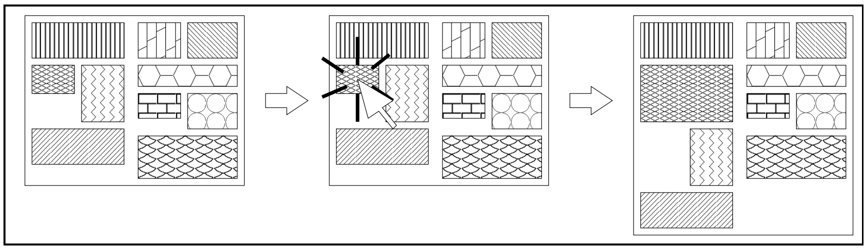

Abbildung 1.2: Einbindung der Sinneinheiten

- die Algorithmen müssen mit symbolischen und numerischen Attributen arbeiten
- die Algorithmen müssen in der Lage sein, inkonsistente Beispielmengen (Objekte mit gleichen Eigenschaften sind unterschiedlich klassifiziert) zu berücksichtigen
- die Algorithmen müssen trotz unvollständiger Beispiele (Beispiele für die unbekannt ist, ob sie eine Eigenschaft besitzen oder nicht) ein gutes Lernergebnis liefern
- leichte, intuitive Darstellungsform des Ergebnisses
- schnelle Auswertung des Benutzerverhaltens und Anpassung des Benutzerprofils, d.h. inkrementelles Lernen
- Unterstützung von Fuzzy-Logik

Es wurde beobachtet, daß Lernalgorithmen, die Entscheidungsbäume oder -listen erstellen, von Vorteil sind, da sie eine gute, intuitive Interpretierung des Lernergebnisses bieten. Dagegen sind Neuronale Netze ungünstig, weil sie eine lange Trainingsphase durchlaufen. Das Einsetzen von Fuzzy-Logik ermöglicht eine flexible Gestaltung der Beispiele und der Klassifikation. Wie oben erwähnt, sind mehrere Verfahren zum Erstellen von Fuzzy-Entscheidungsbäumen bekannt. Ein inkrementelles Lernen ist wünschenswert, um die Zeit zur Erstellung eines neuen Benutzerprofils zu reduzieren. Damit ist es offensichtlich, daß das Verbinden von Fuzzy-Entscheidungsbäumen mit inkrementellem Lernen erstrebenswert ist. Ziel dieser Arbeit ist es, ein Verfahren zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen zu entwickeln, welches das gleiche Ergebnis wie ein Verfahren zum nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen liefert. Somit können wir ohne Verluste den bereits integrierten nicht-inkrementellen Algorithmus zur Induktion von Fuzzy-Entscheidungsbäumen durch den neuen inkrementellen Algorithmus ersetzen.

Die Arbeit beginnt mit einer kurzen Einführung in der Theorie der Fuzzy-Logik und der linguistischen Variablen, die als Grundlage für die unscharfe Modellierung von Attributen dient. Kapitel 3 beschäftigt sich mit dem nicht-inkrementellen Aufbau von klassischen und Fuzzy-Entscheidungsbäumen, wobei der Algorithmus IFDT zum Aufbau von Fuzzy-Entscheidungsbäumen ([Zei99, Jan98]) vorgestellt wird. Die grundlegenden Gedanken des inkrementellen Lernens von klassischen Entscheidungsbäumen (der Algorithmus ITI [UBC97]) und Fuzzy-Entscheidungsbäumen findet man in Kapitel 4. Der Algorithmus IIFDT zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen wird in Kapitel 5 entwickelt. Seine Eigenschaften werden in Kapitel 6 definiert und nachgewiesen. Das Erfüllen dieser Eigenschaften dient als Voraussetzung für die Äquivalenz zum Algorithmus für den nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen, die in Unterkapitel 6.4 gezeigt wird. Kapitel 7 schließt die Arbeit mit einer Zusammenfassung und einem weiteren Ausblick ab.

$A = \{a_1, \dots, a_p\}$	Menge der Attribute
$V_i = \{\mathrm{v}_{i1}, \mathrm{v}_{i2}, \dots, \mathrm{v}_{ij_i}\}$	Menge der Attributwerte eines Attributs $a_i \in A$
$\mathcal{C} = \{c_1, \dots, c_q\}$	Menge der Klassen
$\mathcal{V}$	Bezeichnung einer linguistischen Variable
ν	der Name einer linguistischen Variable
$T(\nu)$	Termmenge, Menge der linguistischen Werte einer Variable
$term(\nu)$	Term einer Variable, der linguistische Wert
$\xi_{term(\nu)}(x)$	Zugehörigkeitsgrad des Eingabewertes x zu dem linguistischen Wert $term(\nu)$
$t = [cond_1^t, \dots, cond_{u_t}^t]$	Test mit den Bedingungen $cond_h^t, 1 \leq h \leq u_t$
n	Knoten eines Entscheidungsbaumes
$T(n)$	Teilbaum eines Entscheidungsbaumes, beginnend am Knoten n
$n.cond_h^t$	Nachfolgerknoten des Knotens n
$\mathcal{L}$	Lernmenge mit $\{1, 2, \dots, f, \dots, m\}$ Beispielen
$\mu_{\mathrm{v}_{il}}(f)$	Zugehörigkeitsgrad des Beispiels f zu dem linguistischen Wert $\mathrm{v}_{il} \in V_i$
$\mu_{c_k}(f)$	Zugehörigkeitsgrad des Beispiels f zu der Klasse $c_k \in \mathcal{C}$
$\mu_{cond_h^t}(f)$	Erfüllungsgrad der Bedingung $cond_h^t$ durch das Beispiel f
$\lambda_n(f)$	Zugehörigkeitsgrad des Beispiels f zu dem Knoten n
$\langle value \rangle$	der Wert $value$ ist an einem Knoten eines inkrementellen Fuzzy-Entscheidungsbaumes gespeichert
$\Delta_{n,f}$	der Wert, um den der Zugehörigkeitsgrad des Beispiels f zum Knoten n, durch die Ausführung der Funktion **addExampleToTree**, erhöht werden soll
$\mathcal{M}$	Methode zur Auswahl des besten Tests
$\mathcal{F}$	Funktion zum Testen der Abbruchbedingungen und zur Beschriftung der Blattknoten
$\mathcal{T}$	Menge der zulässigen Tests

Tabelle 1.1: Tabelle der verwendeten Symbole

Kapitel 2

Fuzzy-Logik

In diesem Kapitel werden grundlegende Begriffe aus der Theorie der Fuzzy-Logik eingeführt, z.B. Fuzzy-Mengen und linguistische Variablen, die zur unscharfen Modellierung von Attributen verwendet werden. Als Darstellungsform einer Fuzzy-Menge wird die lineare charakteristische Funktion beschrieben. Die Funktionen t-Norm und t-Conorm werden eingeführt, um Verknüpfungen (Vereinigung und Durchschnitt) zweier Fuzzy-Mengen zu definieren.

2.1 Fuzzy-Mengen

Betrachten wir ein bestimmtes Universum (Grundmenge), z.B. die Menge der natürlichen Zahlen $\mathbb{N}$. Um eine Eigenschaft relativ zu $\mathbb{N}$ zu charakterisieren, gibt es nach [KKG93] drei Repräsentationsformen. Wollen wir z.B. die Eigenschaft *zweistellig* beschreiben, so ergeben sich folgende drei Möglichkeiten zur Definition der Teilmenge $Z \subseteq \mathbb{N}$:

- Aufzählung der einzelnen Elemente:

$$Z = \{10, 11, 12, \ldots, 97, 98, 99\} \subseteq \mathbb{N}$$

- ein Prädikat (die Elemente des Universums, die das Prädikat erfüllen, gehören zur beschriebenen Menge):

$$Z = \{n \in \mathbb{N} | \; \mathit{zweistellig}(n)\}$$

- Angabe der zugehörigen charakteristischen Funktion:

$$\mathbb{I}_Z(n) = \begin{cases} 1 & \text{wenn } n \text{ \textit{zweistellig}} \\ 0 & \text{sonst} \end{cases}$$

Der Wertebereich einer charakteristischen Funktion ist die Menge $\{0, 1\}$. Die Elemente 0 und 1 werden als Zugehörigkeitsgrade bezeichnet. Dabei bedeutet 1: das Element gehört zu der Menge; und 0: das Element gehört nicht zu der Menge. Damit kann man die Grenze zwischen der Menge und ihrer Komplementmenge genau definieren.

Für viele Begriffe aus unserer täglichen Sprache wirkt so eine scharfe Charakterisierung unnatürlich. Nehmen wir als Beispiel die Eigenschaften einer Sinneinheit von Typ Bild. Oft ist es schwer zu bestimmen, ob das gegebene Bild als groß, mittel oder klein zu definieren ist. In solchen Fällen ist es sinnvoller, abgestufte Übergänge zu modellieren. Neben den

Zugehörigkeitsgraden 0 und 1 sind noch weitere Zugehörigkeitsgrade zu verwenden. Der Wertebereich besteht nicht mehr aus der Menge $\{0,1\}$, sondern aus dem Intervall $[0,1]$. Charakteristische Funktionen, deren Wertebereich das Intervall $[0,1]$ ist, nennt man verallgemeinerte charakteristische Funktionen. Je näher der Zugehörigkeitsgrad bei 1 liegt, desto mehr wird das Prädikat erfüllt. Durch die verallgemeinerte charakteristische Funktion lassen sich nicht nur gewöhnliche Mengen darstellen, sondern auch unscharfe Mengen, sogenannte Fuzzy-Mengen.

Definition 2.1 *Eine* **Fuzzy-Menge** *μ von X ist eine Funktion von der Grundmenge bzw. dem Universum X in das Einheitsintervall, d.h.*

$$\mu : X \to [0,1]$$

Definition 2.2 *Sei μ eine Fuzzy-Menge über die Grundmenge X. Die* **Größe** *(Mächtigkeit) der Fuzzy-Menge bezeichnen wir durch $|\mu|$ und berechnen sie wie folgt:*

$$|\mu| = \sum_{x \in X} \mu(x)$$

Am häufigsten werden Fuzzy-Mengen mit der sie charakterisierenden Funktion identifiziert. Für die Aufstellung einer charakteristischen Funktion sind verschiedene Möglichkeiten bekannt – es wird nach linearen und nicht-linearen charakteristischen Funktionen unterschieden. Die Unterschiede zwischen den linearen charakteristischen Funktionen beziehen sich auf ihre Form. Zur Übersicht werden in Abbildung 2.1 die am häufigsten verwendeten Formen schematisch dargestellt.

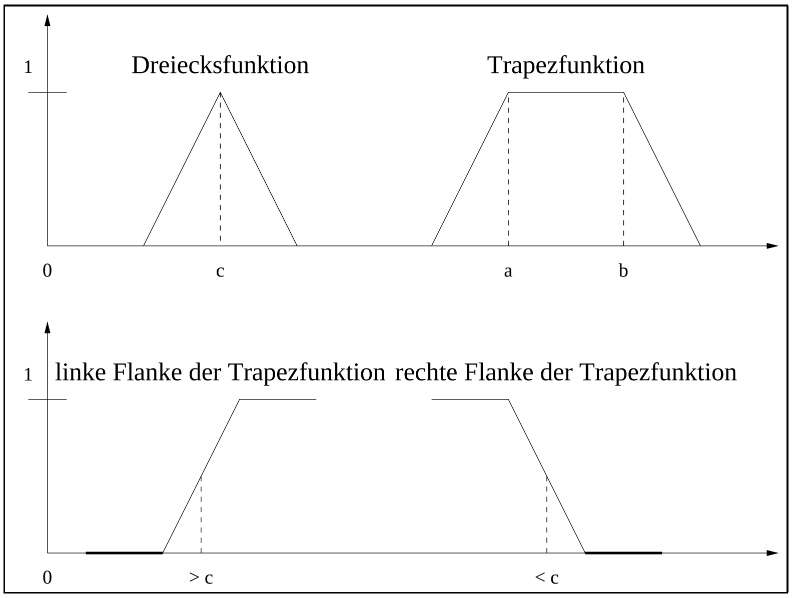

Abbildung 2.1: Beispiele für lineare charakteristische Funktionen

Die Trapezfunktion als Vertreter der linearen charakteristischen Funktionen ist für die Interpretation von linguistischen Ausdrücke der Art *ungefähr zwischen a und b* geeignet. Ein Spezialfall der Trapezfunktion ist die symmetrische Dreiecksfunktion. Sie kommt zum Einsatz bei der Darstellung von Ausdrücken wie *etwa(c)*, wobei *c* eine beliebige Zahl ist. Solche Ausdrücke bezeichnet man auch als Fuzzy-Zahlen.

Die linearen charakteristischen Funktionen können durch wenige Punkte beschrieben werden, was als Vorteil gegenüber den nichtlinearen Funktionen angesehen wird. Nichtlineare charakteristische Funktionen werden ähnlich wie die Trapezfunktionen definiert. Im Gegensatz zu linearen charakteristischen Funktionen werden für die Definition der Flanken aber nichtlineare Funktionen—Polynome dritten Grades oder arctan-Funktionen—benutzt [Zei99].

Die klassischen mengentheoretischen Grundoperationen Durchschnitt und Vereinigung können auf Fuzzy-Mengen übertragen werden. Zur Berechnung von Fuzzy-UND bzw. Fuzzy-ODER zwischen zwei Zahlen aus dem Intervall $[0, 1]$ werden in [KKG93] die Funktionen t-Norm bzw. t-Conorm definiert.

Definition 2.3 *Eine Funktion*

$$\top : [0,1]^2 \to [0,1]$$

heißt **t-Norm***, wenn die Bedingungen*

1. $\top(a, 1) = a$ *(Einselement)*
2. $a \leq b \Rightarrow \top(a, c) \leq \top(b, c)$ *(Monotonie)*
3. $\top(a, b) = \top(b, a)$ *(Kommutativität)*
4. $\top(a, \top(b, c)) = \top(\top(a, b), c)$ *(Assoziativität)*

erfüllt sind.

Definition 2.4 *Eine Funktion*

$$\perp : [0,1]^2 \to [0,1]$$

heißt **t-Conorm***, wenn die Bedingungen*

1. $\perp(a, 0) = a$ *(Einselement)*
2. $a \leq b \Rightarrow \perp(a, c) \leq \perp(b, c)$ *(Monotonie)*
3. $\perp(a, b) = \perp(b, a)$ *(Kommutativität)*
4. $\perp(a, \perp(b, c)) = \perp(\perp(a, b), c)$ *(Assoziativität)*

erfüllt sind.

Zwischen t-Normen und t-Conormen wird die duale Beziehung

$$\perp(a, b) = 1 - \top(1 - a, 1 - b)$$

gefordert.

Um die Notation in dieser Arbeit zu vereinfachen und auf Grund der drei Eigenschaften – Einselement, Kommutativität und Assoziativität, die eine t-Norm bzw. eine t-Conorm besitzen, können wir sie nicht nur als zweistellige, sondern auch als n-stellige Funktionen definieren (für $a_1, a_2, \ldots, a_n \in [0, 1]$):

$$\top(a_1, a_2, \ldots, a_n) = \top(a_1, \top(a_2, \ldots \top(a_{n-1}, a_n) \ldots)) \tag{2.1}$$

$$\perp(a_1, a_2, \ldots, a_n) = \perp(a_1, \perp(a_2, \ldots \perp(a_{n-1}, a_n) \ldots)) \tag{2.2}$$

In der Literatur sind verschiedene Paare von t-Normen und t-Conormen definiert (siehe [KKG93]), die bekanntesten sind (für $a, b \in [0, 1]$):

- von L. A. Zahdeh definierte *min* und *max*

$$\top_{min}(a,b) = min\{a,b\} \quad (2.3)$$
$$\bot_{min}(a,b) = max\{a,b\} \quad (2.4)$$

- *algebraisches Produkt* und *algebraische Summe*

$$\top_{prod}(a,b) = a \cdot b \quad (2.5)$$
$$\bot_{prod}(a,b) = a + b - a \cdot b \quad (2.6)$$

Nachdem wir die Grundoperationen t-Norm und t-Conorm auf Zahlen aus dem Intervall $[0,1]$ definiert haben, können wir jetzt die Basisverknüpfungen (Durchschnitt und Vereinigung) zwischen zwei Fuzzy-Mengen einführen.

Definition 2.5 *Seien die Fuzzy-Mengen μ und μ' über die Grundmenge X gegeben. Der* **Durchschnitt** *zwischen den Fuzzy-Mengen μ und μ' erzeugt die Fuzzy-Menge*

$$\mu \cap \mu' : X \to [0,1]$$

Der Zugehörigkeitsgrad jedes Element $x \in X$ zu der Fuzzy-Menge $\mu \cap \mu'$ wird wie folgt ausgerechnet:

$$\mu \cap \mu'(x) = \top(\mu(x), \mu'(x))$$

Analog kann man auch die Vereinigung zweier Fuzzy-Mengen definieren. Dabei wird anstatt der t-Norm eine t-Conorm eingesetzt.

Die Abbildung 2.2 stellt den Durchschnitt zweier Fuzzy-Mengen dar, wobei für die t-Norm das Minimum (Gl. (2.3)) eingesetzt wird.

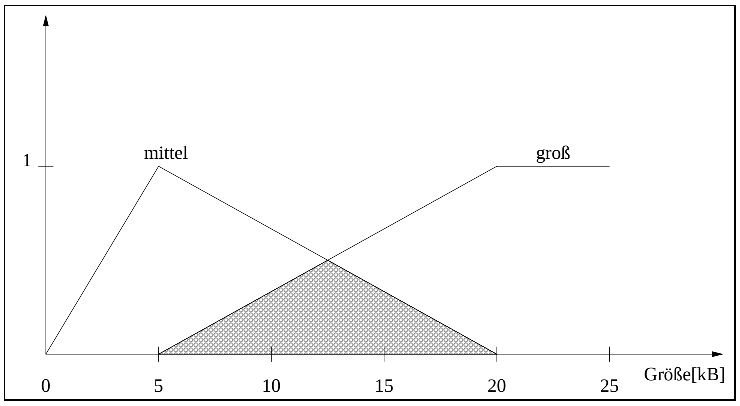

Abbildung 2.2: Beispiel für den Durchschnitt zweier Fuzzy-Mengen

2.2 Linguistische Variablen

Wir haben bereits den Begriff der linguistischen Daten im Zusammenhang mit den Fuzzy-Mengen eingeführt. Stellen wir uns jetzt vor, wir wollen unser Wissen über einen Gegenstandsbereich beschreiben. Handelt es sich dabei um einen Bereich aus dem alltäglichen Leben, so ist es einfacher, die Eigenschaften, das Verhalten eines Systems durch sprachliche Begriffe als durch mathematische Formeln zu spezifizieren. Die Eigenschaften eines Gegenstandes sind durch eine Menge von Attributen beschrieben. Sind die Werte, die von einem solchen Attribut angenommen werden, linguistische Daten, so spricht man von einer linguistischen Variablen.

Definition 2.6 *Eine* **linguistische Variable** $\mathcal{V}$ *ist ein Tupel* $(\nu, T(\nu), X, G, M)$*, mit:*

- ν *ist der Name der Variable*
- $T(\nu)$ *bezeichnet eine Menge von Termen, genannt die linguistischen Werte von* ν*:* $T(\nu) = \{term_1(\nu), term_2(\nu), \ldots, term_j(\nu)\}$ *für* $j \geq 1$ *und* $j \in \mathbb{N}$
- X *bezeichnet die Grundmenge von* $\mathcal{V}$*, bzw. das Universum*
- G *ist eine Grammatik zum Generieren der Termnamen, die linguistische Werte von* ν *sind*
- M *ist eine semantische Regel, die jeden Term aus* $T(\nu)$ *mit seiner Bedeutung assoziiert, also die Zuordnung von linguistischen Werten zu Zugehörigkeitsfunktionen*

In unseren Beispielen wird keine Grammatik G benötigt, da die Termmenge jeweils nur eine endliche Menge von Bezeichnern ist. M kann als eine Fuzzy-Menge $\xi_{term_l(\nu)}$ zu jedem linguistischen Term $term_l(\nu) \in T(\nu)$ erfaßt werden. Für jeden Eingabewert $x \in X$ können wir dann mit Hilfe von M seine Zugehörigkeitsgrade $\xi_{term_1(\nu)}(x), \xi_{term_2(\nu)}(x), \ldots, \xi_{term_j(\nu)}(x)$ zu jedem linguistischen Wert $term_l(\nu) \in T(\nu)$ der Variable ν ausrechnen. Diesen Prozeß bezeichnen wir als Fuzzifizierung des scharfen Eingabewerts x.

Beispiel 2.7 *Abbildung 2.3 stellt eine linguistische Variable mit dem Name* Größe *dar. Die Variable hat die linguistischen Werte* klein, mittel, groß*, die durch die drei Fuzzy-Mengen* $\xi_{\mathsf{klein}}, \xi_{\mathsf{mittel}}, \xi_{\mathsf{groß}}$ *beschrieben sind. Die Grundmenge der Variable ist* $X = \{0, 25\}$*. Für einen Eingabewert* $x = 15$ *erhalten wir folgende Zugehörigkeitsgraden:*

$$\xi_{\mathsf{klein}}(x) = 0, \xi_{\mathsf{mittel}}(x) = 0.33, \xi_{\mathsf{groß}}(x) = 0.67$$

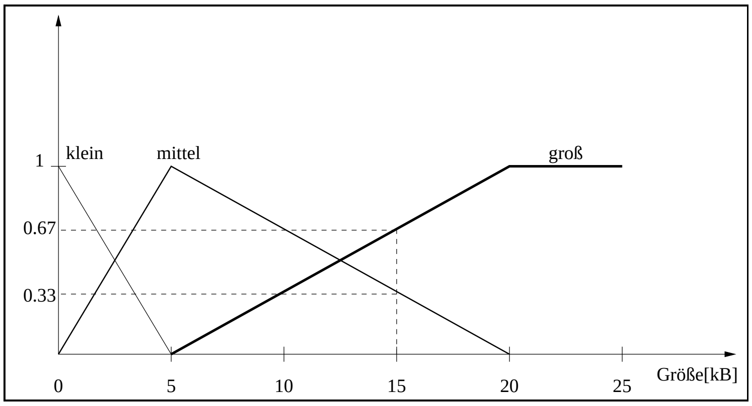

Abbildung 2.3: Beispiel einer linguistischen Variable: Größe

Kapitel 3

Entscheidungsbäume und deren Fuzzy-Erweiterung

Entscheidungsbäume finden eine häufige Anwendung im Gebiet des Maschinellen Lernens. Das Ziel des Maschinellen Lernens ist es, Gegenstände in Abhängigkeit von deren Eigenschaften (Attributen) in verschiedene Klassen einzuordnen und später anhand dieser Klassifizierung unbekannte Gegenstände zu erkennen. Solche Zusammenhänge können unter anderem durch Entscheidungsbäume ausgedrückt werden, wobei gleichzeitig eine sehr gute Visualisierung der Klassifikation erreicht wird.

In Unterkapitel 3.1 werden die Grundbegriffe des klassischen Entscheidungsbaumes erläutert, wie z.B. zulässige Tests und Lernmengen. Ein allgemeines Verfahren zur Induktion von Entscheidungsbäumen, der Algorithmus ID3, wird ebenfalls vorgestellt. Das Unterkapitel 3.2 überträgt die bereits eingeführten Begriffe auf Fuzzy-Entscheidungsbäume. Es werden die wesentlichen Unterschiede und Ähnlichkeiten zwischen den klassischen und unscharfen Entscheidungsbäumen erläutert. Die Unterkapitel 3.3 und 3.4 beschreiben die zum Aufbau eines Baumes benötigten Methoden zur Auswahl des Tests an einem Knoten bzw. die Funktion zum Testen der Abbruchbedingungen. Der Algorithmus IFDT *zum nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen wird in Unterkapitel 3.5 vorgestellt.*

3.1 Klassische Entscheidungsbäume

Ein Entscheidungsbaum ist definiert für eine Menge von zulässigen Tests und eine Menge von Klassen. Jeder innere Knoten eines Entscheidungsbaumes ist durch einen Test gekennzeichnet, der festlegt, unter welchen Voraussetzungen die Beispiele zu den Nachfolgerknoten verzweigt werden. Mit diesen Bedingungen werden die ausgehenden Kanten eines inneren Knotens beschriftet. Jeder Blattknoten repräsentiert eine Klasse.

In Abbildung 3.1 sind zwei Entscheidungsbäume dargestellt, die für die gleichen Attribute und Klassen definiert sind, sich aber in der Art der verwendeten Tests unterscheiden.

Einmal aufgestellt, kann ein Entscheidungsbaum zur Klassifikation von neuen, unbekannten Beispielen verwendet werden. Um ein Beispiel zu klassifizieren, wird beginnend mit dem Wurzelknoten des Baumes genau einem Pfad bis zu einem Blattknoten gefolgt. Dabei

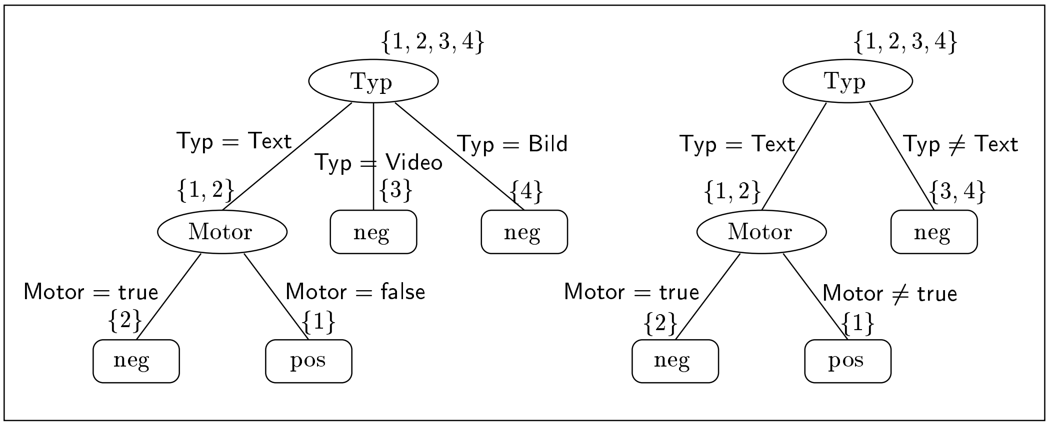

Abbildung 3.1: Beispiel zweier Entscheidungsbäume

wird an jedem inneren Knoten der Kante gefolgt, deren Bedingung durch das Beispiel erfüllt ist. Das Beispiel wird zu der Klasse zugeordnet, mit welcher der erreichte Blattknoten beschriftet ist.

Um einen Entscheidungsbaum zu generieren, benötigen wir eine Menge von klassifizierten Gegenständen, so genannte Lernmenge $\mathcal{L}$ mit $m \in \mathbb{N}$ Beispielen (Gegenstände). Seien die Beispiele in $\mathcal{L}$ durchnumeriert, so können wir jedes Beispiel durch seinen Index repräsentieren. Somit läßt sich $\mathcal{L}$ als die Menge:

$$\{1, 2, \ldots, m\}$$

darstellen.

Die Beispiele einer Lernmenge lassen sich durch

- eine endliche Menge von Attributen – die Attributmenge
$$A = \{a_1, a_2, \ldots, a_p\}$$
- eine endliche Menge von Attributwerten für jedes Attribut $a_i \in A$ – die Attributwertmenge
$$V_i = \{\mathrm{v}_{i1}, \mathrm{v}_{i2}, \ldots, \mathrm{v}_{ij_i}\}$$
- eine endliche Menge von Klassen – die Klassenmenge
$$\mathcal{C} = \{c_1, c_2, \ldots, c_q\}$$

beschreiben.

Ein Beispiel f ist durch jeweils einen Attributwert $\mathrm{v}_{il} \in V_i$ für jedes Attribut $a_i \in A$ und eine Klasse $c_k \in \mathcal{C}$ spezifiziert.

In der Tabelle 3.1 ist eine Lernmenge gegeben, die konsistent ist, d.h. es existieren keine zwei Beispiele mit gleichen Attributwerten, aber unterschiedlichen Klassen. Eine inkonsistente Lernmenge würde entstehen, wenn wir zu der Menge aus Tabelle 3.1 ein zusätzliches Beispiel einfügen mit denselben Attributwerten wie z. B. das Beispiel mit dem Index 4, aber mit der Klasse pos.

Inkonsistente Lernmengen treten oft auf, wenn die Lernmenge keine künstlich aufgestellten Beispiele, sondern Beispiele aus der realen Welt enthält. Die Fehler bei Daten aus der

Index	a_1 Technik	a_2 Motor	a_3 Typ	a_4 Größe	Klasse
1	true	false	Text	klein	pos
2	true	true	Text	mittel	neg
3	false	false	Video	groß	neg
4	true	true	Bild	klein	neg

Tabelle 3.1: Beispiel für eine konsistente Lernmenge

realen Welt sind oft durch physikalische Meßfehler, subjektive Bewertung oder dadurch verursacht, daß nicht alle relevanten Attribute erfaßt sind. Solche Werte nennt man verrauschte Werte (*noisy*). Handelt es sich bei der Lernmenge um Beispiele aus der realen Welt, so kann es auch vorkommen, daß unvollständige Beispiele enthalten sind, d.h. für diese Beispiele sind einige Attributwerte nicht bekannt (*missing attribute*). Um erfolgreich eine Bauminduktion durchführen zu können, müssen wir diese fehlenden Attributwerte ergänzen, wofür verschiedene Vorgehensweisen bekannt sind (siehe [Jan98, Zei99]). Solche Verfahren werden wir später in Unterkapitel 3.3 näher betrachten.

Anhand der Lernmenge (d.h. die Attributmenge A und die Attributwertmengen V_i jedes Attributs $a_i \in A$) können für jedes Attribut die zulässigen Tests bestimmt werden, welche wir in der Menge $\mathcal{T}$ zusammenfassen.

Definition 3.1 *Ein* **zulässiger Test** *t bezieht sich auf ein Attribut $a_i \in A$ und ist durch eine Folge von Bedingungen*

$$t = [cond_1, cond_2, \dots, cond_{u_t}]$$

spezifiziert[1].

Bezüglich dem Attribut $a_i \in A$ mit den Werten $\mathrm{v}_{il} \in V_i, 1 \leq l \leq j_i$ wird genau ein **mehrfacher Test** *gebildet, wobei die Anzahl seiner Bedingungen genau der Anzahl der Attributwerte von a_i entspricht. Jede Bedingung $cond_h$ bezieht sich auf einen anderen Attributwert v_{il} und hat die Form:*

$$a_i = \mathrm{v}_{il}$$

Diese Bedingung wird von alle Beispielen erfüllt, die den Wert v_{il} haben.

Dagegen wird bezüglich dem gleichen Attribut für jeden Attributwert $\mathrm{v}_{il} \in V_i, 1 \leq l \leq j_i$ ein **binärer Test** *(insgesamt j_i Tests) gebildet, wobei die Anzahl der Bedingungen genau zwei ist. Die eine Bedingung hat die gleiche Form wie oben:*

$$a_i = \mathrm{v}_{il}$$

und die andere:

$$a_i \neq \mathrm{v}_{il}$$

Die zweite Bedingung wird von allen Beispielen erfüllt, die den Wert v_{il} nicht haben.

Anhand des Entscheidungsbaumes aus Abbildung 3.1 und der Lernmenge aus Tabelle 3.1 können wir nachvollziehen, wie die Tests gebildet sind und wie die Beispiele in Abhängigkeit von diesen Tests verzweigt werden. Dabei ist der linke Baum für mehrfache Tests, der rechte für binäre Tests definiert.

[1] Bei der gleichzeitigen Verwendung mehrerer Tests kann der Test als oberer Index zu jeder Bedingung notiert werden, damit die Bedingungen verschiedener Tests besser unterschieden werden können. Damit wird der Test t wie folgt bezeichnet: $t = [cond_1^t, cond_2^t, \dots, cond_{u_t}^t]$.

Durch die Beschriftung eines inneren Knotens mit einem Test werden implizit die Anzahl seiner Nachfolgerknoten, sowie die Voraussetzungen zum Erreichen dieser Knoten, festgelegt. Bezeichnen wir einen inneren Knoten[2] mit n, den Test mit $t = [cond_1^t, cond_2^t, \ldots, cond_{u_t}^t]$. Dann gilt:

- der Knoten n hat genau u_t Nachfolger
- der h-te Nachfolger von n bezeichnen wir durch: $n.cond_h^t$
- die Menge aller Nachfolgerknoten von n bezeichnen wir mit:

$$\mathcal{N}(n) = \{n.cond_1^t, \ldots, n.cond_h^t, \ldots, n.cond_{u_t}^t\}$$

Nachdem wir die formale Definition einer Lernmenge, eines Beispiels und eines Tests eingeführt haben, geben wir hier die Definition eines Entscheidungsbaumes nach [Zei99] an.

Definition 3.2 *Ein* **Entscheidungsbaum** T *bezüglich einer Klassenmenge* $\mathcal{C}$ *und einer Menge von zulässigen Tests* $\mathcal{T}$ *ist ein Baum* $T = (N, E, L)$ *mit der Knotenmenge* N, *der Kantenmenge* $E \subseteq N \times N$ *und einer Beschreibungsfunktion* L, *die die Beschriftung jedes Knotens und jeder Kante des Baumes bestimmt, wobei nachfolgenden Bedingungen erfüllt sind:*

1. *Für jeden Knoten* $n \in N$ *gilt:*

$$L(n) = \begin{cases} t \in \mathcal{T} & \text{wenn } n \text{ ein innerer Knoten ist} \\ c_k \in \mathcal{C} & \text{sonst} \end{cases}$$

2. *Für jede Kante* $e = (n, n.cond_h^t) \in E$ *gilt:* $L(e) = cond_h^t$
3. *Je zwei voneinander verschiedene Knoten* n', n *entlang eines Weges von der Wurzel zu einem Blattknoten haben verschiedene Bezeichnungen, d.h.* $L(n') \neq L(n)$.

Das allgemeine Verfahren zum Aufbau eines Entscheidungsbaumes ist TDIDT (*Top Down Induction of Decision Trees*). Die Induktion beginnt am Wurzelknoten des Entscheidungsbaumes mit allen Beispielen aus der gegebenen Lernmenge $\mathcal{L}$ und mit der Menge der verfügbaren Tests $\mathcal{T}' = \mathcal{T}$ ($\mathcal{T}$ enthält alle zulässigen Tests, die anhand der Lernmenge gebildet werden können). Rekursiv wird für jeden Knoten untersucht: Ist die Funktion $\mathcal{F}$, die testet, ob eine der festgelegten Abbruchbedingungen zutrifft, erfüllt, dann wird der aktuelle Knoten als Blattknoten definiert und gemäß $\mathcal{F}$ beschriftet. Ansonsten wird mit Hilfe einer ebenso am Anfang der Induktion festgelegten Methode $\mathcal{M}$ ein Test $t \in \mathcal{T}' \subseteq \mathcal{T}$ ausgewählt, womit der aktuelle Knoten beschriftet wird. Dieser Test wird aus der Menge der verfügbaren Tests $\mathcal{T}'$ dieses Pfades entfernt. In Abhängigkeit von den Bedingungen des Tests t werden ausgehende Kanten und Nachfolgerknoten erzeugt. Die Beispiele, die den aktuellen Knoten erreicht haben, werden zu seinen Nachfolgern verzweigt. Da jedes Beispiel genau eine Bedingung des Tests t erfüllt, ist an jedem Nachfolgerknoten eine Teilmenge der ursprünglichen Lernmenge vertreten. Diese Teilmengen sind disjunkt.

Wie man sieht, werden die Methode $\mathcal{M}$ und die Funktion $\mathcal{F}$ am Anfang der Induktion für ihren Dauer festgelegt. Dagegen werden die Lernmenge $\mathcal{L}$ und die Menge $\mathcal{T}'$ mit jedem rekursiven Aufruf verändert.

[2] Den Wurzelknoten bezeichnen wir durch *root*.

In Abhängigkeit von den ausgewählten $\mathcal{M}$ und $\mathcal{F}$ kann man ausgehend aus einer Lernmenge unterschiedliche Entscheidungsbäume erstellen. Das Ziel ist es, einen möglichst kleinen Baum zu generieren, der alle Beispiele korrekt klassifiziert. Die Größe eines Baumes kann durch verschiedene Kriterien bestimmt werden: wie z.B. die Anzahl der Blattknoten, die Anzahl der Kanten im Baum, die Höhe des Baumes u.s.w. Als Qualitätskriterien für einen optimalen Entscheidungsbaum spielen nach [Min89] neben der Größe auch die Genauigkeit und die Verständlichkeit eine Rolle. Um die Eigenschaft des Baumes, unbekannte Beispielen korrekt zu klassifizieren, testen und bewerten zu können, wird die ursprüngliche Beispielmenge in zwei Teilmengen zerlegt. Man bezeichnet diese als Lern- und Testmenge. Die Lernmenge dient zur Induktion des Baumes. Wenn der Baum entstanden ist, werden die Beispiele der Testmenge klassifiziert und die Fehlerrate bestimmt.

Um einen möglichst kleinen Entscheidungsbaum zu generieren, muß eine solche Methode $\mathcal{M}$ eingesetzt werden, die einen Test auswählt, durch den die Lernmenge in möglichst homogene Teilmengen zerlegt wird. So ein Test wird oft durch statistische Methoden ausgewählt. Solche Methoden $\mathcal{M}$ werden wir später in Unterkapitel 3.3 vorstellen.

Die Baumgröße kann während der Induktion mittels der Funktion $\mathcal{F}$, aber auch nach der Bauminduktion durch das so genannte Baumbeschneiden optimiert werden. Ziel des Baumbeschneidens ist es, die Überanpassung (*overfitting*) eines Entscheidungsbaumes zu beseitigen. Overfitting kommt vor, wenn ein Teilbaum mehrere Beispiele einer Klasse zuordnet (es wäre möglich ihn als Blattknoten zu definieren) und ein oder sehr wenig Beispiele zu einer anderen Klasse zuordnet. Bei dem Baumbeschneiden wird dann der Teilbaum durch einen Blattknoten ersetzt. Dadurch wird vermieden, daß der Entscheidungsbaum zu genau lernt, seine Größe wird verringert und seine Interpretierbarkeit wird erhöht. Die Klassifikation wird kaum beeinträchtigt. Um zu bestimmen, ob ein Zweig des Entscheidungsbaumes beschnitten werden kann/muß, können verschiedenen heuristischen Verfahren eingesetzt werden ([UBC97, RN95]).

Ein Vertreter des Verfahrens *Top Down Induction of Decision Trees* ist der von J.R. Quinlan (siehe [Qui86]) entwickelte Algorithmus ID3. Die Methode zur Auswahl des besten Tests $\mathcal{M}$ sucht nach dem Test, welcher den größten Informationsgewinn - gemessen in *bits* - hat. Die Funktion $\mathcal{F}$ testet nach zwei Abbruchskriterien und in Abhängigkeit davon, welche Bedingung erfüllt ist, wird der als Blattknoten definierte Knoten wie folgt beschriftet:

- wenn die Beispiele an einem Knoten zur selben Klasse gehören, dann wird er mit der entsprechenden Klasse beschriftet

- wenn die Menge der verfügbaren Tests $\mathcal{T}'$ an einem Knoten leer ist, dann wird der Blattknoten mit der Klasse beschriftet, zu der die Mehrheit der Beispiele gehören

Die Idee dieses Verfahrens wird mit dem Konzept der Fuzzy-Logik und Fuzzy-Mengen kombiniert und zu einem Verfahren zur Induktion von unscharfen Entscheidungsbäume erweitert.

In nächstem Unterkapitel werden wir die Idee eines unscharfen Entscheidungsbaumes vorstellen, sowie die Ähnlichkeiten und die Unterschieden zu einem klassischen Baum erläutern. Ein konkretes Verfahren zum Aufbau von Fuzzy-Entscheidungsbäumen wird in Unterkapitel 3.5 beschrieben.

3.2 Grundbegriffe eines Fuzzy-Entscheidungsbaumes

Ein Fuzzy-Entscheidungsbaum ist genauso wie ein klassischer Entscheidungsbaum für eine Menge von zulässigen Tests und eine Menge von Klassen, aber unter Berücksichtigung der Theorie der Fuzzy-Logik, definiert.

Jeder innere Knoten eines unscharfen Entscheidungsbaumes ist ebenfalls durch einen Test gekennzeichnet; die davon ausgehenden Kanten sind mit je einer Bedingung dieses Tests beschriftet und die Klassen sind durch die Blattknoten repräsentiert. Im Gegensatz zu den klassischen Entscheidungsbäumen, wo an einem Blattknoten genau eine Klasse wiederzufinden ist, können bei den unscharfen Entscheidungsbäumen an einem Blattknoten mehrere Klassen mit einer Klassenzugehörigkeit aus dem Intervall $[0, 1]$ repräsentiert werden. Somit ist jeder Blattknoten mit einem Tupel von Klassenzugehörigkeiten beschriftet.

Ein Fuzzy-Entscheidungsbaum bezüglich der Klassen pos und neg ist in der Abbildung 3.2 dargestellt. Jeder Blattknoten ist mit einem Tupel von Klassenzugehörigkeiten beschriftet, wobei als erste die Zugehörigkeit der Klasse pos notiert ist.

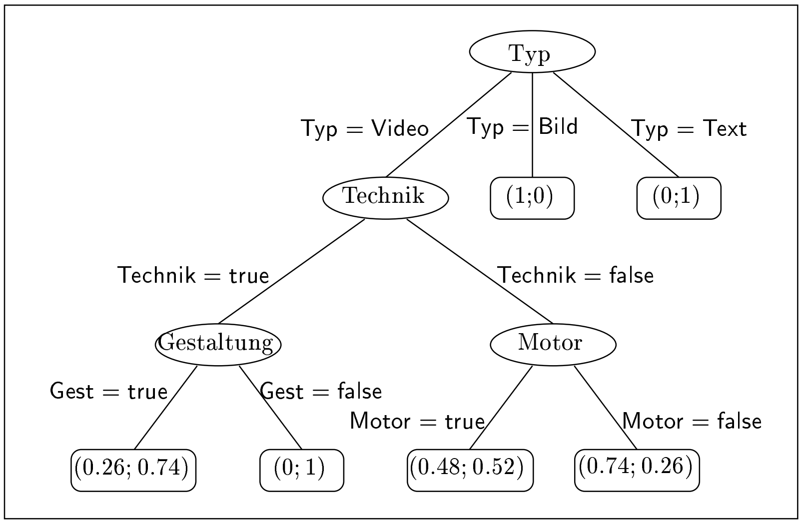

Abbildung 3.2: Beispiel eines Fuzzy-Entscheidungsbaumes

Bei der Klassifikation von neuen Beispielen mittels eines Fuzzy-Entscheidungsbaumes werden, ähnlich wie bei den klassischen Bäumen, an jedem inneren Knoten die Bedingungen des Tests betrachtet. Der Unterschied besteht darin, daß ein Beispiel mit verschiedenen Zugehörigkeitsgraden mehrere Bedingungen eines Tests erfüllen kann. Um dies auszudrücken, wird für jedes Beispiel ein Zugehörigkeitsgrad zu jedem Knoten des Baumes definiert.

Wir untersuchen das Durchlaufen des Entscheidungsbaumes entlang eines Weges von der Wurzel zu einem Blattknoten. Betrachten wir diesen Pfade als eine Folge von Bedingungen, die auf den durchlaufenden Kanten stehen. Die Zugehörigkeitsgrade, mit denen das Beispiel die Bedingungen erfüllt, werden durch eine t-Norm miteinander verknüpft. Das Ergebnis bezeichnen wir als Zugehörigkeitsgrad des Beispiels zu dem erreichten Blattknoten. Das Durchlaufen eines Baumes endet, nachdem alle Pfade des Baumes untersucht wurden. Für jeden Blattknoten wurde dabei ein Zugehörigkeitsgrad des Beispiels zu dem Knoten ausgerechnet und ein Tupel von Klassenzugehörigkeiten zugeordnet. Der Zugehörigkeitsgrad des Beispiels wird mittels einer t-Norm mit jedem Wert aus dem Tupel verknüpft. Das so erhaltene Tupel drückt die Klassenzugehörigkeiten des Beispiels bezüglich des Blattknotens aus.

Die oben bestimmten Tupel für die Klassenzugehörigkeiten des Beispiels an jedem Blattknoten werden elementweise durch eine t-Conorm verknüpft und als Ergebnis erhalten wir ein Tupel von Klassenzugehörigkeiten für das Beispiel bezüglich des Fuzzy-Entscheidungsbaumes. Ist deren Summe größer als Eins, so wird eine Normierung durchgeführt. Wollen wir unser unbekanntes Beispiel zu einer einzigen Klasse zuordnen, können wir eine Defuzzifizierung durchführen, z.B. wird die Klasse mit der größten Zugehörigkeit ausgewählt.

Definition 3.3 *(nach [Zei99]) Ein* **Fuzzy-Entscheidungsbaum** *FT bezüglich einer Klassenmenge $\mathcal{C}$ und einer Menge von zulässigen Tests $\mathcal{T}$ ist ein Baum $FT = (N, E, L)$ mit der Knotenmenge N, der Kantenmenge $E \subseteq N \times N$ und einer Beschreibungsfunktion L, die die Beschriftung jedes Knotens und jeder Kante des Baumes bestimmt, wobei die folgenden Bedingungen erfüllt sind:*

1. *Für jeden Knoten $n \in N$ gilt:*

$$L(n) = \begin{cases} t \in \mathcal{T} & \text{wenn } n \text{ ein innerer Knoten ist} \\ (\mu_k)_{k=1,2,\ldots,q} \in ([0,1])^q & \text{sonst} \end{cases}$$

2. *Für jede Kante $e = (n, n.cond_h^t) \in E$ gilt: $L(e) = cond_h^t$*

3. *Je zwei voneinander verschiedene Knoten n', n entlang eines Weges von der Wurzel zu einem Blattknoten haben verschiedene Bezeichnungen, d.h. $L(n') \neq L(n)$.*

Der Aufbau von Fuzzy-Entscheidungsbäumen setzt, genauso wie die Induktion von klassischen Entscheidungsbäumen, voraus, daß eine Lernmenge $\mathcal{L}$ mit m Beispielen gegeben ist. Im klassischen Fall ist ein Beispiel durch jeweils einen Attributwert $\mathrm{v}_{il} \in V_i$ für jedes Attribut $a_i \in A$ und eine Klasse $c_k \in \mathcal{C}$ spezifiziert. In Bezug auf die Fuzzy-Entscheidungsbäume werden die Attribute und die Klasse als linguistische Variablen angesehen, deren Werte als linguistische Werte. Somit entspricht beispielsweise die Attributwertmenge $V_i = \{\mathrm{v}_{i1}, \mathrm{v}_{i2}, \ldots, \mathrm{v}_{ij_i}\}$ für das Attribut a_i genau der Menge der linguistischen Werte der linguistischen Variable a_i, was wir mit $T(a_i) = \{term_{i1}(a_i), term_{i2}(a_i), \ldots, term_{ij_i}(a_i)\}$ bezeichnet haben. Die Klassen sind ebenfalls als linguistische Werte einer Variable definiert.

Jedes Beispiel f aus $\mathcal{L}$ ist dann durch genau einen Eingabewert x_i^f für jede linguistische Variable $a_i \in A$ und einen Eingabewert für die linguistische Variable, die die Klassen beschreibt, spezifiziert:

$$(x_1^f, x_2^f, \ldots, x_p^f, class^f)$$

Da es sich um linguistische Variablen handelt, muß jeder Eingabewert x_i^f fuzzifiziert werden. Somit erhalten wir für jede linguistische Variable $a_i \in A$ und jedes Beispiel f ein Tupel von Zugehörigkeitsgraden:

$$(\xi_{term_{i1}(a_i)}(x_i^f), \xi_{term_{i2}(a_i)}(x_i^f), \ldots, \xi_{term_{ij_i}(a_i)}(x_i^f))$$

und ein Tupel von Klassenzugehörigkeiten:

$$(\xi_{c_1}(class^f), \xi_{c_2}(class^f), \ldots, \xi_{c_q}(class^f))$$

Die Induktion eines Fuzzy-Entscheidungsbaumes führen wir nicht mit den oben ausgerechneten Tupeln durch, sondern anhand der folgenden Fuzzy-Mengen, die wir über die Elemente der Lernmenge $\mathcal{L}$ definieren:

- für jeden linguistischen Wert $term_{il}(a_i), 1 \leq l \leq j_i$ jedes Attributs $a_i \in A$ definieren wir die Fuzzy-Menge
$$\mu_{\mathrm{v}_{il}} : \mathcal{L} \to [0,1]$$
mit $\mu_{\mathrm{v}_{il}}(f) = \xi_{term_{il}(a_i)}(x_i^f)$ für jedes Beispiel $f \in \mathcal{L}$

- für jede Klasse $c_k \in \mathcal{C}$ definieren wir die Fuzzy-Menge
$$\mu_{c_k} : \mathcal{L} \to [0,1]$$
mit $\mu_{c_k}(f) = \xi_{c_k}(class^f)$ für jedes Beispiel $f \in \mathcal{L}$

Damit ist jedes Beispiel $f \in \mathcal{L}$ durch genau einen Zugehörigkeitsgrad $\mu_{\mathrm{v}_{il}}(f)$ zu jeder der Fuzzy-Mengen $\mu_{\mathrm{v}_{il}}$ und durch einen Zugehörigkeitsgrad $\mu_{c_k}(f)$ zu jeder der Fuzzy-Mengen μ_{c_k} spezifiziert. Solche Beispiele nennen wir Fuzzy-Beispiel, eine Lernmenge, die Fuzzy-Beispiele enthält, Fuzzy-Lernmenge.

Annahme 3.4 *Für die weitere Arbeit wird gefordert, daß für jedes Beispiel f gilt:*

$$\sum_{l=1}^{j_i} \mu_{\mathrm{v}_{il}}(f) = 1 \tag{3.1}$$
$$\sum_{k=1}^{q} \mu_{c_k}(f) = 1 \tag{3.2}$$

Tabelle 3.2 stellt eine Lernmenge dar, wobei die Beispiele durch scharfe Eingabewerte gegeben sind.

Index	a_1 Technik	a_2 Motor	a_3 Innenraum	a_4 Gestaltung	a_5 Typ	a_6 Größe	Klasse
1	0.7	0.5	0.9	0.7	Video	6	neg
2	0.3	0.3	0.8	0.8	Video	12	pos
3	0.8	0.6	0.1	0.1	Text	17	neg
4	0.6	0.8	0.5	0.6	Bild	4	pos
5	0.9	0.8	0.3	0.2	Video	15	neg
6	0.4	0.3	0.6	0.7	Bild	10	pos

Tabelle 3.2: Lernmenge mit scharfen Eingabewerten

Nach der Fuzzifizierung der Eingabewerte (z.B. die linguistische Variable Größe ist in Beispiel 2.7 auf Seite 11 vorgestellt) erhalten wir die Fuzzy-Beispiele aus der Tabelle 3.3. Eine grafische Darstellung der Fuzzy-Menge μ_{mittel} bietet die Abbildung 3.3.

Genauso wie im klassischen Fall wird die Menge der zulässigen Tests $\mathcal{T}$ anhand der Lernmenge, d.h. der linguistischen Variablen und deren Werte erzeugt, wobei für jede linguistische Variable die zulässigen Tests gebildet werden. Da jeder Test auf eine linguistische Variable basiert, ist es möglich, daß ein Beispiel mehrere Bedingungen eines Tests mit unterschiedlichen Zugehörigkeitsgraden erfüllt.

Definition 3.5 *Sei eine Bedingung $cond_h^t$ eines Tests t gegeben. Weiterhin beziehe sich der Test t auf die linguistische Variable a_i, und die Bedingung $cond_h^t$ auf einen der Werte $term_{il}(a_i)$.*

Der Erfüllungsgrad eines Beispiels $f \in \mathcal{L}$ zu der Bedingung $cond_h^t$ wird wie folgt ausgerechnet:

$$\mu_{cond_h^t}(f) = \begin{cases} \mu_{\mathrm{v}_{il}}(f) & \text{wenn die Bedingung die Form } a_i = \mathrm{v}_{il} \text{ hat} \\ 1 - \mu_{\mathrm{v}_{il}}(f) & \text{wenn die Bedingung die Form } a_i \neq \mathrm{v}_{il} \text{ hat} \end{cases}$$

Korollar 3.6 *Sei ein Test $t = [cond_1^t, cond_2^t, \dots, cond_{u_t}^t]$ gegeben. Unter Annahme 3.4 auf Seite 19 gilt für jedes Beispiel $f \in \mathcal{L}$:*

$$\sum_{h=1}^{u_t} \mu_{cond_h^t}(f) = 1$$

Beispiel 3.7 *Bezüglich der linguistischen Variable* Größe *werden die binären Tests:*
$t_1 = [\mathsf{Groeße} = \mathsf{klein}, \mathsf{Groeße} \neq \mathsf{klein}]$ $t_2 = [\mathsf{Groeße} = \mathsf{mittel}, \mathsf{Groeße} \neq \mathsf{mittel}]$
$t_3 = [\mathsf{Groeße} = \mathsf{groß}, \mathsf{Groeße} \neq \mathsf{groß}]$
und der mehrfache Test $t = [\mathsf{Groeße} = \mathsf{klein}, \mathsf{Groeße} = \mathsf{mittel}, \mathsf{Groeße} = \mathsf{groß}]$ *erzeugt.*

Das Beispiel 2 aus Tabelle 3.3 erfüllt z.B. die Bedingung Groeße = mittel *mit dem Zugehörigkeitsgrad 0.53, die Bedingung* Groeße $\neq$ mittel *mit 0.47.*

Der Aufbau eines Fuzzy-Entscheidungsbaumes verläuft analog zu dem Aufbau eines klassischen Entscheidungsbaum (beschrieben auf Seite 15). Der Unterschied besteht darin, daß bei dem klassischen Fall jedes Beispiel genau eine – bei dem Fuzzy-Fall mehrere – Bedingung eines Tests erfüllen kann. Deswegen werden bei der Verzweigung an jedem Knoten keine disjunkten Teilmengen von Beispielen erzeugt, sondern wir betrachten an jedem Knoten n die gesamte Fuzzy-Lernmenge $\mathcal{L}$, wobei jedes Beispiel aus $\mathcal{L}$ unterschiedlich stark zum Knoten n gehört. Um diese Zuordnung auszudrücken, führen wir den Begriff Zugehörigkeitsgrad des Beispiels zu einem Knoten n und die Fuzzy-Menge λ_n über die Beispiele aus $\mathcal{L}$ ein:

$$\lambda_n : \mathcal{L} \to [0,1]$$

wobei für den Wurzelknoten *root* gilt:

$$\forall f \quad \lambda_{root}(f) = 1$$

Die Induktion beginnt am Wurzelknoten mit der Fuzzy-Menge λ_{root}, d.h. jedes Beispiel gehört zu der Wurzel mit dem Zugehörigkeitsgrad Eins. Dieser Zugehörigkeitsgrad nimmt während der Bearbeitung entlang eines Pfades des Baumes ab. Für jeden Nachfolgerknoten wird der neue Zugehörigkeitsgrad als die t-Norm zwischen dem aktuellen Zugehörigkeitsgrad und dem Zugehörigkeitsgrad, mit dem das Beispiel die entsprechende Bedingung erfüllt, gebildet.

Definition 3.8 *Sei der Knoten n mit dem Test $t = [cond_1^t, cond_2^t, \dots, cond_{u_t}^t]$ und die Fuzzy-Menge λ_n gegeben. Der Zugehörigkeitsgrad $\lambda_{n.cond_h^t}(f)$ eines Beispiels f zu einem Nachfolgerknoten $n.cond_h^t$ wird wie folgt ausgerechnet:*

$$\lambda_{n.cond_h^t}(f) = \top(\lambda_n(f), \mu_{cond_h^t}(f))$$

Einen Algorithmus zum Aufbau von Fuzzy-Entscheidungsbäumen werden wir in Unterkapitel 3.5 vorstellen. Dafür werden wir in den nächsten zwei Unterkapiteln einige Methoden $\mathcal{M}$ zur Auswahl eines Tests und eine Funktion $\mathcal{F}$ zum Testen der Abbruchbedingungen beschreiben.

Index	Technik		Motor		Innenraum		Gestaltung		Typ			Größe			Klasse	
	μ_{true}	μ_{false}	μ_{true}	μ_{false}	μ_{true}	μ_{false}	μ_{true}	μ_{false}	μ_{video}	μ_{bild}	μ_{text}	μ_{klein}	μ_{mittel}	$\mu_{\text{groß}}$	μ_{pos}	μ_{neg}
1	0.7	0.3	0.5	0.5	0.9	0.1	0.7	0.3	1	0	0	0	0.93	0.07	0	1
2	0.3	0.7	0.3	0.7	0.8	0.2	0.8	0.2	1	0	0	0	0.53	0.47	1	0
3	0.8	0.2	0.6	0.4	0.1	0.9	0.1	0.9	0	0	1	0	0.2	0.8	0	1
4	0.6	0.4	0.8	0.2	0.5	0.5	0.6	0.4	0	1	0	0.2	0.8	0	1	0
5	0.9	0.1	0.8	0.2	0.3	0.7	0.2	0.8	1	0	0	0	0.33	0.67	0	1
6	0.4	0.6	0.3	0.7	0.6	0.4	0.7	0.3	0	1	0	0	0.67	0.33	1	0

Tabelle 3.3: Fuzzy-Lernmenge nach der Fuzzifizierung

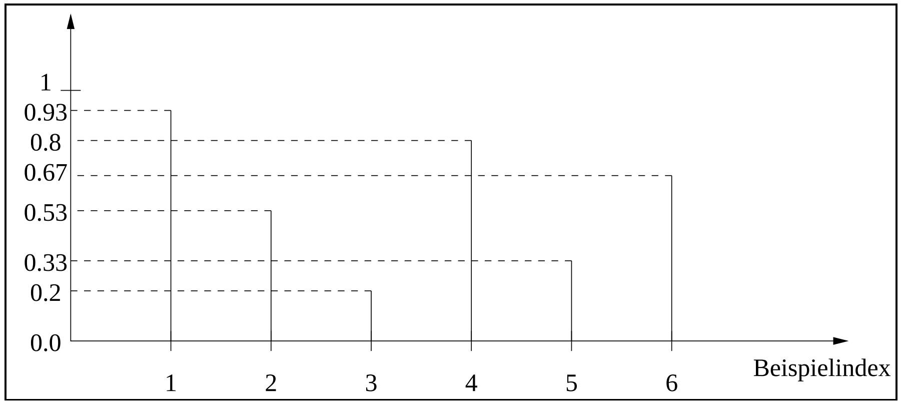

Abbildung 3.3: Fuzzy-Menge μ_{mittel}

3.3 Methoden zur Auswahl des besten Tests

Die Methode zur Auswahl eines Tests muß in der Lage sein deterministisch ein Test auszusuchen. Die meisten bekannten Methoden wählen den Test, welcher die größte Information – gemessen in $bits$ – über die Beispielmenge enthält. Information von $1bit$ genügt, um eine ja/nein Frage ohne zusätzliches Wissen zu beantworten. Je weniger man über die gestellte Frage weiß, desto höher ist der Informationsgehalt der Antwort. Sind die Wahrscheinlichkeiten p_i der möglichen Antworten gegeben, dann wird der Informationsgehalt der Antwort, die so genannte Entropie, wie folgt ausgerechnet:

Definition 3.9 *Sei $W = (p_1, p_2, \ldots, p_n)$ eine Wahrscheinlichkeitsverteilung*[3]*. Die zugehörige* **Entropie** *berechnet sich als*

$$I(W) = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$$

Geben wir einige Beispiele: sei $W = (0.5, 0.5)$, dann ist $I(W) = 1$, sei $W = (0.67, 0.33)$, dann ist $I(W) = 0.92$ und sei $W = (1, 0)$ dann ist $I(W) = 0$ — je gleichmäßiger die Wahrscheinlichkeitsverteilung ist, desto größer ist ihre Entropie.

In diesem Unterkapitel werden wir nur solche Methode betrachten, die auf dem Maß der Entropie basieren. Um so eine Methode $\mathcal{M}$ beschreiben und anwenden zu können, benötigen wir an jedem Knoten n den Durchschnitt nach Definition 2.5 auf Seite 10 zwischen den folgenden Fuzzy-Mengen:

- $\lambda_n \cap \mu_{c_k}$

$$\lambda_n|c_k : \mathcal{L} \to [0,1] \tag{3.3}$$
$$\forall f : \lambda_n|c_k(f) = \top(\lambda_n(f), \mu_{c_k}(f))$$

- $\lambda_n \cap \mu_{cond_h^t}$

$$\lambda_n|cond_h^t : \mathcal{L} \to [0,1] \tag{3.4}$$
$$\forall f : \lambda_n|cond_h^t(f) = \top(\lambda_n(f), \mu_{cond_h^t}(f))$$

- $\lambda_n \cap \mu_{cond_h^t} \cap \mu_{c_k}$

$$\lambda_n|cond_h^t|c_k : \mathcal{L} \to [0,1] \tag{3.5}$$
$$\forall f : \lambda_n|cond_h^t|c_k(f) = \top(\lambda_n(f), \mu_{cond_h^t}(f), \mu_{c_k}(f))$$

für jede Bedingung $cond_h^t$ jedes Tests $t \in \mathcal{T}$ und jede Klasse $c_k \in \mathcal{C}$.

Die Mächtigkeiten (die Größen) dieser Fuzzy-Mengen werden gemäß Definition 2.2 auf Seite 8 ausgerechnet.

Der Entscheidungsbaum muß die Frage nach der korrekten Klasse eines Beispiels beantworten. An einem Knoten n können wir die Verteilung der Beispiele über die Klassen wie folgt bestimmen:

$$(|\lambda_n|c_1|, |\lambda_n|c_2|, \ldots, |\lambda_n|c_q|)$$

[3] Für eine Wahrscheinlichkeitsverteilung gilt $0 \leq p_i \leq 1$ für alle $i = 1, 2, \ldots, n$ und $\sum_{i=1}^{n} p_i = 1$.

Die dazugehörige Wahrscheinlichkeitsverteilung bekommen wir nach einer Normierung:

$$W(\lambda_n) = (\frac{|\lambda_n|c_1|}{\Lambda_{n,\mathcal{C}}}, \frac{|\lambda_n|c_2|}{\Lambda_{n,\mathcal{C}}}, \dots, \frac{|\lambda_n|c_q|}{\Lambda_{n,\mathcal{C}}}) \tag{3.6}$$

mit

$$\Lambda_{n,\mathcal{C}} = \sum_{k=1}^{q} |\lambda_n|c_k| \tag{3.7}$$

Die Entropie der Verteilung mißt die benötigte Information, um ein Beispiel zu klassifizieren ohne die Begrenzung der Lernmenge mittels eines Tests.

Anhand jeder Bedingung $cond_h^t$ eines Tests $t = [cond_1^t, cond_2^t, \dots, cond_{u_t}^t] \in \mathcal{T}$, der an einem Knoten n gesetzt werden könnte, kann die Fuzzy-Menge $\lambda_n|cond_h^t$ bzw. $\lambda_{n.cond_h^t}$ ausgerechnet werden. Die sagt aus, wie stark jedes Beispiel an den Nachfolgerknoten von n vertreten würde, falls der Test t gesetzt würde. Damit erhalten wir die Verteilung der Beispiele gemäß eines Tests t:

$$(|\lambda_n|cond_1^t|, |\lambda_n|cond_2^t|, \dots, |\lambda_n|cond_{u_t}^t|)$$

Die dazugehörige Wahrscheinlichkeitsverteilung ergibt sich nach einer Normierung zu:

$$W(t, \lambda_n) = (\frac{|\lambda_n|cond_1^t|}{\Lambda_{n,t}}, \frac{|\lambda_n|cond_2^t|}{\Lambda_{n,t}}, \dots, \frac{|\lambda_n|cond_{u_t}^t|}{\Lambda_{n,t}}) \tag{3.8}$$

und

$$\Lambda_{n,t} = \sum_{h=1}^{u_t} |\lambda_n|cond_h^t| \tag{3.9}$$

Jedes Beispiel, das Element der Fuzzy-Mengen $\lambda_n|cond_h^t$ ist, gehört mit unterschiedlichen Zugehörigkeitsgraden zu mehreren Klassen. Analog zu der Gleichung (3.6) können wir die Verteilung der Beispiele aus der Fuzzy-Menge $\lambda_n|cond_h^t$ über die Klassen aufstellen:

$$(|\lambda_n|cond_h^t|c_1|, |\lambda_n|cond_h^t|c_2|, \dots, |\lambda_n|cond_h^t|c_q|)$$

Für die entsprechende Wahrscheinlichkeitsverteilung erhalten wir:

$$W(\lambda_n|cond_h^t) = (\frac{|\lambda_n|cond_h^t|c_1|}{\Lambda_{n,cond_h^t,\mathcal{C}}}, \frac{|\lambda_n|cond_h^t|c_2|}{\Lambda_{n,cond_h^t,\mathcal{C}}}, \dots, \frac{|\lambda_n|cond_h^t|c_q|}{\Lambda_{n,cond_h^t,\mathcal{C}}}) \tag{3.10}$$

mit

$$\Lambda_{n,cond_h^t,\mathcal{C}} = \sum_{k=1}^{q} |\lambda_n|cond_h^t|c_k| \tag{3.11}$$

Die Entropie dieser Verteilung drückt die benötigte Information aus, um ein Beispiel innerhalb einer der Fuzzy-Mengen $\lambda_n|cond_h^t$ zu klassifizieren.

Anhand der eingeführten Wahrscheinlichkeitsverteilungen und Begriffe können wir jetzt die konkreten Methoden zur Auswahl eines Tests beschreiben.

Informationsgewinn

Als Maß zur Bestimmung des besten Tests für die Baumzerlegung kann der Informationsgewinn IG (*information gain*) eingesetzt werden. Es wird der Test ausgewählt, der den größten Wert für den Informationsgewinn hat. Es kann vorkommen, daß mehrere Tests den gleichen Wert erhalten, d.h. sie sind gleich gut. Um durch die Methode $\mathcal{M}$ deterministisch den besten Test aussuchen zu können, wird künstlich eine Priorität (siehe [Höl01]) über die Tests angelegt, die eine lineare Ordnung[4] über die Tests bestimmt. Haben zwei Tests gleiche Werte bekommen, so wird der Test mit der höheren Priorität ausgewählt.

Der Informationsgewinn für den Test t an einem Knoten n entspricht der Differenz zwischen der Information, die man braucht, um ein Element aus $\mathcal{L}$ vor und nach der Teilung durch t zu klassifizieren. Die Information, die man braucht um ein Beispiel vor der Teilung zu klassifizieren, wird durch die Entropie gemessen; die Information nach der Teilung durch den Erwartungswert der Information nach Unterteilung der Lernmenge mittels Tests $t \in \mathcal{T}$:

$$EI(t, \lambda_n) = \sum_{h=1}^{u_t} \frac{|\lambda_n | cond_h^t|}{\Lambda_{n,t}} \cdot I(W(\lambda_n | cond_h^t)) \qquad (3.12)$$

Dies entspricht den Entropien der Fuzzy-Mengen $\lambda_n | cond_h^t$ gewichtet nach der relativen Größen der einzelnen Fuzzy-Mengen.

Damit ergibt sich der Informationsgewinn eines Tests t für einen Knoten n zu

$$IG(t, \lambda_n) = I(W(\lambda_n)) - EI(t, \lambda_n) \qquad (3.13)$$

Der Test mit dem größten Informationsgewinn hat den geringsten Erwartungswert der Information nach der Unterteilung, also wird für die Bestimmung der Klasse eines Beispiels weniger Information benötigt – resultierend in weniger Testabfragen.

Informationsgewinnverhältnis

Wird zur Bestimmung des besten Tests den Informationsgewinn benutzt, werden diejenigen Tests vorgezogen, die viele verschiedene Werte annehmen können. Wird so ein Test für die Zerlegung ausgewählt, dann könnte der Entscheidungsbaum sehr stark in die Breite wachsen. Um dieses Problem zu vermeiden, wird eine andere Methode zur Auswahl eines Tests benutzt – das Informationsgewinnverhältnis (*gain ratio*):

$$IGR(t, \lambda_n) = \frac{IG(t, \lambda_n)}{I(W(t, \lambda_n))} \qquad (3.14)$$

Das starke Breitenwachstum wird vermieden, weil bei zwei Attributen mit gleichem Informationsgewinn, jenes Attribut bevorzugt wird, das weniger Verzweigungen in der Breite verursacht.

Man sucht nach dem Test mit dem größten Informationsgewinnverhältnis unter Berücksichtigung der festgelegten Priorität.

[4] Damit wird vermieden, daß zwei Tests die gleiche Priorität haben.

Gewichtete Wahrscheinlichkeitsverteilung über die Klassen an den Nachfolgerknoten

Die Idee dieses Verfahrens ist, bei der Berechnung der Erwartungswert der Information nach der Unterteilung mittels eines Tests t an einem Knoten n, die Verteilung der Beispiele über die Klassen an den Nachfolger von n gleich zu behandeln. Für diese Ziel, wird der Wert $\frac{|\lambda_n|cond_h^t|c_k|}{\Lambda_{n,cond_h^t,\mathcal{C}}}$ für jede Klasse c_k aus der Wahrscheinlichkeitsverteilung (3.10) gewichtet. Das Gewicht $\mathcal{W}(c_k)$ für jede Klasse $c_k \in \mathcal{C}$ wird aus der Wahrscheinlichkeitsverteilung (3.6) am Knoten n wie folgt ausgerechnet:

$$\mathcal{W}(c_k) = \frac{\Lambda_{n,\mathcal{C}}}{|\lambda_n|c_k|} \tag{3.15}$$

Dann ergibt sich die Verteilung über die Klassen an einem Nachfolgerknoten von n zu:

$$(\mathcal{W}(c_1) \cdot \frac{|\lambda_n|cond_h^t|c_1|}{\Lambda_{n,cond_h^t,\mathcal{C}}}, \mathcal{W}(c_2) \cdot \frac{|\lambda_n|cond_h^t|c_2|}{\Lambda_{n,cond_h^t,\mathcal{C}}}, \ldots, \mathcal{W}(c_q) \cdot \frac{|\lambda_n|cond_h^t|c_q|}{\Lambda_{n,cond_h^t,\mathcal{C}}})$$

Nach einer Normierung dieses Tupels erhalten wir die gewichtete Wahrscheinlichkeitsverteilung, bezeichnet als $W(\lambda_n|cond_h^t)_{\mathcal{W}}$. Diese Verteilung wird in Gleichung (3.12) zur Berechnung des Erwartungswerts der Information nach Unterteilung mittels des Tests t anstatt die Verteilung $W(\lambda_n|cond_h^t)$ eingesetzt:

$$EI(t, \lambda_n)_{\mathcal{W}} = \sum_{h=1}^{u_t} \frac{|\lambda_n|cond_h^t|}{\Lambda_{n,t}} \cdot I(W(\lambda_n|cond_h^t)_{\mathcal{W}}) \tag{3.16}$$

Der so berechnete Erwartungswert der Information nach der Unterteilung mittels eines Tests t wird zur Berechnung des Informationsgewinn nach Gleichung (3.13) verwendet:

$$IG(t, \lambda_n)_{\mathcal{W}} = I(W(\lambda_n)) - EI(t, \lambda_n)_{\mathcal{W}} \tag{3.17}$$

Man kann nach dem Test mit dem größten Informationsgewinn $IG(t, \lambda_n)_{\mathcal{W}}$ bzw. dem größten Informationsgewinnverhältnis $IGR(t, \lambda_n)_{\mathcal{W}} = \frac{IG(t,\lambda_n)_{\mathcal{W}}}{I(W(t,\lambda_n))}$ unter Berücksichtigung der festgelegten Priorität suchen.

Behandlung von unvollständigen Beispielen nach [Jan98]

Wenn für ein Beispiel f kein Eingabewert x_i^f für die linguistische Variable a_i gegeben ist, dann können wir das Tupel von Zugehörigkeitsgraden dieser Variable

$$(\mu_{\mathrm{v}_{i1}}(f), \mu_{\mathrm{v}_{i2}}(f), \ldots, \mu_{\mathrm{v}_{ij_i}}(f))$$

nicht ausrechnen. Ebenso können wir nicht bestimmen, zu welchem Grad eine Bedingung eines Tests t, der auf der linguistischen Variable a_i basiert, durch das Beispiel f erfüllt ist. Das Beispiel f benennen wir unvollständiges Beispiel bezüglich a_i bzw. t. Mit solchen Beispielen kann ein Entscheidungsbaum nicht induziert werden, da die oben beschriebenen Methoden zur Auswahl eines Tests nicht anwendbar sind.

Wir definieren einen Zugehörigkeitsgrad des unvollständigen Beispiels f bezüglich a_i zu jedem linguistischen Wert $\mathrm{v}_{il} \in T(a_i)$ als

$$\mu_{\mathrm{v}_{il}}(f) = \frac{1}{|T(a_i)|}$$

wobei $|T(a_i)|$ die Anzahl der linguistischen Werten für die Variable a_i ist.

Bei der Berechnung des Informationsgewinns nach Gleichung (3.13) für einen Test t, der auf eine Variable a_i basiert, müssen die unvollständigen Beispiele bezüglich t berücksichtigt werden. Der Informationsgewinn wird mit deren Anteil gewichtet:

$$IG(t, \lambda_n) = (1 - \frac{|\lambda_{n,t}^{missing}|}{|\lambda_n|})(I(W(\lambda_n)) - EI(t, \lambda_n)) \tag{3.18}$$

mit

$$\lambda_{n,t}^{missing}(f) = \begin{cases} \lambda_n(f) & \text{wenn } f \text{ unvollständiges Beispiel bezüglich } t \\ 0 & \text{sonst} \end{cases} \tag{3.19}$$

Damit können wir die Größe der Fuzzy-Menge $\lambda_{n,t}^{missing}$ nach Definition 2.2 auf Seite 8 ausrechnen.

Sind alle Beispiele in Bezug auf einen Test t vollständig, so ist die Fuzzy-Menge $\lambda_{n,t}^{missing} = \emptyset$ und ihre Größe ergibt sich zu Null. Damit erhalten wir den Informationsgewinn aus Gleichung (3.13).

3.4 Funktion zum Testen der Abbruchbedingungen

Jedes Verfahren zum Aufbau von Entscheidungsbäumen wird mit einer Funktion $\mathcal{F}$ ausgeführt. Jede solche Funktion $\mathcal{F}$ ist für eine Menge von Abbruchbedingungen und zulässige Blattbeschriftungen definiert. Die Abbruchbedingungen sollen so gestellt werden, daß garantiert werden kann, daß der Algorithmus terminiert. Die Blattbeschriftung dient der späteren Klassifikation mittels des aufgestellten Entscheidungsbaumes.

Um sicher zu sein, daß jeder Algorithmus mit jeder beliebigen Funktion $\mathcal{F}$ terminiert, muß die Bedingung $\theta_0 : \mathcal{T}' = \emptyset$ in der Menge von Abbruchbedingungen für jede Funktion $\mathcal{F}$ enthalten sein, wobei $\mathcal{T}' \subseteq \mathcal{T}$ die aktuelle Menge der verfügbaren Tests ist. Da die Menge der zulässigen Tests $\mathcal{T}$ endlich ist und bei jedem Schritt des Algorithmus um einen Test reduziert wird, wird die Bedingung θ_0 immer nach endlich vielen Schritten erfüllt sein.

Für die in diesem Unterkapitel eingeführte Funktion $\mathcal{F}$ definieren wir zwei weitere Abbruchbedingungen θ_1 und θ_2 nach [Zei99]:

- für eine Klasse $c_k \in \mathcal{C}$ gilt:
$$\frac{|\lambda_n|c_k|}{|\lambda_n|} \geq \theta_1$$
mit $0 \leq \theta_1 \leq 1$ als untere Schranke für den Anteil von Beispielen, die zur Klasse c_k gehören.

  Ein sinnvoller Wert für θ_1 ist nach [Zei99]: $\theta_1 = 0.9$.
  Mit einem Wert von $\theta_1 = 1$ erhalten wir das Abbruchskriterium für einen nach dem Verfahren ID3 gebildeten Entscheidungsbaum.

- für die Größe der Fuzzy-Menge λ_n gilt:
$$|\lambda_n| \leq \theta_2$$
mit $0 \leq \theta_2$ als untere Schranke für die Größe von λ an inneren Knoten

  Ein sinnvoller Wert nach [Zei99] wäre $\theta_2 = 1$.

Damit die Funktion $\mathcal{F}$ an einem Knoten n ausgeführt werden kann, braucht sie die Größen der Fuzzy-Mengen $\lambda_n, \lambda_n|c_k$ und die aktuelle Menge der verfügbaren Tests $\mathcal{T}' \subseteq \mathcal{T}$. Ein Pseudocode der Funktion $\mathcal{F}$ kann so aussehen:

$\mathcal{F}(|\lambda_n|, |\lambda_n|c_k|_{k=1,2,\dots,q}, \mathcal{T}')$

1. ist das Abbruchskriterium θ_1 für eine Klasse $c_k \in \mathcal{C}$ erfüllt, dann
   return true und die Beschriftung

$$L(n) \;=\; (0, \dots, \underset{\substack{\uparrow \\ c_k}}{1}, 0, \dots, 0)$$

2. ist das Abbruchskriterium θ_0 oder θ_2 erfüllt, dann
   return true und die Beschriftung

$$L(n) = (\frac{|\lambda_n|c_1|}{\Lambda_{n,\mathcal{C}}}, \frac{|\lambda_n|c_2|}{\Lambda_{n,\mathcal{C}}}, \dots, \frac{|\lambda_n|c_q|}{\Lambda_{n,\mathcal{C}}})$$

3. keine der definierten Abbruchskriterien ist erfüllt, dann
   return false

3.5 Algorithmus zum Aufbau von Fuzzy-Entscheidungsbäumen

In diesem Unterkapitel werden wir einen Algorithmus zum Aufbau von Fuzzy-Entscheidungsbäumen IFDT (*Induction of Fuzzy Decision Trees*) beschreiben, der aus den Algorithmen nach [Jan98] und [Zei99] entwickelt wurde und auf dem Prinzip *Top Down Induction of Decision Trees* (siehe Seite 15) basiert.

Um den Algorithmus ausführen zu können, benötigen wir eine Fuzzy-Lernmenge $\mathcal{L}$, die Menge der zulässigen Test $\mathcal{T}$, eine Methode $\mathcal{M}$ zur Auswahl des besten Tests (aus Unterkapitel 3.3) und die Funktion $\mathcal{F}$ zum Testen der Abbruchbedingungen (aus Unterkapitel 3.4).

Der Algorithmus wird mit Hilfe einer Funktion **IFDT** beschrieben, die als Eingabe eine Fuzzy-Menge λ_n für den aktuellen Knoten n und eine Menge von verfügbaren Tests $\mathcal{T}'$ nimmt, und sie auf einen Fuzzy-Entscheidungsbaum abbildet.

Input	λ_n	Fuzzy-Menge für die Zugehörigkeiten der Beispiele zu dem aktuellen Knoten n
	$\mathcal{T}' \subseteq \mathcal{T}$	die Menge der verfügbaren Tests, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten *root* des Fuzzy-Entscheidungsbaumes zum Knoten n liegen
Output		Fuzzy-Entscheidungsbaum

IFDT$(\lambda_n, \mathcal{T}')$

1. wenn die Funktion $\mathcal{F}(|\lambda_n|, |\lambda_n|c_k|_{k=1,2,\dots,q}, \mathcal{T}')$ für den aktuellen Knoten n erfüllt ist, dann definiere n als Blattknoten und beschrifte ihn gemäß $\mathcal{F}$ mit dem ausgerechneten Tupel von Klassenzugehörigkeiten und

   $\rightarrow$**stop**

2. wähle mit Hilfe der Methode $\mathcal{M}$ den besten Test $t = [cond_1^t, cond_2^t, \ldots, cond_{u_t}^t] \in \mathcal{T}'$ und beschrifte den aktuellen Knoten n mit t

3. für jede Bedingung $cond_h^t$ aus t füge eine ausgehende Kante mit einem leeren Nachfolgerknoten $n.cond_h^t$ an

4. an jedem Nachfolgerknoten $n.cond_h^t$ berechne die neue Fuzzy-Menge $\lambda_{n.cond_h^t}$, d.h. für jedes Beispiel f aus λ_n berechne seinen Zugehörigkeitsgrad zum Knoten $n.cond_h^t$ nach Definition 3.8 auf Seite 20, und führe

$$\textbf{IFDT}\ (\lambda_{n.cond_h^t}, \mathcal{T}' \setminus \{t\})$$

durch

$\rightarrow$ **stop**

Satz 3.10 *Der Algorithmus zum Aufbau von Fuzzy-Entscheidungsbäumen IFDT terminiert.*

Beweis Die Funktion **IFDT** wird zu Beginn mit der endlichen Menge der verfügbaren Tests $\mathcal{T}' = \mathcal{T}$ aufgerufen. Bei jedem rekursiven Aufruf wird diese Menge um einen Test verkleinert. Nach endlich vielen Schritte ist somit die Abbruchbedingung $\theta_0 : \mathcal{T} = \emptyset$ erfüllt (oder vorher bereits eine andere Abbruchbedingung) und der Algorithmus terminiert. □

Das nächste Beispiel skizziert, wie mit dem Algorithmus IFDT ausgehend aus der Lernmenge in der Tabelle 3.2 bzw. in der Tabelle 3.3 auf Seite 21 ein Fuzzy-Entscheidungsbaum induziert wird[5].

Beispiel 3.11 *Vor dem Aufruf der Funktion* **IFDT** *wird anhand der Lernmenge $\mathcal{L}$ die Menge der zulässigen Tests $\mathcal{T}$ für mehrfache Tests erzeugt, d.h. für jede linguistische Variable aus der Menge* $\{\mathsf{Technik}, \mathsf{Motor}, \ldots, \mathsf{Groeße}\}$ *wird ein Test gebildet (analog zu Beispiel 3.7 auf Seite 20). Wir erhalten eine Menge mit sechs verschiedenen Tests: $\mathcal{T} = \{t_1, t_2, \ldots, t_6\}$. Die Methode $\mathcal{M}$ sucht nach dem Informationsgewinn (siehe Unterkapitel 3.3) unter der Priorität:* Größe, Motor, Typ, Technik, Gestaltung, Innenraum. *Für die Funktion $\mathcal{F}$ wird die Funktion aus Unterkapitel 3.4 mit den dort beschriebenen Abbruchbedingungen θ_0, θ_1 und θ_2 eingesetzt.*

Die Ausführung des Algorithmus beginnt mit dem Aufruf der Funktion **IFDT** $(\lambda_{root}, \mathcal{T})$. *Am Wurzelknoten sind alle 6 Beispiele aus der Lernmenge mit einem Zugehörigkeitsgrad von Eins vertreten. Die Funktion $\mathcal{F}(|\lambda_{root}|, |\lambda_{root}|c_k|_{k=1,2,\ldots,q}, \mathcal{T})$ ist nicht erfüllt. Im Schritt 2 wählt die Methode $\mathcal{M}$ anhand der Fuzzy-Menge λ_n den optimalen Test* $t = [\mathsf{Typ} = \mathsf{Video}, \mathsf{Typ} = \mathsf{Bild}, \mathsf{Typ} = \mathsf{Text}]$. *Im nächsten Schritt 3 wird für jede Bedingung des Tests t eine ausgehende Kante mit einem leeren Nachfolgerknoten erzeugt (siehe Abb. 3.4).*

Im Schritt 4 werden die Zugehörigkeitsgrade der Beispiele aus $\mathcal{L}$ zu den neuen Nachfolgern nach Definition 3.8 auf Seite 20 ausgerechnet und die Funktion **IFDT** *für jeden Nachfolgerknoten rekursiv aufgerufen.*

Betrachten wir den Nachfolger von root für die Bedingung Typ = Video. *Für die Zugehörigkeit der Beispiele aus $\mathcal{L}$ zu der Fuzzy-Menge $\lambda_{root.\mathsf{Typ}=\mathsf{Video}}$ erhalten wir: die Beispiele mit dem Index 1, 2 und 5 haben einen Zugehörigkeitsgrad von Eins, der Rest einen Zugehörigkeitsgrad von Null. Die Funktion* **IFDT** *wird rekursiv mit den Parametern $\lambda_{root.\mathsf{Typ}=\mathsf{Video}}$*

[5] Für ein ausführlicheres Beispiel für Induktion mittels IFDT, siehe [Gue01].

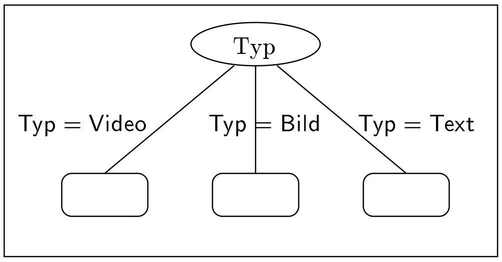

Abbildung 3.4: Fuzzy-Entscheidungsbaum nach dem Schritt 3

und $\mathcal{T}' = \mathcal{T} \backslash \{t\}$ *aufgerufen. Die Funktion* $\mathcal{F}(|\lambda_{root.\mathsf{Typ=Video}}|, |\lambda_{root.\mathsf{Typ=Video}}|c_k|_{k=1,2,\ldots,q}, \mathcal{T}')$ *ist nicht erfüllt, also wird der Knoten als innerer Knoten definiert. Die weiteren rekursiven Aufrufe werden wir hier nicht beschreiben.*

Der Nachfolger von root für Typ = Bild *wird von den Beispielen 4 und 6 erreicht (vgl. Tabelle 3.3 auf Seite 21), die den Zugehörigkeitsgrad von Eins zu der Fuzzy-Menge* $\lambda_{root.\mathsf{Typ=Bild}}$ *haben. Da die beiden Beispiele zur selben Klasse gehören, wird im Schritt 1 des rekursiven Aufrufs die Funktion* $\mathcal{F}(|\lambda_{root.\mathsf{Typ=Bild}}|, |\lambda_{root.\mathsf{Typ=Bild}}|c_k|_{k=1,2,\ldots,q}, \mathcal{T}')$ *(die Bedingung* θ_1*) erfüllt. Der Knoten wird somit als Blattknoten definiert.*

Der dritte Nachfolger $root.$Typ = Test *wird ebenfalls sofort als Blattknoten definiert.*

Am Ende der Induktion erhalten wir den Fuzzy-Entscheidungsbaum aus der Abbildung 3.2 auf Seite 17.

Kapitel 4

Inkrementelles Lernen und Entscheidungsbäume

Aus dem Gebiet des Maschinellen Lernens sind verschiedene Verfahren zum Erzeugen eines Datenmodells, das Zusammenhänge aus einem Gegenstandsbereich wiederspiegelt, bekannt. Man unterscheidet inkrementelles und nicht-inkrementelles Lernen.

Typisch für die nicht-inkrementellen Verfahren ist es, daß bei der Beobachtung von neuen Gegenständen ein neues Modell erzeugt werden muß. Die Verhältnisse zwischen den Gegenständen haben sich verändert und das alte Modell ist nicht mehr verwendbar.

Dagegen ist es für das inkrementelle Lernen typisch, daß die erzeugten Datenmodelle ständig aktualisiert werden können. Wird ein neues Beispiel beobachtet, so ist ein inkrementelles Verfahren in der Lage, das existierende Modell gemäß des beobachteten Beispiels zu ändern. Das so modifizierte Modell gibt exakt die neuen Zusammenhänge in dem betrachteten Gegenstandsbereich wieder.

In Unterkapitel 4.1 wird ein Algorithmus zum inkrementellen Aufbau von klassischen Entscheidungsbäumen vorgestellt. Ausgehend von diesem Algorithmus und dem Algorithmus IFDT zum Aufbau von Fuzzy-Entscheidungsbäumen wird in 4.2 die Idee zur Konstruktion eines Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen entwickelt.

Im letzten Kapitel haben wir die Grundidee des Aufbaus von klassischen Entscheidungsbäumen, sowie einen Algorithmus zur Induktion von Fuzzy-Entscheidungsbäumen kennengelernt. Solche Verfahren, die auf dem Prinzip *Top Down Induction of Decision Trees* basieren, wenden ein nicht-inkrementelles Lernen an, um einen Entscheidungsbaum zu induzieren. Sie benötigen zum Aufbau eines Entscheidungsbaumes eine vorgegebene Lernmenge. Die Anpassung des so erzeugten Entscheidungsbaumes bei der Beobachtung eines neuen Beispiels ist nur durch eine neue Induktion möglich. Dabei wird ein neuer Entscheidungsbaum aufgrund der alten Lernmenge und der neue Beispiele erstellt.

Ein Verfahren zum inkrementellen Aufbau von Entscheidungsbäumen ist in der Lage, ständig neue Beispiele aufzunehmen und die veränderten Verhältnisse innerhalb der Lernmenge ohne neue Induktion des Baumes auszudrucken. Die Anpassung erfolgt durch die Abbildung des existierenden Entscheidungsbaumes und des neuen Beispiels auf einen neuen Entscheidungsbaum.

4.1 Der Algorithmus ITI

Ein bekanntes Verfahren zum inkrementellen Aufbau von klassischen Entscheidungsbäumen ist der Algorithmus ITI (*Incremental Tree Induction*) von [UBC97]. Er basiert auf der wiederholten Ausführung dreier Basis-Funktionen:

1. Hinzufügen eines neuen Beispiels zu einem existierenden Entscheidungsbaum
2. Umstrukturieren des Baumes, d.h. Ersetzen des Tests eines Knotens durch einen andren Test
3. Optimierung eines Entscheidungsbaumes, d.h. es wird dafür gesorgt, daß an jedem Knoten ein optimaler Test bezüglich einer Methode $\mathcal{M}$ steht

Das Hinzufügen eines Beispiels zu einem Entscheidungsbaum wird immer dann ausgeführt, wenn ein neues Beispiels beobachtet wurde. Durch das Hinzufügen des Beispiels wird die Beispielmenge, woraus der Baum erzeugt wurde, verändert. Anhand der Beispielmenge werden auch bestimmte Größen ausgerechnet, die eine Rolle bei der Auswahl eines Tests durch die Methode $\mathcal{M}$ und beim Testen der Abbruchbedingungen durch die Funktion $\mathcal{F}$ spielen. Um die benötigten Größen schnell aktualisieren zu können, werden sie an den Knoten des Baumes gespeichert. Sie werden beim Hinzufügen eines neuen Beispiels an dem Beispiel angepaßt.

Das Hinzufügen eines Beispiels beginnt mit dem Hinzufügen des Beispiels zum Wurzelknoten des Entscheidungsbaumes und danach rekursiv bis zum Erreichen eines Blattknotens. An jedem inneren Knoten wird die gespeicherte Information gemäß des Beispiels aktualisiert und der Knoten markiert. Durch die Markierung wird gekennzeichnet, daß die Information, anhand derer der Test dieses Knotens ausgewählt wurde, sich verändert hat und nicht mehr garantiert werden kann, daß der Test optimal ist. Das Beispiel wird in Abhängigkeit davon, welche Bedingung erfüllt ist, zu einem Nachfolgerknoten weitergeleitet.

An einem Blattknoten wird mittels der Funktion $\mathcal{F}$ getestet, ob eine der Abbruchbedingungen nach dem Hinzufügen des Beispiels immer noch erfüllt ist. Ist dies der Fall, so wird das Beispiel zu der Menge von Beispielen, die den Knoten erreicht haben, hinzugefügt. Ist dies nicht der Fall, so wird mit Hilfe der Methode $\mathcal{M}$ ein optimaler Test ausgesucht. Der ausgewählte Test wird an dem Blattknoten gesetzt, d.h. der Blattknoten wird in einen inneren Knoten umgewandelt. Die am Blattknoten gespeicherten Beispiele werden eines nach dem anderen zu den entsprechenden Nachfolgerknoten hinzugefügt.

Wurde eine Menge von Beispielen gleichzeitig beobachtet, so können die Beispiele der Reihe nach zu einem Entscheidungsbaum hinzugefügt werden oder, falls am Anfang kein Entscheidungsbaum existiert, kann ein Entscheidungsbaum durch ein nicht-inkrementelles Verfahren generiert werden.

Die zweite Basis-Funktion tauscht den Test an einem Knoten durch einen anderen Test um. Das führt zu einer neuen Verzweigung der Beispiele, die an dem Knoten vorhanden sind, und zu einer Umstrukturierung der Unterbäume. Um das Ersetzen durchführen zu können, ist die folgende Eigenschaft wichtig:

Eigenschaft 4.1 *Jeder klassische Entscheidungsbaum ist* **kompositorisch***, d.h. für jeden beliebigen inneren Knoten n des Entscheidungsbaumes sind die folgenden Bedingungen erfüllt:*

1. *die Beispielmenge, worauf der Knoten basiert, entspricht genau der Vereinigung der Beispielmengen aus seinen Nachfolgerknoten*

2. *die gespeicherte Information am Knoten n ergibt sich aus der Summe der gespeicherten Informationen an seinen Nachfolgerknoten*

Wir werden das Ersetzen eines Tests anhand der Beispiele aus Tabelle 3.1 auf Seite 14 illustrieren. Am Wurzelknoten n sind alle Beispiele, d.h. die Menge $\{1, 2, 3, 4\}$ vertreten. Zur Vereinfachung betrachten wir binäre Tests, wobei die Beispiele, die den gefragten Attributwert heben nach links verzweigt werden, die andere nach rechts. Am Wurzelknoten n sei der Test $t_n = [\mathsf{Typ} = \mathsf{Text}, \mathsf{Typ} \neq \mathsf{Text}]$ gesetzt. Somit wird der linke Nachfolgerknoten (in den nachfolgenden Bilder) von der Beispielmenge $\{1, 2\}$ erreicht, der rechte—von der Menge $\{3, 4\}$. Unser Ziel sei es, den Test t_n durch den Test $t = [\mathsf{Typ} = \mathsf{Video}, \mathsf{Typ} \neq \mathsf{Video}]$ zu ersetzen. Man unterscheidet drei Fälle:

- die beiden Nachfolgerknoten von n sind Blattknoten:

  In diesem Fall ersetzt man den Test für den Knoten n durch den gewünschten Test t. Damit werden neue Nachfolgerknoten von n erzeugt und die Beispiele aus den alten Blattknoten müssen bezüglich dem neuen Test t neu verzweigt werden, d.h. sie werden einem nach dem anderen zu einem der neuen Nachfolgerknoten hinzugefügt.

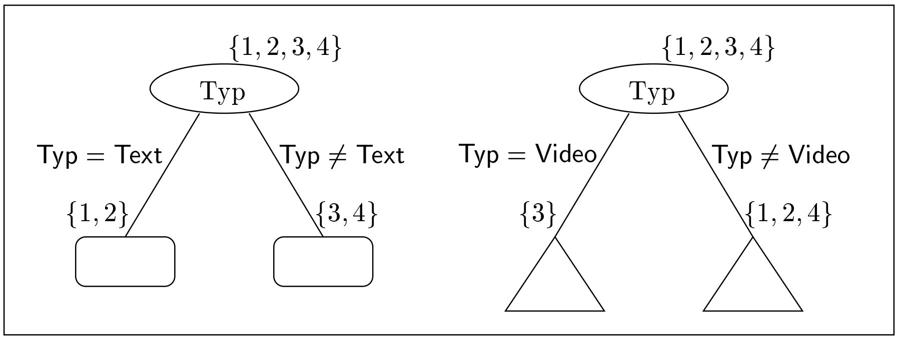

Abbildung 4.1: Beispiel für das Ersetzen eines Tests

  Abbildung 4.1 visualisiert links einen Baum vor und rechts einen Baum nach dem Ersetzen des Tests am Wurzelknoten. Die Beispiele $\{1, 2\}$ aus dem linken Blattknoten erfüllen die Bedingung $\mathsf{Typ} \neq \mathsf{Video}$ und werden zu dem rechten Nachfolgerknoten des neuen Baumes hinzugefügt. Das Beispiel 3 ist das einzige Beispiel, das die Bedingung $\mathsf{Typ} = \mathsf{Video}$ erfüllt und wird zu dem neuen linken Knoten hinzugefügt.

  Es kann vorkommen, daß durch die neue Verzweigung der Beispiele aus dem Knoten n der Entscheidungsbaum in der Höhe wächst, d.h. die neuen Nachfolgerknoten können nicht als Blattknoten definiert werden.

- der eine Nachfolger von n ist Blattknoten, der andere ein innerer Knoten mit dem gewünschten Test t:

  In diesem Fall ersetzen wir den Wurzelknoten n mit dem Test t_n durch den inneren Nachfolgerknoten mit dem gewünschten Test t. Die Beispiele aus dem Blattknoten werden zu dem inneren Nachfolgerknoten (der neue Wurzelknoten) hinzugefügt.

  Die Abbildung 4.2 stellt links die Ausgangssituation dar - rechts das Ergebnis. Nach dem Ersetzen des Wurzelknotens durch den Nachfolgerknoten ist der neue Wurzelknoten auf der Beispielmenge aufgebaut, worauf der Nachfolgerknoten aufgebaut

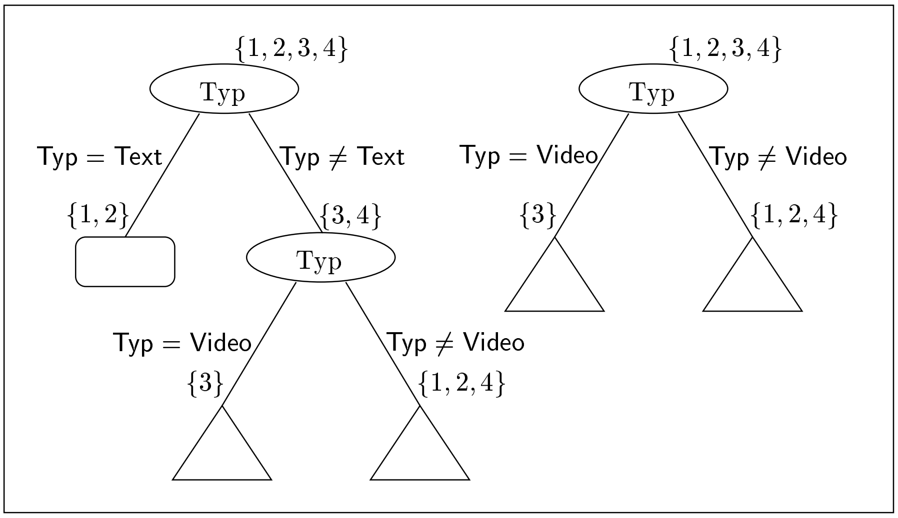

Abbildung 4.2: Beispiel für das Ersetzen eines Tests

war, nämlich $\{3, 4\}$. Durch das Hinzufügen der Beispiele $\{1, 2\}$ aus dem linken Blattknoten zu dem neuen Wurzelknoten erreicht man, daß der neue Wurzelknoten genau auf der gleichen Beispielmenge $\{1, 2, 3, 4\}$ basiert, wie der alte Wurzelknoten. Die Beispiele $\{1, 2\}$ erfüllen die Bedingung $\mathsf{Typ} \neq \mathsf{Video}$ und werden zu dem rechten Nachfolgerknoten hinzugefügt.

- die beiden Nachfolger von n sind innere Knoten mit dem gewünschten Test t:

  In diesem Fall werden die Tests an dem Wurzelknoten und an seinen Nachfolgerknoten vertauscht. Daraufhin ändern sich die Beispielmengen, worauf die Nachfolger von n gebildet sind und sie müssen neue Kinderknoten zugewiesen bekommen.

  Wie man in Abbildung 4.3 (untere Baum) sieht, wird für den Wurzelknoten der gewünschte Test t, für seine Nachfolger der Test t_n gesetzt. Betrachten wir den linken Nachfolgerknoten von n, der durch das Folgen der Kante mit der Bedingung $\mathsf{Typ} = \mathsf{Video}$ erreicht wird. Seine Kinderknoten müssen, ausgehend aus der Beispielmenge am Knoten n, auf die Beispiele aufgebaut werden, die die Bedingung $\mathsf{Typ} = \mathsf{Video} \wedge \mathsf{Typ} = \mathsf{Text}$ bzw. $\mathsf{Typ} = \mathsf{Video} \wedge \mathsf{Typ} \neq \mathsf{Text}$ erfüllen. Das entspricht genau der Beispielmenge, worauf die Knoten A bzw. C des ursprünglichen Entscheidungsbaumes (siehe oberer Baum) gebildet worden waren. Also der linke Nachfolger von n bekommt als Kinder die Knoten A und C (siehe unterer Baum). Die Beispielmenge $\{3\}$, worauf er basiert, ergibt sich durch die Vereinigung der Beispielmengen ($\emptyset$ und $\{3\}$) von seinen Kinderknoten A und C.

Die restlichen Fälle lassen sich auf diese drei Grundsituationen zurückführen.

Die Möglichkeit ein Teilbaum durch einen anderen, dessen Wurzelknoten mit einem gewünschten Test beschriftet ist, zu ersetzen, ist die Voraussetzung für die Optimierung des Entscheidungsbaumes. Ziel des Optimierungsprozesses ist es, den Entscheidungsbaum so umzustrukturieren, daß an jedem Knoten ein optimaler Test steht. Die Optimierung des Baumes wird durch die dritte Basis-Funktion ausgeführt und kann nach dem Hinzufügen eines Beispiels durchgeführt werden. Die Funktion wird entweder jedes Mal oder nach beliebigen Abständen ausgeführt. Bei der Optimierung eines Entscheidungsbaumes wird rekursiv, beginnend mit dem Wurzelknoten, jeder innere Knoten untersucht.

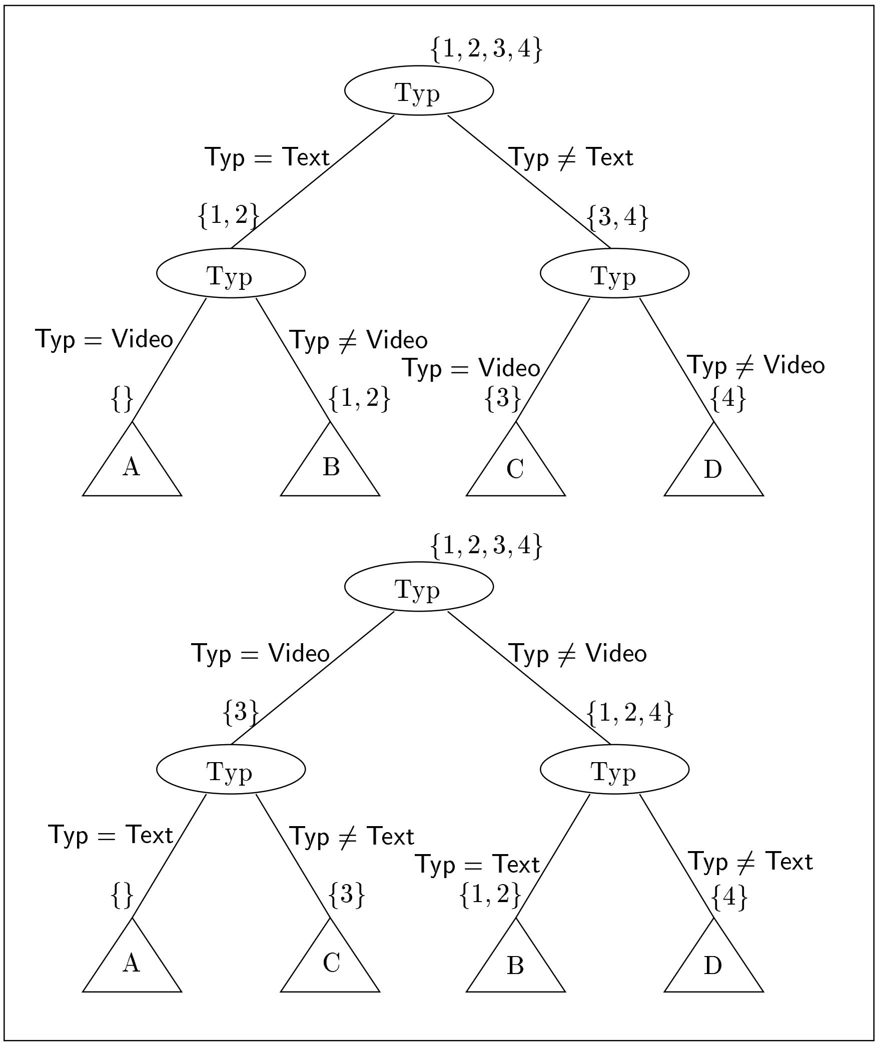

Abbildung 4.3: Beispiel für das Ersetzen eines Tests

Die Funktion $\mathcal{F}$ testet, ob eine der Abbruchbedingungen an dem aktuellen Knoten erfüllt ist. Wenn dies der Fall ist, dann wird der innere Knoten als Blattknoten definiert und gemäß $\mathcal{F}$ beschriftet. Sind die Abbruchbedingungen nicht erfüllt und der Knoten ist markiert, so wird die Markierung aufgehoben und mit Hilfe der gespeicherten Information und der Methode $\mathcal{M}$ der beste Test ermittelt. Steht an dem aktuellen Knoten ein unterschiedlicher Test, so wird er durch die Ausführung der zweiten Basis-Funktion durch den optimalen Test ersetzt. Ansonsten wird sofort mit der Untersuchung der Nachfolgerknoten fortgesetzt.

Im Laufe der Optimierung ist garantiert, daß alle Vorgängerknoten des aktuellen Knotens mit dem optimalen Test bezüglich der Methode $\mathcal{M}$ beschriftet sind. Nach dem Beenden des Optimierungsprozesses wurden alle markierten Knoten untersucht und mit dem besten Test bezüglich der Methode $\mathcal{M}$ versehen. Damit ist garantiert, daß der erhaltene Entscheidungsbaum optimal ist.

Unser Ziel ist es, einen Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen, ausgehend von dem Algorithmus IFDT (vorgestellt in Unterkapitel 3.5) und dem Algorithmus ITI zu entwickeln. Es ist wünschenswert, daß der so inkrementell erzeugte Fuzzy-Entscheidungsbaum gleich dem Fuzzy-Entscheidungsbaum ist, der durch den Algorithmus IFDT ausgehend von der gleichen Lernmenge induziert würde. Im nächsten Unterkapitel werden wir die allgemeinen Überlegungen zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen einführen.

4.2 Idee zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen

Für den Aufbau eines Entscheidungsbaumes benötigen wir eine Lernmenge $\mathcal{L}$. In Kapitel 3 haben wir den Unterschied in der Struktur der Lernmenge in Abhängigkeit davon, ob sie zum Aufbau von klassischen oder unscharfen Entscheidungsbäumen verwendet wird, gesehen. Natürlich benötigen wir für den inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen eine Fuzzy-Lernmenge $\mathcal{L}$ (siehe Unterkapitel 3.2), in der jedes Beispiel f durch einen Zugehörigkeitsgrad $\mu_{\mathrm{v}_{il}}(f)$ zu jedem linguistischen Wert jeder Variable $a_i \in A$ und einen Zugehörigkeitsgrad für jede Klasse $\mu_{c_k}(f)$ spezifiziert ist. Ebenso wird der Zugehörigkeitsgrad jedes Beispiels zu einem Knoten n des Baumes durch eine Fuzzy-Menge λ_n (mit $\lambda_{root} = 1$) beschrieben.

Bei einem inkrementellen Aufbau wird die Lernmenge ständig durch neue Beispiele erweitert, d.h. die Anzahl der Elemente der Lernmenge $\mathcal{L}$ und somit der Fuzzy-Mengen $\mu_{\mathrm{v}_{il}}, \mu_{c_k}$ und λ wird mit der Beobachtung jedes neuen Beispiels vergrößert. Die Zugehörigkeitsgrade des Beispiels werden zu den oben genannten Fuzzy-Mengen hinzugefügt, wobei deren Grundintervall entsprechend erweitert werden muß. Die Zuordnung für bereits betrachtete Beispiele bleibt unverändert.

Der inkrementelle und der nicht-inkrementelle Aufbau von Entscheidungsbäumen benötigen eine Menge von zulässigen Tests $\mathcal{T}$, eine Funktion $\mathcal{F}$, die testet, ob eine der Abbruchbedingungen erfüllt ist, sowie eine Methode $\mathcal{M}$ zur Auswahl von Tests. Die Idee des inkrementellen Aufbaus von klassischen Entscheidungsbäumen besteht darin, an jedem Knoten des Baumes bestimmte Information zu speichern, die durch die Methode zur Auswahl eines Tests, durch die Funktion zum Testen der Abbruchbedingungen und bei der Umstrukturierung des Baumes verwendet wird und nicht jedes Mal neu anhand der aktuellen Lernmenge berechnet werden muß.

Wenn wir uns beim inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen auf solche Methoden $\mathcal{M}$ zur Auswahl eines Tests und Funktionen $\mathcal{F}$ zum Testen der Abbruchbedingungen, die wir für den nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen in Unterkapiteln 3.3 und 3.4 eingeführt haben, beschränken, dann können wir ableiten, welche Information benötigt wird und gespeichert werden muß. An jedem inneren Knoten n brauchen wir die Größe der Fuzzy-Mengen:

$$\lambda_n \ , \ \lambda_n|c_k \ , \ \lambda_n|cond_d^{t^\star} \ , \ \lambda_n|cond_d^{t^\star}|c_k \ , \ \lambda_{n,t^\star}^{missing}$$

An jedem Blattknoten n benötigen wir die Größe der Fuzzy-Mengen:

$$\lambda_n \ , \ \lambda_n|c_k$$

und den Zugehörigkeitsgrad jedes Beispiels f zum Blattknoten n, d.h. den Wert $\lambda_n(f)$. Diese Information wird immer bei dem Hinzufügen eines Beispiels aktualisiert.

Der Prozeß des inkrementellen Aufbaus eines klassischen Entscheidungsbaumes beinhaltet, außer dem Hinzufügen eines Beispiels, noch das Ersetzen eines Tests und die Optimierung des Entscheidungsbaumes.

Das Ersetzen des Tests an einem Knoten durch einen anderen Test, wie im Algorithmus ITI zum inkrementellen Aufbau von klassischen Entscheidungsbäumen beschrieben wurde, ist möglich, weil der klassische Entscheidungsbaum kompositorisch ist (siehe Eigenschaft 4.1 auf Seite 31). Um diese Idee auch für Fuzzy-Entscheidungsbäume überführen zu können, muß der Fuzzy-Entscheidungsbaum ebenfalls kompositorisch sein:

Eigenschaft 4.2 *Ein Fuzzy-Entscheidungsbaum FT bezüglich einer Klassenmenge $\mathcal{C}$ und einer Menge von zulässigen Tests $\mathcal{T}$ erfüllt die Eigenschaft* **kompositorisch**, *wenn für jeden inneren Knoten n (beschriftet mit dem Test $t_n = [cond_1^{t_n}, cond_2^{t_n}, \ldots, cond_{u_{t_n}}^{t_n}] \in \mathcal{T}$), die Menge seiner Nachfolgerknoten $\mathcal{N}(n) = \{n.cond_1^{t_n}, n.cond_2^{t_n}, \ldots, n.cond_{u_{t_n}}^{t_n}\}$, und jeden beliebigen Test $t^\star = [cond_1^{t^\star}, \ldots, cond_{u_{t^\star}}^{t^\star}] \in \mathcal{T}$ gilt:*

1. *die Zugehörigkeitsgrade der Beispiele zu der Fuzzy-Menge λ_n können aus den Zugehörigkeitsgraden der Beispiele zu den Fuzzy-Mengen $\lambda_{n.cond_g^{t_n}}$ an den Nachfolgern von n ermittelt werden, d.h. für jedes Beispiel f gilt:*

$$\lambda_n(f) = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \lambda_{n.cond_g^{t_n}}(f) \tag{4.1}$$

2. *die Größe jeder benötigten Fuzzy-Menge am Knoten n läßt sich als die Summe der Größen der entsprechenden Fuzzy-Mengen an den Nachfolgerknoten von n berechnen, d.h. die folgenden Gleichungen sind erfüllt:*

$$|\lambda_n| = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} |\lambda_{n.cond_g^{t_n}}| \tag{4.2}$$

$$|\lambda_n|c_k| = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} |\lambda_{n.cond_g^{t_n}}|c_k| \tag{4.3}$$

$$|\lambda_n|cond_d^{t^\star}| = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} |\lambda_{n.cond_g^{t_n}}|cond_d^{t^\star}| \tag{4.4}$$

$$|\lambda_n|cond_d^{t^\star}|c_k| = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} |\lambda_{n.cond_g^{t_n}}|cond_d^{t^\star}|c_k| \tag{4.5}$$

für jede Bedingung $cond_d^{t^\star}, 1 \leq d \leq u_{t^\star}$ und jede Klasse $c_k \in \mathcal{C}$

Im Gegensatz zum klassischen Fall ist nicht jeder Fuzzy-Entscheidungsbaum kompositorisch, da ein Beispiel zu verschiedenen Graden mehrere Bedingungen eines Tests erfüllen kann, d.h. mit verschiedenen Zugehörigkeitsgraden zu mehreren Nachfolgerknoten gehören kann. Wir bemerken aber, daß unter bestimmten Bedingungen diese Eigenschaft erfüllt werden kann.

Annahme 4.3 *Der inkrementelle bzw. nicht-inkrementelle Aufbau von Fuzzy-Entscheidungsbäumen wird nur mit einer t-Norm ausgeführt, die die folgende Bedingung:*

$$\top(a,b) + \top(a,c) = \top(a, b+c)$$

für $a, b, c \in [0,1]$ und $b + c \in [0,1]$ erfüllt.

Eine t-Norm, die diese Bedingung erfüllt, ist z.B. die t-Norm, die als Produkt definiert ist (siehe Gleichung(2.5) auf Seite 10):

$$\top(a,b) + \top(a,c) = a \cdot b + a \cdot c = a \cdot (b+c) = \top(a, b+c)$$

Hilfssatz 4.4 *Unter der Annahme 4.3 ist jeder Fuzzy-Entscheidungsbaum kompositorisch, d.h. die Gleichungen* (4.1) *bis* (4.5) *sind erfüllt.*

Beweis

zu Gleichung (4.1):

$$\begin{aligned}
\sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \lambda_{n.cond_g^{t_n}}(f) &= \sum_{g=1}^{u_{t_n}} \lambda_{n.cond_g^{t_n}}(f) \\
&\overset{\text{Def. 3.8 auf S. 20}}{=} \sum_{g=1}^{u_{t_n}} \top(\lambda_n(f), \mu_{cond_g^{t_n}}(f)) \\
&\overset{\text{Ann. 4.3 auf S. 36}}{=} \top(\lambda_n(f), \sum_{g=1}^{u_{t_n}} \mu_{cond_g^{t_n}}(f)) \\
&\overset{\text{Kor. 3.6 auf S. 20}}{=} \top(\lambda_n(f), 1) \\
&= \lambda_n(f)
\end{aligned}$$

zu Gleichung (4.2):

$$\begin{aligned}
\sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} |\lambda_{n.cond_g^{t_n}}| &= \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \sum_{f=1}^{m} \lambda_{n.cond_g^{t_n}}(f) \\
&= \sum_{f=1}^{m} \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \lambda_{n.cond_g^{t_n}}(f) \\
&\overset{\text{Gl. (4.1) auf S. 36}}{=} \sum_{f=1}^{m} \lambda_n(f) \\
&= |\lambda_n|
\end{aligned}$$

zu Gleichung (4.3):

$$\begin{aligned}
\sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} |\lambda_{n.cond_g^{t_n}} | c_k| &= \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \sum_{f=1}^{m} \lambda_{n.cond_g^{t_n}} | c_k(f) \\
&\overset{\text{Gl. (3.3) auf S. 22}}{=} \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \sum_{f=1}^{m} \top(\lambda_{n.cond_g^{t_n}}(f), \mu_{c_k}(f)) \\
&= \sum_{f=1}^{m} \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \top(\lambda_{n.cond_g^{t_n}}(f), \mu_{c_k}(f)) \\
&\overset{\text{Ann. 4.3 auf S. 36}}{=} \sum_{f=1}^{m} \top(\sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \lambda_{n.cond_g^{t_n}}(f), \mu_{c_k}(f)) \\
&\overset{\text{Gl. (4.1) auf S. 36}}{=} \sum_{f=1}^{m} \top(\lambda_n(f), \mu_{c_k}(f)) \\
&= |\lambda_n | c_k|
\end{aligned}$$

Die Gleichungen (4.4) und (4.5) auf Seite 36 lassen sich analog beweisen. □

Die Optimierung eines Fuzzy-Entscheidungsbaumes verläuft nach dem gleichen Schema wie beim Algorithmus ITI und kann entweder jedes Mal nach dem Hinzufügen eines Beispiels oder nach beliebigen Abständen ausgeführt werden. Es ist ratsam, vor der Durchführung einer Klassifikation mittels des Fuzzy-Entscheidungsbaumes diesen zu optimieren.

Wegen Annahme 4.3 vereinfachen sich die Berechnungen innerhalb der eingesetzten Methode $\mathcal{M}$ und der Funktion $\mathcal{F}$.

Hilfssatz 4.5 *Für einen Fuzzy-Entscheidungsbaum, der kompositorisch ist, vereinfacht sich die Berechnung der Größen* $\Lambda_{n,\mathcal{C}}, \Lambda_{n,t^\star}, \Lambda_{n,cond_d^{t^\star},\mathcal{C}}$ *an jedem Knoten* n *und für jeden Test* $t^\star$*. Es gilt:*

$$\begin{aligned}
\Lambda_{n,\mathcal{C}} &= |\lambda_n| \\
\Lambda_{n,t^\star} &= |\lambda_n| \\
\Lambda_{n,cond_d^{t^\star},\mathcal{C}} &= |\lambda_n|cond_d^{t^\star}|
\end{aligned}$$

Beweis

$$\begin{aligned}
\Lambda_{n,\mathcal{C}} &\overset{\text{Gl. (3.7) auf S. 23}}{=} \sum_{k=1}^{q} |\lambda_n|c_k| \\
&= \sum_{k=1}^{q}\sum_{f=1}^{m} \lambda_n|c_k(f) \\
&\overset{\text{Gl. (3.3) auf S. 22}}{=} \sum_{k=1}^{q}\sum_{f=1}^{m} \top(\lambda_n(f), \mu_{c_k}(f)) \\
&= \sum_{f=1}^{m}\sum_{k=1}^{q} \top(\lambda_n(f), \mu_{c_k}(f)) \\
&\overset{\text{Ann. 4.3 auf S. 36}}{=} \sum_{f=1}^{m} \top(\lambda_n(f), \sum_{k=1}^{q}\mu_{c_k}(f)) \\
&\overset{\text{Gl. (3.2) auf S. 19}}{=} \sum_{f=1}^{m} \top(\lambda_n(f), 1) \\
&= \sum_{f=1}^{m} \lambda_n(f) \\
&= |\lambda_n|
\end{aligned}$$

Die beiden anderen Gleichungen lassen sich analog aus den Gleichungen (3.9) und (3.11) auf Seite 23, sowie Korollar 3.6 und die Gleichungen (3.4) und (3.5) ableiten. □

Korollar 4.6 *Wegen Hilfssatz 4.5 lassen sich die Wahrscheinlichkeitsverteilungen* (3.6), (3.8) *und* (3.10) *auf Seite 23 für jeden Knoten* n *und jeden Test* $t^\star$ *wie folgt aufstellen:*

$$\begin{aligned}
W(\lambda_n) &= (\frac{|\lambda_n|c_1|}{|\lambda_n|}, \frac{|\lambda_n|c_2|}{|\lambda_n|}, \ldots, \frac{|\lambda_n|c_q|}{|\lambda_n|}) \\
W(t^\star, \lambda_n) &= (\frac{|\lambda_n|cond_1^{t^\star}|}{|\lambda_n|}, \frac{|\lambda_n|cond_2^{t^\star}|}{|\lambda_n|}, \ldots, \frac{|\lambda_n|cond_{u_{t^\star}}^{t^\star}|}{|\lambda_n|}) \\
W(\lambda_n|cond_d^{t^\star}) &= (\frac{|\lambda_n|cond_d^{t^\star}|c_1|}{|\lambda_n|cond_d^{t^\star}|}, \frac{|\lambda_n|cond_d^{t^\star}|c_2|}{|\lambda_n|cond_d^{t^\star}|}, \ldots, \frac{|\lambda_n|cond_d^{t^\star}|c_q|}{|\lambda_n|cond_d^{t^\star}|})
\end{aligned}$$

Kapitel 5

Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen

Die Idee eines Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen ist durch das Kombinieren der Idee des inkrementellen Aufbaus von klassischen Entscheidungsbäumen (Algorithmus ITI, Unterkapitel 4.1) mit der Idee des nicht-inkrementellen Aufbaus von Fuzzy-Entscheidungsbäumen (Algorithmus IFDT, Unterkapitel 3.5) entstanden. In diesem Kapitel werden wir einen solchen Algorithmus zur inkrementellen Konstruktion von Fuzzy-Entscheidungsbäumen entwickeln.

In Unterkapitel 5.1 wird ein inkrementeller Fuzzy-Entscheidungsbaum definiert und erläutert, welche Information an den Knoten des Baumes gespeichert wird. In 5.2 werden die einzelnen Funktionen des Algorithmus definiert. Zum Schluß geben wir eine Beispielinduktion an, die mit Hilfe der Implementierung unseres Algorithmus im Projekt ISeC durchgeführt wurde.

Ausgehend von den getroffenen Überlegungen über einen inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen, die wir in Unterkapitel 4.2 angestellt haben, werden wir in diesem Kapitel einen Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen IIFDT (*Incremental Induction of Fuzzy Decision Trees*) definieren. Der Algorithmus IIFDT besteht aus den folgenden Basis-Funktionen:

1. **addExampleToTree** zum Hinzufügen eines neuen Beispiels zu einem existierenden Fuzzy-Entscheidungsbaum, beschrieben in Abschnitt 5.2.1 auf Seite 43

2. **transposition** zum Ersetzen eines Tests an einem Knoten durch einen anderen Test, beschrieben in Abschnitt 5.2.2 auf Seite 45

3. **ensureBestTest** zur Optimierung eines Entscheidungsbaumes, beschrieben in Abschnitt 5.2.3 auf Seite 48

Die einzelnen Funktionen können nicht getrennt von einander betrachtet werden, da sie sich gegenseitig aufrufen, d.h. es besteht eine gewisse Schachtelung. Neben den spezifischen Übergabeparametern für jede Funktion, existieren auch globale Parameter, die durch alle

Funktionen gebraucht werden und sich im Laufe des inkrementellen Aufbaus eines Fuzzy-Entscheidungsbaumes nicht ändern. Solche Parameter sind die Methode $\mathcal{M}$ zur Auswahl des besten Tests und die Funktion $\mathcal{F}$ zum Testen der Abbruchbedingungen.

Theoretisch kann bei der Ausführung des Algorithmus IIFDT für die Methode $\mathcal{M}$ und die Funktion $\mathcal{F}$ jede beliebige Methode bzw. Funktion eingesetzt werden, die folgende Bedingungen erfüllt:

- die an den Knoten gespeicherte Information ist ausreichend
- die Methode $\mathcal{M}$ wählt deterministisch einen Test aus
- die Funktion $\mathcal{F}$ berücksichtigt die Abbruchbedingung θ_0 von Seite 26
- für einen leeren Blattknoten[1] soll die Funktion $\mathcal{F}$ immer erfüllt sein
- die Funktion $\mathcal{F}$ liefert deterministisch eine Blattbeschriftung zurück

Die in Unterkapitel 3.3 beschriebenen Methoden $\mathcal{M}$ und die in Unterkapitel 3.4 vorgestellte Funktion $\mathcal{F}$ erfüllen die gestellten Bedingungen und können mit dem Algorithmus IIFDT eingesetzt werden.

5.1 Inkrementeller Fuzzy-Entscheidungsbaum

In Unterkapitel 4.2 haben wir bereits intuitiv erläutert, wozu die gespeicherten Daten an jedem Knoten des Baumes nützlich sind, und davon abgeleitet, welche Information gespeichert werden muß. Wir bezeichnen mit der Notation $\langle \ldots \rangle$ die an einem Knoten gespeicherten Daten. In Abhängigkeit davon, ob es sich um einen inneren oder um einen Blattknoten handelt, wird unterschiedliche Information gespeichert:

- handelt es sich um einen inneren Knoten n, dann speichere die Werte:
  - $\langle |\lambda_n| \rangle$: die Größe der Fuzzy-Menge λ_n
  - $\langle |\lambda_n|c_k| \rangle$: die Größe der Fuzzy-Mengen $\lambda_n|c_k$ für jede Klasse $c_k \in \mathcal{C}$
  - $\langle |\lambda_n|cond_d^{t^\star}| \rangle$: die Größe der Fuzzy-Mengen $\lambda_n|cond_d^{t^\star}$ für jede Bedingung $cond_d^{t^\star}$, $1 \leq d \leq u_{t^\star}$ jedes zulässigen Tests $t^\star \in \mathcal{T}$
  - $\langle |\lambda_n|cond_d^{t^\star}|c_k \rangle$: die Größe der Fuzzy-Mengen $\lambda_n|cond_d^{t^\star}|c_k$ für jede Bedingung $cond_d^{t^\star}, 1 \leq d \leq u_{t^\star}$ jedes zulässigen Tests $t^\star \in \mathcal{T}$ und jede Klasse $c_k \in \mathcal{C}$
  - $\langle |\lambda_{n,t^\star}^{missing}| \rangle$: die Größe der Menge der unvollständigen Beispiele $\lambda_{n,t^\star}^{missing}$ bezüglich jedes zulässigen Tests $t^\star \in \mathcal{T}$
  - eine Markierung, die gesetzt ist, wenn nach der letzten Optimierung die gespeicherten Daten geändert wurden, also nicht mehr garantiert werden kann, daß der Test korrekt anhand der gespeicherten Daten ausgewählt wurde und die Abbruchbedingungen tatsächlich nicht erfüllt sind
- handelt es sich um einen Blattknoten n, dann speichere:
  - $\langle \lambda_n(f) \rangle$: der Zugehörigkeitsgrad $\lambda_n(f)$ jedes Beispiels $f \in \mathcal{L}$, das den Blattknoten erreicht hat, zum Knoten n

[1] Unter einem leeren Nachfolgerknoten verstehen wir einen leeren Blattknoten, wo die gespeicherte Information mit Null initialisiert ist.

- $\langle|\lambda_n|\rangle$: die Größe der Fuzzy-Menge λ_n
- $\langle|\lambda_n|c_k|\rangle$: die Größe der Fuzzy-Mengen $\lambda_n|c_k$ für jede Klasse $c_k \in \mathcal{C}$

Wir erweitern die Definition 3.3 eines Fuzzy-Entscheidungsbaumes zu der Definition eines inkrementellen Fuzzy-Entscheidungsbaumes.

Definition 5.1 *Ein* **inkrementeller Fuzzy-Entscheidungsbaum** *IFT bezüglich einer Lernmenge $\mathcal{L}$, einer Klassenmenge $\mathcal{C}$ und einer Menge von zulässigen Tests $\mathcal{T}$ ist ein Baum $IFT = (N, E, L, S)$ mit der Knotenmenge N, der Kantenmenge $E \subseteq N \times N$, einer Beschreibungsfunktion L, die die Knoten und die Kanten des Baumes beschriftet, und einer Funktion S, die bestimmt, welche Werte an einem Knoten gespeichert werden, wobei die nachfolgenden Bedingungen erfüllt sind:*

1. *Für jeden Knoten $n \in N$:*

$$L(n) = \begin{cases} t_n \in \mathcal{T} & \text{wenn } n \text{ ein innerer Knoten ist} \\ (\mu_k)_{k=1,2,\ldots,q} \in ([0,1])^q & \text{sonst} \end{cases}$$

2. *Für jede Kante $e = (n, n.cond_g^{t_n}) \in E : L(e) = cond_g^{t_n}$*

3. *Für jeden inneren Knoten $n \in N$:*

$$S(n) = \langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle, \langle|\lambda_n|cond_d^{t^\star}|\rangle, \langle|\lambda_n|cond_d^{t^\star}|c_k|\rangle, \langle|\lambda_{n,t^\star}^{missing}|\rangle$$

für jede Bedingung $cond_d^{t^\star}, 1 \leq d \leq u_{t^\star}$ jedes Tests $t^\star \in \mathcal{T}$ und jede Klasse $c_k \in \mathcal{C}$

4. *Für jeden Blattknoten $n \in N$:*

$$S(n) = \langle\lambda_n(f)\rangle, \langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle$$

für jedes Beispiel $f \in \mathcal{L}$, das den Blattknoten erreicht hat[2] und jede Klasse $c_k \in \mathcal{C}$

5. *Jeder innere Knoten n ist mit einer Markierung $true$ oder $false$ versehen.*

6. *Je zwei voneinander verschiedene Knoten n', n entlang eines Weges von der Wurzel zu einem Blattknoten haben verschiedene Bezeichnungen, d.h. $L(n') \neq L(n)$.*

Definition 5.2 *Unter einem durch den Algorithmus $IIFDT$ induzierten inkrementellen Fuzzy-Entscheidungsbaum bezüglich der Lernmenge $\mathcal{L}$, der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ verstehen wir einen Entscheidungsbaum nach Definition 5.1, der wie folgt entstanden ist:*

- *jedes Beispiel $f \in \mathcal{L}$ wird durch die Funktion* **addExampleToTree** *hinzugefügt (die Reihenfolge der Beispiele ist ohne Bedeutung)*
- *zwischendurch kann beliebig oft mit* **ensureBestTest** *optimiert werden*
- *nach dem Hinzufügen des letzten Beispiels aus $\mathcal{L}$ muß die Funktion* **ensureBestTest** *ausgeführt werden*

[2]Für die weitere Arbeit betrachten wir $\langle\lambda_n(f)\rangle = 0$, falls für das Beispiel $f \in \mathcal{L}$ kein Wert am Blattknoten n gespeichert ist.

Wir bemerken, daß für einen inkrementellen Fuzzy-Entscheidungsbaum die Zugehörigkeitsgrade der Beispiele aus $\mathcal{L}$ nur an den Blattknoten gespeichert sind. Für jeden inneren Knoten n des Baumes ergibt sich durch rekursive Summation ein Zugehörigkeitsgrad $\langle\langle\lambda_n(f)\rangle\rangle$ eines Beispiels $f \in \mathcal{L}$ wie folgt:

Definition 5.3 *Sei ein inkrementeller Fuzzy-Entscheidungsbaum bezüglich der Menge von Beispielen $\mathcal{L}$ gegeben. Anhand dieses Baumes können wir für jedes Beispiel $f \in \mathcal{L}$ (das zu dem Baum hinzugefügt worden ist) den* **Repräsentationsgrad** *dieses Beispiels zu einem Knoten n (d.h., den Zugehörigkeitsgrad des Beispiels zum Knoten n gemäß des Baumes) wie folgt ausrechnen:*

- *für jeden inneren Knoten n mit dem Test t_n definieren wir:*

$$\langle\langle\lambda_n(f)\rangle\rangle = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \langle\langle\lambda_{n.cond_g^{t_n}}(f)\rangle\rangle$$

- *für jeden Blattknoten n definieren wir:*

$$\langle\langle\lambda_n(f)\rangle\rangle = \langle\lambda_n(f)\rangle$$

Während der Ausführung des Algorithmus IIFDT ist oft nur ein Teil des inkrementellen Fuzzy-Entscheidungsbaumes zu sehen. Deswegen definieren wir den Begriff Teilbaum eines inkrementellen Fuzzy-Entscheidungsbaumes.

Definition 5.4 *Sei ein inkrementeller Fuzzy-Entscheidungsbaum IFT bezüglich der Lernmenge $\mathcal{L}$, einer Klassenmenge $\mathcal{C}$ und einer Menge von zulässigen Tests $\mathcal{T}$ gegeben. Den* **Teilbaum dieses inkrementellen Fuzzy-Entscheidungsbaumes**, *der an einem Knoten n des Baumes IFT beginnt, bezeichnen wir mit $T(n)$.*

Wird für den Knoten n der Wurzelknoten des inkrementellen Fuzzy-Entscheidungsbaumes eingesetzt, so erhalten wir den Gesamtbaum nach Definition 5.1.

Da jeder inkrementelle Fuzzy-Entscheidungsbaum kompositorisch ist, werden die Berechnungen innerhalb der Methode $\mathcal{M}$ und der Funktion $\mathcal{F}$ nach Hilfssatz 4.5 auf Seite 37 vereinfacht. Dabei werden stets die gespeicherten Werten an einem Knoten verwendet, d.h. benötigen wir beispielsweise die Größe der Fuzzy-Menge λ_n, so wird der gespeicherte Wert $\langle|\lambda_n|\rangle$ eingesetzt[3].

Korollar 5.5 *Bei der Ausführung der Methode $\mathcal{M}$ und der Funktion $\mathcal{F}$ im Rahmen des Algorithmus IIFDT werden für die Größen $\Lambda_{n,\mathcal{C}}, \Lambda_{n,t^\star}, \Lambda_{n,cond_d^{t^\star},\mathcal{C}}$ an jedem Knoten n und für jeden Test $t^\star \in \mathcal{T}$ die folgenden Werte eingesetzt:*

$$\begin{aligned}\Lambda_{n,\mathcal{C}} &= \langle|\lambda_n|\rangle \\ \Lambda_{n,t^\star} &= \langle|\lambda_n|\rangle \\ \Lambda_{n,cond_d^{t^\star},\mathcal{C}} &= \langle|\lambda_n|cond_d^{t^\star}|\rangle\end{aligned}$$

[3] Für einen Blattknoten sind die Werte $\langle|\lambda_n|cond_d^{t^\star}|\rangle$ und $\langle|\lambda_n|cond_d^{t^\star}|c_k|\rangle$ nicht vorhanden. Sie müssen explizit aus den gespeicherten Werten $\langle\lambda_n(f)\rangle$ ausgerechnet werden.

Die benötigten Wahrscheinlichkeitsverteilungen ergeben sich nach Korollar 4.6 zu:

$$\begin{aligned}
W(\lambda_n) &= \left(\frac{\langle|\lambda_n|c_1|\rangle}{\langle|\lambda_n|\rangle}, \frac{\langle|\lambda_n|c_2|\rangle}{\langle|\lambda_n|\rangle}, \ldots, \frac{\langle|\lambda_n|c_q|\rangle}{\langle|\lambda_n|\rangle}\right) \\
W(t^\star, \lambda_n) &= \left(\frac{\langle|\lambda_n|cond_1^{t^\star}|\rangle}{\langle|\lambda_n|\rangle}, \frac{\langle|\lambda_n|cond_2^{t^\star}|\rangle}{\langle|\lambda_n|\rangle}, \ldots, \frac{\langle|\lambda_n|cond_{u_{t^\star}}^{t^\star}|\rangle}{\langle|\lambda_n|\rangle}\right) \\
W(\lambda_n|cond_d^{t^\star}) &= \left(\frac{\langle|\lambda_n|cond_d^{t^\star}|c_1|\rangle}{\langle|\lambda_n|cond_d^{t^\star}|\rangle}, \frac{\langle|\lambda_n|cond_d^{t^\star}|c_2|\rangle}{\langle|\lambda_n|cond_d^{t^\star}|\rangle}, \ldots, \frac{\langle|\lambda_n|cond_d^{t^\star}|c_q|\rangle}{\langle|\lambda_n|cond_d^{t^\star}|\rangle}\right)
\end{aligned}$$

5.2 Die einzelnen Funktionen des Algorithmus

5.2.1 Hinzufügen eines Beispiels zu einem inkrementellen Fuzzy-Entscheidungsbaum

Input	f	das Beispiel f, d.h. die Zugehörigkeitsgrade von f zu den Fuzzy-Mengen $\mu_{\mathrm{v}_{il}}$ und μ_{c_k}
	$\Delta_{n,f} \in [0,1]$	der Wert, um den der Zugehörigkeitsgrad des Beispiels f zum Knoten n erhöht werden soll
	n	Wurzelknoten eines Teilbaumes $T(n)$ eines inkrementellen Fuzzy-Entscheidungsbaumes
	$\mathcal{T}' \subseteq \mathcal{T}$	die Menge der verfügbaren Tests, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten *root* des Fuzzy-Entscheidungsbaumes zum Knoten n liegen
Output	$T_a(n_a)$	aktualisierter Teilbaum beginnend am aktualisierten Knoten n_a

addExampleToTree($f, \Delta_{n,f}, n, \mathcal{T}'$)

1. ist $\Delta_{n,f}$ gleich Null, dann $\rightarrow$ **stop**
2. ist der Knoten n ein Blattknoten?

   ja $\rightarrow$ 3.
   nein $\rightarrow$ 4.

3. füge das Beispiel f zu dem Blattknoten n mit $\Delta_{n,f} \in [0,1]$ hinzu (den aktualisierten Knoten n bezeichnen wir mit n_a), d.h. speichere die Werte:
   (a) $\langle\lambda_{n_a}(f)\rangle = \langle\lambda_n(f)\rangle + \Delta_{n,f}$
   (b) $\langle|\lambda_{n_a}|\rangle = \langle|\lambda_n|\rangle + \Delta_{n,f}$
   (c) $\langle|\lambda_{n_a}|c_k|\rangle = \langle|\lambda_n|c_k|\rangle + \top(\Delta_{n,f}, \mu_{c_k}(f))$, für jede Klasse $c_k \in \mathcal{C}$
   (d) wenn die Funktion $\mathcal{F}(\langle|\lambda_{n_a}|\rangle, \langle|\lambda_{n_a}|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ für den aktuellen Knoten n_a erfüllt ist, dann beschrifte n_a mit dem ausgerechneten Tupel von Klassenzugehörigkeiten gemäß $\mathcal{F}$ und
   $\rightarrow$**stop**
   ansonsten wandle den Knoten n_a in einen inneren Knoten um:
      i. anhand der Werte $\langle\lambda_{n_a}(f')\rangle$, die für jedes Beispiel $f' \in \mathcal{L}$, das den Blattknoten n_a erreicht hat, gespeichert sind, berechne und speichere die Größen

der Fuzzy-Mengen:

$$\begin{aligned}
\langle|\lambda_{n_a}|cond_d^{t^\star}|\rangle &= \sum_{f'} \top(\langle\lambda_{n_a}(f')\rangle, \mu_{cond_d^{t^\star}}(f')) \\
\langle|\lambda_{n_a}|cond_d^{t^\star}|c_k|\rangle &= \sum_{f'} \top(\langle\lambda_{n_a}(f')\rangle, \mu_{cond_d^{t^\star}}(f'), \mu_{c_k}(f')) \\
\langle|\lambda_{n_a,t^\star}^{missing}|\rangle &= \sum_{f'} \begin{cases} \langle\lambda_{n_a}(f')\rangle & \text{wenn } f' \text{ unvollständiges Beispiel bezüglich } t^\star \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

für jede Bedingung $cond_d^{t^\star}, 1 \leq d \leq u_{t^\star}$ jedes Tests $t^\star \in \mathcal{T}$ und für jede Klasse $c_k \in \mathcal{C}$

ii. wähle den besten Test $t = [cond_1^t, cond_2^t, \ldots, cond_{u_t}^t] \in \mathcal{T}'$ durch die Methode $\mathcal{M}$, unter Benutzung der gespeicherten Werte und beschrifte den aktuellen Knoten n_a mit t

iii. für jede Bedingung $cond_h^t$ aus t füge eine ausgehende Kante mit einem leeren Nachfolgerknoten $n_a.cond_h^t$ an

iv. für jedes Beispiel $f' \in \mathcal{L}$, das am Knoten n_a gespeichert ist, wiederhole:

A. für jeden Nachfolgerknoten $n_a.cond_h^t, 1 \leq h \leq u_t$ von n_a wiederhole:

- berechne den Wert (nach Definition 3.8 auf Seite 20), um den der Zugehörigkeitsgrad des Beispiels f' zum Knoten $n_a.cond_h^t$ erhöht werden muß (das ist genau die Zugehörigkeit des Beispiels f' zum Knoten $n_a.cond_h^t$):

$$\Delta_{n_a.cond_h^t, f'} = \lambda_{n_a.cond_h^t}(f') = \top(\langle\lambda_{n_a}(f')\rangle, \mu_{cond_h^t}(f'))$$

- führe

addExampleToTree$(f', \Delta_{n_a.cond_h^t, f'}, n_a.cond_h^t, \mathcal{T}' \setminus \{t\})$

durch

B. lösche den Wert $\langle\lambda_{n_a}(f')\rangle$

→ **stop**

4. füge das Beispiel f zu dem inneren Knoten n mit $\Delta_{n,f}$ hinzu (den aktualisierten Knoten n bezeichnen wir mit n_a), d.h. speichere die Werte

(a) $\langle|\lambda_{n_a}|\rangle = \langle|\lambda_n|\rangle + \Delta_{n,f}$

(b) $\langle|\lambda_{n_a}|c_k|\rangle = \langle|\lambda_n|c_k|\rangle + \top(\Delta_{n,f}, \mu_{c_k}(f))$, für jede Klasse $c_k \in \mathcal{C}$

(c) $\langle|\lambda_{n_a}|cond_d^{t^\star}|\rangle = \langle|\lambda_n|cond_d^{t^\star}|\rangle + \top(\Delta_{n,f}, \mu_{cond_d^{t^\star}}(f))$, für jede Bedingung $cond_d^{t^\star}$, $1 \leq d \leq u_{t^\star}$ jedes Tests $t^\star \in \mathcal{T}$

(d) $\langle|\lambda_{n_a}|cond_d^{t^\star}|c_k|\rangle = \langle|\lambda_n|cond_d^{t^\star}|c_k|\rangle + \top(\Delta_{n,f}, \mu_{cond_d^{t^\star}}(f), \mu_{c_k}(f))$, für jede Bedingung $cond_d^{t^\star}, 1 \leq d \leq u_{t^\star}$ jedes Tests $t^\star \in \mathcal{T}$ und für jede Klasse $c_k \in \mathcal{C}$

(e) $\langle|\lambda_{n_a,t^\star}^{missing}|\rangle = \langle|\lambda_{n,t^\star}^{missing}|\rangle + \begin{cases} \Delta_{n,f} & \text{wenn } f \text{ unvollständiges Beispiel bezüglich } t^\star \\ 0 & \text{sonst} \end{cases}$

für jeden Test $t^\star \in \mathcal{T}$

(f) markiere den Knoten n_a

(g) sei $t_n = [cond_1^{t_n}, cond_2^{t_n}, \ldots, cond_{u_{t_n}}^{t_n}] \in \mathcal{T}'$ der Test des aktuellen Knotens n und somit auch von n_a. Für jeden Nachfolgerknoten $n.cond_g^{t_n}, 1 \leq g \leq u_{t_n}$ von n_a wiederhole:

- berechne den Wert (nach Definition 3.8 auf Seite 20), um den der Zugehörigkeitsgrad des Beispiels f zum Knoten $n.cond_g^{t_n}$ erhöht werden muß:

$$\Delta_{n.cond_g^{t_n},f} = \top(\Delta_{n,f}, \mu_{cond_g^{t_n}}(f))$$

- führe

  addExampleToTree$(f, \Delta_{n.cond_g^{t_n},f}, n.cond_g^{t_n}, \mathcal{T}' \setminus \{t_n\})$

  durch

$\rightarrow$ **stop**

5.2.2 Austausch des Tests an einem Knoten

Input	n	Wurzelknoten eines Teilbaumes $T(n)$ eines inkrementellen Fuzzy-Entscheidungsbaumes, wobei n ein innerer Knoten des Baumes ist
	$t_n = [cond_1^{t_n}, \ldots, cond_{u_{t_n}}^{t_n}] \in \mathcal{T}'$	der Test am Knoten n
	$t = [cond_1^{t}, \ldots, cond_{u_t}^{t}] \notin \mathcal{T}'$	der gewünschte Test für den Knoten n, d.h. der Test, der am Knoten n gesetzt werden soll
	$\mathcal{T}' \subseteq \mathcal{T} \setminus \{t\}$	die Menge der verfügbaren Tests, d.h. die Tests (außer t), die nicht auf dem Weg vom Wurzelknoten *root* des Fuzzy-Entscheidungsbaumes zum Knoten n liegen
Output	$T_t(n_t)$	aktualisierter Teilbaum, dessen Wurzelknoten n_t mit dem gewünschten Test t beschriftet ist

transposition$(n, t_n, t, \mathcal{T}')$

1. Untersuche die Menge

$$\mathcal{N}(n) = \{n.cond_1^{t_n}, n.cond_2^{t_n}, \ldots, n.cond_{u_{t_n}}^{t_n}\}$$

   der Nachfolgerknoten von n und unterscheide folgende Fälle:

   - jeder Nachfolgerknoten $n.cond_g^{t_n} \in \mathcal{N}(n)$ ist ein Blattknoten $\rightarrow$ 2.
   - genau ein Nachfolgerknoten $n.cond_{g^\star}^{t_n} \in \mathcal{N}(n)$ ist kein Blattknoten $\rightarrow$ 3.
   - es gibt zwei oder mehrere Nachfolgerknoten $n.cond_{g^\star}^{t_n} \in \mathcal{N}(n)$, die innere Knoten sind, sowie mindestens einen Blattknoten $n.cond_g^{t_n} \in \mathcal{N}(n) \rightarrow$ 4.
   - jeder Nachfolgerknoten $n.cond_g^{t_n} \in \mathcal{N}(n)$ ist ein innerer Knoten $\rightarrow$ 5.

2. (a) berechne (nach Definition 5.3 auf Seite 42) den Repräsentationsgrad jedes Beispiels $f \in \mathcal{L}$ (das betrachtet wurde) zum Knoten n:

$$\langle\langle\lambda_n(f)\rangle\rangle = \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \langle\lambda_{n.cond_g^{t_n}}(f)\rangle$$

   (b) entferne alle Nachfolgerknoten des Knotens n

(c) beschrifte den Knoten n mit dem gewünschten Test t
$\rightarrow$ den so modifizierten Knoten n bezeichnen wir mit n_t

(d) markiere den Knoten n_t

(e) für jede Bedingung $cond_h^t$ aus t füge eine ausgehende Kante mit einem leeren Nachfolgerknoten $n_t.cond_h^t$ an

(f) für jedes Beispiel $f \in \mathcal{L}$ mit $\langle\langle\lambda_n(f)\rangle\rangle > 0$ wiederhole:

i. für jeden Nachfolgerknoten $n_t.cond_h^t, 1 \leq h \leq u_t$ von n_t wiederhole:

- berechne den Wert (nach Definition 3.8 auf Seite 20), um den der Zugehörigkeitsgrad des Beispiels f zum Knoten $n_t.cond_h^t$ erhöht werden muß (dies entspricht genau der Zugehörigkeit des Beispiels f zum Knoten $n_t.cond_h^t$):

$$\Delta_{n_t.cond_h^t,f} = \lambda_{n_t.cond_h^t}(f) = \top(\langle\langle\lambda_n(f)\rangle\rangle, \mu_{cond_h^t}(f))$$

- führe

$$\textbf{addExampleToTree(}f, \Delta_{n_t.cond_h^t,f}, n_t.cond_h^t, \mathcal{T}'\textbf{)}$$

durch

$\rightarrow$ **stop**

3. (a) ist der Test $t_{n.cond_{g^\star}^{t_n}}$ für den inneren Nachfolgerknoten $n.cond_{g^\star}^{t_n}$ nicht zu dem gewünschten Test t identisch, dann führe

$$\textbf{transposition(}n.cond_{g^\star}^{t_n}, t_{n.cond_{g^\star}^{t_n}}, t, \mathcal{T}'\textbf{)}$$

durch, um dies zu erreichen.

Jetzt bezeichnen wir die Menge der Nachfolgerknoten des Knotens $n.cond_{g^\star}^{t_n}$ durch

$$\mathcal{N}(n.cond_{g^\star}^{t_n}) = \{n.cond_{g^\star}^{t_n}.cond_1^t, n.cond_{g^\star}^{t_n}.cond_2^t, \ldots, n.cond_{g^\star}^{t_n}.cond_{u_t}^t\}$$

(b) ersetze den Knoten n durch $n_t = n.cond_{g^\star}^{t_n}$

(c) für jeden Knoten $n.cond_g^{t_n} \in \mathcal{N}(n) \setminus \{n.cond_{g^\star}^{t_n}\}$ (diese sind Blattknoten) wiederhole:

i. für jedes Beispiel $f \in \mathcal{L}$, das am Knoten $n.cond_g^{t_n}$ gespeichert ist, wiederhole:

- für den Wert, um den der Zugehörigkeitsgrad des Beispiels f zum Knoten n_t erhöht werden muß, gilt:

$$\Delta_{n_t,f} = \langle\lambda_{n.cond_g^{t_n}}(f)\rangle$$

- führe

$$\textbf{addExampleToTree(}f, \Delta_{n_t,f}, n_t, \mathcal{T}' \cup \{t\}\textbf{)}$$

durch

$\rightarrow$ **stop**

4. (a) für jeden Blattknoten $n.cond_g^{t_n} \in \mathcal{N}(n)$ wiederhole:
   i. anhand der Werte $\langle\lambda_{n.cond_g^{t_n}}(f)\rangle$, die für jedes Beispiel $f \in \mathcal{L}$, das den Blattknoten $n.cond_g^{t_n}$ erreicht hat, gespeichert sind, berechne und speichere die Größen der Fuzzy-Mengen:

$$\begin{aligned}
\langle|\lambda_{n.cond_g^{t_n}}|cond_d^{t^\star}|\rangle &= \sum_f \top(\langle\lambda_{n.cond_g^{t_n}}(f)\rangle, \mu_{cond_d^{t^\star}}(f)) \\
\langle|\lambda_{n.cond_g^{t_n}}|cond_d^{t^\star}|c_k|\rangle &= \sum_f \top(\langle\lambda_{n.cond_g^{t_n}}(f)\rangle, \mu_{cond_d^{t^\star}}(f), \mu_{c_k}(f)) \\
\langle|\lambda^{missing}_{n.cond_g^{t_n},t^\star}|\rangle &= \sum_f \begin{cases} \langle\lambda_{n.cond_g^{t_n}}(f)\rangle & \text{wenn } f \text{ unvollständiges} \\ & \text{Beispiel bezüglich } t^\star \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

   für jede Bedingung $cond_d^{t^\star}, 1 \le d \le u_{t^\star}$ jedes Tests $t^\star \in \mathcal{T}$ und für jede Klasse $c_k \in \mathcal{C}$
   ii. beschrifte den Knoten $n.cond_g^{t_n}$ mit dem gewünschten Test t
   iii. markiere den Knoten $n.cond_g^{t_n}$
   iv. für jede Bedingung $cond_h^t$ aus t füge eine ausgehende Kante mit einem leeren Nachfolgerknoten $n.cond_g^{t_n}.cond_h^t$ an
   v. für jedes Beispiel f, das am Knoten $n.cond_g^{t_n}$ gespeichert ist, wiederhole:
      A. für jeden Nachfolgerknoten $n.cond_g^{t_n}.cond_h^t, 1 \le h \le u_t$ von $n.cond_g^{t_n}$ wiederhole:
         - berechne den Wert (nach Definition 3.8 auf Seite 20), um den der Zugehörigkeitsgrad des Beispiels f zum Knoten $n.cond_g^{t_n}.cond_h^t$ erhöht werden muß (dies entspricht genau der Zugehörigkeit des Beispiels f zum Knoten $n.cond_g^{t_n}.cond_h^t$):

$$\Delta_{n.cond_g^{t_n}.cond_h^t,f} = \lambda_{n.cond_g^{t_n}.cond_h^t}(f) = \top(\langle\lambda_{n.cond_g^{t_n}}(f)\rangle, \mu_{cond_h^t}(f))$$

         - führe **addExampleToTree**$(f, \Delta_{n.cond_g^{t_n}.cond_h^t,f}, n.cond_g^{t_n}.cond_h^t, \mathcal{T}' \setminus \{t_n\})$ durch
      B. lösche den Wert $\langle\lambda_{n.cond_g^{t_n}}(f)\rangle$

   (b) gehe zu 5

5. (a) für jeden Knoten $n.cond_g^{t_n}$ aus $\mathcal{N}(n)$ wiederhole:
   - ist der Test $t_{n.cond_g^{t_n}}$ dieses Knotens nicht zu dem Test t identisch, dann führe

     transposition$(n.cond_g^{t_n}, t_{n.cond_g^{t_n}}, t, \mathcal{T}' \setminus \{t_n\})$

     durch, um dies zu erreichen.

   Jetzt ist jeder Nachfolgerknoten $n.cond_g^{t_n} \in \mathcal{N}(n)$ mit dem gewünschten Test t beschriftet und die Menge seiner Nachfolgerknoten ergibt sich zu:

$$\mathcal{N}(n.cond_g^{t_n}) = \{n.cond_g^{t_n}.cond_1^t, n.cond_g^{t_n}.cond_2^t, \ldots, n.cond_g^{t_n}.cond_{u_t}^t\}$$

   (b) entferne die Nachfolgerknoten von n
   (c) beschrifte den Knoten n mit dem gewünschten Test t
   (d) markiere den Knoten n
   $\rightarrow$ den so modifizierten Knoten n bezeichnen wir mit n_t

(e) für jede Bedingung $cond_h^t$ aus t füge zu dem Knoten n_t eine ausgehende Kante mit einem leeren Nachfolgerknoten $n_t.cond_h^t$ an

(f) für jeden Knoten $n_t.cond_h^t \in \mathcal{N}(n_t)$ wiederhole:

i. markiere den Knoten als modifiziert

ii. beschrifte den Knoten mit dem Test t_n

iii. für jede Bedingung $cond_g^{t_n}$ aus t_n füge zu dem Knoten $n_t.cond_h^t$ eine ausgehende Kante mit einem Nachfolgerknoten $n_t.cond_h^t.cond_g^{t_n}$ an. Dieser Knoten entspricht genau dem alten Knoten $n.cond_g^{t_n}.cond_h^t \in \mathcal{N}(n.cond_g^{t_n})$.

iv. am Knoten $n_t.cond_h^t$ speichere die Werte:

$$\begin{aligned}
\langle|\lambda_{n_t.cond_h^t}|\rangle &= \sum_{g=1}^{u_{t_n}} \langle|\lambda_{n.cond_g^{t_n}.cond_h^t}|\rangle \\
\langle|\lambda_{n_t.cond_h^t}|c_k|\rangle &= \sum_{g=1}^{u_{t_n}} \langle|\lambda_{n.cond_g^{t_n}.cond_h^t}|c_k|\rangle \\
\langle|\lambda_{n_t.cond_h^t}|cond_d^{t^\star}|\rangle &= \sum_{g=1}^{u_{t_n}} \langle|\lambda_{n.cond_g^{t_n}.cond_h^t}|cond_d^{t^\star}|\rangle \\
\langle|\lambda_{n_t.cond_h^t}|cond_d^{t^\star}|c_k|\rangle &= \sum_{g=1}^{u_{t_n}} \langle|\lambda_{n.cond_g^{t_n}.cond_h^t}|cond_d^{t^\star}|c_k|\rangle \\
\langle|\lambda^{missing}_{n_t.cond_h^t,t^\star}|\rangle &= \sum_{g=1}^{u_{t_n}} \langle|\lambda^{missing}_{n.cond_g^{t_n}.cond_h^t,t^\star}|\rangle
\end{aligned}$$

→ **stop**

5.2.3 Optimierung eines inkrementellen Fuzzy-Entscheidungsbaumes

Input	n	Wurzelknoten eines Teilbaumes $T(n)$ eines inkrementellen Fuzzy-Entscheidungsbaumes
	$\mathcal{T}' \subseteq \mathcal{T}$	die Menge der verfügbaren Tests, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten *root* des Fuzzy-Entscheidungsbaumes zum Knoten n liegen
Output	$T_e(n_e)$	optimierter Teilbaum beginnend am Knoten n_e

ensureBestTest$(n, \mathcal{T}')$

1. ist der Knoten n ein Blattknoten, dann → **stop**

2. wenn die Funktion $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ für den aktuellen Knoten erfüllt ist, dann definiere ihn als Blattknoten n_e (sonst → 3):

   (a) beschrifte n_e mit dem durch die Funktion $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ ausgerechneten Tupel von Klassenzugehörigkeiten

   (b) für jedes Beispiel $f \in \mathcal{L}$ berechne und speichere den Zugehörigkeitsgrad dieses Beispiels zum Knoten n_e:

$$\langle\lambda_{n_e}(f)\rangle = \langle\langle\lambda_n(f)\rangle\rangle$$

(c) lösche die gespeicherten Werte $\langle \lambda_n | cond_d^{t^\star} \rangle$, $\langle \lambda_n | cond_d^{t^\star} | c_k \rangle$, $\langle \lambda_{n,t^\star}^{missing} \rangle$ für jede Bedingung $cond_d^{t^\star}$ jedes Tests $t^\star \in \mathcal{T}$ und jede Klasse $c_k \in \mathcal{C}$

(d) entferne alle Nachfolgerknoten von n

(e) lösche die Markierung des Knotens

$\rightarrow$ **stop**

3. ist der Knoten nicht markiert, dann bezeichne ihn mit n_e und gehe zu 7.
4. wähle mit Hilfe der am Knoten n gespeicherten Information den besten Test $t = [cond_1^t, cond_2^t, \ldots, cond_{u_t}^t] \in \mathcal{T}'$ bezüglich der Methode $\mathcal{M}$
5. ist der Test t_n am Knoten n nicht zu dem ausgewählten Test t identisch, dann führe

$$\mathbf{transposition}(n, t_n, t, \mathcal{T}' \setminus \{t\})$$

durch und bezeichne das Resultat als n_e

6. setze die Markierung des Knotens n_e zurück
7. am Knoten n_e steht der beste Test t, für jeden Nachfolgerknoten $n_e.cond_h^t, 1 \leq h \leq u_t$ von n_e führe

$$\mathbf{ensureBestTest}(n_e.cond_h^t, \mathcal{T}' \setminus \{t\})$$

durch

$\rightarrow$ **stop**

5.3 Beispielinduktion

In diesem Unterkapitel geben wir eine Beispielinduktion an, die mit Hilfe der Implementierung des Algorithmus IIFDT im Projekt ISeC durchgeführt wurde. Sie hat als Ziel, zu veranschaulichen, wie ein inkrementeller Fuzzy-Entscheidungsbaum durch das Hinzufügen von Beispielen entsteht und wie dieser zum Schluß optimiert wird. Zur Vereinfachung werden wir an den Bilder nur $\langle |\lambda_n| \rangle$ an allen Knoten und $\langle \lambda_n(f) \rangle$ an den Blattknoten notieren (aus Platzgründe kann $\langle . \rangle$ weggelassen werden).

Der Algorithmus IIFDT wurde unter den gleichen Voraussetzungen ausgeführt, wie der Algorithmus IFDT im Beispiel 3.11 auf Seite 28. Als Ergebnis erhalten wir einen inkrementellen Fuzzy-Entscheidungsbaum bezüglich der Lernmenge $\mathcal{L} = \{1, 2, \ldots, 6\}$ aus Tabelle 3.3 auf Seite 21, der Klassenmenge $\mathcal{C} = \{c_1, c_2\}$ und der Menge der zulässigen Tests $\mathcal{T} = \{t_1, t_2, \ldots, t_6\}$.

Beginnen wir mit dem Hinzufügen des ersten Beispiels mit dem Index 1. Zu diesem Zeitpunkt existiert kein Entscheidungsbaum, also wird ein leerer Wurzelknoten *root* generiert (die gespeicherten Werte sind mit Null initialisiert). Das Beispiel wird zu der Menge der betrachteten Beispiele hinzugefügt, sowie die Mengen $\mu_{\mathrm{v}_{il}}$ und μ_{c_k} werden erweitert.

Das Beispiel muß zu dem Fuzzy-Entscheidungsbaum hinzugefügt werden und dadurch den Zugehörigkeitsgrad des Beispiels zu dem Baum um Eins erhöhen (jedes Beispiel gehört zu dem Wurzelknoten mit Eins). Um dies zu erreichen, wird die Funktion (siehe Seite 43):

$$\mathbf{addExampleToTree}(1, \Delta_{root,1}, root, \mathcal{T})$$

mit $\Delta_{root,1} = 1$ aufgerufen.

Da *root* ein Blattknoten ist, wird Schritt 3 ausgeführt. Nach der Aktualisierung der gespeicherten Größen erhalten wir:

$$\begin{array}{rcccccl} \langle\lambda_{root_a}(1)\rangle & = & 0 & + & 1 & = & 1 \quad \text{und} \\ \langle|\lambda_{root_a}|\rangle & = & 0 & + & 1 & = & 1 \end{array}$$

Im Schritt 3d wird getestet, ob $\mathcal{F}(\langle|\lambda_{root_a}|\rangle, \langle|\lambda_{root_a}|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T})$ erfüllt ist. Da dies der Fall ist (die Abbruchbedingung θ_2 ist erfüllt), wird der Blattknoten mit dem ausgerechneten Tupel $(0,1)$ beschriftet und damit ist das Hinzufügen des ersten Beispiels beendet.

Interessanter wird es beim Hinzufügen des Beispiels mit dem Index 2. Analog werden die Lernmenge und die Fuzzy-Mengen $\mu_{\mathrm{v}_{il}}$ und μ_{c_k} um das Beispiel erweitert und die Funktion:

$$\mathbf{addExampleToTree}(2, \Delta_{root,2}, root, \mathcal{T})$$

mit $\Delta_{root,2} = 1$ aufgerufen. Da *root* immer noch ein Blattknoten ist, wird der Schritt 3 ausgeführt. Durch die Aktualisierung erhalten wir:

$$\begin{array}{rcccccl} \langle\lambda_{root_a}(2)\rangle & = & 0 & + & 1 & = & 1 \quad \text{und} \\ \langle|\lambda_{root_a}|\rangle & = & 1 & + & 1 & = & 2 \end{array}$$

Die Funktion $\mathcal{F}(\langle|\lambda_{root_a}|\rangle, \langle|\lambda_{root_a}|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T})$ ist nicht mehr erfüllt und deshalb wird im Schritt 3d der Knoten *root* in einen inneren Knoten umgewandelt. Nachdem die benötigten Größen ausgerechnet und gespeichert wurden, wird im Schritt 3(d)ii anhand der gespeicherten Werte und der Methode $\mathcal{M}$ der beste Test aus $\mathcal{T}$ ausgewählt, nämlich $t_6 = [\mathsf{Groeße} = \mathsf{klein}, \mathsf{Groeße} = \mathsf{mittel}, \mathsf{Groeße} = \mathsf{groß}]$. Im Schritt 3(d)iii wird für jede Bedingung $cond_g^{t_6}$ aus t_6 ein leerer Nachfolgerknoten $root.cond_g^{t_6}$ erzeugt (siehe Abb. 5.1).

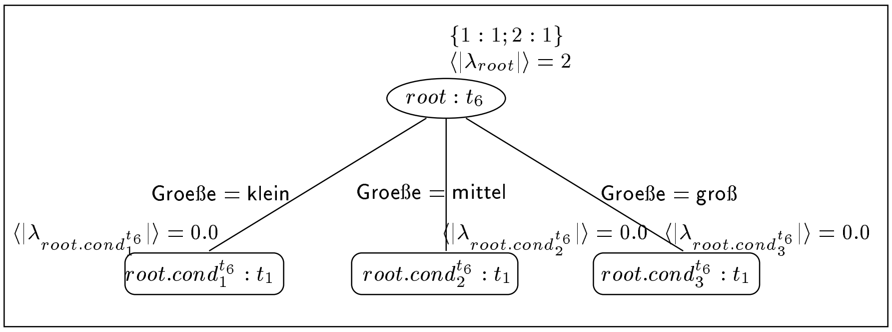

Abbildung 5.1: Inkrementeller Fuzzy-Entscheidungsbaum während der Umwandlung des Knotens *root* in einen inneren Knoten

Im Schritt 3(d)iv wird jedes Beispiel f, das an *root* gespeichert ist, zu den neuen Nachfolgern hinzugefügt.

Zuerst betrachten wir das Beispiel mit dem Index 1. Den rekursiven Aufruf von **addExampleToTree** für den ersten Nachfolger $root.cond_1^{t_6}$ brauchen wir nicht zu betrachten, da $\Delta_{root.cond_1^{t_6},1} = 0$. Der Wert $\Delta_{root.cond_2^{t_6},1}$, um den der Zugehörigkeitsgrad des Beispiels zum zweiten Nachfolgerknoten $root.cond_2^{t_6}$ erhöht werden soll, ergibt sich zu: $\Delta_{root.cond_2^{t_6},1} = \top(1, \mu_{\mathsf{mittel}}(1)) = 0.93$. Jetzt wird

$$\mathbf{addExampleToTree}(1, \Delta_{root.cond_2^{t_6},1}, root.cond_2^{t_6}, \mathcal{T} \setminus \{t_6\})$$

aufgerufen und dabei Schritt 3 ausgeführt. Für den dritten Nachfolger erhalten wir den Wert $\Delta_{root.cond_3^{t_6},1} = \top(1, \mu_{\mathsf{groß}}(1)) = 0.07$. Es wird dafür ebenfalls Schritt 3 der Funktion

addExampleToTree ausgeführt. In beiden Fällen bleibt der Knoten $root.cond_2^{t6}$ bzw. $root.cond_3^{t6}$ nach dem Hinzufügen des ersten Beispiels Blattknoten und wir speichern die Werte: $\langle\lambda_{root.cond_2^{t6}}(1)\rangle = 0.93$ und $\langle|\lambda_{root.cond_2^{t6}}|\rangle = 0.93$ bzw. $\langle\lambda_{root.cond_3^{t6}}(1)\rangle = 0.07$ und $\langle|\lambda_{root.cond_3^{t6}}|\rangle = 0.07$. Nachdem das Beispiel 1 zu allen Nachfolgerknoten hinzugefügt wurde, löschen wir den gespeicherten Wert $\langle\lambda_{root}(1)\rangle = 1$.

Für das zweite Beispiel werden wir das Hinzufügen zu den Nachfolgern $root.cond_1^{t6}$ und $root.cond_3^{t6}$ nicht betrachten. Hier stellen wir nur das Hinzufügen zu dem zweiten Nachfolger, d.h. die Ausführung von Schritt 3 der Funktion

$$\mathbf{addExampleToTree}(2, \Delta_{root.cond_2^{t6},2}, root.cond_2^{t6}, \mathcal{T}')$$

mit $\Delta_{root.cond_2^{t6},2} = \top(1, \mu_{\mathsf{mittel}}(2)) = 0.53$ und $\mathcal{T}' = \mathcal{T} \setminus \{t_6\}$ dar. Mit Aktualisierung der gespeicherten Werte erhalten wir

$$\begin{array}{rcrcrcrl} \langle\lambda_{root.cond_2^{t6}}(2)\rangle & = & 0 & + & 0.53 & = & 0.53 & \text{und} \\ \langle|\lambda_{root.cond_2^{t6}}|\rangle & = & 0.93 & + & 0.53 & = & 1.46 & \end{array}$$

Im Schritt 3d ist die Funktion $\mathcal{F}(\langle|\lambda_{root.cond_2^{t6}}|\rangle, \langle|\lambda_{root.cond_2^{t6}}|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ nicht erfüllt und der Knoten wird in einen inneren Knoten umgewandelt indem die Beispiele 1 und 2 gemäß Test t_1 zu neuen Nachfolgern verzweigt werden (siehe Abbildung 5.2).

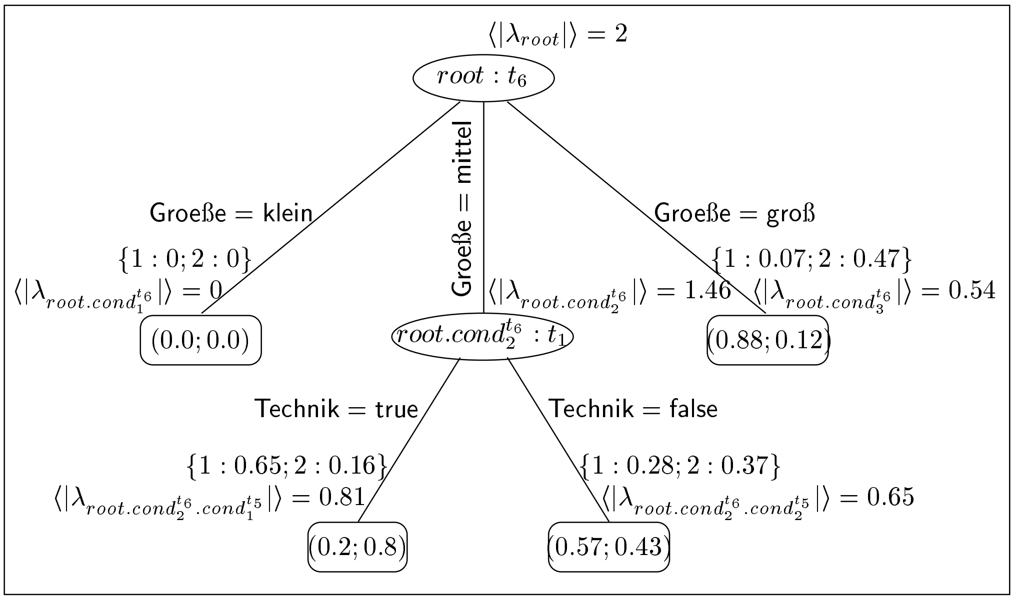

Abbildung 5.2: Inkrementeller Fuzzy-Entscheidungsbaum nach dem Hinzufügen der Beispiele mit dem Index 1 und 2

Mit dem Hinzufügen der restlichen Beispiele 3 bis 6 werden die Knoten markiert und wir erhalten den inkrementellen Fuzzy-Entscheidungsbaum aus Abbildung 5.3.

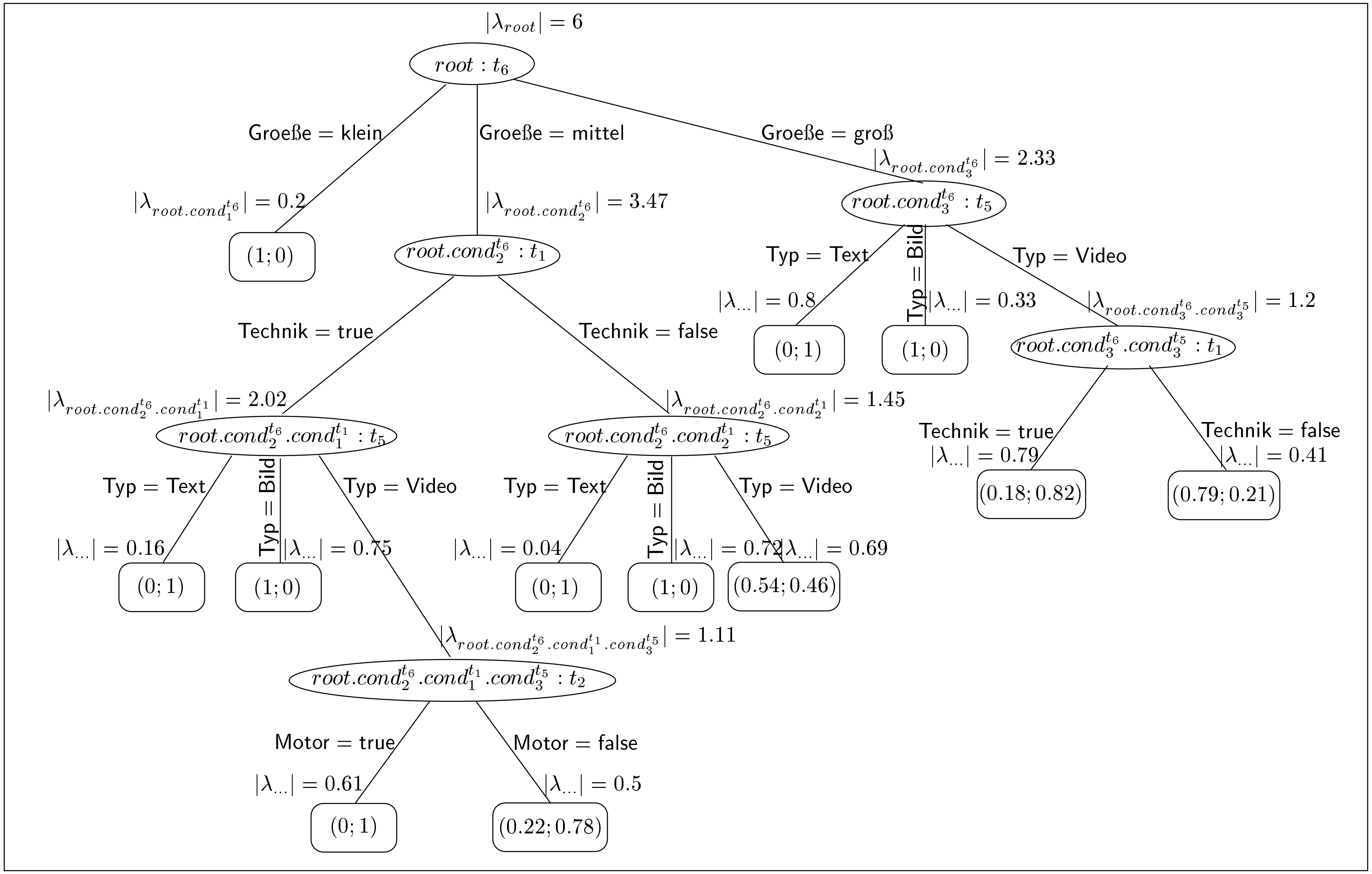

Abbildung 5.3: Inkrementeller Fuzzy-Entscheidungsbaum

Nachdem alle Beispiele aus $\mathcal{L}$ hinzugefügt wurden, muß der inkrementelle Fuzzy-Entscheidungsbaum optimiert werden. Dafür wird die Funktion (siehe Seite 48):

$$\textbf{ensureBestTest}(root, \mathcal{T})$$

ausgeführt.

Da $root$ weder bereits ein Blattknoten ist, noch $\mathcal{F}(\langle|\lambda_{root}|\rangle, \langle|\lambda_{root}|c_k|\rangle_{k=1,2,\dots,q}, \mathcal{T})$ erfüllt ist, wird sofort zum Schritt 3 übergegangen. Der Knoten ist markiert, also muß getestet werden, ob der Test t_6 tatsächlich durch die Methode $\mathcal{M}$ anhand der gespeicherten Daten ausgewählt würde. Im Schritt 4 wird der Test $t_5 = [\mathsf{Typ} = \mathsf{Text}, \mathsf{Typ} = \mathsf{Bild}, \mathsf{Typ} = \mathsf{Video}]$ als bester Test ausgewählt. Somit wird im nächsten Schritt die Funktion:

$$\textbf{transposition}(root, t_6, t_5, \mathcal{T} \setminus \{t_5\})$$

aufgerufen und dabei Schritt 4 ausgeführt. Am Blattknoten $root.cond_1^{t_6}$ wird der gewünschte Test t_5 gesetzt und somit in einen inneren Knoten umgewandelt. Da jetzt alle Nachfolger von $root$ innere Knoten sind, wird mit der Ausführung des Schritts 5 fortgesetzt. Zuerst muß mittels einem rekursiven Aufruf des Schritts 5 von **transposition** der Test t_1 am Knoten $root.cond_1^{t_6}$ durch den gewünschten Test t_5 ersetzen. Jetzt sind alle Nachfolger von $root$ mit dem Test t_5 beschriftet und wir können die Tests für $root$ und seine Nachfolger umtauschen. Wir erhalten einen inkrementellen Fuzzy-Entscheidungsbaum, dessen Wurzelknoten $root$ mit dem Test t_5 beschriftet ist und seine Nachfolger mit t_6.

Im nächsten Schritt von **ensureBestTest** wird die Markierung des Knotens $root$ zurückgesetzt und im Schritt 7 wird **ensureBestTest** rekursiv auf den neuen Nachfolgern von $root$ aufgerufen.

Nach dem Beenden der rekursiven Aufrufe erhalten wir den inkrementellen Fuzzy-Entscheidungsbaum aus Abbildung 5.4, der äquivalent zu dem Fuzzy-Entscheidungsbaum ist, den wir im Beispiel 3.11 induziert haben (vgl. Abb. 3.2 auf Seite 17).

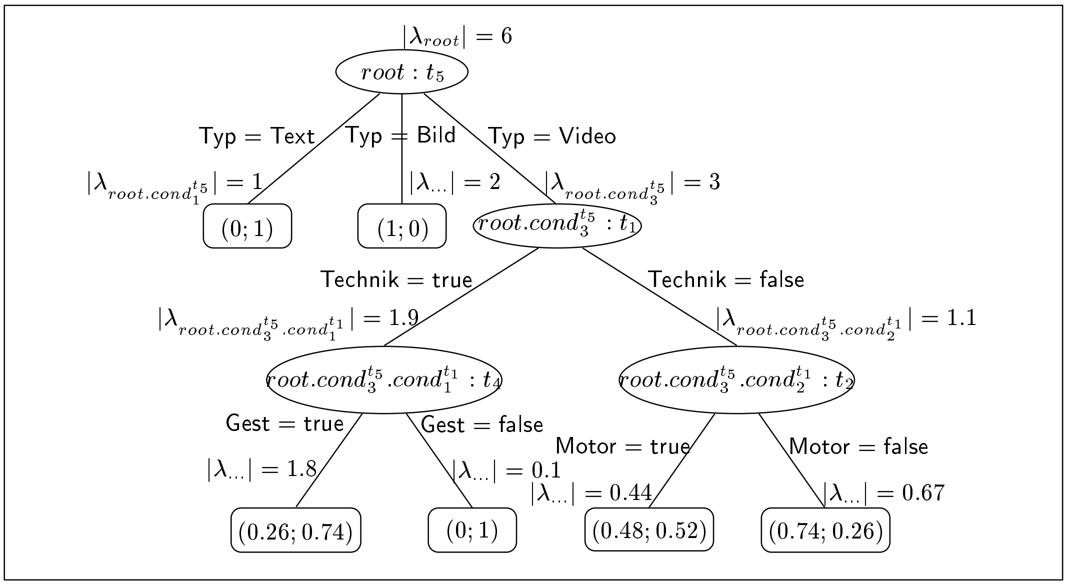

Abbildung 5.4: Optimaler inkrementeller Fuzzy-Entscheidungsbaum

Kapitel 6

Eigenschaften des Algorithmus

Unser Ziel ist es zu zeigen, daß unter gleichen Voraussetzungen ausgehend von einer Beispielmenge ein- und derselbe Fuzzy-Entscheidungsbaum durch die Algorithmen IFDT zum nicht-inkrementellen und IIFDT zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen erzeugt wird. Für diesen Zweck soll der entwickelte Algorithmus IIFDT bestimmte Eigenschaften besitzen: Termination, Korrektheit und Optimalität.

In Unterkapitel 6.1 wird die Eigenschaft Termination definiert und bewiesen. Die Bedingungen, unter denen ein inkrementeller Fuzzy-Entscheidungsbaum korrekt und optimal ist, definieren wir in den Unterkapiteln 6.2 und 6.3. In Unterkapitel 6.4 wird gezeigt, daß ein solcher korrekter und optimaler Baum äquivalent zu dem Entscheidungsbaum ist, der durch den Algorithmus IFDT erzeugt wird. Anschließend wird argumentiert, daß der Algorithmus IIFDT einen korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaum induziert, indem diese Eigenschaften für jede Funktion des Algorithmus einzeln nachgewiesen werden.

Unter Korrektheit verstehen wir, daß die gespeicherten Werte an den Knoten des Baumes den tatsächlichen Werten entsprechen, d.h. der Wert $\langle|\lambda_n|\rangle$ entspricht dem Wert $|\lambda_n|$, mit dem der Algorithmus zum nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen am Knoten n aufgerufen würde, vorausgesetzt die beiden Algorithmen betrachten die gleiche Lernmenge. Korrektheit bedeutet auch, daß an den Blattknoten die Abbruchbedingungen tatsächlich erfüllt sind.

Dagegen ist die Eigenschaft Optimalität mit der Struktur des Fuzzy-Entscheidungsbaumes verbunden, d.h. ein Baum ist optimal bezüglich einer Methode $\mathcal{M}$ zur Auswahl der Tests und einer Abbruchsfunktion $\mathcal{F}$, wenn an jedem inneren Knoten ein optimaler Test gemäß $\mathcal{M}$ steht und die Abbruchbedingungen von $\mathcal{F}$ nicht erfüllt sind.

Unser Ziel ist es zu beweisen, daß ein inkrementeller Fuzzy-Entscheidungsbaum, der die Eigenschaften Korrektheit und Optimalität erfüllt, gleich zu dem Fuzzy-Entscheidungsbaum ist, der durch den Algorithmus zum nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen erzeugt wird.

6.1 Termination

Satz 6.1 *Der Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen IIFDT terminiert.*

Beweis Um zu zeigen, daß der Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen IIFDT terminiert, müssen wir begründen, wieso jede der drei Basis-Funktionen **addExampleToTree, transposition** und **ensureBestTest** terminiert, da der Algorithmus IIFDT aus den rekursiven, gegenseitigen Aufrufen dieser Funktionen besteht.

Die Funktion **addExampleToTree** ist die einzige Funktion, die keine der anderen zwei Funktionen aufruft. Innerhalb der Funktion werden nur rekursive Aufrufe ausgeführt. Bei jedem rekursiven Aufruf wird die Menge der verfügbaren Tests verkleinert. Nach endlich vielen Schritten wird die Menge der verfügbaren Tests leer und die Funktion bricht ab.

Komplizierter wird es bei der Funktion **transposition**, wo zusätzlich zu den rekursiven Aufrufen auch die Funktion **addExampleToTree** ausgeführt wird, die allerdings, wie soeben begründet, terminiert. Außerdem werden die rekursiven Aufrufe von **transposition** weder generell mit einer kleineren Menge von verfügbaren Tests, noch generell mit einem kleineren Entscheidungsbaum ausgeführt. Wir beobachten aber, daß bei jedem rekursiven Aufruf entweder die Testmenge kleiner wird, oder sich die Höhe des Baumes verkleinert und dabei die Testmenge unverändert bleibt. Deswegen können wir durch Induktion über das Paar von Aufrufparametern: Menge der verfügbaren Tests der Größe z und Höhe des Baumes h, beweisen, daß die Funktion **transposition** terminiert. Dabei definieren wir die wohl-fundierte Ordnung:

$$(z', h') < (z, h) \qquad \text{gdw.} \quad z' < z \vee (z' = z \wedge h' < h)$$

und behaupten, daß für jeden rekursiven Aufruf der Funktion **transposition** gilt:

$$(z', h') < (z, h)$$

wobei wir mit (z', h') die Größen für den rekursiven Aufruf bezeichnen. Hier werden wir nur die Struktur der Induktion und nicht den vollständigen Beweis angeben:

Behauptung Aufgerufen mit einem Teilbaum $T(n)$ der Höhe h und mit einer Menge von verfügbaren Tests der Größe z, terminiert die Funktion **transposition**.

Induktionsanfang Die Funktion **transposition** terminiert für das Paar $(z = 1, h = 1)$.

Bei einem Teilbaum $T(n)$ der Höhe 1 handelt sich um einen inneren Knoten n, dessen Nachfolger Blattknoten sind. Aufgerufen mit solch einem Teilbaum führt die Funktion **transposition** den Schritt 2 auf Seite 45 durch, der keinen rekursiven Aufruf beinhaltet, und terminiert (Aufrufe von **addExampleToTree** terminieren, wie oben begründet).

Induktionsschritt

Induktionsvoraussetzung Die Funktion **transposition** terminiert für alle Paare (z', h'), für die $(z', h') < (z, h)$ gilt.

Induktionsbehauptung Die Funktion **transposition** terminiert für das Paar (z, h).

Wir unterscheiden folgende Fälle:

- Aufruf des Schritts 2 auf Seite 45:
  Es gibt keine rekursiven Aufrufe von **transposition**, Aufrufe von **addExampleToTree** terminieren (siehe Induktionsanfang).
- Aufruf des Schritts 3 auf Seite 46:
  Falls ein rekursiver Aufruf im Schritt 3a erfolgt, gilt: $z' = z \wedge h' < h$, also $(z', h') < (z, h)$. Daraus folgt wegen Induktionsvoraussetzung, daß der rekursive Aufruf von **transposition** terminiert.
  Die Schritte 3b bis 3c werden auf jeden Fall ausgeführt und sie terminieren, da die Funktion **addExampleToTree** terminiert.
- Aufruf des Schritts 4 auf Seite 47:
  Die Ausführung des Schritts 4a terminiert, da **addExampleToTree** terminiert. Weiter werden die Schritte aus 5 bearbeitet (siehe den nächsten Punkt).
- Aufruf des Schritts 5 auf Seite 47:
  Für jeden rekursiven Aufruf in 5a gilt: $z' < z$, also $(z', h') < (z, h)$ und wegen Induktionsvoraussetzung terminiert dieser Aufruf.
  Die weitere Ausführung der Schritte 5b bis 5f terminiert.

So wurde durch Induktion bewiesen, daß **transposition** terminiert.

Für die Funktion **ensureBestTest** stellen wir ähnliche Überlegungen wie für die Funktion **addExampleToTree** an, d.h. **ensureBestTest** wird rekursiv immer mit einer kleineren Menge von Tests aufgerufen. Es ist auch möglich, daß innerhalb **ensureBestTest** die Funktion **transposition** aufgerufen wird. Wie gerade gezeigt, terminiert jeder solcher Aufruf, also terminiert auch **ensureBestTest**. □

6.2 Korrektheit

In diesem Unterkapitel wollen wir eine formale Definition des Begriffs Korrektheit eines inkrementellen Fuzzy-Entscheidungsbaumes geben. Wir führen verallgemeinerte Definitionen der Korrektheit eines inneren Knotens und eines Blattknotens bezüglich eines beliebigen Knotens n' und einer Fuzzy-Menge κ ein. Dabei liegt der Knoten n' auf dem Weg vom Wurzelknoten eines inkrementellen Fuzzy-Entscheidungsbaumes bis zum aktuellen Knoten. Die Fuzzy-Menge κ ist über die Beispiele aus $\mathcal{L}$ definiert und gibt eine Zuordnung jedes Beispiels $f \in \mathcal{L}$ zum Knoten n' an (die Beispiele aus der Menge $\mathcal{L}$ sind zur Induktion des Entscheidungsbaumes verwendet).

Wir erweitern die bisherige Notation, um zu zeigen, daß ein bestimmter Zugehörigkeitsgrad nicht vom Wurzelknoten *root* mit λ_{root} aus berechnet ist (siehe Definition 3.8 auf Seite 20), sondern vom Knoten n' mit der Fuzzy-Menge κ. Dies bezeichnen wir durch den Index $n' \to n$ und den oberen Index κ. Somit läßt sich die Definition 3.8 wie folgt verallgemeinern:

Definition 6.2 *Seien ein Knoten n' und eine Fuzzy-Menge κ, die jedem Beispiel einen Zugehörigkeitsgrad zuordnet, gegeben. Für den Knoten n' gilt:*

$$\lambda^{\kappa}_{n' \to n'}(f) = \kappa(f)$$

und für jeden Nachfolger $n.cond^{t_n}_g$ eines Knotens n, der nicht oberhalb von n' liegt:

$$\lambda^{\kappa}_{n' \to n.cond^{t_n}_g}(f) = \top(\lambda^{\kappa}_{n' \to n}(f), \mu_{cond^{t_n}_g}(f))$$

Wenn n' und κ identisch zum Wurzelknoten *root* bzw. λ_{root} sind, dann sind die Definitionen 3.8 und 6.2 äquivalent. In diesem Fall werden die Bezeichnung $n' \rightarrow$ und κ weggelassen.

Definition 6.3 *Ein Blattknoten n eines inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$, der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ ist* **korrekt bezüglich eines Knotens** *n' des Baumes, der auf dem Weg von der Wurzel root des Baumes bis zum Knoten n liegt, und* **bezüglich der Fuzzy-Menge** *κ über die Beispiele aus $\mathcal{L}$ wenn:*

- *für jedes Beispiel $f \in \mathcal{L}$ gilt:*
  - *wenn das Beispiel am Blattknoten gespeichert ist, dann:*
  $$\langle \lambda_n(f) \rangle \quad = \quad \lambda^{\kappa}_{n' \rightarrow n}(f) \tag{6.1}$$
  - *wenn für das Beispiel f kein Wert gespeichert ist, dann gilt:*
  $$\lambda^{\kappa}_{n' \rightarrow n}(f) = 0$$

  wobei $\lambda^{\kappa}_{n' \rightarrow n}(f)$ nach Definition 6.2 beginnend am Knoten n' mit dem Wert $\kappa(f)$ ausgerechnet wird.
- *die folgenden Gleichungen erfüllt sind:*
  $$\langle |\lambda_n| \rangle \quad = \quad |\lambda^{\kappa}_{n' \rightarrow n}| \tag{6.2}$$
  $$\langle |\lambda_n | c_k| \rangle \quad = \quad |\lambda^{\kappa}_{n' \rightarrow n} | c_k| \tag{6.3}$$

  für jede Klasse $c_k \in \mathcal{C}$.

  Dabei wird die Fuzzy-Menge $\lambda^{\kappa}_{n' \rightarrow n}|c_k$ nach Gleichung (3.3) auf Seite 22 bestimmt, die Mächtigkeit der beiden Fuzzy-Mengen nach Definition 2.2 auf Seite 8 unter Berücksichtigung der Beispiele aus $\mathcal{L}$ ausgerechnet. Die Zugehörigkeit jedes Beispiels wird nach Definition 6.2, beginnend am Knoten n' mit der Fuzzy-Menge κ, ausgerechnet.
- *die Abbruchskriterien gemäß $\mathcal{F}(\langle |\lambda_n| \rangle, \langle |\lambda_n|c_k| \rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ erfüllt sind und der Knoten gemäß $\mathcal{F}(\langle |\lambda_n| \rangle, \langle |\lambda_n|c_k| \rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ beschriftet ist, wobei $\mathcal{T}' \subseteq \mathcal{T}$ die Menge der verfügbaren Tests ist, d.h. die Tests, die nicht auf dem Weg von root zu n liegen.*

Wir bemerken, daß ein leerer Blattknoten immer bezüglich sich selbst und einer Fuzzy-Menge κ^0 korrekt ist, zu der jedes Beispiel mit dem Zugehörigkeitsgrad Null gehört.

Definition 6.4 *Ein innerer Knoten n eines inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$, der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ ist* **korrekt bezüglich eines Knotens** *n' des Baumes, der auf dem Weg vom Wurzelknoten root des Baumes bis zum Knoten n liegt, und* **bezüglich der Fuzzy-Menge** *κ über die Beispiele aus $\mathcal{L}$ wenn:*

- *die folgenden Gleichungen erfüllt sind:*
  $$\langle |\lambda_n| \rangle \quad = \quad |\lambda^{\kappa}_{n' \rightarrow n}| \tag{6.4}$$
  $$\langle |\lambda_n | c_k| \rangle \quad = \quad |\lambda^{\kappa}_{n' \rightarrow n} | c_k| \tag{6.5}$$
  $$\langle |\lambda_n | cond^{t^\star}_d| \rangle \quad = \quad |\lambda^{\kappa}_{n' \rightarrow n} | cond^{t^\star}_d| \tag{6.6}$$
  $$\langle |\lambda_n | cond^{t^\star}_d | c_k| \rangle \quad = \quad |\lambda^{\kappa}_{n' \rightarrow n} | cond^{t^\star}_d | c_k| \tag{6.7}$$
  $$\langle |\lambda^{missing}_{n,t^\star}| \rangle \quad = \quad |\lambda^{\kappa,missing}_{n' \rightarrow n,t^\star}| \tag{6.8}$$

für jede Klasse $c_k \in \mathcal{C}$ *und jede Bedingung* $cond_d^{t^\star}, 1 \leq d \leq u_{t^\star}$ *jedes Tests* $t^\star \in \mathcal{T}$.

Die Fuzzy-Mengen sind nach Gleichungen (3.3), (3.4), (3.5) *auf Seite 22 und* (3.19) *auf Seite 26 bestimmt, deren Größe nach Definition 2.2 auf Seite 8 unter Berücksichtigung der Beispiele aus* $\mathcal{L}$ *ausgerechnet. Die Zugehörigkeit jedes Beispiels wird nach Definition 6.2, beginnend am Knoten* n' *mit der Fuzzy-Menge* κ, *ausgerechnet.*

- *die Markierung des Knotens korrekt ist, d.h. wenn der Knoten nicht markiert ist, dann sind die Abbruchbedingungen* $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ *nicht erfüllt und der Test am Knoten* n *ist der beste Test aus* $\mathcal{T}'$ *bezüglich der Methode* $\mathcal{M}$ *und der dort gespeicherten Daten, wobei* $\mathcal{T}' \subseteq \mathcal{T}$ *die Menge der verfügbaren Tests ist, d.h. die Tests, die nicht auf dem Weg von* $root$ *zu* n *liegen.*

Ausgehend von den Definitionen 6.3 und 6.4 der relativen Korrektheit eines Knotens, können wir die Korrektheit eines Teilbaumes $T(n)$ eines inkrementellen Fuzzy-Entscheidungsbaumes definieren.

Definition 6.5 *Ein Teilbaum* $T(n)$ *eines inkrementellen Fuzzy-Entscheidungsbaumes ist* **korrekt bezüglich der Fuzzy-Menge** κ, *wenn:*

- *der Wurzelknoten* n *des Teilbaumes bezüglich sich selbst und der Fuzzy-Menge* κ *korrekt ist*
- *die Nachfolgerknoten von* n *korrekt bezüglich* n *und* κ *sind*
- *die Teilbäume* $T(n.cond_g^{t_n})$, *beginnend an den Nachfolgerknoten von* n, *korrekt bezüglich der Fuzzy-Menge* κ' *sind, wobei* κ' *nach Definition 6.2, beginnend an* n *mit* κ, *erhalten wird:* $\forall f \in \mathcal{L} : \kappa'(f) = \top(\kappa(f), \mu_{cond_g^{t_n}}(f))$

d.h. jeder Knoten des Teilbaumes korrekt bezüglich n *und* κ *ist.*

Für den gesamten inkrementellen Fuzzy-Entscheidungsbaum wissen wir, daß jedes Beispiel exakt mit dem Zugehörigkeitsgrad Eins zum Wurzelknoten des Baumes gehört. Die Korrektheit des inkrementellen Fuzzy-Entscheidungsbaumes können wir von der Korrektheit eines Teilbaumes ableiten.

Definition 6.6 *Ein inkrementeller Fuzzy-Entscheidungsbaum ist* **korrekt**, *wenn jeder Knoten* n *des Fuzzy-Entscheidungsbaumes korrekt bezüglich des Wurzelknotens* $root$ *und der Fuzzy-Menge* λ_{root} *ist.*

Durch eine einfache Induktion läßt sich folgendes Korollar beweisen:

Korollar 6.7 *Jeder Teilbaum* $T(n)$ *eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes ist korrekt bezüglich der Fuzzy-Menge* λ_n $(\lambda_{root \to n}^{\lambda_{root}})$.

In Unterkapitel 6.4 werden wir argumentieren, daß der Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen IIFDT immer einen korrekten Baum erzeugt. Dafür werden wir den Satz 6.8 und daraus folgenden Hilfssatz 6.10 benötigen.

Satz 6.8 *Sei ein korrekter Teilbaum* $T(n')$ *bezüglich der Fuzzy-Menge* κ *gegeben. Weiterhin sei* $T(n')$ *ein Teilbaum eines inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge* $\mathcal{L}$.

Für jeden Knoten n *des Teilbaumes und jedes Beispiel* $f \in \mathcal{L}$ *ist der nach Definition 5.3 auf Seite 42 ausgerechnete Repräsentationsgrad des Beispiels* f *zum Knoten* n *korrekt* *bezüglich Knoten n' und der Fuzzy-Menge κ:*

$$\langle\langle \lambda_n(f) \rangle\rangle = \lambda^{\kappa}_{n' \to n}(f)$$

Beweis Beweis durch Induktion

Behauptung Für jeden Knoten n eines korrekten Teilbaumes $T(n')$ bezüglich einer Fuzzy-Menge κ ist der Repräsentationsgrad $\langle\langle \lambda_n(f) \rangle\rangle$ jedes Beispiels f zum Knoten n korrekt bezüglich n' und der Fuzzy-Menge κ.

Induktionsanfang Die Behauptung gilt für jeden Blattknoten n.

Für jeden Blattknoten n gilt:

$$\begin{aligned} \langle\langle \lambda_n(f) \rangle\rangle & \overset{\text{Def. 5.3 auf S. 42}}{=} \langle \lambda_n(f) \rangle \\ & \underset{\text{Def. 6.3 und Def. 6.5}}{\overset{T(n') \text{ korrekt bez. } \kappa}{=}} \lambda^{\kappa}_{n' \to n}(f) \end{aligned}$$

Induktionsschritt

Induktionsvoraussetzung Sei n ein innerer Knoten mit dem Test t_n und gelte die Behauptung für alle seine Nachfolgerknoten.

Induktionsbehauptung Die Behauptung ist auch für den Knoten n erfüllt.

Für den inneren Knoten n erhalten wir:

$$\begin{aligned} \langle\langle \lambda_n(f) \rangle\rangle & \overset{\text{Def. 5.3 auf S. 42}}{=} \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \langle\langle \lambda_{n.cond_g^{t_n}}(f) \rangle\rangle \\ & \overset{\text{Induktionsvorausssetung}}{=} \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \lambda^{\kappa}_{n' \to n.cond_g^{t_n}}(f) \\ & \overset{\text{Def. 6.2 auf S. 56}}{=} \sum_{n.cond_g^{t_n} \in \mathcal{N}(n)} \top(\lambda^{\kappa}_{n' \to n}(f), \mu_{cond_g^{t_n}}(f)) \\ & = \lambda^{\kappa}_{n' \to n}(f) \end{aligned}$$

□

Korollar 6.9 *Insbesondere für den Wurzelknoten des korrekten Teilbaumes $T(n')$ bezüglich der Fuzzy-Menge κ ist nach Satz 6.8 der Repräsentationsgrad jedes Beispiel f identisch zu dem Zugehörigkeitsgrad von f zu κ:*

$$\langle\langle \lambda_{n'}(f) \rangle\rangle = \lambda^{\kappa}_{n' \to n'}(f) = \kappa(f)$$

Hilfssatz 6.10 *Sei ein korrekter Teilbaum $T(n')$ bezüglich der Fuzzy-Menge κ gegeben. $T(n')$ sei ein Teilbaum eines inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$. Für die gespeicherte Information an jedem korrekten Knoten n des Teilbaumes gilt:*

$$\begin{aligned} \langle | \lambda_n | \rangle &= \sum_{f=1}^{m} \langle\langle \lambda_n(f) \rangle\rangle \\ \langle | \lambda_n | c_k | \rangle &= \sum_{f=1}^{m} \top(\langle\langle \lambda_n(f) \rangle\rangle, \mu_{c_k}(f)) \end{aligned}$$

Ist n zusätzlich ein innerer Knoten, dann gelten auch die Gleichungen:

$$\begin{aligned}
\langle|\lambda_n|cond_d^{t^\star}|\rangle &= \sum_{f=1}^{m} \top(\langle\langle\lambda_n(f)\rangle\rangle, \mu_{cond_d^{t^\star}}(f)) \\
\langle|\lambda_n|cond_d^{t^\star}|c_k|\rangle &= \sum_{f=1}^{m} \top(\langle\langle\lambda_n(f)\rangle\rangle, \mu_{cond_d^{t^\star}}(f), \mu_{c_k}(f)) \\
\langle|\lambda_{n,t^\star}^{missing}|\rangle &= \sum_{f=1}^{m} \begin{cases} \langle\langle\lambda_n(f)\rangle\rangle & \text{wenn } f \text{ unvollständiges Beispiel bezüglich } t^\star \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Beweis

$$\begin{aligned}
\langle|\lambda_n|\rangle &\overset{\text{Korrektheit}}{=} |\lambda_{n'\to n}^{\kappa}| \\
&= \sum_{f=1}^{m} \lambda_{n'\to n}^{\kappa}(f) \\
&\overset{\text{Satz 6.8 auf S. 58}}{=} \sum_{f=1}^{m} \langle\langle\lambda_n(f)\rangle\rangle
\end{aligned}$$

Die restlichen Gleichungen werden analog bewiesen. □

Korollar 6.11 *Der Satz 6.8 und Hilfssatz 6.10 sind natürlich auf einen korrekten inkrementellen Fuzzy-Entscheidungsbaumes anwendbar. Der Gesamtbaum wird dann als der korrekte Teilbaum $T(root)$ bezüglich der Fuzzy-Menge λ_{root} angesehen.*

Insbesondere für den Knoten root und jedes Beispiels f gilt:

$$\langle\langle\lambda_{root}(f)\rangle\rangle = 1$$

6.3 Optimalität

In diesem Unterkapitel definieren wir die Eigenschaft Optimalität eines inkrementellen Fuzzy-Entscheidungsbaumes.

Definition 6.12 *Ein innerer Knoten n eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes IFT bezüglich der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ ist* **optimal bezüglich eines Verfahrens** $\mathcal{M}$ *zur Auswahl des besten Tests und* **einer Funktion** $\mathcal{F}$ *zum Testen der Abbruchskriterien, wenn*

- *der Knoten n durch den besten Test aus $\mathcal{T}'$ bezüglich $\mathcal{M}$ und der gespeicherten Information beschriftet ist*
- *die Funktion $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ nicht erfüllt ist*

wobei $\mathcal{T}' \subseteq \mathcal{T}$ die Menge der verfügbaren Tests ist, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten root des Baumes zu n liegen.

Ein Blattknoten n eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes ist immer **optimal**.

Definition 6.13 *Ein korrekter inkrementeller Fuzzy-Entscheidungsbaum ist* **optimal bezüglich eines Verfahrens** $\mathcal{M}$ *zur Auswahl des besten Tests und* **einer Funktion** $\mathcal{F}$ *zum*

Testen der Abbruchskriterien, wenn jeder Knoten des Baumes optimal bezüglich $\mathcal{M}$ und $\mathcal{F}$ ist.

Korollar 6.14 *Jeder Teilbaum $T(n)$ eines korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaumes ist optimal und nach Korollar 6.7 korrekt bezüglich der Fuzzy-Menge λ_n ($\lambda_{root \to n}^{\lambda_{root}}$).*

In Unterkapitel 6.4 werden wir zeigen, daß der Algorithmus IIFDT, abgeschlossen mit **ensureBestTest**, immer einen optimalen Entscheidungsbaum erzeugt.

6.4 Äquivalenz des inkrementellen und nicht-inkrementellen Aufbaus von Fuzzy-Entscheidungsbäumen

Hauptsatz 1 *Seien eine Klassenmenge $\mathcal{C}$, eine Menge von zulässigen Tests $\mathcal{T}$, eine Lernmenge $\mathcal{L}$, eine Methode $\mathcal{M}$ und Funktion $\mathcal{F}$ gegeben. Weiterhin sei ein* **korrekter** *und* **optimaler** *inkrementeller Fuzzy-Entscheidungsbaum $IFT = (N_{IFT}, E_{IFT}, L_{IFT}, S_{IFT})$ bezüglich $\mathcal{C}$ und $\mathcal{T}$ gegeben.*

Dann ist IFT äquivalent zu dem Baum, der durch den Algorithmus $IFDT$ zum nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen induziert wird, d.h.:

$$T(root_{IFT}) = \mathbf{IFDT}(\lambda_{root}, \mathcal{T})$$

Beweis Wir können die Äquivalenz der Entscheidungsbäume mit Hilfe der Induktion über die Struktur des Entscheidungsbaumes IFT zeigen, da er endlich ist.

Behauptung Jeder korrekte (bezüglich der Fuzzy-Menge $\lambda_{root \to n}^{\lambda_{root}} = \lambda_n$) und optimale (bezüglich der Methode $\mathcal{M}$ und der Funktion $\mathcal{F}$) Teilbaum $T(n)$ eines korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaumes IFT ist äquivalent zu dem Fuzzy-Entscheidungsbaum, der durch die Ausführung der Funktion $\mathbf{IFDT}(\lambda_n, \mathcal{T}')$ erzeugt wird:

$$T(n) = \mathbf{IFDT}(\lambda_n, \mathcal{T}')$$

wobei $\mathcal{T}' \subseteq \mathcal{T}$ die Menge der verfügbaren Tests ist, d.h. die Tests, die nicht auf dem Weg von $root_{IFT}$ nach n liegen.

Induktionsanfang Die Behauptung gilt für jeden Blattknoten.

Ist n ein Blattknoten, dann gilt: $T(n) = n$.

Wegen Korrektheit von n bezüglich des Knotens $root$ wissen wir, daß die Funktion $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ erfüllt ist, der Blattknoten n eine korrekte Beschriftung $L_{IFT}(n)$ nach $\mathcal{F}$ hat und die Gleichungen (6.2) und (6.3) auf Seite 57 erfüllt sind, d.h.

$$\begin{aligned} \langle|\lambda_n|\rangle &= |\lambda_{root \to n}^{\lambda_{root}}| = |\lambda_n| \\ \langle|\lambda_n|c_k|\rangle &= |\lambda_{root \to n}^{\lambda_{root}}|c_k| = |\lambda_n|c_k| \end{aligned}$$

Somit ist für die Funktion $\mathbf{IFDT}(\lambda_n, \mathcal{T}')$ ebenfalls $\mathcal{F}(|\lambda_n|, |\lambda_n|c_k|_{k=1,2,\ldots,q}, \mathcal{T}')$ erfüllt und sie liefert einen Blattknoten zurück, der genauso nach $\mathcal{F}$ beschriftet ist, also die gleiche Beschriftung wie n aufweist. Es folgt:

$$T(n) = n = \mathbf{IFDT}(\lambda_n, \mathcal{T}')$$

Induktionsschritt

Induktionsvoraussetzung Sei n ein innerer Knoten und gelte die Behauptung für alle seine Nachfolgerknoten.

Induktionsbehauptung Die Behauptung ist auch für den inneren Knoten n erfüllt.

Wegen Optimalität des Baumes IFT ist der Knoten n optimal (s. Def. 6.12 und Def. 6.13), d.h. $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ ist nicht erfüllt und der Knoten n ist mit dem besten Test $t = [cond_1^t, cond_2^t, \ldots, cond_{u_t}^t] \in \mathcal{T}'$ bezüglich $\mathcal{M}$ beschriftet.

Wegen der Induktionsvoraussetzung wissen wir, daß für jeden Nachfolgerknoten von n, betrachten wir den h-ten Nachfolger $n.cond_h^t$, gilt:

$$T(n.cond_h^t) = \mathbf{IFDT(}\lambda_{n.cond_h^t}, \mathcal{T}''\mathbf{)}$$

wobei $\mathcal{T}'' \subseteq \mathcal{T}$ die Menge der verfügbaren Tests ist, d.h. alle Tests, die nicht oberhalb von $n.cond_h^t$ liegen. Das ist genau die Menge $\mathcal{T}'' = \mathcal{T}' \setminus \{t\}$.

Für die Funktion $\mathbf{IFDT(}\lambda_n, \mathcal{T}'\mathbf{)}$ ist $\mathcal{F}(|\lambda_n|, |\lambda_n|c_k|_{k=1,2,\ldots,q}, \mathcal{T}')$ ebenfalls nicht erfüllt (da wegen Korrektheit von n bezüglich $root$ die Gleichungen (6.4) und (6.5) erfüllt sind) und sie liefert einen inneren Knoten n' mit demselben besten Test t bezüglich $\mathcal{M}$ zurück. Der Knoten n' ist mit dem Test t beschriftet, d.h. er hat u_t Nachfolgerknoten. Die Knoten n und n' sind gleich beschriftet (mit dem Test t) und haben jeweils genau u_t Nachfolger.

Jeden Nachfolger des Knotens n' erhalten wir durch die Ausführung der Funktion $\mathbf{IFDT(}\lambda_{n.cond_h^t}, \mathcal{T}' \setminus \{t\}\mathbf{)}$. Da $\mathcal{T}'' = \mathcal{T}' \setminus \{t\}$ gilt, ist der h-te Nachfolger genau $T(n.cond_h^t)$ (siehe oben).

Aus der Gleichheit des Knotens n mit dem Knoten n' und der Gleichheit der Bäume, die an den Nachfolgerknoten hängen, können wir schließen, daß

$$T(n) = \mathbf{IFDT(}\lambda_n, \mathcal{T}'\mathbf{)}$$

Den Wurzelknoten $root_{IFT}$ haben wir anhand der Fuzzy-Menge λ_{root} und der Menge der zulässigen Tests erhalten. Es gilt also, nach soeben bewiesenen Behauptung:

$$T(root_{IFT}) = \mathbf{IFDT(}\lambda_{root}, \mathcal{T}\mathbf{)}$$

□

Korollar 6.15 *Wenn der Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen IIFDT einen korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaum erzeugt, dann ist dieser Entscheidungsbaum wegen dem Hauptsatz 1 äquivalent zum Fuzzy-Entscheidungsbaum, der durch den Algorithmus IFDT generiert wird.*

Wir können zeigen, daß der Algorithmus IIFDT einen korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaum erzeugt, indem wir für jede Basis-Funktion beweisen, daß sie die Eigenschaft Korrektheit erhält und zusätzlich beweisen, daß die Funktion **ensureBestTest** einen korrekten Entscheidungsbaum in einen korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaum überführt.

In den nächsten Unterkapiteln argumentieren wir, daß diese Behauptungen zutreffen.

6.4.1 Eigenschaften von addExampleToTree

Die Funktion **addExampleToTree** beschreibt das Hinzufügen eines Beispiels zu einem Fuzzy-Entscheidungsbaum, was ähnlich dem Algorithmus ITI verläuft.

Würde eine Menge von Beispielen gleichzeitig beobachtet, so werden die Beispiele der Reihe nach zu einem Fuzzy-Entscheidungsbaum mittels der Funktion **addExampleToTree** hinzugefügt.

Bei der Beobachtung des ersten Beispiels existiert kein Entscheidungsbaum und die Lernmenge $\mathcal{L}$ ist leer. Es wird ein inkrementeller Fuzzy-Entscheidungsbaum mit nur einem leeren Knoten *root* erzeugt, d.h. die gespeicherte Information ist mit Null initialisiert. Danach wird die Menge der zulässigen Tests $\mathcal{T}$ bestimmt. Die weiteren Schritte werden bei jedem Eintreffen eines neuen Beispiels f und dessen Einfügen zum Wurzelknoten des Baumes ausgeführt, nämlich:

- füge das Beispiel f zu der Lernmenge $\mathcal{L}$ hinzu
- erweitere das Grundintervall der Fuzzy-Mengen $\mu_{\mathrm{v}_{il}}$ und μ_{c_k} mit dem Beispiel f
- erweitere die beiden Fuzzy-Mengen mit den entsprechenden Zugehörigkeitsgraden von f
- führe

  $$\textbf{addExampleToTree(}f, \Delta_{root,f}, root, \mathcal{T}\textbf{)}$$

  durch, mit $\Delta_{root,f} = \lambda_{root}(f) = 1$

Verallgemeinert wird die Funktion **addExampleToTree** jedes Mal auf einem, bezüglich einer Fuzzy-Menge κ, korrekten Teilbaum $T(n)$ eines inkrementellen Fuzzy-Entscheidungsbaumes ausgeführt. Unser Ziel ist es, einen Teilbaum $T_a(n_a)$ zu erzeugen, der aus dem Baum $T(n)$ entstehen soll, indem das Beispiel f um den Wert $\Delta_{n,f}$ mehr repräsentiert wird. Dafür muß der Teilbaum $T_a(n_a)$ korrekt bezüglich der folgenden Fuzzy-Menge κ_a sein:

$$\begin{aligned} \kappa_a(f) &= \kappa(f) + \Delta_{n,f} \quad \text{für das Beispiel } f && (6.9) \\ \kappa_a(f') &= \kappa(f') \quad \text{für } \forall f' \in \mathcal{L} \text{ mit } f' \neq f && (6.10) \end{aligned}$$

wobei nach Korollar 6.9 auf Seite 59 gilt für $\forall f' \in \mathcal{L}$: $\kappa(f') = \langle\langle \lambda_n(f') \rangle\rangle$.

Satz 6.16 *Sei ein korrekter Teilbaum $T(n)$ eines inkrementellen Fuzzy-Entscheidungsbaumes IFT bezüglich der Lernmenge $\mathcal{L}$, der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ gegeben. Weiterhin sei $\mathcal{T}' \subseteq \mathcal{T}$ die Menge der verfügbaren Tests, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten des Baumes bis zum Knoten n liegen.*

Durch die Ausführung der Funktion **addExampleToTree(**$f, \Delta_{n,f}, n, \mathcal{T}'$**)** *auf dem, bezüglich der Fuzzy-Menge κ, korrekten Teilbaum $T(n)$ wird ein korrekter Teilbaum $T_a(n_a)$ bezüglich der Fuzzy-Menge κ_a zurückgeliefert. Die Fuzzy-Menge κ_a ist durch die Gleichungen (6.9) und (6.10) definiert.*

Beweis Induktion über die Größe der Menge $\mathcal{T}' \subseteq \mathcal{T}$

Behauptung Aufgerufen mit einem, bezüglich der Fuzzy-Menge κ, korrekten Teilbaum $T(n)$, einer Menge $\mathcal{T}'$ von z Tests, einem Beispiel $f \in \mathcal{L}$ und einem Wert $\Delta_{n,f}$, liefert die Funktion

addExampleToTree($f, \Delta_{n,f}, n, \mathcal{T}'$**)**

einen korrekten Teilbaum $T_a(n_a)$ bezüglich der Fuzzy-Menge κ_a zurück, wobei κ_a nach Gleichungen (6.9) und (6.10) auf Seite 63 definiert ist.

Induktionsanfang Die Behauptung gilt für $z = 0$, d.h. $\mathcal{T}' = \emptyset$

Wenn $\Delta_{n,f}$ gleich Null ist, bricht die Funktion **addExampleToTree** ab und die Behauptung ist trivial erfüllt.

Im Schritt 2 auf Seite 43 wird unterschieden, ob der Wurzelknoten des Teilbaumes Blattknoten ist oder nicht. Da $\mathcal{T}' = \emptyset$ erfüllt ist, besteht der Teilbaum $T(n)$ genau aus dem Blattknoten n (da alle Tests bereits auf dem Weg von *root* nach n liegen), also wird mit der Bearbeitung des Schritts 3 fortgesetzt.

In den Schritten 3a bis 3c werden die gespeicherten Daten am Knoten n gemäß des Wertes $\Delta_{n,f}$ aktualisiert. Die Gleichungen (6.1), (6.2) und (6.3) auf Seite 57 sind damit für den Knoten n_a bezüglich der Fuzzy-Menge κ_a erfüllt. Wir zeigen dies für die Gleichung (6.2):

$$\begin{aligned}
\langle|\lambda_{n_a}|\rangle &\overset{\text{Schritt 3b}}{=} \langle|\lambda_n|\rangle + \Delta_{n,f} \\
&\overset{\text{Hilfssatz 6.10 auf S. 59}}{=} \sum_{f'=1}^{m} \langle\langle\lambda_n(f')\rangle\rangle + \Delta_{n,f} \\
&= \sum_{f'\in\mathcal{L}, f'\neq f} \langle\langle\lambda_n(f')\rangle\rangle + \langle\langle\lambda_n(f)\rangle\rangle + \Delta_{n,f} \\
&\overset{\text{Gl. (6.9) u. (6.10), Kor. 6.9}}{=} \sum_{f'\in\mathcal{L}, f'\neq f} \kappa_a(f') + \kappa_a(f) \\
&= \sum_{f'\in\mathcal{L}, f'\neq f} \lambda^{\kappa_a}_{n_a\to n_a}(f') + \lambda^{\kappa_a}_{n_a\to n_a}(f) \\
&= \sum_{f'=1}^{m} \lambda^{\kappa_a}_{n_a\to n_a}(f') \\
&= |\lambda^{\kappa_a}_{n_a\to n_a}|
\end{aligned}$$

Die anderen zwei Gleichungen lassen sich analog beweisen.

Im Schritt 3d wird getestet, ob die Abbruchbedingungen erfüllt sind. Da $\mathcal{T}' = \emptyset$ gilt, ist $\mathcal{F}(\langle|\lambda_{n_a}|\rangle, \langle|\lambda_{n_a}|c_k|\rangle_{k=1,2,\dots,q}, \mathcal{T}')$ erfüllt, d.h. der Knoten wird gemäß $\mathcal{F}$ beschriftet. Der Knoten n_a ist somit nach Definition 6.3 korrekt bezüglich sich selbst und der Fuzzy-Menge κ_a. Damit ist der zurückgelieferte Teilbaum $T_a(n_a) = n_a$ ebenfalls korrekt bezüglich κ_a.

Induktionsschritt

Induktionsvoraussetzung Die Behauptung gilt für jede Menge $\mathcal{T}''$ von $z-1$ Tests.

Induktionsbehauptung Die Behauptung gilt auch für jeden Aufruf der Funktion **addExampleToTree** mit einer Menge $\mathcal{T}'$ von z Tests.

Für den Fall $\Delta_{n,f} = 0$ wird nichts verändert und die Behauptung gilt trivial.

Im Schritt 2 auf Seite 43 wird folgende Fallunterscheidung gemacht:

- n ist Blattknoten, d.h. es wird Schritt 3 auf Seite 43 ausgeführt:
  In den Schritten 3a bis 3c werden die gespeicherten Daten am Knoten n gemäß des Wertes $\Delta_{n,f}$ aktualisiert. Die Gleichungen (6.1), (6.2) und (6.3) auf Seite 57 sind für den Knoten n_a bezüglich der Fuzzy-Menge κ_a erfüllt. Die Gleichung (6.2) haben wir im Induktionsanfang bewiesen. Hier

beweisen wir zusätzlich auch die Gleichung (6.1):

$$\begin{array}{lcl}
\langle\lambda_{n_a}(f)\rangle & \stackrel{\text{Schritt 3a}}{=} & \langle\lambda_n(f)\rangle + \Delta_{n,f} \\
& \stackrel{\text{Def. 5.3 auf S. 42}}{=} & \langle\langle\lambda_n(f)\rangle\rangle + \Delta_{n,f} \\
& \stackrel{\text{Gl. (6.9) u. Kor. 6.9}}{=} & \kappa_a(f) \\
& = & \lambda^{\kappa_a}_{n_a\to n_a}(f)
\end{array}$$

und für $\forall f' \in \mathcal{L}, f' \neq f$:

$$\begin{array}{lcl}
\langle\lambda_{n_a}(f')\rangle & = & \langle\lambda_n(f')\rangle \\
& \stackrel{\text{Def. 5.3 auf S. 42}}{=} & \langle\langle\lambda_n(f')\rangle\rangle \\
& \stackrel{\text{Gl. (6.10) u. Kor. 6.9}}{=} & \kappa_a(f') \\
& = & \lambda^{\kappa_a}_{n_a\to n_a}(f')
\end{array}$$

Die Gleichung (6.3) läßt sich analog überprüfen.
Im Schritt 3d wird getestet, ob die Abbruchbedingungen am Knoten n_a erfüllt sind. Wir unterscheiden folgende Fälle:

- $\mathcal{F}(\langle|\lambda_{n_a}|\rangle, \langle|\lambda_{n_a}|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ ist erfüllt, d.h. n_a bleibt Blattknoten und wird gemäß $\mathcal{F}$ neu beschriftet:
  Der Knoten n_a ist nach Definition 6.3 korrekt bezüglich sich selbst und der Fuzzy-Menge κ_a. Damit ist der zurückgelieferte Teilbaum $T_a(n_a) = n_a$ ebenfalls korrekt bezüglich κ_a.
- $\mathcal{F}(\langle|\lambda_{n_a}|\rangle, \langle|\lambda_{n_a}|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ ist nicht erfüllt, d.h. n_a wird in einen inneren Knoten umgewandelt:
  Mit Hilfe des gespeicherten Zugehörigkeitsgrads $\langle\lambda_{n_a}(f')\rangle$ jedes Beispiels f' aus $\mathcal{L}$ werden im Schritt 3(d)i für jeden Test und jede Klasse die Größen der Fuzzy-Mengen $\lambda_{n_a}|cond^{t^\star}_d$, $\lambda_{n_a}|cond^{t^\star}_d|c_k$, $\lambda^{missing}_{n_a,t^\star}$ ausgerechnet und gespeichert. Die so ausgerechneten Werte sind korrekt (im Sinne von Definition 6.4, letzte drei Gleichungen bezüglich n_a und κ), da der gespeicherte Wert $\langle\lambda_{n_a}(f')\rangle$ für jedes Beispiel f' korrekt ist (siehe oben).
  Im Schritt 3(d)ii wird anhand der gespeicherten Information durch die Methode $\mathcal{M}$ der optimale Test $t = [cond^t_1, cond^t_2, \ldots, cond^t_{u_t}] \in \mathcal{T}'$ ausgewählt und der Knoten n_a damit beschriftet. Der Knoten muß nicht markiert werden, da der optimale Test gesetzt ist und wir gerade getestet haben, daß die Funktion $\mathcal{F}$ nicht erfüllt ist. Der Knoten n_a ist somit nach Definition 6.4 auf Seite 57 korrekt bezüglich sich selbst und der Fuzzy-Menge κ_a.
  Im Schritt 3(d)iii werden u_t leere Nachfolgerknoten von n_a erzeugt. Jeder Nachfolger $n_a.cond^t_h, 1 \leq h \leq u_t$ ist ein korrekter Teilbaum $T(n_a.cond^t_h)$ bezüglich einer Fuzzy-Menge κ^0, zu der jedes Beispiel f' mit dem Zugehörigkeitsgrad Null gehört.
  Im Schritt 3(d)ivA wird für jedes Beispiel $f' \in \mathcal{L}$, das an n_a gespeichert ist und für jeden Nachfolgerknoten $n_a.cond^t_h, 1 \leq h \leq u_t$ der Wert $\Delta_{n_a.cond^t_h,f'}$ ausgerechnet, um den der Zugehörigkeitsgrad des Beispiels f' zu $n_a.cond^t_h$ erhöht wird, und
  $$\textbf{addExampleToTree}(f', \Delta_{n_a.cond^t_h,f'}, n_a.cond^t_h, \mathcal{T}' \setminus \{t\})$$
  ausgeführt.
  Wegen Induktionsvoraussetzung wird, nach dem Hinzufügen eines Bei-

spiels f', für jeden Nachfolger ein korrekter Teilbaum $T_a(n_a.cond_h^t)$ bezüglich der Fuzzy-Menge κ^1 zurückgeliefert, mit:

$$\kappa^1(f') = \kappa^0(f') + \Delta_{n_a.cond_h^t, f'}$$

und

$$\kappa^1(f'') = \kappa^0(f'')$$

für $\forall f'' \in \mathcal{L}$ und $f'' \neq f'$.
Nachdem das Beispiel f' zu allen Nachfolgern hinzugefügt wurde, wird im Schritt 3(d)ivB der Wert $\langle\lambda_{n_a}(f')\rangle$ gelöscht, da er an einem inneren Knoten nicht gespeichert sein soll.
Nach der Bearbeitung des letzten, m-ten Beispiels, das am Knoten n_a gespeichert war, ist jeder Teilbaum $T_a(n_a.cond_h^t)$ korrekt bezüglich der Fuzzy-Menge κ^m, wobei für jedes Beispiel f' gilt:

$$\begin{aligned}\kappa^m(f') = \Delta_{n_a.cond_h^t, f'} &\overset{\text{Schritt 3(d)ivA}}{=} \top(\langle\lambda_{n_a}(f')\rangle, \mu_{cond_h^t}(f')) \\ &\overset{\text{siehe oben}}{=} \top(\kappa_a(f'), \mu_{cond_h^t}(f'))\end{aligned}$$

Wie man sieht, kann die Fuzzy-Menge κ^m nach Definition 6.2 aus der Fuzzy-Menge κ_a erhalten werden.
Um zu zeigen, daß $T_a(n_a)$ nach Definition 6.5 auf Seite 58 korrekt bezüglich κ_a ist, müssen wir noch beweisen, daß der Wurzelknoten $n_a.cond_h^t$ jedes korrekten Teilbaumes $T_a(n_a.cond_h^t)$ korrekt bezüglich des Knotens n_a und der Fuzzy-Menge κ_a ist. Wir zeigen dies nur für die Gleichung (6.4):

$$\begin{aligned}\langle|\lambda_{n_a.cond_h^t}|\rangle &\overset{T_a(n_a.cond_h^t)\text{ korrekt bez. }\kappa^m}{=} |\lambda^{\kappa^m}_{n_a.cond_h^t \to n_a.cond_h^t}| \\ &= \sum_{f'=1}^{m} \lambda^{\kappa^m}_{n_a.cond_h^t \to n_a.cond_h^t}(f') \\ &= \sum_{f'=1}^{m} \kappa^m(f') \\ &\overset{\text{siehe oben}}{=} \sum_{f'=1}^{m} \top(\kappa_a(f'), \mu_{cond_h^t}(f')) \\ &\overset{\text{Def. 6.2 auf S. 56}}{=} \sum_{f'=1}^{m} \top(\lambda^{\kappa_a}_{n_a \to n_a}(f'), \mu_{cond_h^t}(f')) \\ &\overset{\text{Def. 6.2 auf S. 56}}{=} \sum_{f'=1}^{m} \lambda^{\kappa_a}_{n_a \to n_a.cond_h^t}(f') \\ &= |\lambda^{\kappa_a}_{n_a \to n_a.cond_h^t}|\end{aligned}$$

Die restlichen Gleichungen der Definition 6.3 bzw. der Definition 6.4 auf Seite 57(wer wissen nicht, ob es sich um inneren oder Blattknoten handelt) lassen sich ebenfalls analog beweisen. Damit ist der Knoten $n_a.cond_h^t$ korrekt bezüglich n_a und κ_a, also der Teilbaum $T_a(n_a)$ ist korrekt bezüglich κ_a.

- n ist innerer Knoten mit dem Test t_n, d.h. es wird Schritt 4 auf Seite 44 ausgeführt:
In den Schritten 4a bis 4e wird die gespeicherte Information gemäß des Wertes $\Delta_{n,f}$ aktualisiert. Für die Gleichung (6.5) der Definition 6.4 auf Seite 57 gilt:

$$
\begin{aligned}
\langle|\lambda_{n_a}|c_k|\rangle &\overset{\text{Schritt 4b}}{=} \langle|\lambda_n|c_k|\rangle + \top(\Delta_{n,f}, \mu_{c_k}(f)) \\
&\overset{T(n)\text{ korrekt bez. }\kappa}{=} |\lambda^{\kappa}_{n\to n}|c_k| + \top(\Delta_{n,f}, \mu_{c_k}(f)) \\
&= \sum_{f'=1}^{m} \lambda^{\kappa}_{n\to n}|c_k(f') + \top(\Delta_{n,f}, \mu_{c_k}(f)) \\
&= \sum_{f'=1}^{m} \top(\lambda^{\kappa}_{n\to n}(f'), \mu_{c_k}(f')) + \\
&\qquad \top(\Delta_{n,f}, \mu_{c_k}(f)) \\
&= \sum_{f'=1}^{m} \top(\kappa(f'), \mu_{c_k}(f')) + \top(\Delta_{n,f}, \mu_{c_k}(f)) \\
&= \sum_{f'\in\mathcal{L}, f'\neq f} \top(\kappa(f'), \mu_{c_k}(f')) + \\
&\qquad \top(\kappa(f), \mu_{c_k}(f)) + \top(\Delta_{n,f}, \mu_{c_k}(f)) \\
&\overset{\text{Ann. 4.3 auf S. 36}}{=} \sum_{f'\in\mathcal{L}, f'\neq f} \top(\kappa(f'), \mu_{c_k}(f')) + \\
&\qquad \top(\kappa(f) + \Delta_{n,f}, \mu_{c_k}(f)) \\
&\overset{\text{Gl. (6.9) u. Gl. (6.10)}}{=} \sum_{f'\in\mathcal{L}, f'\neq f} \top(\kappa_a(f'), \mu_{c_k}(f')) + \\
&\qquad \top(\kappa_a(f), \mu_{c_k}(f)) \\
&= \sum_{f'=1}^{m} \top(\kappa_a(f'), \mu_{c_k}(f')) \\
&= \sum_{f'=1}^{m} \lambda^{\kappa_a}_{n_a\to n_a}|c_k(f') \\
&= |\lambda^{\kappa_a}_{n_a\to n_a}|c_k|
\end{aligned}
$$

Analog zu dem Induktionsanfang (Beweis von Gleichung (6.2)) können wir zeigen, daß die restlichen Gleichungen (6.4) und (6.6) bis (6.8) korrekt bezüglich n_a und κ_a sind.
Im Schritt 4f wird der Knoten n_a markiert, da nach dem Hinzufügen des Beispiels nicht garantiert werden kann, daß der Test t_n immer noch der optimale Test für den Knoten ist und wir nicht mehr sicher sind, daß $\mathcal{F}(\langle|\lambda_{n_a}|\rangle, \langle|\lambda_{n_a}|c_k|\rangle_{k=1,2,\dots,q}, \mathcal{T}')$ nicht erfüllt ist. Damit ist der Knoten n_a nach Definition 6.4 korrekt bezüglich sich selbst und der Fuzzy-Menge κ_a.
Im Schritt 4g wird für jeden Nachfolgerknoten $n.cond_g^{t_n}, 1 \leq g \leq u_{t_n}$ von n_a der Wert $\Delta_{n.cond_g^{t_n},f}$ ausgerechnet, um den der Zugehörigkeitsgrad des Beispiels f zum Knoten $n.cond_g^{t_n}$ erhöht werden muß, und die Funktion

addExampleToTree($f, \Delta_{n.cond_g^{t_n},f}, n.cond_g^{t_n}, \mathcal{T}' \setminus \{t_n\}$**)**

ausgeführt.
Wegen Korrektheit des ursprünglichen Teilbaumes $T(n)$ und Definition 6.5 auf Seite 58 war jeder Teilbaum $T(n.cond_g^{t_n}), 1 \leq g \leq u_{t_n}$, vor der Ausführung des Schritts 4g, korrekt bezüglich der folgenden Fuzzy-Menge κ':

$$\kappa'(f') = \top(\kappa(f'), \mu_{cond_g^{t_n}}(f'))$$

für $\forall f' \in \mathcal{L}$.
Nach der Ausführung von **addExampleToTree** erhalten wir die Teilbäume $T_a(n_a.cond_g^{t_n}), 1 \leq g \leq u_{t_n}$, die wegen Induktionsvoraussetzung korrekt bezüglich der Fuzzy-Menge κ'_a sind mit:

$$
\begin{aligned}
\kappa'_a(f) = \kappa'(f) + \Delta_{n.cond_g^{t_n},f} &\overset{\text{Schritt 4g}}{=} \kappa'(f) + \top(\Delta_{n,f}, \mu_{cond_g^{t_n}}(f)) \\
&\overset{\text{siehe oben}}{=} \top(\kappa(f), \mu_{cond_g^{t_n}}(f)) +
\end{aligned}
$$

$$
\begin{aligned}
& \top(\Delta_{n,f}, \mu_{cond_g^{tn}}(f)) \\
\stackrel{\text{Ann. 4.3 auf S. 36}}{=} \; & \top(\kappa(f) + \Delta_{n,f}, \mu_{cond_g^{tn}}(f)) \\
\stackrel{\text{Gl. (6.9) auf S. 63}}{=} \; & \top(\kappa_a(f), \mu_{cond_g^{tn}}(f))
\end{aligned}
$$

und

$$
\begin{aligned}
\kappa'_a(f') = \kappa'(f') \; & \stackrel{\text{siehe oben}}{=} \; \top(\kappa(f'), \mu_{cond_g^{tn}}(f')) \\
& \stackrel{\text{Gl. (6.10) auf S. 63}}{=} \; \top(\kappa_a(f'), \mu_{cond_g^{tn}}(f'))
\end{aligned}
$$

für $\forall f' \in \mathcal{L}$ mit $f' \neq f$.
Wie man sieht, gehört jedes Beispiel $f' \in \mathcal{L}$ zu κ'_a mit :

$$\kappa'_a(f') = \top(\kappa_a(f'), \mu_{cond_g^{tn}}(f'))$$

Wir behaupten, daß der Wurzelknoten $n_a.cond_g^{tn}$ jedes korrekten Teilbaumes $T_a(n_a.cond_g^{tn})$ korrekt bezüglich des Knotens n_a und der Fuzzy-Menge κ_a ist (die Gleichungen (6.4) bis (6.8) sind erfüllt). Wir führen den Beweis nur für die Gleichung (6.4) durch:

$$
\begin{aligned}
\langle|\lambda_{n_a.cond_g^{tn}}|\rangle \; & \stackrel{T_a(n_a.cond_g^{tn}) \text{ korrekt bez. } \kappa'_a}{=} \; |\lambda^{\kappa'_a}_{n_a.cond_g^{tn} \to n_a.cond_g^{tn}}| \\
& = \; \sum_{f'=1}^{m} \lambda^{\kappa'_a}_{n_a.cond_g^{tn} \to n_a.cond_g^{tn}}(f') \\
& = \; \sum_{f'=1}^{m} \kappa'_a(f') \\
& \stackrel{\text{siehe oben}}{=} \; \sum_{f'=1}^{m} \top(\kappa_a(f'), \mu_{cond_g^{tn}}(f')) \\
& \stackrel{\text{Def. 6.2 auf S. 56}}{=} \; \sum_{f'=1}^{m} \top(\lambda^{\kappa_a}_{n_a \to n_a}(f'), \mu_{cond_g^{tn}}(f')) \\
& \stackrel{\text{Def. 6.2 auf S. 56}}{=} \; \sum_{f'=1}^{m} \lambda^{\kappa_a}_{n_a \to n_a.cond_g^{tn}}(f') \\
& = \; |\lambda^{\kappa_a}_{n_a \to n_a.cond_g^{tn}}|
\end{aligned}
$$

Die anderen Gleichungen können analog bewiesen werden. Damit ist jeder Nachfolgerknoten $n_a.cond_g^{tn}$ korrekt bezüglich n_a und κ_a und somit ist der zurückgelieferte Teilbaum $T_a(n_a)$ nach Definition 6.5 auf Seite 58 korrekt bezüglich κ_a.

Durch Induktion haben wir bewiesen, daß die Funktion **addExampleToTree**, aufgerufen mit einer Menge $\mathcal{T}'$ von z verfügbaren Tests, mit einem Beispiel f, mit einem Wert $\Delta_{n,f}$ und mit einem korrekten Teilbaum $T(n)$ bezüglich einer Fuzzy-Menge κ, einen korrekten Teilbaum $T_a(n_a)$ bezüglich der Fuzzy-Menge κ_a, die durch die Gleichungen (6.9) und (6.10) auf Seite 63 definiert ist, zurück liefert. □

Korollar 6.17 *Wenn der korrekte Teilbaum $T(n)$ bezüglich κ dem gesamten inkrementellen Fuzzy-Entscheidungsbaum entspricht, dann ist dieser Baum korrekt bezüglich die Fuzzy-Menge λ_{root}. Aufgerufen mit dem Wert $\Delta_{root,f} = 1$ auf dem gesamten korrekten inkrementellen Fuzzy-Entscheidungsbaum liefert die Funktion* **addExampleToTree** *nach Satz 6.16 einen korrekten inkrementellen Fuzzy-Entscheidungsbaum zurück.*

Also erzeugt der Algorithmus IIFDT durch das Hinzufügen von beliebig vielen Beispielen einen korrekten inkrementellen Fuzzy-Entscheidungsbaum.

Die Funktion **addExampleToTree** (ebenso wie **transposition** und **ensureBestTest**) wird rekursiv auf korrekten Teilbäumen eines inkrementellen Fuzzy-Entscheidungsbaumes aufgerufen. Wir haben gezeigt, daß **addExampleToTree** einen korrekten Teilbaum in einen korrekten Teilbaum überführt, indem der Zugehörigkeitsgrad genau eines Beispiels erhöht wird und für alle anderen unverändert bleibt. Im Gegensatz dazu werden bei **transposition** und **ensureBestTest** keine neuen Beispiele hinzugefügt, sondern die bereits betrachteten neu verzweigt. Die beiden Funktionen werden immer auf einem, bezüglich einer Fuzzy-Menge κ, korrekten Teilbaum $T(n)$ eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes aufgerufen. Damit der zurückgelieferte Teilbaum wieder zu dem ursprünglichen inkrementellen Fuzzy-Entscheidungsbaum eingefügt werden kann, ohne die Konsistenz und die Korrektheit dieses Entscheidungsbaumes zu verletzen, soll er ebenfalls korrekt bezüglich der Fuzzy-Menge κ sein.

In den nächsten zwei Unterkapiteln wird für die Funktion **transposition** bzw. **ensureBestTest** argumentiert, daß dies der Fall ist.

6.4.2 Eigenschaften von transposition

Ziel der Funktion **transposition** ist es, aufgerufen auf einem, bezüglich einer Fuzzy-Menge κ, korrekten Teilbaum eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes, den Test am Wurzelknoten des Teilbaumes durch einen ausgewählten Test zu ersetzen und dabei den Teilbaum so umzustrukturieren, daß er korrekt bezüglich κ bleibt. Deswegen kann der neue Teilbaum wieder zu dem inkrementellen Fuzzy-Entscheidungsbaum angehängt werden, ohne „Lücken“ entstehen zu lassen und ohne die Korrektheit des Entscheidungsbaumes zu verletzen.

In Unterkapitel 4.1 haben wir die Eigenschaft kompositorisch für einen klassischen Entscheidungsbaum als Voraussetzung zum erfolgreichen Durchführen des Ersetzens eines Tests an einem Knoten definiert. In Unterkapitel 4.2 haben wir die Eigenschaft kompositorisch für einen Fuzzy-Entscheidungsbaum definiert (siehe Eigenschaft 4.2 auf Seite 36) und gezeigt, daß unter Annahme 4.3 auf Seite 36 jeder Fuzzy-Entscheidungsbaum diese Eigenschaft besitzt (siehe Hilfssatz 4.4 auf Seite 37).

Der Algorithmus IIFDT bzw. die Funktion **transposition** arbeitet mit den gespeicherten Daten an einem Knoten, d.h. damit die Umstrukturierung funktionieren kann, muß die Eigenschaft kompositorisch in Bezug auf die gespeicherten Daten erfüllt sein. Da die Funktion **transposition** auf einen korrekten inkrementellen Entscheidungsbaum angewandt wird, ist dies der Fall.

Um zu zeigen, daß die Funktion **transposition** ausgehend von einem, bezüglich einer Fuzzy-Menge κ, korrekten Teilbaum $T(n)$ eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes, einen korrekten Teilbaum zurück liefert, der wieder zu dem gesamten Entscheidungsbaum angehängt werden kann, werden wir jeden Fall einzeln betrachten und argumentieren, daß der erhaltene Teilbaum $T_t(n_t)$ korrekt bezüglich κ ist.

Alle Nachfolgerknoten eines Knotens sind Blattknoten

Der einfachste Fall ist, wenn alle Nachfolgerknoten des Knotens n Blattknoten sind, siehe Abbildung 6.1.

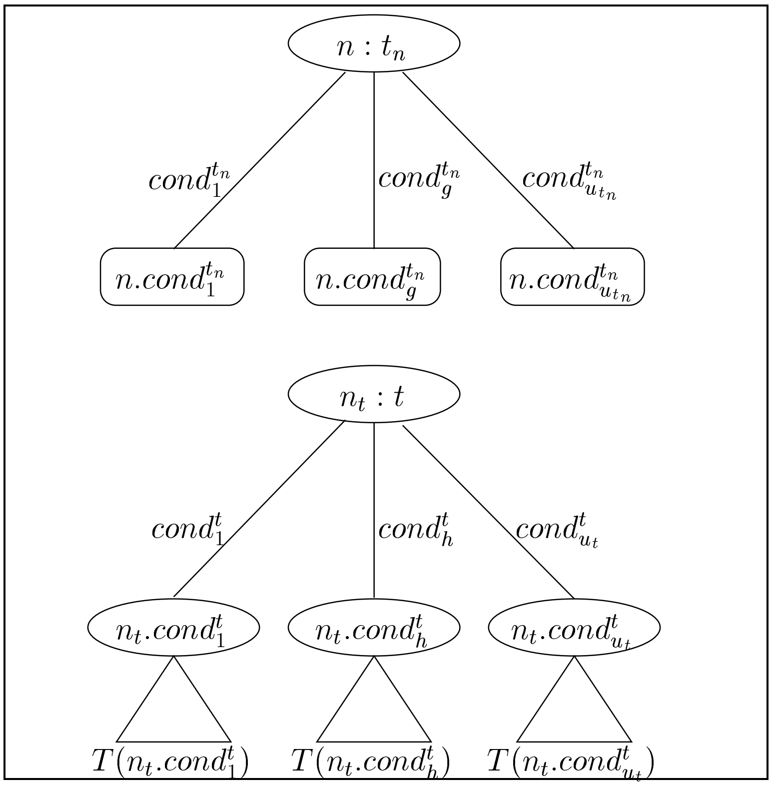

Abbildung 6.1: Bild zum Schritt 2 auf Seite 45

Die Vorgehensweise ist ähnlich des bereits in Unterkapitel 4.1 beschrieben Verfahrens (siehe Seite 32) und wurde im Abschnitt 5.2.2 auf Seite 45, Schritte 2a bis 2f, formal beschrieben.

Hilfssatz 6.18 *Sei ein korrekter Teilbaum $T(n)$ bezüglich der Fuzzy-Menge κ, beginnend am Knoten n mit dem Test t_n, eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$ gegeben, d.h. κ ist identisch zu λ_n bzw. $\lambda_{root\to n}^{\lambda_{root}}$. Weiterhin seien alle Nachfolgerknoten von n Blattknoten.*

Die Ausführung der Schritte 2a bis 2f auf Seite 46 auf dem korrekten Teilbaum $T(n)$ liefert einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ bzw. λ_n zurück, dessen Wurzelknoten n_t mit dem Test t beschriftet ist.

Beweis Im Schritt 2a wird der Repräsentationsgrad jedes Beispiels $f \in \mathcal{L}$ zum Knoten n nach Definition 5.3 auf Seite 42 ausgerechnet. Wegen Korrektheit von $T(n)$ und nach Satz 6.8 auf Seite 58 entspricht der so ausgerechnete Wert $\langle\langle\lambda_n(f)\rangle\rangle$ für jedes Beispiel f genau dem Zugehörigkeitsgrad dieses Beispiels zur Fuzzy-Menge κ:

$$\langle\langle\lambda_n(f)\rangle\rangle = \kappa(f)$$

In den Schritte 2b bis 2d werden die Nachfolger von n entfernt, die Beschriftung des Knotens n geändert (der Knoten wird durch n_t mit dem Test t bezeichnet) und anschließend wird im Schritt 2d der Knoten markiert, da wir nicht garantieren können, daß der gewünschte Test t tatsächlich durch die Methode $\mathcal{M}$ anhand der gespeicherten Daten ausgewählt würde. Der so modifizierte Knoten n_t ist korrekt bezüglich sich selbst und der Fuzzy-Menge κ, da die gespeicherten Größen (gegenüber dem Knoten n) nicht verändert wurden.

Im Schritt 2e wird für jede Bedingung $cond_h^t, 1 \leq h \leq u_t$ des Tests t eine ausgehende Kante mit einem leeren Knoten $n_t.cond_h^t$ erzeugt. Jeder Nachfolgerknoten $n_t.cond_h^t, 1 \leq h \leq u_t$

bzw. jeder Teilbaum $T(n_t.cond_h^t)$ ist korrekt bezüglich der Fuzzy-Menge κ^0, zu der alle Beispiele aus $\mathcal{L}$ mit dem Zugehörigkeitsgrad von Null gehören.

Zu diesem Zeitpunkt besteht der neue Teilbaum $T_t(n_t)$ aus dem Wurzelknoten n_t, der bezüglich der Fuzzy-Menge κ korrekt ist, und den neuen leeren Nachfolgerknoten von n_t, die bezüglich der Fuzzy-Menge κ^0 korrekt sind.

Im letzten Schritt 2f wird für jedes Beispiel f aus $\mathcal{L}$ und für jeden Nachfolgerknoten $n_t.cond_h^t, 1 \leq h \leq u_t$ (anhand dem im Schritt 2a ausgerechneten Zugehörigkeitsgrad $\langle\langle\lambda_n(f)\rangle\rangle$ zum Knoten n_t) der Wert $\Delta_{n_t.cond_h^t,f}$ ausgerechnet, um den der Zugehörigkeitsgrad des Beispiels f zum Knoten $n_t.cond_h^t$ erhöht werden muß, und die Funktion

$$\mathbf{addExampleToTree}(f, \Delta_{n_t.cond_h^t,f}, n_t.cond_h^t, \mathcal{T}')$$

ausgeführt.

Nach dem Hinzufügen eines Beispiels f wird nach Satz 6.16 auf Seite 63 für jeden Nachfolgerknoten ein korrekter Teilbaum $T_t(n_t.cond_h^t)$ bezüglich der Fuzzy-Menge κ^1 zurückgeliefert mit:

$$\kappa^1(f) = \kappa^0(f) + \Delta_{n_t.cond_h^t,f}$$

und

$$\kappa^1(f') = \kappa^0(f')$$

für $\forall f' \in \mathcal{L}$ mit $f' \neq f$.

Nach der Bearbeitung des letzten, m-ten Beispiels aus $\mathcal{L}$ ist jeder Teilbaum $T_t(n_t.cond_h^t)$ korrekt bezüglich der Fuzzy-Menge κ^m, wobei für jedes Beispiel f gilt:

$$\kappa^m(f) = \Delta_{n_t.cond_h^t,f} \overset{\text{Schritt 2f}}{=} \top(\langle\langle\lambda_n(f)\rangle\rangle, \mu_{cond_h^t}(f)) \overset{\text{Gl. auf S. 70}}{=} \top(\kappa(f), \mu_{cond_h^t}(f))$$

Wie man sieht, ergibt sich die Fuzzy-Menge κ^m nach Definition 6.2 auf Seite 56 aus der Fuzzy-Menge κ. Um zu zeigen, daß der Teilbaum $T_t(n_t)$ nach Definition 6.5 auf Seite 58 korrekt bezüglich κ ist, müssen wir noch beweisen, daß der Wurzelknoten $n_t.cond_h^t$ jedes korrekten Teilbaumes $T_t(n_t.cond_h^t)$ korrekt bezüglich des Knotens n_t und der Fuzzy-Menge κ ist. Wir zeigen dies nur für die Gleichung (6.4):

$$\begin{aligned}
\langle|\lambda_{n_t.cond_h^t}|\rangle &\overset{T_t(n_t.cond_h^t)\text{ korrekt bez. }\kappa^m}{=} |\lambda^{\kappa^m}_{n_t.cond_h^t \to n_t.cond_h^t}| \\
&= \sum_{f=1}^{m} \lambda^{\kappa^m}_{n_t.cond_h^t \to n_t.cond_h^t}(f) \\
&= \sum_{f=1}^{m} \kappa^m(f) \\
&\overset{\text{siehe oben}}{=} \sum_{f=1}^{m} \top(\kappa(f), \mu_{cond_h^t}(f)) \\
&\overset{\text{Def. 6.2 auf S. 56}}{=} \sum_{f=1}^{m} \lambda^{\kappa}_{n_t \to n_t.cond_h^t}(f) \\
&= |\lambda^{\kappa}_{n_t \to n_t.cond_h^t}|
\end{aligned}$$

Die restlichen Gleichungen der Definition 6.3 bzw. der Definition 6.4 auf Seite 57 lassen sich ebenfalls analog beweisen. Damit ist der Knoten $n_t.cond_h^t$ korrekt bezüglich des Knotens n_t und der Fuzzy-Menge κ, also der Teilbaum $T_t(n_t)$ ist korrekt bezüglich κ. □

Damit haben wir gezeigt, daß wir jeden korrekten Teilbaum $T(n)$ eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes durch den Teilbaum $T_t(n_t)$ ersetzen können, der durch die Ausführung der Schritte 2a bis 2f auf Seite 46 entstanden ist, ohne die Korrektheit des Gesamtbaumes zu verletzen. Das ist möglich, weil der Teilbaum $T_t(n_t)$ korrekt bezüglich der gleichen Fuzzy-Menge κ ($\kappa = \lambda_n$ bzw. $\lambda^{\lambda_{root}}_{root \to n}$) ist, wie der Teilbaum $T(n)$.

Genau ein Nachfolgerknoten eines Knotens ist ein innerer Knoten

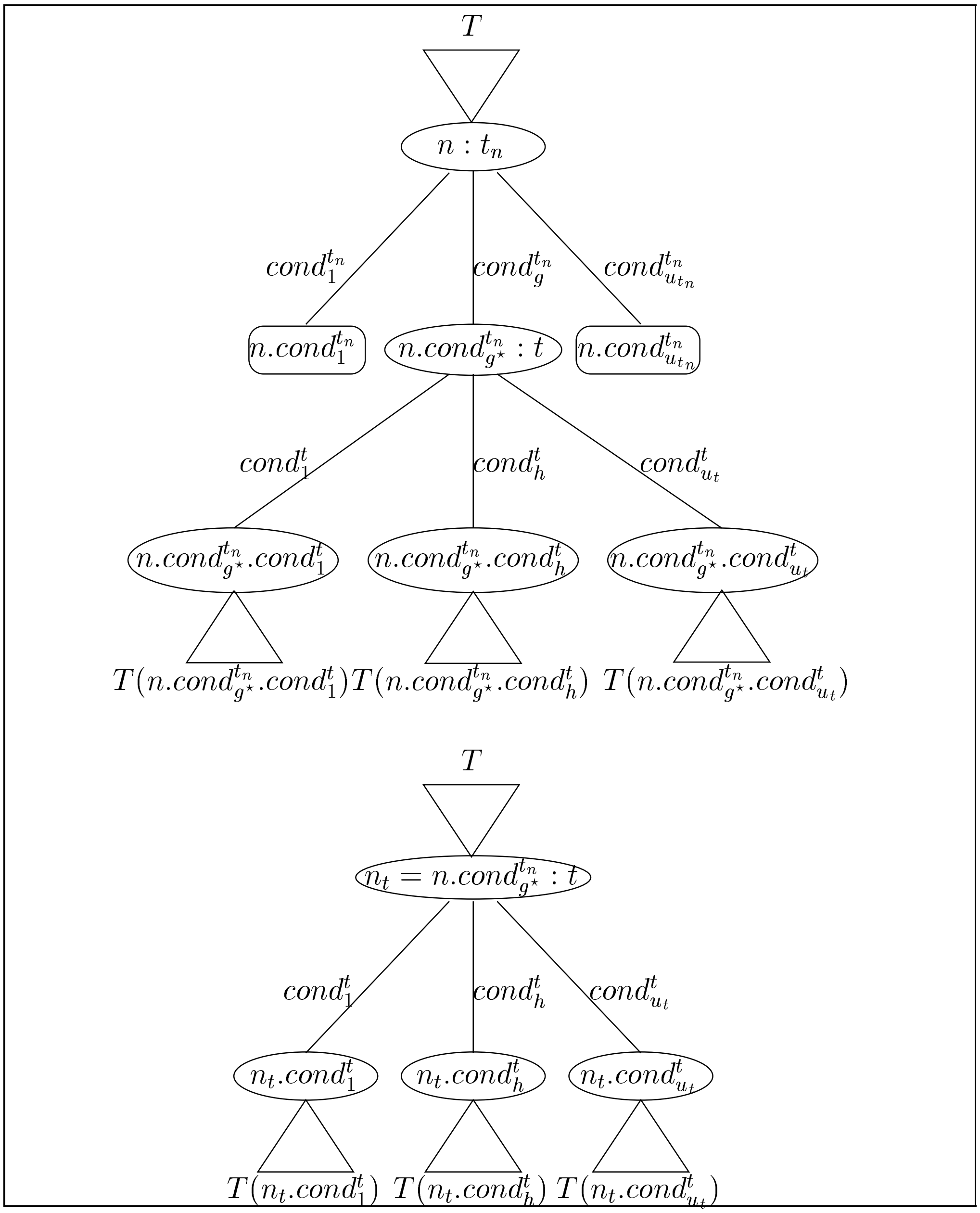

Abbildung 6.2: Bild zum Schritt 3 auf Seite 46

Eine formale Beschreibung, wie der Test ersetzt wird, ist in Abschnitt 5.2.2 auf Seite 46, Schritte 3a bis 3c gegeben.

Sei $n.cond_{g^\star}^{t_n}$ der innere Nachfolgerknoten von n. Ist der Test am Knoten $n.cond_{g^\star}^{t_n}$ identisch zu dem gewünschten Test t (siehe Abbildung 6.2), dann überspringen wir den Schritt 3a (den rekursiven Aufruf von **transposition**) und setzen mit Schritt 3b fort.

Ist der Test an $n.cond_{g^\star}^{t_n}$ unterschiedlich von t, so wird zuerst der gewünschte Test t durch die Ausführung der Funktion **transposition** am Knoten $n.cond_{g^\star}^{t_n}$ gesetzt. Danach werden normal die Schritte 3b bis 3c bearbeitet.

Hilfssatz 6.19 *Sei ein korrekter Teilbaum $T(n)$ bezüglich der Fuzzy-Menge κ, beginnend am Knoten n mit dem Test t_n, eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$ gegeben, d.h. κ ist identisch zu λ_n bzw. $\lambda_{root\to n}^{\lambda_{root}}$. Weiterhin seien die Nachfolgerknoten von n mit einer Ausnahme Blattknoten. Der Nachfolgerknoten von n, der innerer Knoten ist, sei mit dem Test t beschriftet.*

Die Ausführung der Schritte 3b bis 3c auf Seite 46 auf dem korrekten Teilbaum $T(n)$ liefert einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ bzw. λ_n zurück, dessen Wurzelknoten n_t mit dem Test t beschriftet ist.

Beweis Im Schritt 3b ersetzen wir den Teilbaum $T(n)$ durch den Teilbaum $T_t(n_t)$, wobei $T_t(n_t) = T(n.cond_{g^\star}^{t_n})$ gilt, d.h. an jedem Knoten des Teilbaumes $T_t(n_t)$ ist die gleiche Information gespeichert wie an dem entsprechenden Knoten aus $T(n.cond_{g^\star}^{t_n})$. Der Teilbaum $T(n.cond_{g^\star}^{t_n})$ ist als Teil des korrekten Teilbaumes $T(n)$ korrekt bezüglich der Fuzzy-Menge $\kappa'_{g^\star}$, die nach Definition 6.2 auf Seite 56 für jedes Beispiel $f \in \mathcal{L}$ wie folgt definiert ist:

$$\kappa'_{g^\star}(f) = \top(\kappa(f), \mu_{cond_{g^\star}^{t_n}}(f))$$

Da die Teilbäume $T(n.cond_{g^\star}^{t_n})$ und $T_t(n_t)$ zu diesem Zeitpunkt identisch sind, ist der Teilbaum $T_t(n_t)$ ebenfalls bezüglich $\kappa'_{g^\star}$ korrekt.

Unser Ziel ist es zu zeigen, daß nachdem alle Beispiele aus den Nachfolgerknoten von n, die Blattknoten sind, zum Knoten n_t bzw. zu $T_t(n_t)$ hinzugefügt werden (Ausführung des Schritts 3c), ein korrekter Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ entsteht. Durch das Hinzufügen der Beispiele zum Knoten n_t wird er markiert, was korrekt ist, da wir nicht überprüfen, ob die Methode $\mathcal{M}$ anhand der gespeicherten Information tatsächlich den gesetzten Test t auswählen würde.

Im Schritt 3c wird für jeden Nachfolgerknoten $n.cond_g^{t_n} \in \mathcal{N}(n) \setminus \{n.cond_{g^\star}^{t_n}\}$ und jedes Beispiel f, das dort gespeichert ist, die Funktion

$$\mathbf{addExampleToTree}(f, \Delta_{n_t,f}, n_t, \mathcal{T}' \cup \{t\})$$

mit

$$\Delta_{n_t,f} = \langle \lambda_{n.cond_g^{t_n}}(f) \rangle$$

ausgeführt. Nach Satz 6.16 auf Seite 63 erhalten wir einen Teilbaum, der korrekt bezüglich der Fuzzy-Menge κ_g^1 ist:

$$\kappa_g^1(f) = \kappa'_{g^\star}(f) + \langle \lambda_{n.cond_g^{t_n}}(f) \rangle$$

und

$$\kappa_g^1(f') = \kappa'_{g^\star}(f')$$

für $f' \neq f$.

Nach dem Hinzufügen des letzten, m-ten Beispiels aus dem Knoten $n.cond_g^{t_n}$ erhalten wir nach Satz 6.16 einen Teilbaum, der korrekt bezüglich der Fuzzy-Menge κ_g^m ist:

$$\kappa_g^m(f) = \kappa'_{g^\star}(f) + \langle \lambda_{n.cond_g^{t_n}}(f) \rangle$$

für alle f.

Nachdem alle $u_{t_n}-1$ Blattknoten bearbeitet sind, ist der Teilbaum $T_t(n_t)$ korrekt bezüglich der Fuzzy-Menge κ_t, wobei für jedes Beispiel f gilt:

$$\kappa_t(f) = \kappa'_{g^\star}(f) + \sum_{n.cond_g^{t_n} \in \mathcal{N}(n) \setminus \{n.cond_{g^\star}^{t_n}\}} \langle \lambda_{n.cond_g^{t_n}}(f) \rangle$$

Wir behaupten, daß die Fuzzy-Menge κ_t identisch zu der Fuzzy-Menge κ ist, d.h. für jedes Beispiel f gilt:

$$\begin{aligned}
\kappa_t(f) &= \kappa'_{g^\star}(f) + \sum_{n.cond_g^{t_n} \in \mathcal{N}(n) \setminus \{n.cond_{g^\star}^{t_n}\}} \langle \lambda_{n.cond_g^{t_n}}(f) \rangle \\
&\overset{T(n) \text{ korrekt bez. } \kappa}{=} \kappa'_{g^\star}(f) + \sum_{n.cond_g^{t_n} \in \mathcal{N}(n) \setminus \{n.cond_{g^\star}^{t_n}\}} \lambda^{\kappa}_{n \to n.cond_g^{t_n}}(f) \\
&\overset{\substack{\text{Gl. auf S. 73} \\ \text{Def. 6.2 auf S. 56}}}{=} \top(\kappa(f), \mu_{cond_{g^\star}^{t_n}}(f)) + \\
&\qquad \sum_{n.cond_g^{t_n} \in \mathcal{N}(n) \setminus \{n.cond_{g^\star}^{t_n}\}} \top(\kappa(f), \mu_{cond_g^{t_n}}(f)) \\
&= \sum_{n.cond_j^{t_n} \in \mathcal{N}(n)} \top(\kappa(f), \mu_{cond_j^{t_n}}(f)) \\
&\overset{\text{Ann. 4.3 auf S. 36}}{=} \top(\kappa(f), \sum_{n.cond_j^{t_n} \in \mathcal{N}(n)} \mu_{cond_j^{t_n}}(f)) \\
&= \top(\kappa(f), 1) \\
&= \kappa(f)
\end{aligned}$$

Da die Fuzzy-Mengen κ und κ_t identisch sind, erhalten wir am Ende des Schritts 3c einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ, dessen Wurzelknoten mit dem Test t beschriftet ist. □

Wir haben gezeigt, wieso jeder Teilbaum $T(n)$ eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes durch einen Teilbaum $T_t(n_t)$ ersetzt werden darf, der aus dem Teilbaum $T(n)$ und der Ausführung der Schritte 3b bis 3c gewonnen wurde. Dabei wird keine Information verloren und der Gesamtbaum bleibt konsistent und korrekt, weil der neue Teilbaum $T_t(n_t)$ korrekt bezüglich der gleichen Fuzzy-Menge κ ($\kappa = \lambda_n$ bzw. $\lambda^{\lambda_{root}}_{root \to n}$) ist, wie der ursprüngliche Teilbaum $T(n)$.

Zwei oder mehrere Nachfolgerknoten sind innere Knoten, mindestens einer ist Blattknoten

Werden beim inkrementellen Aufbau eines Fuzzy-Entscheidungsbaumes mehrfache Tests verwendet, so kann der Fall auftreten, wo zwei oder mehrere Nachfolgerknoten keine Blattknoten sind und mindestens ein Nachfolgerknoten ein Blattknoten ist (siehe Abbildung 6.3).

Diese Situation wird auf die nachfolgend beschriebene zurückgeführt, indem die Nachfolger-Blattknoten durch innere Knoten ersetzt werden. Eine formale Beschreibung des Ersetzens ist im Abschnitt 5.2.2 auf Seite 47, Schritte 4a bis 4b gegeben.

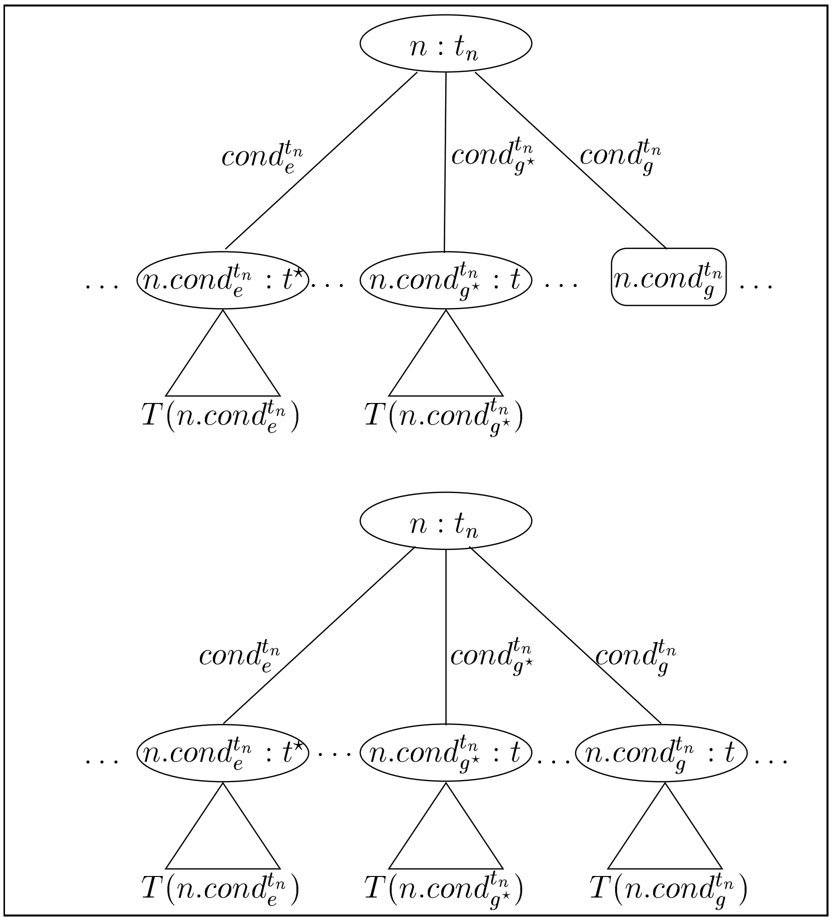

Abbildung 6.3: Bild zum Schritt 4 auf Seite 47

Alle Nachfolgerknoten eines Knotens sind innere Knoten

Eine formale Beschreibung des Austausches ist im Abschnitt 5.2.2 auf Seite 47, Schritte 5a bis 5f gegeben. Im Schritt 5a wird durch die rekursiven Aufrufe der Funktion **transposition** dafür gesorgt, daß alle inneren Nachfolgerknoten mit dem gewünschten Test beschriftet werden. Sind alle Nachfolger bereits mit dem gewünschten Test beschriftet (siehe Abbildung 6.4), so muß der Schritt 5a (der rekursive Aufruf von **transposition**) nicht ausgeführt werden und wir setzen mit der Bearbeitung der Schritte 5b bis 5f fort.

Hilfssatz 6.20 *Sei ein korrekter Teilbaum $T(n)$ bezüglich der Fuzzy-Menge κ, beginnend am Knoten n mit dem Test t_n, eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$ gegeben, d.h. κ ist identisch zu λ_n bzw. $\lambda_{root \to n}^{\lambda_{root}}$. Weiterhin seien alle Nachfolgerknoten von n innere Knoten, die mit dem Test t beschriftet sind.*

Die Ausführung der Schritte 5b bis 5f auf Seite 48 liefert einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ bzw. λ_n zurück, dessen Wurzelknoten n_t mit dem Test t beschriftet ist.

Beweis In den Schritten 5b bis 5e werden alle Nachfolgerknoten von n bzw. die Teilbäume $T(n.cond_g^{t_n}), 1 \leq g \leq u_{t_n}$ entfernt, der Knoten n mit dem gewünschten Test t beschriftet, markiert und als n_t bezeichnet. Anschließend wird im Schritt 5e für jede Bedingung $cond_h^t, 1 \leq h \leq u_t$ des Tests t eine ausgehende Kante mit einem neuen, leeren Nachfolgerknoten $n_t.cond_h^t$ erzeugt.

Zu diesem Zeitpunkt besteht der Teilbaum $T_t(n_t)$ aus dem Wurzelknoten n_t, beschriftet mit t und seinen leeren Nachfolgerknoten. Für den Knoten n_t haben sich gegenüber dem Knoten n der Test (die Beschriftung), die Markierung und die Nachfolgerknoten verändert. Der Knoten muß markiert werden, weil wir uns nicht sicher sind, daß der Test t tatsächlich durch die Methode $\mathcal{M}$ anhand der gespeicherten Daten ausgewählt würde. Die gespeicherte Information am Knoten n_t bleibt korrekt, somit ist er nach Definition 6.4 auf Seite 57 korrekt bezüglich sich selbst und der Fuzzy-Menge κ.

Jeder Teilbaum $T_t(n_t.cond_h^t)$ ist korrekt bezüglich der Fuzzy-Menge κ^0, zu der jedes Beispiel aus $\mathcal{L}$ mit dem Zugehörigkeitsgrad von Null gehört.

Im Schritt 5f werden die neuen Knoten $n_t.cond_h^t, 1 \leq h \leq u_t$ mit dem Test t_n beschriftet, markiert (es ist nicht garantiert, daß der Test t_n der optimale Test für den Knoten ist) und es wird für jede Bedingung $cond_g^{t_n}, 1 \leq g \leq u_{t_n} \in t_n$ eine ausgehende Kante mit einem Knoten $n_t.cond_h^t.cond_g^{t_n}$ definiert. An Stelle jedes Knotens $n_t.cond_h^t.cond_g^{t_n}, 1 \leq h \leq u_t, 1 \leq g \leq u_{t_n}$ können wir die alten Teilbäume $T(n.cond_g^{t_n}.cond_h^t)$ hinzufügen (siehe Abbildung 6.5 auf Seite 79), da die Beispielmengen, worauf jeder neue Knoten $n_t.cond_h^t.cond_g^{t_n}$ und jeder alte Knoten $n.cond_g^{t_n}.cond_h^t$ gebildet sind, identisch sind. Die Knoten werden von den gleichen Beispielen mit den gleichen Zugehörigkeitsgraden erreicht, da ausgehend vom Knoten n sich nur die Reihenfolge der Bedingungen verändert hat. Jeder Teilbaum $T(n.cond_g^{t_n}.cond_h^t)$ als Teil des korrekten Teilbaumes $T(n)$ ist korrekt bezüglich der Fuzzy-Menge $\kappa''_{g,h}$, wobei für jedes Beispiel f nach Definition 6.2 gilt:

$$\begin{aligned} \kappa''_{g,h}(f) &= \top(\kappa(f), \mu_{cond_g^{t_n}}(f), \mu_{cond_h^t}(f)) \\ &= \top(\kappa(f), \mu_{cond_h^t}(f), \mu_{cond_g^{t_n}}(f)) \end{aligned}$$

Der Teilbaum $T_t(n_t)$ besteht jetzt aus dem Wurzelknoten n_t, seinen neuen Nachfolgerknoten $n_t.cond_h^t, 1 \leq h \leq u_t$ und den alten Teilbäumen $T(n.cond_g^{t_n}.cond_h^t), 1 \leq g \leq u_{t_n}$, wobei an den Knoten $n_t.cond_h^t$ keine Information gespeichert ist.

Im Schritt 5(f)iv werden die Größen, die an jedem Knoten $n_t.cond_h^t$ gespeichert werden müssen, aus den gespeicherten Größen an seinen Nachfolgerknoten ermittelt und gespeichert. Wegen der Korrektheit der alten Teilbäume $T(n.cond_g^{t_n}.cond_h^t)$ und der Eigenschaft kompositorisch, die für jeden Teilbaum $T_t(n_t.cond_h^t)$ erfüllt ist (Annahme 4.3 auf Seite 36 gilt), behaupten wir, daß der Wurzelknoten jedes Teilbaumes $T_t(n_t.cond_h^t), 1 \leq h \leq u_t$ korrekt bezüglich der folgenden Fuzzy-Menge κ'_h ist:

$$\begin{aligned} \kappa'_h(f) &= \sum_{g=1}^{u_{t_n}} \lambda^{\kappa''_{g,h}}_{n.cond_g^{t_n}.cond_h^t \to n.cond_g^{t_n}.cond_h^t}(f) \\ &\overset{\text{Def. 6.2 auf S. 56}}{=} \sum_{g=1}^{u_{t_n}} \kappa''_{g,h}(f) \\ &\overset{\text{siehe oben}}{=} \sum_{g=1}^{u_{t_n}} \top(\kappa(f), \mu_{cond_g^{t_n}}(f), \mu_{cond_h^t}(f)) \\ &\overset{\text{Ann. 4.3 auf S. 36}}{=} \top(\kappa(f), \sum_{g=1}^{u_{t_n}} \mu_{cond_g^{t_n}}(f), \mu_{cond_h^t}(f)) \\ &= \top(\kappa(f), 1, \mu_{cond_h^t}(f)) \\ &= \top(\kappa(f), \mu_{cond_h^t}(f)) \end{aligned}$$

Dafür betrachten wir den Knoten $n_t.cond_h^t$ und werden zeigen, daß die Gleichung (6.4) auf Seite 57 erfüllt ist.

$$\begin{aligned}
\langle|\lambda_{n_t.cond_h^t}|\rangle &\overset{\text{Schritt 5(f)iv}}{=} \sum_{g=1}^{u_{t_n}} \langle|\lambda_{n.cond_g^{t_n}.cond_h^t}|\rangle \\
&\overset{T(n.cond_g^{t_n}.cond_h^t)\text{ korrekt bez. }\kappa''_{g,h}}{=} \sum_{g=1}^{u_{t_n}} |\lambda^{\kappa''_{g,h}}_{n.cond_g^{t_n}.cond_h^t \rightarrow n.cond_g^{t_n}.cond_h^t}| \\
&= \sum_{g=1}^{u_{t_n}} \sum_{f=1}^{m} \lambda^{\kappa''_{g,h}}_{n.cond_g^{t_n}.cond_h^t \rightarrow n.cond_g^{t_n}.cond_h^t}(f) \\
&= \sum_{f=1}^{m} \sum_{g=1}^{u_{t_n}} \lambda^{\kappa''_{g,h}}_{n.cond_g^{t_n}.cond_h^t \rightarrow n.cond_g^{t_n}.cond_h^t}(f) \\
&\overset{\text{siehe oben}}{=} \sum_{f=1}^{m} \kappa'_h(f) \\
&= |\lambda^{\kappa'_h}_{n_t.cond_h^t \rightarrow n_t.cond_h^t}|
\end{aligned}$$

Die Gleichungen (6.5) bis (6.8) können analog gezeigt werden, also ist der Knoten $n_t.cond_h^t$ korrekt bezüglich sich selbst und der Fuzzy-Menge κ'_h. Außerdem können wir leicht beweisen, daß der Wurzelknoten jedes Teilbaumes $T_t(n_t.cond_h^t.cond_g^{t_n}) = T(n.cond_g^{t_n}.cond_h^t)$ korrekt bezüglich des Knotens $n_t.cond_h^t$ und der Fuzzy-Menge κ'_h ist, da für alle f gilt:

$$\begin{aligned}
\kappa''_{g,h}(f) &= \top(\kappa(f), \mu_{cond_h^t}(f), \mu_{cond_g^{t_n}}(f)) \\
&= \top(\kappa'_h(f), \mu_{cond_g^{t_n}}(f))
\end{aligned}$$

Somit ist jeder Teilbaum $T_t(n_t.cond_h^t), 1 \leq h \leq u_t$ nach Definition 6.5 auf Seite 58 korrekt bezüglich der Fuzzy-Menge κ'_h.

Wir können weiter beweisen, daß der Wurzelknoten jedes Teilbaumes $T_t(n_t.cond_h^t), 1 \leq h \leq u_t$ korrekt bezüglich n_t und κ ist, da $\kappa'_h(f) = \top(\kappa(f), \mu_{cond_h^t}(f))$ für alle f, und die Teilbäume $T_t(n_t.cond_h^t)$ korrekt bezüglich κ'_h sind. Es folgt, daß der Teilbaum $T_t(n_t)$ nach Definition 6.5 auf Seite 58 korrekt bezüglich der Fuzzy-Menge κ ist. □

Wir haben bewiesen, daß durch die Ausführung der Schritte 5b bis 5f auf Seite 48 auf einem korrekten Teilbaum $T(n)$ bezüglich einer Fuzzy-Menge κ, ein korrekter Teilbaum $T_t(n_t)$ bezüglich κ zurückgeliefert wird. Analog zu den vorigen Fällen können wir folglich argumentieren, daß ein korrekter Teilbaum eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes durch den Teilbaum ersetzt werden kann, der durch die Ausführung der Schritte 5b bis 5f auf Seite 48 entsteht.

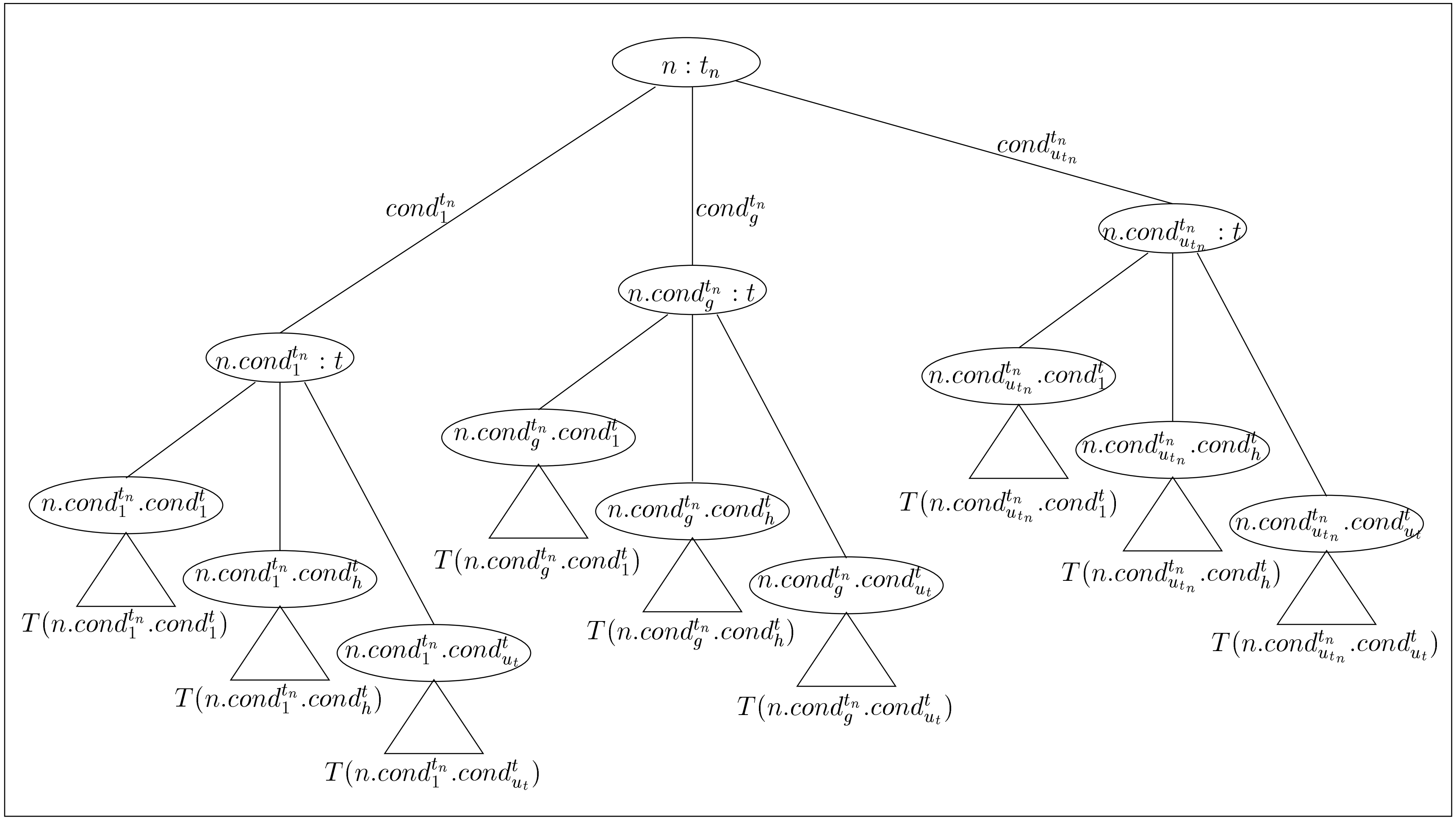

Abbildung 6.4: Ausgangssituation zum Schritt 5 auf Seite 47

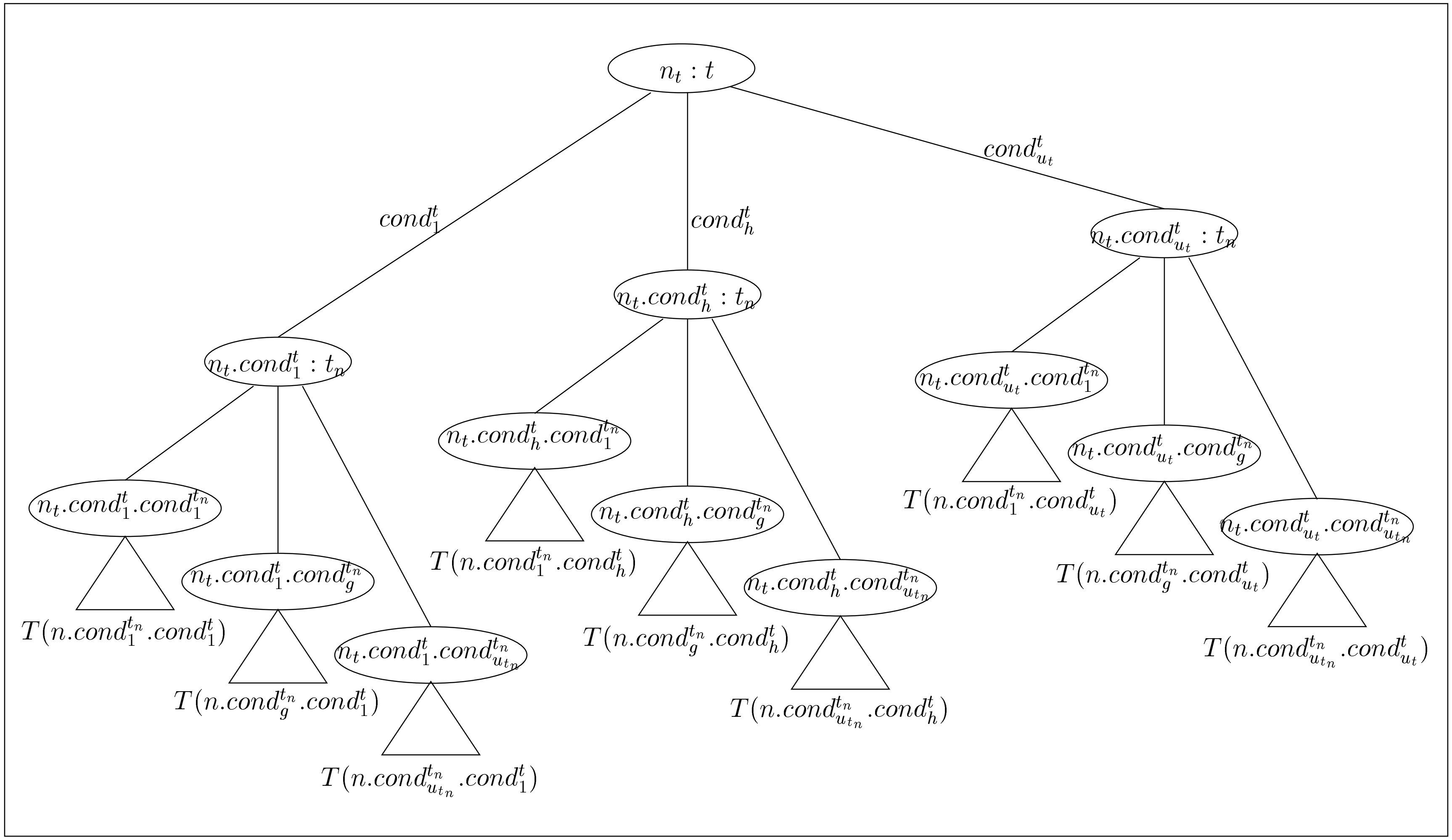

Abbildung 6.5: Ergebnis nach der Ausführung des Schritts 5 auf Seite 47

Satz 6.21 *Sei ein korrekter Teilbaum $T(n)$ bezüglich der Fuzzy-Menge κ, beginnend an einem inneren Knoten n mit dem Test t_n, eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes bezüglich der Lernmenge $\mathcal{L}$, der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ gegeben, d.h. κ ist identisch zu λ_n bzw. $\lambda_{root\to n}^{\lambda_{root}}$. Weiterhin sei $t \neq t_n$ ein Test und $\mathcal{T}' \subseteq \mathcal{T} \setminus \{t\}$ die Menge der verfügbaren Tests, d.h. die Tests (außer t), die nicht auf dem Weg vom Wurzelknoten root bis zum Knoten n liegen.*

Die Funktion **transposition**$(n, t_n, t, \mathcal{T}')$ *liefert einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ bzw. λ_n zurück, dessen Wurzelknoten n_t mit dem Test t beschriftet ist.*

Beweis Wir beweisen diese Aussage durch Induktion über die Höhe h des Teilbaumes und die Größe z der Menge $\mathcal{T}'$ (Induktionsstruktur bereits zum Beweis des Satzes 6.1 auf Seite 55 eingeführt).

Behauptung Aufgerufen mit einem, bezüglich der Fuzzy-Menge κ, korrekten Teilbaum $T(n)$ der Höhe h und mit einer Menge von verfügbaren Tests der Größe z, liefert die Funktion **transposition** einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ zurück, dessen Wurzelknoten n_t mit dem Test t beschriftet ist.

Induktionsanfang Die Behauptung gilt für das Paar $(z = 1, h = 1)$.

Bei einem Teilbaum $T(n)$ der Höhe 1 handelt sich um einen inneren Knoten n, dessen Nachfolger Blattknoten sind. Aufgerufen mit solch einem korrekten Teilbaum führt die Funktion **transposition** den Schritt 2 auf Seite 45 durch.

Nach Hilfssatz 6.18 auf Seite 70 liefert die Funktion einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ zurück, dessen Wurzelknoten mit dem Test t beschriftet ist.

Die Behauptung gilt also für einen Teilbaum der Höhe 1.

Induktionsschritt

Induktionsvoraussetzung Die Behauptung gilt für alle Paare (z', h'), für welche $(z', h') < (z, h)$ gilt (bezüglich der wohl-fundierten Ordnung von Seite 55).

Induktionsbehauptung Die Behauptung ist auch für das Paar (z, h) erfüllt. Wir unterscheiden folgende Fälle:

- Aufruf des Schritts 2 auf Seite 45, d.h. alle Nachfolger sind Blattknoten: die Behauptung gilt analog zum Induktionsanfang
- Aufruf des Schritts 3 auf Seite 46, d.h. genau ein Nachfolgerknoten $n.cond_{g^\star}^{t_n}$ ist innerer Knoten:
  Im Schritt 3a wird überprüft, ob der Test am Knoten $n.cond_{g^\star}^{t_n}$ identisch zu dem gewünschten Test t ist. Man unterscheide:
  - es wird sofort mit der Bearbeitung der Schritte 3b bis 3c fortgesetzt, weil der Test identisch ist. Wegen Hilfssatz 6.19 auf Seite 73 ist dann die Behauptung erfüllt.
  - ansonsten wird **transposition** auf dem Teilbaum $T(n.cond_{g^\star}^{t_n})$ rekursiv aufgerufen, der korrekt bezüglich folgender Fuzzy-Menge κ' ist:

$$\kappa'(f) = \top(\kappa(f), \mu_{cond_{g^\star}^{t_n}}(f)) \quad \text{für alle } f$$

Bei diesem Aufruf ist die Höhe des Teilbaumes $T(n.cond_{g^\star}^{t_n})$ kleiner (während die Testmenge gleich bleibt), also wird wegen der Induktionsvoraussetzung ein korrekter Teilbaum $T_t(n.cond_{g^\star}^{t_n})$ bezüglich der

Fuzzy-Menge κ' zurückgeliefert, dessen Wurzelknoten $n.cond_{g^\star}^{t_n}$ mit dem Test t beschriftet ist.
Dieser Teilbaum kann anstatt des alten Teilbaumes $T(n.cond_{g^\star}^{t_n})$ gesetzt werden, da die beiden Teilbäume die gleiche Menge von Beispielen repräsentieren. Dabei bleibt der ursprüngliche Teilbaum $T(n)$ korrekt bezüglich κ.
Somit ist die Voraussetzung erfüllt, um die Schritte 3b bis 3c auf dem korrekten Teilbaum $T(n)$ auszuführen. Nach Hilfssatz 6.19 wird ein korrekter Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ zurückgeliefert, dessen Wurzelknoten mit dem Test t beschriftet ist. Also ist die Behauptung erfüllt.

- Aufruf des Schritts 4 auf Seite 47, d.h. es gibt zwei oder mehrere Nachfolgerknoten $n.cond_{g^\star}^{t_n} \in \mathcal{N}(n)$, die innere Knoten sind, sowie mindestens einen Blattknoten $n.cond_g^{t_n} \in \mathcal{N}(n)$:
Ziel dieses Schritts ist es, jeden Blattknoten $n.cond_g^{t_n}$ in einen inneren Knoten umzuwandeln. Wegen Korrektheit des Teilbaumes $T(n)$ bezüglich der Fuzzy-Menge κ ist jeder Blattknoten $n.cond_g^{t_n}$ einerseits korrekt bezüglich n und κ, anderseits bezüglich sich selbst und der Fuzzy-Menge κ' mit

$$\kappa'(f) = \top(\kappa(f), \mu_{cond_g^{t_n}}(f)) \quad \text{für alle } f$$

Mit Hilfe der Werte $\langle \lambda_{n.cond_g^{t_n}}(f) \rangle$, die für jedes Beispiel $f \in \mathcal{L}$, das den Blattknoten erreicht hat, gespeichert sind, werden im Schritt 4(a)i für jeden Test und jede Klasse die Größen der Fuzzy-Mengen:

$$\lambda_{n.cond_g^{t_n}} | cond_d^{t^\star} \;,\; \lambda_{n.cond_g^{t_n}} | cond_d^{t^\star} | c_k \;,\; \lambda^{missing}_{n.cond_g^{t_n}, t^\star}$$

ausgerechnet und gespeichert. Die so ausgerechneten Werte sind korrekt bezüglich des Knotens n und der Fuzzy-Menge κ und bezüglich $n.cond_g^{t_n}$ und κ', da die Werte $\langle \lambda_{n.cond_g^{t_n}}(f) \rangle$ für jedes Beispiel korrekt sind.
In den Schritten 4(a)ii bis 4(a)iv wird der Knoten $n.cond_g^{t_n}$ mit dem gewünschten Test t beschriftet, markiert (da der Knoten eigentlich die Abbruchbedingungen erfüllt) und es werden u_t leere Nachfolgerknoten von $n.cond_g^{t_n}$ (d.h. korrekte Teilbäume) erzeugt. Jeder Teilbaum $T(n.cond_g^{t_n}.cond_h^t)$, $1 \leq h \leq u_t$ ist korrekt bezüglich einer Fuzzy-Menge κ^0 zu der alle Beispiele mit dem Zugehörigkeitsgrad von Null gehören.
Das Hinzufügen der Beispiele aus $n.cond_g^{t_n}$ zu $T(n.cond_g^{t_n}.cond_h^t)$ durch die Ausführung des Schritts 4(a)v verläuft analog zum Schritt 3(d)iv auf Seite 44 der Funktion **addExampleToTree**. Analog zum Beweis des Satzes 6.16 auf Seite 65 wird argumentiert, daß der so erhaltene Teilbaum $T(n.cond_g^{t_n})$ korrekt bezüglich der Fuzzy-Menge κ' bleibt. Somit bleibt der Teilbaum $T(n)$ ebenfalls korrekt bezüglich κ.
Der nächste Schritt 4b verweist auf der Ausführung des Schritts 5 (siehe unten) mit dem korrekten Teilbaum $T(n)$ bezüglich κ.
- Aufruf des Schritts 5 auf Seite 47, d.h. jeder Nachfolgerknoten $n.cond_g^{t_n}$ ist innerer Knoten:
Im Schritt 5a wird für jeden Nachfolgerknoten $n.cond_g^{t_n}$ getestet, ob sein Test identisch zu dem gewünschten Test t ist.
Falls dies für einen Knoten $n.cond_g^{t_n}$ nicht der Fall ist, so wird die Funktion **transposition** mit dem Teilbaum $T(n.cond_g^{t_n})$ rekursiv aufgerufen, der (wegen Korrektheit von $T(n)$) korrekt bezüglich der Fuzzy-Menge κ' ist,

mit

$$\kappa'(f) = \top(\kappa(f), \mu_{cond_g^{t_n}}(f)) \quad \text{für alle } f$$

Der rekursive Aufruf von **transposition** erfolgt auf einem korrekten Teilbaum $T(n.cond_g^{t_n})$, wobei die Menge der verfügbaren Tests verkleinert wird. Wegen Induktionsvoraussetzung liefert dieser Aufruf einen korrekten Teilbaum $T_t(n.cond_g^{t_n})$ bezüglich der Fuzzy-Menge κ' zurück, dessen Wurzelknoten mit dem Test t beschriftet ist. Somit bleibt der Teilbaum $T(n)$ trotz des Austausches des Teilbaumes $T(n.cond_g^{t_n})$ durch den Teilbaum $T_t(n.cond_g^{t_n})$ korrekt bezüglich κ.
Dies ist die Voraussetzung dafür, daß die Ausführung der Schritte 5b bis 5(f)iv mit dem korrekten Teilbaum $T(n)$ nach Hilfssatz 6.20 auf Seite 75 einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ zurück liefert, dessen Wurzelknoten mit dem Test t beschriftet ist, womit die zu beweisende Behauptung erfüllt ist.

Durch Induktion haben wir gezeigt, daß die Funktion **transposition** aufgerufen mit einem korrekten Teilbaum $T(n)$ der Höhe $h \geq 1$, einen korrekten Teilbaum zurück liefert, wobei die beiden Teilbäume korrekt bezüglich derselben Fuzzy-Menge sind, d.h. an den Wurzelknoten dieser Teilbäume sind die gleichen Beispiele mit identischen Zugehörigkeitsgraden vertreten. □

Jetzt wissen wir, daß **transposition** einen korrekten Teilbaum so in einen anderen korrekten Teilbaum umstrukturiert, daß die beiden Bäume äquivalente Information darstellen und der Wurzelknoten des neuen Teilbaumes mit dem gewünschten Test beschriftet ist. Deswegen kann die Funktion **ensureBestTest** einen korrekten Teilbaum durch den Teilbaum ersetzen, der durch die Ausführung von **transposition** entstanden ist, und überführt dabei einen korrekten Entscheidungsbaum in einen korrekten und optimalen Entscheidungsbaum . Diese Eigenschaft von **ensureBestTest** werden wir im nächsten Unterkapitel beweisen.

6.4.3 Eigenschaften von ensureBestTest

Ziel der Funktion **ensureBestTest** ist es, den durch das Hinzufügen von Beispielen generierten inkrementellen Fuzzy-Entscheidungsbaum in einen optimalen Entscheidungsbaum umzustrukturieren. Voraussetzung für den Erfolg von **ensureBestTest** ist ein korrekter inkrementeller Fuzzy-Entscheidungsbaum, d.h.

- die Markierung der Knoten muß korrekt sein
- die gespeicherte Information muß korrekt sein
- jeder Blattknoten muß die Abbruchbedingungen der Funktion $\mathcal{F}$ erfüllen und gemäß der Funktion $\mathcal{F}$ beschriftet sein

Würde ein Knoten, dessen Information durch das Hinzufügen eines Beispiels oder dessen Test durch **transposition** verändert wurde, nicht markiert, so wird er durch **ensureBestTest** nicht untersucht und es könnte vorkommen, daß ein falscher Test als optimal betrachtet würde.

Wäre der Entscheidungsbaum nicht korrekt, so könnte ein nicht optimaler Test an einem Knoten wegen der falsch gespeicherten Daten durch die Methode $\mathcal{M}$ ausgewählt und gesetzt werden oder es könnte passieren, daß die Funktion $\mathcal{F}$ unter Beachtung der falsch gespeicherten Information an einem Knoten, diesen Knoten falsch als Blattknoten oder inneren Knoten definiert.

Ein Blattknoten wird generell durch die Funktion **ensureBestTest** nicht untersucht. Würde der Knoten unter Mißachtung der Funktion $\mathcal{F}$ falsch als Blattknoten definiert, so könnte der Entscheidungsbaum eine ungenauere, sogar falsche Klassifikation liefern.

Satz 6.22 *Sei ein, bezüglich der Fuzzy-Menge κ, korrekter Teilbaum $T(n)$ eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes IFT bezüglich der Lernmenge $\mathcal{L}$, der Klassenmenge $\mathcal{C}$ und der Menge von zulässigen Tests $\mathcal{T}$ gegeben, d.h. κ ist identisch zu λ_n bzw. $\lambda_{root\to n}^{\lambda_{root}}$. Weiterhin sei $\mathcal{T}' \subseteq \mathcal{T}$ die Menge der verfügbaren Tests, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten des Baumes bis zum Knoten n liegen.*

Die Funktion **ensureBestTest**$(n, \mathcal{T}')$ *liefert einen korrekten und optimalen Teilbaum $T_e(n_e)$ bezüglich der Fuzzy-Menge κ zurück.*

Beweis Induktion über die Menge $\mathcal{T}' \subseteq \mathcal{T}$ der verfügbaren Tests, d.h. die Tests, die nicht auf dem Weg vom Wurzelknoten des Baumes bis zum aktuellen Knoten liegen.

Behauptung Aufgerufen mit einem, bezüglich der Fuzzy-Menge κ, korrekten Teilbaum $T(n)$ und mit einer Menge von verfügbaren Tests $\mathcal{T}' \subseteq \mathcal{T}$ der Größe z, liefert die Funktion **ensureBestTest**$(n, \mathcal{T}')$ einen korrekten und optimalen Teilbaum $T_e(n_e)$ bezüglich der Fuzzy-Menge κ zurück.

Induktionsanfang Die Behauptung gilt für $\mathcal{T}' = \emptyset$, d.h. $z = 0$

Im Schritt 1 wird getestet, ob der Knoten n ein Blattknoten ist. Wir unterscheiden folgende Fälle:

- der Knoten ist ein Blattknoten:

  In diesem Fall wird der Teilbaum $T(n)$ unverändert zurückgeliefert, d.h. wir setzen $T_e(n_e) = T(n)$. Offensichtlich ist der Teilbaum $T_e(n_e) = n_e$ korrekt bezüglich κ. Der Teilbaum ist auch optimal, da ein korrekter Blattknoten immer optimal ist.

- der Knoten ist kein Blattknoten:

  Im nächsten Schritt 2 testen wir, ob die Abbruchbedingungen für den Knoten n erfüllt sind. Da $\mathcal{T}' = \emptyset$, ist die Funktion $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\dots,q}, \mathcal{T}')$ erfüllt und der Teilbaum $T(n)$ wird als ein Blattknoten n_e definiert.

  Der Knoten wird im Schritt 2a gemäß $\mathcal{F}$ beschriftet. Die Größen für die Fuzzy-Mengen λ_{n_e} und $\lambda_{n_e}|c_k, c_k \in \mathcal{C}$ werden aus den gespeicherten Informationen am Knoten n übernommen. Also wegen Korrektheit von n bezüglich κ sind die Gleichungen (6.2) und (6.3) für den Blattknoten n_e ebenfalls bezüglich κ erfüllt. Am Blattknoten n_e müssen zusätzlich die einzelnen Beispiele gespeichert werden. Diese Zugehörigkeitsgrade werden im Schritt 2b ausgerechnet und gespeichert, d.h. für jedes Beispiel f speichern wir den Wert $\langle\lambda_{n_e}(f)\rangle = \langle\langle\lambda_n(f)\rangle\rangle$, was nach Korollar 6.9 auf Seite 59 genau dem Wert $\kappa(f)$ entspricht.

  In den Schritten 2c bis 2e wird die überflüssige Information (solche, die an einem Blattknoten nicht gespeichert sein soll) gelöscht und die Nachfolger von n entfernt.

  Damit ist der Knoten n_e nach Definition 6.3 auf Seite 57 korrekt bezüglich sich selbst und der Fuzzy-Menge κ. Wir erhalten einen Blattknoten n_e, der korrekt und nach Definition 6.12 auf Seite 60 optimal ist.

Die Funktion **ensureBestTest** liefert damit den korrekten und optimalen Teilbaum $T_e(n_e)$ zurück.

Induktionsschritt

Induktionsvoraussetzung Die Behauptung gilt für jede Menge $\mathcal{T}''$ mit $z-1$ Tests.

Induktionsbehauptung Die Behauptung gilt auch für jede Menge $\mathcal{T}'$ mit z Tests.

Im ersten Schritt 1 der Funktion testen wir, ob der Knoten n ein Blattknoten ist.

Ist dies der Fall, so bricht **ensureBestTest** ab und liefert den Teilbaum $T_e(n_e)$ zurück, der aus einem Blattknoten n_e besteht, der identisch zum Knoten n ist. Somit ist $T(n_e)$ korrekt bezüglich κ und optimal.

Im Schritt 2 überprüfen wir, ob $\mathcal{F}(\langle|\lambda_n|\rangle, \langle|\lambda_n|c_k|\rangle_{k=1,2,\ldots,q}, \mathcal{T}')$ erfüllt ist. Für den Fall, daß die Funktion $\mathcal{F}$ erfüllt ist, haben wir bereits gezeigt, daß durch die Ausführung des Schritts 2 ein korrekter bezüglich κ und optimaler Teilbaum $T_e(n_e)$ zurückgeliefert wird (siehe Induktionsanfang).

Ansonsten setzen wir in Abhängigkeit davon, ob der Knoten n markiert ist oder nicht, mit der Bearbeitung des Schritts 4 bzw. des Schritts 7 fort.

- Wenn der Knoten markiert ist, wird im Schritt 4 anhand der gespeicherten Information der optimale Test bezüglich $\mathcal{M}$ ausgewählt. Wegen Korrektheit von n wird tatsächlich der optimale Test t bezüglich $\mathcal{M}$ bestimmt.

  Ist der Knoten n mit einem Test verschieden von t beschriftet, so wird im Schritt 5 die Funktion **transposition** mit dem korrekten Teilbaum $T(n)$ bezüglich der Fuzzy-Menge κ aufgerufen. Nach Satz 6.21 auf Seite 80 liefert die Funktion **transposition** einen korrekten Teilbaum $T_t(n_t)$ bezüglich der Fuzzy-Menge κ, dessen Wurzelknoten (im folgenden n_e) mit dem Test t beschriftet ist, zurück.

  Nach der Ausführung des Schritts 5 ist garantiert, daß der Wurzelknoten n_e des Teilbaumes $T(n_e)$ mit dem optimalen Test beschriftet ist. Jetzt dürfen wir im Schritt 6 die Markierung des Knotens n_t zurücksetzen (weil außerdem geprüft wurde, daß $\mathcal{F}$ nicht erfüllt ist) und mit der Bearbeitung des korrekten Teilbaumes $T(n_e)$ bezüglich κ im Schritt 7 fortsetzen.

- Wenn der Knoten n nicht markiert ist, dann wissen wir wegen Korrektheit von n, daß der richtige Test gesetzt ist und die Abbruchbedingungen nicht erfüllt sind. Also der Knoten $n_e = n$ ist korrekt bezüglich sich selbst und κ und optimal.

Im letzten Schritt 7 wird für jeden Nachfolgerknoten $n_e.cond_h^t, 1 \le h \le u_t$ (gemeint ist der korrekte Teilbaum $T(n_e.cond_h^t)$ bezüglich κ' mit $\kappa'(f) = \top(\kappa(f), \mu_{cond_h^t}(f))$ für alle f) rekursiv die Funktion

$$\textbf{ensureBestTest(}n_e.cond_h^t, \mathcal{T}' \setminus \{t\}\textbf{)}$$

aufgerufen.

Die Menge $\mathcal{T}' \setminus \{t\}$ womit die Funktion aufgerufen wird, hat $z-1$ Tests und wegen der Induktionsvoraussetzung ist jeder zurückgelieferte Teilbaum $T_e(n_e.cond_h^t)$ optimal und korrekt bezüglich der Fuzzy-Menge κ'.

Nach Definition 6.5 auf Seite 58 bleibt der Teilbaum $T_e(n_e)$ (nachdem jeder Teilbaum $T(n_e.cond_h^t)$ durch den Teilbaum $T_e(n_e.cond_h^t)$ ersetzt wird) korrekt bezüglich κ.

Der Teilbaum $T_e(n_e)$ ist auch optimal, da jeder $T_e(n_e.cond_h^t)$ optimal ist und der Knoten n_e ebenfalls optimal ist. Damit ist der Teilbaum $T_e(n_e)$, der zurückgeliefert wird, optimal und korrekt bezüglich κ.

Durch Induktion haben wir gezeigt, daß jeder korrekte Teilbaum $T(n)$ eines korrekten inkrementellen Fuzzy-Entscheidungsbaumes durch einen korrekten und optimalen Teilbaum $T_e(n_e)$ ersetzt werden kann, ohne die Korrektheit des Gesamtbaumes zu verletzen (der Gesamtbaum repräsentiert weiterhin die gleichen Beispiele), wobei der Teilbaum $T_e(n_e)$ durch die Ausführung der Funktion **ensureBestTest** auf dem Teilbaum $T(n)$ entstanden ist. □

Korollar 6.23 *Satz 6.22 gilt insbesondere, wenn der korrekte Teilbaum $T(n)$ dem korrekten inkrementellen Fuzzy-Entscheidungsbaum entspricht, d.h. $T(n) = T(root)$. Damit ist bewiesen, daß die Funktion* **ensureBestTest** *einen korrekten inkrementellen Fuzzy-Entscheidungsbaum in einen korrekten und optimalen inkrementellen Fuzzy-Entscheidungsbaum umwandelt.*

Nachdem wir alle notwendigen Eigenschaften zur Äquivalenz nachgewiesen haben, geben wir in der folgenden Zusammenfassung das Hauptresultat an.

Kapitel 7

Zusammenfassung und Ausblick

In dieser Arbeit haben wir einen neuen (unseres Wissens den ersten) Algorithmus zum inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen entwickelt. Die Vorteile eines solchen inkrementellen Verfahrens wurden anhand des Projekts ISeC erläutert.

Der entwickelte Algorithmus IIFDT basiert auf den Ideen des Algorithmus ITI (zum inkrementellen Aufbau von klassischen Entscheidungsbäumen [UBC97]) und des Algorithmus IFDT (zum nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen [Zei99, Jan98]). Dafür wurden Bedingungen gefunden, welche die verwendete t-Norm erfüllen muß, damit das Prinzip des inkrementellen Aufbaus auf Fuzzy-Entscheidungsbäume übertragen werden kann.

Der durch den Algorithmus IIFDT induzierte inkrementelle Fuzzy-Entscheidungsbaum erfüllt wegen Definition 5.2 auf Seite 41, und den Sätzen 6.16 auf Seite 63 und 6.22 auf Seite 83 die Eigenschaften Korrektheit und Optimalität. Damit ist er nach Korollar 6.15 auf Seite 62 äquivalent zu dem Fuzzy-Entscheidungsbaum, der durch nicht-inkrementellen Aufbau induziert würde:

Hauptsatz 2 *Die Algorithmen zum inkrementellen und nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen sind äquivalent, d.h. sie erzeugen unter gleichen Voraussetzungen identische Fuzzy-Entscheidungsbäume.*

Wir haben beide Algorithmen im Rahmen des Projekts ISeC implementiert. Für die nächste Projektphase ist ein umfangreicher experimenteller Vergleich der Algorithmen zum inkrementellen und nicht-inkrementellen Aufbau von Fuzzy-Entscheidungsbäumen wünschenswert. Darauf aufbauend sollen Kriterien entwickelt werden, unter welchen einer der beiden Algorithmen vorzuziehen ist. Wir vermuten, daß bei einer großen Lernmenge mit häufigem Eintreffen neuer Beispiele der hier vorgestellte Algorithmus IIFDT deutlich effizienter ist. Dabei ist eine Heuristik zu finden, wie oft **ensureBestTest** durchgeführt werden sollte, um einen Kompromiß zwischen der Qualität des Baumes und dem Aufwand zu seiner Erstellung zu erreichen.

Literaturverzeichnis

[BFOS84] L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone. Classification and regression trees. Wadsworth International Group, CA, 1984.

[Cra89] S.L. Crawford. Extensions to the CART algorithm. *International Journal of Man-Machine Studies*, 31:197–217, 1989.

[Dre01] Technische Universität Dresden. Fakultät Informatik, Institut Künstliche Intelligenz, Wissensverarbeitung. http://www.ki.inf.tu-dresden.de, 2001.

[FKY96] J.H. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. In *AAAI-96*, pages 717–724, 1996. http://robotics.stanford.edu/~ronnyk.

[Gue01] M. Guetova. Fuzzy Lernverfahren. Großer Beleg, TU Dresden, 2001.

[hA01] hybris AG. Hybris webshop. http://www.hybris.de, 2001.

[Höl01] T. Hölldobler. Temporäre Benutzermodellierung für multimediale Produktpräsentationen im World Wide Web. *KI*, 1:22–27, 2001.

[HSMB01] S. Hölldobler, H.-P. Störr, Y. McIntyre, and S.-E. Bornscheuer. User adaption in a web shop system. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference Internet and Multimedia Systems and Applications*, pages 208–213. ACTA Press, 2001.

[Jan98] C.Z. Janikow. Fuzzy decision trees: Issues and methods. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 28, pages 1–14, 1998. http://www.cs.umsl.edu/~janikow/fid/.

[KKG93] R. Kruse, F. Klawonn, and Gebhardt. *Fuzzy-Systeme.* Stuttgart-Teubner, 1993. ISBN 3-519-02130-7.

[LFM95] W.-M. Lippe, Th. Feuring, and L. Mischke. Supervised learning in fuzzy neural networks. Technical report, Bericht Nr. 12/95-I der Angewandten Mathematik und Informatik der Westfälische Wilhelms-Universität Münster, 1995.

[Min89] J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342, March 1989.

[Nau94] D. Nauck. A Fuzzy Perceptron as a Generic Model for Neuro-Fuzzy Approaches. In *Proceedings of the 2nd German GI-Workshop Fuzzy-Systeme'94*, Munich, October 1994.

[NK95] D. Nauck and R. Kruse. NEFCLASS - a Neuro-Fuzzy Approach for Classification of Data. In Ed Deaton Dave Oppenheim K.M. George, Janice H. Carrol and Jim Hightower, editors, *Proceedings of the 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26-28*, pages 461–465. ACM Press, New York, 1995.

[NK96] D. Nauck and R. Kruse. Neuronale Fuzzy-Systeme. *Beitrag zur Herbstschule Konnektionismus und Neuronale Netze 1996, HeKoNN'96*, 1996.

[NK97] D. Nauck and R. Kruse. What are Neuro-Fuzzy Classifiers? In *Proceedings of the Seventh International Fuzzy Systems Association World Congress IFSA'97*, volume III, pages 228–233. Academia, Prague, 1997.

[NKK94] D. Nauck, R. Kruse, and F. Klawonn. *Neuronale Netze und Fuzzy-Systeme, 2., revised and extended edition.* Vieweg Verlag, Wiesbaden, 1994. ISBN 3-528-15265-6.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:86–106, 1986.

[Qui93] J.R. Quinlan. C4.5. In *Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach.* Prentice Hall, 1995.

[SF86] J.C. Schlimmer and D. Fischer. A case study of incremental concept induction. In *Proceedings of the Fifth National Conferemce on Artificial Intelligence*, pages 496–501. Morgan Kaufmann, 1986.

[She97] W.-M. Shen. An active and semi-incremental algorithm for learning decision lists. Technical report, USC-ISI-96-011, Information Sciences Institute, University of Southern California, 1997. http://www.isi.edu/∼shen.

[UBC97] P.E. Utgoff, N.C. Berkmann, and J.A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.

[Utg88] P.E. Utgoff. ID5: An incremental ID3. In *Proceedings of the Fifth International Conferemce on Machine Leaning*, pages 107–120. Morgan Kaufmann, 1988.

[Utg94] P.E. Utgoff. An improved algorithm for incremental induction of decision trees. In *Machine Leaning: Proceedings of the Eleventh International Conferemce*, pages 318–325, New Brunswick, NJ, 1994. Morgan Kaufmann.

[Zei99] J. Zeidler. *Unscharfe Entscheidungsbäume.* PhD thesis, Technische Universität Chemnitz, 1999. http://www.tu-chemnitz/∼jzei/STAFF/JENS/lernen.html.

Index