Diplomarbeit

Effiziente Entdeckung unabhängiger Subgruppen in großen Datenbanken

Dirk Dach

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

Mittwoch , 2. August 2006

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Martin Scholz

Danksagung

Ein herzliches Dankeschön an alle, die mir bei der Erstellung der vorliegenden Arbeit helfend zur Seite gestanden haben. Mein besonderer Dank gilt meinen beiden Betreuern Prof. Dr. Katharina Morik und Dipl.-Inform. Martin Scholz.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Entwicklung der Computertechnik ist soweit fortgeschritten, dass es fast keine Beschränkung der Speicherkapazität mehr gibt. Festplatten mit mehreren 100 Gigabyte Kapazität sind zum Standard geworden. Betrachtet man das Verhältnis von Kosten der Archivierung zu dem potenziellen Nutzen der Informationen in den Daten wird klar, warum Unternehmen verstärkt dazu übergehen, alle Daten zu speichern, die während ihres Geschäftsbetriebes entstehen. Als ein Beispiel betrachte man das Internet. Hier bekommen Online Versandhändler Daten über das Kaufverhalten der Nutzer kostenlos mitgeliefert. Das Wissen in diesen Daten wird auch genutzt, was beim Betrachten eines Produktes offensichtlich wird durch den Hinweis "Kunden, die sich dieses Produkt angesehen haben, interessierten sich auch für jene Produkte". Eine weitere nahezu unerschöpfliche Quelle von Daten sind die mittlerweile überall eingerichteten Scannerkassen. Einkäufe in jedem größeren Geschäft werden erfasst und ermöglichen z.B. das Gewinnen von Informationen über populäre Produktkombinationen. Informationen über das Kaufverhalten der Kunden ermöglichen eine optimale Planung des Produktsortimentes. Ein letztes Beispiel sind die Datenbestände, die bei Telekommunikationsanbietern entstehen. Jede Mobilfunk- oder Festnetzverbindung wird aufgezeichnet. Gerade in Industrieländern entsteht jeden Tag eine Unmenge an Daten. Die darin enthaltenen Informationen können z.B. zur Kapazitätsplanung oder Erstellung von speziellen Angeboten (Wochenendtarif, Sondertarife für häufig gewählte Rufnummern) für bestimmte Kundengruppen genutzt werden.
Damit diese nützlichen Informationen von den Unternehmen genutzt werden können, bedarf es einer Möglichkeit, die Informationen in den Datenbeständen aufzufinden. Insbesondere muss dieses für große Datenbestände effizient möglich sein. Für die in den Daten enthaltenen Informationen wird in der hier verwendeten Terminologie der Begriff Muster benutzt. Der Vorgang der automatischen Mustererkennung in großen Datenbeständen wird als Data Mining bezeichnet. Die Suche nach neuen Data Mining-Methoden ist ein aktives Forschungsgebiet. Die Methoden setzen u. a. Techniken aus dem Maschinellen Lernen und der Statistik ein. Sie führen die Mustererkennung durch und werden wegen der großen Datenmengen von Computern ausgeführt.
Im Rahmen dieser Diplomarbeit wird ein neues Data Mining-Verfahren vorgestellt und untersucht. Bevor in Kapitel 3 das weitere Vorgehen und die Zielsetzungen beschrieben werden, ist es zunächst nötig, die grundlegende Terminologie aus den Bereichen des Data Mining und des Machinellen Lernens einzuführen.

2 Grundlagen

In diesem Kapitel wird zunächst beschrieben, wie man den Begriff des Lernens durch die Definition von Lernaufgaben näher spezifizieren kann. Anhand eines einfachen Beispiels erfolgt die Beschreibung der grundlegenden Terminologie des Maschinellen Lernens und es werden zwei wichtige Lernaufgaben vorgestellt. Hierbei handelt es sich um das Funktionslernen aus Beispielen und die Subgruppenentdeckung.

2.1 Lernaufgaben

Data Mining setzt viele Techniken aus dem Bereich des Maschinellen Lernens ein. Generell ist im Maschinellen Lernen die Frage interessant, was es bedeutet etwas zu verstehen oder zu lernen und inwieweit ein Computer dieses kann. Diese Frage kann vermieden werden, indem die Aufgabe der Mustererkennung in Datenbeständen nicht als Lernen im allgemeinen Sinne bezeichnet wird. Vielmehr wird die Aufgabe viel spezieller anhand der vorhandenen Eingabedaten und der gewünschten Ausgaben definiert. Als typisches Beispiel für die Form von Eingabedaten betrachte man die in Tabelle 2.1 dargestellten Wetterdaten [22].

Sie beschreiben, bei welchen Wetterbedingungen ein nicht näher spezifiziertes Spiel betrieben werden kann. Die Wetterbedingungen heißen *Merkmale* oder *Attribute*. Attribute können *nominal* sein, wie Himmel, Wind und Spielen oder *numerisch*, wie Temperatur und Luftfeuchtigkeit. Ein konkreter Wert eines Attributes heißt *Ausprägung* und jede Zeile ist eine bestimmte Kombination der verschiedenen Ausprägungen der Attribute und wird *Beispiel* oder *Instanz* genannt. Eine besondere Rolle spielt das Attribut 'Spielen', da man daran interessiert ist, zu beschreiben wie dessen Wert von den Ausprägungen der anderen Attribute abhängt. Es wird daher *Zielattribut* oder *abhängiges Merkmal* genannt. Es sollen die in den Daten enthaltene Muster bezüglich des Zielattributes strukturiert und möglichst komprimiert dargestellt werden. In der Terminologie des Maschinellen Lernens spricht man von einem *Konzept*, genauer dem *Zielkonzept*, das von den Daten beschrieben wird; die gefundene Darstellung heißt *Konzeptbeschreibung* oder *Modell*. Eine Möglichkeit der Konzeptbeschreibung ist die Darstellung durch Regeln der Form

WENN (Aussicht=sonnig) DANN (Spielen=Ja) oder
WENN (Aussicht=bewölkt) UND (Temperatur=warm) DANN (Spielen=Ja).

Ein Data Mining Verfahren lernt, wenn es eventuell unter Einhaltung von Nebenbedingungen aus dem gegebenen Eingabedatensatz eine Konzeptbeschreibung der gewünschten Form ableitet. Man spricht in diesem Zusammenhang von einer *Lernaufgabe* [16]. Die beiden Lernaufgaben Konzeptlernen aus Beispielen und Subgruppenentdeckung, die für diese Diplomarbeit im Vordergrund stehen, sollen im folgenden zusammen mit der benötigten Terminologie genauer beschrieben werden.

Himmel	Temperatur	Luftfeuchtigkeit	Wind	Spielen
Sonne	85	85	Nein	Nein
Wolken	83	78	Nein	Ja
Regen	70	96	Nein	Ja
Regen	68	80	Nein	Ja
Regen	65	70	Ja	Nein
Wolken	64	65	Ja	Ja
Sonne	72	95	Nein	Nein
Sonne	69	70	Nein	Ja
Regen	75	80	Nein	Ja
Sonne	75	70	Ja	Ja
Wolken	72	90	Ja	Ja
Wolken	81	75	Nein	Ja
Regen	71	80	Ja	Nein

Tabelle 2.1: Die Wetterdaten

2.2 Konzeptlernen aus Beispielen

Für das Beispiel der Wetterdaten ist das Zielkonzept durch die Werte des Attributes Spielen beschrieben. Definiert ist das Zielkonzept über der *Instanzenmenge* X, die aus allen möglichen Kombinationen der gegebenen Attribute besteht. Obwohl Zielattribute mit mehr als zwei Ausprägungen üblich sind, wird hier nur der boolesche Fall betrachtet. Man spricht bei den beiden Ausprägungen des Zielattributes auch von *positiver und negativer Klasse* bzw. analog von *positiven und negativen Beispielen*. Formal ist das Zielkonzept eine Funktion $f : X \rightarrow \{0, 1\}$. Ein *Lernverfahren* erhält nun *Trainingsbeispiele* $x \in X$ mit bekanntem Zielattribut. Daraus soll das Verfahren eine Beschreibung des Zielkonzeptes ableiten. Wie diese Beschreibung genau aussieht muss durch Wahl einer geeigneten Menge von Konzeptbeschreibungen, dem sogenannten *Hypothesenraum* H, festgelegt werden. Es gibt eine Fülle von Lernverfahren mit zugehörigen Hypothesenräumen, die die Lernaufgabe Konzeptlernen aus Beispielen lösen, wie z.B. regelbasierte Lernverfahren([3, 4, 7]), Entscheidungsbäume und Neuronale Netze [22] oder Support Vector Machines [32]. Die Aufgabe des Lernverfahrens besteht darin eine Hypothese $h \in H$ auszuwählen, so dass für die Trainingsbeispiele möglichst immer f(x)=h(x) gilt. Der Anteil an Beispielen, für den $f(x) \neq h(x)$ gilt, nennt man *Trainingsfehler*.

Definition 1. *(Trainingsfehler)*
Sei T eine Trainingsmenge der Größe n von Instanzen aus X, f die zugehörige Zielfunktion und H der gegebene Hypothesenraum. Der Trainingsfehler einer Hypothese $h \in H$ bezüglich T und f ist definiert als der durchschnittliche Fehler über alle Instanzen der Trainingsmenge:

$$error_T(h) = \frac{1}{n} \sum_{i=1}^{n} error(f(x), h(x)).$$

Hierbei ist der Fehler für eine einzelne Instanz definiert als:

$$error(f(x), h(x)) = 0,\ falls\ h(x) = f(x),\ sonst\ 0\ (0 - 1 - loss).$$

Da die gefundene Hypothese Instanzen in negative und positive Beispiele klassifizieren soll, bezeichnet man sie im Rahmen dieser Lernaufgabe auch als *Klassifikator*. Für die Art der Vorhersage, die der gelernte Klassifikator machen soll, gibt es zwei Möglichkeiten: harte (crisp) und weiche (soft) Vorhersagen. Beispielsweise kann es sein, dass eine bestimmte Regel für acht positive und zwei negative Beispiele anwendbar ist. Bei harter Klassifizierung würde in diesem Fall die positive Klasse vorhergesagt, da diese die Mehrheitsklasse ist. Bei weicher Klassifizierung würde eine Wahrscheinlichkeit von 0.8 für die positive und 0.2 für die negative Klasse vorhergesagt.
Für den Fall einer weichen Klassifizierung existiert eine Alternative zur Messung des Trainingsfehler. Der *Root Mean Squared Error (RMSE)* misst die durchschnittliche Abweichung der Vorhersagen einer Hypothese von den tatsächlichen Ausprägungen des Zielattributes.

Definition 2. *(Root Mean Squared Error)*
Seien T, X, f und H wie in obiger Definition. Der Root Mean Squared Error (RMSE) einer Hypothese h ist definiert als Quadratwurzel aus dem mittleren quadrierten Fehler:

$$RMSE_T(h) := \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x) - h(x))^2}.$$

Die alleinige Minimierung des Trainingsfehlers ist aber i.d.R. nicht ausreichend. Als Beispiel betrachte man eine Bank, die aus ihren Kundendaten das Konzept 'Kreditwürdigkeit' ableiten möchte. Es soll eine Hypothese gefunden werden, die für jeden Kunden aus seinen spezifischen Daten bestimmt, ob er einen Kredit erhält. Dazu werden als Trainingsbeispiele solche Kunden verwendet, die schon einen Kredit haben, deren Kreditwürdigkeit also schon bekannt ist. Diese Trainingsmenge von Kunden kann als Stichprobe aus der Menge aller potentiellen Kunden angesehen werden, für die der Trainingsfehler bestimmt werden kann. Der wirkliche Nutzen der Bank ergibt sich erst, wenn sie für Neukunden bei denen die Kreditwürdigkeit unbekannt, ist eine zuverlässige Vorhersage über deren Kreditwürdigkeit machen kann. Damit es überhaupt möglich ist mit der aus den Trainingsinstanzen gelernten Hypothese eine Aussage über zukünftige Instanzen zu machen ist die Annahme nötig, dass eine gute Approximation der Zielfunktion f für die Trainingsmenge auch eine gute Approximation von f für ungesehene Instanzen ist. Formal nimmt man an, dass alle Instanzen gemäß einer Wahrscheinlichkeitsverteilung D gezogen werden. Den erwarteten Fehler, den die gefundene Hypothese für Instanzen macht, die gemäß D gezogen wurden, bezeichnet man als wahren Fehler.

Definition 3. *(Wahrer Fehler)*
Sei $D : X \rightarrow \mathbb{R}^+$ eine Wahrscheinlichkeitsverteilung über dem Instanzenraum X, f die zugehörige Zielfunktion und H der gegebene Hypothesenraum. Der wahre Fehler einer Hypothese $h \in H$ bezüglich Instanzen, die gemäß D zufällig gezogen werden und f ist definiert als

$$error_D(h) = \int_{x \in X} error(f(x), h(x))\ D(x)\ dx,$$

falls der Fehler für eine einzelne Instanz mit dem 0-1-loss gemessen wird bzw.

$$error_D(h) = \sqrt{\int_{x \in X} (f(x) - h(x))^2\ D(x)\ dx},$$

falls der Fehler für eine einzelne Instanz der quadratische Fehler ist.

Den Vorgang bei dem eine Hypothese ungesehene Instanzen (richtig) klassifiziert, bezeichnet man auch als *Generalisierung*.
Nun lässt sich die Lernaufgabe Funktionslernen aus Beispielen definieren.

Definition 4. *(Funktionlernen aus Beispielen)*
Die Aufgabe beim Funktionslernen aus Beispielen besteht darin, für eine gemäß der Verteilung $D : X \to \mathbb{R}^+$ gezogene Trainingsmenge von Instanzen und einen gegebenen Hypothesenraum H, eine Hypothese h^* zu finden, die den wahren Fehler bezüglich H und D minimiert:

$$h^* = argmin_{h \in H} \{error_D(h)\}.$$

Der wahre Fehler kann nicht direkt bestimmt werden, da Zielfunktion f und die Verteilung D unbekannt sind. Der Trainingsfehler ist im allgemeinen eine schlechte Abschätzung des wahren Fehlers; er unterschätzt ihn meistens. Dieses kann mehrere Ursachen haben. Zum einen kann die Trainingsmenge fehlerhafte Beispiele enthalten, die der wahren Funktion f widersprechen. Dann klassifiziert eine Hypothese, die auf der Trainingsmenge fehlerfrei ist, neue Beispiele schlecht. Gleiches gilt, wenn der Hypothesenraum zu komplex gewählt wurde, so dass einfach nur die Trainingsmenge auswendig gelernt wurde anstatt die darin ‚versteckte' Zielfunktion f zu approximieren. Man bezeichnet dieses mit *Überanpassung (Overfitting)* an die Trainingsmenge. Verfahren, die die Lernaufgabe Konzeptlernen aus Beispielen lösen, beinhalten daher Mechanismen, die versuchen eine Überanpassung zu verhindern.

Da der Trainingsfehler häufig keine gute Schätzung für den wahren Fehler ist, wird in der Regel ein Teil der Trainingsmenge als Testdatensatz benutzt. Der auf den übrigen Beispielen der Trainingsmenge gelernte Klassifikator wird benutzt, um Vorhersagen für die ungesehenen Testbeispiele zu treffen. Den Fehler auf diesen Testdaten benutzt man zur Schätzung des wahren Fehlers. Eine wichtige Variation dieses Verfahrens ist die Kreuzvalidierung. Hierbei wird die Trainingsmenge in mehrere disjunkte Teilmengen zerlegt; eine übliche Wahl sind zehn Teilmengen. Eine Teilmenge wird als Testdatensatz zurückgehalten, mit dem Rest wird gelernt. Dieses wird wiederholt bis alle Teilmengen einmal Testdatensatz waren. Bei der 10-fachen Kreuzvalidierung wird beispielsweise zehn mal mit 90% der Daten gelernt und auf einer immer anderen Teilmenge von 10% der Daten getestet. Zur Schätzung des wahren Fehlers wird der Trainingsfehler über alle zehn Testmengen gemittelt. In der Praxis wird die 10-fache Kreuzvalidierung häufig eingesetzt, da sie meist eine gute Schätzung für den wahren Fehler liefert.

2.3 Subgruppenentdeckung

Beim Konzeptlernen aus Beispielen handelt es sich um eine prädiktive Lernaufgabe, d.h. man ist daran interessiert ein globales Modell zu finden, das für jede zukünftige Instanz eine Vorhersage machen kann. Bei der Subgruppenentdeckung handelt es sich dagegen um eine deskriptive Lernaufgabe. Die gefundenen Konzepte sind lokale Aussagen über interessante Teilbereiche der Trainingsmenge. Als ein Beispiel betrachte man wieder die Kundendaten einer Bank und einen Investmentfond, den die Bank anbietet. Aus Marketinggesichtspunkten ist die Bank an Kundengruppen interessiert, bei denen der Fond unterrepräsentiert ist. Aus den Daten der Bank lässt sich leicht ein binäres Merkmal konstruieren, das für jeden Kunde angibt, ob er in den Fond investiert hat. Die Aussage, dass Rentner wesentlich seltener in den Investmentfond anlegen als die gesamte Kundschaft, kann für die Bank wertvoll sein und das Management zu dem Versuch bewegen, die gefundene Subgruppe der Rentner durch besondere Maßnahmen von den Vorteilen des Investmentsfond zu überzeugen. Wie man an diesem Beispiel sieht, ist auf diese Weise keine globale Vorhersage möglich: Man kann zwar aussagen, dass die Verteilung des interessierenden Merkmals bei Instanzen in der gefundenen Subgruppe von der Verteilung in der gesamtem Trainingsmenge abweicht, aber keine Aussage über Instanzen außerhalb der Subgruppe machen. Die Lernaufgabe Subgruppenentdeckung bedarf der Festlegung des interessierenden Merkmals des Hypothesenraumes H und einer *Nutzenfunktion* q, die das Maß für die Interessantheit einer Subgruppe ist. Fasst man das Zielattribut als interessierendes Merkmal auf, so benötigt man wie schon beim Konzeptlernen aus Beispielen zur Subgruppenentdeckung ebenfalls eine Trainingsmenge von bereits klassifizierten Instanzen. Desweiteren muss für jede durch ein Hypothese definierte Subgruppe bestimmbar sein, welche Instanzen zu ihr gehören. Damit lässt sich die Lernaufgabe Subgruppenentdeckung definieren.

Definition 5. *(Subgruppenentdeckung)*
Es sei T eine Trainingsmenge von Instanzen aus dem Instanzenraum X und $\mathcal{P}(X)$ die Potenzmenge von X. Weiterhin sei H ein Hypothesenraum, in dem jede Hypothese mit einer Teilmenge des Instanzenraumes identifiziert werden kann.
Gegeben eine Nutzenfunktion q: $H \times \mathcal{P}(X) \rightarrow \mathbb{R}$ besteht die Lernaufgabe Subgruppenentdeckung darin, die Menge von Hypothesen $h \in H$ mit dem höchsten Nutzen bezüglich q zu finden.

Die Anzahl der gefundenen Hypothesen kann entweder durch einen minimalen Wert für den Nutzen einer Hypothese h beschränkt werden oder es wird festgelegt, dass nur die k besten Hypothesen zur Lösung gehören sollen.
Obwohl die Lernaufgabe Subgruppentdeckung oft als deskriptiv angesehen wird, ist auch eine Definition als prädiktive Lernaufgabe denkbar. Dazu muss angenommen werden, dass die Trainingsmenge T eine Stichprobe aus dem Instanzenraum X ist, die gemäß einer Wahrscheinlichkeitsverteilung D über X gezogen wurde. Desweiteren bedarf es einer Veränderung der Nutzenfunktion q. Es werden nicht mehr die Subgruppen gesucht, die bezüglich q auf den Trainingsdaten optimal sind. Stattdessen sucht man nach Subgruppen, die optimal sind für Beispielmengen die unter der Verteilung D gezogen wurden. Die gefundenen Subgruppen treffen dann Aussagen über den gesamten Instanzenraum und nicht nur über die Trainingsmenge.
Der wichtigste Parameter der Lernaufgabe Subgruppenentdeckung ist die Nutzenfunktion q. Unterschiedliche Arten von Nutzenfunktionen erlauben eine Veränderung dessen, was als interessante Subgruppen angesehen wird. Den verschiedenen gebräuchlichen Nutzenfunktionen und ihren Eigenschaften ist das folgende Kapitel gewidmet.

2.4 Nutzenfunktionen

Für die Subgruppenentdeckung und dem Konzeptlernen aus Beispielen gibt es eine Reihe von Nutzenfunktionen. Sie sind komplementär zu den im vorherigen Kapitel definierten Verlustfunktionen. Daher werden Verlustfunktionen auch minimiert, während Nutzenfunktionen maximiert werden. Für eine ausführliche Diskussion von Nutzenfunktionen verweise ich auf [14] und [19]. Analog zu wahrem Fehler und Trainingsfehler gibt es für Nutzenfunktionen einen auf den Trainingsdaten geschätzten Nutzen und einen wahren Nutzen. Auch hier kann die Kreuzvalidierung zur Schätzung des wahren Wertes benutzt werden. Wenn im Folgenden der Unterschied zwischen wahrem und geschätztem Nutzen wichtig ist, wird besonders darauf hingewiesen, welcher Wert gemeint ist. Damit eine präzise Definition von Nutzenfunktionen möglich ist bedarf es einer genaueren Festlegung des Hypothesenraumes. In dieser Arbeit werden als Hypothesen nur Regeln, genauer Hornklauseln, verwendet. Hornklauseln sind eine Teilmenge der Formeln der Prädikatenlogik, wobei es hier genügt, sich auf den aussagenlogischen Fall zu beschränken. Sie bestehen aus einer Disjunktion von Literalen, von denen maximal eines positiv ist:

$$\neg\mathcal{P}_1 \vee \neg\mathcal{P}_2 \vee \cdots \vee \neg\mathcal{P}_N \vee \mathcal{K}.$$

Dieses lässt sich zu einer Implikation umformen:

$$\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_N \Rightarrow \mathcal{K}.$$

Bei den benutzten Literalen handelt es sich um Attribut-Wert-Paare, die wahr werden, wenn das Attribut im Literal und das entsprechende im Beispiel die gleiche Ausprägung haben.

Definition 6. *(Hornklauseln)*
Eine Hornklausel besteht aus einer Prämisse A und einer Konklusion B, dargestellt als $A \rightarrow B$. Die Prämisse A besteht aus einer Konjunktion von Literalen über den Attributen des Instanzenraumes, während die Konklusion B einen Wert für das Zielattribut vorhersagt. Eine Hornklausel ist anwendbar, falls die Prämisse wahr ist. Ist die Konklusion ebenfalls wahr, d.h. hat das Zielattribut in der Konklusion und im betrachteten Beispiel die gleiche Ausprägung, ist die Hornklausel korrekt.
Im Folgenden wird statt Hornklausel der kürzere und übliche Begriff Regel verwendet.

Eine Regel R erzeugt eine Partitionierung aller Beispiele in zwei Teilmengen:

$$r = \{x \in X \mid R\ ist\ auf\ x\ anwendbar\},\ \bar{r} = X \backslash h.$$

Die Beispiele in der Menge r werden auch als die von der Regel R abgedeckten Beispiele bezeichnet Da sich diese Diplomarbeit auf den Fall eines booleschen Zielattributes beschränkt, werden folgende Abkürzungen für die Menge aller positiven bzw. negativen Beispiele verwendet:

$$Y_+ = \{x \in X \mid x\ ist\ ein\ positives\ Beispiel\}\ bzw.\ Y_- = X \backslash Y_+.$$

Damit ergeben sich die Schreibweisen $r \rightarrow Y_+$ bzw. $r \rightarrow Y_-$ für eine Regel, die die positive bzw. negative Klasse vorhersagt. Eine Regel kann identifiziert werden mit der Menge der Beispiele, für die sie anwendbar ist und der Klasse, die sie vorhersagt. Somit beschreibt eine Regel eindeutig eine Subgruppe: Das interessierende Merkmal wird durch die Vorhersage bestimmt und die Subgruppe besteht aus allen Instanzen, auf die sie anwendbar ist. Da es für alle Nutzenfunktionen Sinn macht, sowohl nach dem Nutzen bezogen auf die positive als auch bezogen auf die negative Klasse zu fragen und die Wahl der positiven Klasse relativ willkürlich ist, wird in den Definitionen der Nutzenfunktionen die interessierende Klasse mit Y_* bezeichnet. Desweiteren werden Regeln oft um die Angabe der Wahrscheinlichkeit der vorhergesagten Klasse in der Menge r der abgedeckten Beispiele erweitert, so dass sie genauer als probabilistische Regeln bezeichnet werden können. Die Regel

$$r \rightarrow Y_+ \ [90\%].$$

besagt, dass die Wahrscheinlichkeit für ein positives Beispiel 90% beträgt, falls die Regel anwendbar ist.

2.4.1 Coverage, Precision, Bias und Lift

Die grundlegenden Konzepte zur Bewertung von Hypothesen und im speziellen von Regeln sind Generalität und Konfidenz. Die Generalität ist ein Maß dafür wie allgemein eine Regel ist bzw. für wieviele Instanzen sie anwendbar ist, während man mit Konfidenz misst, wie präzise eine anwendbare Regel ist. Bezeichnet man mit $Pr_D[W]$ die Wahrscheinlichkeit, ein Beispiel aus der Teilmenge $W \in X$ von Instanzen unter der Verteilung D zu ziehen, lassen sich konkrete Maße für Generalität und Konfidenz definieren, wobei der Index D weggelassen wird, falls die Verteilung eindeutig ist.

Definition 7. *(Coverage)*
Die Coverage einer Regel $r \rightarrow Y_*$ gibt die Wahrscheinlichkeit an, dass eine Regel anwendbar ist.

$$COV(r \rightarrow Y_*) := Pr[r].$$

Definition 8. *(Precision)*
Die Precison einer Regel $r \rightarrow Y_*$ gibt die Wahrscheinlichkeit an, dass eine Regel korrekt ist, falls sie anwendbar ist.

$$PREC(r \rightarrow Y_*) := Pr[Y_*|r].$$

Betrachtet man die Teilmenge der Instanzen, für die eine Regel anwendbar ist, so wird dort im Allgemeinen die Verteilung des Zielattributes (der Anteil von Beispielen mit der interessierenden Klasse) anders sein als im gesamten Instanzenraum. Ansonsten ist die von der Regel beschriebene Subgruppe uninteressant. Der *Bias* ist ein Maß für diese Abweichung.

Definition 9. *(Bias)*
Der Bias einer Regel $r \rightarrow Y_*$ ist definiert als

$$BIAS(r \rightarrow Y_*) := Pr[Y_*|r] - Pr[Y_*] = PREC(r \rightarrow Y_*) - Pr[Y_*].$$

Das multiplikative Gegenstück zum Bias ist der *Lift*.

Definition 10. *(Lift)*
Der Lift einer Regel $r \rightarrow Y_*$ ist definiert als

$$LIFT(r \rightarrow Y_*) := \frac{Pr[r \cap Y_*]}{Pr[r]Pr[Y_*]} = \frac{Pr[Y_*|r]}{Pr[Y_*]} = \frac{PREC(r \rightarrow Y_*)}{Pr[Y_*]}.$$

Hierbei wurden die Definitionen von bedingten Wahrscheinlichkeiten und Precision eingesetzt.
Sofern sich die Wahrscheinlichkeit, ein Beispiel mit der interessierenden Klasse zu sehen, in der durch r bestimmten Subgruppe nicht von der Wahrscheinlichkeit für die interessierende Klasse im gesamten Instanzenraum, der *apriori Wahrscheinlichkeit*, unterscheidet, ergibt sich entsprechend dem multiplikativen bzw. additiven Charakter von Lift bzw. Bias, dass $LIFT(r \to Y_*) = 1$ und $BIAS(r \to Y_*) = 0$. Ist die Wahrscheinlichkeit der interessierenden Klasse größer als im gesamten Instanzenraum, gilt $LIFT(r \to Y_*) > 1$ und $BIAS(r \to Y_*) > 0$, während $0 \leq LIFT(r \to Y_*) < 1$ und $BIAS(r \to Y_*) < 0$, falls die Wahrscheinlichkeit der interessierende Klasse in der Subgruppe unterdurchschnittlich ist.

2.4.2 Accuracy und Weighted Relative Accuracy

Eine beim Konzeptlernen aus Beispielen häufig verwendete Nutzenfunktion ist die *Accuracy*.

Definition 11. *(Accuracy)*
Die Accuracy einer Regel $r \to Y_*$ ist definiert als

$$ACC(r \to Y_*) := Pr[r \cap Y_*] + Pr[\bar{r} \cap \bar{Y}_*].$$

Eine Regel hat eine hohe Accuracy, wenn sie für viele Beispiele der interessierenden Klasse anwendbar und für alle anderen Beispiele möglichst nicht anwendbar ist. Das gebräuchlichste Maß für die Lernaufgabe Subgruppenentdeckung ist die *Weighted Relative Accuracy* [19]. Interessantheit wird als Produkt von Coverage und Bias definiert. In der einfachsten Version werden beide Größen gleich stark gewichtet.

Definition 12. *(Weighted Relative Accuracy)*
Die Weighted Relative Accuracy einer Regel ist das Produkt aus deren Coverage und Bias:

$$WRACC(r \to Y_*) := COV(r \to Y_*) \cdot BIAS(r \to Y_*).$$

Eine alternative Definition von Interessantheit einer Subgruppe erhält man durch quadrieren der Coverage [19].

Definition 13. *(Squared)*

$$SQUARED(r \to Y_*) := COV(r \to Y_*)^2 \cdot BIAS(r \to Y_*)$$

2.4.3 Nutzenfunktion auf Basis des Binomialtests

Bei der Lernaufgabe Subgruppenentdeckung sind solche Subgruppen interessant, bei denen die Precision der korrespondierenden Regel $r \to Y_*$ von der apriori Wahrscheinlichkeit $Pr[Y_*]$ abweicht. Ob die empirisch beobachtete Differenz signifikant oder zufällig ist, lässt sich mit Hilfe des Binomialtests[9] überprüfen. Um festzustellen, ob die Subgruppe wirklich interessant ist, formuliert man eine Nullhypothese H_0 und eine Alternative H_1 über den wahren Bias der Subgruppe:

$$H_0 : BIAS(r \to Y_*) = 0 \text{ und } H_1 : BIAS(r \to Y_*) \neq 0.$$

Als Prüfgröße zur Entscheidung über das Ablehnen der Nullhypothese zugunsten der Alternative dient die in der Subgruppe beobachtete Anzahl von Beispiele der interessierenden Klasse. Dazu wird zunächst eine dichotome Zufallsvariable X_i definiert.

$$X_i = \begin{cases} 1, & falls\ x\ ein\ Beispiel\ f\ddot{u}r\ die\ interessierende\ Klasse\ in\ der\ Subgruppe\ ist \\ 0, & sonst \end{cases}.$$

Dann ergibt sich die Prüfgröße für m Beispiele als $X = \sum_{i=1}^{m} X_i$. Man nimmt an, dass die Nullhypothese H_0 gilt und überprüft, wie groß die Werte für X werden müssen, damit es extrem unwahrscheinlich ist, dass sie unter H_0 zustande gekommen sind. Dazu muss festgelegt werden, was unter extrem unwahrscheinlich zu verstehen ist. Übliche Werte sind Wahrscheinlichkeiten von 0.1, 0.05 oder 0.01, die auch als Signifikanzniveau bezeichnet werden. Da die Beispiele unabhängig und gleichverteilt gezogen werden ist X unter Annahme von H_0 binomialverteilt mit den Parametern $P[Y_*]$ und m. Nun kann man anhand einer Tabelle der Binomialverteilung bestimmen, welche Werte für die Prüfgröße X unwahrscheinlicher als das gewählte Signifikanzniveau sind und erhält die Werte für X, bei denen die Nullhypothese zugunsten der Alternative zu verwerfen ist. Für große m ist X annähernd normalverteilt mit Erwartungswert $m \cdot Pr[Y_*]$ und Standardabweichung $\sqrt{m \cdot Pr[Y_*](1 - Pr[Y_*])}$. Damit erhält man folgende standardnormalverteilte Prüfgröße:

$$Z = \frac{X - m \cdot Pr[Y_*]}{\sqrt{m \cdot Pr[Y_*](1 - Pr[Y_*])}}$$

Analog zur Binomialverteilung wird mit der Tabelle der Standardnormalverteilung bestimmt, für welche Werte der Prüfgröße Z die Nullhypothese bei gegebenem Signifikanzniveau zu verwerfen ist. Bei Vernachlässigung des Signifikanzniveaus ist es umso wahrscheinlicher, dass die Alternative gilt, je größer der Z-Wert ist. Der Z-Wert lässt sich folgendermaßen umformen:

$$Z = \frac{X - m \cdot Pr[Y_*]}{\sqrt{m \cdot Pr[Y_*](1 - Pr[Y_*])}} = \frac{m(\frac{1}{m}X - Pr[Y_*])}{\sqrt{m} \cdot \sqrt{Pr[Y_*](1 - Pr[Y_*])}} = \frac{\sqrt{m} \cdot BIAS(r \to Y_*)}{\sqrt{Pr[Y_*](1 - Pr[Y_*])}}.$$

Da der Term $\sqrt{Pr[Y_*](1 - Pr[Y_*])}$ im Nenner für alle Regeln gleich ist, erzeugt $\sqrt{m} \cdot BIAS(r \to Y_*)$ eine Ordnung der Regeln entsprechend der Signifikanz der Abweichung ihres Bias von der apriori Wahrscheinlichkeit. Durch den Wechsel von der absoluten Anzahl von Beispielen auf den relativen Anteil wird diese Ordnung nicht verändert. Es ergibt sich die folgende Definition.

Definition 14. *(Binomial)*
Die Nutzenfunktion auf Basis des Binomialtests ist definiert als

$$BINOMIAL(r \to Y_*) := \sqrt{COV(h \to Y_*)} \cdot BIAS(r \to Y_*).$$

3 Ziele der Diplomarbeit

Beim Generic Sequential Sampling [25] Algorithmus handelt es sich um ein Verfahren zur Lösung der Lernaufgabe Subgruppenentdeckung, das erlaubt als Kriterium der Interessantheit von Subgruppen, eine der in Kapitel 2.4 vorgestellten Nutzenfunktionen Accuracy, Weighted Relative Accuracy, Squared oder Binomial auszuwählen. Zur Bestimmung der besten Subgruppen wird nicht der komplette Datensatz durchsucht, sondern mit einer Stichprobe gearbeitet. Der Algorithmus gibt probabilistische Garantien für die Qualität der Lösung. Der Vorteil dieses Ansatzes liegt in der guten Skalierbarkeit für große Datenbanken. Allerdings muss auch eine Reihe von Nachteilen in Kauf genommen werden, von denen zwei nachfolgend angesprochen werden sollen.
Das Verfahren bedarf der Aufzählung des kompletten Hypothesenraumes, wodurch es unmöglich ist mit numerischen Attributen umzugehen. Vor der Anwendung ist zwingend eine Diskretisierung numerischer Attribute nötig. Auch nach einer Diskretisierung ist der komplette Hypothesenraum meist so komplex, dass es nicht möglich ist, ihn effizient zu handhaben. Ein weiterer Nachteil ergibt sich durch Korrelationen zwischen den Attributen eines Datensatzes. Hypothesen wie z.B. Regeln beschreiben Subgruppen mit Hilfe der Attribute. Aufgrund der Korrelationen ergeben sich mehrere Hypothesen, die die gleiche Subgruppe beschreiben. Dadurch ist die gefundene Lösung unnötig groß und unübersichtlich. Für den Nutzer ist sie schwer zu interpretieren, da die wichtigen Aussagen aus der Vielzahl redundanter Lösungen nicht klar hervorstechen.
Mit Knowledge-Based Sampling [26] steht eine Methode zur Verfügung, um das Finden von redundanten Hypothesen zu vermeiden. Eine Hypothese in Form einer Regel repräsentiert durch ihre Vorhersagen Vorwissen über das Zielattribut. Knowledge-Based Sampling entfernt dieses Vorwissen aus den Daten, indem es die Korrelation zwischen Vorhersagen der Regel und den tatsächlichen Werten des Zielattributes aus den Daten entfernt. Danach sind weder diese noch andere Regeln, von denen die gleiche Subgruppe beschrieben wird, in den Daten zu finden.
Vorrangiges Ziel dieser Diplomarbeit ist die Untersuchung der Kombination des Generic Sequential Sampling Algorithmus mit Knowledge-Based Sampling bezüglich der Eignung für die Lernaufgaben Subgruppenentdeckung und Konzeptlernen aus Beispielen. Die Kombination der beiden Verfahren verspricht eine präzise und kompakte Beschreibung der Trainingsdaten. Insbesondere sollte ein effizienter Umgang mit sehr großen Datenmengen möglich sein, da statt des gesamten Datensatzes nur eine Stichprobe verarbeitet wird.
Zunächst werden in Kapitel 4.1 der Generic Sequential Sampling Algorithmus und in Kapitel 4.2 das Knowledge-Based Sampling vorgestellt. Um für die Lernaufgabe Konzeptlernen aus Beispielen Vorhersagen über das Zielattribut zu treffen, wird eine Kombination mehrerer Regeln benutzt, die mit dem GSS Algorithmus unter Einsatz von Knowledge-Based Sampling gefunden wurden. Ein Verfahren zur Kombination mehrerer unter dem Einsatz von Knowledge-Based Sampling gefunderer Modelle ist in Kapitel 4.2.4 beschrieben. Um dem Generic Sequential Sampling Algorithmus zu ermöglichen, mit numerischen Attributen umzugehen, bedarf es einer Methode zur Diskretisierung, wie sie

in Kapitel 4.3 vorgestellt ist. In Kapitel 5 wird mit dem Iterating Generic Sequential Sampling Algorithmus ein Verfahren zur Kombination des Generic Sequential Sampling Algorithmus mit Knowledge-Based Sampling vorgestellt. Insbesondere ist in diesem Kapitel beschrieben, wie eine effiziente Suche in großen Hypothesenräumen möglich ist. Es werden mehrere Nachteile des Generic Sequential Sampling Algorithmus beschrieben und mögliche Lösungen aufgezeigt. Außerdem werden eine Reihe von Erweiterungen vorgestellt. In Kapitel 6 wurden mit dem Iterating Generic Sequential Sampling Algorithmus Experimente mit synthetischen und echten Datensätzen durchgeführt. Zur Erzeugung der synthetischen Datensätze bedurfte es der Erstellung eines geeigneten Werkzeuges. Die Experimente dienten dabei dem Zweck, die folgenden Fragen zu klären:

- Wie eignet sich das Verfahren zur Subgruppenentdeckung?
- Wie verändern sich der Umfang und die Aussagekraft der Lösung durch den Einsatz von Knowledge-Based Sampling?
- Wie ist die Vorhersagequalität der Kombination der einzelnen Regeln zu einem Gesamtmodell?
- Wie gut ist die Vorhersagequalität im Vergleich zu anderen Methoden für die Lösung der Lernaufgabe Konzeptlernen aus Beispielen?
- Wie ist die Laufzeit im Vergleich zu anderen Methoden?
- Wie wirkt sich die Veränderung der Nutzenfunktion auf Qualität und Laufzeit aus?
- Welche Auswirkungen hat die Verwendung von Hypothesenräumen unterschiedlicher Komplexität auf Laufzeit und Qualität der Ergebnisse?

4 Verwendete Techniken und Algorithmen

Ziel der Diplomarbeit ist die Untersuchung der Eignung der Kombination des Generic Sequential Sampling Algorithmus mit Knowledge-Based Sampling für Subgruppenentdeckung und Konzeptlernen aus Beispielen. Im Folgenden werden die verwendeten Verfahren näher erläutert sowie auf deren Vor- und Nachteile eingegangen. Die Subgruppenentdeckung wird mit dem Generic Sequential Sampling Algorithmus [25] durchgeführt (Kapitel 4.1). Knowledge-Based Sampling [26] stellt die Unabhängigkeit der gefundenen Subgruppen sicher und ermöglicht die Kombination der gefundenen Subgruppen, um eine Vorhersage für das Zielattribut zu treffen (Kapitel 4.2). Um den Umgang mit numerischen Attributen zu ermöglichen, bedarf es schließlich noch einer Diskretisierung numerischer Attribute. Ein geeignetes Verfahren ist das *Recursive Minimal Entropy Partitioning* [10], das in Kapitel 4.3 vorgestellt wird. Für eine Übersicht der verwendeten Notationen verweise ich auf Anhang A.

4.1 Der Generic Sequential Sampling Algorithmus

Wird Data Mining auf sehr großen Datenbanken betrieben, ist neben der Maximierung der Nutzenfunktion auch die Skalierbarkeit von Bedeutung. Für viele Anwendungen ist es nicht möglich, in akzetabler Zeit alle Daten zu verarbeiten. Eine mögliche Strategie, um mit diesem Problem umzugehen, ist mit einer zufällig gezogenen Stichprobe der Daten zu arbeiten. Natürlich bedeutet dieses neben der Laufzeitverkürzung und Verkleinerung des benötigten Speicherplatzes auch, dass nicht mehr garantiert werden kann, dass die gewonnenen Resultate identisch sind mit denen für die gesamten Daten. Von Bedeutung ist hierbei der Unterschied im Nutzen. Es ist wichtig, dass ein Lernverfahren, das nur auf einer Stichprobe der Daten arbeitet, dem Nutzer Garantien gibt, wie stark sich die Resultate bezüglich des Nutzens unterscheiden. Es gibt zwei Möglichkeiten, wie diese Garantien aussehen können. Zum einen kann für eine feste Stichprobengröße berechnet werden, wie gut die Garantie für den Nutzen auf den gesamten Daten ist [12], zum anderen kann vom Benutzer eine feste Qualität der Lösung vorgegeben werden. Das Ziel ist, mit minimal möglicher Stichprobengröße die geforderte Qualität zu garantieren. Letztgenannter Ansatz wird beim sequentiellen Sampling verfolgt, das im Bereich des Maschinellen Lernens erstmals im Rahmen des Hoeffding Race Algorithmus [18] benutzt wurde. Hierbei werden die Instanzen bzw. die Stichprobe inkrementell gezogen und der Nutzen aller Hypothesen des betrachteten Hypothesenraumes gleichzeitig aktualisiert. Wenn sicher ist, dass eine Hypothese sehr gut bzw. schlecht ist, gibt der Algorithmus diese Hypothese als Lösung aus bzw. verwirft sie. Ein wichtiger Vorteil ist, dass die Beschaffenheit der Daten berücksichtigt wird: Erlauben die Daten die Hypothesen schnell in Gut und Schlecht zu separieren, werden nur wenige Beispiele in der Stichprobe benötigt, ansonsten wird die Stichprobe größer. Der Generic Sequential Sampling (GSS) Algorithmus [25] benutzt ebenfalls den Ansatz des sequentiellen Samplings. Der Nutzer gibt dabei vor, wie gut die gefundene Lösung sein soll und wie groß die Irrtumswahrscheinlichkeit sein darf. Außerdem kann unter verschiedenen Nutzenfunktionen gewählt werden, so dass sich der Algorithmus prinzipiell sowohl für die Lernaufgabe Konzeptlernen aus Beispielen als auch für die Subgruppenentdeckung eignet und für Letztere unterschiedliche Definitionen von Interessantheit zulässt. In den nächsten Abschnitten werden die formalen Grundlagen und der GSS Algorithmus erläutert sowie Schranken für die Qualität und benötigte Beispielanzahl einiger populärer Nutzenfunktionen angegeben.

4.1.1 Grundlagen

Das Hauptanwendungsgebiet des GSS Algorithmus besteht in der Subgruppenentdeckung. Der Hypothesenraum kann beliebig sein, solange es möglich ist, ihn geordnet aufzuzählen. Ein Beispiel für einen geordneten Hypothesenraum sind die bereits angesprochenen Regeln aus konjunktiv verknüpften Literalen. Die Menge der gefundenen Hypothesen in der Ausgabe des Algorithmus wird durch einen vom Nutzer festgelegten Parameter k bestimmt. In der benutzten Terminologie spricht man vom k-beste Hypothesen Problem. Da die k besten Hypothesen bzw. ihr Nutzen auf Basis einer Stichprobe bestimmt werden, ist nicht garantiert, dass sie auch für die gesamten Trainingsmenge T optimal sind. Man definiert für diesen Fall das approximativ k-beste Hypothesen Problem.

Definition 15. *(Das approximativ k-beste Hypothesen Problem)*
Gegeben sind eine Trainingsmenge T von Instanzen aus dem Instanzenraum X, eine Nutzenfunktion q und ein Hypothesenraum H. Desweiteren seien k die gesuchte Anzahl an Lösungen, $\epsilon \in \mathbb{R}^+$ der maximal zulässige Fehler und δ, $0 < \delta \leq 1$ die gewünschte Irrtumswahrscheinlichkeit. Das approximativ k-beste Hypothesen Problem besteht darin eine Menge $G \subseteq H$ der Größe k zu finden, so dass es mit Konfidenz $1-\delta$ keine Hypothese $h' \in H$ gibt, für die gilt:

$$h' \notin G \text{ und } q(h',T) > q_{min} + \epsilon.$$

Hierbei bezeichnet q_{min} den Nutzen der bezüglich q schlechtesten der k Hypothesen in der Menge G:

$$q_{min} := min_{h \in G} q(h,T).$$

Zu beachten ist, dass das Problem bezüglich der Trainingsmenge T und nicht bezüglich des Instanzenraumes X definiert ist. Da immer mit einem statistischen Verfahren ein unbekannter wahrer Nutzen einer Hypothese aufgrund einer Stichprobe geschätzt wird, macht es keinen Unterschied, ob es sich um den Nutzen der Hypothese bezüglich der wesentlich größeren gesamten Trainingsmenge oder des Instanzenraumes handelt. Die Aussage ist in beiden Fällen gültig, so dass auch statt der Trainingsmenge T der Instanzenraum X verwendet werden kann.
Es kann leicht passieren, dass eine der Hypothesen nach wenigen gezogenen Beispielen schon einen sehr guten Nutzen hat. Dieses kann bei geringer Beispielanzahl zufällig bedingt sein. Man betrachte als Beispiel für ein Zufallsexperiment das wiederholte Werfen einer fairen Münze, bei der die mit p bezeichnete Wahrscheinlichkeit für Kopf 0.5 beträgt. Gemäß der Tabelle der Binomialverteilung beträgt die Wahrscheinlichkeit, dass nach drei Münzwürfen dreimal Kopf geworfen wurde, 0.125. Damit ist dieser Fall nicht unrealistisch; es wäre aber nicht gerechtfertigt p=1, als Schätzung für die Wahrscheinlichkeit für Kopf anzugeben. Daher wird zusätzlich eine Konfidenzschranke angegeben,

welche für die durchgeführte Anzahl von Zufallsexperimenten m und die gegebene Irrtumswahrscheinlichkeit δ ein Intervall um den geschätzten Wert für p festlegt. Der wahre Wert für p liegt mit Wahrscheinlichkeit bzw. Konfidenz $1-\delta$ innerhalb dieses Intervalls. Das Beispiel des wiederholten Münzwurfes lässt sich analog auf den Fall übertragen, bei dem der Wert einer Nutzenfunktion nach wiederholter Durchführung des Zufallsexperimentes ‚Ziehen eines Beispiels x aus der Trainingsmenge T' geschätzt wird. Abhängig von der gewünschten Konfidenz und der bisher gezogenen Anzahl von Beispielen wird eine Konfidenzschranke angegeben, so dass der wahre Nutzen der betrachteten Hypothese mit Konfidenz $1-\delta$ im durch diese Schranke bestimmten Intervall um den geschätzten Nutzen liegt.

Definition 16. *(Konfidenzintervall für den Nutzen)*
Seien Trainingsmenge T, Nutzenfunktion q und Hypothesenraum H gegeben. Für eine Hypothese $h \in H$ bezeichnet $q(h,T)$ den Nutzen von h für die gesamte Trainingsmenge T und $\hat{q}(h,Q_m)$ den Nutzen von h auf einer Stichprobe $Q_m \subseteq T$ der Größe m. Dann ist $E : \mathbb{N} \times \mathbb{R} \rightarrow \mathbb{R}$ eine Konfidenzschranke für den Nutzen q, falls für jede Fehlerwahrscheinlichkeit $\delta,\ 0 < \delta \leq 1$ gilt:

$$Pr[|\hat{q}(h,Q_m) - q(h,T)| \leq E(m,\delta)] \geq 1-\delta.$$

Wie durch die Betragsstriche deutlich wird, handelt es sich um ein zweiseitiges Konfidenzintervall. $E(m,\delta)$ liefert einen Wert ϵ, so dass für die gegebene Stichprobengröße m mit Konfidenz $1-\delta$ der wahre Wert der Nutzenfunktion in dem durch ϵ bestimmten Intervall um den geschätzten Nutzen liegt. Für kleine Fehlerwahrscheinlichkeiten δ liegt die Konfidenz nahe bei eins. In diesem Fall wird das Konfidenzintervall um $\hat{q}(h,Q_m)$ groß. Ein Unterschied zum Zufallsexperiment des wiederholten Münzwurfes besteht darin, dass die Nutzenfunktion nicht auf eine einfache Wahrscheinlichkeit beschränkt ist; es muss nur möglich sein, ein Konfidenzintervall für sie anzugeben. Eine wichtige Eigenschaft des Konfidenzintervalles ist, dass es mit steigender Beispielanzahl immer kleiner wird und schließlich ganz verschwindet. D.h. für alle möglichen Werte von ϵ und δ, gibt es eine Stichprobengröße m, so dass $E(m,\delta) \leq \epsilon$ ist. Der GSS Algorithmus erlaubt, dass ein Konfidenzintervall von den Eigenschaften einer Hypothese h, wie z.B. der Standardabweichung des Nutzens von h, abhängt. Verdeutlicht wird dieses durch den Index h bei der Konfidenzschranke $E_h(m,\delta)$.

4.1.2 Algorithmus

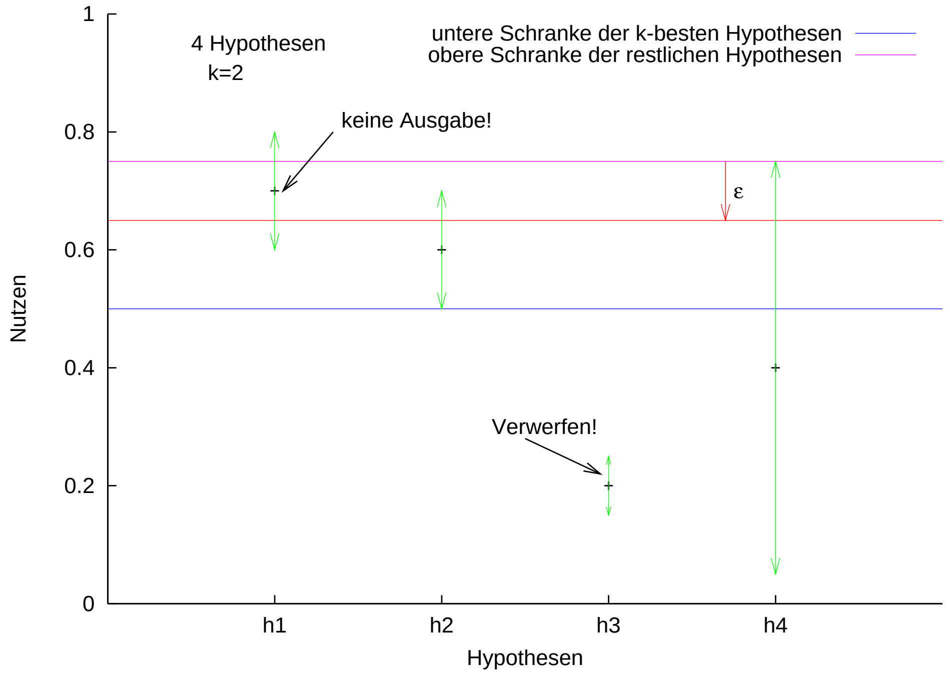
In Tabelle 4.1 ist der GSS Algorithmus angegeben. Es kann bewiesen werden, dass der Algorithmus terminiert und das approximativ k-beste Hypothesen Problem löst [25]. An drei Stellen werden die im vorangegangenen Abschnitt eingeführten Konfidenzintervalle berechnet, die ebenso wie die Aufteilung von δ und ϵ einer näheren Erläuterung bedürfen. In Schritt 2 des Algorithmus wird die maximale Beispielanzahl M berechnet, nach der sicher ist, dass die Abweichung des geschätzten empirischen Nutzens **jeder** Hypothese $h \in H$ in beide Richtungen höchstens $\frac{\epsilon}{2}$ beträgt. M bezeichnet zugleich die maximale Anzahl von Schleifendurchläufen in Schritt 3. Wird die maximale Beispielanzahl M erreicht, steht mit gewünschter Konfidenz fest, dass der wahre Nutzen der Hypothesen maximal $\frac{\epsilon}{2}$ um den geschätzten Nutzen schwankt. Es können dann in Schritt 4 die k-besten Hypothesen ausgegeben werden, da selbst im schlechtesten Fall der maximal zulässige Fehler

Eingabe: $X, T, k, q, \delta, \epsilon$
Ausgabe: Die approximativ k-besten Hypothesen mit Maximalfehler ϵ und Konfidenz $1 - \delta$

1. Initialisierung.
   a) Erzeuge H, die Menge aller Hypothesen für den Instanzenraum X.
   b) Setze i=1 (Schleifenzähler).
   c) Sei $Q_0 = \emptyset$.
2. Berechne die kleinste Zahl M, so dass $E(M, \frac{\delta}{2|H|}) \leq \frac{\epsilon}{2}$ ist.
3. **do**
   a) Ziehe zufällig mit Zurücklegen eine Instanz x_i aus T und füge sie Q_i hinzu: $Q_i = Q_{i-1} \cup x_i$.
   b) Aktualisiere den empirischen Nutzen $\hat{q}(h, Q_i)$ aller verbliebenen Hypothesen $h \in H$.
   c) Bestimme die Menge H^* der Hypothesen $h \in H$, die den größten empirischen Nutzen $\hat{q}(h, Q_i)$ haben.
   d) **for** $(h \in H)$ **do**
      i. **if** $(\hat{q}(h, Q_i) - E_h(i, \frac{\delta}{2M|H|}) \geq \max_{h' \in H \setminus H^*} \left\{ \hat{q}(h', Q_i) + E_{h'}(i, \frac{\delta}{2M|H|}) \right\} - \epsilon$ **und** $h \in H^*)$
         - Ausgabe h.
         - Entferne h aus H.
         - Setze k=k-1.
         - Berechne H^* neu.
      ii. **if** $(\hat{q}(h, Q_i) + E_h(i, \frac{\delta}{2M|H|}) \leq \min_{h' \in H^*} \left\{ \hat{q}(h', Q_i) - E_{h'}(i, \frac{\delta}{2M|H|}) \right\})$
         - Entferne h aus H.
   e) Setze i=i+1.

   while $(k \neq 0$ **und** $|H| \neq k$ **und** $E(i, \frac{\delta}{2|H|}) > \frac{\epsilon}{2})$
4. Gib die verbliebenen k Hypothesen in H^* aus.

Abbildung 4.1: Der Generic Sequential Sampling Algorithmus



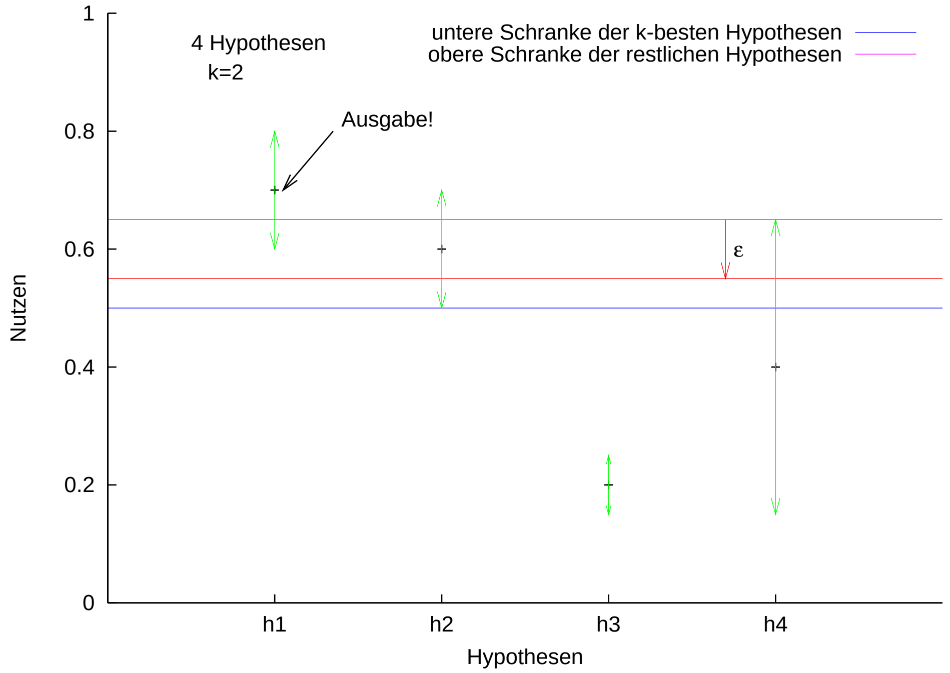


Abbildung 4.2: Funktionsweise des GSS Algorithmus in Schritt 3d

ϵ nicht überschritten wird. Dieser Fall tritt ein, wenn der wahre Nutzen der schlechtesten ausgegebenen Hypothese um $\frac{\epsilon}{2}$ nach unten abweicht, während der wahre Nutzen der besten Hypothese, die nicht ausgegeben wurde, um $\frac{\epsilon}{2}$ nach oben abweicht.
Von der zur Verfügung stehenden Irrtumswahrscheinlichkeit δ wird jeweils $\frac{\delta}{2}$ für die Berechnung der Konfidenzintervalle in der Abbruchbedingung der Schleife in Schritt 3 und innerhalb der Schleife (Schritt 3d) benutzt. Trifft die Abbruchbedingung der Schleife zu, wurde für jede verbliebene Hypothese ein Konfidenzintervall berechnet. Daher ist es nötig, die Irrtumswahrscheinlichkeit auf alle verbliebenen Hypothesen zu verteilen. Der wahre Nutzen einer Hypothese liegt dann nur mit einer Wahrscheinlichkeit von $\frac{\delta}{2|H|}$ außerhalb des Konfidenzintervalles um ihren geschätzten Nutzen. Es bleibt zu klären, ob dadurch die Irrtumswahrscheinlichkeit von $\frac{\delta}{2}$ eingehalten wird. Der Fall, dass der wahre Nutzen einer Hypothese außerhalb des Konfidenzintervalles liegt, wird als negatives Ereignis bezeichnet. Das negative Ereignis tritt für jede Hypothese mit Wahrscheinlichkeit $\frac{\delta}{2|H|}$ ein. Es ergibt sich eine Menge von Ereignissen, die alle die gleiche Wahrscheinlichkeit haben. Die Boolesche Ungleichung (Union Bound) besagt, dass für eine Menge von Ereignissen die Wahrscheinlichkeit, dass mindestens eines dieser Ereignis eintritt, nicht größer ist als die Summe der Wahrscheinlichkeiten aller Ereignisse. Damit wird insgesamt die Irrtumswahrscheinlichkeit von $\frac{\delta}{2}$ eingehalten. In Schritt 3d des GSS Algorithmus wird mit der anderen Hälfte der Irrtumswahrscheinlichkeit in **jedem** Schleifendurchlauf ein Konfidenzintervall für **alle** verbliebenen Hypothesen berechnet. Da die Schleife im schlechtesten Fall M-mal durchlaufen wird, muss diese Hälfte der Irrtumswahrscheinlichkeit zusätzlich durch M geteilt werden. Die Einhaltung der Irrtumswahrscheinlichkeit folgt wieder aus der booleschen Ungleichung. Das Vorgehen des Algorithmus in Schritt 3d wird in Abbildung 4.2 verdeutlicht. Exemplarisch sind Nutzen und die Konfidenzintervalle von vier Hypothesen dargestellt. Große Konfidenzintervalle bedeuten, dass nicht viel über die Qualität der Hypothese bekannt ist und der wahre Nutzen stark von der Schätzung abweichen kann. Wichtig ist zum einen die durch die schlechteste der k besten Hypothesen und deren Konfidenzintervall festgelegte untere Schranke. Jede Hypothese, die für den Fall, dass ihr wahrer Nutzen am oberen Ende der durch ihren geschätzten Nutzen und Konfidenzintervall festgelegten Reichweite liegt, schlechter ist als die untere Schranke, kann verworfen werden. Es ist für die gewünschte Konfidenz sicher, dass noch k bessere Hypothesen vorhanden sind. Zum anderen ist die durch (k+1)-beste Hypothese, deren Konfidenzintervall und ϵ festgelegte obere Schranke von Bedeutung. Gilt für eine der k-besten Hypothesen, dass ihr Nutzen auch im schlechtesten Fall noch über der Schranke liegt, kann sie ausgegeben werden. Es ist sicher, dass sie für die gegebene Konfidenz $1-\delta$ und den maximalen Fehler ϵ gut genug ist, um zu den approximativ k-besten Hypothesen zu gehören. Das frühzeitige Ausgeben bzw. Verwerfen von Hypothesen kann dazu führen, dass der Algorithmus terminiert bevor die maximale nötige Anzahl von Beispielen gezogen wurde, wenn vorher schon alle k Lösungen gefunden wurden. In der Praxis kommt dieses häufig vor und ist essentiell für eine gute Laufzeit. Der Vorteil des Tests in Schritt 3d beginnt sich auszuwirken, sobald die erste Hypothese aus H gelöscht wird. Mit jeder gelöschten Hypothese wird |H| kleiner und δ muss auf weniger Hypothesen verteilt werden, wodurch die berechneten Konfidenzintervalle besser werden. Dieses macht deutlich, dass komplexe (große) Hypothesenräume für den Algorithmus problematisch sind. In der Praxis muss die Komplexität beschränkt werden. Insbesondere müssen numerische Attribute diskretisiert werden, da es sonst nicht möglich ist, den Hypothesen-

raum komplett aufzuzählen. Ein weiteres Problem für die Laufzeit ist die Aufteilung der Irrtumswahrscheinlichkeit auf alle M Schleifendurchläufe. Der Wert von M kann abhängig vom betrachteten Hypothesenraum und gegebener Nutzenfunktion sehr groß werden (Kapitel 4.1.3). Es bietet sich an, Schritt 3d nicht in jedem Schleifendurchlauf durchzuführen und M entsprechend zu verkleinern. Dadurch werden kleinere Konfidenzintervalle möglich und es müssen nicht in jedem Schleifendurchlauf alle Berechnungen durchgeführt werden. Diese und andere Verbesserungen des GSS Algorithmus werden in Kapitel 5.1 beschrieben. Für verschiedene Nutzenfunktionen ergeben sich verschiedene Konfidenzintervalle und stark unterschiedliche Werte für M.

4.1.3 Konfidenzintervalle für verschiedene Nutzenfunktionen

In [25] werden Konfidenzintervalle für populäre Nutzenfunktionen hergeleitet. Das wiederholte Ziehen eines Beispiels im GSS Algorithmus entspricht dem Zufallsexperiment des Ziehens mit Zurücklegen. Handelt es sich bei der betrachteten Nutzenfunktion um eine Wahrscheinlichkeit wie es bei der Accuracy der Fall ist, unterliegt die nach m Versuchen beobachtete relative Häufigkeit der Binomialverteilung. Mit der Hoeffding-Ungleichung [17] ist möglich, für die absolute Summe $X = \sum_{i=1}^{m} X_i$ beschränkter Zufallsvariablen X_i eine Schranke für die Wahrscheinlichkeit anzugeben, mit der X weit vom erwarteten Wert liegt. Nehmen die X_i Werte zwischen 0 und Λ an, so gilt:

$$Pr[|X - E(X)| \leq \epsilon] \geq 1 - 2exp\left\{-2m\frac{\epsilon^2}{\Lambda^2}\right\}.$$

Analog ist diese Abschätzung möglich, wenn X eine relative Häufigkeit und der Erwartungswert E(X) die zugehörige Wahrscheinlichkeit ist. Für sehr große m kann die Binomialverteilung nach dem zentralen Grenzwertsatz[9] durch die Normalverteilung approximiert werden. Mit Hilfe der Normalverteilung können engere Schranken berechnet werden als mit der Hoeffding-Ungleichung. Daher wird für große m angenommen, dass die Abweichungen des geschätzten vom wahren Nutzen der Normalverteilung unterliegen. Das Konfidenzintervall wird mit Hilfe der Normalverteilung berechnet.

Als einführendes Beispiel für die Berechnung eines Konfidenzintervalles dienen Nutzenfunktionen, die berechnet werden, indem man den Durchschnitt über eine Instanznutzenfunktion bildet (*Instance-Averaging* Funktionen). Bei der Accuracy handelt es sich um eine solche Funktion. Für diesen Typ ist die Instanznutzenfunktion $q_{inst}(h, x_i)$ für eine Hypothese h und eine einzelne Instanz x_i definiert als

$$q_{inst}(h, x_i) = \begin{cases} 1, & falls\ h\ eine\ korrekte\ Vorhersage\ für\ x_i\ macht, \\ 0, & falls\ h\ eine\ falsche\ Vorhersage\ für\ x_i\ macht. \end{cases}$$

Für eine Trainingsmenge T der Größe n und eine Hypothese h ergibt sich dann der Nutzen q als

$$q(h, T) = \frac{1}{n}\sum_{i=1}^{n} q_{inst}(h, x_i).$$

Verwendet man in der Hoeffding-Ungleichung den nach m gezogenen Beispielen geschätzten empirischen Nutzen $\hat{q}(Q_m, h)$ als relative Frequenz und den wahren Wert q(T,h) als zugehörige Wahrscheinlichkeit, ist die Wahrscheinlichkeit, dass die Abweichung von wahrem und geschätztem Nutzen außerhalb des Konfidenzintervalles $E(m, \delta)$ liegt durch

$$Pr[|\hat{q}(Q_m, h) - q(T, h)| > E(m, \delta)] \leq 2exp\left\{-2m\frac{E(m, \delta)^2}{\Lambda^2}\right\}$$

beschränkt. Zu beachten ist, dass die Gegenwahrscheinlichkeit benutzt wird, da anders als in der Definition der Hoeffding-Ungleichung hier die Wahrscheinlichkeit dafür, dass der beobachtete Wert außerhalb des Konfidenzintervalles liegt, abgeschätzt wird. Wählt man die Konfidenzschranke abhängig von der Beispielanzahl m und der gewünschten Irrtumswahrscheinlichkeit δ als

$$E(m, \delta) = \sqrt{\frac{\Lambda^2}{2m} log\frac{2}{\delta}}$$

und berücksichtigt, dass für diesen Typ von Nutzenfunktionen $\Lambda = 1$ ist, wird die Irrtumswahrscheinlichkeit eingehalten:

$$2exp\left\{-2m\left(\sqrt{\frac{\Lambda^2}{2m} log\frac{2}{\delta}}\right)^2\right\} \leq 2exp\left\{-log\frac{2}{\delta}\right\} = \delta.$$

Nun müssen die Konfidenzschranken für die Approximation durch die Normalverteilung bestimmt werden. $\hat{q}(h, Q_m) - q(h)$ ist eine Zufallsvariable mit Mittelwert 0, wobei $\hat{q}(h, Q_m)$ Werte zwischen 0 und Λ annimmt. Um die Normalverteilung berechnen zu können, wird die Standardabweichung benötigt. In der Abbruchbedingung in Schritt 3 des GSS Algorithmus wird das Konfidenzintervall ohne Bezug zu einer konkreten Hypothese berechnet. Deshalb muss an dieser Stelle die größtmögliche Standardabweichung verwendet werden. Das Konfidenzintervall, welches man auf diese Weise erhält, ist aber immer noch kleiner als das mit der Hoeffding-Ungleichung bestimmte Intervall. Die empirische Standardabweichung

$$s_{\hat{q}(h,Q_m)-q(h)} = \frac{1}{m}\sqrt{\sum_{i=1}^{m}(q_{inst}(h, x_i) - \hat{q}(h, Q_m))^2}$$

wird maximiert, wenn $\hat{q}(h, Q_m) = \frac{\Lambda}{2}$ ist und die Instanznutzenfunktionen $q_{inst}(h, x_i)$ zur Hälfte 0 und zur Hälfte Λ beträgt. In diesem Fall beträgt sie $\frac{\Lambda}{2\sqrt{m}}$. Folglich unterliegt $2\sqrt{m} \cdot \left(\frac{\hat{q}(h,Q_m)-q(h)}{\Lambda}\right)$ der Standardnormalverteilung und

$$E(m, \delta) = z_{1-\frac{\delta}{2}} \cdot \frac{\Lambda}{2\sqrt{m}}$$

ist eine geeignete Konfidenzschranke.

Bei $z_{1-\frac{\delta}{2}}$ handelt es sich um das $1 - \frac{\delta}{2}$-Quantil der Standardnormalverteilung. In Abbildung 4.3 ist die Dichtekurve der Standardnormalverteilung dargestellt. Für $\delta = 0.1$ bezeichnet $z_{1-\frac{\delta}{2}}$ das durch den schraffierten Bereich dargestellte 0.95-Quantil. $z_{1-\frac{\delta}{2}}$ ist der Punkt auf der x-Achse, so dass $(1-\frac{\delta}{2})$% der Fläche unter der Kurve der Standardnormalverteilung links dieses Punktes liegen. Wählt man für eine standardnormalverteilte

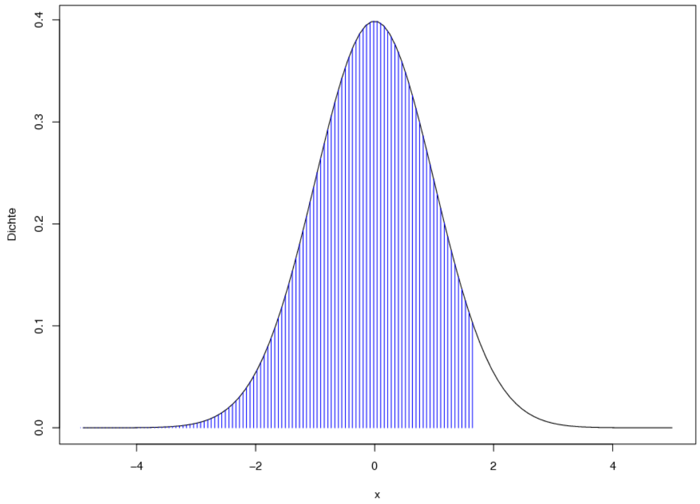


Abbildung 4.3: Das 0.95-Quantil der Standardnormalverteilung

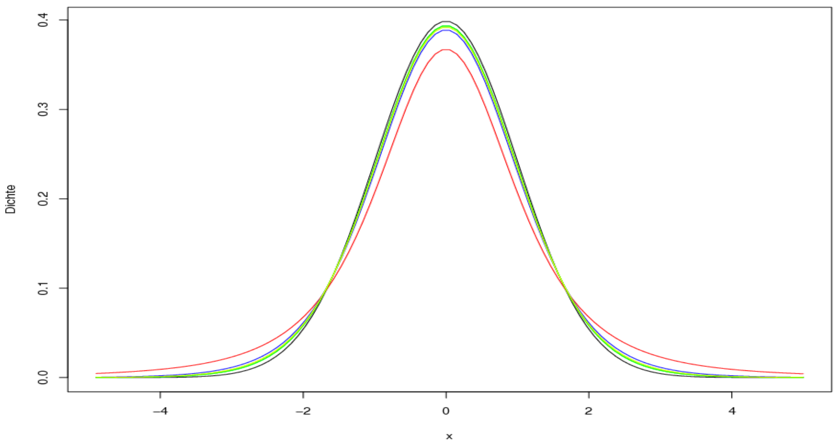


Abbildung 4.4: Normalverteilung(schwarz) und t-Verteilung für 5(rot), 10(blau) sowie 15(grün) Freiheitsgrade

Zufallsvariable diesen Punkt als Grenze des Konfidenzintervalles ist gesichert, dass der Wert der Zufallsvariablen mit Wahrscheinlichkeit $z_{1-\frac{\delta}{2}}$ nicht größer als dieser x-Wert ist. Für die Standardnormalverteilung leistet dieser Punkt das Gewünschte. Um die Konfidenzschranke für die tatsächliche Verteilung zu erhalten, genügt es den z-Wert mit der Standardabweichung zu multiplizieren. Die Halbierung von δ resultiert daraus, dass ein zweiseitiges Konfidenzintervall gesucht wird. Damit der Wert der Zufallsvariablen mit Konfidenz $1-\delta$ innerhalb des Intervalls liegt, werden die Intervallgrenzen so gewählt, dass er mit Wahrscheinlichkeit $\frac{\delta}{2}$ nicht kleiner als die untere und mit $\frac{\delta}{2}$ nicht größer als die obere Intervallgrenze ist.
In Schritt 3d wird das Konfidenzintervall für eine konkrete Hypothese gesucht. Anstatt die größstmögliche Standardabweichung zu verwenden, kann die empirische Standardabweichung von $\hat{f}(h, Q_m)$ für die Hypothese h berechnet werden. Die spezifische Konfidenzschranke $E_h(m, \delta)$ lautet

$$E_h(m,\delta) = z_{1-\frac{\delta}{2}} \cdot \frac{1}{m}\sqrt{\sum_{i=1}^{m}(q_{inst}(h,x_i) - \hat{q}(h,Q_m))^2}.$$

Diese Vorgehensweise ist allerdings ungenau. Da die wahre Standardabweichung für die gesamten Daten unbekannt ist, wird die empirische Standardabweichung der Stichprobe zur Abschätzung verwendet. In diesem Fall liegt keine Standardnormalverteilung vor, sondern eine t-Verteilung. Abhängig von der Zahl der Freiheitsgrade besitzt die Dichtekurve einer t-Verteilung im Gegensatz zur Standardnormalverteilung breitere Enden, d.h. die Wahrscheinlichkeit für große und kleine Werte von x sind größer; dafür liegt weniger Wahrscheinlichkeitsmasse im Zentrum (Abbildung 4.4). Es ergeben sich größere Konfidenzintervalle als für die Standardnormalverteilung, weshalb die t-Verteilung für robuste statistische Verfahren eingesetzt wird. Die Anzahl der Freiheitsgrade einer statistischen Kenngröße ist die Anzahl der unabhängigen Beobachtungswerte, die deren Berechnung zugrunde liegt, reduziert um die Anzahl der in die Berechnung eingehenden zusätzlichen Parameter, die ebenfalls auf den Beobachtungswerten basieren. So stehen für die Berechnung der empirischen Standardabweichung m Werte zur Verfügung. Als weiterer Parameter wird der Mittelwert benötigt. Daher ist die Anzahl der Freiheitsgrade gleich m-1. Für $m \to \infty$ konvergiert die Dichtekurve der t-Verteilung gegen die Dichte der Standardnormalverteilung und ab m=30 ist die Approximation bereits sehr gut. Da bei Experimenten mit dem Algorithmus festgestellt wurde, dass keine Hypothesen ausgegeben werden bevor die Größe der Stichprobe einige hundert Beispiele erreicht, wird die Normalverteilung verwendet. Da eine Kombination von Coverage und Precision häufig in die Berechnung komplexer Nutzenfunktion eingeht, bleibt dieses Vorgehen für Hypothesen mit kleiner Coverage und einer Precision, deren Wert weit von 0.5 entfernt ist, problematisch. Eine Stichprobe von einigen hundert Beispielen genügt nicht, wenn die Hypothese nur wenige dieser Beispiele abdeckt. Die Schätzung der Coverage erfolgt auf Basis aller Beispiele in der Stichprobe, die der Precision aber nur aufgrund der wenigen abgedeckten Beispiele. In diesem Fall ist die Approximation der binomialverteilten Precision durch die Normalverteilung schlecht. Auf diese Problematik und eine mögliche Lösung wird in Kapitel 5.1.2 eingegangen.
Abschließend wird die maximale Beispielanzahl hergeleitet, die der Algorithmus für diese Klasse von Funktionen benötigt, bevor er terminiert. Die Schleife in Schritt 3 wird

spätestens abgebrochen, wenn die Abbruchbedingung $E\left(m, \frac{\delta}{2|H|}\right) \leq \frac{\epsilon}{2}$ eintritt. Falls $m = \frac{2\Lambda^2}{\epsilon^2} log \frac{4|H|}{\delta}$ ergibt sich mit Hilfe der Hoeffding-Schranke:

$$E\left(\frac{2\Lambda^2}{\epsilon^2} log \frac{4|H|}{\delta}, \frac{\delta}{2|H|}\right) = \sqrt{\frac{\Lambda^2}{2\left(\frac{2\Lambda^2}{\epsilon^2} log \frac{4|H|}{\delta}\right)} log \frac{2}{\frac{\delta}{2|H|}}} = \sqrt{\frac{\epsilon^2 log \frac{4|H|}{\delta}}{4 log \frac{4|H|}{\delta}}} = \frac{\epsilon}{2}.$$

In der Regel wird der Algorithmus weniger Beispiele brauchen als es die worst-case-Schranke vermuten lässt. Zum einen werden statt der konservativen Hoeffding-Schranken die mit Hilfe der Normalverteilung berechneten Schranken verwendet, was zu besseren (kleineren) Konfidenzintervallen führt. Zum anderen wird die maximale Beispielanzahl nur für den Fall benötigt, dass der Nutzen aller Hypothesen ähnlich gut ist. Unterscheiden sich gute und schlechte Hypothesen, führt der Ansatz des sequentiellen Sampling schneller zur Terminierung. In den Tabellen 4.1 und 4.2 sind Konfidenzschranken und maximal benötigte Stichprobengröße für alle betrachteten Nutzenfunktionen angegeben. Weighted Relative Accuracy, Squared und Binomial multiplizieren Coverage und Bias. Daher werden bei diesen Konfidenzschranken die empirischen Standardabweichungen von Coverage s_g und Precision in der Subgruppe s_p benötigt. In der Tabelle sind sowohl die mit der Hoeffding-Schranke als auch die beiden mit der Normalverteilung berechneten Konfidenzschranken angegeben. Eine Besonderheit besteht für die Weighted Relative Accuracy. Falls die apriori Wahrscheinlichkeit 0.5 beträgt, sind die Nutzenfunktionen Accuracy und Weighted Relative Accuracy proportional [26]. Für bekannte apriori Wahrscheinlichkeit läßt sich zeigen, dass die Weighted Relative Accuracy ebenfalls eine Instance-Averaging Nutzenfunktion ist [28]. Daher können statt der Schranken aus [25] die genaueren Schranken für Instance-Averaging Funktionen verwendet werden.

4.2 Knowledge-Based Sampling

Ein Nachteil vieler Verfahren zur Subgruppenentdeckung - auch des GSS Algorithmus - ist, dass kein Vorwissen über gefundene Subgruppen berücksichtigt wird. Das Vorwissen kann sowohl vor dem Data-Mining Schritt durch Experten des Problembereiches geliefert als auch durch frühere Anwendung eines Lernverfahrens gefunden werden. Zur Repräsentation des Vorwissens über Subgruppen werden die in Kapitel 2.4 eingeführten probabilistischen Regeln verwendet. Beim Expertenwissen besteht die Gefahr, diese Regeln erneut in den Daten zu finden, wenn es sich dabei um solche Subgruppen handelt, die bezüglich der gewählten Nutzenfunktion auffällig sind. Selbst bei Verfahren, die sicherstellen, dass jede gute Hypothese bzw. Regel nur einmal gefunden wird, indem sie bei Ausgabe aus dem Hypothesenraum entfernt wird, werden oft redundante Informationen gefunden. Der Grund sind korrelierte Attribute. Regeln beschreiben Subgruppen und bestehen aus der Menge von Beispielen für die sie anwendbar sind. Bei stark korrelierten Attributen gibt es mehrere Regeln, die ähnliche Subgruppen beschreiben. Beim Beispiel der Daten einer Bank sind für jeden Kunden Alter und Beruf gespeichert. Als interessierendes Merkmal wurde das boolesche Attribut genannt, das angibt, ob der Kunde in einen speziellen Investmentfond der Bank investiert hat. Wurde für die Subgruppe der über 65-jährigen Kunden festgestellt, dass dort Anlagen in den Investmentfond entgegen den Rest der Kundschaft selten sind, gilt dieses mit Sicherheit auch für die Subgruppe

Nutzenfunktion	Konfidenzintervalle
Instance-Averaging	$E(m,\delta) = \sqrt{\frac{\Lambda^2}{2m} log\frac{2}{\delta}}$ $E(m,\delta) = z_{1-\frac{\delta}{2}} \cdot \frac{\Lambda}{2\sqrt{m}}$ $E_h(m,\delta) = z_{1-\frac{\delta}{2}} \cdot \frac{1}{m}\sqrt{\sum_{i=1}^{m}(q_{inst}(h,x_i) - \hat{q}(h,Q_m))^2}$
WRACC	$E(m,\delta) = 3\sqrt{\frac{1}{2m} log\frac{\delta}{4}}$ $E(m,\delta) = \frac{z_{1-\frac{\delta}{4}}}{\sqrt{m}} + \frac{(z_{1-\frac{\delta}{4}})^2}{4m}$ $E_h(m,\delta) = z_{1-\frac{\delta}{4}}(s_g + s_p + z_{1-\frac{\delta}{4}} s_g s_p)$
SQUARED	$E(m,\delta) = (\frac{1}{2m} log\frac{4}{\delta})^{\frac{3}{2}} + 3(\frac{1}{2m} log\frac{4}{\delta}) + 3\sqrt{\frac{1}{2m} log\frac{4}{\delta}}$ $E(m,\delta) = \frac{3}{2\sqrt{m}} z_{1-\frac{\delta}{2}} + \frac{m+\sqrt{m}}{4m\sqrt{m}}(z_{1-\frac{\delta}{2}})^2 + \frac{1}{8m\sqrt{m}}(z_{1-\frac{\delta}{2}})^3$ $E_h(m,\delta) = 2s_g z_{1-\frac{\delta}{2}} + s_g^2(z_{1-\frac{\delta}{2}})^2 + s_p z_{1-\frac{\delta}{2}} + 2s_g s_p(z_{1-\frac{\delta}{2}})^2 + s_p s_g^2(z_{1-\frac{\delta}{2}})^3$
BINOMIAL	$E(m,\delta) = \sqrt[2]{\frac{1}{2m} log\frac{4}{\delta}} + \sqrt[4]{\frac{1}{2m} log\frac{4}{\delta}} + (\frac{1}{2m} log\frac{4}{\delta})^{\frac{3}{4}}$ $E(m,\delta) = \sqrt{\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}}} + \frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}} + (\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}})^{\frac{3}{2}}$ $E_h(m,\delta) = \sqrt{s_g z_{1-\frac{\delta}{4}}} + s_p z_{1-\frac{\delta}{4}} + \sqrt{s_g z_{1-\frac{\delta}{4}}} s_p z_{1-\frac{\delta}{4}}$

Tabelle 4.1: Konfidenzintervalle für die verschiedenen Nutzenfunktionen

Nutzenfunktion	maximale Beispielanzahl
Instance-Averaging	$\frac{2\Lambda}{\epsilon^2} log\frac{4\lvert H\rvert}{\delta}$
WRACC	$\frac{18}{\epsilon^2} log\frac{8\lvert H\rvert}{\delta}$
SQUARED	$\frac{98}{\epsilon^2} log\frac{8\lvert H\rvert}{\delta}$
BINOMIAL	$\frac{648}{\epsilon^2} log\frac{8\lvert H\rvert}{\delta}$

Tabelle 4.2: Maximal benötigte Beispielanzahlen

aller Kunden, die Rentner sind. Es sollten dann nicht beide Subgruppen ausgegeben werden. Aus diesem Grunde ist es nötig, die durch bereits gefundene Regeln beschriebenen Korrelationen aus den Daten zu entfernen und das Lernverfahren zu 'zwingen', andere Strukturen in den Daten zu finden. Ein weiterer Nachteil tritt im Zusammenhang mit dem Hypothesenraum der probabilistischen Regeln auf. Klassische Verfahren können keine Ausnahmen von gefunden Regeln entdecken, obwohl dieses zur genauen Beschreibung des Datensatzes hilfreich sein kann. Gilt beispielsweise für eine Regel $r \rightarrow Y_+$

$$Pr[Y_+|r] = Pr[Y - +] = 0.5$$

ist diese zunächst uninteressant, da $BIAS(r \rightarrow Y_+) = 0$ ist. Gibt es jedoch im Vorwissen eine Regel

$$r' \rightarrow Y_+ \ [0.1] \ mit \ r \subset r'$$

ist diese als Ausnahme vom Vorwissen von Interesse. Für das Beispiel der Subgruppe der Rentner mit geringem Anteil von Anlagen in einen speziellen Investmentfond ($r' \rightarrow Y_+$), entspricht $r \rightarrow Y_+$ der kleineren Subgruppe der vermögenden Rentner. Für diesen speziellen Personenkreis bieten Banken meist eine intensivere Beratung an, wodurch eine stärkere Anlage in Investmentfond fokussiert wird.
Aus diesen Gründen bietet sich an, die Interessantheit von Subgruppen unter der Bedingung der Unerwartetheit zu betrachten. Dazu wird ein iteratives Verfahren angewandt, das in jedem Schritt eine Subgruppe findet und dieses Wissen aus den Daten entfernt. Im folgenden Durchgang wird die Aufmerksamkeit auf solche Subgruppen gelenkt, die einen hohen Nutzen haben, dabei aber unabhängig vom gesamten Vorwissen sind. Ziel ist eine Menge von Subgruppen zu finden, die möglichst klein ist, da sie wenig Redundanzen enthält und trotzdem die Informationen in den gegebenen Daten bezüglich des interessierenden Merkmals aufgrund ihrer hohen Diversität gut beschreibt. In Anlehnung an [26] wird dieses Verfahren in den nächsten Abschnitten formalisiert und beschrieben, wie es sich operationalisieren lässt.

4.2.1 Grundlagen

Zur Beantwortung der Frage, wie Vorwissen aus dem gegebenen Datensatz entfernt werden soll, muss bestimmt werden wie Vorwissen aussieht. Sowohl Konzeptlernen aus Beispielen als auch Subgruppenentdeckung bedürfen der Festlegung eines Zielattributes. Durch in Form einer Regel bestimmtes Vorwissen wird eine Vorhersage für die Klasse des Zielattributes getroffen, egal ob die Regel als Repräsentation einer Subgruppe oder als Klassifikationsregel interpretiert wird. Vorwissen wird daher als Korrelation zwischen den tatsächlichen Klassen der Instanzen und den Vorhersagen der Regel definiert. Beim Knowledge-Based Sampling (KBS) [26] handelt es sich um ein iteratives Verfahren, das diese Korrelation entfernt, indem es die Verteilung der Beispiele über dem Instanzenraum verändert. Für den Fall, dass zu Beginn kein Vorwissen bekannt ist, hat jedes Beispiel die gleiche Wahrscheinlichkeit gezogen zu werden (Gleichverteilung). Ist die erste Regel gefunden, wird eine neue Verteilung konstruiert, so dass die Korrelation zwischen den Vorhersagen der Regel und dem Zielattribut nicht mehr vorhanden ist. Im folgenden Durchgang wird eine Stichprobe von Beispielen gemäß der veränderten Verteilung gezogen und nach neuen Regeln gesucht. Dieses Vorgehen kann beliebig wiederholt werden, wobei dem Vorwissen in jedem Durchgang eine Regel hinzugefügt wird, die unkorreliert zu allen bisher gefundenen ist. Zur Bestimmung der Regeln kann ein beliebiges Lernverfahren als *Basislerner* eingesetzt werden.

4.2.2 Bedingungen an die neue Verteilung

Um das in Form von Regeln gegebene oder gefundene Vorwissen in mehreren KBS-Iterationen aus den Daten zu entfernen, wurde eine Verteilung D konstruiert, nach der die Beispiele in der folgenden Iteration gezogen werden müssen. Wird eine weitere zum Vorwissen orthogonale Regel $R : r \rightarrow Y_*$ gefunden, muss eine neue Verteilung D' konstruiert werden, die auch dieses Wissen entfernt. Gleichzeitig sollte D' möglichst ähnlich zur alten Verteilung D sein, damit die verbleibenden Muster in den Daten nicht zerstört werden. Die Eigenschaften, die dafür erfüllt werden müssen, lassen sich anschaulich in Form von drei Bedingungen an D' formulieren. Zur Verdeutlichung werden die Bedingungen anhand der bekannten Subgruppe der Rentner unter den Kunden einer Bank demonstriert, in der der Anteil von Anlagen in einen speziellen Investmentfond gegenüber dem Rest der Kundschaft gering ist.
Da das durch die Regel (Subgruppe) R repräsentierte Wissen entfernt werden soll, müssen r und Y_* unter der neuen Verteilung unabhängige Ereignisse sein: Die Wahrscheinlichkeit für die interessierende Klasse in der Subgruppe muss gleich der apriori Wahrscheinlichkeit sein.

$$Pr_{D'}[Y_*|r] = Pr_{D'}[Y_*]$$

Dieses bedeutet übertragen auf das Beispiel der Bankkunden, dass unter der neuen Verteilung der Anteil der Kunden, die in den Fond investiert haben, in den gesamten Daten gleich dem Anteil in der Subgruppe von Rentnern ist. Da nur das Wissen über R entfernt werden und ansonsten die neue Verteilung möglichst unverändert bleiben soll, müssen die Wahrscheinlichkeit, dass R anwendbar ist und die apriori Wahrscheinlichkeit gleich bleiben.

$$Pr_{D'}[r] = Pr_D[r]$$
$$Pr_{D'}[Y_*] = Pr_D[Y_*]$$

Für die Beispielregel dürfen sich der Anteil von Kunden, die in den Fond investiert haben und die Wahrscheinlichkeit, zufällig einen Rentner aus der Menge aller Kunden zu ziehen, unter der neuen Verteilung nicht verändern. Durch die Vorhersagen der Regel R und den wahren Klassen der Instanzen wird auf der Menge der Daten eine Partition aus vier Klassen induziert. Betrachtet man jede Klasse für sich, muss innerhalb dieser Klasse die Wahrscheinlichkeit, eine bestimmte Instanz zu ziehen, gleich bleiben. Der Grund besteht darin, dass alle Instanzen innerhalb einer Partition bezüglich des Vorwissens in Form der Regel R nicht zu unterscheiden sind. Es verändern sich nur die Randwahrscheinlichkeiten.

$$Pr_{D'}[x|r \cap Y_+] = Pr_D[x|r \cap Y_+]$$
$$Pr_{D'}[x|r \cap Y_-] = Pr_D[x|r \cap Y_-]$$
$$Pr_{D'}[x|\bar{r} \cap Y_+] = Pr_D[x|\bar{r} \cap Y_+]$$
$$Pr_{D'}[x|\bar{r} \cap Y_-] = Pr_D[x|\bar{r} \cap Y_-]$$

Für das Beispiel der Bankkunden wird die Partition bzw. werden die vier Klassen durch die Attribute ‚Rentner' und ‚Fond' festgelegt, die beide die Ausprägungen ‚ja' und ‚nein'

haben. Wenn sich die Wahrscheinlichkeit einen Rentner zu sehen, der in den Fond investiert hat, unter der neuen Verteilung halbiert, passiert dasselbe auch für alle anderen Kunden mit diesen zwei Eigenschaften. Es verändert sich nur die Wahrscheinlichkeit einen Rentner zu sehen, der in den Fond investiert hat, d.h. die Größe der Partition als Ganzes wird verändert.
Werden die Instanzen nach der neuen Verteilung D' gezogen, ist die Regel R nicht mehr in der Stichprobe zu finden. Andere interessante Regeln bleiben in der Stichprobe erhalten, selbst wenn sie mit R überlappen. Für Subgruppen, die eine Teilmenge von R sind wie z.B. die häufiger in den Fond investierten vermögenden Rentner, wird nur die Größe, entsprechend der Partition in der sie liegen, angepasst.

4.2.3 Konstruktion der neuen Verteilung

Nach der Beschreibung der Anforderungen an die neue Verteilung im vorherigen Abschnitt muss gezeigt werden, wie sie konstruiert werden kann. Desweiteren wird eine Möglichkeit benötigt, um Instanzen gemäß dieser neuen Verteilung zu ziehen. Zunächst wird der Lift einer Instanz x für eine Regel $R : r \rightarrow Y_*$ definiert.

$$LIFT(r \rightarrow Y_*, x) := \begin{cases} LIFT(r \rightarrow Y_*), & falls\ x \in r \cap Y_* \\ LIFT(r \rightarrow \bar{Y_*}), & falls\ x \in r \cap \bar{Y_*} \\ LIFT(\bar{r} \rightarrow Y_*), & falls\ x \in \bar{r} \cap Y_* \\ LIFT(\bar{r} \rightarrow \bar{Y_*}), & falls\ x \in \bar{r} \cap \bar{Y_*} \end{cases}$$

$Pr_D(x)$ steht für die Wahrscheinlichkeit, eine Instanz x aus dem Instanzenraum X unter einer Verteilung D zu ziehen. $LIFT_D(R, x)$ bezeichnet den Lift einer Instanz x für eine Regel R unter der Verteilung D. Es lässt sich für jede Anfangsverteilung D zeigen, dass die Bedingungen an die neue Verteilung D' genau dann eingehalten werden, wenn

$$Pr_{D'}(x) = Pr_D(x) \cdot (LIFT_D(R, x))^{-1}$$

ist [26].
Auf diese Weise ist es möglich, die Wahrscheinlichkeit für jede Instanz unter der neuen Verteilung zu berechnen. Während bei Normal- oder Gleichverteilung möglich ist Instanzen direkt zu ziehen, funktioniert dieses bei vielen anderen Wahrscheinlichkeitsverteilungen nicht. In diesem Fall kann die *Verwerfungsmethode(Rejection Sampling)* [20] eingesetzt werden. Zufallszahlen für eine Zielverteilungsfunktion F werden mit einer Hilfsverteilungsfunktion G erzeugt, für die sich leicht Zufallszahlen erzeugen lassen. Sind f und g die zugehörigen Dichtefunktionen, müssen für ein konstantes $k > 0$ folgende zwei Bedingungen gelten:

$$f(x) \leq k \cdot g(x), \forall x \in \mathbb{R}\ und$$

$$f(x) = 0 \Rightarrow g(x) = 0, \forall x \in \mathbb{R}.$$

Anschaulich muss die Kurve der Dichtefunktion von F komplett unter der mit k multiplizierten Kurve der Dichtefunktion von G liegen. Ist u die Realisierung einer auf dem Intervall [0,1] gleichverteilten Zufallsvariablen und v eine unter der Hilfsverteilungsfunktion G erzeugte Zufallszahl, genügt v der Zielverteilung, falls die Bedingung

$$u \leq \frac{f(v)}{k \cdot g(v)}$$

eingehalten wird. Da $k \cdot g(v)$ immer größer ist als $f(v)$, ist das Verhältnis beider Werte ebenfalls eine Zahl aus dem Intervall [0,1]. Falls u kleiner ist als das Verhältnis, genügt v der Zielverteilung, ansonsten wird v verworfen. Dieses wird solange wiederholt bis die Zufallszahl u zum ersten Mal die Bedingung einhält. Mit anderen Worten wartet man solange bis u zum ersten Mal unter der Kurve der Dichtefunktion der Zielverteilung liegt. Dabei wird auch deutlich, dass die Hilfsverteilung die Zielverteilung gut approximieren sollte, da sonst viele Zufallszahlen benötigt werden. Beim Knowledge-Based Sampling wird die Verwerfungsmethode genutzt, um Beispiele unter der neu erzeugten Verteilung zu ziehen. Als Hilfverteilung wird die Gleichverteilung über dem Instanzenraum X benutzt. Das Verhältnis von Hilfs- und Zielverteilung wird für jede Instanz durch ein Gewichtsattribut repräsentiert. Da zu Beginn die Beispiele gleichverteilt gezogen werden, erfolgt die Initialisierung der Gewichte mit Eins. Nach einer KBS-Iteration werden die Gewichte durch Multiplikation mit dem passenden Lift aktualisiert. Durch Lifts kleiner als eins kann es passieren, dass das durch die Gewichte repräsentierte Verhältnis größer als eins wird. Daher erfolgt nach jeder Iteration eine Normierung auf [0,1], indem jedes Beispielgewicht durch das Maximum aller Gewichte dividiert wird. In Abbildung 4.5 wird anhand von zwei Beispielen verdeutlicht, wie die Verwerfungsmethode eingesetzt wird, um Beispiele unter der Zielverteilung zu ziehen. Da durch die Normierung der Gewichte die Fläche unter der Kurve nicht mehr eins beträgt, ist der Begriff Verteilung nicht korrekt. Daher wird in Abbildung 4.5 der Begriff Zielfunktion verwendet[1]. Wird beispielsweise Instanz Nummer 3 gezogen, besteht eine Wahrscheinlichkeit von 0.8, dass diese Instanz in die Stichprobe aufgenommen wird, was durch das Gewicht von 0.8 ausgedrückt wird.

4.2.4 Anwendung zur Klassifikation

Da für per Knowledge-Based Sampling gefundene Regeln die Hoffnung besteht, dass sie eine gute Charakterisierung des Zielattributes mit hoher Diversität liefern, können die Ergebnisse auch zur Klassifikation verwendet werden. Ein *Ensemble* kombiniert mehrere Einzelmodelle zu einem Gesamtmodell, um die Vorhersagequalität für das Zielattribut zu erhöhen. Aus mehreren präzisen Modellen, die eine hohe Diversität aufweisen, lassen sich gute Ensembles konstruieren [5, 31]. Die Güte der Vorhersage des Ensembles kann daher als Qualitätsmaß für die gefundenen Subgruppen verwendet werden. Die Kombination der Einzelmodelle im Ensemble kann auf mehrere Arten erfolgen. Gebräuchlich sind ‚und'-Verknüpfungen der einzelnen Vorhersagen oder Mehrheitsentscheide. Im Fall des Knowledge-Based Sampling bietet sich eine andere Vorgehensweise an.

Ein probabilistischer Bayes-Klassifikator benutzt den Satz von Bayes zur Vorhersage der Klasse einer Instanz. Dazu werden die Merkmalsvariablen (*Features*) $F_1, ..., F_n$ benötigt, die originäre Attribute des Instanzenraumes oder konstruierte Merkmale beschreiben. Das Zielattribut wird in diesem Zusammenhang als abhängiges Merkmal bezeichnet. Für eine durch die Ausprägungen $\hat{f}_1, ..., \hat{f}_n$ der Merkmalsvariablen beschriebene Instanz lässt sich die bedingte Wahrscheinlichkeit für eine Klasse Y_* des abhängigen Merkmals mit

[1] Für die anfänglichen Beispielgewichte von eins kann nur von einer Verteilung gesprochen werden, wenn man die einzelnen Gewichte durch das Gesamtgewicht teilt.

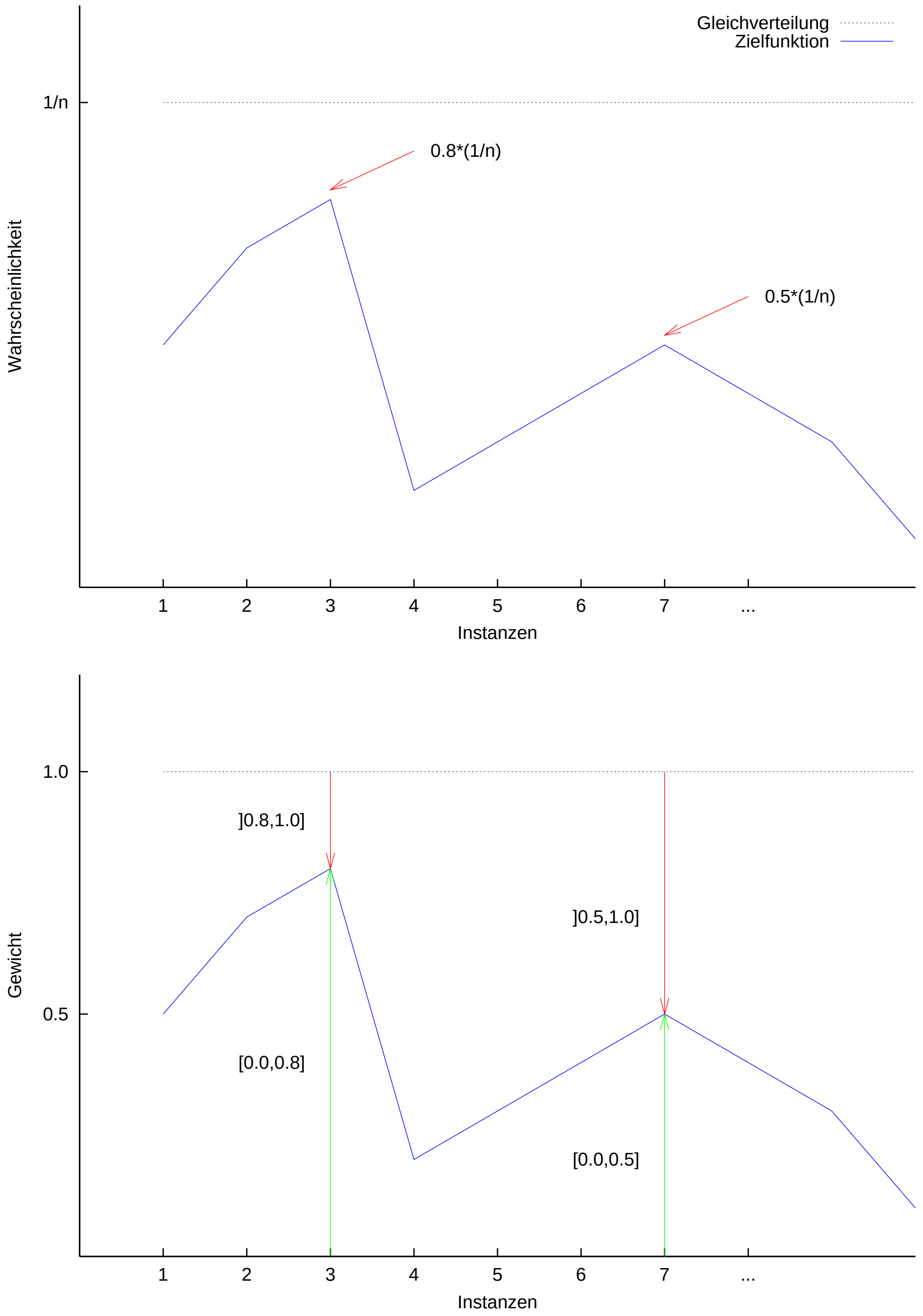


Abbildung 4.5: Anwendung der Verwerfungsmethode beim Knowledge-Based Sampling

Bayes Theorem berechnen:

$$Pr[Y_*|F_1 = \hat{f}_1, ..., F_n = \hat{f}_n] = \frac{Pr[Y_*] \cdot Pr[F_1 = \hat{f}_1, ..., F_n = \hat{f}_n|Y_*]}{Pr[F_1 = \hat{f}_1, ..., F_n = \hat{f}_n]}.$$

Als Vorhersage wird die Klasse gewählt, die die größte Wahrscheinlichkeit hat. Im folgenden werden für eine bessere Übersichtlichkeit die Ausprägungen der Merkmalsvariablen weggelassen; es wird aber immer von einer konkreten Instanz ausgegangen. Für die Entscheidung welche Klasse vorhergesagt wird, ist nur der Zähler von Bedeutung, da im Nenner nur Merkmalsvariablen vorkommen. Der Nenner ist für alle Klassen gleich und muss nicht berechnet werden. Trotzdem benötigt dieses Verfahren eine große Zahl von geschätzten Wahrscheinlichkeiten, was durch wiederholte Anwendung der Definition von bedingten Wahrscheinlichkeiten auf den Zähler deutlich wird.

$$Pr[Y_*] \cdot Pr[F_1, ..., F_n|Y_*] =$$

$$Pr[Y_*] \cdot Pr[F_1|Y_*] \cdot Pr[F_2, ..., F_n|Y_*, F_1] =$$

$$Pr[Y_*] \cdot Pr[F_1|Y_*] \cdot Pr[F_2|F_1, Y_*] \cdot Pr[F_2, ..., F_n|Y_*, F_1, F_2], \ usw.$$

Um die benötigte Anzahl von Wahrscheinlichkeiten zu verkleinern wird die ‚naive' Annahme gemacht, dass die Merkmalvariablen voneinander unabhängig sind. Damit wird obige Berechnung erheblich vereinfacht:

$$Pr[Y_*] \cdot Pr[F_1|Y_*] \cdot ... \cdot Pr[F_n|Y_*] = Pr[Y_*] \cdot \prod_{i=1}^{n} Pr[F_i|Y_*].$$

Dieses mit *Naive Bayes* bezeichnete Verfahren funktioniert in der Praxis oft gut, auch wenn die Annahme der Unabhängigkeit der Merkmalsvariablen verletzt wird.
Betrachtet man vor diesem Hintergrund Regeln, die iterativ mit Knowledge-Based Sampling gefunden wurden, ist klar, dass sich Naive Bayes zur Kombination der Vorhersagen der Regel anbietet. Korrelationen zwischen der gefundenen Regel und dem Zielattribut werden aus den Daten entfernt, bevor eine neue Regel gesucht wird. Man kann daher annehmen, dass die Vorhersagen unabhängig voneinander sind. Für die Regeln $r_i \rightarrow Y_*,\ 1 \leq i \leq n$ bzw. deren Vorhersagen $r_i(x) = y_i$ mit $y_i, Y_* \in \{Y_+, Y_-\}$ ergibt sich zusammen mit dem Satz von Bayes und der Annahme der Unabhängigkeit

$$Pr[Y_*|r_1 = y_1, ..., r_n = y_n] = Pr[Y_*] \cdot \frac{Pr[r_1 = y_i, ..., r_n = y_n|Y_*]}{Pr[r_1 = y_1, ..., r_n = y_n]}$$

$$\approx Pr[Y_*] \cdot \frac{\prod_{i=1}^{n} Pr[r_i = y_i|Y_*]}{Pr[r_1 = y_1, ..., r_n = y_n]}$$

$$= Pr[Y_*] \cdot \frac{\prod_{i=1}^{n} Pr[r_i = y_i]}{Pr[r_1 = y_1, ..., r_n = y_n]} \prod_{i=1}^{n} \frac{Pr[Y_*|r_i = y_i]}{Pr[Y_*]}.$$

Wendet man die Definition des Lift an, beträgt die bedingte Wahrscheinlichkeit für eine Klasse Y_*

$$Pr[Y_*] \cdot \frac{\prod_{i=1}^{n} Pr[r_i = y_i]}{Pr[r_1 = y_1, ..., r_n = y_n]} \prod_{i=1}^{n} LIFT((r_i = y_i) \rightarrow Y_*).$$

Daraus kann die wahrscheinlichste Klasse $\hat{y}$ bestimmt werden:

$$\hat{y} = argmax_{y_i \in \{Y_+, Y_i\}} \left[Pr[Y_*] \cdot \frac{\prod_{i=1}^{n} Pr[r_i = y_i]}{Pr[r_1 = y_1, ..., r_n = y_n]} \prod_{i=1}^{n} LIFT((r_i = y_i) \to Y_*) \right].$$

Sind die apriori Wahrscheinlichkeiten und die Lifts der gefundenen Regeln bekannt, ist es möglich aus ihnen ein Ensemble zu konstruieren. Damit steht eine Möglichkeit zur Verfügung, um die per Knowledge-Based Sampling gefundenen Subgruppen zu einem Ensemble zu kombinieren.
Obwohl die Kombination der einzelnen Regeln zu einem Ensemble über Naive Bayes motiviert ist, sind die Vorhersagen der Regeln nicht unabhängig, da der Lift einer Regel $r_i \to Y_*$ unter Annahme der Verteilung D_i geschätzt wird, wobei sich D_i aus den Lifts der schon gefundenen Regeln ergibt. Das in [28] vorgeschlagene Verfahren ähnelt der Logistischen Regression.

4.3 Verfahren zur Diskretisierung

4.3.1 Überblick

Es gibt zahlreiche Verfahren, um eine Diskretisierung numerischer Attribute vorzunehmen. Gemeinsam ist allen Verfahren, dass sie den Wertebereich des numerischen Attributes in mehrere Teilintervalle aufteilen. Diese Teilintervalle entsprechen den Ausprägungen des neu erstellten nominalen Attributes. Jede Instanz erhält ihre Ausprägung entsprechend dem Teilintervall, in dem ihr Wert für das numerische Attribut liegt. Die Häufigkeit, mit der eine bestimmte Ausprägung des nominalen Attributes auftritt, ergibt sich aus der Anzahl von Instanzen, deren Wert für das numerischen Attribut im entsprechenden Teilintervall liegt.
Eine Einteilung der Verfahren zur Diskretisierung kann entlang dreier Dimensionen vorgenommen werden: global und lokal, überwacht und unüberwacht sowie statisch und dynamisch [8]. Lokale Verfahren können Diskretisierungen speziell für Teilbereiche des Instanzenraumes vornehmen. Ein Beispiel hierfür ist das Entscheidungsbaumverfahren C4.5 [23]. Ein Pfad im Entscheidungsbaum beschreibt einen Teilbereich des Instanzenraumes; die Diskretisierung erfolgt nur anhand der Beispiele in der Trainingsmenge, die in diesem Teilbereich liegen. Bei globalen Verfahren erfolgt die Diskretisierung jedes numerischen Attributes unabhängig von allen anderen Attributen. So bildet man im einfachsten Fall eine Einteilung des Wertebereiches des numerischen Attributes in k gleich große Teilintervalle für einen gegebenen Parameter k (Equal Interval Binning). Hierbei kann passieren, dass einige Intervalle viele Instanzen enthalten, während in anderen wenige liegen. Dieses hängt von der ursprünglichen Verteilung im Wertebereich des numerischen Attributes ab. Wenn es einen Ausreißer gibt, dessen Wert sehr weit entfernt von allen anderen Werten liegt, kann es durchaus passieren, dass manche Intervalle überhaupt keine Instanzen enthalten. Alternativ kann daher die Einteilung so gewählt werden, dass alle k Intervalle gleich viele benachbarte Instanzen enthalten (Equal Frequency Binning). Die Größe der Intervalle muss nicht gleich sein.
Ein Nachteil dieser unüberwachten Verfahren ist, dass die Partitionierung ohne Berücksichtigung des Zielattributes vorgenommen wird. Die für die Klassifikation von Beispielen wichtigen Infomationen können verloren gehen, indem Instanzen mit Werten, die stark

mit unterschiedlichen Klassen des Zielattributes korreliert sind, demselben Intervall des neu erzeugten nominalen Attributes zugeordnet werden. Die Werte fünf und sechs eines ganzzahligen numerischen Attributes liegen nah zusammen. Ist fünf stark mit der positiven Klasse und sechs stark mit der negativen Klasse des Zielattributes korreliert, sollten die beiden Werte nicht zur selben Ausprägung gehören. Es wurde eine Reihe von Verfahren zur überwachten Diskretisierung entworfen, die versuchen solche Information zu bewahren. Die Verfahren benutzen u.a. den χ^2-Test und die Entropie, um die Intervallgrenzen so zu wählen, dass die in dem numerischen Attribut enthaltenen Informationen über das Zielattribut nicht verloren gehen. Eine Übersicht findet man in [8]. Das im Rahmen dieser Diplomarbeit benutzte überwachte entropiebasierte Verfahren wird im nächsten Abschnitt erläutert.
Alle vorgestellten Verfahren sind statisch, d.h. sie legen die Intervallgrenzen für jedes Attribut unabhängig von den anderen fest. Dynamische Verfahren hingegen versuchen, Abhängigkeiten zwischen den Attributen beim Festlegen der Intervallgrenzen zu berücksichtigen [15].
Ein wichtiger Aspekt der Diskretisierung ist die mögliche Verkürzung der Laufzeit gegenüber dem Lernen mit unveränderten Datensätzen, da nominale Attribute (mit nur wenigen Ausprägungen) für viele Lernverfahren einfacher zu handhaben sind als numerische Attribute.

4.4 Recursive Minimal Entropy Partitioning

4.4.1 Entropie und Informationsgewinn

Die Entropie ist ein Maß aus der Informationstheorie, das von Shannon [30] eingeführt wurde. Sie gibt an, wieviel ‚Zufälligkeit' in einem Signal oder einer Informationsfolge steckt. Häufig spricht man in diesem Zusammenhang auch von Unsicherheit. Die Entropie Ent(I) einer gegebenen Information I über einem Alphabet A ist definiert als

$$Ent(I) = -\sum_{i=1}^{|A|} p_i \cdot log_2 p_i,$$

wobei mit p_i die Wahrscheinlichkeit bezeichnet wird, mit der das i-te Symbol des Alphabets A in der Information I auftritt. Multipliziert man Ent(I) mit der Anzahl der Symbole in der Information I, erhält man die erwartete minimal benötigte Anzahl von Bits, um die Information darzustellen. Als Beispiel stelle man sich einen deutschen Text vor. Die Information ist in diesem Fall eine Folge von Buchstaben. Einige dieser Buchstaben kommen häufiger vor als andere. Z.B ist der Buchstabe 'e' viel häufiger als das 'y'. Die Zeichenfolge ist nicht völlig zufällig. Andererseits ist nie sicher, welcher Buchstabe der nächste ist. Mit der Entropie misst man den Grad dieser Zufälligkeit. Ein weiteres Beispiel ist der wiederholte Wurf einer Münze. Ist die Münze fair, d.h. Kopf und Zahl sind gleich wahrscheinlich, ist der Ausgang des Zufallsexperimentes völlig ungewiss. Die Entropie nimmt dann den maximalen Wert an, nämlich eins. Ist die Münze gezinkt, also z.B. die Wahrscheinlichkeit für Kopf wesentlich höher als die für Zahl, wird die Unsicherheit über den Ausgang geringer. Die Entropie wird kleiner und nähert sich immer mehr null, je sicherer der Ausgang des Zufallsexperimentes ist. Für eine Menge von Beispielen lässt sich die Entropie bezüglich des Zielattributes bestimmen. Die Ausprägungen

des Zielattributes bilden das Alphabet A. Nach Ermittlung der relativen Häufigkeiten für jede Ausprägung lässt sich die Entropie berechnen. Analog zum Münzwurf wird die Entropie maximal, wenn alle Klassen in der Beispielmenge gleich häufig sind und beträgt null, wenn es nur eine Klasse gibt.
Der Informationsgewinn ist ein Kriterium, das von Entscheidungsbaumlernverfahren eingesetzt wird. Die Beispielmenge wird dabei anhand der Ausprägungen eines Attributes in mehrere Teilmengen aufgespaltet. Für jede dieser Teilmengen lässt sich nun wieder die Entropie berechnen. Die Summe dieser Entropien, jeweils gewichtet mit der Anzahl der Beispiele in der Teilmenge, ergibt einen neuen Entropiewert für die gesamte Beispielmenge. Die Differenz zwischen altem und neuem Entropiewert bezeichnet man als Informationsgewinn.

4.4.2 Algorithmus

Das Grundprinzip dieses globalen überwachten Verfahrens [10] zur Diskretisierung ist, für das betrachtete numerische Attribut einen Teilungspunkt zu finden, anhand dessen Wert die Beispielmenge in zwei Teilmengen aufgeteilt wird: Jede Instanz, deren Attributwert kleiner ist als der Wert des Teilungspunktes gehört zum ersten, der Rest zum zweiten Teilintervall. Zur Bestimmung des Teilungspunktes testet der Algorithmus alle möglichen Werte des Attributes und wählt den aus, für den der Informationsgewinn am größten ist. Für eine Menge T von Instanzen bezeichnet Ent(T) die Entropie in den Ausprägungen des Zielattributes. Mit K wird ein Kandidat für einen Teilungspunkt bezeichnet, der die Instanzenmenge in die Teilmengen T_1 und T_2 aufteilt. Für einen Teilungspunktkandidaten K, ein Attribut A und einer Menge T von Instanzen berechnet der Algorithmus

$$E(A;K;T) = \frac{|T_1|}{|T|}Ent(T_1) + \frac{|T_2|}{|T|}Ent(T_2).$$

Anhand des besten Teilungspunktes K_{min} erfolgt die Aufteilung in zwei Teilintervalle, auf die das Verfahren rekursiv angewendet wird, bis die auf dem Prinzip der minimalen Beschreibungslänge basierende Abbruchbedingung zutrifft. Diese ist definiert als

$$Gain(A;K;T) < \frac{log_2(|T|-1)}{|T|} + \frac{\Delta(A;K;T)}{|T|}.$$

Hierbei steht T für die gerade betrachtete Beispielmenge und

$$Gain(A;K;T) = Ent(T) - E(A;K;T),\ sowie$$

$$\Delta(A;K;T) = log_2(3^k - 2) - [k \cdot Ent(T) - k_1 \cdot Ent(T_1) - k_2 \cdot Ent(T_2)].$$

Die Anzahl der verschiedenen Klassen in der Instanzenmenge S_i wird mit k_i bezeichnet. Die Rekursion wird abgebrochen, wenn die Verkleinerung der Unsicherheit nicht mehr groß genug ist. Damit werden die Bereiche des numerischen Attributes, die eine hohe Entropie haben, wo also alle Ausprägungen des Zielattributes möglichst gleich häufig vertreten sind, sehr fein partitioniert, während Bereiche mit geringer Entropie nur grob partitioniert werden.

Das Verfahren wurde für die Lernumgebung YALE [21] implementiert. Der erste Test erfolgte mit einem synthetisch erzeugten Datensatz bestehend aus 200.000 Beispielen. Der

	Adult	Quantenphysik
J48 (ohne)	67.89	86.17
J48 (RMEP)	69.53	86.22
J48 (EFB5)	67.62	82.99
J48 (EFB10)	68.28	82.81

Tabelle 4.3: Accuracy von J48 auf Datensätzen ohne Diskretisierung (ohne), mit Recursive Minimal Entropy Partitioning (RMEP), Equal Frequency Binning mit 5 Bins (EFB5) und Equal Frequency Binning mit 10 Bins (EFB10).

Instanzenraum bestand dabei aus 20 numerischen Attributen mit ganzzahligem Wertebereich von 0 bis 19 und einen nominalen Zielattribut mit zwei Klassen. Hierbei wurde nur für das erste Attribut eine Korrelation der Werte 0 und 1 mit dem Zielattribut erzeugt. Als Ergebnis lieferte das Recursive Minimal Entropy Partitioning nur 0 und 1 als Teilungspunkte für das erste Attribut. Aus allen anderen numerischen Attributen wurden nominale Attribute mit einer Ausprägung erzeugt; sie wurden korrekt als uninteressant erkannt. Danach wurden die Verfahren auf die später noch verwendeten Adult- und Quantenphysik-Datensätze angewandt und für die diskretisierten Daten mit dem J48-Verfahren ein Entscheidungsbaum erzeugt. Die Ergebnisse sind in Tabelle 4.4.2 zusammengefasst. Eine nähere Beschreibung der Datensätze befindet sich in Kapitel 6.2. Das Minimal Entropy Partitioning schneidet in allen Fällen gut ab. Zusammen mit der Eigenschaft keinen Parameter zu brauchen, gab dieses den Ausschlag bei allen weiteren Experimenten nur noch dieses Verfahren zur Diskretisierung zu verwenden. Ein erwähnenswerter Nachteil des Verfahrens ist die Laufzeit. Das Equal Frequency Binning war in allen Experimenten schneller.

5 Iterating GSS

In diesem Kapitel ist der Iterating GSS Algorithmus beschrieben. Er kombiniert den Generic Sequential Sampling Ansatz mit Knowledge-Based Sampling um möglichst schnell eine kompakte und aussagekräftige Menge von Subgruppen bzw. Regeln zu finden. Außerdem nimmt er viele Verbesserungen am GSS Algorithmus vor. Um Knowledge-Based Sampling einsetzen zu können, bedurfte der GSS Algorithmus einer Erweiterung, um mit Gewichten umgehen zu können. Es musste das Ziehen von Beispielen nach einer durch die Gewichte simulierten Verteilung mit Hilfe der Verwerfungsmethode ermöglicht werden. Desweiteren wurde zur Verkürzung der Laufzeit die Häufigkeit, mit der die Berechnung von Nutzen und Konfidenzintervall für die Hypothesen durchgeführt wird, verringert. Auf die Problematik der Verwendung der Approximation durch die Normalverteilung bei Berechnung der Konfidenzintervalle wurde bereits in Kapitel 4.1 kurz eingegangen. Die Problematik wird in diesem Kapitel vertieft und eine Lösung vorgeschlagen. Eine Beschreibung der einzelnen Verbesserungen des GSS Algorithmus erfolgt im Kapitel 5.1. Das Problem der großen Hypothesenräume wurde angegangen, indem das Durchsuchen des Hypothesenraumes nicht als Ganzes sondern sukzessive von einfachen zu komplexen Hypothesen erfolgte. Dazu bedurfte es zum einen der Möglichkeit, den Hypothesenraum nach Komplexität geordnet aufzuzählen (Kapitel 5.2). Zum anderen war ein Kriterium erforderlich, um zu entscheiden, wann die Suche für eine bestimmte Komplexität abgebrochen werden kann (Kapitel 5.3). Um nicht zu viele Hypothesen von großer Komplexität zu erzeugen, bedurfte es weiterhin einer Methode, um solche Teile des Hypothesenraumes zu verwerfen, die keine guten Hypothesen enthalten können (Kapitel 5.4). Der Vorgang wird als *Pruning* bezeichnet. Neu an diesem Ansatz ist die Kombination aus probabilistischer Subgruppenentdeckung durch den GSS Algorithmus und der Suche in strukturierten Hypothesenräumen [34]. Im weiteren Verlauf des Kapitels wird der Iterating GSS Algorithmus zusammen mit seinen Variationsmöglichkeiten angegeben und erläutert (Kapitel 5.5 und 5.6). Am Ende des Kapitels erfolgt eine genauere Beschreibung der Implementierung der zum Durchsuchen des Hypothesenraumes und fürs Pruning eingesetzten Methoden (Kapitel 5.7).

5.1 Modifikationen des GSS Algorithmus

Der GSS Algorithmus wurde an einigen Stellen modifiziert, um die Kombination mit Knowledge-Based Sampling zu ermöglichen und Nachteile zu vermeiden, die in der Standardversion auftreten.

5.1.1 Verwerfungsmethode und direkte Verwendung von Beispielgewichten

Um Knowledge-Based Sampling einsetzen zu können bedarf es der Modifikation des GSS Algorithmus. In Schritt 3a wird zufällig ein Beispiel aus der Trainingsmenge T gezogen. Da die Beispiele im Gegensatz zu dem im Kapitel 4.1 vorgestellten Algorithmus nicht mehr ausschließlich gleichverteilt gezogen werden, ist die Einführung von Gewichten für

- **while** ($true$) **do**
- Ziehe zufällig mit Zurücklegen eine Instanz x_i aus T.
- Erzeuge eine Zufallszahl $z \in [0, 1]$.
- **if** ($w(x_i) \geq z$) **then**
  - Füge x_i in Q_i ein.
  - **break**.

Abbildung 5.1: Modifikation von Schritt 3a des Generic Sequential Sampling Algorithmus mit der Verwerfungsmethode

jedes Beispiel der Trainingsmenge nötig, um die aktuelle Verteilung zu repräsentieren. Um Beispiele unter dieser Verteilung zu ziehen, nutzt der GSS Algorithmus die in Kapitel 4.2.3 vorgestellte Verwerfungsmethode. Dazu muss Schritt 3a des GSS Algorithmus (Abbildung 4.1) durch die Zeilen in Abbildung 5.1.1 ersetzt werden.

Die Verwendung der Verwerfungsmethode bedarf der Normierung der Beispielgewichte auf das Intervall [0,1] nach jeder Iteration (Kapitel 4.2.3). Dazu wird das maximale Gewicht bestimmt und alle Beispielgewichte hierdurch dividiert. Nach vielen Iterationen tendieren die Beispielgewichte dazu, klein zu werden. Dieses hat das Verwerfen vieler Beispiele und die häufige Wiederholung des Zufallsexperimentes ‚Ziehen eines zufälligen Beispiels mit Zurücklegen' zur Folge. Eine Alternative bietet die Verwendung der Gewichte ohne eine Normierung auf das Intervall [0,1]. In diesem Fall wird kein Beispiel verworfen und die veränderte Verteilung direkt durch die Gewichte simuliert. Beispiele mit einem Gewicht größer als eins zählen verstärkt, während Beispiele mit Gewicht kleiner eins von verminderter Wichtigkeit sind. Die Betrachtung der Bewertung von abgedeckten und korrekt vorhergesagten Beispielen für eine Regel macht dieses deutlich. Im Gegensatz zur Verwerfungsmethode werden nicht die abgedeckten und die korrekt vorhergesagten Beispiele gezählt, sondern das abgedeckte und das korrekt vorhergesagte Beispielgewicht. Für die Performanz einer Regel ist wichtig, Beispiele mit großem Gewicht abzudecken und korrekt vorherzusagen. Dagegen zählt bei der Verwerfungsmethode jedes Beispiel mit einem Gewichte von eins. Die Wichtigkeit eines Beispiels wird durch die geringe Wahrscheinlichkeit für ein Verwerfen repräsentiert. Die Regel mit dem größten unnormierten Gewicht erhält durch die Normierung ein Gewicht von eins und wird deshalb nie verworfen. Bei direkter Verwendung der Gewichte kann Schritt 3a des GSS Algorithmus unverändert übernommen werden. Lediglich die Einführung von Beispielgewichten ist nötig. Damit entfällt das zweite Zufallsexperiment ‚Erzeugen einer Zufallszahl aus dem Intervall [0,1]'. Zusammen mit der Tatsache, dass keine Beispiele verworfen werden, resultiert dieses meist in einer geringeren Laufzeit als bei Verwendung der Verwerfungsmethode.

5.1.2 Veränderung der Schrittgröße

Zum besseren Verständnis dieser Modifikation des GSS Algorithmus wird zunächst auf die Aufteilung der Irrtumswahrscheinlichkeit δ eingegangen. Mit ϵ wird der maximale

Fehler und mit |H| die Anzahl der verbliebenen Regeln bezeichnet.
Der GSS Algorithmus (Abbildung 4.1) verwendet eine Hälfte der zur Verfügung stehenden Irrtumswahrscheinlichkeit δ, um Konfidenzintervalle für den Nutzen aller verbliebenen Regeln in der Abbruchbedingung der Schleife in Schritt 3 zu schätzen. Diese muss auf alle verbliebenen Regeln aufgeteilt werden. Es wird ein Konfidenzintervall um den geschätzten Nutzen berechnet, in dem der wahre Nutzen jeder Regel mit Wahrscheinlichkeit $\frac{\delta}{2|H|}$ liegt. In Summe ergibt sich dann eine Irrtumswahrscheinlichkeit von $\frac{\delta}{2}$. Die andere Hälfte von δ wird innerhalb der Schleife in Schritt 3d verwendet, um nach jedem gezogenen Beispiel für alle Regeln ein Konfidenzintervall um deren geschätzten Nutzen zu berechnen. Auf dieser Basis werden Regeln verworfen oder der Lösung hinzugefügt. Deshalb muss $\frac{\delta}{2}$ zusätzlich auf die maximal benötigte Beispielanzahl M aufgeteilt werden. Das Konfidenzintervall um den geschätzten Nutzen muss den wahren Wert mit Wahrscheinlichkeit $\frac{\delta}{2|H|M}$ enthalten. Der Wert von M hängt von der verwendeten Nutzenfunktion, dem Maximalfehler ϵ und von δ ab (Tabelle 4.1). Je nach Wahl dieser Parameter kann M sehr groß werden, wodurch die Irrtumswahrscheinlichkeit in kleine Teile aufgespaltet wird. Damit werden die Konfidenzintervalle größer und es werden mehr Beispiele benötigt bevor der Algorithmus terminiert. Hinzu kommt, dass durch die häufigen Berechnungen ein hoher Rechenaufwand betrieben wird. Durch die gewählte Aufteilung der Irrtumswahrscheinlichkeit wird davon ausgegangen, dass die Schätzungen für den wahren Nutzen in jedem Durchlauf durch die Schleife unabhängig voneinander sind. Die Schätzungen würden auch gelten, wenn die Beispiele nach jedem Durchlauf durch die Schleife komplett neu gezogen werden. Da aber stets nur ein neues Beispiel gezogen wird, sind die Schätzungen nicht unabhängig voneinander und die Aufteilung der Irrtumswahrscheinlichkeit konservativ. Schritt 3d sollte daher nicht in jedem Schleifendurchlauf ausgeführt werden.
Der GSS Algorithmus wurde um den Parameter Schrittgröße erweitert. Damit ist es möglich, die Anzahl von Beispielen anzugeben, die jeweils gezogen werden muss, bevor Schritt 3d erneut ausgeführt wird. Beispielsweise führt der Algorithmus Schritt 3d bei einer Schrittgröße von 1000 jeweils nach 1000 gezogenen Beispielen durch.

5.1.3 Verwendung der Approximation durch die Normalverteilung

Der GSS Algorithmus benutzt bei der Berechnung der Konfidenzintervalle Schranken, die mit Hilfe der Normalverteilung berechnet wurden, anstatt der durch die Hoeffding Ungleichung gelieferten Schranken. Der Grund dafür ist, dass bei Verwendung der Normalverteilung bessere (kleinere) Konfidenzintervalle gefunden werden. Als Argument dient der zentrale Grenzwertsatz [9] und die Tatsache, dass der Algorithmus keine Lösungen ausgibt bevor die Stichprobengröße einige hundert Beispiele beträgt. Der zentrale Grenzwertsatz besagt, dass sich die Verteilung der (normierten) Summe unabhängiger und identisch verteilter Zufallsvariablen immer mehr der Normalverteilung annähert je größer die Anzahl der Zufallsvariablen ist. Dieses Argument soll für die verschiedenen Nutzenfunktionen näher untersucht werden.
Als erste Nutzenfunktion wird die Accuracy betrachtet. Für Regel R sei X_i die Zufallsvariable, die angibt, ob R das i-te Beispiel korrekt vorhersagt:

$$X_i = \begin{cases} 0, & falls\ R\ das\ i-te\ Beispiel\ falsch\ vorhersagt, \\ 1, & falls\ R\ das\ i-te\ Beispiel\ korrekt\ vorhersagt. \end{cases}$$

Die Accuracy ist definiert als die Wahrscheinlichkeit für das Ereignis, dass die Regel R eine korrekte Vorhersage macht. Sie ergibt sich als normierte Summe ($\frac{1}{n}\sum_{i=1}^{n} X_i$ bei n Zufallsexperimenten) der unabhängig und identisch verteilten Zufallsvariablen X_i. Damit ist sie binomialverteilt, lässt sich aber durch die Normalverteilung approximieren, falls die Anzahl von gezogenen Beispielen groß genug ist. Ab wievielen Beispielen die Approximation gut ist, hängt von der Wahrscheinlichkeit ab, mit der das betrachtete Ereignis eintritt. Bei dem Ereignis handelt es sich in diesem Fall um die korrekte Vorhersage eines Beispiels durch die Regel. Dessen Wahrscheinlichkeit entspricht der Accuracy. Allgemein wird die Wahrscheinlichkeit für das betrachtete Ereignis mit p bezeichnet.

In Abbildung 5.2 ist für unterschiedliche Beispielanzahlen und Werte von p die Güte der Approximation dargestellt. Die Approximation ist umso besser je näher der Wert von p an 0.5 liegt. Ab einer Beispielanzahl von 100 ist die Approximation auch für extreme Werte von p hinreichend gut. Da das Ziehen eines Beispiels bei der Accuracy immer einem Zufallsexperiment entspricht, ist in diesem Fall ab Beispielanzahlen von 100 die Approximation durch die Normalverteilung hinreichend gut.
Für Nutzenfunktionen wie Weighted Relative Accuracy, Squared und Binomial ist dieses Vorgehen problematisch, da in deren Berechnung Coverage und Precision eingehen. Diese sind ebenfalls binomialverteilt. Bei der Berechnung der Coverage ist jedes Ziehen eines Beispiels ein Zufallsexperiment. Dieses gilt nicht bei der Precision. Hier entspricht die Anzahl von durchgeführten Zufallsexperimenten der Anzahl der durch die Regel abgedeckten Beispiele. Bei Regeln, die für wenige Beispiele anwendbar sind, ist ein Rückschluss von der Anzahl der gezogenen Beispiele auf die Güte der Approximation der Binomialverteilung durch die Normalverteilung nicht möglich. Dieses gilt insbesondere, wenn die Precision der Regel extreme Werte nahe eins oder null annimmt. Der GSS Algorithmus wurde deshalb dahingehend erweitert, dass er bei der Berechnung der Konfidenzintervalle erst die Normalverteilung eingesetzt, wenn die Anzahl der durch eine Regel abgedeckten Beispiele einen vorgegebenen Wert erreicht. Davor werden die mit Hilfe der Hoeffding-Ungleichung gewonnenen Schranken zur Berechnung verwendet. Die Verwendung der Normalverteilung hängt nicht direkt von der Anzahl der gezogenen Beispiele ab.

5.2 Erzeugung des Hypothesenraumes

Ein Parameter des GSS Algorithmus ist der verwendete Hypothesenraum. Dieser muss nicht zwangsläufig aus allen Hypothesen bestehen, die für den gegebenen Instanzenraum erzeugt werden können. Es ist möglich, den GSS Algorithmus wiederholt mit einem bestimmten Teil des Hypothesenraumes auszuführen. Dieses kann man nutzen, indem der GSS Algorithmus zuerst nur mit Hypothesen von geringer Komplexität aufgerufen wird und der Übergang zu komplexeren Hypothesen dann erfolgt, wenn bezüglich der Nutzenfunktion keine guten Hypothesen mehr gefunden werden. Ein solches Vorgehen ist sinnvoll und nötig, da selbst für nicht unendliche Instanzenräume häufig die Menge aller Hypothesen so groß ist, dass diese nicht mehr effizient zu handhaben sind. Der verwendete Hypothesenraum besteht ausschließlich aus Regeln mit konjunktiv verknüpften Literalen. Der Einfachheit halber wird für sie stets die Abkürzung Regeln verwendet. Um Regeln von geringer zu hoher Komplexität zu erzeugen, bedarf es einer Definition der Komplexität einer Regel.

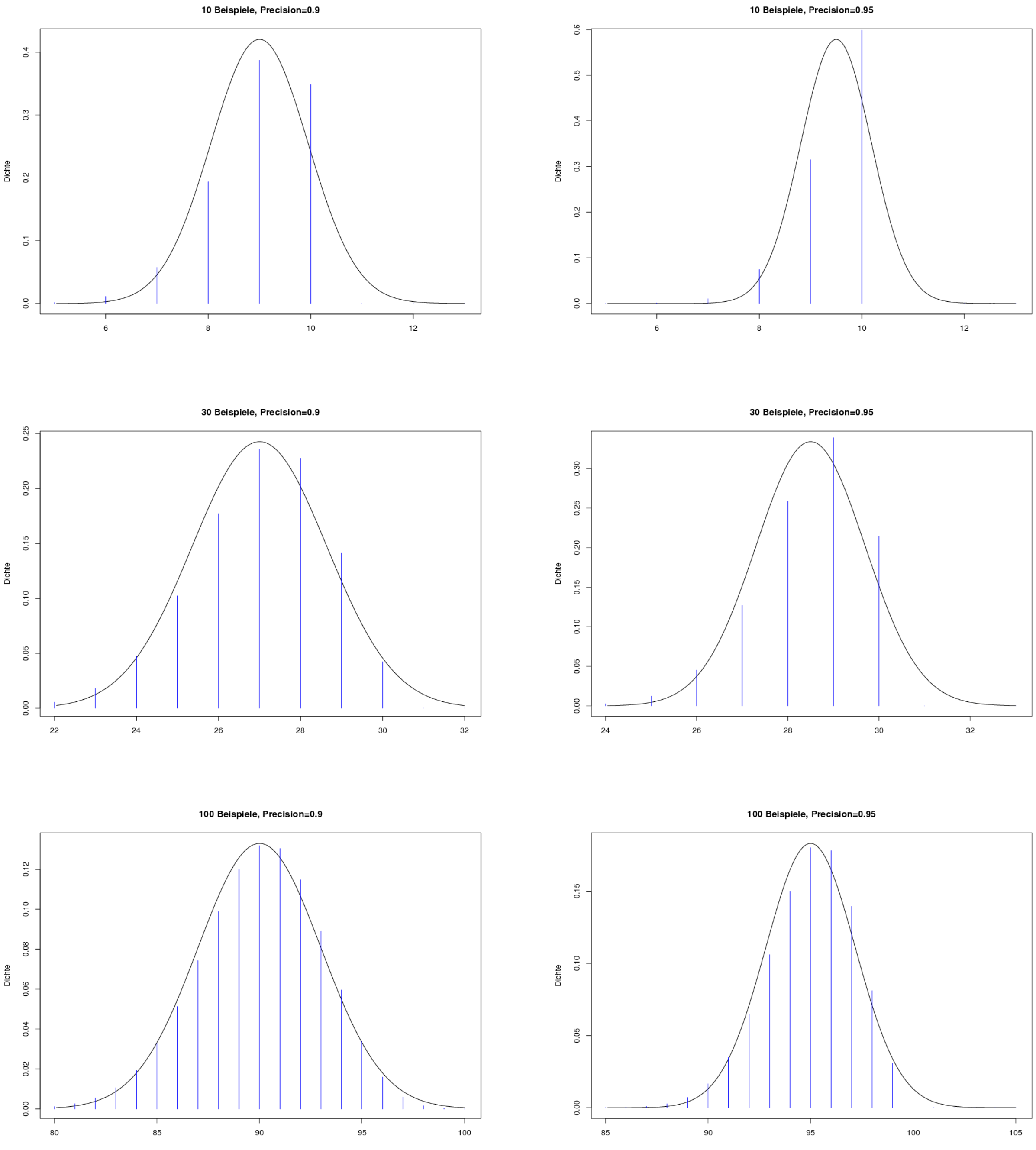


Abbildung 5.2: Approximation von Binomialverteilungen mit verschiedenen Parametern durch Dichtekurven der Normalverteilung.

Definition 17. *(Komplexität einer Regel)*
Für eine Regel R ist die Komplexität c(R) definiert als die Anzahl von Literalen in der Prämisse der Regel. Die Komplexität einer Regel wird auch mit Regellänge bezeichnet.

Es wird eine Möglichkeit benötigt, Regeln unter Vermeidung der Erzeugung von Duplikaten aufzuzählen. In Anlehnung an [34] werden Verfeinerungen verwendet, die durch Anhängen eines weiteren Literals an die Prämisse einer Regel erzeugt werden. Die Attribute des Instanzenraumes werden bei eins beginnend indexiert. Gleiches gilt für die Ausprägungen eines Attributes. A_i bezeichnet das i-te Attribut und a_{ij} die j-te Ausprägung des i-ten Attributes. Für Literale $\mathcal{P}' = (A_k = a_{kl})$ und $\mathcal{P} = (A_i = a_{ij})$ lässt sich eine totale Ordnung definieren.

Definition 18. *(Totale Ordnung von Literalen)*

$$\mathcal{P}' = (A_k = a_{kl}) > \mathcal{P} = (A_i = a_{ij}) :\Leftrightarrow (k > i) \textit{ oder } (k = i \textit{ und } l > k).$$

So ist es möglich, die direkten Nachfolger einer Regel zu bestimmen.

Definition 19. *(Direkte Nachfolger einer Regel)*
Für eine Regel $R = \mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_i \Rightarrow \mathcal{K}$ sind die direkten Nachfolger definiert als alle Regeln $R' = \mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_i \wedge \mathcal{P}_k \Rightarrow \mathcal{K}$ mit $P_k > P_i$. Eine Regel $\tilde{R}$ ist Nachfolger einer Regel R, wenn $\tilde{R}$ durch eine Kette von direkten Nachfolgern mit R verbunden ist.

Die direkten Nachfolger enstehen, indem an die Prämisse alle Literale angehängt werden, die gemäß der definierten Ordnung größer sind als das letzte Literal der betrachteten Regel. Dieser Vorgang wird mit *Verfeinerung* bezeichnet. Wiederholte Anwendung der Verfeinerung liefert die weiteren Nachfolger der Regel.
So steht eine Möglichkeit zur Verfügung, alle Regeln strukturiert nach ihrer Komplexität unter Vermeidung der Erzeugung von Duplikaten aufzuzählen. Zuerst werden alle Regeln der Komplexität eins erzeugt. Alle Regeln der Komplexität zwei erhält man daraus durch Generierung der direkten Nachfolger. Dieses Vorgehen kann beliebig wiederholt werden, um Regeln mit höherer Komplexität zu erzeugen. Dabei sinkt die Coverage der Verfeinerung im Vergleich zur Coverage der ursprünglichen Regel. Dieses wird beim Pruning 5.4 ausgenutzt.

5.3 Suche im Hypothesenraum

Um den Hypothesenraum aller Regeln sinnvoll von geringer zu hoher Komplexität zu durchsuchen, ist es nötig ein Kriterium zu finden, mit dem entschieden werden kann, wann eine Erhöhung der Komplexität nötig ist. Eine Möglichkeit ist die Erhöhung vorzunehmen, wenn die beste im Hypothesenraum der aktuellen Komplexität gefundene Regel von schlechter Qualität ist. Es bietet sich an, die Güte einer Regel R anhand des Nutzen und der Größe des Konfidenzintervalles zu beurteilen.
Im einfachsten Fall wird ein *Mindestnutzen* q_{min} vorgegeben. Für die beste Regel R und die Nutzenfunktion q wurden aus einer Stichprobe der Größe m ein empirischer Nutzen $\hat{q}(R)$ geschätzt und ein Konfidenzintervall $E(R, q, \delta, m)$ berechnet. Gilt $\hat{q}(R) + E(R, q, \delta, m) < q_{min}$ wird die Regel als nicht *nützlich* eingestuft, da sie mit hoher Wahrscheinlichkeit nicht mehr den Minimalnutzen erreichen kann, selbst wenn der wahre Nutzen am oberen Ende des Konfidenzintervalles liegt.

Eine andere Möglichkeit beruht auf der Annahme, dass die beste Regel R nicht ausgegeben wird, bevor die Stichprobe eine große Anzahl m von Beispielen umfasst. Es wird der Wert $\hat{q}(R) - E(R, q, \delta, m)$ berechnet, also die Differenz aus Nutzen und Konfidenzintervall von R. Ergibt sich dabei ein Wert kleiner oder gleich q_{min}, kann dieses zum einen dadurch bedingt sein, dass die Regel nur wenige Beispiele abdeckt und somit das Konfidenzintervall auch nach vielen Beispielen noch groß ist. Zum anderen besteht die Möglichkeit, dass die Regel zwar viele Beispiele abdeckt und das Konfidenzintervall klein ist, sie aber einen so geringen Nutzen hat, dass sich durch Abziehen des Konfidenzintervalles ein Wert kleiner oder gleich q_{min} ergibt. Beide Fällen werden als Zeichen für eine Regel mit geringem wahren Nutzen angesehen. Eine alternative Vorstellung dieses Kriteriums ist, dass die gefundene Regel mit hoher Wahrscheinlichkeit einen wahren Nutzen größer als q_{min} haben muss, um nützlich zu sein. Der schlimmsten Fall tritt ein, wenn der wahre Nutzen am unteren Ende des Konfidenzintervalles liegt. Wird beispielsweise q_{min}=0 gewählt, werden alle Regeln, die einen positiven Nutzen garantieren, als nützlich eingestuft.
Desweiteren kann die Güte einer Regel nur auf Basis des geschätzten Nutzens unter Vernachlässigung der berechneten Konfidenzintervalle beurteilt werden. In diesem Fall gilt eine Regel R als nützlich, falls $\hat{q}(R) > q_{min}$ ist.
Als weiteres Kriterium für die Qualität einer Regel kann die Anzahl der benötigten Beispiele bis zur Ausgabe benutzt werden. Der GSS Algorithmus benötigt nur eine kleine Stichprobe, wenn die guten Regeln sich bezüglich der Nutzenfunktion stark von den schlechten unterscheiden. Im Umkehrschluss bedeutet eine große Anzahl von Beispiele in der Stichprobe, dass sich die Regeln bezüglich des Nutzens nicht stark voneinander unterscheiden. Wächst die Stichprobengröße, nachdem bereits einige Regeln gefunden wurden, bis zur Ausgabe der nächste Regel im Vergleich zu vorher stark an, kann dieses so interpretiert werden, dass schon alle guten Regeln der betrachteten Komplexität gefunden sind. Die Unterscheidung von guten und schlechten Regeln für verschiedene Datensätze ist unterschiedlich schwer. Außerdem kommt es pro gefundener Regel meist zu einem gewissen Anstieg der Stichprobengröße. Als Vergleichswert wird die durchschnittlich benötigte Stichprobengröße aller bisher gefundenen Regeln einer Komplexität verwendet. Die Stichprobengröße bis zur Ausgabe der nächsten Regel wird mit diesem Wert verglichen. Weicht die Stichprobengröße der neuen Regel um mehr als einen gegebenen Faktor f (*Beispielfaktor*) von der durchschnittlichen Stichprobengröße ab, wird die Regel als nicht nützlich bewertet.

5.4 Pruning

Mit einem Kriterium zur Bewertung der Nützlichkeit einer Regel ist es möglich, den Hypothesenraum von geringer zu hoher Komplexität zu durchsuchen. Allerdings wird der Hypothesenraum schon bei Regeln der Länge drei oder vier so groß, dass er nicht mehr effizient zu handhaben ist. Es ist meist nicht nötig alle Regeln einer bestimmten Komplexität zu betrachten, wenn schon Informationen über die Regeln einer niedrigeren Komplexität bekannt sind und ein Mindestnutzen q_{min} vorgegeben ist. Der Grund liegt darin, dass durch die Art der Erzeugung der direkten Nachfolger vom Nutzen einer Regel Rückschlüsse auf den Nutzen ihrer direkten Nachfolger gezogen werden können. Es ist möglich eine obere Schranke für den Nutzen aller direkten Nachfolger einer Regel zu be-

rechnen. Diese Schranke gilt nicht nur für die direkten sondern für alle Nachfolger. Ist sie kleiner als der Mindestnutzen q_{min}, ist die Betrachtung der Nachfolger nicht nötig. Es ist zu beachten, dass aufgrund der Natur des GSS Algorithmus die Schranke nur mit einer vorgegebenen Wahrscheinlichkeit gültig ist. Der Nutzen, der Basis für die Berechnung ist, wird aufgrund einer Stichprobe geschätzt. Es kann nicht garantiert werden, dass er dem wahren Nutzen entspricht. Im Folgenden wird beschrieben, wie die obere Schranke berechnet werden kann.
Die direkten Nachfolger einer Regel werden erzeugt, indem ein weiteres Literal an die Prämisse angehängt wird. Durch diese Verfeinerung wird eine Regel spezieller. Ist R' ein direkter Nachfolger der Regel R, deckt R' nur eine Teilmenge der Beispiele ab, die von R abgedeckt werden. Beispielsweise handelt es sich bei der Menge der Kunden einer Bank, die älter als 65 sind und mehr als 10.000 Euro pro Monat verdienen, um eine Teilmenge aller Kunden, die älter als 65 sind. Zur Veranschaulichung der möglichen Entwicklungen bei Verfeinerung einer Regel wird eine grafischen Darstellung verwendet. Eine Regel teilt die Menge der Beispiele in vier Klassen ein. Es handelt sich um die abgedeckten positiven Beispiele (*True Positives*), die abgedeckten negativen Beispiele (*False Positives*), die nicht abgedeckten positiven Beispiele (*False Negatives*) und die nicht abgedeckten negativen Beispiele (*True Negatives*). Der *Coverage Space* [14] erlaubt den Vergleich verschiedener Regeln anhand der abgedeckten positiven und negativen Beispiele. Dazu werden die Regeln in ein zweidimensionales Koordinatensystem eingezeichnet. Auf den beiden Achsen werden die abgedeckten positiven und negativen Beispiele abgetragen. $|Y_+|$ bezeichnet die Anzahl aller positiven Beispiele, während $|Y_-|$ die Anzahl aller negativen Beispiele bezeichnet. Jede Regel kann zwischen null und $|Y_+|$ positive und zwischen null und $|Y_-|$ negative Beispiele abdecken. In Abbildung 5.3 ist der Coverage Space für eine Trainingsmenge mit $|Y_+|$ positiven und $|Y_-|$ negativen Beispielen dargestellt. Auf der diagonalen Linie liegen alle Regeln, die gleich viele positive und negative Beispiele abdecken.

Wie kann man Rückschlüsse vom Nutzen einer Regel auf den Nutzen der Nachfolger ziehen? Wird eine Regel R zu einem direkten Nachfolger R' verfeinert, verringert sich die Anzahl der abgedeckten Beispiele. Daher liegen Verfeinerungen $R^{'}$ einer Regel R in der grafischen Darstellung des Coverage Space stets unterhalb oder links von R. In Abbildung 5.3 sind einige mögliche Resultate der Verfeinerung eingezeichnet. Es kann sowohl die Anzahl der abgedeckten positiven Beispiele als auch die Anzahl der abgedeckten negativen Beispiele kleiner werden. Für die Nutzenfunktionen Accuracy, Weighted Relative Accuracy, Squared und Binomial steigt der Nutzen monoton, wenn durch die Verfeinerung nur die Anzahl der abgedeckten negativen Beispiele kleiner wird [35]. Deshalb erreicht derjenige direkte Nachfolger $R^{'}$ einer Regel R den bestmöglichen Nutzen, der nach der Verfeinerung keine negativen Beispiele mehr abgedeckt. In Abbildung 5.3 ist dieser Punkt mit R* gekennzeichnet. Damit erhält man allerdings noch nicht die gesuchte obere Schranke für den Nutzen aller Nachfolger einer Regel. Der Nutzen wird aufgrund einer Stichprobe geschätzt. Es bedarf daher der Berechnung eines Konfidenzintervalles um den geschätzten Nutzen, in dem der wahre Nutzen mit vorgegebener Irrtumswahrscheinlicheit δ liegt. Bei optimistischer Schätzung ergibt sich der beste Nutzen einer Regel, wenn der wahre Nutzen am oberen Ende des berechneten Konfidenzintervalles liegt. Mit $R^{'}_{opt}$ wird der direkte Nachfolger einer Regel R bezeichnet, der keine negativen Beispiele abdeckt, aber alle positiven von R abgedeckten Beispiele. Die obere Schranke für den Nutzen der

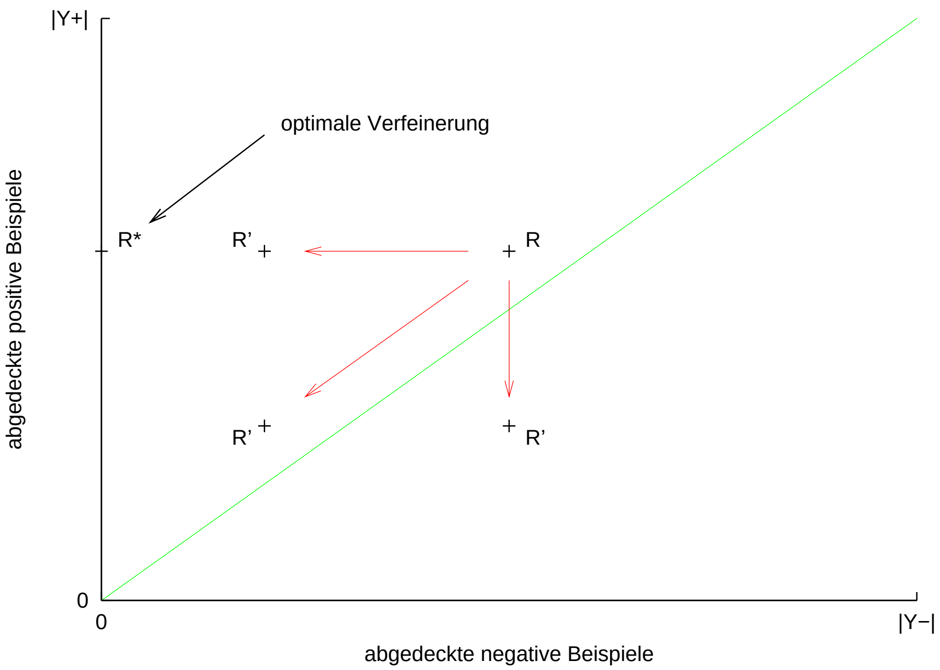


Abbildung 5.3: Grafische Darstellung des Coverage Space zur Verdeutlichung der möglichen Resultate der Verfeinerung einer Regel

Nachfolger einer Regel R wird berechnet, indem die Summe aus Nutzen und Größe des Konfidenzintervalles[1] von R'_{opt} gebildet wird. Da eine obere Schranke gesucht wird kann angenommen werden, dass ein solcher direkten Nachfolger existiert.
Um die Korrektheit der Schranke zu zeigen muss zum einen bewiesen werden, dass kein direkter Nachfolger R' der Regel R existiert, für den die Summe aus Nutzen und Größe des Konfidenzintervalles größer ist als für R'_{opt}. Zum anderen muss gezeigt werden, dass die Summe aus Nutzen und Größe des Konfidenzintervalles monoton kleiner wird, wenn durch Verfeinerungen die Nachfolger von R'_{opt} erzeugt werden.
Wie bereits erwähnt ist der Nutzen der Regel R'_{opt} optimal unter allen Verfeinerungen von R. Ebenso können die Nachfolger von R'_{opt} keinen besseren Nutzen haben, da durch weitere Verfeinerung nur weniger positive Beispiele abgedeckt werden. Eine größere Summe aus Nutzen und Größe des Konfidenzintervalles als für R'_{opt} kann sich daher nur für einen Nachfolger R' der Regel R ergeben, der ein größeres Konfidenzintervall als R'_{opt} hat.
Bei der Berechnung des Konfidenzintervalles ist zu beachten, dass durch die Annahme, dass keine negativen Beispiele mehr abgedeckt werden, die Regel R'_{opt} eine Precision von eins hat. Da eine obere Schranke gesucht wird und die Precision nicht größer als eins werden kann, muss sie bei der Berechnung des Konfidenzintervalles nicht mehr berücksichtigt werden. Das hat Konsequenzen für die Nutzenfunktionen Weigted Relative Accuracy, Squared und Binomial, die Coverage und Bias multiplizieren. Dieses wird am

[1] Die Größe des Konfidenzintervalles bezieht sich hier nur auf die maximal mögliche Abweichung des Nutzens nach oben. Genauer müßte man von halber Größe des Konfidenzintervalles sprechen.

Beispiel der Weighted Relative Accuracy ersichtlich:

$$WRACC(r \rightarrow Y_*) := COV(r \rightarrow Y_*) \cdot BIAS(r \rightarrow Y_*)$$
$$= COV(r \rightarrow Y_*) \cdot (PREC(r \rightarrow Y_*) - Pr[Y_*])$$
$$= COV(r \rightarrow Y_*) \cdot (1 - Pr[Y_*]).$$

Da die apriori Wahrscheinlichkeit konstant ist, genügt die Berechnung des Konfidenzintervalles für die Coverage. Die Coverage wird als Durchschnitt über eine Instanznutzenfunktion berechnet (Instance-Averaging). Bei Verwendung der entsprechenden Schranken ergibt sich für die Nutzenfunktionen Binomial und Squared ein besseres (kleineres) Konfidenzintervall als mit den Standardschranken. Accuracy und Weighted Relative Accuracy verwenden ohnehin diese Schranken. Probleme bereiten Verfeinerungen, die noch einige negative Beispiele abdecken. Bei solchen Regeln müssten aufgrund der Tatsache, dass die Precision nicht eins ist, die schlechteren Schranken für Binomial und Squared verwendet werden. Dabei nimmt man allerdings eine Verfeinerung vor, die in einem schlechteren Nutzen resultiert als R'_{opt}. Gleichzeitig ergeben sich wesentlich ungenauere Konfidenzschranken. Aus diesen Gründen ist die Auswertung solcher Regeln unnötig. Somit genügt zur Klärung der Frage, ob es einen Nachfolger R' der Regel R gibt, der ein größeres Konfidenzintervall als R'_{opt} hat, die Betrachtung von Instance-Averaging Nutzenfunktionen, zu denen Coverage, Accuracy und Weigted Relative Accuracy gehören.
Formal lautet die Frage, ob es einen von R'_{opt} verschiedenen Nachfolger R' der Regel R gibt, so dass für eine Instance-Averaging Nutzenfunktion q gilt:

$$q(R'_{opt}) + E(R'_{opt}, q, m, \delta) < q(R') + E(R', q, m, \delta).$$

Die Nachfolger R' der Regel R, die in der grafischen Darstellung des Coverage Space unterhalb bzw. rechts von R'_{opt} liegen (Abbildung 5.3), decken weniger bzw. mehr Beispiele ab als R'_{opt}. Für die betrachteten Nutzenfunktionen stellt unabhängig davon jedes gezogene Beispiel ein Zufallsexperiment dar. Die Beispielanzahl m, auf deren Basis das Konfidenzintervall berechnet wird, ist daher für jeden möglichen Nachfolger von R gleich. Für geringe Beispielanzahlen m wird die Größe $E(m, \delta)$ des Konfidenzintervalles in beide Richtungen folgendermaßen berechnet:

$$E(m, \delta) = \sqrt{\frac{1}{2m} log \frac{2}{\delta}}.$$

Die Größe des Konfidenzintervalles hängt nur von der Anzahl gesehener Beispiele ab und ist für jeden möglichen Nachfolger von R gleich. Für große Beispielanzahlen m wird die beidseitige Größe des Konfidenzintervalles mit Hilfe der Normalverteilung berechnet, wobei $z_{1-\frac{\delta}{2}}$ das $(1 - \frac{\delta}{2})$-Quantil der Normalverteilung und $s_{R'}$ die empirische Standardabweichung der Regel R' bezeichnet:

$$E(m, \delta) = z_{1-\frac{\delta}{2}} \cdot s_{R'}.$$

Das $(1 - \frac{\delta}{2})$-Quantil der Normalverteilung hängt nur von δ ab und ist für alle Nachfolger von R gleich. Verschieden große Konfidenzintervalle können nur aus einer unterschiedlichen Standardabweichung resultieren. Für Instance-Averaging Nutzenfunktionen

Nutzenfunktion	obere Schranke
ACC	U(ACC,R,δ)=$ACC(R^{'}_{opt}) + E^{*}(R^{'}_{opt}, ACC, m, \delta)$
WRACC	U(WRACC,R,δ)=$(COV(R^{'}_{opt}) + E^{*}(R^{'}_{opt}, COV, m, \delta)) \cdot (1 - Pr[Y_{*}])$
SQUARED	U(SQUARED,R,δ)=$(COV(R^{'}_{opt}) + E^{*}(R^{'}_{opt}, COV, m, \delta))^{2} \cdot (1 - Pr[Y_{*}])$
BINOMIAL	U(BINOMIAL,R,δ)=$\sqrt{COV(R^{'}_{opt}) + E^{*}(R^{'}_{opt}, COV, m, \delta)} \cdot (1 - Pr[Y_{*}])$

Tabelle 5.1: Obere Schranken für verschiedene Nutzenfunktionen

wie Accuracy und Weighted Relative Accuracy, die als Durchschnitt der Werte einer Instanznutzenfunktion q_{inst} definiert sind, wird die empirische Standardabweichung $s_{R'}$ des Nutzens einer Regel $R^{'}$ berechnet als:

$$\frac{1}{m}\sqrt{\sum_{i=1}^{m}(q_{inst}(h, x_i) - \hat{q}(h, Q_m))^2}.$$

Wiederum ist die Zahl m der Beispiele für alle Nachfolger gleich. Unterschiede in der Standardabweichung ergeben sich durch die individuellen Werte der Instanznutzenfunktion. Sie nimmt die Werte null oder eins an. Die empirische Standardabweichung wird maximiert, wenn q_{inst} für eine Hälfte der Beispiele den Wert null annimmt und für die andere den Wert eins. Der maximale Wert beträgt $\frac{1}{2\sqrt{m}}$. Durch die Verfeinerung einer Regel R zu einem Nachfolger $R^{'}$, verändern sich für einige q_{inst} die Werte. Die Standardabweichung wird maximal für den Nachfolger $R^{'}$, bei dem die Anzahl der Instanznutzenfunktionen mit $q_{inst} = 1$ am nächsten an $\frac{m}{2}$ liegt. Dieses muss nicht zwangsläufig für $R^{'}_{opt}$ der Fall sein. Daher kann es einen direkten Nachfolger $R^{'}$ der Regel R geben, der ein größeres Konfidenzintervall hat als $R^{'}_{opt}$. Außerdem kann auf diese Weise nicht garantiert werden, dass die Summe von Nutzen und Größe des Konfidenzintervalles bei weiterer Verfeinerung von $R^{'}_{opt}$ monoton abnimmt. Um eine konservative obere Schranke zu erhalten, kann deshalb bei der Berechnung des Konfidenzintervalles von $R^{'}_{opt}$ die maximale Standardabweichung $\frac{1}{2\sqrt{m}}$ benutzt werden.
Bezeichnet $E^{*}(R^{'}_{opt}, q, m, \delta)$ das Konfidenzintervall, dass man bei Verwendung der maximalen Standardabweichung für gegebenes δ und die Nutzenfunktion q erhält, ergeben sich die in Tabelle 5.1 angegebenen oberen Schranken.

5.5 Algorithmus

Der Iterating GSS Algorithmus verwendet alle in den obigen Abschnitten vorgestellten Konzepte. Der Hypothesenraum aller Regeln wird von geringer zu hoher Komplexität

durchsucht. Es werden zunächst alle nützlichen Regeln der Länge eins bestimmt. Zur Beurteilung der Nützlichkeit einer Regel der aktuellen Komplexität wird eines der Kriterien aus Kapitel 5.3 verwendet. Wird eine Regel gefunden, die nicht nützlich gemäß dieses Kriteriums ist, erfolgt eine Erhöhung der Komplexität. Die Anzahl der erzeugten Regeln wird durch den Einsatz von Pruning kontrolliert. Dieses Vorgehen wird bis zu einer gegebenen maximalen Komplexität wiederholt.
Der Iterating GSS Algorithmus ist mit den grundlegenden Parametern in Abbildung 5.5 angegeben. Die Notation lehnt sich stark an den in [24] verwendeten Pseudocode an. Für Zuweisungen wird das Symbol „←“ benutzt, Methoden- und Funktionsnamen werden in Großbuchstaben, Anweisungen in fetten Buchstaben notiert. Für Vergleiche und die vier Grundrechenarten finden die mathematischen Standardsymbole Verwendung. Eine Übersicht der für den Iterating GSS Algorithmus verwendeten Notationen befindet sich in Anhang A. Bei den Datentypen und -strukturen *Boolean, Integer, Double, Menge* und *Liste* wird vorausgesetzt, dass sie zusammen mit den möglichen Operationen bekannt sind. Die Datenstruktur *Resultat* verwaltet eine Regel inklusive Nutzen sowie Konfidenzintervall und erlaubt den Zugriff auf die einzelnen Bestandteile. Diese und andere verwendete Datenstrukturen werden genauer in Kapitel 5.7 beschrieben. Eine Übersicht der Datentypen und -strukturen ggf. unter Angabe der möglichen Operationen ist in Anhang B dargestellt. Bei den Datenstrukturen *Menge* und *Liste* wird stets der enthaltene Datentyp angegeben. Der Algorithmus erhält als Eingabe zunächst den Instanzenraum X, die Trainingsmenge T von Beispielen und die Nutzenfunktion q. Dabei handelt es sich um eine der Funktionen Weighted Relative Accuracy, Accuracy, Squared und Binomial. Desweiteren geben δ, ϵ und q^p_{min} die Irrtumswahrscheinlichkeit, den maximal zulässigen Fehler und den Minimalnutzen an, den eine Regel haben muss, um in die Lösung aufgenommen zu werden. Die Anzahl von Regeln in der Lösung wird durch den Parameter k angegeben. Der Eingabeparameter c_{start} gibt die Komplexität der Regeln zu Beginn an. Der initiale Hypothesenraum besteht aus allen Regeln, deren Komplexität kleiner oder gleich c_{min} ist. Mit c_{max} wird die maximale Komplexität angegeben, die Regeln haben können. Um zu entscheiden, wann der Algorithmus die Komplexität erhöht, benötigt er ein Kriterium, um die Qualität einzelner Regeln bewerten zu können. Der Eingabeparameter α bestimmt welches Kriterium benutzt wird. Bei q^u_{min} und f handelt es sich um die Parameter Mindestnutzen und Beispielfaktor, die zur Bewertung der Qualität einer Regel benötigt werden (Kapitel 5.3). Zusätzlich geben β die Schrittgröße für den GSS Algorithmus und γ die Anzahl von Beispielen an, die eine Regel abdecken muss, bevor die Approximation durch die Normalverteilung benutzt wird. Schließlich legt der Parameter ω fest, ob die Verwerfungsmethode oder die Beispielgewichte verwendet werden.
Zu Beginn des Algorithmus werden für alle Beispiele x die Gewichte $w(x)$ mit eins initialisiert, da zunächst alle Beispiele gleichverteilt gezogen werden. Der initiale Hypothesenraum wird gemäß dem Eingabeparameter c_{start} erzeugt und die in der Variable c_{akt} gespeicherte aktuelle Komplexität auf c_{start} gesetzt. Mit der Schleife in Schritt 3 beginnt das zentrale Element des Algorithmus. Die verbleibende Irrtumswahrscheinlickeit wird immer gleichmäßig auf alle verbleibenden Iterationen aufgeteilt. Der für die aktuelle Iteration zur Verfügung stehende Teil wird im Verhältnis zwei zu ein Drittel auf die Variablen δ_{gss} und $delta_p$ aufgeteilt. Die verbleibende Irrtumswahrscheinlichkeit wird in der Variablen δ verwaltet und um $delta_{gss}$ und $delta_p$ verringert. Dann wird mit dem GSS Algorithmus die beste Regel für die aktuelle Verteilung unter Verwendung der Irrtums-

Eingabe:

1. X: Instanzenraum; T: Trainingsmenge; q: Nutzenfunktion
2. $\delta, \epsilon, q^p_{min}, q^u_{min}, f$: Double
3. $k, c_{start}, c_{max}, \alpha, \beta, \gamma$: Integer; ω:Boolean

Ausgabe: Die approximativ k-besten gemäß Knowledge-Based Sampling unabhängigen Regeln mit Maximalfehler ϵ und Konfidenz $1 - \delta$
Lokale Variablen:

1. $\mathcal{M}, \mathcal{M}'$: Menge<Regel>; $\mathcal{N}$: Menge<Resultat>
2. $\mathcal{S}$: Resultat
3. δ_{gss}, δ_p: Double

Algorithmus:

1. **for** $x \in T$ **do** $w(x) \leftarrow 1$.
2. ADD-ALL($\mathcal{M}$,GENERATE-RULES (c_{start},X))
3. $c_{akt} \leftarrow c_{start}$
4. **for** i $\leftarrow$ 0 **to** k **do**
   a) $\delta_{gss} \leftarrow \frac{2 \cdot \delta}{3 \cdot (k-i)}$
   b) $\delta_p \leftarrow \frac{\delta}{3 \cdot (k-i)}$
   c) $\delta \leftarrow \delta - \delta_{gss} - \delta_p$
   d) $\mathcal{S} \leftarrow GSS(\mathcal{M}, T, 1, q, \delta_{gss}, \epsilon, \beta, \gamma, \omega)$
   e) **if** (IS-USEFUL ($\mathcal{S}, \mathcal{N}, \alpha, q^u_{min}, f$)) **then**
      i. ADD($\mathcal{N}, \mathcal{S}$)
      ii. **for** ($x \in T$) **do** $w(x) \leftarrow w(x) \cdot LIFT(\mathcal{S}, x)^{-1}$
      iii. **if** ($\omega = false$) **then**
         A. $w^* = \max\{w(x) | x \in T\}$
         B. **for** ($x \in T$) **do** $w(x) \leftarrow w(x) \cdot w(x)^{-1}$
      iv. $\delta = \delta + \delta_p$
   f) **else**
      i. **if** ($c_{akt} = c_{max}$) **then break**
      ii. ADD-ALL($\mathcal{M}'$,PRUNE-RULES ($\mathcal{M}, q^p_{min}, \delta_p$))
      iii. ADD-ALL($\mathcal{M}$,GENERATE-SUCCESSORS ($\mathcal{M}'$))
      iv. $c_{akt} \leftarrow c_{akt} + 1$
5. Ausgabe $\mathcal{N}$

Abbildung 5.4: Der Iterating Generic Sequential Sampling Algorithmus wahrscheinlickeit δ_{gss} bestimmt und zusammen mit Nutzen und Konfidenzintervall im Resultat $\mathcal{S}$ gespeichert. Ist die Qualität der Regel für das gewählte Beurteilungskriterium hoch genug, wird $\mathcal{S}$ der Lösung hinzugefügt. Anschließend wird jedes Beispielgewicht *w(x)* durch den Lift der Regel dividiert. Falls die Verwerfungsmethode benutzt wird, erfolgt die Normierung der Gewichte mit dem Maximalgewicht w^*. Da sich die Komplexität nicht erhöht, erfolgt kein Pruning. Die hierfür vorgesehene Irrtumswahrscheinlichkeit δ_p wird δ hinzugefügt und steht in der nächsten Iteration wieder zur Verfügung. Ist die Qualität zu schlecht erfolgt eine Erhöhung der Komplexität, falls noch nicht die größtmögliche Komplexität c_{max} erreicht ist. Dazu werden zunächst per Pruning alle Regeln aus dem aktuellen Hypothesenraum entfernt, deren Nachfolger nicht mehr den Minimalnutzen q^p_{min} erreichen können. Aus den verbliebenen Regeln werden die der nächstgrößeren Komplexität erzeugt und die Variable c_{akt} um eins erhöht. Der Algorithmus terminiert, wenn alle k Iterationen gemacht wurden oder eine Regel der Komplexität c_{max} gefunden wurde, deren Qualität nicht hoch genug ist. Als Ausgabe liefert er einen Ensembleklassifikator, der gemäß dem in Kapitel 4.2.4 beschriebenen Verfahren aus den gefundenen Regeln erzeugt wurde. Der verwendete GSS Algorithmus wird um die in Kapitel 5.1 angesprochenen Aspekte modifiziert. Als Parameter erhält er zusätzlich den Hypothesenraum der aktuellen Komplexität und durch γ die Angabe, ab welcher Anzahl von abgedeckten Beispielen für eine Regel die Normalverteilungsapproximation verwendet wird (5.1.3). Desweiteren gibt der Parameter ω an, ob die unnormierten Gewichte verwendet werden oder die Verwerfungsmethode (5.1.1). Durch β wird die Beispielanzahl festgelegt nach der GSS Algorithmus jeweils versucht, Regeln frühzeitig als Lösung auszugeben oder zu verwerfen.

Desweiteren muss auf die Aufteilung der Irrtumswahrscheinlichkeit auf Pruning und den GSS Algorithmus eingegangen werden. Beim Pruning wird der wahre Nutzen aller Regeln der aktuellen Komplexität auf Basis des nach einer bestimmten Beispielanzahl empirisch beobachteten Nutzen geschätzt. Aufgrund dieser Schätzung können Regeln verworfen werden, weil sie so schlecht sind, dass es nicht mehr nötig ist ihre Nachfolger zu betrachten. Um die Irrtumswahrscheinlichkeit δ einzuhalten, ist nötig einen Teil davon zu verwenden, um ein Konfidenzintervall um den geschätzten Nutzen zu berechnen, in dem der wahre Nutzen der Regel liegt. Einerseits ist wichtig hierfür viel von der Irrtumswahrscheinlichkeit zu verwenden, da auf diese Weise die Berechnung der Konfidenzintervalle genauer ist. Es können mehr schlechte Regeln verworfen werden, wodurch sich der Hypothesenraum und die Laufzeit verkleinern. Andererseits steht das verbrauchte δ nicht dem GSS Algorithmus zur Verfügung. Dieser verwendet das gegebene δ sowohl innerhalb der Schleife in Schritt 3, um frühzeitig Hypothesen zu verwerfen oder auszugeben als auch für die Terminierungsbedingung am Ende dieses Schrittes (Tabelle 4.1). Daher wird für das Pruning ein Drittel der verbleibenden Irrtumswahrscheinlichkeit zur Verfügung gestellt, d.h. jeder der drei Vorgänge erhält in jeder Iteration ein Drittel des restlichen δ. Aus der bereits in Kapitel 4.1.2 verwendeten Booleschen Ungleichung folgt, dass bei dieser Vorgehensweise die maximale Irrtumswahrscheinlichkeit von δ nicht überschritten wird.

In der Regel wird weniger als ein Drittel der gesamten Irrtumswahrscheinlichkeit für das Pruning verwendet, da nicht in jeder Iteration eine Erhöhung der Komplexität stattfindet. In diesem Fall wird der reservierte Anteil δ_p der Irrtumswahrscheinlichkeit vom Iterating GSS Algorithmus dem verbleibenden δ hinzugefügt.

Der Iterating GSS Algorithmus wurde für die Lernumgebung YALE implementiert [21]. Eine Übersicht über alle Parameter befindet sich in Anhang C.
Abschließend soll auf die Möglichkeit eingegangen werden, den Iterating GSS Algorithmus zur Verarbeitung von *Datenströmen* einzusetzen (*Streaming*). Man spricht von einem Datenstrom, falls die Trainingsdaten nicht als Ganzes (*Batch*) vorliegen, sondern die Daten über einen längeren Zeitraum gesammelt werden. Dem Lernalgorithmus werden die Beispiele nur sequentiell zur Verfügung gestellt. Dabei ist nicht möglich im Voraus zu bestimmen, wann und ob der Datenstrom endet. Ein Beispiel für einen Datenstrom sind die an den Kassen von Supermärkten erfassten Daten über die Einkäufe der Kunden. Es handelt sich dabei um einen kontinuierlichen Datenstrom mit Informationen über das Kaufverhalten der Kunden. Für den Betreiber des Supermarktes sind die Informationen in diesem Datenstrom für die Planung seines Sortiments von Interesse. Eine Schwierigkeit im Umgang mit Datenströmen besteht in der Bestimmung der Beispielanzahl, nach der vom Lernalgorithmus ein Modell ausgeben werden kann.
Obwohl der Iterating GSS Algorithmus in der jetzigen Version die gesamte Trainingsmenge einliest, arbeitet er implizit mit einem Datenstrom. Die Beispiele werden sequentiell verarbeitet und der Algorithmus bestimmt, wann bei gegebener Irrtumswahrscheinlichkeit δ und maximalem Fehler ϵ genug Beispiele verarbeitet wurden, um eine Regel als Lösung auszugeben. Der Algorithmus kann leicht für die direkte Verwendung eines Datenstromes angepasst werden. Liegt der gesamte Datensatz vor, kann die apriori Wahrscheinlichkeit für die positive Klasse berechnet werden. Da diese bei direkter Verwendung eines Datenstromes nicht bekannt ist, muss sie aufgrund der bisher gesehenen Beispiele unter Verwendung eines Teiles der Irrtumswahrscheinlichkeit geschätzt werden. Dazu kann die Hoeffding-Ungleichung eingesetzt werden.

5.6 Variationen des Iterating GSS Algorithmus

Neben den bereits vorgestellten Variationsmöglichkeiten verfügt der Iterating GSS Angorithmus über eine Reihe weiterer, auf die in diesem Abschnitt eingegangen wird.
Überanpassung ist ein bekanntes Problem im Maschinellen Lernen. Man spricht von Überanpassung an die Trainingsdaten, wenn der Lernalgorithmus ein Modell findet, das an zufällige Fluktuationen oder Beispiele mit falschem Wert für das Zielattribut angepasst ist. Ein Grund für fehlerhafte Beispiele sind Fehler bei der Erhebung oder inkorrekte Eingabe der Trainingsdaten. Mit zunehmender Komplexität der Hypothesen ist möglich, immer feinere Muster in den Daten zu finden. In der statistischen Lerntheorie wurde gezeigt, dass ein Zusammenhang zwischen der Gefahr für eine Überanpassung an die Trainingsdaten und der Komplexität des verwendeten Hypothesenraumes besteht. Durch den Einsatz von Knowledge-Based Sampling entfernt der Iterating GSS Algorithmus nach jeder Iteration eine Regel bzw. die Korrelation zwischen den Vorhersagen und den tatsächlichen Werten des Zielattributes aus den Daten. Wurden nach vielen Iterationen die besten Regeln aus den Daten entfernt, besteht verstärkt die Gefahr Regeln zu finden, die nur zufällige Schwankungen in den Daten wiedergeben. Durch die Erhöhung der Komplexität wird die Gefahr der Überanpassung verstärkt. Die Nutzenfunktion Binomial (Kapitel 2.4) basiert auf einem Signifikanztest, d.h. sie ist für Überanpassung weniger anfällig als andere Nutzenfunktionen. Es bietet sich an, statt einer Erhöhung der Komplexität zunächst von der bisherigen Nutzenfunktion auf Binomial zu wechseln.

Es ist vorteilhaft, Binomial nicht ab der ersten Iteration einzusetzen, weil die Laufzeit bei Verwendung dieser Nutzenfunktion lang ist. Die Konfidenzintervalle und die maximal benötigte Beispielanzahl bis zur Terminierung des GSS Algorithmus sind bei Binomial wesentlich größer als bei den anderen Nutzenfunktionen. Es bietet sich die Kombination von Weighted Relative Accuracy oder Accuracy mit Binomial an, weil in den frühen Iterationen die Gefahr für eine Überanpassung gering ist. Die Konfidenzintervalle und maximal benötigten Beispielanzahlen für Accuracy und Weighted Relative Accuracy sind kleiner als die von Binomial. Es ist sinnvoll, in den frühen Iterationen mit geringer Gefahr der Überanpassung die gezielteren Nutzenfunktion einzusetzen und in den späteren Iterationen auf Binomial zu wechseln, um eine Überanpassung zu vermeiden. In den durchgeführten Experimenten wurde dieses verifiziert 6.2.
Eine weitere Variationsmöglichkeit besteht darin, für jede Komplexität einen erweiterten Hypothesenraum zu betrachten. Die Subgruppenentdeckung bedarf der Wahl einer interessierenden Klasse Y_+ des Zielattributes. Der Hypothesenraum besteht aus allen Regeln der Form $r \rightarrow Y_+$ und kann Subgruppen finden, in denen der Anteil an positiven Beispielen gegenüber der gesamten Trainingsmenge erhöht ist. Subgruppen, in denen negative Beispiele überrepräsentiert sind, können nur indirekt gefunden werden. Als Beispiel betrachte man ein Attribut mit zwei Ausprägungen. Anhand der beiden Ausprägungen des Attributes lässt sich die Menge der Beispiele in zwei Klassen partitionieren. Sind in der einen Klasse die negativen Beispiele überrepräsentiert, bedeutet das zugleich, dass in der anderen Klasse die positiven Beispiele überrepräsentiert sind. Das indirekte finden von Subgruppen wird schwerer, je mehr Ausprägungen ein Attribut hat. Da die Beispielmenge in mehr als zwei Klassen partitioniert ist, kann eine starke Überrepräsentation von negativen Beispielen in einer Klasse von mehreren leichten Überrepräsentationen der positiven Beispiele in den anderen Klassen resultieren. Der Iterating GSS Algorithmus bietet daher die Option auch alle Hypothesen der Form $r \rightarrow Y_-$ zu generieren, wodurch möglich ist, Subgruppen mit großem Anteil von negativen Beispielen direkt zu finden.
Desweiteren ist möglich, die Terminierungsbedingung zu verändern. Normalerweie terminiert der Iterating GSS Algorithmus, wenn er eine Regel mit der maximal möglichen Komplexität findet, die nicht als nützlich eingestuft wird. Es ist möglich, diese Terminierungsbedingung außer Kraft zu setzen und eine feste Anzahl von Iterationen vorzugeben, die in jedem Fall durchgeführt werden.
Speziell für die Subgruppenentdeckung können die Beispielgewichte für jede Komplexitätsstufe wieder mit eins initialisiert werden. Der Iterating GSS Algorithmus beginnt die Subgruppenentdeckung meist mit einer geringen Komplexität, z.B. mit allen Regeln der Länge eins. Durch die Veränderung der Verteilung nach jeder Iteration können Muster in den Daten, die nur mit einer höheren Komplexität darzustellen sind, zerstört werden. Man betrachte das Beispiel einer Subgruppe, die sich durch eine Regel mit zwei Prämissen dargestellen läßt. Diese ist in den Daten nicht zu finden, wenn die den zwei Prämissen entsprechenden Regeln der Länge eins gefunden und aus den Daten entfernt wurden. Ist man nur an der Entdeckung von Subgruppen interessiert, kann es sinnvoll sein, die Gewichte nach Erhöhung der Komplexität wieder auf eins zu setzen. Eine Kombination der gefundenen Subgruppen bzw. Regeln zu einem Ensemble gemäß dem in Kapitel 4.2.4 vorgestellten Verfahren ist nicht mehr möglich.
Eine Übersicht über alle Parameter zur Wahl der zu verwendenden Variante des Iterating GSS Algorithmus befindet sich in Anhang C.

5.7 Implementierung der Hilfsmethoden

Neben den primitiven Datentypen *Boolean*, *Double* und *Integer* sowie den bekannten Datenstrukturen *Menge* und *Liste* wurden für den Iterating GSS Algorithmus die Datenstrukturen *Regel*, *Resultat* und *Nutzen* implementiert. Bevor auf die Hilfsmethoden, die vom Iterating GSS Algorithmus zur Durchsuchung des Hypothesenraumes benutzt werden eingegangen wird, erfolgt eine Beschreibung dieser drei Datenstrukturen. Eine Übersicht aller Datenstrukturen und möglichen Operationen befindet sich in Anhang B.
Eine Regel aus konjunktiv verknüpften Literalen wird von der gleichnamigen Datenstruktur repräsentiert. Sie verwaltet das abgedeckte und das korrekt vorhergesagte Beispielgewicht und stellt Methoden zum Umgang mit Regeln zur Verfügung. Insbesondere bietet sie die Möglichkeit, die direkten Nachfolger einer Regel zu erzeugen. Sie spielt daher eine wichtige Rolle bei der Generierung des Hypothesenraumes. An einigen Stellen des Iterating GSS Algorithmus ist es nötig, auf Resultate des GSS Algorithmus zurückzugreifen. Die Datenstruktur *Resultat* verwaltet eine Regel und die zugehörigen Statistiken wie Nutzen, Größe des Konfidenzintervalles und benötigte Stichprobengröße bis zur Ausgabe. Abschließend wird noch auf die Datenstruktur *Nutzen* eingegangen. Um sie zu beschreiben, werden Begriffe aus der *Objektorientierten Programmierung* [6] verwendet. Es handelt sich um einen *abstrakten* Datentyp, der als *Superklasse* für die Implementierung der mit *abgeleiteten Klassen* bezeichneten konkreten Nutzenfunktionen wie Weighted Relative Accuracy oder Accuracy dient. Die abgeleiteten Klassen müssen die von der Superklasse Nutzen vorgeschriebenen Methoden zur Berechnung von Nutzen, Konfidenzintervall und der für das Pruning benötigten oberen Schranke implementieren. Zur Laufzeit wird die Methode der konkreten Nutzenfunktion ausgeführt. Die Methoden müssen für eine gegebene Regel als Rückgabe die für die konkrete Nutzenfunktion korrekten Werte von Nutzen, Konfidenzintervall und oberer Schranke liefern. Die Methode zur Berechnung der oberen Schranke benötigt zusätzlich eine maximale Irrtumswahrscheinlichkeit als Parameter.

5.7.1 Methoden zur Erzeugung und Durchsuchung des Hypothesenraumes

In *GENERATE-RULES* bzw. *GENERATE-SUCCESSORS* wird die Methode *REFINE* der Datenstruktur *Regel* eingesetzt, um Regeln zu generieren bzw. die direkten Nachfolger zu erzeugen. Sie sind in den Abbildungen 5.5 und 5.6 angegeben. In der *GENERATE-RULES*-Methode werden im ersten Schritt alle Regeln der Komplexität eins erzeugt. Solange die maximale initiale Komplexität c_{start} nicht erreicht ist, werden alle Regeln einer als Queue fungierenden Liste hinzugefügt. Als Rückgabewert wird die Menge aller Regeln geliefert, die eine Komplexität von c_{start} oder geringer hat. Die *GENERATE-SUCCESSORS*-Methode erzeugt aus einer gegebenen Liste von Regeln deren direkte Nachfolger.
Die Methode *IS-USEFUL* benutzt die in der Datenstruktur Resultat gespeicherten Statistiken einer Regel, um deren Nützlichkeit zu bewerten. Sie ist in Abbildung 5.7 beschrieben. Zur Bewertung der Nützlichkeit einer Regel wird eines der in Kapitel 5.3 beschriebenen Kriterien verwendet. Die Kriterien werden über einen Index selektiert (Tabelle

GENERATE-RULES (c_{start},X)
Parameter:

1. c_{max}: Integer
2. X: Instanzenraum

Rückgabewert: Menge aller Regeln, deren Länge c_{start} beträgt oder kürzer ist.
Lokale Variablen:

1. $\mathcal{L}$: Liste<Regel>
2. $\mathcal{M}$: Menge<Regel>
3. $\mathcal{R}$: Regel

Algorithmus:

1. APPEND-ALL($\mathcal{L}, \{\mathcal{R} | LENGTH(\mathcal{R}) = 1\}$)
2. **while** ($\mathcal{L} \neq \emptyset$) **do**
   a) $\mathcal{R} \leftarrow$ REMOVE-FIRST($\mathcal{L}$)
   b) ADD($\mathcal{M}, \mathcal{R}$)
   c) **if** ($LENGTH(\mathcal{R}) < c_{start}$) **then** APPEND-ALL($\mathcal{L}$,REFINE($\mathcal{R}$))
3. **return** $\mathcal{M}$

Abbildung 5.5: Die GENERATE-RULES-Methode

GENERATE-SUCCESSORS ($\mathcal{L}$)
Parameter:

1. $\mathcal{L}$: Liste$<$Regel$>$

Rückgabewert: Menge aller Regeln, die direkte Nachfolger der Regeln in $\mathcal{L}$ sind.
Lokale Variablen:

1. $\mathcal{M}$: Menge$<$Regel$>$
2. $\mathcal{R}$: Regel

Algorithmus:

1. **while** ($\mathcal{L} \neq \emptyset$) **do**
   a) $\mathcal{R} \leftarrow$ REMOVE-FIRST($\mathcal{L}$)
   b) ADD-ALL($\mathcal{M}$,REFINE($\mathcal{R}$))
2. **return** $\mathcal{M}$

Abbildung 5.6: Die GENERATE-SUCCESSORS-Methode

Name des Kriteriums	Beschreibung	Index
schlechtester Nutzen	Vergleicht den wahren Nutzen im schlechtesten Fall mit dem Mindestnutzen	0
bester Nutzen	Vergleicht den wahren Nutzen im besten Fall mit dem Mindestnutzen	1
Nutzen	Vergleicht den geschätzten Nutzen mit dem Mindestnutzen	2
Beispielanzahl	Vergleicht die benötigte Beispielanzahl der letzten gefundenen Regel mit der durchschnittlich benötigten Beispielanzahl	3

Tabelle 5.2: Bewertungskriterien der IS-USEFUL-Methode

5.2). Die Güte einer Regel kann anhand des schlechtesten oder besten möglichen wahren Nutzens bewertet werden, der bei geschätztem Nutzen und zugehörigem Konfidenzintervall möglich ist. Auch besteht die Möglichkeit, das Konfidenzintervall zu vernachlässigen und ausschließlich den geschätzten Nutzen zu verwenden. Desweiteren kann die benötigte Beispielanzahl bis zur Ausgabe durch den GSS Algorithmus als Kriterium eingesetzt werden. Ist die Anzahl um einen gegebenen Faktor f größer als die bisher durchschnittlich benötigte Beispielanzahl bis zur Ausgabe, wird die Regel als nicht nützlich eingestuft.

5.7.2 Pruning

Die Hilfsmethode *PRUNE-RULES* zum Verwerfen von Regeln, deren direkte Nachfolger auf der nächsthöheren Komplexitätsstufe mit hoher Wahrscheinlichkeit keinen guten

IS-USEFUL $(\mathcal{S}, \mathcal{M}, \alpha, q_{min}, f)$
Parameter:

1. $\mathcal{S}$: Resultat
2. $\mathcal{M}$: Menge<Resultat>
3. α: Integer; f, q_{min}: Double

Rückgabewert: Boolescher Wert, der angibt, ob die Regel gemäß des gegebenen Bewertungskriteriums α nützlich ist.
Lokale Variablen:

1. b: Boolean; c: Integer; d: Double
2. $\mathcal{S}'$: Resultat

Algorithmus:

1. $b \leftarrow false$
2. **case** $\alpha = 0$
   a) **if** (GET-UTIL($\mathcal{S}$)-GET-CONF($\mathcal{S}$)$>q_{min}$) **then** $b \leftarrow true$
3. **case** $\alpha = 1$
   a) **if** (GET-UTIL($\mathcal{S}$)$+$GET-CONF($\mathcal{S}$)$>q_{min}$) **then** $b \leftarrow true$
4. **case** $\alpha = 2$
   a) **for** ($\mathcal{S}' \in \mathcal{M}$) **do** $c \leftarrow c+$GET-NEEDED-EXAMPLES($\mathcal{S}'$)
   b) $d \leftarrow c\div$ SIZE($\mathcal{M}$)
   c) **if** (GET-NEEDED-EXAMPLES($\mathcal{S}$)$<d \cdot f$) **then** $b \leftarrow true$
5. **return** b

Abbildung 5.7: Die IS-USEFUL-Methode

PRUNE-RULES $(\mathcal{L}, q_{min}, \delta_p)$
Parameter:

1. $\mathcal{L}$: Liste<Regel>
2. q_{min}: Integer; δ_p: Double

Rückgabewert: Menge von Regeln, die den minimalen Nutzen q_{min} erreichen.
Lokale Variablen:

1. $\mathcal{M}$: Menge<Regel>
2. $\mathcal{R}$: Regel
3. u: Double

Algorithmus:

1. **while** $(\mathcal{L} \neq \emptyset)$ **do**
   a) $\mathcal{R} \leftarrow$ REMOVE-FIRST($\mathcal{L}$)
   b) u $\leftarrow$ GET-UPPER-BOUND($\mathcal{R}$, $\frac{\delta_p}{SIZE(\mathcal{L})}$)
   c) **if** $(u \geq q_{min})$ **then** ADD($\mathcal{M}, \mathcal{R}$)
2. **return** $\mathcal{M}$

Abbildung 5.8: Die PRUNE-RULES-Methode

Nutzen haben, ist in Tabelle 5.8 dargestellt. Die Methode erhält als Parameter eine Liste von Resultaten, den Minimalnutzen q_{min} sowie die zur Verfügung stehende Irrtumswahrscheinlichkeit δ_p. Zentral ist die Methode *GET-UPPER-BOUND* der Datenstruktur Nutzen, die für jede verwendete Nutzenfunktion separat zu implementieren ist. Sie berechnet eine obere Schranke für den Nutzen einer Regel gemäß dem in Kapitel 5.4 angegebenen Verfahren. Es werden alle Regeln durchlaufen und jeweils eine obere Schranke berechnet. Daher ist die zur Verfügung stehende Irrtumswahrscheinlichkeit δ_p auf alle Regeln aufzuteilen. Als Rückgabewert liefert die Methode die Menge der Regeln, deren obere Schranke größer ist als q_{min}.

6 Experimente

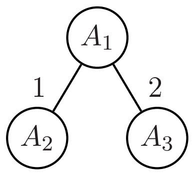
In diesem Kapitel wird zunächst anhand von synthetischen Datensätzen die Eignung des Iterating GSS Algorithmus für die Lernaufgaben Subgruppenentdeckung und Konzeptlernen aus Beispielen getestet. Zur Generierung von synthetischen Datensätzen wurde der BiasedExampleGenerator eingesetzt 6.1.1. Nach der Anwendung auf synthetischen Daten, erfolgte die Messung der Performanz des Algorithmus auf einigen großen realen Datensätze, indem die gefundenen Ergebnisse mit denen einiger anderen Lernalgorithmen verglichen wurden.

6.1 Experimente mit synthetischen Datensätzen

6.1.1 Der BiasedExampleGenerator Operator

Zur Generierung von synthetischen Datensätzen wurde der Operator BiasedExampleGenerator entworfen und für die Lernumgebung YALE [21] implementiert. Er ermöglicht Datensätze zu generieren und gezielt Muster einzubauen. Zentrales Element des Operators ist der *BiasTree*. Er gibt die bedingte Wahrscheinlichkeit für das Auftreten eines bestimmten Beispieles in Form eines Entscheidungsbaumes an. In Abbildung 6.1 ist ein einfacher BiasTree dargestellt. Die Knoten des Baumes sind mit den Namen der Attribute A_1, A_2 und A_3 beschriftet, die jeweils zehn Ausprägungen haben. Für jeden Knoten ist eine Verteilung angegeben, die festlegt mit welcher Wahrscheinlichkeit die Ausprägungen des Attributes bzw. die entsprechenden Indizes[1] auftreten. Die mit Zahlen beschrifteten von einem Knoten ausgehenden Kanten geben an, bei welcher Ausprägung des Attributes zum nächst tieferen Knoten verzweigt wird.
Jedes Beispiel wird zunächst zufällig erzeugt, d.h. für alle Attribute hat jede Ausprägung die gleiche Wahrscheinlichkeit gezogen zu werden. Danach durchläuft das Beispiel

[1]Hat ein Attribut $|A_i|$ Ausprägungen, werden diese von 0 bis $|A_i| - 1$ durchnummeriert.

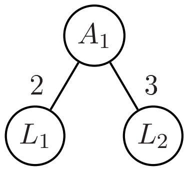


Attribut	Verteilung über die Ausprägungen
A_1	0.1 0.1 0.2 0.1 0.1 0.05 0.05 0.1 0.1 0.1
A_2	0.1 0.1 0.2 0.1 0.1 0.05 0.05 0.1 0.1 0.1
A_3	0.1 0.2 0.1 0.1 0.1 0.05 0.05 0.1 0.1 0.1

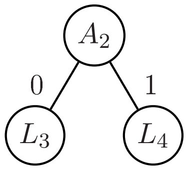
Abbildung 6.1: Einfaches Beispiel für einen BiasTree

Attribut	Verteilung über die Ausprägungen
A_1	0.05 0.05 0.2 0.2 0.1 0.05 0.05 0.1 0.1 0.1

Attribut	Verteilung über die Ausprägungen
A_2	0.2 0.2 0.1 0.05 0.05 0.05 0.05 0.1 0.1 0.1



Attribut	Verteilung über die Ausprägungen
A_1	keine neue Verteilung
L_1	0.3 0.7
L_2	0.4 0.6

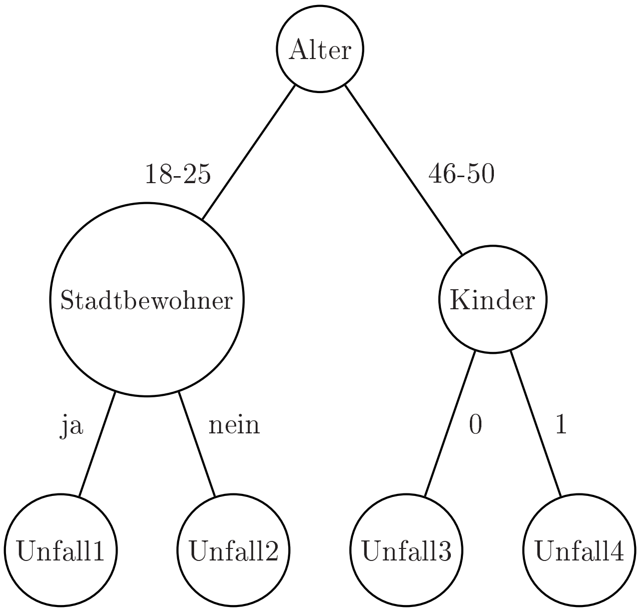


Attribut	Verteilung über die Ausprägungen
A_2	keine neue Verteilung
L_3	0.6 0.4
L_4	0.5 0.5

Abbildung 6.2: Menge von mehreren BiasTrees, die zunächst die Verteilung der Ausprägungen der Attribute angeben und dann Bedingungen für die Ausprägung des Zielattributes formulieren.

den BiasTree. An jedem Knoten wird die Ausprägung des entsprechenden Attributes gemäß der angegebenen Verteilung neu ausgewürfelt. Entspricht der Index der Ausprägung dem an einer von diesem Knoten ausgehenden Kante notierten Index, wird in den entsprechenden Knoten verzweigt. Ansonsten wird der Durchlauf durch den BiasTree abgebrochen. Wurde für ein Beispiel aufgrund der Verteilung im Wurzelknoten für Attribut A_1 die Ausprägung 2 ausgewürfelt, wird in den mit A_3 beschrifteten Knoten verzweigt. Hier wird gemäß der gegebenen Verteilung die Ausprägung von A_3 für das betrachtete Beispiel bestimmt, bevor der Durchlauf durch den BiasTree beendet wird.
Es ist möglich mehrere BiasTrees anzugeben, die der Reihe nach abgearbeitet werden. Desweiteren ist nicht nötig, für jeden Knoten eine Verteilung anzugeben. Ist für einen Knoten keine Verteilung vorhanden, wird die Ausprägung des betrachteten Attributes für das Beispiel nicht neu ausgewürfelt. Stattdessen wird die bisherige Ausprägung beibehalten und benutzt, um zum nächst tieferen Knoten innerhalb des BiasTrees zu verzweigen, falls dieser vorhanden ist. Da jedes Beispiel zufällig initialisiert wurde, ist eine Ausprägung immer vorhanden. Es bietet sich an, zunächst mit mehreren BiasTrees, die nur aus einem Knoten bestehen (*entarteter BiasTree*), die Verteilung der Ausprägungen der Attribute anzugeben. Danach können mit weiteren BiasTrees Bedingungen über die Ausprägung des Zielattributes formuliert werden. In Abbildung 6.2 ist ein Beispiel dargestellt. Zunächst werden die Wahrscheinlichkeiten für die jeweils zehn Ausprägungen der Attribute A_1 und A_2 angegeben. Danach erfolgen Angaben über die Verteilung der beiden Ausprägungen des mit L bezeichneten Zielattributes in zwei weitern BiasTrees. Während mit den Beschriftungen A_1 und A_2 unterschiedliche Attribute bezeichnet werden, stehen L_1 bis L_4 für das selbe Attribut, wobei jeweils eine unterschiedliche Verteilung zur Bestimmung der Ausprägung benutzt wird. Da die BiasTrees der Reihenfolge nach abgearbeitet werden, erfolgt zunächst die Veränderung der Verteilung der Attribute A_1 und A_2. Danach wird für Beispiele, die die entsprechenden Ausprägungen von A_1 und A_2 haben, die Veränderung des Zielattributes gemäß den Angaben der letzten beiden BiasTrees vorgenommen.
Es können beliebig große Datensätze erzeugt werden und die Vorgabe der apriori Wahrscheinlichkeit der positiven Klasse des Zielatributes ist als entarteter BiasTree möglich. Außerdem kann dem Datensatz *Rauschen* hinzugefügt werden. In realen Datensätzen spricht man von Rauschen, wenn einzelne Beipiele aufgrund fehlerhafter Erhebung oder falscher Eingabe widersprüchlich zu den Mustern sind, die den Daten unterliegen. Der Operator erzeugt ein Beispiel zunächst gemäß des gegebenen BiasTree. Danach wird entsprechend der Wahrscheinlichkeit für das Auftreten von Rauschen die Klasse des Zielattributes zufällig neu festgelegt. Beträgt die Wahrscheinlichkeit eins, sind die Klassen des Zielattributes völlig zufällig; ist sie null entsprechen die erzeugten Daten genau dem vorgegebenen BiasTree. Desweiteren ist es möglich festzulegen, ab und bis zu welchem erzeugten Beispiel die durch BiasTrees festgelegten Muster in den Daten zu finden sind. Dieses kann sinnvoll sein, falls mit Datenströmen gearbeitet wird. Insbesondere ist es möglich, mit diesen Werten einen *Konzeptdrift* im Datenstrom zu simulieren. Unter einem Konzeptdrift versteht man die Veränderung des den Daten unterliegenden Konzeptes im Laufe der Zeit. Ein Beispiel für einen Konzeptdrift ist die Veränderung des Kaufverhaltens der Kunden eines Unternehmens mit der Zeit. Die Erkennung dieser Veränderung ist für das Unternehmen von großer Bedeutung. Ein Ansatz für das Lernen aus Datenströmen und den Umgang mit Konzeptdrifts ist in [29] dargestellt. Diese Möglichkeit

Attribut	Verteilung über die Ausprägungen
Unfall	0.2 0.8
Alter	0.25 0.15 0.1 0.1 0.05 0.2 0.05 0.05 0.02 0.01 0.01 0.01
Stadtbewohner	0.6 0.4
Kinder	0.3 0.25 0.2 0.05 0.05 0.05 0.05 0.02 0.02 0.01



Attribut	Verteilung über die Ausprägungen
Unfall1	0.4 0.6
Unfall2	0.6 0.4
Unfall3	0.05 0.95
Unfall4	0.35 0.65

Abbildung 6.3: Der für die Erzeugung des synthetischen Datensatzes verwendete BiasTree

wurde der Vollständigkeit halber erwähnt und wird nicht weiter verfolgt.
Zur Darstellung des BiasTree wird eine XML [2] basierte Syntax verwendet. Eine Übersicht der Parameter des BiasedExampleGenerator, der verwendeten XML-Tags und die XML-Beschreibung des BiasTree aus Abbildung 6.2 befindet sich in Anhang D.

6.1.2 Ergebnisse

Zum Test der korrekten Arbeitsweise des Iterating GSS Algorithmus und der Eignung für die Lernaufgaben Subgruppenentdeckung und Konzeptlernen aus Beispielen wurde mit dem Operator BiasedExampleGenerator ein synthetischer Datensatz erzeugt. Er ist der Datenbank einer Kfz-Versicherung nachempfunden, d.h. jede Instanz beschreibt die Eigenschaften eines Versicherten. Die Versicherung ist daran interessiert, in der Datenbank solche Subgruppen zu finden, für die ein erhöhtes Unfallrisiko besteht, um dement-

sprechend ihre Tarife anzupassen. Die formulierten Tatsachen über das Unfallrisiko bestimmter Personenkreise wurden für Testzwecke erdacht und basieren in keiner Weise auf Untersuchungen in der Realität.
Der Instanzenraum besteht aus zehn Attributen, von denen drei die Muster in den Daten enthalten. Das Attribut ‚Alter' hat zwölf Altersklassen als Ausprägungen. Die Altersklassen reichen von 18-25 Jahren bis zum Alter von 76-80 Jahren. Ein weiteres Attribut mit zehn Ausprägungen gibt die Anzahl der Kinder des Versicherten von null bis neun an. Als letztes teilt das Attribut ‚Stadtbewohner' mit den Ausprägungen ‚Ja' und ‚Nein' die Versicherten in Stadt- und Landbevölkerung ein. Das Zielattribut ‚Unfall' hat ebenfalls die Ausprägungen ‚Ja' und ‚Nein'. Es gibt an, ob die Person schon einen Unfall verursacht hat. Für die anderen Attribute wurden keine Muster erzeugt, d.h. für jedes generierte Beispiel wurde eine Ausprägung zufällig gewählt. Mit Hilfe der ersten drei Attribute wurden Muster in den Daten eingearbeitet, die bekannte Tatsachen widerspiegeln: Junge Fahrer verursachen oft Unfälle, besonders wenn sie auf dem Land leben. Entgegen der Meinung, dass mit zunehmendem Alter die Unfallwahrscheinlichkeit sinkt, haben Personen im Alter zwischen 46 und 50 Jahren ein erhöhtes Unfallrisiko. Der Grund besteht darin, dass dieser Personenkreis häufig Kinder hat, die zwischen 18 und 25 Jahren alt sind. Eltern leihen das auf sie angemeldete Auto oft ihren Kindern, die ein erhöhtes Unfallrisiko haben. Dadurch ergibt sich in den Daten der Versicherung für Menschen im Alter von 46 bis 50 Jahren ebenfalls ein erhöhtes Unfallrisiko.
Der erzeugte Datensatz bestand aus 200000 Instanzen und wurde mit dem in Abbildung 6.3 dargestellten BiasTree erzeugt. Zum besseren Verständnis wurden die Kanten direkt mit den Namen der Ausprägungen beschriftet anstatt die zugehörigen Indizes zu verwenden.
Zunächst wurde mit einem entarteten BiasTree die apriori Wahrscheinlichkeit für einen Unfall auf 0.2 gesetzt. Der Anteil der Personen, die schon einen Unfall verursacht haben, ist gering. Mit drei weiteren entarteten BiasTrees wurden die Verteilungen der Attribute ‚Alter' , ‚Stadtbewohner' und ‚Kinder' vorgegeben. Die Verteilung des Alters wurde so gewählt, dass junge Personen ebenso wie Personen zwischen 46 und 50 Jahren häufig vorkommen. Stadtbewohner kommen unter den Kunden der Versicherung häufiger vor als Menschen vom Land. Für die Anzahl der Kinder gilt, dass Personen mit keinem, einem oder zwei Kindern am wahrscheinlichsten sind. Mit einem weiteren BiasTree wurden bezüglich des Zielattributes Muster in die Daten eingearbeitet:

1. Junge Fahrer haben ein erhöhtes Unfallrisiko,

2. bei jungen Fahrern vom Land ist das Risiko für einen Unfall noch höher,

3. Personen im Alter zwischen 46 und 50 Jahren mit einem Kind haben ein erhöhtes Unfallrisiko und

4. bei kinderlosen Personen im Alter von 46 bis 50 Jahren ist das Unfallrisiko geringer als beim Rest der Kundschaft.

Zu beachten ist, dass die Subgruppe aller Personen im Alter von 46 bis 50 Jahren kaum interessant ist. Die erhöhte bzw. geringere Unfallwahrscheinlichkeit der Personen mit einem Kind bzw. der kinderlosen Personen in der gesamten Subgruppe gleichen sich in

etwa aus[1].
Der Iterating GSS Algorithmus wurde auf diese Daten angewandt. Als Parameter wurden $\delta = 0.1$, $\epsilon = 0.04$ und die Weighted Relative Accuracy als Nutzenfunktion gewählt. Die Approximation durch die Normalverteilung wurde ab 100 abgedeckten Beispielen verwendet und Schrittgröße 1000 gewählt. Die Anzahl der Iterationen betrug zehn und wurde vor der Durchführung aller Iterationen abgebrochen, falls der Algorithmus keine sinnvollen Regeln mehr fand. Der minimale Nutzen für Pruning und Bewertung der Güte einer Regel betrug 0.01. Als Vergleichskriterium wurde der geschätzte Nutzen der Regel verwendet. Die Variationen in den Experimenten bestanden in der Wahl der Komplexität der Regeln. Zuerst wurden nur Regeln der Form $r \to Y_+$ betrachtet, die Subgruppen mit erhöhtem Unfallrisiko beschreiben. Bei Regellänge eins fand der Algorithmus die folgenden Regeln:

WENN (Alter=18-25) DANN (Unfall=ja),
WENN (Kinder=1) DANN (Unfall=ja),
WENN (Stadtbewohner=nein) DANN (Unfall=ja).

Es wurden die ‚Bestandteile' der Muster in den Daten gefunden. Die Subgruppe der Personen im Alter von 46 bis 50 Jahren wurde nicht als interessant befunden.
Bei zusätzlicher Betrachtung aller Regeln der Form $r \to Y_-$ wurde als vierte Regel

WENN (Kinder=0) DANN (Unfall=nein)

gefunden. Die Verwendung des veränderten Hypothesenraumes ermöglicht das Auffinden dieser zusätzlichen Subgruppe mit großem Anteil von Beispielen der negativen Klasse. Die Accuracy des erzeugten Ensembles betrug in beiden Fällen 73.10%. Es ergab sich durch den veränderten Hypothesenraum in dieser Hinsicht keine Verbesserung. Bei einer Erhöhung der maximalen Regellänge auf zwei fand der Algorithmus zusätzlich die Regel

WENN (Alter=46-50) UND (Kinder=0) DANN (Unfall=nein).

Die Accuracy des Ensembles aus den fünf Regeln stieg auf 73.91%. Schon an dieser Stelle deutet sich an, dass für die Lernaufgabe Konzeptlernen aus Beispielen andere Verfahren besser geeignet sind. Ein mit dem J48-Verfahren induzierterer Entscheidungsbaum erreichte eine Accuracy von 75.35%. Besteht das Interesse nur in der Entdeckung von Subgruppen, kann von der Möglichkeit Gebrauch gemacht werden, die Beispielgewichte bei Erhöhung der Komplexität wieder auf eins zu setzen. Dadurch konnten alle Subgruppen der Komplexität zwei gefunden werden:

WENN (Alter=18-25) DANN (Unfall=ja),
WENN (Stadtbewohner=nein) DANN (Unfall=ja),
WENN (Kinder=1) DANN (Unfall=ja),

[1] Die Subgruppe aller Personen im Alter von 46 bis 50 Jahren wäre völlig uninteressant, wenn sie einen Bias von null hätte. Durch die Wahl der Unfallwahrscheinlichkeiten der beiden enthaltenen Subgruppen ist diese in der gesamten Subgruppe leicht erhöht. Durch alle im BiasTree angegebenen Wahrscheinlichkeiten beträgt die apriori Wahrscheinlichkeit nicht genau 0.2, sondern ist etwas höher. Insgesamt ergibt sich damit ein Bias nahe null für die Subgruppe der 46 bis 50-jährigen Kunden

Name	Attribute	Beispiele	Nominal	Numerisch	Klassen	Positive
Covtype	54	581012	0	54	7	
Covtype1	54	581012	0	54	2	36,5%
Covtype2	54	581012	0	54	2	48,7%
Covtype3	54	581012	0	54	2	6,1%
Covtype4	54	581012	0	54	2	0,4%
Covtype5	54	581012	0	54	2	1,6%
Covtype6	54	581012	0	54	2	3,0%
Covtype7	54	581012	0	54	2	3,5%
Adult	14	48842	8	6	2	24,1%
Quantenphysik	71	50000	0	71	2	50%

Tabelle 6.1: Eigenschaften der drei verwendeten Datensätze: Name, Gesamtanzahl von Attributen sowie Anzahl von nominalen und numerischen Attributen, Größe der Beispielmenge und Anteil von positiven Beispielen

WENN (Kinder=0) DANN (Unfall=nein),
WENN (Alter=18-25) UND (Stadtbewohner=nein) DANN (Unfall=ja),
WENN (Alter=18-25) UND (Stadtbewohner=ja) DANN (Unfall=ja),
WENN (Alter=46-50) UND (Kinder=1) DANN (Unfall=ja),
WENN (Alter=46-50) UND (Kinder=0) DANN (Unfall=nein)

Wurde die Verwendung von Knowledge-Bases Sampling deaktiviert, fand der Algorithmus insgesamt 23 Regeln. Viele davon waren nur Spezialisierungen wie

WENN (Alter=18-25) UND (Geschlecht=weiblich) DANN (Unfall=ja).

Die Ergebnisse sind dadurch sehr unübersichtlich gegenüber der kompakten Regelmenge, die beim Einsatz von Knowledge-Based Sampling gefunden wird.

6.2 Experimente mit echten Datensätzen

6.2.1 Datensätze

Zwei der verwendeten Datensätze entstammen dem UCI Machine Learning Repository[1]. Da die GSS und Iterating GSS Algorithmen insbesondere für große Datensätze geeignet sind, wurden die beiden größten Datensätze gewählt. Bei den Covtype Daten handelt es sich um Beschreibungen von Waldflächen. Für eine gegebene Waldfläche mit bestimmten Eigenschaften soll vorhergesagt werden, welcher Baumtyp auf der Fläche wächst. Die Adult Daten beschreiben Personen, anhand deren Eigenschaften vorhergesagt werden soll, ob ihr Einkommen 50000 Dollar pro Jahr überschreitet. Der Quantenphysik Datensatz wurde für den KDD Cup 2004 verwendet und enthält Messdaten, mit denen das nicht näher beschriebene binäre Zielattribut vorhergesagt werden soll. In Tabelle 6.1 sind die wichtigsten Eigenschaften dargestellt. Beim Covtype Datensatz hat das Zielattribut sieben verschiedene Ausprägungen. Da der Iterating GSS Algorithmus nur mit einem binären Zielattribut umgehen kann, wurden sieben neue Datensätze Covtype1 bis Covtype7 mit binärem Zielattribut erzeugt. Jede Klasse des Zielattributs stellt in einem der Datensätze die positive Klasse dar. Die verbleibenden Klassen bilden jeweils die negative Klasse.

6.2.2 Ergebnisse

In diesem Kapitel werden Ergebnisse präsentiert, die zusammen mit den Erkenntnissen aus Kapitel 6.1.2 Antworten auf die Fragen geben, die zu Beginn der Diplomarbeit in Kapitel 3 gestellt wurden. Unter anderem wird zur Bewertung der Qualität der Ergebnisse mit Hilfe des in Kapitel 4.2.4 vorgestellten Verfahrens ein Ensembleklassifikator aus den mit dem Iterating GSS Algorithmus gefundenen Subgruppen erzeugt. Er wird mit Klassifikatoren verglichen, die mit anderen Methoden gefunden wurden. Ein Verfahren, das zur Zeit häufig für das Konzeptlernen aus Beispielen eingesetzt wird, ist AdaBoost [13]. AdaBoost ist ein Algorithmus, der mehrere mit einem Basislerner gefundene Modelle zu einem Ensemble kombiniert, um die Performanz zu verbessern. Die einzige Bedingung an den Basislerner ist, dass seine Vorhersagen besser sind als zufällig erzeugte. Die einzelnen Modelle des Basislerners werden iterativ erzeugt. Nach jeder Iteration werden die Gewichte falsch klassifizierter Instanzen erhöht, um den Basislerner zu ‚zwingen', mehr Wert auf die Korrektheit dieser Instanzen zu legen. Eine beliebte Wahl für den Basislerner ist der *Decision Stump* [33]. Bei einem Decision Stump handelt es sich um einen entarteten Entscheidungsbaum, der nur aus einem Knoten besteht. AdaBoost in Kombination mit dem Basislerner Decision Stump liefert in der Praxis oft gute Ergebnisse. Ada^{2}Boost [27] benutzt eine Variation des Knowledge-Based Sampling, um nach jeder Iteration die neuen Beispielgewichte zu bestimmen. Im Folgenden wird zunächst die Performanz des Iterating GSS Algorithmus im Vergleich zu AdaBoost und Ada^{2}Boost jeweils mit dem Decision Stump als Basislerner untersucht. Dazu wurde die Lernumgebung YALE [21] verwendet, die diese drei Verfahren zur Verfügung stellt. Desweiteren erfolgen Untersuchungen der erzielten Ergebnisse für einige Variationen des Iterating GSS Algorithmus. Variiert wurden die Schrittgröße, die Nutzenfunktion und die Komplexität des verwendeten Hypothesenraumes. Numerische Attribute wurden für alle Verfahrem vor der Durchführung des eigentlichen Experimentes mit Recursive Minimal Entropy Partitioning diskretisiert. Zur Überprüfung der Ergebnisse wurde eine 10-fache Kreuzvalidierung benutzt. Die Parameter $\epsilon = 0.04$, $\delta = 0.1$, $large = 100$ und $stepsize = 100$ des Iterating GSS Algorithmus wurden für alle Versuche identisch gewählt, sofern nichts anderes erwähnt wird. Es wurden stets alle Iterationen erzwungen und Knowledge-Based Sampling eingesetzt. Um die Kombination der einzelnen Regeln zu einem Ensemble zu erlauben, erfolgte kein Zurücksetzen der Gewichte auf eins bei Erhöhung der Komplexität.

In Abbildung 6.4 sind die Entwicklung von Accuracy und Laufzeit bei der Erzeugung eines Ensembleklassifikators mit AdaBoost und Ada^{2}Boost unter Verwendung von DecisionStumps als Basislerner und dem Iterating GSS Algorithmus mit der Nutzenfunktion Weighted Relative Accuracy für verschiedene Datensätze dargestellt. Die Regellänge für den Iterating GSS Algorithmus wurde auf eins beschränkt. Aus technischen Gründen ist die Laufzeit von AdaBoost nicht eingezeichnet, sie ist aber vergleichbar mit der Laufzeit von Ada^{2}Boost. Für den Iterating GSS Algorithmus sind jeweils vier Kurven angegeben.

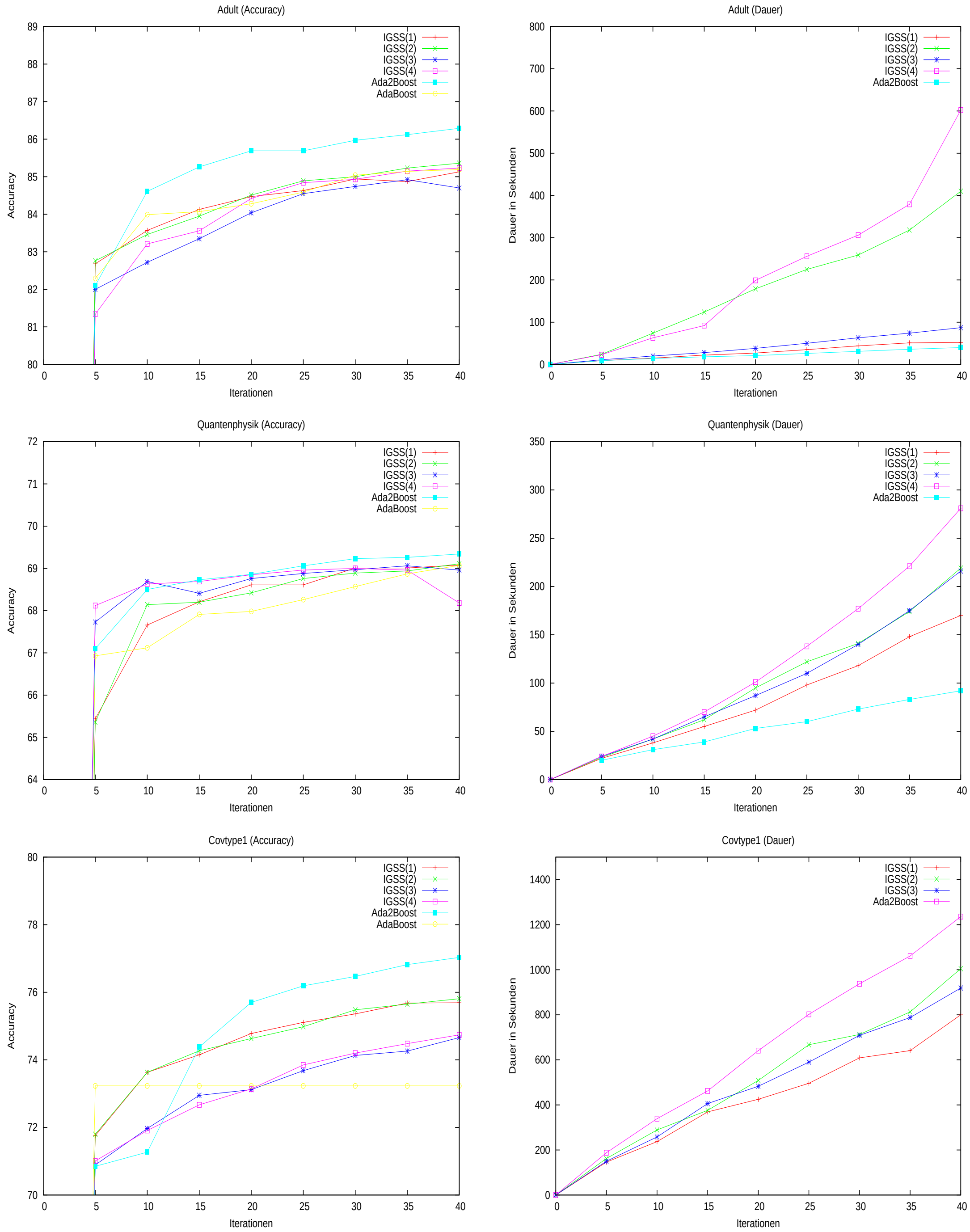


Abbildung 6.4: Accuracy und Laufzeit von AdaBoost, Ada^{2}Boost und von vier Iterating GSS Varianten mit Weighted Relative Accuracy als Nutzenfunktion und auf eins beschränkter Regellänge: (1) unnormierte Gewichte und Regeln $r \to Y_+$, (2) Verwerfungsmethode und Regeln $r \to Y_+$, (3) unnormierte Gewichte und alle Regeln, (4) Verwerfungsmethode und alle Regeln

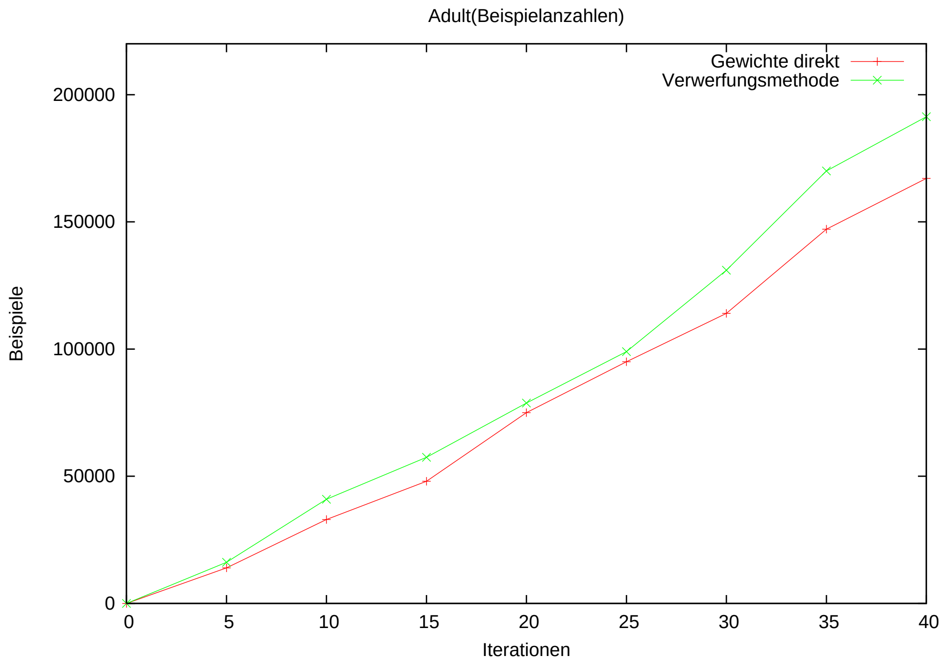


Abbildung 6.5: Vergleich des benötigten Beispielgewichtes bei direkter Verwendung der Gewichte und Einsatz der Verwerfungsmethode auf den Adult Daten und Regellänge eins

Dabei wurde variiert zwischen dem Einsatz der Verwerfungsmethode oder dem direkten Verwenden der Beispielgewichte sowie der Erzeugung aller oder nur der Regeln, die die positive Klasse vorhersagen. Die Komplexität des betrachteten Hypothesenraumes betrug 129 bzw. 258 Regeln für den Adult Datensatz, 79 bzw. 158 Regeln für den Quantenphysik Datensatz und 229 bzw. 458 Regeln für den Covtype1 Datensatz. Es wird deutlich, dass der Iterating GSS Algorithmus bezüglich der erreichten Accuracy des Ensembles konkurrenzfähig zu AdaBoost ist, jedoch schlechter abschneidet als Ada^{2}Boost. Bei den Covtype1 Daten schneidet AdaBoost schlecht ab. Nach der fünften Iteration verbessert sich die Accuracy nicht mehr. Die Erzeugung aller Regeln einer Komplexitätsstufe führt für den Iterating GSS Algorithmus eher zu einer Verschlechterung, was besonders am Covtype1 Datensatz erkennbar ist. Die Verwendung der Verwerfungsmethode führt zu keiner Verbesserung der Accuracy gegenüber der direkten Verwendung der Beispielgewichte. Vorteile in der Laufzeit ergeben sich für den Iterating GSS Algorithmus beim großen Covtype1 Datensatz, wodurch die besondere Eignung für große Datensätze untermauert wird. Besonders gut schneidet dabei die Variante ab, die anstatt der Verwerfungsmethode die unnormierten Gewichte benutzt und nur solche Regeln erzeugt, die die positive Klasse vorhersagen. Bei den Adult Daten fällt auf, dass die Benutzung der Verwerfungsmethode zu einem starken Anstieg der Laufzeit führt. Gleiches gilt für die Covtype1 Daten bei Erzeugung aller Regeln und dem Einsatz der Verwerfungmethode. Hierbei stieg die Laufzeit so stark an, so dass es nicht mehr sinnvoll war, die zugehörige Kurve einzuzeichnen. In Abbildung 6.5 sind das benötigte Beispielgewicht für verschiedene Anzahlen von Iterationen bei Einsatz der Verwerfungsmethode und der Verwendung der unnormierten Gewichte auf den Adult Daten gegenübergestellt. Bei Einsatz der Verwerfungsmetho-

de wird mehr Beispielgewicht benötigt als bei Verwendung der unnormierten Gewichte. Daraus ergibt sich eine Erklärung für die längere Laufzeit durch Betrachtung der Funktionsweise der Verwerfungsmethode. Gibt es in den letzen Iterationen viele Beispiele mit kleinem Gewicht, wird von der Verwerfungsmethode pro akzeptiertem Gewicht von eins eine große Anzahl gezogener Beispiele verworfen. Dieser Zusatzaufwand entfällt bei direkter Verwendung der Beispielgewichte.
Als eine Verbesserung am GSS Algorithmus besteht die Möglichkeit, die Häufigkeit zu ändern, mit der versucht wird gute bzw. schlechte Hypothesen auszugeben bzw. zu verwerfen. Mit dem Parameter Schrittgröße kann diese Häufigkeit geändert werden. In Abbildung 6.6 ist die benötigte Laufzeit auf den Covtype1 Daten für verschiedene Werte der Schrittgröße dargestellt. Als Nutzenfunktion wurde die Weighted Relative Accuracy verwendet. Es wurden nur Regeln der Komplexität eins zugelassen. Es wird deutlich, dass die Erhöhung der Schrittgröße zu einer Verkleinerung der Laufzeit führt. Ab einer Schrittgröße von 100 kommt es zu keiner wesentlichen Verbesserung mehr.
Ein Vergleich der Performanz des Iterating GSS Algorithmus bei Verwendung verschiedener Nutzenfunktionen befindet sich in Abbildung 6.7. Die Accuracy wird auf zwei Arten eingesetzt. Zum einen wird sie vom Iterating GSS Algorithmus als Nutzenfunktion verwendet, um die beste Regel in jeder Iteration zu bestimmen. Zum anderen wird sie benutzt, um die Performanz des erzeugten Ensembles für den jeweiligen Datensatz zu messen. Es wurden direkt die Beispielgewichte verwendet und nur solche Regeln erzeugt, die die positive Klasse vorhersagen. Außerdem wurde die Regellänge auf eins beschränkt und als Kriterium zum Wechsel der Nutzenfunktion von Weighted Relative Accuracy zu Binomial der geschätzte Nutzen ohne Berücksichtigung der Konfidenzintervalle verwendet.
Es fällt auf, dass die Nutzenfunktion mit quadrierter Coverage (Squared) bei beiden Datensätzen zu schlechten Ergebnissen führt. Für den Adult Datensatz ergibt sich für die Nutzenfunktion Accuracy bei den späteren Iterationen keine Verbesserung. Da für den Quantenphysik Datensatz die apriori Wahrscheinlichkeit für die positive Klasse 0.5 beträgt, sind die Ergebnisse für Weighted Relative Accuracy und Accuracy identisch. Besonders bei Betrachtung der Ergebnisse für den Adult Datensatz wird ersichtlich, dass die Nutzenfunktion auf Basis des Binomialtests (Binomial) am besten abschneidet. Sie liefert nach wenigen Iterationen gute Ergebnisse. Gut schneidet die Kombination von Weighted Relative Accuracy und Binomial ab. In den ersten Iterationen ist die von der Kombination erreichte Accuracy identisch mit der bei Verwendung von Weigthed Relative Accuracy erreichten. Nach dem Wechsel nähert sie sich der mit Binomial erreichten Accuracy an. Der Vorteil dieser Kombination wird deutlich bei Betrachtung der Laufzeiten für die verschiedenen Nutzenfunktionen beim Adult Datensatz in Abbildung 6.8. Der Nachteil von Binomial liegt in der längeren Laufzeit bedingt durch eine größere benötigte Stichprobengröße. Durch die Kombination von Weighted Relative Accuracy und Binomial ergibt sich bei gleicher Vorhersagequalität eine wesentliche Verkürzung der Laufzeit. Mit Abstand die kürzeste Laufzeit ergibt sich, wenn Weighted Relative Accuracy oder Accuracy als Nutzenfunktion benutzt wird. Dabei wird die Vorhersagequalität für die Weighted Relative Accuracy nur bei Verwendung von Binomial überboten.

Für die Kombination mehrerer Regeln zu einem Ensemble ist die Qualität der einzelnen Vorhersagen von Bedeutung. Ebenso bedarf es einer Diversität unter den Vorhersagen. Mit Diversität ist gemeint, dass Aussagen über verschiedene Teile des Instanzenraumes

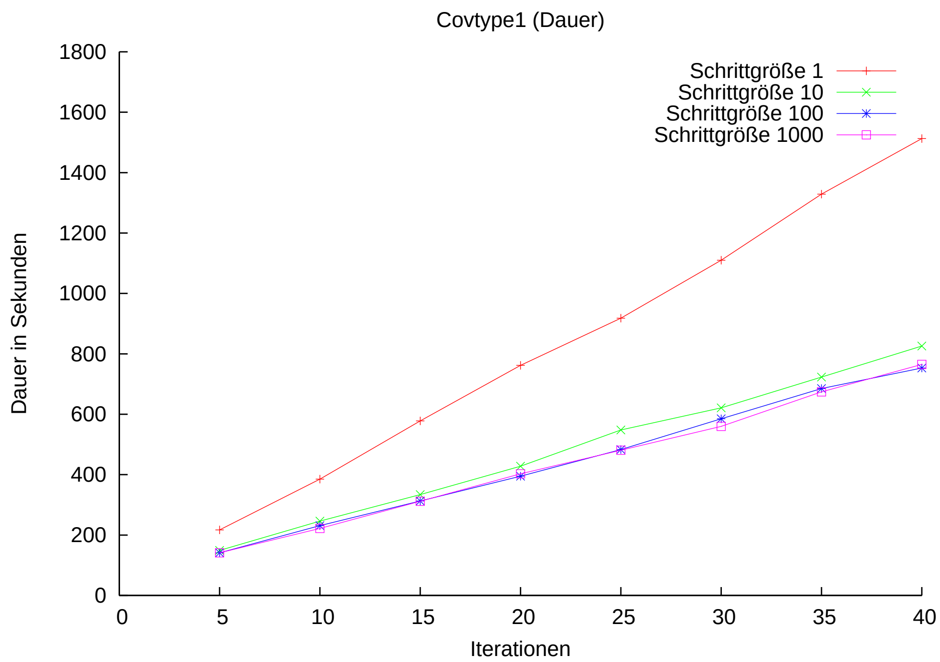


Abbildung 6.6: Laufzeit des Iterating GSS Algorithmus mit verschiedenen Schrittgrößen für den Covtype1 Datensatz mit der Nutzenfunktion Weighted Relative Accuracy und auf eins beschränkter Regellänge

getroffen werden [5]. Sagt eine Regel die positive Klasse des Zielattributes vorher, trifft sie eine Aussage über die Teile des Instanzenraumes, für die sie anwendbar ist[1].
Für die Adult und Quantenphysik Datensätze sind in Abbildung 6.9 für ein Ensemble aus 20 Regeln, die mit verschiedenen Nutzenfunktionen vom Iterating GSS Algorithmus gefunden wurden, die relativen Häufigkeiten für die 1 bis 10-fache Abdeckung eines Beispiels angegeben. Man sieht deutlich, dass Binomial mit einer Ausnahme für diese Daten die schlechteste Abdeckung aufweist, während Squared viele Beispiele mehrfach abdeckt. Bei Accuray und Weighted Relative Accuracy sind die Werte in etwa gleich, da die apriori Wahrscheinlichkeit 0.5 beträgt und Weighted Relative Accuracy und Accuracy in diesem Fall identische Ergebnisse liefern. Für den Adult Datensatz ist der Wert der Accuracy schlecht. Ansonsten entsprechen die Ergebnisse denen des Quantenphysik Datensatzes.
In Abbildung 6.10 ist für die Adult und Quantenphysik Datensätze und verschiedene Nutzenfunktionen die ‚Unreinheit' in den Vorhersagen der durch den Iterating GSS Algorithmus gefundenen Regeln dargestellt. Dazu wurde für jedes Beispiel die Entropie in den Vorhersagen gemessen. Aus den Entropien in den Vorhersagen für die einzelnen Beispiele wurde der Mittelwert bestimmt. Für den betrachteten Fall von zwei Klassen liegt dieser Wert zwischen null und eins. Eins bedeutet eine maximale Unreinheit in den Vorhersagen, während bei einem Wert von null alle Regeln stets die gleiche Vorhersage

[1] Eine Regel sagt für die Beispiele, auf die sie nicht anwendbar ist, implizit die negative Klasse voraus. Nutzenfunktionen wie die Weighted Relative Accuracy können kleine Teile des Instanzenraumes identifizieren, in denen die Wahrscheinlichkeit für positive Beispiele hoch ist. Daher ist die Aussage über den Teil des Instanzenraumes, für den die Regel nicht anwendbar ist, als weniger stark einzuschätzen.

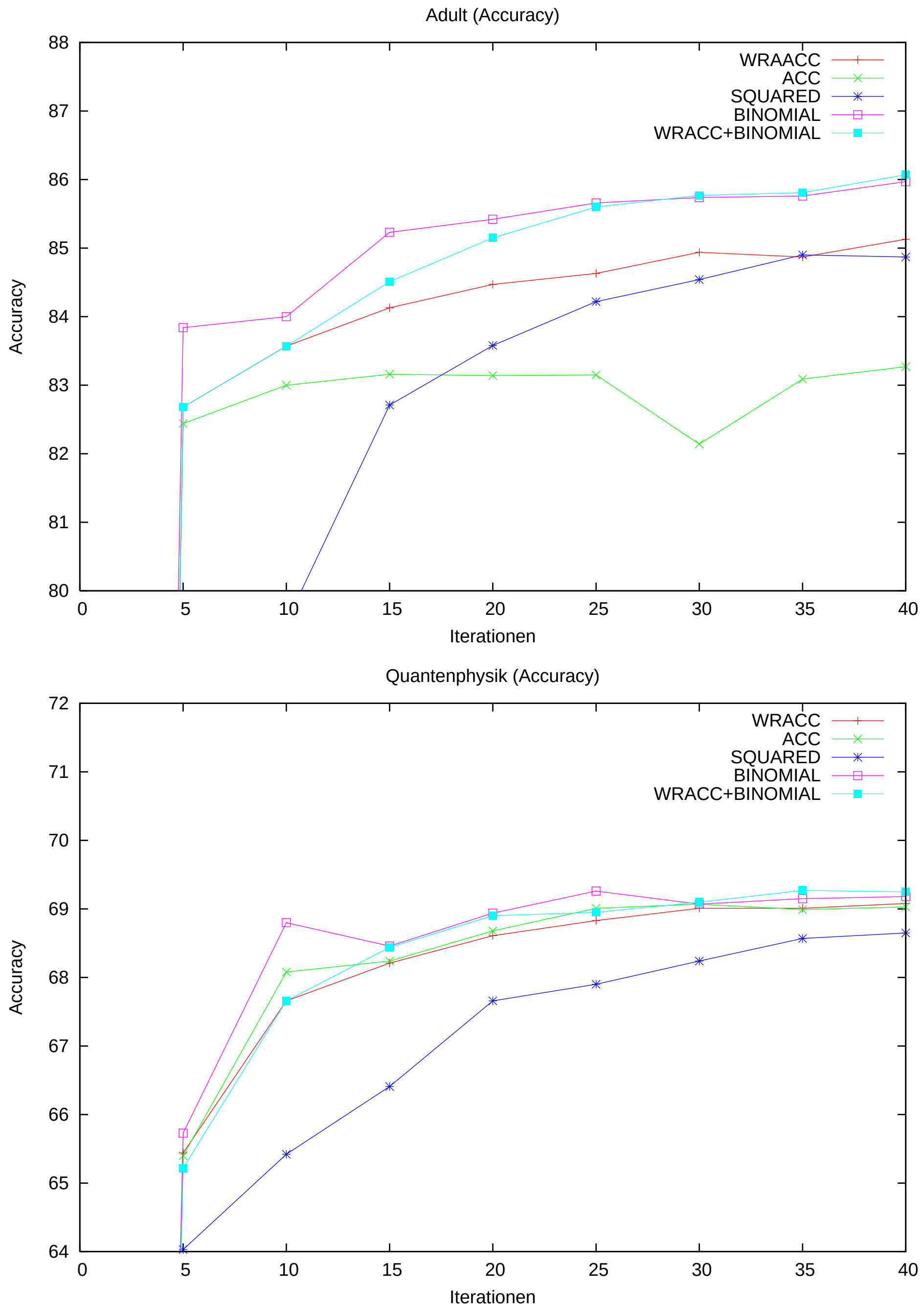


Abbildung 6.7: Vergleich der Accuracy eines mit dem Iterating GSS Algorithmus gefundenen Ensembles bei Einsatz verschiedener Nutzenfunktionen für die Adult und Quantenphysik Datensätze

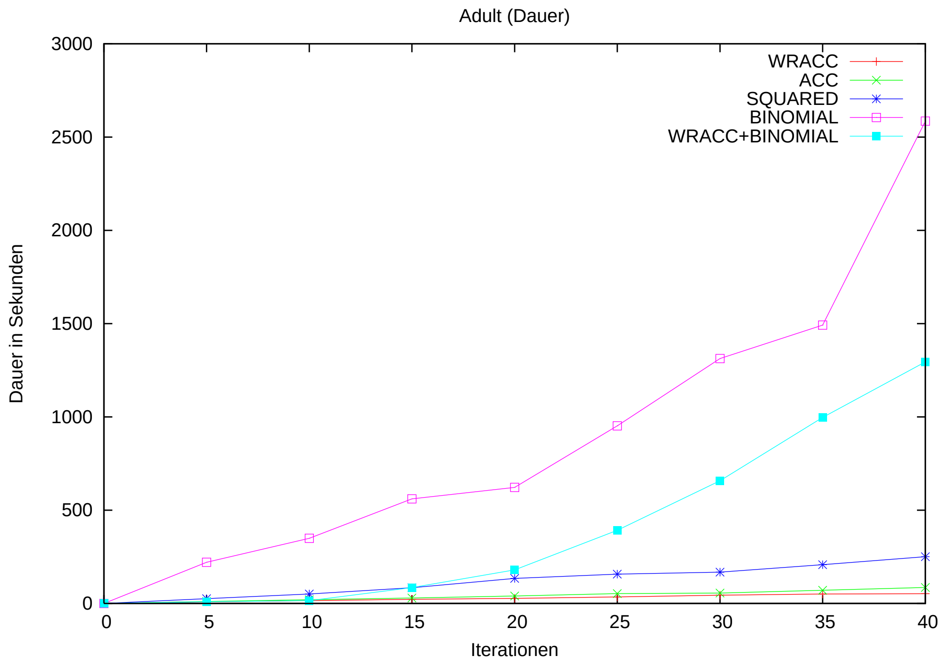


Abbildung 6.8: Vergleich der Laufzeit bei verschiedenen Nutzenfunktionen für den Adult Datensatz

machen. Für Squared ist die Unreinheit hoch. Der hohe Grad der Abdeckung resultiert in vielen widersprüchlichen Vorhersagen, wodurch mit Hilfe dieser Nutzenfunktion gefundene Ensembles eine schlechte Performanz haben. Binomial deckt Beispiele nicht so häufig ab wie Squared. Die Vorhersagen sind aber wesentlich eindeutiger. Weighted Relative Accuracy und Accuracy liegen zwischen diesen beiden Extremen. Eine Ausnahme für die Accuracy bildet der Adult Datensatz. Die Vorhersagen sind eindeutig. Der Grund ist eine zu geringe Abdeckung. Dieses resultiert in einer schlechten Performanz der mit Accuracy gefundenen Regeln auf den Adult Daten.
Abschließend ist in Abbildung 6.11 die Performanz des Iterating GSS Algorithmus für fünf unterschiedlich komplexe Hypothesenräume auf den Covtype1 Daten dargestellt. Der Hypothesenraum bei Komplexität 1 bestand aus allen 229 Regeln der Länge eins. Bei Komplexität 2 wurden alle Regeln mit Länge 2 oder kürzer erzeugt. Der Hypothesenraum bestand dabei aus 25188 Regeln. Der Hypothesenraum bei Komplexität 1-2 bestand zunächst aus allen 229 Regeln der Länge eins. Regeln der Länge zwei wurden erst betrachtet, wenn keine guten Regeln der Länge eins mehr gefunden werden konnten. Zur Beurteilung der Güte der gefundenen Regeln wurde der geschätzte Nutzen der Regel ohne Berücksichtigung des Konfidenzintervalles benutzt (Kriterium Nutzen). Der Mindestnutzen, den eine Regel erreichen musste, betrug stets 0.01. Beim Mindestnutzen fürs Pruning q^p_{min} wurden die Werte 0.01, 0.03, 0.05 getestet. Stufte das gewählte Nutzenkriterium eine Regel als zu schlecht ein, wurden durch Pruning mit dem jeweiligen Minimalnutzen alle schlechten Regeln der Länge eins entfernt. Aus den verbliebenen Regeln wurden die neuen der Länge zwei erzeugt. Für die Komplexität des neuen Hypothesenraumes der Komplexität zwei ergaben sich abhängig von q^p_{min} folgende Werte:

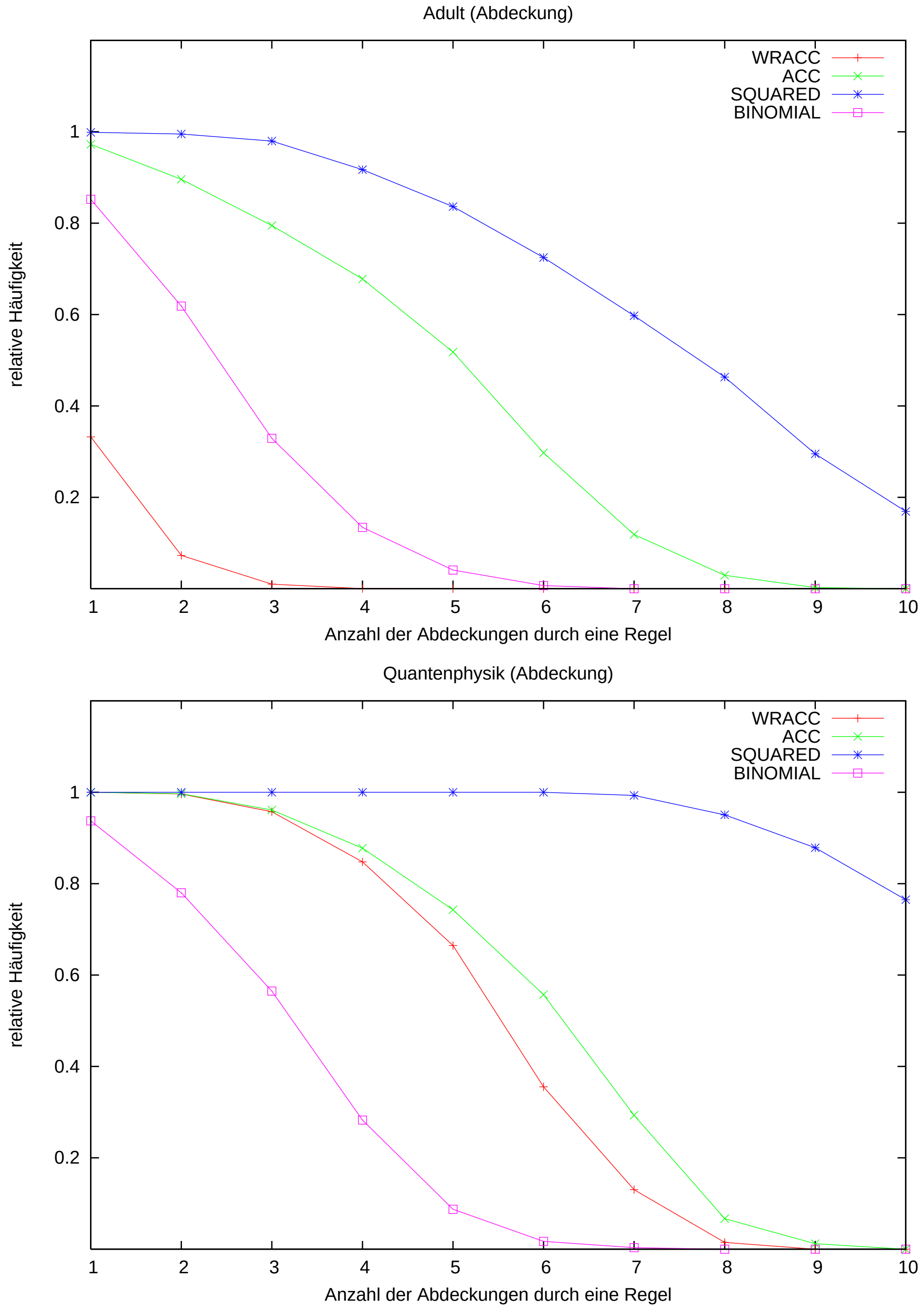
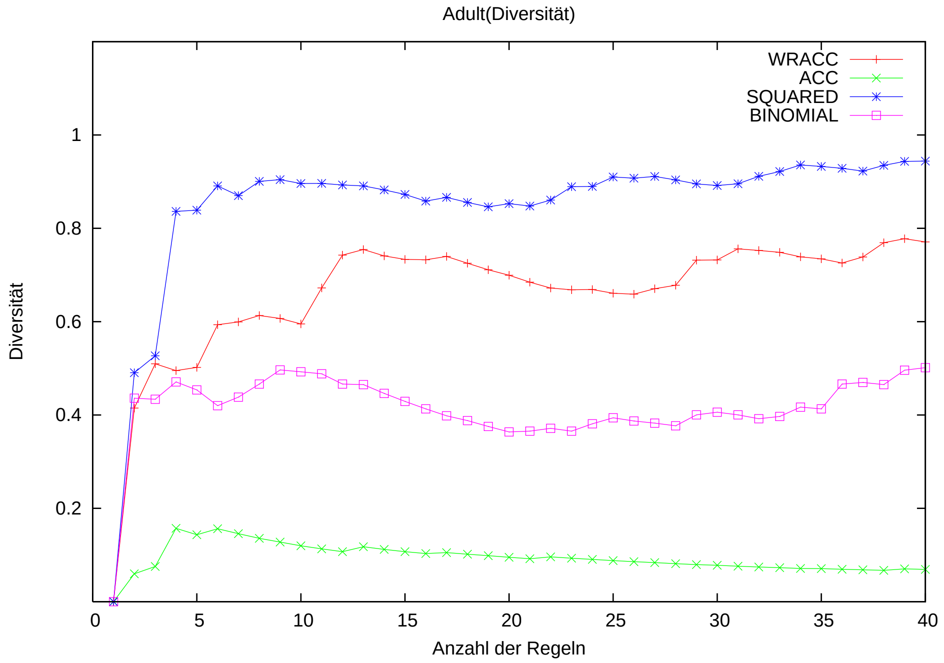


Abbildung 6.9: Relative Häufigkeiten der Anzahl an Abdeckungen von Beispielen durch die Regeln eines Ensembles auf den Adult und Quantenphysik Daten für verschiedene Nutzenfunktionen



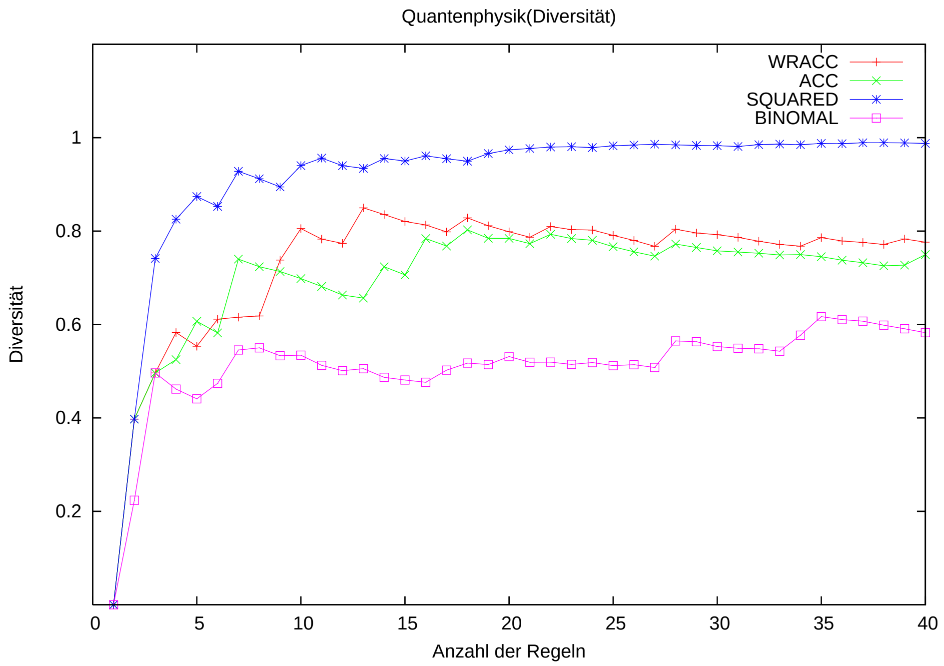


Abbildung 6.10: Diversität eines Ensembles aus unterschiedlichen Anzahlen von verwendeten Regeln auf den Adult und Quantenphysik Daten für verschiedene Nutzenfunktionen

1. 24959 Regeln für $q_{min}^{p} = 0.01$,
2. 16328 Regeln für $q_{min}^{p} = 0.03$ und
3. 6121 Regeln für $q_{min}^{p} = 0.05$.

In Abbildung 6.11 sind die Ergebnisse dargestellt. Es ist ersichtlich, dass die direke Verwendung der Komplexität 2 zu keiner Verbesserung der Accuracy des gefundenen Ensembles führt und die Laufzeit stark ansteigt. Das Durchsuchen des Hypothesenraumes von geringer zu hoher Komplexität verbessert Accuracy und Laufzeit im Vergleich zur direkten Verwendung des Hypothesenraumes aus allen Regeln der Länge zwei. Für kleine Werte von q_{min}^{p}, ergibt sich bei vielen Iterationen eine leicht verbesserte Accuracy gegenüber der bei Beschränkung auf Komplexität eins erreichten. Allerdings bewirkt die Verwendung von Komplexität 1-2 eine Verlängerung der Laufzeit gegenüber Komplexität 1. Das Ausmaß der Erhöhung hängt vom Parameter q_{min}^{p} ab, der bestimmt wie stark das Pruning ausfällt.

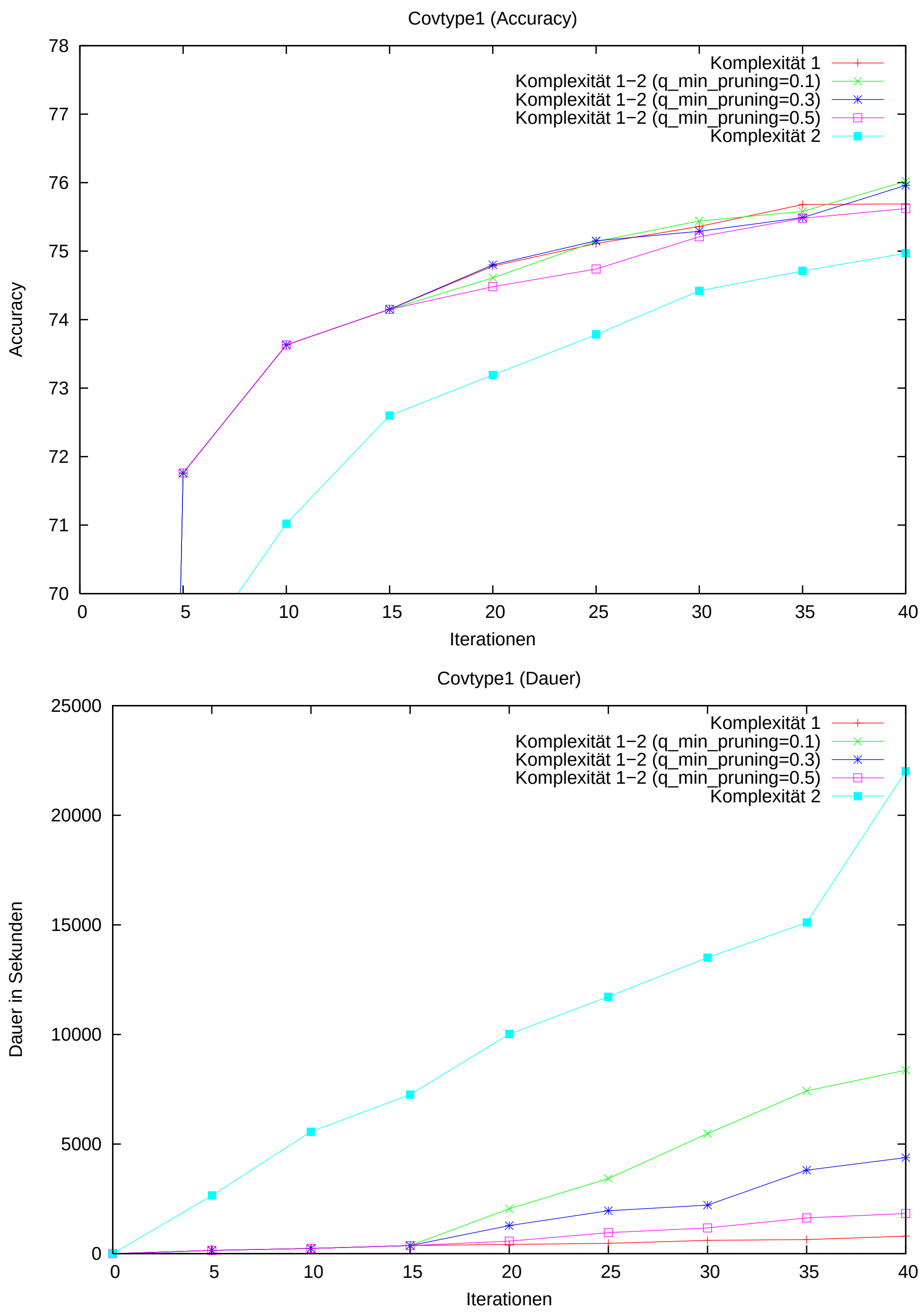


Abbildung 6.11: Vergleich von Accuracy und Laufzeit bei verschiedenen Komplexitätsstufen für den Covtype1 Datensatz

7 Zusammenfassung

Mit dem Iterating GSS Algorithmus steht ein Verfahren zur Lösung der Lernaufgaben Subgruppenentdeckung und Konzeptlernen aus Beispielen zur Verfügung. Es ist der erste Versuch, probabilistische Subgruppenentdeckung in strukturierten Hypothesenräumen mit Knowledge-Based Sampling zu kombinieren. Die Stärken liegen im Auffinden einer kompakten und gleichzeitig aussagekräftigen Menge von Subgruppen und im Umgang mit großen Datenmengen. Während bei kleinen Datensätzen der Zusatzaufwand für Berechnung von Nutzen und Konfidenzintervallen überwiegt, bewirkt der Ansatz des sequentiellen Samplings für große Datensätze eine Verbesserung der Laufzeit. Bedingt ist dieses auch durch die Verringerung der Häufigkeit innerhalb des GSS Algorithmus, mit der versucht wird Hypothesen frühzeitig auszugeben oder zu verwerfen.

Durch Kombination der gefundenen Subgruppen zu einem Ensemble kann die prädiktive Lernaufgabe Konzeptlernen aus Beispielen gelöst werden. Im Vergleich zu anderen Verfahren, die nicht auf Basis einer Stichrobe arbeiten, ist der Ansatz konkurrenzfähig bezüglich der Güte der Klassifikation. Allerdings sind abhängig vom Datensatz andere Verfahren besser für diese Lernaufgabe geeignet.

Eine wichtige Verbesserung des Iterating GSS Algorithmus ist das Durchlaufen des Hypothesenraumes von einfachen zu komplexen Hypothesen in Verbindung mit Pruning. Dadurch ist es möglich, Subgruppen höherer Komplexität zu finden, die zum Verständnis der Lösung beitragen. Für die Kombination der Subgruppen zu einem Ensemble bedeutet eine höhere Komplexität keine deutliche Verbesserung der Güte des Ensembles. Sie resultiert aber in einer wesentlichen Verschlechterung der Laufzeit. Für die Lernaufgabe Konzeptlernen aus Beispielen bietet sich die Beschränkung auf eine kleine Komplexität an.

Ein Vergleich der verschiedenen Nutzenfunktionen zeigte, dass die Funktion Binomial am besten geeignet ist, um nach wenigen Iterationen gute Subgruppen zu finden und ein gutes Ensemble zu konstruieren. Die Nutzenfunktion Squared hat sich als schlechte Wahl erwiesen. Die Laufzeit bei Verwendung von Binomial ist bedingt durch die großen Schranken für die Konfidenzintervalle schlecht. Bei Erhöhung der Anzahl von Iterationen wurde insbesondere mit der Nutzenfunktion Weighted Relative Accuracy eine gute Vorhersagequalität in Verbindung mit einer kurzen Laufzeit erzielt. Soll aus den gefundenen Subgruppen ein Ensemble zur Klassfikation erzeugt werden, empfiehlt sich bei Berücksichtigung der Laufzeit die Verwendung der Weighted Relative Accuracy als Nutzenfunktion und die Beschränkung der Komplexität auf Regellänge eins.

Für weitere Untersuchungen bietet sich an, die Eignung des Algorithmus im Umgang mit Datenströmen näher zu untersuchen.

Literaturverzeichnis

[1] C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases.* Dept. of Information and Computer Sciences, University of California - Irvine, Irvine, CA, USA, 1998. http://www.ics.uci.edu/ mlearn/MLRepository.html.

[2] Tim Bray and C. M. Sperberg-McQueen. Extensible Markup Language (XML): Part I. syntax. Technical report, World Wide Web Consortium, 1997. http://www.w3.org/pub/WWW/TR/WD-xml-lang-970331.html.

[3] William W. Cohen. Efficient pruning methods for separate–and–computer rule learning systems. In Ruzena Bajcy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI 93*, pages 988–994, San Mateo, CA, 1993. Morgan Kaufmann.

[4] W.W. Cohen. Fast effective rule induction. In *Proccedings of the Twelfth International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann.

[5] Pádraig Cunningham and John Carney. Diversity versus Quality in Classification Ensembles Based on Feature Selection. In Ramon López de Mántaras and Enric Plaza, editors, *Proceedings of the 11th Conference on Machine Learning (ECML 2000)*, volume 1810 of *LNCS*, pages 109 – 116. Springer Verlag Berlin, Barcelona, Spain, 2000.

[6] Stefan Dißmann and Ernst-Erich Doberkat. *Einführung in die objektorientierte Programmierung mit JAVA.* Oldenbourg, 1998.

[7] Pedro Domingos. Linear-time rule induction. In *Knowledge Discovery and Data Mining*, pages 96–101, 1996.

[8] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In Armand Prieditis and Stuart J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, California, USA, 1995. Morgan Kaufmann.

[9] L. Fahrmeir, R. Künstler, I. Pigeot, and G. Tutz. *Statistik - Der Weg zur Datenanalyse, vierte, verbesserte Auflage.* Springer-Verlag, Berlin, 2003.

[10] Usama M. Fayyad and Keki B. Irani. Multi–interval discretization of continuous–valued attributes for classification learning. In Ruzena Bajcy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI 93*, pages 1022–1029, San Mateo, CA, 1993. Morgan Kaufmann.

[11] Simon Fischer, Ralf Klinkenberg, Ingo Mierswa, and Oliver Ritthoff. YALE: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany, June 2002. ISSN 1433-3325. http://yale.sf.net/ .

[12] Yoav Freund. Self bounding learning algorithms. In *Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[13] Yoav Freund and Robert R. Schapire. A decision–theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[14] Johannes Fürnkranz and Peter Flach. ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58(1):39–77, 2005.

[15] Joao Gama, Luis Torgo, and Carlos Soares. Dynamic discretization of continuous attributes. In *IBERAMIA*, pages 160–169, 1998.

[16] G. Görz, J. Schneeberger, and C. Rollinger. *Handbuch der künstlichen Intelligenz.* Oldenbourg, 2000.

[17] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, (58):13–30, 1963.

[18] G. Tesauro Jack D. Cowan and J. Alspector, editors. *Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation*, volume 6, 340 Pine Street, 6th Fl., San Francisco, CA 94104, April 1994. Morgan Kaufmann.

[19] Willi Klösgen. *Advances in Knowledge Discovery and Data Mining*, chapter Explora: A multipattern and multistategey discovery assistant, 12, pages 249–271. AAAI/MIT Press, Cambridge, USA, 1996.

[20] D.J.C. Mackay. Introduction To Monte Carlo Methods. In *Learning in Graphical Models*, pages 175–204. 1998.

[21] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. YALE: Rapid Prototyping for Complex Data Mining Tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*. ACM Press, 2006.

[22] Tom M. Mitchell. *Machine Learning.* McGraw Hill, New York, 1997.

[23] John Ross Quinlan. *C4.5: Programs for Machine Learning.* Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[24] Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach.* Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey, 2 edition, 2003.

[25] Tobias Scheffer and Stefan Wrobel. Finding the Most Interesting Patterns in a Database Quickly by Using Sequential Sampling. *Journal of Machine Learning Research*, 3:833–862, 2002.

[26] Martin Scholz. Knowledge-Based Sampling for Subgroup Discovery. In Katharina Morik, Jean-Francois Boulicaut, and Arno Siebes, editors, *Local Pattern Detection*, volume LNAI 3539 of *Lecture Notes in Artificial Intelligence*, pages 171–189. Springer, 2005.

[27] Martin Scholz. Boosting in PN Spaces. In *Proceedings of the 17th European Conference on Machine Learning (ECML-06)*. Springer, 2006.

[28] Martin Scholz. *Scalable and Accurate Knowledge Discovery in Real-World Databases*. PhD thesis, Fachbereich Informatik, Universität Dortmund, 2006.

[29] Martin Scholz and Ralf Klinkenberg. Boosting Classifiers for Drifting Concepts. *Intelligent Data Analysis (IDA), Special Issue on Knowledge Discovery from Data Streams (accepted for publication)*, 2006.

[30] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. Chapman and Hall, 4 edition edition, September 1969.

[31] R. Tibshirani. Bias, variance and prediction error for classification rules, 1996.

[32] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

[33] Ian Witten and Eibe Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

[34] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In Jan Komorowski and Jan Zytkow, editors, *Proc. First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, Berlin, 1997. Springer Verlag.

[35] Michael Wurst and Martin Scholz. Distributed Subgroup Discovery. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-06)*, Berlin, Germany.

A Verzeichnis der verwendeten Notationen

Im Folgenden sind die benutzten Notationen aufgelistet, gestaffelt nach dem Themenbereich ihres erstmaligen Auftretens.

X	Instanzenraum
N	Anzahl der Attribute im Instanzenraum
x	eine Instanz aus dem Instanzenraum
D	Verteilung der Instanzen über dem Instanzenraum
T	eine Trainingsmenge von Instanzen
n	die Größe der Trainingsmenge
H	der betrachtete Hypothesenraum
h	eine einzelne Hypothese
f	das gesuchte Zielkonzept
q	Nutzenfunktion
$Y_+/Y-$	Menge der positiven/negativen Instanzen
Y_*	Menge der Instanzen der interessierenden Klasse
$\bar{Y_*}$	Menge der Instanzen, die nicht in der interessierenden Klasse liegen
$r/\bar{r}$	Menge der von einer Regel abgedeckten/nicht abgedeckten Instanzen
R	Regel aus Prämisse und Konklusion
$\mathcal{P}_\rangle$	ein Literal in der Prämisse einer Regel
$\mathcal{K}$	das Literal in der Konklusion einer Regel

Tabelle A.1: Allgemeine Notationen

k	gesuchte Anzahl der Lösungen
m	bisher gesehene Beispielanzahl
M	maximal benötigte Beispielanzahl bis zur Terminierung
Q_m	Stichprobe der Größe m
q(h,T)/$\hat{q}(h, Q_m)$	wahrer/geschätzter Wert einer Nutzenfuktion q
ϵ	maximal zulässiger Fehler
δ	maximale Irrtumswahrscheinlichkeit
$E(m, \delta)$	von der Beispielanzahl m und δ abhängige Konfidenzschranke
$E_h(m, \delta)$	von der Beispielanzahl m und δ abhängige hypothesenspezifische Konfidenzschranke

Tabelle A.2: Notationen für den GSS Algorithmus

$\leftarrow$	Zuweisung an eine Variable
c(R)	Regellänge der Regel R
c_{start}	maximale Regellänge zu Beginn
c_{max}	maximale Regellänge
A_i	i-tes Attribut des Instanzenraumes
a_{ij}	j-te Ausprägung des i-ten Attributes des Instanzenraumes
q^p_{min}	minimaler Nutzen für das Pruning
q^p_{min}	minimaler Nutzen die Nützlichkeit einer Regel
$\mathcal{M}, \mathcal{M}', \mathcal{N}$	Symbole für die Datenstruktur Menge
$\mathcal{L}$	Symbol für die Datenstruktur Liste
$\mathcal{R}$	Symbol für die Datenstruktur Regel
$\mathcal{S}, \mathcal{S}'$	Symbole für die Datenstruktur Resultat
α	das verwendete Nutzenkriterium
β	Schrittgröße
γ	Anzahl der abgedeckten Beispiele einer Regel
	bevor die Normalverteilungsapproximation benutzt wird
ω	gibt an, ob Verwerfungsmethode oder Gewichte benutzt werden
U(R,q,δ,m)	obere Schranke für den Nutzen der Regel R,
	die Nutzenfunktion q und Konfidenz, nach m Beispielen δ
R'_{opt}	optimaler Nachfolger einer Regel R
$E(R, q, \delta, m)$	Konfidenzintervall für die Regel R
	die Nutzenfunktion q, Konfidenz δ, nach m Beispielen

Tabelle A.3: Zusätzliche Notationen für den Iterating GSS Algorithmus und den Biased-ExampleGenerator Operator

B Benutzte primitive Datentypen und -strukturen

Name	Beschreibung	Operationen
Boolean	boolescher Datentyp (true/false)	-
Integer	Ganzzahliger Datentyp	-
Double	Fließkommazahl	-
Menge<E>	Menge aus Elementen vom Typ E	ADD($\mathcal{M}$,E) GET($\mathcal{M}$,E) REMOVE($\mathcal{M}$,E) SIZE($\mathcal{M}$) ADD-ALL($\mathcal{M}$,$\mathcal{M}$')
Liste<E>	Liste aus Elementen vom Typ E	ADD-FIRST($\mathcal{L}$,E) ADD-LAST($\mathcal{L}$,E) GET-FIRST($\mathcal{L}$) REMOVE-FIRST($\mathcal{L}$) GET-LAST($\mathcal{L}$) REMOVE-LAST($\mathcal{L}$) SIZE($\mathcal{L}$) APPEND-ALL($\mathcal{L}$,$\mathcal{M}$)
Regel	Konjunktive Regel	LENGTH($\mathcal{R}$) GET-COVERED-WEIGHT($\mathcal{R}$) GET-COVERED-CORRECT-WEIGHT($\mathcal{R}$) REFINE($\mathcal{R}$)
Resultat	Speichert eine Regel mit zugehörigen Statistiken	GET-RULE($\mathcal{S}$) GET-NEEDED-EXAMPLES($\mathcal{S}$) GET-UTIL($\mathcal{S}$) GET-CONF($\mathcal{S}$)
Nutzen	Berechnet Nutzen, Konfidenzintervalle und obere Schranken	UTIL($\mathcal{R}$) CONF($\mathcal{R}$) GET-UPPER-BOUND($\mathcal{S}$,δ)

Tabelle B.1: Übersicht über die verwendeten primitiven Datentypen und -strukturen

C Parameter des Iterating GSS Algorithmus

In Tabelle C.1 sind alle Einstellungsmöglichkeiten des Iterating GSS Algorithmus aufgelistet. Im Folgenden werden die Parameter kurz beschrieben und aufgezeigt, zu welchen in Kapitel 5 vorgestellen Konzepten sie korrespondieren.

Name	mögliche Werte	Standardwert
min_utility_pruning	[0,1]	-
min_utility_useful	[0,1]	-
stepsize	ganze Zahl aus [1,10000]	1000
large	ganze Zahl aus [1,10000]	100
min_complexity	ganze Zahl aus [1,10]	1
max_complexity	ganze Zahl aus [1,10]	1
iterations	ganze Zahl aus [1,100]	10
utility_function	{wracc, accuracy, linear, squared, binomial}	wracc
use_binomial	{true, false}	false
use_kbs	{true,false}	false
rejection_sampling	{true, false}	true
useful_criterion	{worst_utility, best_utility, example}	utility
example_factor	[1,5]	1.5
force_iterations	{true, false}	false
generate_all_hypothesis	{true, false}	true
reset_weight	{true, false}	false

Tabelle C.1: Übersicht über die Parameter des Iterating GSS Algorithmus

min_utility_pruning Hierbei handelt es sich um den für das Pruning benötigten Minimalnutzen q_{min}. Details sind in Kapitel 5.4 zu finden.

min_utility_useful Zur Bewertung der Nützlichkeit einer Regel (5.3) kann mit diesem Parameter einer anderer Mindestnutzen als für das Pruning gewählt werden. Während beim Pruning ein niedriger Wert des Mindestnutzen zu einem starken Anwachsen des Hypothesenraumes führt, ist bei der Bewertung der Nützlichkeit die Wahl eines niedrigen Wertes weniger kritisch.

stepsize Der Parameter gibt Anzahl von gezogenen Beispielen an, nach denen jeweils Schritt 3d des GSS Algorithmus ausgeführt wird.

large Hiermit wird die Anzahl von Beispielen angegeben, die eine Regel abdecken muss, bevor die Approximation durch die Normalverteilung bei der Berechnung des Konfidenzintervalles benutzt wird.

min_complexity und max_complexity Diese Parameter entsprechen den in Kapitel 5.5 vorgestellten c_{start} und c_{max}, die die initiale und maximale Komplexität der betrachteten Hypothesen angeben. Als Standard sind Anfangs- und Endkomplexität auf eins gesetzt, d.h. es werden nur Regeln der Länge eins betrachtet. Es ist möglich die Komplexität bis zu einer Regellänge von zehn zu erhöhen.

iterations Hiermit wird die Anzahl der Iterationen bzw. die Anzahl der Lösungen festgelegt. Standardmäßig führt der Algorithmus zehn Iterationen durch.

utility_function Mit diesem Parameter wird eine der Nutzenfunktionen Accuracy, Weighted Relative Accuracy, Linear, Squared oder Binomial gewählt. Als Standard ist die Weighted Relative Accuracy eingestellt. Bei der Nutzenfunktion Linear handelt es sich um die Weighted Relative Accuracy, wobei zur Berechnung des Konfidenzintervalles die schlechteren Schranken aus [25] verwendet werden.

use_binomial Bei Aktivierung dieser Option wird vor der Erhöhung der Komplexität auf die Nutzenfunktion Binomial umgeschaltet.

use_kbs Standardmäßig wird nach jeder Iteration durch Veränderung der Beispielgewichte das durch die gefundene Regel repräsentierte Vorwissen aus den Daten entfernt. Mit diesem Parameter kann die Umgewichtung verhindert werden. Der Iterating GSS Algorithmus entspricht dann dem normalen GSS Algorithmus.

rejection_sampling Dieser Parameter gibt an, ob die Verwerfungsmethode oder direkt die Beispielgewichte verwendet werden. Voreingestellt ist die Verwerfungsmethode.

useful_criterion Dieser Parameter bestimmt das Kriterium zur Bewertung der Nützlichkeit einer Regel.

example_factor Falls das Beispielkriterium verwendet wird, erfolgt ein Vergleich der benötigten Beispielanzahl des aktuellen Durchgang des GSS Algorithmus mit der bisher benötigten durchschnittlichen Beispielanzahl aller vorherigen Durchläufe. Dieser Parameter gibt den Faktor an, um den die aktuell benötigte Beispielanzahl vom Durchschnitt abweichen darf bevor die gefundene Regel als nicht nützlich bewertet wird. Falls keine Angabe des Faktors erfolgt, verwendet der Algorithmus den Wert 1.5. Möglich sind Werte aus dem Intervall [1,5].

force_iterations Mit dieser Option kann erzwungen werden, dass der Iterating GSS Algorithmus trotz Eintreten der Terminierungsbedingung alle Iterationen durchgeführt.

generate_all_hypothesis In der Standardversion betrachtet der Iterating GSS Algorithmus nur Regeln der Form $r \rightarrow Y_+$. Durch Aktivierung dieser Option werden auch alle Regeln der Form $r \rightarrow Y_-$ generiert. So ist es möglich ist, Subgruppen mit großem Anteil von Beispielen der negativen Klasse direkt zu finden. Diese Option ist standardmäßig deaktiviert.

reset_weight Wird diese Option aktiviert, werden nach Erhöhung der Komplexität alle Beispielgewichte wieder auf eins gesetzt. In der Grundeinstellung ist sie deaktiviert.

D Parameter des BiasedExampleGenerator Operator

In Tabelle D.2 werden die Parameter des BiasedExampleGenerator Operators beschrieben, die in der Implementierung für die Lernumgebung YALE [21] zur Verfügung stehen. Eine Einführung in den Umgang mit YALE, insbesondere zur Erstellung einer Datei mit Attributbeschreibungen, ist in [11] zu finden. Der Operator benötigt neben den Parametern eine Datei, in der ein BiasTree beschrieben wird. Zur Beschreibung werden zwei Tags verwendet.

<forest> Dieser äußere Tag umschließt alle Beschreibungen von BiasTrees in der Datei.

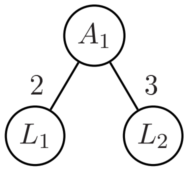
<node> Mit den <node>-Tags werden die Knoten im BiasTree beschrieben. Die Tags werden ineinander verschachtelt und beschreiben dadurch die Struktur des Baumes. Innerhalb des Tags muss der Name des an diesem Knoten repräsentierten Attributes angegeben werden. Die Ausprägungen aller Attribute werden zunächst für jedes Beispiel zufällig ausgewürfelt, d.h. jede der $|A_i|$ Ausprägungen eines Attributes A_i hat die gleiche Wahrscheinlichkeit. Es ist möglich, in jedem <node>-Tag eine alternative Verteilung des dort repräsentierten Attributes anzugeben, die nur bei Erreichen dieses Knoten angewandt wird. Für jede der $|A_i|$ Ausprägungen des Attributes ist eine Wahrscheinlichkeit anzugeben. Die Wahrscheinlichkeiten der einzelnen Ausprägungen wird dabei nicht als Zahl aus dem Intervall [0,1] angegeben, um das Auftreten von Rundungsfehlern zu vermeiden. Stattdessen werden ganze Zahlen aus dem Intervall [0,100] verwendet. Die Summe aller Wahrscheinlichkeiten muss dabei immer 100 ergeben. Ebenfalls optional ist die Angabe der Indizes der Ausprägungen des Attributes, an denen die Kindknoten hängen. Die Indizes können Werte von 0 bis $|A_i|$ annehmen. Der erste Kindknoten des aktuellen Knotens ‚hängt' im BiasTree an der Ausprägung mit dem ersten angegebenen Index; der zweite Kindknoten an der Ausprägung mit dem zweiten angegebenen Index, usw.. Fehlt die Angabe, ist der Knoten ein Blatt. Werden Indizes angegeben, muss die Anzahl an untergeordneten <node>-Tags der Anzahl der Indizes entsprechen.
Ein Beispiel für eine Beschreibung von mehreren BiasTrees befindet sich in Abbildung D.1, die zugehörige XML-Beschreibung in Abbildung D.2. Eine Übersicht über die angesprochenen Attribute des <node>-Tags ist in Tabelle D.1 dargestellt.

Name	Beschreibung	optional?
attribute	Name des Attributes dieses Knotens	Nein
dist	Verteilung des Attributes dieses Knotens	Ja
children	Indizes der Ausprägungen des Attributes, für die in die Kindknoten verzweigt wird.	Ja

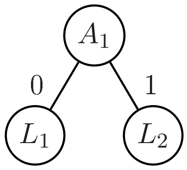
Tabelle D.1: Übersicht über die Attribute des <node>-Tags zur Beschreibung der Knoten des BiasTrees

Attribut	Verteilung über die Ausprägungen
A_1	0.05 0.05 0.2 0.2 0.1 0.05 0.05 0.1 0.1 0.1

Attribut	Verteilung über die Ausprägungen
A_2	0.2 0.2 0.1 0.05 0.05 0.05 0.05 0.1 0.1 0.1



Attribut	Verteilung über die Ausprägungen
A_1	keine neue Verteilung
L_1	0.3 0.7
L_2	0.4 0.6



Attribut	Verteilung über die Ausprägungen
A_2	keine neue Verteilung
L_3	0.6 0.4
L_3	0.5 0.5

Abbildung D.1: Menge von mehreren BiasTrees, die zunächst die Verteilung der Ausprägungen der Attribute angeben und dann Bedingungen für die Ausprägungen des Zielattributes formulieren.

```
<forest>
<node attribute="A1" dist="5 5 20 20 10 5 5 10 10 10">
</node>
<node attribute="A2" dist="20 20 10 5 5 5 5 10 10 10">
</node>
<node attribute="A1" children="2 3">
<node attribute="L1" dist="30 70">
</node>
<node attribute="L1" dist="40 60">
</node>
</node>
<node attribute="A2" children="0 1">
<node attribute="L1" dist="60 40">
</node>
<node attribute="L1" dist="50 50">
</node>
</node>
</forest>
```

Abbildung D.2: XML Beschreibung eines BiasTree

Name	Beschreibung	mögliche Werte	Standard
attributes	Name der Attributbeschreibungsdatei	<Dateiname>	-
number_of_examples	Anzahl der zu erzeugenden Beispiele	[1000,20000000]	10000
tree_file	Name der Datei mit der Beschreibung des Bias Tree	<Dateiname>	-
noise	Wahrscheinlichkeit für das Auftreten von Rauschen	[0,1]	0 0
start_example	Ab diesem Beispiel treten die Muster in den Daten auf. (-1 für alle Beispiele)	[-1,20000000]	0
end_example	Ab diesem Beispiel treten die Muster in den Daten nicht mehr auf. (-1 für alle Beispiele)	[-1,20000000]	0

Tabelle D.2: Übersicht über die Parameter des BiasedExampleGenerator Operators