

KOMPLEXITÄTSTHEORETISCHE UNTERSUCHUNGEN ZU
FALLBASIERTEM LERNEN

Diplomarbeit

vorgelegt von

JAN-MARTIN KUHNIGK

aus

KASSEL

angefertigt am

Institut für Theoretische Informatik der
Medizinischen Universität zu Lübeck

2001

Zusammenfassung

Wir setzen uns intensiv mit dem in [Sat98] und [SN00] untersuchten fallbasierten Modell zum Lernen boolescher Funktionen auseinander. Dabei leiten wir zunächst die dort erhaltenen Resultate aus den Bereichen Repräsentierbarkeit boolescher Funktionen, Minimalität von Fallbasen und PAC-Lernbarkeit polynomiell repräsentierbarer Funktionsklassen unter einem leicht veränderten Blickwinkel neu her.

Einen Nachteil des in [SN00] vorgestellten Lernalgorithmus stellt die Verwendung nicht unerheblich vieler Membership Queries dar. Wir untersuchen im letzten Abschnitt der Arbeit die Frage, ob dieses Hilfsmittel zum effizienten Erlernen der betrachteten Funktionsklasse möglicherweise nicht zwingend notwendig ist. In diesem Zusammenhang zeigen wir die \mathcal{NP} -Härte und Nicht-Approximierbarkeit eines Teilproblems.

Inhaltsverzeichnis

1	Einleitung	1
2	Präliminarien	5
3	Lernen von Konzepten	8
3.1	Grundlagen	8
3.2	Effiziente PAC-Lernbarkeit	12
4	Fallbasiertes Lernen boolescher Funktionen	15
4.1	Satohs fallbasiertes Klassifikationsmodell	16
4.1.1	Spezifikation	16
4.1.2	Repräsentierbarkeit	19
4.1.3	Eigenschaften von Fallbasen	24
4.2	Ein erster Lernalgorithmus	29
4.3	Lernalgorithmus nach Satoh und Nakagawa	32
5	Komplexitätsanalyse	44
5.1	Optimierungsprobleme	46
5.2	Definition der betrachteten Probleme	47
5.3	Reduktion von Optimierungsproblemen	51
5.4	Komplexität der Fallbasenminimierung	54
5.5	Schlussfolgerungen	65
	Literaturverzeichnis	67

1 Einleitung

Die Entwicklung lernfähiger Software und damit auch der Bedarf an der Theorie maschinellen Lernens mag einem aktuellen Trend entsprechen, setzt jedoch eigentlich nur einen sehr alten konsequent fort: Der Einsatz menschlicher Arbeitskraft verlagert sich von der tatsächlichen Ausführung einer bestimmten Tätigkeit mehr und mehr zur Konzeption und Herstellung immer ausgefeilterer Werkzeuge, die diese komplett für uns übernehmen. Das gilt insbesondere für Computerprogramme, die als aus Software bestehende Werkzeuge gesehen werden müssen. Je komplexer, mächtiger und vielseitiger die Software jedoch wird, desto aufwendiger ist meist ihre Anpassung und Optimierung bezüglich spezieller Anforderungen. Es wäre daher wünschenswert, wenn Programme diese Aufgabe in gewissem Maße selbst übernehmen könnten.

Einem solchen Gedankengang entsprang die Motivation dieser Arbeit: Um den Wartungsaufwand einer viele Millionen Teile enthaltenden Maschinenbauteile-Datenbank möglichst gering zu halten, soll diese ihre eigene Zugriffsstruktur dahingehend optimieren, dass die von einem Ingenieur benötigte Zeit zum Suchen eines Bauteils möglichst gering ist. Zu diesem Zweck möchte man ihm stets bevorzugt diejenigen Teile anbieten, die, maschinenbautechnisch betrachtet, zu den im Vorfeld ausgewählten passen. Beispielsweise ist nach der Auswahl eines Generators die Suche nach einem Verbrennungsmotor deutlich wahrscheinlicher als die nach einem Getriebe. Zusätzlich könnte dem Benutzer die Auswahl dadurch erleichtert werden, dass von der Vielfalt verfügbarer Motoren nur solche angeboten werden, die eine gemeinsame Verwendung mit dem ausgewählten Generator erlauben, was anhand der technischen Daten der Bauteile entscheidbar sein sollte. Da eine manuelle Programmierung entsprechender Entscheidungsregeln umfassenden Sachverstand, also hochbezahlte Fachleute erfordert, und folglich angesichts der immensen Größe und zu erwartenden Dynamik der Datenbank eine erhebliche finanzielle Dauerbelastung darstellen würde, wird eine automatisierte Lösung angestrebt.

Somit erscheint ein Ansatz sinnvoll, bei dem die Datenbank allgemeine Beziehungen zwischen Teileklassen erlernt. Dabei soll im Gegensatz zu einer statistikbasierten Zusammenhangsbeschreibung, die lediglich erkennen lässt, *dass* zwei Teile häufig gemeinsam gewählt wurden und daher wohl zusammenpassen müssen, erlernt werden, *warum* dies so ist. Auf diese Weise könnten auch für neu eingeführte Teile, für die noch keine statistischen Daten zur Verfügung stehen, bereits entsprechende Vorhersagen getroffen werden. Dabei wird davon ausgegangen, dass jedes Teil derselben Bauteileklasse (zum Beispiel "Generator"), zumindest annähernd dieselben

Attribute (zum Beispiel “Eingangsleistung”) besitzt, sich aber über die Werte der Attribute von den anderen Teilen in der Klasse abgrenzt. Die Hoffnung ist nun, dass die Datenbank für jedes Paar von Bauteileklassen mit Hilfe positiver und negativer Trainingsbeispiele¹ lernt, für jedes Teilepaar aus den beiden betrachteten Klassen anhand von Beziehungen zwischen Attributswerten die gemeinsame Verwendbarkeit zu entscheiden.

Es liegt damit eine Fragestellung vor, die in das Gebiet des sogenannten *Concept Learning* passt. Beim *Lernen von Konzepten*, das wir in Kapitel 3 kurz einführen, geht es darum, eine Menge von Objekten (in unserem Beispiel eine Menge von Paaren aus einem Motor und einem Generator) in zwei verschiedene Klassen aufzuteilen (“gemeinsam verwendbar” oder “nicht gemeinsam verwendbar”). Die Aufgabe der Maschinenbauteiledatenbank wäre somit, Konzepte wie “Motor und Getriebe sind gemeinsam verwendbar” zu erlernen. Eine interessante Frage dabei ist beispielsweise, wie “schwierig” die Entscheidungsregel für dieses Konzept sein darf, damit sie noch mit vertretbarem Aufwand erlernt werden kann. Diese Frage bildet die Motivation für die theoretischen Untersuchungen dieser Arbeit, bei denen wir für eine spezielle Ausprägung des Concept Learning die Komplexität der effizient erlernbaren Zusammenhänge analysieren.

Dabei verwenden wir die ersten Gedanken darauf, wie man den Begriff “erlernbar” zu verstehen hat. So ist es eher unwahrscheinlich, dass eine meist relativ begrenzte Anzahl von Trainingsbeispielen ausreicht, um die Entscheidungsregel eines Konzeptes vollständig korrekt zu erlernen. Daher verwenden wir für unsere Untersuchungen ein entsprechend relaxiertes Lernbarkeits-Modell, das *Probably Approximately Correct (PAC)*-Modell, welches in seiner ursprünglichen Form von Valiant in [Val84] eingeführt wurde. Wir fordern lediglich, dass “wahrscheinlich” eine “annähernd richtige” Entscheidungsregel, auch *Hypothese* genannt, effizient erlernt werden kann.

Damit haben wir uns auf einen Lernbarkeitsbegriff festgelegt. Doch auf welche Weise sollen die Beziehungen erlernt werden? Eine interessante Ausprägung des Concept Learning stellt das *Fallbasierte Lernen* dar. Dabei wird das Erfahrungswissen durch eine Datenbank (*Fallbasis*) gespeicherter Referenzfälle repräsentiert, die während des Lernprozesses aufgebaut wird. Gewöhnlich wird ein neu zu klassifizierendes Objekt anhand der bekannten Klassenzuordnung der ähnlichsten Referenzobjekte in der Fallbasis eingestuft².

Prinzipiell wird im hier betrachteten fallbasierten Modell zum Lernen boolescher Funktionen, das Ken Satoh in [Sat98] verwendet, auf ähnliche Weise verfahren. Al-

¹Die Trainingsbeispiele könnten beispielsweise automatisch aus Zugriffsstatistiken gewonnen werden.

²Da mit der Bestimmung des ähnlichsten Objektes, die durchaus einen gewissen Rechenaufwand beinhalten kann, ein Teil der Lernarbeit dem Klassifikationsvorgang überlassen wird, wird das Fallbasierte Lernen oft als “faule” Lernmethode bezeichnet.

lerdings verwendet Satoh als Ähnlichkeitsmaß das bitweise exklusive “Oder” zweier Binärvektoren, und damit ist die Frage, welche Referenzfälle (also vorklassifizierte Binärvektoren aus der Fallbasis) einem Eingabevektor am ähnlichsten sind, nicht durch eine einfache Minimumbildung zu lösen. Es wird folglich ein komplexerer Klassifikator benötigt. In Kapitel 4 wenden wir uns dem durch Ähnlichkeitsfunktion und Klassifikator bestimmten Modell ausführlich zu und leiten die in [Sat98] und [SN00] erzielten Resultate unter einem leicht veränderten Blickwinkel neu her. Zunächst betrachten wir die repräsentative Mächtigkeit des Modells, indem wir die Menge aller booleschen Funktionen bestimmen, die von einer Fallbasis effizient dargestellt werden können. Damit erhält man offenbar bereits eine obere Schranke für die Komplexität der erlernbaren Funktionsklassen. Besonders die vorgestellten Minimalitätskriterien und Verfahren zur konsistenzzerhaltenden³ Verringerung der benötigten Anzahl von Referenzfällen spielen eine wichtige Rolle bei der Analyse der im Anschluss eingeführten Lernalgorithmen.

Der in Abschnitt 4.3 betrachtete Algorithmus wurde in seiner ursprünglichen Form von Satoh und Nakagawa in [SN00] vorgestellt, und erhält seine Bedeutung dadurch, dass er stets eine Hypothese (also eine erlernte Fallbasis) ausgibt, mit deren Hilfe nicht nur alle Trainingsbeispiele korrekt klassifiziert werden, sondern deren Größe außerdem abhängig von der Komplexität der zu lernenden Funktion beschränkt werden kann. Diese Tatsache ermöglicht den Nachweis, dass der betrachtete Algorithmus ein effizienter PAC-Algorithmus für jede überhaupt vom verwendeten Modell effizient repräsentierbare Klasse boolescher Funktionen⁴ ist.

Beim Lernvorgang bedient sich der angesprochene Lernalgorithmus jedoch sogenannter *Membership Queries*, die wir an dieser Stelle frei als “Experten-Anfragen” übersetzen wollen. Diese würde man in der Praxis gern vermeiden, denn diese Interaktion widerspricht nicht nur dem Prinzip einer sich autonom optimierenden Datenbank, sondern erfordert auch wieder die Verfügbarkeit entsprechend qualifizierter Fachkräfte. Im Verlauf der Untersuchungen mehren sich jedoch die Zweifel, dass die betrachtete Funktionsklasse auch ohne die Verwendung dieses mächtigen Hilfsmittels im betrachteten Modell effizient erlernbar ist. In Kapitel 5 fundieren wir diese Zweifel durch eigene Überlegungen, indem wir zeigen, dass weder eine exakte noch eine gute approximative Generalisierung von Hypothesen effizient realisiert werden kann. Dies gelingt durch die Herleitung eines Approximationsgüte-erhaltenden Reduktionsbegriffs, unter dessen Verwendung wir das \mathcal{NP} -vollständige Problem MINIMUM CLIQUE PARTITION [GJ79], für das eine starke untere Approximationsschranke bekannt ist⁵, auf das untersuchte Optimierungsproblem reduzieren.

³Konsistenz ist die Eigenschaft einer Hypothese, mindestens die bereits betrachteten Trainingsbeispiele korrekt zu klassifizieren (siehe auch Abschnitt 3.1).

⁴Diese umfasst beispielsweise die Menge aller k -Term-DNF beziehungsweise k -Term-CNF.

⁵unter der Voraussetzung $\mathcal{ZPP} \neq \mathcal{NP}$, siehe [Gil77] und [FK98]

Ich möchte mich sehr herzlich bei Gerhard Buntrock vom Institut für Theoretische Informatik für die gute Betreuung bedanken. Er hatte fast immer Zeit für mich und schaffte es, mich bei unseren vielen Treffen immer wieder neu zu motivieren. Außerdem gilt mein Dank Thomas Zeugmann, ebenfalls vom Institut für Theoretische Informatik, der wesentlich an der Themenfindung beteiligt war, und seine Zeit opferte, um mich in den Belangen des Maschinellen Lernens auf den richtigen Weg zu bringen. Ferner bedanke ich mich bei meiner Schwester Kathrin dafür, dass sie meine Arbeit komplett gelesen hat, und ich somit die Lesbarkeit anhand der akribischen und ausführlichen Korrekturen einer Mathematikerin in Form und Stil verbessern konnte.

Jan-Martin Kuhnigk,

30. Oktober 2001

2 Präliminarien

Bevor wir diverse Komplexitätstheoretische und mathematische Grundlagen anführen, wollen wir uns bezüglich einiger Schreibweisen festlegen.

Bezeichnungen

Folgende Namen und Notationen werden in der gesamten Arbeit verwendet:

- $\text{Pot}(M)$ bezeichnet die Potenzmenge einer Menge M .
- $\mathbf{P}(E)$ bezeichnet die Wahrscheinlichkeit eines Ereignisses E .
- Wir identifizieren eine boolesche Funktion f mit der Menge derjenigen Parameter, die von f auf 1 abgebildet werden:

$$x \in f \iff f(x) = 1.$$

- Funktions- beziehungsweise Problemklassen werden durch kalligraphische Buchstaben repräsentiert (z.B. $\mathcal{F}, \mathcal{NP}$).
- Für die Namen Komplexitätstheoretischer Probleme verwenden wir die Schriftart SMALLCAPS.

Einige Komplexitätstheoretische Grundlagen

Das Berechnungsmodell **Turingmaschine (TM)**, sowie die Klassen \mathcal{P} und \mathcal{NP} der Probleme, die sich mit Hilfe von deterministischen beziehungsweise nicht-deterministischen Turingmaschinen in polynomieller Zeit akzeptieren lassen, setzen wir als bekannt voraus. Gleiches gilt für die polynomiell zeitbeschränkte Reduktion $\leq^{\mathcal{P}}$. Für genauere Informationen zu diesen Themen verweisen wir auf [GJ79] sowie [BC94].

Probabilistische Turingmaschinen heben sich von herkömmlichen dadurch ab, dass sie sogenannte Münzwurf-Zustände besitzen können, in denen Entscheidungen abhängig vom Ergebnis eines fairen Münzwurfs getroffen werden. Dabei ist nach [Gil77]

- \mathcal{PP} die Klasse aller Sprachen, die von polynomiell zeitbeschränkten probabilistischen Turingmaschinen erkannt werden,

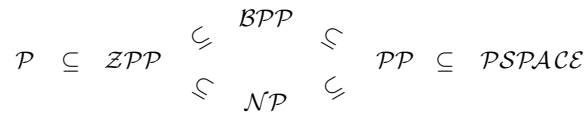


Abbildung 1 Hierarchische Einordnung von \mathcal{ZPP} nach [Gil77]. Dabei ist von keiner der Inklusionen bekannt, ob sie echt ist.

- \mathcal{BPP} die Klasse aller Sprachen, die von polynomiell zeitbeschränkten probabilistischen Turingmaschinen mit beschränkter Fehlerwahrscheinlichkeit erkannt werden,
- \mathcal{ZPP} die Klasse aller Sprachen, die von probabilistischen Turingmaschinen mit polynomiell beschränkter Durchschnitts-Laufzeit fehlerfrei erkannt werden.

Uns interessiert in diesem Zusammenhang besonders die Einordnung der Klasse \mathcal{ZPP} gemäß Abbildung 2.

Einige Mathematische Grundlagen

Definition 1 [Ger67] Ein **vollständiger Verband** ist eine quasi-geordnete Menge (M, \leq) , in der es zu jeder Teilmenge $M' \subseteq M$ sowohl ein größtes gemeinsames Unterelement $\inf(M')$ als auch genau ein kleinstes gemeinsames Oberelement $\sup(M')$ gibt.

Definition 2 Wir definieren eine Bijektion χ_n , die eine Teilmenge M von $\{1, 2, \dots, n\}$ auf einen Vektor c aus $\{0, 1\}^n$ mit folgender Eigenschaft abbildet:

$$i \in M \iff c[i] = 1.$$

Mit anderen Worten, jede Komponente von c ist genau dann Eins, wenn ihr Index in M enthalten ist. Wir sagen dann, c **korrespondiert zu M** und umgekehrt.

Bemerkung 3 Für diese Funktion χ_n gelten die folgenden Eigenschaften:

- (1) χ_n ist ein Isomorphismus zwischen den Monoiden $(\{1, 2, \dots, n\}, \cup)$ und $(\{0, 1\}^n, \vee)$.
- (2) **Monotonie:** Für alle $s_1, s_2 \subseteq \{1, 2, \dots, n\}$ gilt: $s_1 \subseteq s_2 \iff \chi_n(s_1) \preceq \chi_n(s_2)$.

Definition 4 Zu einer booleschen Formel f bezeichne $\mathbf{DNF}_{\min}(f)$ eine zu f äquivalente Formel in disjunktiver Normalform, welche aus der geringsten Anzahl von Monomen besteht. Analog dazu sei $\mathbf{CNF}_{\min}(f)$ als eine zu f äquivalente Formel in konjunktiver Normalform mit minimaler Anzahl von Klauseln.

3 Lernen von Konzepten

Machine Learning oder **Maschinelles Lernen** ist ein Teilgebiet des Themenbereichs **Künstliche Intelligenz (KI)** und widmet sich der Aufgabe, Computerprogramme zu entwerfen, die sich durch Erfahrungen selbstständig verbessern. Hatte man früher die Vorstellung, eine Art Elementarsystem zu entwickeln, welches viele verschiedene Aufgaben mit Hilfe einiger unspezifischer Basisalgorithmen erlernen kann, so ist man im Laufe der Zeit mehr dazu übergegangen, überwachte Lerner für spezielle Probleme zu entwickeln. Im Gegensatz zu vielen anderen Aspekten der KI sind in diesem Bereich bereits beachtliche Erfolge zu verzeichnen, die sich im praktischen Einsatz lernfähiger Software manifestieren. Erprobte Anwendungen existieren beispielsweise in den Bereichen Sprach- und Bilderkennung, Robotik und Data Mining [Mit97].

Beim **Concept Learning** oder **Lernen von Konzepten**, auch **Lernen von Begriffen** genannt, wird das Problem betrachtet, anhand von positiven und negativen Beispielen Klassifikationsregeln für die Zugehörigkeit von Objekten zu einer bestimmten Klasse zu erarbeiten. Eine solche Aufgabe wäre beispielsweise, aus einer Menge von Passfotos diejenigen herauszufiltern, auf denen die abgebildete Person einen Bart trägt, oder in einer Online-Shopping-Datenbank Artikel als für einen Benutzer interessant oder uninteressant zu klassifizieren. Es geht darum, Konzepte wie “Passfotos von Menschen, die einen Bart tragen” oder “ein für Herrn Müller interessanter Artikel sein” zu erlernen. Gewissermaßen kann man ein Konzept als binärwertige Funktion betrachten, die es möglichst gut zu approximieren gilt. Die gegebenen Positiv- und Negativbeispiele können als Stützstellen dieser Funktion betrachtet werden, die durch einen Lernalgorithmus in ihrer Gesamtheit angenähert werden soll.

3.1 Grundlagen

Wir definieren einen formalen Rahmen, innerhalb dessen wir uns bei der Untersuchung der hier betrachteten Methoden bewegen werden.

Gegenstand der Betrachtungen ist ein **Objekt-** oder auch **Instanzenraum X** . Ein **Konzept f_t** über X , auch **Zielfunktion** genannt, ist eine boolesche Funktion über X und Element eines **Konzeptraums** beziehungsweise einer **Konzeptklasse $\mathcal{F} \subseteq \text{Pot}(X)$** von Konzepten über X .

Ein **Lernalgorithmus** L für \mathcal{F} hat die Aufgabe, eine Zielfunktion $f_t \in \mathcal{F}$ mit Hilfe einer **Trainingsmenge** beziehungsweise eines **Samples** $T = \langle T_+, T_- \rangle$ zu identifizieren, wobei T_+ eine Teilmenge der von f_t positiv und T_- eine Teilmenge der negativ klassifizierten Objekte ist (wir schreiben $T_+ \subseteq f_t$ beziehungsweise $T_- \subseteq \bar{f}_t$).

Dieser Vorgang entspricht einer Suche nach der richtigen, das heißt f_t entsprechenden **Hypothese**. Dabei macht es mitunter Sinn, einen vom Konzeptraum abweichenden **Hypothesenraum** (auch **Hypothesenklasse**) \mathcal{H} als Suchraum zu verwenden. Dabei können Unterschiede in der Mächtigkeit und auch in der Repräsentation der enthaltenen Konzepte bestehen. Bekanntermaßen gibt es mehrere Darstellungen für ein und dieselbe Funktion, und möglicherweise lässt sich in einer Hypothesenklasse besser suchen als in einer anderen. Deshalb werden wir \mathcal{H} mitunter auch **Repräsentationsklasse** nennen. Wir gehen in der Folge stets davon aus, dass jedes mögliche Zielkonzept f_t zumindest von einer Hypothese h aus dem Hypothesenraum \mathcal{H} repräsentiert wird.

Beispiel 5 *In Stromaggregaten wird meist ein Generator verwendet, der die von einem Verbrennungsmotor mit konstanter Drehzahl erzeugte kinetische Energie in elektrischen Strom umwandelt.*

Sei M eine Menge von Motoren, die sich durch n_m Attribute auszeichnen und G eine Menge von Generatoren, die ihrerseits durch n_g Eigenschaften beschrieben werden. Es sei für jedes Paar $(m, g) \in M \times G$ aus einem Motor und einem Generator entscheidbar, ob die beiden Teile prinzipiell zusammen verwendet werden können oder nicht. Ein Lernsystem soll anhand von positiven und negativen Beispielpaaren diejenigen Attributsbeziehungen zwischen M und G erlernen, die notwendig sind, um später neue, nicht in den Beispielen enthaltene Paare (m, g) als zum Konzept "Paare aus Motor und Generator, die zusammen verwendet werden können" zugehörig oder nicht zu klassifizieren.

Die so erlernten Beziehungen könnten beispielsweise in einer Online-Datenbank für Maschinenteile verwendet werden, um einem Benutzer nach der Auswahl eines Generators eine Menge dazu passender Motoren anzubieten und damit den Bedienungsaufwand zu verringern. Der Vorteil gegenüber einem bloßen "auswendig Lernen" von Teilepaaren bestünde vor allem darin, dass nicht sämtliche Kombinationen spezieller Motoren und Generatoren vorher einzeln vorklassifiziert werden müssten, sondern nur eine (möglicherweise auch zufällige) Auswahl davon. Gerade bei dynamischen Datenbanken, bei denen im Laufe der Zeit immer wieder neue Objekte hinzugefügt werden müssen, ist dies sicherlich hilfreich.

Als Objektraum X böte sich zunächst das kartesische Produkt aus M und G an, also die Menge aller möglichen Paare von Motoren und Generatoren. Jedoch enthalten beide Klassen höchstwahrscheinlich Attribute verschiedenster, auch komplexerer

Datentypen. Dies erschwert die Definition einer halbwegs “einfachen” Konzeptklasse erheblich. Deutlich angenehmer wäre es beispielsweise, wenn man Eingabeobjekte mit ausschließlich binärwertigen Attributen vorliegen hätte. Zu einer geeigneten Relation R wäre eine naheliegende Transformation $\phi : G \times M \rightarrow \{0, 1\}^{n_m \cdot n_g}$ mit

$$\phi(g, m) := \begin{pmatrix} m_1 & R & g_1 \\ m_1 & R & g_2 \\ \vdots & & \\ m_1 & R & g_{n_g} \\ m_2 & R & g_1 \\ \vdots & & \\ m_2 & R & g_{n_g} \\ m_3 & R & g_1 \\ \vdots & & \\ m_{n_m} & R & g_{n_g} \end{pmatrix}$$

Dabei muss R mit den vielen verschiedenen Datentypen funktionieren. Wir wollen hier nicht weiter ins Detail gehen, und verwenden für R die Relation ‘=’, die über vergleichbaren Datentypen definiert ist. Sämtliche Attributspaare, wo ein Vergleich aufgrund unvergleichbarer Datentypen unmöglich ist, werden nicht übernommen. Jeder binäre neue Eingabevektor $\phi(g, m)$ hat damit die Länge $n \leq n_m \cdot n_g$.¹

Bei der Wahl einer geeigneten Konzeptklasse kann man bereits den nächsten Fehler begehen: Wählt man sie zu klein, so ist das unbekannte Zielkonzept vielleicht überhaupt nicht darin enthalten, wird sie zu groß gewählt, wird die Suche danach eventuell zu aufwendig.

Nehmen wir an, hinreichend für eine Entscheidung über die gemeinsame Verwendbarkeit eines Motors m und eines Generators g seien die Attribute m_3 Drehzahl, m_6 Leistung sowie g_2 benötigte Antriebsleistung und g_5 Betriebsdrehzahl. Im Vektor $x := \phi(g, m)$ stehe der Wert ($g_5 = m_3$) und ($g_2 = m_6$) an den Stellen i und j . Eine denkbare Zielfunktion wäre damit:

$$f_t(x) := x_i \wedge x_j$$

Offenbar handelt es sich hier um einen recht einfachen Zusammenhang, ein Monom in zwei Variablen. Falls nur Monome als Zielfunktionen in Frage kommen, wäre die Menge aller Monome über X als Konzeptklasse naheliegend. Das würde uns auch folgenden einfachen Lernalgorithmus in die Hand spielen:

¹Man kann sich sicherlich ausgefeiltere Transformationen eines Tupels (m, g) in einen Binärvektor vorstellen, aber dieses Problem soll hier nicht näher betrachtet werden.

1. Wir warten auf das erste Positivbeispiel und übernehmen jedes mit 0 belegte Literal negiert und jedes mit 1 belegte nicht-negiert in das Hypothesen-Monom auf.
2. Für jedes weitere Positivbeispiel konstruieren wir ein Monom analog, bilden die Schnittmenge der enthaltenen Literale mit denen aus dem Hypothesen-Monom, und konstruieren eine neue Hypothese durch UND-Verknüpfung der gemeinsamen Literale.

Ist das zuerst betrachtete positive Beispielobjekt im ersten Attribut mit 1 belegt, so könnte man vermuten, dass diese 1 für die positiv-Klassifikation ausschlaggebend war. Dementsprechend würden wir im ersten Schritt das Literal x_1 in das Hypothesen-Monom mit aufnehmen. Besitzt das zweite Positivbeispiel jedoch an gleicher Stelle eine 0, so ist diese Vermutung dahin, und wir können das Literal x_1 wie im zweiten Schritt angegeben zusammen mit allen anderen abweichenden aus dem Hypothesen-Monom wieder streichen. Bei einem solchen Schritt wird zwar die Menge der vom konstruierten Monom positiv klassifizierten Objekte größer, die Menge aller noch möglichen Hypothesen jedoch kleiner. Beim Lernvorgang wird also die Hypothesenklasse (und damit der abzusuchende Raum) sukzessiv eingeschränkt.

Bei der Suche nach der “richtigen” Hypothese lassen sich, sofern alle Trainingsbeispiele korrekt vorklassifiziert sind², sicherlich alle Hypothesen ausschließen, für die es ein Gegenbeispiel in den Trainingsdaten gibt. Diejenigen Hypothesen, für die dies nicht gilt, nennt man konsistent:

Definition 6 Eine Hypothese h über X heißt **konsistent mit einer Trainingsmenge** $T = \langle T_+, T_- \rangle$ genau dann, wenn alle Instanzen aus T von h korrekt klassifiziert werden. Es muss also gelten:

$$T_+ \subseteq h \quad \text{und} \quad T_- \subseteq \bar{h}.$$

Da die Trainingsdaten T nach Voraussetzung fehlerfrei sind, muss sich offenbar die Zielfunktion f_t in der Menge aller zu T konsistenten Hypothesen befinden. Besitzt diese Menge jedoch mehr als ein Element — was der Regelfall ist — so kann f_t nicht mit Sicherheit identifiziert werden, und das spätere Auftreten von Klassifikationsfehlern ist unvermeidbar.

Hilfreich wäre es daher wenn man zeigen könnte, dass ab einer gewissen Anzahl von Trainingsbeispielen die Menge der in Frage kommenden Hypothesen derart eingeschränkt ist, dass nur äußerst wenig Fehler zu erwarten sind. Weiterhin sollte sich dieses Limit in realisierbaren Grenzen halten. Diese Faktoren hängen natürlich

²Wir gehen in dieser Arbeit von fehlerfreien Trainingsdaten aus. Ansätze zum Umgang mit verrauschten Samples beim fallbasierten Lernen werden beispielsweise in [Aha91] vorgestellt.

nicht nur von der Lernmethode ab; es stellt sich vielmehr die Frage, ob jede beliebige Zielfunktion aus der betrachteten Konzeptklasse überhaupt auf effiziente Weise ausreichend angenähert werden kann.

Im folgenden Abschnitt führen wir ein oft verwendetes Lernmodell ein, welches sich dieser Problematik annimmt.

3.2 Effiziente PAC-Lernbarkeit

Wann bezeichnet man eine Klasse von Funktionen als effizient erlernbar?

Valiant führt in [Val84] das **Probably Approximately Correct (PAC)**-Lernmodell ein. Ein PAC-Algorithmus liefert *wahrscheinlich* eine *annähernd korrekte* Hypothese, denn es wird gefordert, dass der Lerner zu beliebig kleinen vorgegebenen Parametern ε und δ höchstens bei einem Anteil von δ Fällen eine Hypothese berechnet, deren Fehler größer als ε ist. Mit anderen Worten, es wird garantiert, dass ein Anteil von mindestens $(1 - \delta)$ Lernvorgängen erfolgreich verläuft, was wir damit gleichsetzen, dass die Wahrscheinlichkeit, ein beliebiges Objekt anhand der berechneten Hypothese falsch zu klassifizieren, höchstens ε ist.

Hierbei ist eine wichtige Voraussetzung, dass sich die zumeist unbekannte Wahrscheinlichkeitsverteilung \mathcal{D} über dem Instanzenraum zwischen dem Lernvorgang und der späteren Anwendung des Erlernten nicht ändert, denn der Lerner geht davon aus, dass das zufällig gezogene Sample auf die Wahrscheinlichkeitsverteilung über dem gesamten Objektraum schließen lässt. Wäre das Sample hingegen nicht repräsentativ, so bestünde die Möglichkeit, dass ein Lernalgorithmus auch aus sehr vielen Beispielen eine Hypothese bestimmt, welche lediglich einen bezüglich der wirklichen Verteilung unwichtigen Randbereich der Eingabedaten korrekt klassifiziert³.

Wir bestimmen den Fehler $error_{\mathcal{D}}(h)$ einer Hypothese h bei gegebener Verteilung \mathcal{D} über dem Objektraum als die Summe der Auftretens-Wahrscheinlichkeiten derjenigen Objekte, die von h falsch klassifiziert werden. Wir verwenden die folgende Definition für effiziente PAC-Lernbarkeit:

Definition 7 [Mit97] *Gegeben sei eine Konzeptklasse \mathcal{F} über dem Objektraum $X := \{0, 1\}^n$ sowie ein Lernalgorithmus L , der die Hypothesenklasse \mathcal{H} verwendet.*

*\mathcal{F} heißt **effizient PAC-lernbar unter Verwendung von \mathcal{H}** , falls L für jedes Konzept f_t aus \mathcal{F} , jede Verteilung \mathcal{D} über X , jedes ε mit $0 < \varepsilon < 1/2$ und jedes δ mit $0 < \delta < 1/2$ mit einer Wahrscheinlichkeit von mindestens $(1 - \delta)$ und in einer*

³Es wäre gewissermaßen so, als würde man einen Experten für Pferdekutschen in der heutigen Automobilindustrie beschäftigen. Sein Wissen wäre heute nicht falsch, jedoch vermutlich recht irrelevant.

Laufzeit, die polynomiell in $1/\varepsilon$, $1/\delta$ und n beschränkt ist, eine Hypothese h aus H ausgibt, so dass $\text{error}_{\mathcal{D}}(h) \leq \varepsilon$ gilt.

Ein solcher Lernalgorithmus heißt **effizienter PAC-Lernalgorithmus für \mathcal{F}** .

Der folgende Satz stellt ein wichtiges Hilfsmittel beim Nachweisen von PAC-Lernbarkeit dar. Dabei bezeichnet $\mathcal{H}_{n,m}$ die Menge aller Hypothesen, die von L nach der Verarbeitung von maximal m Trainingsbeispielen generiert werden können.

Satz 8 [KV94] (*Occams Razor*⁴, Kardinalitäts-Version) Sei L ein Lernalgorithmus für eine Menge n -stelliger boolescher Funktionen \mathcal{F}_n , der zu einem gegebenen Sample T der Größe m in einer Zeit, die polynomiell in $1/\varepsilon$, $1/\delta$, n und m beschränkt ist, eine bezüglich T konsistente Hypothese h aus $\mathcal{H}_{n,m}$ berechnet. Dann gilt, dass für jede Verteilung \mathcal{D} über den Objekten eine Menge von

$$m > 1/\varepsilon \cdot (\log |\mathcal{H}_{n,m}| + \log(1/\delta))$$

zufällig gezogenen Trainingsobjekten ausreicht, damit die von L ausgegebene Hypothese h mit Wahrscheinlichkeit von mindestens $(1 - \delta)$ einen Fehler $\text{error}_{\mathcal{D}}(h)$ von maximal ε besitzt.

Beweis: Wir sagen, eine Hypothese $h \in \mathcal{H}_{n,m}$ ist **schlecht**, wenn sie einen Fehler $\text{error}_{\mathcal{D}}(h)$ besitzt, der größer als ε ist. Laut Voraussetzung muss jede von L errechnete Hypothese konsistent mit den m Trainingsinstanzen sein. Die Wahrscheinlichkeit, dass eine solche konsistente Hypothese dennoch schlecht ist, beträgt folglich maximal $(1 - \varepsilon)^m$.

Wir wollen die Menge aller schlechten Hypothesen, die von L nach m Schritten berechnet werden könnten, mit $\tilde{\mathcal{H}}_{n,m}$ bezeichnen. Mit $P(A \cup B) \leq P(A) + P(B)$ (für zwei beliebige Ereignisse A und B) folgt, dass die Wahrscheinlichkeit, dass irgendeine schlechte Hypothese aus $\tilde{\mathcal{H}}_{n,m}$ konsistent mit T ist, kleiner als $|\tilde{\mathcal{H}}_{n,m}| \cdot (1 - \varepsilon)^m$ ist. Wir fordern, dass diese Wahrscheinlichkeit kleiner als das vorgegebene δ ist. Es genügt demnach, m so auszuwählen, dass folgende Ungleichung gilt:

$$|\tilde{\mathcal{H}}_{n,m}| \cdot (1 - \varepsilon)^m < \delta. \quad (1)$$

Da $\tilde{\mathcal{H}}_{n,m} \subseteq \mathcal{H}_{n,m}$, gilt auch $|\tilde{\mathcal{H}}_{n,m}| \leq |\mathcal{H}_{n,m}|$, und mit $(1 - \varepsilon) \leq e^{-\varepsilon}$ ist somit Aussage (1) sicherlich auch erfüllt, wenn wir m gemäß

$$|\mathcal{H}_{n,m}| \cdot e^{-\varepsilon m} < \delta.$$

⁴Nach William von Occam, Theologe und Philosoph des 14. Jahrhunderts. Ihm wird das diesem Satz zugrunde liegende Prinzip "Keep it simple, stupid" zugeschrieben: Einfache (kurze) Hypothesen seien komplizierteren (längeren) vorzuziehen.

wählen. Wird auf beiden Seiten logarithmiert und nach m aufgelöst, so erhalten wir die gewünschte Ungleichung.

◇

Offenbar ist L genau dann ein effizienter PAC-Lernalgorithmus, falls m durch ein Polynom in n , $1/\varepsilon$ und $1/\delta$ nach oben beschränkt werden kann. Dazu muss der Term

$$1/\varepsilon \cdot (\log |\mathcal{H}_{n,m}| + \log(1/\delta))$$

ein Polynom in diesen Parametern darstellen. Wir können also formulieren:

Satz 9 *Falls ein Lernalgorithmus für \mathcal{F}_n , der für eine m -elementige Trainingsmenge die Hypothesenklasse $\mathcal{H}_{n,m}$ verwendet, die Voraussetzungen aus Satz 8 erfüllt, und $\log |\mathcal{H}_{n,m}|$ polynomiell ist in n , $1/\varepsilon$ und $1/\delta$, so ist er ein effizienter PAC-Lernalgorithmus für \mathcal{F}_n .*

4 Fallbasiertes Lernen boolescher Funktionen

Konzeptuelles Lernen gibt es in vielen Ausprägungen. Wir wollen uns hier auf die des **Fallbasierten Lernens** einschränken, welche auf der schon seit längerer Zeit in der Statistik angewendeten Klassifikationstechnik des **Fallbasierten Schließens** (**Case-Based Reasoning**) beruht.

Der Grundgedanke beim Fallbasierten Schließen ist die Klassifikation von Objekten, die in diesem Zusammenhang **Fälle** genannt werden, durch Bewertung ihrer “Ähnlichkeit” zu Elementen einer aus Trainingsdaten gewonnenen Menge von Referenzfällen, die wir **Fallbasis** nennen werden.

Ein typisches Beispiel für eine Anwendung dieses Ansatzes ist der recht bekannte k -Nächste-Nachbarn-Algorithmus. Dabei ist das Ziel, Merkmalsvektoren mit Hilfe einer vorgegebenen Basis bereits korrekt klassifizierter Beispiele in zwei oder mehr Klassen einzuteilen. Zu einem zu klassifizierenden Eingabevektor sucht der Algorithmus nun die k ähnlichsten Vektoren in der Fallbasis, wofür zunächst ein **Ähnlichkeits-** oder **Abstandsmaß** benötigt wird. Offenbar muss weiterhin bestimmbar sein, ob ein Vektor einem anderen ähnlicher ist als ein dritter. Somit muss auf dem Wertebereich des Abstandsmaßes zumindest eine Halbordnung definiert sein. Entstammen die Eingabevektoren beispielsweise dem \mathbb{R}^n , so käme als Abstandsfunktion die euklidische Metrik in Frage. Da diese in die reellen Zahlen abbildet, könnten Abstände einfach mit “ \leq ” verglichen werden.

Hat man die k ähnlichsten Vektoren bestimmt, kann man diese zur Bestimmung der Klasse des Eingabevektors verwenden. Ein typisches Verfahren ist beispielsweise, die Klasse auszuwählen, zu der die meisten dieser k Vektoren gehören. Häufig wird zusätzlich der Abstand der einzelnen Vektoren zum Eingabevektor gewichtend in die Entscheidung mit eingebracht.

Die maßgeblichen Fragestellungen bei der Definition eines fallbasierten Klassifikationssystems sind in diesem Beispiel bereits aufgetreten:

- Wie definiert man die Ähnlichkeit von Fällen?
- Wann ist ein Abstand kleiner als ein anderer?
- Wie sieht die Klassifikationsregel aus?

Wir werden jetzt ein spezifisches Klassifikationsmodell definieren, indem wir uns bezüglich dieser Parameter festlegen.

4.1 Satohs fallbasiertes Klassifikationsmodell

Das in der Folge definierte fallbasierte Klassifikationsmodell wurde von Ken Satoh¹ in [Sat98] eingeführt und untersucht.

Die in diesem Abschnitt erzielten Resultate entstammen im Wesentlichen den Arbeiten [Sat98] und [SN00], jedoch werden wir sämtliche Beweise neu formulieren, was auch gewisse Änderungen im Aussagengebäude hervorruft. Insbesondere soll hier die Aufmerksamkeit auf den kardinalitätsbezogenen Ansatz aus Occams Razor (Satz 8) zum Beweis der PAC-Lernbarkeit gelenkt werden. Wir werden den Lernalgorithmus aus [SN00] daher entsprechend anpassen.

4.1.1 Spezifikation

Wir werden als Objektraum X die Menge $\{0, 1\}^n$ verwenden, und uns somit auf das Erlernen n -stelliger boolescher Funktionen konzentrieren.

Definition 10 [Sat98] *Einen n -stelligen Binärvektor nennen wir **Fall**. Für zwei Fälle $c_1, c_2 \in \{0, 1\}^n$ definieren wir die **Abstandsfunktion** $d: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ als das bitweise XOR zwischen den Instanzen. Der **Abstand** beziehungsweise die **Distanz** $d(c_1, c_2)$ von c_1 und c_2 bestimmt sich damit wie folgt:*

$$d(c_1, c_2)[i] := c_1[i] \oplus c_2[i] \quad ; \quad 1 \leq i \leq n.$$

Weiterhin definieren wir die Relation \preceq als Teilmenge von $\{0, 1\}^n \times \{0, 1\}^n$ durch das bitweise \leq :

$$d(c_1, c_2) \preceq d(c_3, c_4) \iff \forall i \in \{1, 2, \dots, n\} : d(c_1, c_2)[i] \leq d(c_3, c_4)[i].$$

Damit erbt \preceq auch die Eigenschaften Reflexivität, Antisymmetrie und Transitivität von \leq und verleiht der betrachteten Menge $\{0, 1\}^n$ als Halbordnung Struktur. Darüber hinaus stellt $(\{0, 1\}^n, \preceq)$ nicht nur eine quasi-geordnete Menge dar, sondern es existiert für jeder Untermenge von $\{0, 1\}^n$ stets ein Supremum sowie ein Infimum (vergleiche Abbildung 4.1.1):

¹Ken Satoh ist gegenwärtig Professor an der Foundations of Information Research Division des National Institute of Informatics in Tokyo.

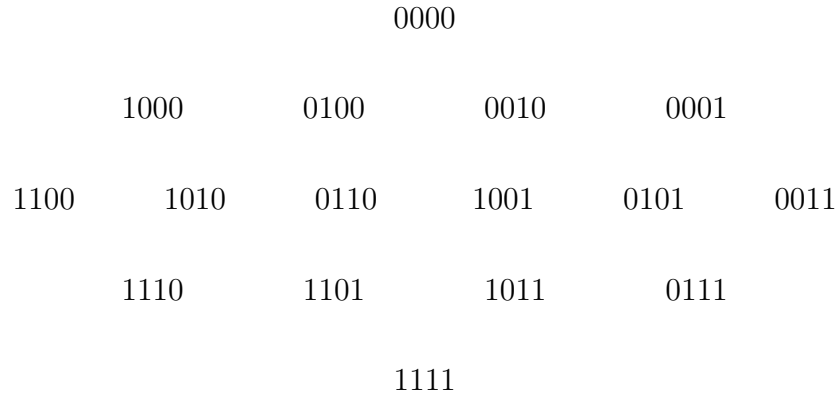


Abbildung 2 Der Verband $(\{0, 1\}^4, \preceq)$ als Graph dargestellt. Die Pfeile zeigen dabei stets vom kleineren auf das größere Element. Das Supremum der Binärvektoren 0110 und 0001 ist der “kleinste” gemeinsame Nachfahre, also 0111, und das Infimum mit 0000 der “größte” gemeinsame Vorfahre im abgebildeten Graphen.

Bemerkung 11 Die Struktur $(\{0, 1\}^n, \preceq)$ bildet einen vollständigen Verband. Sei X eine beliebige Teilmenge von $\{0, 1\}^n$, und bezeichne \wedge das bitweise AND sowie \vee das bitweise OR zweier Binärvektoren, dann gilt:

$$\inf(X) := \bigwedge_{x \in X} x \quad ; \quad \sup(X) := \bigvee_{x \in X} x.$$

Nun macht es im Allgemeinen wenig Sinn, zu fragen, ob ein Fall “kleiner” als ein anderer ist. Vielmehr verwenden wir die Halbordnungsrelation \preceq über Abständen von Fällen, die ebenfalls in $\{0, 1\}^n$ liegen.

Definition 12 Für zwei gegebene Fälle $c_1, c_2 \in \{0, 1\}^n$ definieren wir die Menge $U(c_1, c_2) \subseteq \{0, 1\}^n$ wie folgt:

$$U(c_1, c_2) := \{c \in \{0, 1\}^n \mid d(c_1, c) \preceq d(c_1, c_2)\}.$$

$(U(c_1, c_2), \preceq)$ ist damit wiederum ein Verband. Falls $c \in U(c_1, c_2)$ gilt, sagen wir, c befindet sich im **Unterverband zwischen c_1 und c_2** .

Um zu überprüfen, ob sich ein Fall c zwischen c_1 und c_2 befindet, bilden wir $U(c_1, c_2)$ bijektiv auf den Verband zwischen $d(c_1, c_1)$ (also 0^n) und $d(c_1, c_2)$ ab. Somit werden nicht Fälle sondern Fallabstände verglichen, und eine sinnvolle Verwendung der Relation \preceq ist möglich.

Definition 15 [Sat98] Sei mit $CB = \langle CB_+, CB_- \rangle$ eine Fallbasis gegeben, dann ist die boolesche Funktion f_{CB} definiert durch

$$f_{CB} := \{c \in \{0, 1\}^n \mid \exists c_{ok} \in CB_+ : \forall c_{ng} \in CB_- : c_{ng} \notin U(c_{ok}, c)\}.$$

Wir sagen, **CB wird von f_{CB} repräsentiert**.

Mit anderen Worten: ein Fall c wird genau dann positiv klassifiziert, wenn kein Negativbeispiel aus CB_- existiert, welches c bezüglich c_{ok} abdeckt (Abbildung 4.1.1). Dabei ist die vorgestellte Funktionsdefinition als zusammengehörig mit der vorhergehenden Festlegung der Abstandsfunktion sowie der Halbordnung \preceq aus Definition 10 anzusehen, da diese entscheidend sind für die Beantwortung der Frage $c_{ng} \notin U(c_{ok}, c)$.

Die in der folgenden Bemerkung dargelegten, natürlich anmutenden Eigenschaften belegen den Sinn der obigen Klassifikationsregel.

Bemerkung 16 Gegeben sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$. Seien $c_{ok} \in CB_+$, $c_{ng} \in CB_-$, $c_+ \notin CB_-$ und $c_- \notin CB_+$ Fälle, dann gelten folgende Eigenschaften:

- (a) $c_{ok} \in f_{\langle \{c_{ok}\}, CB_- \rangle} \wedge c_{ng} \notin f_{\langle CB_+, \{c_{ng}\} \rangle}$.
Jeder Fall aus CB_+ oder CB_- klassifiziert sich selbst gemäß der Fallbasis, in der er enthalten ist.
- (b) $f_{CB} \subseteq f_{\langle CB_+ \cup \{c_+\}, CB_- \rangle} \wedge f_{CB} \subseteq f_{\langle CB_+, CB_- \setminus \{c_{ng}\} \rangle}$.
Fügt man einen Fall zu CB_+ hinzu oder entfernt einen aus CB_- , so werden dadurch keine zuvor positiv klassifizierten Fälle nun negativ klassifiziert.
- (c) $f_{\langle CB_+, CB_- \cup \{c_-\} \rangle} \subseteq f_{CB} \wedge f_{\langle CB_+ \setminus \{c_{ng}\}, CB_- \rangle} \subseteq f_{CB}$.
Fügt man einen Fall zu CB_- hinzu oder entfernt einen aus CB_+ , so werden dadurch keine zuvor negativ klassifizierten Fälle nun positiv klassifiziert.

4.1.2 Repräsentierbarkeit

Unser Fernziel ist es, mit Hilfe satohscher Fallbasen boolesche Funktionen zu erlernen. Dafür müssen wir zunächst herleiten, wie eine boolesche Funktion überhaupt von einer solchen Fallbasis repräsentiert werden kann. Weiterhin werden wir untersuchen, für welche Funktionsklassen eine Repräsentation jeder darin enthaltenen Funktion möglich ist, ohne dass ineffizient viele Fälle dazu benötigt werden.

Mit Bemerkung 16 würde eine n -stellige boolesche Funktion f von der trivialen Fallbasis $CB := \langle f, \bar{f} \rangle$ korrekt repräsentiert. Eine Fallbasis der Größe 2^n ist jedoch nicht

erwünscht, sie wäre damit nichts anderes als eine Wertetabelle zu f . Untersuchen wir daher zunächst, mit wievielen positiven Referenzfällen sich eine boolesche Funktion repräsentieren lässt. Das folgende Lemma beantwortet diese Frage konstruktiv:

Lemma 17 [Sat98] *Sei F eine DNF zu einer booleschen Funktion f . Sei CB_+ eine Menge von Fällen aus f , so dass jedes Monom aus F von genau einem Fall c_{ok} aus CB_+ erfüllt werde. Dann gilt*

$$f = f_{\langle CB_+, \bar{f} \rangle}.$$

Beweis: Sei $CB := \langle CB_+, \bar{f} \rangle$. Mit Bemerkung 16(a) klassifiziert sich jeder Fall aus \bar{f} selbst negativ. Damit gilt $\bar{f} \subseteq \bar{f}_{\langle CB_+, \bar{f} \rangle}$.

Es bleibt zu zeigen, dass aus $c_+ \in f$ folgt: $c_+ \in f_{CB}$. Zu jedem c_+ aus f existiert nach Voraussetzung ein Fall c_{ok} in CB_+ , welcher dasselbe Monom M_j wie c_+ in F erfüllt. Damit erfüllt auch der Unterverband $U(c_{ok}, c_+)$ komplett dieses Monom M_j , denn alle in M_j vorkommenden Attribute müssen offenbar sowohl in c_{ok} als auch in c_+ gleich belegt sein. Folglich erfüllt auch jeder Fall aus $U(c_{ok}, c_+)$ das Monom M_j und es kann kein Negativfall aus \bar{f} darin liegen, der c_+ bezüglich c_{ok} abdeckt. Damit muss c_+ in f_{CB} liegen.

◇

Damit haben wir eine einfache Konstruktionsmethode für eine positive Fallbasis, jedoch ist die zugehörige negative Fallbasis, die aus ganz \bar{f} besteht, noch nicht akzeptabel. Es wäre wünschenswert, auch diese Menge auf einige Repräsentanten zu reduzieren. Um dies zu bewerkstelligen, werden wir zunächst den Begriff der nächsten Nachbarn einer Instanz in einer Fallmenge benötigen:

Definition 18 [Sat98] *Gegeben seien eine Fallmenge $S \subseteq \{0, 1\}^n$ und ein Fall $c \in \{0, 1\}^n \setminus S$. Die **nächsten Nachbarn von c in S** , geschrieben $NN(c, S)$, sind wie folgt definiert:*

$$NN(c, S) := \{c' \in S \mid c' \in f_{\{\{c\}, S \setminus \{c'\}\}}\}.$$

In Abbildung 4.1.1 sind die nächsten Nachbarn von c_{ok} in der Menge der eingezeichneten Negativfälle diejenigen, die keinen negativen Vorgänger besitzen, also die Negativfälle mit Abstand 0101 beziehungsweise 1000 zu c_{ok} . Es gilt nun die folgende Eigenschaft:

Lemma 19 *Betrachten wir eine Fallmenge $S \subseteq \{0, 1\}^n$ und einen Fall $c \in \{0, 1\}^n \setminus S$. Für jedes $c_1 \in S$ existiert ein c' in $\text{NN}(c, S)$, das c_1 bezüglich c abdeckt, welches also in $U(c, c_1)$ enthalten ist.*

Beweis: Ist c_1 selbst in $\text{NN}(c, S)$, so ist die Bedingung natürlich für $c' := c_1$ erfüllt. Sei also $c_1 \notin \text{NN}(c, S)$. Das gilt genau dann, wenn

$$c_1 \notin f_{\langle \{c\}, S \setminus \{c_1\} \rangle},$$

was äquivalent dazu ist, dass es ein c_2 in S gibt, für das

$$d(c, c_2) \preceq d(c, c_1)$$

gilt. Entweder ist c_2 nun selbst in $\text{NN}(c, S)$, dann wäre das Gesuchte c' gleich c_2 und wir wären fertig, oder c_2 wird wiederum von einem anderen Negativfall c_3 abgedeckt. Genau dieselbe Betrachtung ist nun für diesen Fall durchzuführen. Aus der Antisymmetrie von \preceq und der Endlichkeit von S als Teilmenge von $\{0, 1\}^n$ folgt, dass diese Kette endlich ist. Damit wird man irgendwann auf einen Fall c_k treffen, welcher von keinem anderen abgedeckt wird, und der folglich in $\text{NN}(c, S)$ liegt. Mit

$$d(c, c_k) \preceq \dots \preceq d(c, c_3) \preceq d(c, c_2) \preceq d(c, c_1)$$

und der Transitivität von \preceq folgt $d(c, c_k) \preceq d(c, c_1)$. Da $c_k \in \text{NN}(c, S)$, haben wir damit unser c' gefunden. ◇

Die Nächste-Nachbarn-Operation wird im folgenden Verfahren Verwendung finden, das aus einer negativen Fallbasis alle nicht benötigten Fälle aussortiert ohne die repräsentierte Funktion zu verändern.

Satz 20 [Sat98] *Gegeben sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$. Dann gilt*

$$f_{CB} = f_{\langle CB_+, \bigcup_{c_{ok} \in CB_+} \text{NN}(c_{ok}, CB_-) \rangle}.$$

Beweis: Sei $CB' := \langle CB_+, CB'_- \rangle$ mit

$$CB'_- := \bigcup_{c_{ok} \in CB_+} \text{NN}(c_{ok}, CB_-).$$

Betrachten wir einen beliebigen Fall c_{ng} aus CB_- , der in CB'_- nicht mehr vertreten ist. In Lemma 19 hatten wir gezeigt, dass für jedes c_{ok} aus CB_+ ein Fall in CB'_- existieren muss, welcher c_{ng} bezüglich c_{ok} abdeckt. Somit ist dieser Fall in CB'_- überflüssig; er grenzt nicht an einen positiven Bereich. Da CB'_- eine Untermenge von CB_- ist, galt dies bereits in CB_- . Das Entfernen eines solchen Falles beeinflusst folglich die repräsentierte Funktion nicht. ◇

Mit diesem Satz könnten wir nun auch die negative Fallbasis aus Lemma 17 konsistenzertaltend reduzieren. Doch kann man das Ausmaß der Reduktion quantifizieren? Das folgende Lemma beschreibt eine wichtige Eigenschaft dieses Modells und liefert die Basis für eine positive Antwort auf diese Frage.

Lemma 21 *Sei mit f eine boolesche Funktion gegeben, und bezeichne \overline{F} eine DNF-Repräsentation von \overline{f} . Sei weiterhin c_+ ein beliebiger Fall aus f , dann wird kein Monom aus \overline{F} von mehr als einem Fall aus $\text{NN}(c_+, \overline{f})$ erfüllt. Damit gilt $|\text{NN}(c_+, \overline{f})| \leq |\overline{F}|$ und insbesondere*

$$|\text{NN}(c_+, \overline{f})| \leq |\text{CNF}_{\min}(f)|.$$

Beweis: Wir zeigen, dass für ein beliebiges Monom M aus \overline{F} genau ein erfüllender Fall c existiert, welcher das komplette Monom bezüglich c_+ abdeckt. Diesen Fall c konstruieren wir wie folgt: Wir setzen zunächst $c := c_+$, und überschreiben danach alle für M relevanten Attribute so, dass c das Monom M erfüllt. Folglich hat der Abstand d von c zu c_+ höchstens in den für M relevanten Attributen den Wert 1. Da jeder andere M erfüllende Fall c' an diesen Stellen genauso belegt ist, muss

$$d(c_+, c) \leq d(c_+, c')$$

sein, das heißt, c' wird von c bezüglich c_+ abgedeckt. Folglich kann kein solches c' in $\text{NN}(c_+, \overline{f})$ sein.

Damit kann $\text{NN}(c_+, \overline{f})$ als einzigen M erfüllenden Fall lediglich den von uns konstruierten Fall c enthalten. Folglich ist die Kardinalität von $\text{NN}(c_+, \overline{f})$ durch die Anzahl der Monome in \overline{F} beschränkt. Da $|\text{NN}(c_+, \overline{f})|$ jedoch offenbar nicht von einem speziellen \overline{F} abhängt, können wir für \overline{F} eine solche DNF-Repräsentation von \overline{f} annehmen, die sich aus der Negation einer minimalen CNF zu f ergibt.

Es gilt also:

$$|\text{NN}(c_+, \overline{f})| \leq |\text{CNF}_{\min}(f)|.$$

◇

Wir werden die **Größe einer Fallbasis** $CB = \langle CB_+, CB_- \rangle$ in der Folge mit $|CB|$ bezeichnen, und meinen damit die Summe von $|CB_+|$ und $|CB_-|$.

Definition 22 [Sat98] *Sei \mathcal{F}_n eine Klasse n -stelliger boolescher Funktionen und es sei $\mathcal{F} := \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$. Falls es ein Polynom p gibt, so dass für jedes $n \in \mathbb{N}$ und $f \in \mathcal{F}_n$ eine Fallbasis CB mit $f_{CB} = f$ und $|CB| \leq p(n)$ existiert, so nennen wir \mathcal{F} **polynomiell repräsentierbar**.*

Der folgende Satz liefert eine Obergrenze für die Repräsentierbarkeit boolescher Funktionen.

Satz 23 [Sat98] *Sei f eine boolesche Funktion. Dann existiert eine f repräsentierende Fallbasis $CB = \langle CB_+, CB_- \rangle$, die nicht mehr als*

$$|CB| \leq |DNF_{\min}(f)| \cdot (1 + |CNF_{\min}(f)|)$$

Fälle enthält. Insbesondere gilt

$$|CB_+| \leq |DNF_{\min}(f)| \quad \text{und} \quad |CB_-| \leq |DNF_{\min}(f)| \cdot |CNF_{\min}(f)|.$$

Beweis: Mit Hilfe von Lemma 17 erzeugen wir aus einer minimalen DNF zu f eine f repräsentierende Fallbasis $\langle CB_+, \bar{f} \rangle$. Diese besteht nach Konstruktion aus höchstens $|DNF_{\min}(f)|$ Fällen, es ist also

$$|CB_+| \leq |DNF_{\min}(f)|.$$

Nun setzen wir $CB_- := \bigcup_{c_{ok} \in CB_+} \text{NN}(c_{ok}, \bar{f})$. Laut Satz 20 gilt $f_{\langle CB_+, CB_- \rangle} = f_{\langle CB_+, \bar{f} \rangle}$, und damit ist auch

$$f_{\langle CB_+, CB_- \rangle} = f.$$

Nach Lemma 21 besitzt $\text{NN}(c_{ok}, CB_-)$ für jedes c_{ok} aus CB_+ maximal $|CNF_{\min}(f)|$ Fälle. Mit $|CB_+| \leq |DNF_{\min}(f)|$ gilt für CB_- insgesamt:

$$|CB_-| \leq |DNF_{\min}(f)| \cdot |CNF_{\min}(f)|.$$

◇

Aus dem vorhergehenden Satz lässt sich sofort folgern:

Bemerkung 24 [Sat98] *Sei p ein Polynom und n eine natürliche Zahl, und sei \mathcal{F}_n^p die Klasse n -stelliger boolescher Funktionen, für die gilt:*

$$f \in \mathcal{F}_n^p \iff |DNF_{\min}(f)| \leq p(n) \wedge |CNF_{\min}(f)| \leq p(n).$$

Dann ist $\mathcal{F}^p := \bigcup_{n \in \mathbb{N}} \mathcal{F}_n^p$ polynomiell repräsentierbar.

Offenbar erhalten wir damit bereits eine obere Schranke für die effizient erlernbaren Klassen: Was sich nicht effizient darstellen lässt können wir auch nicht effizient erlernen.

Es gibt einige interessante Klassen, die polynomiell repräsentierbar sind. Dazu zählt beispielsweise die Menge aller k -Term- DNF , da es zu jeder n -stelligem Funktion, die eine k -Term- DNF besitzt, eine äquivalente CNF mit höchstens n^k Klauseln gibt. Dies gilt natürlich analog für die Menge aller k -Term- CNF .

4.1.3 Eigenschaften von Fallbasen

Haben wir uns bisher auf die Konstruktion einer Fallbasis zu einer gegebenen booleschen Funktion konzentriert, so werden wir im Folgenden die Eigenschaften einer gegebenen Fallbasis beziehungsweise der von ihr repräsentierten Funktion untersuchen.

Wir werden den Begriff einer kritischen Fallbasis einführen. Bei einer solchen Fallbasis bewirkt jede Entnahme von Fällen eine Änderung der repräsentierten Funktion:

Definition 25 [SN00] Eine Fallbasis $CB = \langle CB_+, CB_- \rangle$ heißt **kritisch**, falls für jedes $CB' = \langle CB'_+, CB'_- \rangle$ mit $CB'_+ \subseteq CB_+$, $CB'_- \subseteq CB_-$ und $CB' \neq CB$ gilt:

$$f_{CB'} \neq f_{CB}.$$

Definition 26 [SN00] Sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$ gegeben.

- Die negative Fallbasis CB_- heißt **minimal bezüglich CB_+ und f** , falls kein $CB'_- \subsetneq CB_-$ existiert mit

$$f_{\langle CB_+, CB'_- \rangle} = f_{CB}.$$

- Die positive Fallbasis CB_+ heißt **minimal bezüglich f** , falls kein $CB'_+ \subsetneq CB_+$ existiert, so dass für irgendein CB'_- gilt

$$f_{\langle CB'_+, CB'_- \rangle} = f_{CB}.$$

Die in Abbildung 4.1.1 eingezeichnete negative Fallbasis ist offenbar nicht minimal, man könnte den Fall 1101 entfernen, ohne dass sich die repräsentierte Funktion ändert.

Es fällt auf, dass die beiden Definitionen für Minimalität nicht völlig symmetrisch sind. Das liegt daran, dass das Entfernen eines Positivfalles aus CB_+ durchaus Auswirkungen auf f_{CB} haben kann, obwohl dieser Fall auch ohne eigenes Auftreten in CB_+ bereits positiv klassifiziert würde.

In Abbildung 4.1.1 könnte man beispielsweise zusätzlich den Fall 0110 in CB_+ einfügen, wodurch die repräsentierte Funktion tatsächlich verändert würde. So würden unter anderem die Fälle 0111 und 1110 auf einmal positiv klassifiziert (um die ursprüngliche Funktion zu erhalten, wäre eine Anpassung von CB_- notwendig). Aus diesem Grund wird für die Minimalität der positiven Fallbasis gefordert, dass keine

negative Fallbasis existiert, bei der nach dem Entfernen von Positivfällen aus CB_+ die repräsentierte Funktion erhalten bleibt.

Wir haben soeben festgestellt, dass das Hinzufügen von bereits korrekt klassifizierten Fällen nicht unbedingt ein harmloser Vorgang sein muss. Die hier nicht gegebene Eigenschaft, dass die repräsentierte Funktion diesbezüglich invariant ist, nennt man Stabilität.

Definition 27 [GL96] *Eine Fallbasis $CB = \langle CB_+, CB_- \rangle$ heißt **stabil**, falls für beliebige Mengen $P \subseteq f_{CB}$ und $N \subseteq \bar{f}_{CB}$ gilt:*

$$f_{\langle CB_+ \cup P, CB_- \cup N \rangle} = f_{CB}.$$

Wie schon obiges Beispiel vermuten lässt, sind Fallbasen im satohschen Modell im Allgemeinen nicht stabil.

Jetzt wollen wir uns wieder dem Begriff der kritischen Fallbasis widmen. Das folgende Lemma liefert die naheliegende hinreichende Bedingung für eine solche kritische Fallbasis:

Lemma 28 *Sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$ gegeben. Falls CB_+ minimal ist bezüglich f_{CB} , und CB_- minimal ist bezüglich f_{CB} und CB_+ , dann ist CB kritisch.*

Beweis: Wir nehmen an, CB sei nicht kritisch. Demnach existiert eine Fallbasis CB' für welche $CB'_+ \subseteq CB_+$, $CB'_- \subseteq CB_-$ und $CB' \neq CB$ gilt, und die dennoch $f_{CB'} = f_{CB}$ erfüllt.

Nach Voraussetzung ist CB_+ minimal bezüglich f_{CB} . Damit existiert zu keinem $CB'_+ \subsetneq CB_+$ eine negative Fallbasis CB'_- , so dass $CB' := \langle CB'_+, CB'_- \rangle$ die ursprüngliche Funktion f_{CB} repräsentiert. Dies gilt natürlich auch dann, wenn wir CB'_- als Teilmenge von CB_- voraussetzen. Damit muss $CB'_+ = CB_+$ gelten, und mit $CB' \neq CB$ folgt $CB'_- \subsetneq CB_-$. Die Existenz eines solchen CB'_- ist jedoch durch die Minimalität von CB_- ausgeschlossen. Offenbar war unsere Annahme falsch, CB ist tatsächlich kritisch.

◇

Um zu zeigen, dass eine Fallbasis kritisch ist, kann man sich also auf den Nachweis der Minimalität der positiven und negativen Fallbasis beschränken. Wir geben zunächst ein hinreichendes Kriterium für die Minimalität von CB_+ an:

Lemma 29 Gegeben sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$. Wird kein Monom einer beliebigen DNF zu f_{CB} von mehr als einem Fall aus CB_+ erfüllt, so ist CB_+ minimal bezüglich f_{CB} .

Beweis: Angenommen, CB_+ ist nicht minimal bezüglich f_{CB} , dann existiert eine echte Teilmenge CB'_+ von CB_+ , zu der es eine negative Fallbasis CB'_- gibt, so dass $f_{CB} = f_{\langle CB'_+, CB'_- \rangle}$ gilt. Dafür müsste folglich mindestens ein Fall c_{ok}^0 aus CB_+ entfernt werden. Wir zeigen, dass ein solcher Fall c_{ok}^0 von keinem verbliebenen Fall aus CB'_+ positiv klassifiziert werden kann, wenn die repräsentierte Funktion unverändert bleiben soll.

Betrachten wir zu einem beliebigen c'_{ok} aus CB'_+ den Unterverband $U(c'_{ok}, c_{ok}^0)$. Ein solcher Unterverband enthält alle Fälle aus $\{0, 1\}^n$, deren Werte in allen Attributen denjenigen von c'_{ok} entsprechen, in denen dieser Fall mit c_{ok}^0 übereinstimmt. Damit entspricht ein solcher Unterverband einem Monom in diesen Attributen. Dieses kann jedoch laut Voraussetzung nicht komplett in f_{CB} enthalten sein, denn mit c'_{ok} und c_{ok}^0 würden zwei Fälle dieses Monom erfüllen. Damit existiert ein negativer Fall $c_- \in \bar{f}_{CB}$ in diesem Unterverband $U(c'_{ok}, c_{ok}^0)$, es gilt also

$$d(c'_{ok}, c_-) \preceq d(c'_{ok}, c_{ok}). \quad (1)$$

Da c_- von f_{CB} negativ klassifiziert wurde, ist es entweder selbst in CB_+ vertreten, oder es gibt zu jedem Fall aus CB_+ einen aus CB_- , welcher c_- bezüglich diesem abdeckt. Da f_{CB} gleich $f_{\langle CB'_+, CB'_- \rangle}$ sein soll, muss c_- auch bezüglich c'_{ok} durch einen Fall c'_{ng} aus CB'_- abgedeckt bleiben. Folglich gilt $c'_{ng} \in U(c_-, c'_{ok})$ und damit

$$d(c'_{ok}, c'_{ng}) \preceq d(c'_{ok}, c_-). \quad (2)$$

Mit (2), (1) und der Transitivität von \preceq gilt $d(c'_{ok}, c'_{ng}) \preceq d(c'_{ok}, c_{ok})$. Damit existiert für unser c_{ok} aus $CB_+ \setminus CB'_+$ zu jedem c'_{ok} aus CB'_+ ein solches c'_{ng} und damit kann c_{ok} für kein CB'_- in $f_{\langle CB'_+, CB'_- \rangle}$ sein. Damit war CB_+ offenbar bereits minimal.

◇

Eine wichtige Eigenschaft des Modells stellt das folgende Lemma vor.

Lemma 30 Gegeben sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$ sowie ein Konzept $f \subseteq \{0, 1\}^n$ mit $CB_+ \subseteq f$ und $CB_- \subseteq \bar{f}$. Sei außerdem F eine beliebige disjunktive Normalform zu f .

Falls das Entfernen eines Falles c_{ok} aus CB_+ dazu führt, dass dieser daraufhin negativ klassifiziert wird, so erfüllt c_{ok} mindestens ein Monom aus F , welches von keinem

anderen in CB_+ befindlichen Fall erfüllt wurde.

Falls das für jedes c_{ok} aus CB_+ der Fall ist, gilt somit $|CB_+| \leq |DNF_{\min}(f)|$.

Beweis: Für jedes i aus $1, 2, \dots, |F|$ sei M_i ein Monom über $\{0, 1\}^n$. Sei $M_1 \vee M_2 \vee \dots \vee M_{|F|}$ die f_{CB} repräsentierende Formel F . Betrachten wir nun einen Fall $c_{ok} \in CB_+$ und definieren $CB'_+ := CB_+ \setminus \{c_{ok}\}$. Es gelte nun $c_{ok} \notin f_{\langle CB'_+, CB_- \rangle}$. Wegen $CB_+ \subseteq f$ erfüllt c_{ok} mindestens ein Monom M_j aus F .

Wir nehmen nun an, dass es einen Fall c'_{ok} in CB'_+ gibt, der ebenfalls das Monom M_j erfüllt. Da c_{ok} und c'_{ok} dasselbe Monom M_j erfüllen, unterscheiden sie sich offenbar nicht in den Attributen, die in M_j vorkommen. Weiterhin wird laut Voraussetzung jeder der Fälle negativ klassifiziert, sobald er aus CB_+ entfernt wird. Damit muss ein $c_{ng} \in CB_-$ existieren, welches im Unterverband zwischen diesen Fällen, in $U(c_{ok}, c'_{ok})$, liegt.

Da sich c_{ok} und c'_{ok} aber in allen in M_j vorkommenden Attributen gleichen, können die Fälle aus $U(c_{ok}, c'_{ok})$ nur in den für M_j irrelevanten Attributen von c_{ok} und c'_{ok} abweichen. Somit erfüllt jedes Element aus $U(c_{ok}, c'_{ok})$ dieses Monom M_j . Genau in diesem Unterverband hatten wir jedoch die Existenz des Negativfalles $c_{ng} \in CB_- \subseteq \bar{f}$ vorausgesetzt, und dieser darf sicherlich kein Monom aus F erfüllen.

Unsere Annahme muss folglich falsch gewesen sein, und c_{ok} erfüllt ein bis dato unerfülltes Monom. Ist dies für alle c_{ok} aus CB_+ gegeben, so gilt damit für eine beliebige, f repräsentierende DNF F die Aussage $|CB_+| \leq |F|$, also auch für $F = DNF_{\min}(f)$.

◇

Jetzt haben wir die Mittel zur Verfügung, um eine hinreichende Bedingung für die Minimalität einer positiven Fallbasis herzuleiten, die im Allgemeinen einfacher zu überprüfen ist als die in Lemma 30 vorgestellte. Es genügt zu zeigen, dass jeder Fall aus CB_+ zu seiner eigenen positiven Klassifikation zwingend notwendig ist:

Satz 31 [SN00] Gegeben sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$. Dann gilt:

$$\forall c_{ok} \in CB_+ : c_{ok} \notin f_{\langle CB_+ \setminus \{c_{ok}\}, CB_- \rangle} \implies CB_+ \text{ ist minimal bezüglich } f_{CB}.$$

Beweis: Wenn wir in Lemma 30 für f die Funktion f_{CB} selbst einsetzen, folgt aus der linken Seite der obigen Aussage, dass kein Monom einer beliebigen DNF zu f_{CB} von zwei Fällen aus CB_+ gleichzeitig erfüllt wird. Somit können wir mit Lemma 29 folgern, dass CB_+ minimal ist bezüglich f_{CB} .

◇

Falls also das Entfernen eines Falles c_{ok} aus CB_+ dazu führt, dass dieser daraufhin negativ klassifiziert wird, so ist CB_+ minimal bezüglich f_{CB} . Es bleibt noch, ein hinreichendes Kriterium für die Minimalität der negativen Fallbasis anzugeben. Die folgende Aussage, mit der wir diesen Abschnitt abschließen werden, liefert neben einem solchen auch ein Verfahren, das aus einer negativen Fallbasis alle nicht benötigten Fälle aussortiert; die Fallbasis wird gewissermaßen verlustfrei komprimiert.

Satz 32 [SN00] *Gegeben sei eine Fallbasis $CB = \langle CB_+, CB_- \rangle$. Dann ist*

$$CB'_- := \bigcup_{c_{ok} \in CB_+} \text{NN}(c_{ok}, CB_-)$$

minimal bezüglich CB_+ und f_{CB} .

Beweis: Angenommen, CB'_- wäre nicht minimal, dann müsste man einen Fall entfernen können, ohne dass die repräsentierte Funktion sich ändert. Sei c_{ng} ein solcher Fall in CB'_- . Damit existiert ein $c_{ok} \in CB_+$ mit $c_{ng} \in \text{NN}(c_{ok}, CB_-)$, es gilt also nach Definition der nächsten Nachbarn, dass c_{ng} in $f_{\langle c_{ok}, CB'_- \setminus \{c_{ng}\} \rangle}$ ist. Mit $c_{ok} \in CB_+$ folgt daraus unter Verwendung von Bemerkung 16(b):

$$c_{ng} \in f_{\langle CB_+, CB'_- \setminus \{c_{ng}\} \rangle}.$$

Andererseits ist c_{ng} ein Fall aus CB_- und damit nach Bemerkung 16(a) nicht in f_{CB} enthalten. Nach unserer Annahme ändert sich die repräsentierte Funktion nicht durch ein Entfernen von c_{ng} , also muss auch gelten:

$$c_{ng} \notin f_{\langle CB_+, CB'_- \setminus \{c_{ng}\} \rangle}.$$

Offenbar widerspricht dies der oben hergeleiteten Aussage. Die getroffene Annahme war folglich falsch und CB'_- ist minimal bezüglich CB_+ und f_{CB} .

◇

4.2 Ein erster Lernalgorithmus

Wir wollen zunächst einen einfachen, auf Satohs fallbasiertem Modell beruhenden Lernalgorithmus vorstellen und analysieren. Dieser Algorithmus wird zunächst als negative Fallbasis CB_- alle negativen Trainingselemente verwenden. Dann wird er nacheinander alle positiven Trainingsinstanzen betrachten und, falls sie von der aktuellen Fallbasis negativ klassifiziert werden, zu CB_+ hinzufügen. Zum Abschluss entfernen wir dann noch überflüssige Fälle aus der negativen Fallbasis, indem wir nur die nächsten Nachbarn eines jeden Falles aus CB_+ darin belassen.

Wir werden nachweisen, dass die auf diese einfache Weise konstruierte Fallbasis kritisch und konsistent mit den Trainingsdaten ist, und dass die Größe der positiven Fallbasis abhängig von der Komplexität der Zielfunktion, also in diesem Fall der Länge ihrer Darstellung als DNF , nach oben beschränkt werden kann.

Algorithmus 33 *Es sei mit $X := \{0, 1\}^n$ ein Objektraum und mit $T := \langle T_+, T_- \rangle$ eine Trainingsmenge über X gegeben.*

1. $CB_+ := \emptyset$; $CB_- := T_-$;
2. FOR EACH $c_+ \in T_+$ DO
 IF $c_+ \notin f_{CB}$ THEN $CB_+ := CB_+ \cup \{c_+\}$;
3. $CB_- = \bigcup_{c_{ok} \in CB_+} \text{NN}(c_{ok}, CB_-)$;
4. OUTPUT $CB := \langle CB_+, CB_- \rangle$;

Satz 34 *Die in Algorithmus 33 konstruierte Fallbasis ist konsistent bezüglich T .*

Beweis: Wir betrachten zunächst die Fälle aus T_+ . Diese werden in Schritt 2 des Algorithmus nur zu CB_+ hinzugefügt, falls sie sonst selbst falsch, also negativ klassifiziert würden. Damit werden sie direkt nach Ihrer Betrachtung auf jeden Fall korrekt klassifiziert. In Schritt 2 werden gegebenenfalls weitere Fälle zu CB_+ hinzugefügt, und in Schritt 3 eventuell einige aus CB_- entfernt. Mit Eigenschaft (b) aus Bemerkung 16 werden somit alle Fälle aus T_+ von der resultierenden Fallbasis positiv klassifiziert.

Untersuchen wir die negativen Trainingsfälle aus T_- , so klassifizieren sich diese gemäß Bemerkung 16(a) selbst korrekterweise negativ. Da wir an CB_- in Schritt 2 keine Veränderung vornehmen, gilt dies weiterhin. Schritt 3 ist nach Satz 20 ebenfalls konsistenzerhaltend.

◇

Satz 35 Sei $CB = \langle CB_+, CB_- \rangle$ die in Algorithmus 33 konstruierte Fallbasis. Dann gilt:

- (i) CB_+ ist minimal bezüglich f_{CB} .
- (ii) CB_- ist minimal bezüglich f_{CB} und CB_+ .

Die Fallbasis CB ist damit kritisch.

Beweis: Dank Satz 32 ist sofort klar, dass CB_- nach Schritt 3 minimal bezüglich f_{CB} und CB_+ ist.

Betrachten wir nun CB_+ . Ein Fall aus T_+ wurde in Schritt 2 nur dann zu CB_+ hinzugefügt, wenn er sonst negativ klassifiziert würde. Demnach ist die positive Fallbasis CB_+ mit Satz 31 nach diesem Schritt minimal bezüglich f_{CB} . In Schritt 3 ändert sich CB_+ nicht, und auch die repräsentierte Funktion bleibt mit Satz 20 dieselbe. Da die Minimalität von CB_+ nur bezüglich f_{CB} gelten muss und damit nicht direkt von CB_- abhängt, bleibt sie auch nach Schritt 3 erhalten.

◇

Mit diesem Satz können wir nun Folgendes für unseren einfachen Algorithmus zeigen:

Satz 36 Die von Algorithmus 33 konstruierte positive Fallbasis CB_+ enthält maximal $|DNF_{\min}(f_t)|$ viele Fälle.

Beweis: Offenbar wurden in die resultierende Fallbasis nur Fälle aus T aufgenommen, es gilt also $CB_+ \subseteq T_+ \subseteq f_t$ und $CB_- \subseteq T_- \subseteq \bar{f}_t$. Ferner ist CB_+ nach Satz 35 minimal, womit das Entfernen eines Falles c_{ok} aus CB_+ sofort zur negativen Klassifikation desselben führt. Damit ist Lemma 30 auf CB anwendbar, und wir erhalten die gewünschte Aussage.

◇

Damit ist es also möglich, die Größe von CB_+ durch den Wert $|DNF_{\min}(f_t)|$, welcher als ein Maß für die “Schwierigkeit” der Zielfunktion interpretiert werden kann, nach oben zu beschränken. Könnte man ein ähnliches Resultat für CB_- herleiten, so hätte man eine Aussage über die Größe einer vollständigen Repräsentation von f_t im Hypothesenraum gewonnen.

Es kann jedoch für den vorgestellten Algorithmus 33 unglücklicherweise nachgewiesen werden, dass die benötigte Fallanzahl einer negativen Fallbasis CB_- selbst für eine einfache Zielfunktion exponentiell mit n wachsen kann². Mehr noch, wir zeigen, dass für diese Funktion unter Annahme einer bestimmten Verteilung \mathcal{D} aber unabhängig von der Beschaffenheit des Samples selbst, exponentiell viele Trainingsbeispiele erforderlich sind, um die Fehlerwahrscheinlichkeit der zu der von diesem Algorithmus ausgegebenen Fallbasis repräsentierten Hypothese unter einen festen Anteil $\varepsilon > 0$ zu drücken.

Beispiel 37 Sei für eine natürliche Zahl die Zielfunktion f_t das Monom $\bigwedge_{i=1}^n c_i$. Damit gilt offenbar $|DNF(f_t)| = 1$ und $|CNF(f_t)| = n$. Jetzt nehmen wir an, die Verteilung \mathcal{D} sei so geartet, dass für einen Vektor c , der gemäß \mathcal{D} gezogen wurde, $\mathbf{P}_{\mathcal{D}}(c = 1^n) = 1/2$ gelte. Weiterhin sei die verbliebene Wahrscheinlichkeit gleichverteilt auf den $\binom{n}{\lfloor n/2 \rfloor}$ Elementen, die genau $\lfloor n/2 \rfloor$ Einsen enthalten.

Da jeder Negativfall offenbar gleich viele Bits Abstand zum einzigen Positivfall besitzt, kann keiner der Negativfälle einen anderen abdecken. Damit ist jeder Fall, der in der Trainingsmenge vorkommt, auch in der resultierenden Fallbasis vertreten, was bedeutet, dass diese bis zu $\binom{n}{\lfloor n/2 \rfloor}$, also exponentiell in n viele Elemente enthalten kann. Des Weiteren heißt dies aber auch, dass jeder dieser Fälle ausschließlich sich selbst klassifiziert und keinerlei generalisierende Wirkung besitzt. Aufgrund der zugrundeliegenden uniformen Verteilung auf diesen Fällen benötigt man folglich ebenfalls exponentiell viele Fälle, um eine Fehlerwahrscheinlichkeit garantieren zu können, die geringer als ein gegebenes $\varepsilon > 0$ ist. Damit kann Algorithmus 33 nach Definition 7 kein effizienter PAC-Algorithmus sein.

In diesem Beispiel sind trotz einer eigentlich einfachen Zielfunktion weitgehend unabhängig von der Beschaffenheit des Samples³ inakzeptabel viele Trainingsfälle notwendig, da die generalisierende Wirkung eines Negativfalles zu gering ist. Da wir jedoch nach Satz 35 nachweislich die kleinsten Fallmengen ausgewählt haben, die

²Folglich kann der zu durchsuchende Hypothesenraum doppelt exponentiell in n viele Hypothesen enthalten. Damit scheitert bereits hier der Versuch, mit Satz 8 zu beweisen, dass es sich bei Algorithmus 33 um einen effizienten PAC-Algorithmus handelt. Den Nachweis, dass er kein solcher ist, führt jedoch erst Beispiel 37.

³Lediglich der Fall 1^n sollte vorkommen, dies ist aber angesichts seiner hohen Auftretenswahrscheinlichkeit keine einschränkende Forderung.

das Sample noch korrekt klassifizieren, kommen wir hier offenbar mit einer einfachen Speicherung von Trainingselementen nicht weiter.

4.3 Lernalgorithmus nach Satoh und Nakagawa

Warum erreicht unser einfacher Algorithmus im Allgemeinen nicht die gewünschte Performance?

Wir bemerken, dass die klassifizierende Wirkung eines Negativfalles c_{ng} aus CB_- gegenüber einem bestimmten $c_{ok} \in CB_+$ umso schlechter wird, je größer dessen Abstand zu dem positiven Bereich ist, der den von c_{ok} abgedeckten Klauseln aus $DNF(f_t)$ entspricht. In dem Beispiel aus Abbildung 4.3 kann der Algorithmus über die weißen und schraffierten Felder anhand der Trainingsdaten begründete Aussagen treffen. Für die grauen Felder jedoch existieren keinerlei Anhaltspunkte. Sie werden gewissermaßen “auf gut Glück” positiv klassifiziert.

Es liegt folglich nahe, c_{ng} in Richtung von c_{ok} zu verschieben, jedoch nicht so weit, dass es in diesen Positivbereich fällt. Dessen Grenzen allerdings sind uns natürlich unbekannt, sonst gäbe es ja nichts mehr zu lernen. Wie also überprüfen wir, ob wir uns bereits im positiven Bereich befinden oder nicht?

Wenn man sich über eine Entscheidung im Unklaren ist, so bietet die typisch menschliche Herangehensweise zwei Möglichkeiten:

1. Man versucht, die Entscheidung aufgrund von vorliegenden Indizien zu treffen.
2. Man fragt jemanden, der sich damit auskennt.

Unglücklicherweise sind Indizien meist in zu geringer Menge und mit unzureichender Aussagekraft gegeben, um mit Sicherheit eine korrekte Antwort ableiten zu können. Vielmehr lässt sich die Gesamtmenge aller möglichen Lösungen im Normalfall mit Hilfe dieser Anzeichen auf eine kleinere Untermenge, die der wahrscheinlichsten Hypothesen, einschränken, von denen eine einzelne schließlich zufällig ausgewählt wird. Diese Methode hat den Vorteil, dass sie die Existenz einer Instanz, die unsere Frage sicher beantworten kann, nicht voraussetzt. Jedoch ist die beschriebene Vorgehensweise meistens recht kompliziert zu realisieren und die Richtigkeit der “geratenen” Hypothese bleibt unsicher.

Wir werden daher zunächst annehmen, wir hätten einen Fachmann zur Verfügung, der uns, um auf Beispiel 5 zurückzukommen, die Frage beantworten kann, ob ein bestimmter Generator mit einem bestimmten Motor verwendet werden kann. Jedoch sei dieser nicht dafür ausgebildet, allgemeine Kriterien für das Zusammenpassen zweier solcher Teile zu definieren. Die Aufgabe des Findens dieser Bedingungen

$$c_{ok}$$
Fall aus T_+ Fall aus T_- Fall aus f_t Fall aus \bar{f}_t

Positiv klassifizierter Fall, Korrektheit der Klassifikation unsicher

Abbildung 4 Die Klassifikationslücke von Algorithmus 33

bleibt folglich dem Lernalgorithmus vorbehalten.

Diese Situation entspricht einem erweiterten Lernmodell, in dem ein Orakel (unser Experte) zur Verfügung steht, welches sogenannte Membership Queries beantworten kann.

Definition 38 Gegeben sei ein Orakel ω , welches für jedes $c \in \{0, 1\}^n$ in konstanter Zeit die Frage entscheiden kann, ob c in f_t ist. Es gilt also $\omega(c) = f_t(c)$. Eine entsprechende Anfrage an das Orakel nennen wir **Membership Query**.

Bisher hatten wir uns darauf verlassen, dass uns die Trainingsmenge als Informationsquelle ausreicht. Sie stellte uns die Funktionswerte der “wahrscheinlichsten” Fälle zur Verfügung. In obigem Beispiel genügt dies nicht, da wir sehr viele gleichwahrscheinliche Fälle vorausgesetzt haben, wobei der Funktionswert eines Falles modellbedingt keine Rückschlüsse auf den eines anderen zuließ.

Wie bereits angedeutet, wäre es hilfreich, die Negativfälle möglichst direkt an den positiven Bereich angrenzen zu lassen. Um dabei keine Fehler zu machen, benötigen

wir in Beispiel 37 jedoch Funktionswerte an Stellen mit der Wahrscheinlichkeit 0, also würden wir unendlich lange auf solch einen Fall in der Trainingsmenge warten. Mit Hilfe von Membership Queries lassen sich die Werte solcher Fälle dennoch mit Sicherheit erfragen.

Indem wir Membership Queries zulassen, erweitern wir also das ursprüngliche Modell aus Definition 7, das zwar gewissermaßen auch Anfragen an ein Orakel zuließ, welches jedoch nur für *zufällig* ausgewählte Instanzen den Funktionswert vorgab. Deswegen werden wir, wenn wir von PAC-Lernbarkeit sprechen, diese Erweiterung stets gesondert erwähnen.

So stellen Satoh und Nakagawa in [SN00] einen eleganten Lernalgorithmus vor, der tatsächlich einen effizienten PAC-Algorithmus für eine interessante Funktionenklasse darstellt, wenn man eine bestimmte Anzahl von Membership Queries erlaubt. Diesen wollen wir hier vorstellen.

Wie in Algorithmus 33 wird mit einer leeren positiven Fallbasis CB_+ sowie der Menge negativer Trainingsfälle als negativer Fallbasis CB_- gestartet. Ebenfalls analog zum einfachen Algorithmus wird anschließend jeder positive Trainingsfall nur dann zu CB_+ hinzugefügt, falls er sonst negativ klassifiziert würde. Neu ist, dass hier die vorhandenen Negativfälle an die neue Situation angepasst werden, sobald dieser Fall eintritt.

Wie bei einem positiv geladenes Teilchen lassen wir die Anziehungskraft des neuen Positivfalles in alle Richtungen auf die ihm am nächsten gelegenen Negativfälle wirken. Diese werden dadurch angezogen — das heißt, abweichende Bits werden an die Werte des Positivfalles angeglichen — können sich ihm aber nur soweit nähern, wie sie sich in negativem Terrain befinden. Würden sie dieses irgendwann in allen Dimensionen, in denen sie sich noch auf den positiven Fall zubewegen können, verlassen, so haben sie ihre Endposition erreicht und lagern sich dort an.

Um diese Situation fehlerfrei erkennen zu können, muss sich der Algorithmus des Hilfsmittels der Membership Queries bedienen, da positive und negative Bereiche von der unbekanntes Zielfunktion bestimmt sind. Die Aufgabe, einen einzelnen Negativfall c_- an den positiven Fall c_{ok} heranzuführen, wird von der Funktion `generalize()` übernommen:

```

generalize( $c_-$ ,  $c_{ok}$ )

FOR EACH  $j \in \chi_n^{-1}(d(c_{ok}, c_-))$  DO BEGIN
   $c_j := c_- \oplus \chi_n(\{j\})$ 
  IF  $\omega(c_j) = 0$  THEN
    RETURN generalize ( $c_j$ ,  $c_{ok}$ )
END FOR
RETURN  $c_-$ 

```

Der so erzeugte Fall wird zu CB_- hinzugefügt und deckt dabei den Ausgangsfall c_- bezüglich c_{ok} ab. Diese Eigenschaft begründet sich darin, dass von c_- ausgehend ausschließlich “in Richtung” c_{ok} gesucht wird. Das heißt, wir gleichen sukzessive die Attributswerte von c_- an, die sich von denen in c_{ok} unterscheiden. Wir können also schreiben:

Bemerkung 39 Für zwei Fälle $c_- \in \bar{f}_t$ und $c_{ok} \in f_t$ gilt:

$$c_- \in \bar{f}_{\{c_{ok}, \{\text{generalize}(c_-, c_{ok})\}\}}.$$

Jedoch dürfen wir den ursprünglichen Fall noch nicht aus CB_- entfernen, da er im Allgemeinen auch zu anderen Positivfällen benachbart ist. Tritt diese Situation auf, so wird derselbe Ausgangsfall also noch einmal bewegt, diesmal in Richtung des anderen Positivfalles. Weiterhin ist zu beachten, dass durch das Hinzufügen eines Negativfalles als Ergebnis von **generalize**() ein zuvor betrachteter und nicht in CB_+ übernommener Fall aus T_+ plötzlich notwendig werden kann. Deshalb wird mit der Anweisung **GOTO 2** nach jedem Hinzufügen eines Positivfalles und entsprechender Bearbeitung von CB_- ganz T_+ neu durchlaufen. Es wird dennoch kein Fall zweimal zu CB_+ hinzugefügt, da er sich ja gemäß Bemerkung 16(a) selbst positiv klassifiziert. Folglich terminiert die Schleife. In Schritt 3 übernehmen wir schließlich nur noch diejenigen Fälle in das finale CB_- , welche aus **generalize**() entstanden sind. Wir werden sehen, dass wir dadurch wie in Algorithmus 33 eine kritische Fallbasis erhalten.

Der Lernalgorithmus, wie wir ihn gleich vorstellen werden, unterscheidet sich äußerlich recht stark von dem in [SN00] abgebildeten Original, basiert aber auf derselben Idee. Neben einigen kosmetischen Änderungen wurde vor allem eine Anpassung an das hier verwendete PAC-Lernbarkeitsmodell durchgeführt, um eine diesbezügliche Bewertung des Algorithmus zu ermöglichen.

c_{ok} Fall aus T_+ Fall aus T_- Fall aus f_t Fall aus \bar{f}_t

Von Alg. 33 fälschlicherweise positiv klassifizierte Fälle

Abbildung 5 Die Negativen Trainingsfälle werden durch `generalize()` in den grauen Bereich in Richtung c_{ok} geschoben, so dass jeder in einer von dessen ‘Spitzen’ landet.

Algorithmus 40 [SN00] *Es sei mit $X := \{0, 1\}^n$ ein Objektraum und mit $T := \langle T_+, T_- \rangle$ eine Trainingsmenge über X gegeben.*

```

1.  $CB_+ := \emptyset$ ;  $CB_- := T_-$ ;

2. FOR EACH  $c_+ \in T_+$  DO
  IF  $c_+ \notin f_{CB}$  THEN BEGIN
     $c_{ok} := c_+$ ;
     $CB_+ := CB_+ \cup \{c_{ok}\}$ ;
     $M_-^{c_{ok}} := \bigcup_{c_- \in T_-} \text{generalize}(c_-, c_{ok})$ ;
     $CB_- := CB_- \cup M_-^{c_{ok}}$ ;
    GOTO 2.;
  END IF;

3.  $CB_- := \bigcup_{c_{ok} \in CB_+} M_-^{c_{ok}}$ ;

4. OUTPUT  $CB := \langle CB_+, CB_- \rangle$ ;
```

Lemma 41 *Algorithmus 40 terminiert immer. Seine Laufzeit verhält sich polynomiell in $|T_+|$, $|T_-|$ und n .*

Beweis: Schritt 1 ist offenbar unkritisch.

Betrachten wir nun Schritt 2. Wir stellen fest:

- (a) CB_+ ist eine Auswahl von Fällen aus T_+ und kann somit nicht mehr als $|T_+|$ Fälle beinhalten.
- (b) CB_- kann maximal die Größe $(1 + |T_+|) \cdot |T_-|$ besitzen, da zu Beginn $CB_- = T_-$ gilt, und jedes $M_-^{c_{ok}}$ offenbar höchstens $|T_-|$ Fälle enthält. Damit ist die Fallbasisgröße zu jeder Zeit polynomiell in $|T_-|$, $|T_+|$ und n beschränkt, und folglich ist die Beantwortung der Frage $c_+ \notin f_{CB}$ unproblematisch.
- (c) Die Ausführung von `generalize()` kann maximal $O(n^2)$ Schritte benötigen: Der Abstand zweier Fälle kann höchstens (1^n) sein. Innerhalb einer Inkarnation der Funktion werden also maximal n Bits des Negativfalles c_- daraufhin überprüft, ob sie negiert werden dürfen ohne dass c_- dadurch in den positiven Bereich fällt. `Generalize()` wird nur dann neu aufgerufen, falls ein solches Bit gefunden wurde. Da kein Bit zweimal “in Richtung c_+ ” gedreht werden kann, kann ein solcher rekursiver Aufruf insgesamt nur n -mal auftreten.
- (d) Jede Anweisung im Schleifenrumpf wird insgesamt maximal $|T_+|^2$ -mal ausgeführt: Da jedes c_+ aus T_+ nur einmal falsch klassifiziert werden kann (danach ist es in CB_+ enthalten und klassifiziert sich selbst korrekt), kann auch nur einmal pro $c_+ \in T_+$ der GOTO-Sprung ausgeführt werden. Die Schleife, die selbst maximal $|T_+|$ mal durchlaufen wird, wird damit höchstens $|T_+|$ mal neu gestartet.

Somit benötigt auch Schritt 2 lediglich polynomielle Laufzeit in $|T_+|$, $|T_-|$ und n . Da jede der Mengen $M_-^{c_{ok}}$ wie in (b) erwähnt nicht mehr als $|T_-|$ Fälle enthalten kann, gilt dies auch für Schritt 3.

◇

Satz 42 *Die in Algorithmus 40 konstruierte Fallbasis ist konsistent bezüglich T .*

Beweis: Betrachten wir ein beliebiges $c_+ \in T_+$. Wenn c_+ selbst in CB_+ ist, so ist es natürlich auch in f_{CB} . Sei also c_+ nicht in CB_+ . Würden wir wie in Algorithmus 33 lediglich eine Auswahl der mit T_+ gegebenen Fälle als CB_- übernehmen, so könnten wir hier ganz analog argumentieren. Jedoch erarbeitet die Routine `generalize()` Fälle, die nicht in der Trainingsmenge vorkommen müssen. Somit ist es möglich,

dass zuvor korrekt klassifizierte Positivfälle durch diese neuen Instanzen von den sie klassifizierenden Referenzfällen aus CB_+ abgeschnitten werden. Aus diesem Grund wird jeder Fall aus T_+ nach dem Hinzufügen eines solchen neuen Falles zu CB_- neu überprüft, und im Falle einer Fehlklassifikation zu CB_+ hinzugefügt. In Schritt 3 wird die positive Fallbasis nicht angetastet und die negative allenfalls verkleinert. Folglich gilt nach Beendigung des Algorithmus $T_+ \subseteq f_{CB}$.

Untersuchen wir einen negativen Trainingsfall c_- , so gilt mit Bemerkung 39, dass jeder aus $\text{generalize}(c_-, c_{ok})$ neu entstandene Negativfall seinen Ausgangsfall c_- bezüglich c_{ok} abdeckt. Da im Laufe des Algorithmus der betrachtete Fall c_- bezüglich jedem $c_{ok} \in CB_+$ einmal generalisiert wird, muss er in \bar{f}_{CB} liegen. Gemäß Satz 20 verändert Schritt 3 die repräsentierte Funktion nicht. Folglich ist die konstruierte Fallbasis konsistent bezüglich T .

◇

Satz 43 Sei $CB = \langle CB_+, CB_- \rangle$ die in Algorithmus 40 konstruierte Fallbasis. Dann gilt:

- (i) CB_+ ist minimal bezüglich f_{CB} .
- (ii) CB_- ist minimal bezüglich f_{CB} und CB_+ .

Die Fallbasis CB ist damit kritisch.

Beweis: CB_- ist aufgrund von Schritt 3 mit Satz 32 minimal bezüglich f_{CB} und CB_+ . Um zu zeigen, dass auch CB_+ minimal ist bezüglich f_{CB} werden wir erneut Satz 31 verwenden. Es ist damit zu zeigen, dass jedes c_{ok} aus der positiven Fallbasis negativ klassifiziert wird, falls es aus dieser entfernt wird.

Dies weisen wir mit vollständiger Induktion nach: Zu Beginn ist CB_+ leer und daher unproblematisch. Nehmen wir nun an, CB_+ erfülle die besagte Eigenschaft zu Beginn eines beliebigen Schleifendurchlaufs aus Schritt 2. Ein Fall aus T_+ wird jetzt nur dann zu CB_+ hinzugefügt, wenn er sonst negativ klassifiziert würde. Das bedeutet im Umkehrschluss, dass er keine der bereits in CB_+ vorhandenen Fälle klassifizieren kann. Damit bleibt jedes CB_+ notwendig zu seiner eigenen Klassifikation. Das Hinzufügen von Negativfällen zu CB_- mit $\text{generalize}()$ ändert mit Bemerkung 16(c) nichts an dieser Eigenschaft; sie gilt damit nach jedem Schleifendurchlauf.

Damit ist CB_+ nach Schritt 2 minimal bezüglich f_{CB} . Wir können nun analog zum Beweis von Satz 35 argumentieren, dass die Minimalität von CB_+ durch Schritt 3 nicht aufgehoben werden kann, da sich lediglich CB_- und nicht f_{CB} ändert.

◇

Wir wollen an dieser Stelle eine Bemerkung über die Stabilität der Fallbasen einschieben. Wie wir schon in Abschnitt 4.1.3 in der Einleitung zu Definition 27 nachgewiesen hatten, sind Fallbasen dieses Modells nicht notwendigerweise stabil. Wir hatten gezeigt, dass man für $CB_+ := \{0000\}$ und $CB_- := \{1000, 0101\}$ den Fall 0110 nicht zu CB_+ hinzufügen darf, ohne dass sich die repräsentierte Funktion f_{CB} ändert.

Eben diese Fallbasis ist aber auch das Ergebnis von Algorithmus 40, wenn die zu lernende Zielfunktion $f_t = f_{CB}$ ist, die negative Trainingsmenge die Fälle 1000 und 0101 enthält, und das erste erhaltene Positivbeispiel der Fall 0000 ist.

Bemerkung 44 *Die in Algorithmus 40 konstruierte Fallbasis ist im Allgemeinen nicht stabil.*

Dies ist jedoch nur eine Randbemerkung, unser eigentliches Interesse gilt der Suche nach einer oberen Schranke für die Größe der negativen Fallbasis. Der folgende Satz liefert diesbezüglich die wesentliche Aussage.

Satz 45 *Gegeben seien eine Fallbasis $CB = \langle CB_+, CB_- \rangle$ und eine boolesche Funktion f_t , deren Komplement \bar{f}_t eine DNF-Repräsentation \bar{F} besitzt. Es gelte $CB_+ \subseteq f_t$ und $CB_- \subseteq \bar{f}_t$, und für jedes c_{ok} aus CB_+ sei $M_-^{c_{ok}} := \bigcup_{c_- \in CB_-} \text{generalize}(c_-, c_{ok})$. Dann erfüllt jeder Fall c_{ng} aus $M_-^{c_{ok}}$ ein Monom aus der DNF \bar{F} , welches von keinem anderen Fall aus $M_-^{c_{ok}}$ erfüllt wird. Insbesondere ist*

$$|M_-^{c_{ok}}| \leq |CNF_{\min}(f_t)|.$$

Beweis: Sei $c_{ok} \in CB_+$ und $c_{ng} \in M_-^{c_{ok}}$. Angenommen c_{ng} erfüllt ein bestimmtes Monom M aus \bar{F} . Damit sind diejenigen Attribute, die für M relevant sind, entsprechend belegt. Da c_{ng} nach Definition von $M_-^{c_{ok}}$ aus Anwendung der Funktion $\text{generalize}()$ entstanden ist, wurden alle weiteren Attribute an die in c_{ok} angeglichen. Somit kann pro c_{ok} und pro Monom von \bar{F} nur ein Negativfall existieren und es gilt $|M_-^{c_{ok}}| \leq |\bar{F}|$.

Da aus jeder CNF F zu f_t natürlich eine DNF \bar{F} zu \bar{f}_t derselben Größe konstruiert werden kann, gilt obige Aussage auch, wenn \bar{F} die Negation einer minimalen CNF zu f_t verkörpert. Damit erhalten wir insgesamt $|M_-^{c_{ok}}| \leq |CNF_{\min}(f_t)|$.

◇

Damit bietet Satohs Algorithmus einen großen Vorteil gegenüber der einfachen Variante aus Abschnitt 4.2: Die Größe der negativen Fallbasis lässt sich anhand der Komplexität einer *CNF*-Darstellung der Zielfunktion nach oben abschätzen. Insgesamt erhalten wir die folgende Aussage.

Satz 46 Jede von Algorithmus 40 beim Lernen einer booleschen Funktion f_t konstruierte Fallbasis $CB = \langle CB_+, CB_- \rangle$ besteht aus nicht mehr als

$$|CB| \leq |DNF_{\min}(f_t)| \cdot (1 + |CNF_{\min}(f_t)|)$$

vielen Fällen. Insbesondere gilt:

$$|CB_+| \leq |DNF_{\min}(f_t)| \quad \text{und} \quad |CB_-| \leq |DNF_{\min}(f_t)| \cdot |CNF_{\min}(f_t)|.$$

Beweis: Wie im Beweis zu Satz 43 ausgeführt wurde, führt das Entfernen eines beliebigen Falles aus CB_+ zu dessen Negativklassifikation. Da die konstruierte Fallbasis nach Satz 42 konsistent mit den Trainingsdaten zur Zielfunktion f_t ist, muss $CB_+ \subseteq f_t$ und $CB_- \subseteq \bar{f}_t$ gelten. Mit Lemma 30 folgt dann $|CB_+| \leq |DNF_{\min}(f_t)|$.

Für CB_- folgt mit Satz 45, dass für jeden Fall c_{ok} aus CB_+ die Menge $M_-^{c_{ok}}$ maximal $|CNF_{\min}(f_t)|$ viele Elemente besitzen kann. Damit ist die Größe von CB_- als die Vereinigung aller $M_-^{c_{ok}}$ nach oben durch $|DNF_{\min}(f_t)| \cdot |CNF_{\min}(f_t)|$ beschränkt.

◇

Bemerkung 47 Um die Anzahl benötigter Membership Queries möglichst klein zu halten und ebenfalls durch eine Funktion in n , $|DNF(f_t)|$ und $|CNF(f_t)|$ nach oben abzuschätzen, können in Algorithmus 40 die beiden Zeilen

$$\begin{aligned} M_-^{c_{ok}} &:= \bigcup_{c_- \in CB_-} \text{generalize}(c_-, c_{ok}); \\ CB_- &:= CB_- \cup M_-^{c_{ok}}; \end{aligned}$$

ersetzt werden durch

```

CB'_- := CB_-;
FOR EACH c_- ∈ CB'_- DO
  IF c_- ∈ NN(c_{ok}, CB_-) THEN BEGIN
    M_-^{c_{ok}} := M_-^{c_{ok}} ∪ {generalize(c_-, c_{ok})};
    CB_- := CB_- ∪ M_-^{c_{ok}};
  END IF;
END FOR;

```

wobei alle zuvor bewiesenen Eigenschaften der resultierenden Fallbasis erhalten bleiben. Der entscheidende Unterschied in der unteren Variante⁴ besteht darin, dass jeder generalisierte Fall einzeln zu CB_- hinzugefügt wird. Auf diese Weise kann man nach jedem solchen Schritt die nächsten Nachbarn von c_{ok} im aktuellen CB_- neu bestimmen. Weil der generalisierte Fall, wie wir in Satz 45 erläutert haben, ein ganzes Monom aus einer DNF zu \bar{f}_t gegenüber c_{ok} abdeckt, wird kein anderer dasselbe Monom erfüllender Fall aus CB'_- in $NN(c_{ok}, CB_-)$ sein. Somit wird verhindert, dass `generalize()` überflüssigerweise für einen weiteren Fall desselben Monoms aufgerufen wird. Pro hinzugefügtem c_{ok} wird die Routine `generalize()` also nicht öfter als benötigt ausgeführt, das heißt höchstens $|CNF_{\min}(f_t)|$ -mal.

Folglich ändert sich die maximale Anzahl benötigter Membership Queries von $|DNF_{\min}(f_t)| \cdot |T_-| \cdot n^2$ in der alten Variante auf

$$|DNF_{\min}(f_t)| \cdot |CNF_{\min}(f_t)| \cdot n^2$$

in der neuen Version. Damit hängt die Anzahl nur noch von der Zielfunktion selbst ab und nicht mehr von der Größe der Trainingsmenge.

Da wir mit Satz 46 nun eine obere Grenze für die Fallbasisgröße zur Verfügung haben, können wir diese für die Anwendung von Occams Razor verwenden und damit auf recht elegante Weise zeigen, dass Satohs Algorithmus ein effizienter PAC-Algorithmus für polynomiell repräsentierbare Funktionsklassen ist, wenn man eine bestimmte Anzahl von Membership Queries zulässt. Da solche Funktionsklassen jedoch aufgrund der nicht zwangsläufig eingeschränkten Eingabedimension unendlich groß sein können, trifft dies auch auf die benötigte Hypothesenklasse zu. Natürlich kann aber die Stelligkeit einer Zielfunktion den Eingabeinstanzen entnommen werden. Damit schränkt sich die Hypothesenklasse spätestens nach dem ersten Trainingsfall entsprechend ein. Somit kann man aus der PAC-Lernbarkeit jeder auf n -stellige Funktionen eingeschränkten Konzeptklasse \mathcal{F}_n auch auf die von $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$ schließen.

Definition 48 Sei eine natürliche Zahl n und eine Funktion $g : \mathbb{N} \rightarrow \mathbb{N}$ gegeben. Wir setzen

$$\mathcal{H}^g := \bigcup_{n \in \mathbb{N}} \mathcal{H}_n^g,$$

wobei \mathcal{H}_n^g als die Menge derjenigen booleschen Funktionen definiert sei, die sich von Fallbasen über $\{0, 1\}^n$ repräsentieren lassen, die maximal die Größe $g(n)$ besitzen.

Wir zeigen jetzt, dass jede polynomiell repräsentierbare Funktionenklasse auch effizient von Algorithmus 40 erlernt werden kann.

⁴In der ursprünglichen Algorithmen-Definition wurde darauf aus Übersichtlichkeitsgründen verzichtet.

Satz 49 Für ein Polynom p und eine natürliche Zahl n sei \mathcal{F}_n^p wie in Bemerkung 24 eine Klasse n -stelliger boolescher Funktionen, so dass für jedes $f \in \mathcal{F}_n^p$ gilt:

$$|DNF_{\min}(f)| \leq p(n) \quad \text{und} \quad |CNF_{\min}(f)| \leq p(n).$$

Erlaubt man die Durchführung von bis zu $(p(n))^2 n^2$ Membership Queries zum Erlernen einer Funktion $f \in \mathcal{F}_n^p$, so ist die Klasse $\mathcal{F}^p = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n^p$ effizient PAC-lernbar unter Verwendung der Hypothesenklasse $\mathcal{H}^{p(1+p)}$.

Beweis: Wir werden mit Hilfe von Satz 9 zeigen, dass Algorithmus 40 ein effizienter PAC-Algorithmus für jedes \mathcal{F}_n^p ist, der nicht mehr als $(p(n))^2 n^2$ Membership Queries verwendet. Dazu zeigen wir zunächst, dass die Voraussetzungen für Occams Razor (Satz 8) gegeben sind.

Nach Satz 42 ist jede konstruierte Fallbasis konsistent mit den Trainingsdaten.

Weiterhin ist die Polynomialität der Algorithmen-Laufzeit in der Objekt-Dimension n und der Samplegröße m sowie $1/\varepsilon$ und $1/\delta$ zu zeigen. Die Laufzeit war nach Lemma 41 in der ursprünglichen Form des Algorithmus offenbar noch polynomiell in n , m . Durch die Maßnahmen aus Bemerkung 47, die wir wegen der Forderung nach maximal $(p(n))^2 n^2$ Membership Queries hier verwenden müssen, ändert sich diese Laufzeitabschätzung. Jedoch ist die Menge $\text{NN}(c_{ok}, CB_-)$ sicherlich in der Zeit $O(n \cdot |CB_-|^2)$ bestimmbar. Somit ändern die Modifikationen nichts an der Polynomialität der Gesamtlaufzeit. Weiterhin bemerken wir, dass die Parameter $1/\varepsilon$ und $1/\delta$ die Laufzeit offenbar nicht direkt beeinflussen, und damit sind zunächst alle Voraussetzungen für die Anwendung von Occams Razor erfüllt.

Um zu zeigen, dass der Algorithmus auch ein *effizienter* PAC-Algorithmus ist, können wir nun Satz 9 verwenden, der besagt, dass zusätzlich zu den Voraussetzungen von Occams Razor lediglich der Logarithmus der Größe der verwendeten Hypothesenklasse polynomiell in $1/\varepsilon$, $1/\delta$ und n beschränkt sein muss. Wir erinnern uns, dass in diesem Fall die aus Occams Razor abgeleitete hinreichende Größe m der Trainingsmenge ebenfalls polynomiell in $1/\varepsilon$, $1/\delta$ und n beschränkt werden kann, und wir damit eine in diesen Parametern polynomielle Obergrenze für die Gesamtlaufzeit besitzen.

In Satz 46 hatten wir gezeigt, dass für jede erlernte Fallbasis $|CB_+| \leq |DNF_{\min}(f_t)|$ und $|CB_-| \leq |DNF_{\min}(f_t)| \cdot |CNF_{\min}(f_t)|$ gilt. Da sowohl $|DNF_{\min}(f_t)|$ als auch $|CNF_{\min}(f_t)|$ nach Voraussetzung durch $p(n)$ beschränkt sind, liegt offenbar jede eine solche Fallbasis repräsentierende Hypothese in der Menge $\mathcal{H}_n^{p(1+p)}$. Deren Kardinalität $|\mathcal{H}_n^{p(1+p)}|$ kann $2^{p(n) \cdot (1+p(n))}$ nicht überschreiten, und damit ist auch $\log |\mathcal{H}_n^{p(1+p)}|$ offenbar polynomiell in n beschränkt.

◇

Dieses Resultat ist recht bemerkenswert, besagt es doch, dass eine boolesche Funktion, deren Darstellung in *DNF* beziehungsweise *CNF* nicht übermäßig groß ist, *effizient* unter Verwendung einer, komplexitätstheoretisch gesehen, akzeptablen Anzahl von Membership Queries erlernt werden kann.

5 Komplexitätsanalyse

Wie wir bereits zu Beginn des vorangehenden Abschnitts 4.3 festgestellt hatten, ist die Verfügbarkeit eines Orakels (also im Allgemeinen eines Experten), welches zur Beantwortung einer hinreichenden Anzahl von Anfragen zur Verfügung steht, in der Praxis auch aus wirtschaftlichen Gründen eher die Ausnahme.

Betrachten wir als Beispiel wieder die beiden Maschinenbauteilklassen Motor und Generator mit n_m beziehungsweise n_g Attributen. Wir wollen entscheiden lassen, ob zwei bestimmte Instanzen dieser Klassen prinzipiell zusammen verwendbar sind. Um binäre Eingabedaten zu erhalten, gehen wir wie in Beispiel 5 davon aus, dass jedes aus einem bestimmten Motor und Generator bestehende Tupel durch vollständigen Attributsvergleich zu einem Binärvektor der Länge $n = n_m \cdot n_g$ wird.

Nehmen wir an, dass ein Motor als recht komplexes Bauteil in einer solchen Datenbank durch mindestens 20 und ein Generator etwa durch 10 Eckdaten repräsentiert wird, so besitzt jeder Eingabevektor folglich $n = 200$ Binärstellen. Gehen wir nun weiterhin davon aus, dass die Zielfunktion beispielsweise ein einfaches Monom in zwei Variablen sei, so benötigen wir bereits bis zu $|DNF_{\min}(f_t)| \cdot |CNF_{\min}(f_t)| \cdot n^2 = 1 \cdot 2 \cdot 4000 = 8000$ Membership Queries. Und das nur, um den Zusammenhang zwischen zweien von vielen tausend Klassen zu erlernen!

Auch wenn man sicherlich die Attributsanzahl durch geschicktere Konstruktion binärwertiger Attributsvektoren noch verringern kann, so wäre der Aufwand bei der Erstellung dieser Beziehungen doch unwirtschaftlich groß. Erschwerend kommt hinzu, dass eine solche Datenbank sicherlich nicht statisch ist, und das Hinzufügen von Teilen oder Klassen eine Aktualisierung beziehungsweise Neuerkennung von Zusammenhängen erfordert.

Zwangsläufig stellt sich die Frage, wie weit man in diesem Modell ohne Membership Queries kommt. Lässt sich die Fallbasis auch auf andere Weise in ausreichendem Maß konsistenzerhaltend komprimieren, so dass wir erneut etwa mit Occams Razor die Existenz eines effizienten PAC-Algorithmus nachweisen können? Wäre es nämlich möglich, aus dem Ergebnis unseres einfachen Lernalgorithmus die kleinstmögliche negative Fallbasis zu berechnen, die immer noch konsistent mit den zugrundeliegenden Trainingsdaten ist, so wäre diese offenbar nicht größer als jede wirklich f_t repräsentierende Fallbasis, die ja ebenfalls mit der Trainingsmenge konsistent sein muss.

Mit Satz 23 existiert für jede boolesche Funktion f eine diese repräsentierende Fallbasis der Größe $|DNF_{\min}(f)| \cdot (1 + |CNF_{\min}(f)|)$. Könnten wir also die negative Fallbasis effizient minimieren, so wäre ein Nachweis der PAC-Lernbarkeit analog zu Satz 49 möglich, ohne dabei auf die Verwendung von Membership Queries hinweisen zu müssen.

Dabei würde auch eine ausreichend gute Approximation des Minimums genügen: Nehmen wir einmal an, es existierte ein $\gamma > 0$, so dass man, ausgehend von einer höchstens m Trainingsbeispiele umfassenden negativen Fallbasis, eine kleinste negative Fallbasis in polynomieller Zeit bis auf einen Faktor $m^{1-\gamma}$ konsistenzhaltend approximieren könnte. Sei zu einer beliebigen n -stelligen Zielfunktion f_t aus der betrachteten Konzeptklasse eine obere Schranke für $|DNF_{\min}(f_t)|$ und $|CNF_{\min}(f_t)|$ durch den Wert $p(n)$ eines Polynoms p gegeben.

Ergänzen wir nun den einfachen Algorithmus aus Abschnitt 4.2 um eine abschließende, die Konsistenz mit den Trainingsdaten erhaltende “Kompression” der negativen Fallbasis durch einen Approximationsalgorithmus, der die entsprechende Güte besitzt. Die positive Fallbasis wird dabei nicht verändert und besitzt mit Satz 36 maximal $|DNF_{\min}(f_t)| \leq p(n)$ Fälle. Die kleinste negative Fallbasis, also das gesuchte Optimum, kann nach Satz 23 höchstens von der Größe $p(n)^2$ sein. Für den neuen Algorithmus kommen dann unter Beachtung des obigen Approximationsfaktors nicht mehr als

$$|\mathcal{H}_{n,m}| \leq 2^{p(n)(1+p(n))} m^{1-\gamma}$$

Hypothesen in Frage. Mit Occams Razor (Satz 8) ist damit für die geforderte PAC-Eigenschaft eine Anzahl von

$$m > 1/\varepsilon \cdot (p(n)(1+p(n)) m^{1-\gamma} + \log(1/\delta)) \quad (1)$$

Trainingsbeispielen hinreichend. Wir können die rechte Seite nach oben durch

$$\frac{m^{1-\gamma}}{\varepsilon} \cdot (p(n)(1+p(n)) + \log(1/\delta))$$

abschätzen. Damit erfüllt m auch (1), wenn gilt:

$$m > \left(1/\varepsilon \cdot (p(n)(1+p(n)) + \log(1/\delta)) \right)^{1/\gamma}.$$

Wird das Polynom $1/\varepsilon \cdot (p(n)(1+p(n)) + \log(1/\delta))$ mit dem festen Exponenten $1/\gamma$ potenziert, ist es immer noch ein Polynom. Auf diese Weise könnten wir folglich nachweisen, dass jede polynomiell repräsentierbare Konzeptklasse auch ohne Membership Queries PAC-lernbar ist, wenn die Existenz eines entsprechenden Approximationsalgorithmus gezeigt werden könnte.

Dieser Umstand stellte die Motivation für die Untersuchungen in diesem Abschnitt dar. Wir werden zeigen, dass ein solcher Approximationsalgorithmus *nicht* existiert.

Dazu wollen wir zunächst das Problem, zu einem gegebenen Sample die kleinstmögliche dazu konsistente Fallbasis im satohschen Modell zu finden, genauer analysieren. Da es sich dabei offenbar um ein Optimierungsproblem handelt, werden wir diesen Begriff jetzt formal einführen.

5.1 Optimierungsprobleme

Unser Problem der Minimierung einer Fallmenge fällt in die Klasse der Optimierungsprobleme. Für diese werden wir jetzt eine formale Definition einführen. Dabei übersetzen wir die sogenannten “feasible solutions” ([BC94]), also diejenigen Lösungen eines Optimierungsproblems, die alle Kriterien außer dem Optimalitätskriterium erfüllen, hier durch **geeignete Lösungen**. Der Begriff der **möglichen Lösungen** hingegen ist schwächer zu sehen und kann je nach Problem angemessen definiert werden. Die möglichen Lösungen müssen jedoch stets die geeigneten umfassen.

Definition 50 [BC94]

Ein **Optimierungsproblem** ist ein Tupel $(I, S, \pi, m, \text{GOAL})$, wobei gilt:

- (i) I ist die Menge aller möglichen Eingaben.
- (ii) S ist eine Funktion, die jede Eingabe x aus I auf eine nicht-leere Menge möglicher Lösungen zu x abbildet.
- (iii) π ist ein Prädikat, so dass für ein x aus I und ein y aus $S(x)$ genau dann $\pi(x, y) = 1$ gilt, wenn y eine geeignete Lösung zu x ist. Wir nehmen an, dass für jedes x aus I ein solches y in $S(x)$ existiert.
- (iv) m ordnet einer Eingabe x aus I und einer zugehörigen geeigneten Lösung y eine positive ganze Zahl $m(x, y)$ zu, die das **Maß** der Lösung y beschreibt.
- (v) GOAL ist entweder der Minimum- oder Maximum-Operator, also \min oder \max .

Zu einer Eingabe x aus I bezeichnen wir mit $\text{OPT}(x)$ das Maß einer optimalen Lösung:

$$\text{OPT}(x) := \text{GOAL}\{m(x, y) \mid y \in S(x) \wedge \pi(x, y)\}.$$

Der **Approximationsfaktor** $q(x, y)$ einer geeigneten Lösung y zu einer Eingabe x ist definiert als

$$q(x, y) := \frac{m(x, y)}{\text{OPT}(x)}.$$

Falls es für ein Optimierungsproblem einen polynomiell zeitbeschränkten Approximationsalgorithmus α und eine Funktion $h : I \rightarrow \mathbb{R}$ gibt, so dass für für alle x aus I

$$q(x, \alpha(x)) \leq h(x)$$

gilt, so nennen wir dieses Problem ***h*-approximierbar**.

Ausgehend von einem Optimierungsproblem kann man stets ein dazugehöriges Entscheidungsproblem definieren:

Definition 51 Zu einem gegebenen Optimierungsproblem $A = (I_A, S_A, \pi_A, m_A, \text{GOAL}_A)$ definieren wir das **zu A gehörige Entscheidungsproblem A_{dec}** wie folgt:

Für eine Eingabe x aus I_A und eine natürliche Zahl k ist (x, k) genau dann in A_{dec} enthalten, wenn ein y in $S_A(x)$ existiert, so dass gilt:

$$\pi_A(x, y) \wedge \begin{cases} m(x, y) \leq k & \text{falls } \text{GOAL}_A = \min \\ m(x, y) \geq k & \text{falls } \text{GOAL}_A = \max \end{cases} .$$

5.2 Definition der betrachteten Probleme

Zunächst werden wir das von uns betrachtete Problem der Minimierung einer negativen Fallbasis formalisieren. Dabei beschränken wir uns auf den Fall, dass die positive Fallbasis CB_+ lediglich aus einem Fall c_{ok} besteht. Wir werden sehen, dass bereits dieses vereinfachte Szenario zum Beweis der Schwierigkeit unseres Problems ausreicht.

Wir stellen eine gewöhnliche Problemdefinition gewissermaßen als Erläuterung der darauf folgenden, an Definition 50 ausgerichteten Darstellung voran.

Definition 52 Das Problem MINIMUM NEGATIVE CASEBASE (MNCB) definieren wir wie folgt:

Gegeben seien eine natürliche Zahl n , zwei Mengen $M_+, M_- \subseteq \{0, 1\}^n$ und ein ausgezeichneter Fall $c_{ok} \in M_+$, wobei gilt:

$$M_+ \subseteq f_{\{c_{ok}, M_-\}}.$$

Gesucht ist eine Menge $M_-^{min} \subseteq \{0, 1\}^n$ mit den folgenden Eigenschaften:

- M1:** $M_+ \subseteq f_{\langle \{c_{ok}\}, M_-^{min} \rangle}$, das heißt:
 Jeder Fall aus M_+ wird von der neuen Fallbasis korrekt (positiv) klassifiziert.
- M2:** $M_- \subseteq \bar{f}_{\langle \{c_{ok}\}, M_-^{min} \rangle}$, das heißt:
 Jeder Fall aus M_- wird von der neuen Fallbasis korrekt (negativ) klassifiziert.
- M3:** M_-^{min} ist minimal bezüglich $|M_-^{min}|$.

Gemäß Definition 50 ist somit $\text{MNCB} := (I_{\text{MNCB}}, S_{\text{MNCB}}, \pi_{\text{MNCB}}, m_{\text{MNCB}}, \text{min})$ mit

- (i) $I_{\text{MNCB}} := \{(n, c_{ok}, M_+, M_-) \mid n \in \mathbb{N}, M_+, M_- \subseteq \{0, 1\}^n, c_{ok} \in M_+ \text{ und } M_+ \subseteq f_{\langle \{c_{ok}\}, M_- \rangle}\}$,
- (ii) $S_{\text{MNCB}}(n, c_{ok}, M_+, M_-) := \text{Pot}(\{0, 1\}^n)$,

und mit $(n, c_{ok}, M_+, M_-) \in I_{\text{MNCB}}$ sowie $M_-^{min} \in S_{\text{MNCB}}(n, c_{ok}, M_+, M_-)$ definieren wir:

- (iii) $\pi_{\text{MNCB}}((n, c_{ok}, M_+, M_-), M_-^{min}) = 1 \iff \mathbf{M1} \wedge \mathbf{M2}$,
- (iv) $m_{\text{MNCB}}((n, c_{ok}, M_+, M_-), M_-^{min}) := |M_-^{min}|$.

M1 und **M2** sind äquivalent zu der Forderung, dass $f_{\langle \{c_{ok}\}, M_-^{min} \rangle}$ konsistent mit $\langle M_+, M_- \rangle$ sein muss.

Satz 53 Das Problem MNCB_{dec} ist in \mathcal{NP} .

Beweis: Eine nicht-deterministische Turingmaschine rät k Fälle aus $\{0, 1\}^n$ für M_-^{min} . Danach überprüft sie, ob die geratene Fallmenge die Eigenschaften **M1** und **M2** besitzt und akzeptiert im positiven Fall.

Das Raten kostet lediglich $O(k \cdot n)$ Schritte. Um zu überprüfen, ob ein Fall in $f_{\langle \{c_{ok}\}, M_-^{min} \rangle}$ liegt oder nicht, muss sein Abstand zu c_{ok} im schlechtesten Fall mit dem Abstand zu jedem der k Fälle aus M_-^{min} verglichen werden. Die Abstandsrechnung erfolgt durch bitweises XOR und benötigt somit ebenso die Zeit $O(n)$ wie der Vergleich zweier Abstände mit \preceq . Somit kann ein Fall in $O(k \cdot n)$ Schritten von $f_{\langle \{c_{ok}\}, M_-^{min} \rangle}$ klassifiziert werden. Insgesamt benötigt die Turingmaschine folglich maximal

$$O((|M_+| + |M_-|) \cdot k \cdot n)$$

viele Schritte, um $((n, c_{ok}, M_+, M_-), k) \in \text{MNCB}_{dec}$ zu überprüfen und arbeitet damit in polynomieller Zeit.

◇

Vor der folgenden Definition sei daran erinnert, dass V_1, V_2, \dots, V_k genau dann eine **Zerlegung** oder **Partition** einer Menge V ist, falls gilt

- (i) Die V_i sind paarweise disjunkt: $i \neq j \implies V_i \cap V_j = \emptyset$.
- (ii) Die V_i bilden eine **Überdeckung** oder auch **Abdeckung** von V : $\bigcup_{i=1}^k V_i = V$.

Weiterhin verstehen wir unter einer **Clique** eines Graphen einen vollständigen Subgraphen desselben.

Um die Schwierigkeit von MNCB nachzuweisen, werden wir eine Reduktion verwenden. Auf der Suche nach einem Problem, welches sich geeignet auf MNCB reduzieren lässt, stoßen wir auf das Folgende:

Definition 54 [GJ79] *Das Problem MINIMUM CLIQUE PARTITION (MCP) ist wie folgt definiert:*

Gegeben sei ein Graph $G = (V, E)$. Dabei bezeichne V die Menge der Knoten und $E \subseteq V \times V$ die der Kanten von G .

Gesucht ist eine Partition $\{V_1, V_2, \dots, V_k\}$ von V , so dass gilt:

- (i) *Für jedes i aus $\{1, 2, \dots, k\}$ ist V_i eine Clique in G .*
- (ii) *Für V existiert keine Partition aus $k' < k$ Cliquen.*

Gemäß Definition 50 ist somit $\text{MCP} := (I_{\text{MCP}}, S_{\text{MCP}}, \pi_{\text{MCP}}, m_{\text{MCP}}, \text{min})$, mit

- (i) *I_{MCP} ist die Menge aller Graphen $G = (V, E)$,*
- (ii) *$S_{\text{MCP}}(G) := \{\nu \mid \nu \subseteq \text{Pot}(V)\}$,*

und für $G \in I_{\text{MCP}}$ sowie $\nu \in S_{\text{MCP}}(G)$ definieren wir:

- (iii) *$\pi_{\text{MCP}}(G, \nu) = 1 \iff \nu$ ist eine Cliquen-Überdeckung zu G ,*
- (iv) *$m_{\text{MCP}}(G, \nu) := |\nu|$.*

Zu MCP ist das Folgende bekannt:

Fakt 55 *Es gilt:*

1. [GJ79] *Das zu MCP gehörige Entscheidungsproblem MCP_{dec} ist \mathcal{NP} -hart.*
2. [FK98] *Unter der Voraussetzung $\mathcal{ZPP} \neq \mathcal{NP}$ existiert kein $\gamma > 0$, so dass MCP $|V|^{1-\gamma}$ -approximierbar ist.*

Dabei beeinflusst die Restriktion, dass die Mengen V_i paarweise disjunkt sein müssen, weder die \mathcal{NP} -Härte noch die Approximierbarkeit des Problems:

Lemma 56 *Es gilt:*

- (a) *Jede Cliques-Partition für V ist auch eine Cliques-Überdeckung für V .*
- (b) *Aus einer Cliques-Überdeckung für V kann in Polynomialzeit eine Cliques-Partition für V konstruiert werden, die nicht aus mehr Mengen besteht als die zugrunde liegende Überdeckung.*

Beweis: Aussage (a) ist klar.

Zu Aussage (b): Wir konstruieren eine Partition aus einer Überdeckung, indem wir die gegebenen Mengen sukzessive in die Partition übernehmen, wobei wir jeweils alle Elemente entfernen, die bereits von einer vorher betrachteten Menge abgedeckt wurden. Entsteht dabei eine leere Menge, so wird diese nicht in die Partition übernommen.

Dies gelingt effizient, denn man könnte beispielsweise die schon abgedeckten Elemente in einer Liste halten, die dann für jede Menge einmal durchlaufen werden muss. Des Weiteren bilden die nach dem Entfernen einiger Knoten aus einer Clique verbleibenden Knoten offenbar weiterhin eine Clique. Damit haben wir pro Clique aus der Überdeckung maximal eine Clique für die Partition konstruiert.

◇

Folglich ist die Schwierigkeit der Suche einer Cliques-Überdeckung zumindest bezüglich polynomieller Zeitbeschränkung genauso groß wie die einer Cliques-Partition. Unser Ziel ist es, mit Hilfe einer Reduktion von MCP auf MNCB sowohl die \mathcal{NP} -Härte als auch die Approximierbarkeits-Eigenschaften von MCP auf MNCB zu übertragen.

Für dieses Unterfangen benötigen wir einen geeigneten Reduktionsbegriff.

5.3 Reduktion von Optimierungsproblemen

Gesucht ist eine Reduktion mit der Eigenschaft, dass aus der Reduzierbarkeit eines Optimierungsproblems A auf ein anderes B und der Existenz eines Approximationsalgorithmus für B auf die eines solchen für A geschlossen werden kann.

Dabei kann der vorhandene Algorithmus α_B für B beim Bestimmen einer Approximation einer Lösung in A gewissermaßen als Unterprogramm verwendet werden. Die Schwierigkeit besteht darin, auf effiziente Weise einerseits eine beliebige Eingabeinstanz x_A für A in den Eingaberaum I_B für B , und andererseits die von α_B generierte Approximation in B auf eine Lösung in A derart abzubilden, dass diese tatsächlich eine geeignete Lösung zu x_A ist (Abbildung 5.3). Weiterhin sollte die Güte der Approximation in einem gewissen Maße erhalten bleiben.

Ein solcher Reduktionsbegriff steht uns mit der \mathcal{APX} -Reduktion zur Verfügung.

Dabei ist \mathcal{APX} die Klasse derjenigen Optimierungsprobleme, deren Entscheidungsprobleme in \mathcal{NP} sind, und die in Polynomialzeit bis auf einen festen Faktor ε approximiert werden können [BC94]. Wir werden jedoch mit Hilfe dieses Reduktionsbegriffs Aussagen für Probleme ableiten, deren Approximationsfaktoren keine Konstanten sind.

Da es sich bei den in Abschnitt 5.2 vorgestellten Problemen ausschließlich um Minimierungsprobleme handelt, werden wir in der Folge ausschließlich solche betrachten. Die Definitionen und Aussagen dieses Abschnitts können jedoch einfach auf Maximierungsprobleme übertragen werden.

Definition 57 [BC94] *Ein Minimierungsproblem $A = (I_A, S_A, \pi_A, m_A, \min)$ heißt \mathcal{APX} -reduzierbar auf ein zweites Minimierungsproblem $B = (I_B, S_B, \pi_B, m_B, \min)$, falls zwei Funktionen r und t existieren, so dass gilt:*

- (1) r und t sind in polynomieller Zeit berechenbar.
- (2) Für jedes x_A aus I_A liegt $r(x_A)$ in I_B . Mit anderen Worten, r bildet Eingaben für A auf Eingaben für B ab.
- (3) Für jedes x_A aus I_A und jedes y_B aus $S_B(r(x_A))$ gilt $t(x_A, y_B) \in S_A(x_A)$ und

$$\pi_B(r(x_A), y_B) \implies \pi_A(x_A, t(x_A, y_B)).$$

Mit anderen Worten, t bildet geeignete Lösungen zu $r(x_A)$ in B auf geeignete Lösungen zu x_A in A ab.

- (4) Für jedes $q_0 \geq 1$ existiert ein $q'_0 \geq 1$, so dass für jedes $x_A \in I_A$ und jedes $y_B \in S_B(r(x_A))$ mit $\pi_B(r(x_A), y_B) = 1$ gilt:

$$q_B(r(x_A), y_B) \leq q_0 \implies q_A(x_A, t(x_A, y_B)) \leq q'_0.$$

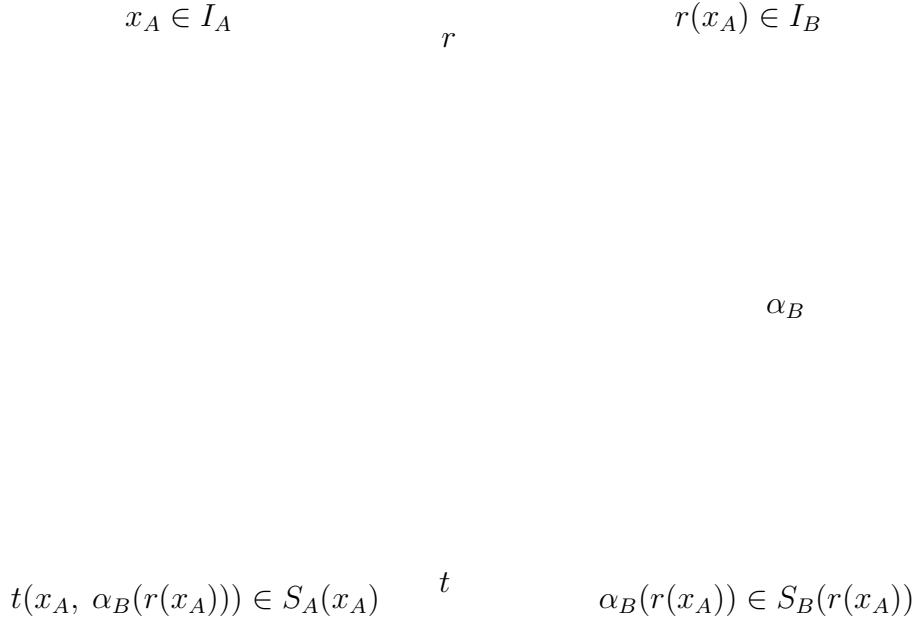


Abbildung 6 Schema einer \mathcal{APX} -Reduktion von A auf B

Wir schreiben dann $\mathcal{A} \leq_{\mathcal{APX}} \mathcal{B}$.

Offenbar folgt mit $\mathcal{A} \leq_{\mathcal{APX}} \mathcal{B}$ aus der q_0 -Approximierbarkeit von \mathcal{B} die q'_0 -Approximierbarkeit von \mathcal{A} . Wir benötigen hier jedoch die folgende Negativaussage:

Lemma 58 Gegeben seien zwei Optimierungsprobleme A und B sowie eine Funktion $h_A : I_A \rightarrow \mathbb{R}$. Es seien die folgenden Voraussetzungen erfüllt:

- (a) $A \leq_{\mathcal{APX}} B$, unter Verwendung von Definition 57 entsprechenden Funktionen r und t .
- (b) Für jedes x_A aus I_A und y_B aus $S_B(r(x_A))$ gilt:

$$q_A(x_A, t(x_A, y_B)) \leq q_B(r(x_A), y_B).$$

Dann gilt für jede Funktion $h_B : I_B \rightarrow \mathbb{R}$, die für alle x_B aus $r(I_A)$ die Ungleichung

$$h_B(x_B) \leq \min \{h_A(x_A) \mid x_A \in I_A \wedge r(x_A) = x_B\}$$

erfüllt, die Aussage:

$$A \text{ ist nicht } h_A\text{-approximierbar} \implies B \text{ ist nicht } h_B\text{-approximierbar.}$$

Beweis: Wir werden zeigen, dass aus der h_B -Approximierbarkeit von B sofort folgen würde, dass A h_A -approximierbar ist, denn man könnte die Eingabeinstanz mit Hilfe von r nach I_B abbilden, eine Lösung in B geeignet approximieren und diese mit Hilfe von t nach A rücktransformieren. Da wir in (b) gefordert haben, dass der relative Fehler bei der Rücktransformation nicht wachsen darf, ist die Lösung für A mindestens so “gut” wie die für B approximierte.

Wir nehmen an, B sei h_B -approximierbar für ein h_B gemäß obiger Bedingung. Diese bedeutet, dass h_b für jedes x_B aus B , welches ein Urbild x_A bezüglich r in A besitzt, den besten Approximationsfaktor h_A aller Urbilder übernimmt.

Da B nach Voraussetzung h_B -approximierbar ist, existiert ein Approximationsalgorithmus α_B , so dass für alle x_B aus I_B

$$q_B(x_B, \alpha_B(x_B)) \leq h_B(x_B)$$

ist. Damit folgt für ein beliebiges x_A aus I_A und sein Bild $r(x_A)$ in $r(I_A) \subseteq I_B$:

$$q_B(r(x_A), \alpha_B(r(x_A))) \leq h_B(r(x_A)).$$

Mit obiger Bedingung für h_B gilt $h_B(r(x_A)) \leq h_A(x_A)$, also insgesamt

$$q_B(r(x_A), \alpha_B(r(x_A))) \leq h_A(x_A). \quad (2)$$

Die Approximation $\alpha_B(r(x_A))$ ist eine geeignete Lösung zu $r(x_A)$ in B , folglich muss auch $\alpha_B(r(x_A)) \in S_B(r(x_A))$ gelten. Somit folgt aus (b) mit $y_B := \alpha_B(r(x_A))$ die Aussage

$$q_A\left(x_A, t(x_A, \alpha_B(r(x_A)))\right) \leq q_B\left(r(x_A), \alpha_B(r(x_A))\right). \quad (3)$$

Mit (2) und (3) erhalten wir insgesamt:

$$q_A\left(x_A, t(x_A, \alpha_B(r(x_A)))\right) \leq h_A(x_A).$$

Da r und t nach Definition 57 außerdem in polynomieller Zeit berechenbar sind, ist α_A mit

$$\alpha_A(x_A) := t(x_A, \alpha_B(r(x_A)))$$

ein h_A -Approximationsalgorithmus für A .

◇

5.4 Komplexität der Fallbasenminimierung

Wir werden nicht nur zeigen, dass MNCB_{dec} \mathcal{NP} -vollständig ist, sondern weiterhin mit Hilfe von Lemma 58 nachweisen, dass sich MNCB nicht geeignet approximieren lässt.

Zu diesem Zweck werden wir unter Anderem eine \mathcal{APX} -Reduktion von MCP auf MNCB vorstellen. Dabei dürfen wir bei der Betrachtung des Problems MCP nach Lemma 56 die Forderung nach Disjunktheit der Knotenmengen vernachlässigen. Wir werden also im Folgenden stets von einer Überdeckung anstelle einer Partition sprechen.

Wir konstruieren zwei Reduktionsfunktionen r und t . Gemäß Definition 57 soll r dabei Eingaben für MCP auf Eingaben für MNCB , und t geeignete Lösungen zu MNCB auf solche in MCP abbilden (vergleiche auch Abbildung 5.3).

Die Verwandtschaft beider Probleme liegt zunächst darin, dass es jeweils eine Menge von Objekten abzudecken (beziehungsweise zu überdecken) gilt. Bei MCP werden die Knoten des Graphen dadurch abgedeckt, dass sie in eine Menge der Lösungsabdeckung aufgenommen werden, während wir bei MNCB die Fälle aus M_- bezüglich des Positivfalles c_{ok} abdecken müssen. Es liegt also nahe, Knoten mit Hilfe von r auf Fälle in M_- abzubilden. Weiterhin sind die abdeckenden Elemente einerseits Knotenmengen, andererseits Fälle aus M^{min} . Diese dürfen aber jeweils nicht frei gewählt werden: Die Knotenmengen in MCP müssen Cliques des Graphen sein, und in MNCB dürfen durch die Fälle aus M_-^{min} keine Positivfälle aus M_+ bezüglich c_{ok} abgedeckt werden. Wir werden daher diese Positivfälle aus der Kantenmenge E konstruieren.

Definition 59 *Wir definieren eine Funktion r , die einem Graphen $G = (V, E)$ ein Tupel (n, c_{ok}, M_-, M_+) mit $n \in \mathbb{N}$, $c_{ok} \in \{0, 1\}^n$, $M_-, M_+ \subseteq \{0, 1\}^n$ wie folgt zuordnet:*

1. $n := |V| + 1$
2. $c_{ok} := 1^n$
3. $M_- := \{\chi_n(\{i\}) \mid i \in V\}$
4. $M_+ := \{c_{ok}, \chi_n(\{n\})\} \cup \{\chi_n(\{i, j\}) \mid (i, j) \in \overline{E}\}^1$

Zunächst bemerken wir, dass für jeden Knoten aus V ein Fall in M_- erzeugt wird, es gilt also $|M_-| = |V|$.

¹Dabei ist das Komplement \overline{E} einer Kantenmenge E bekanntlich definiert als die Menge aller Kanten zwischen *verschiedenen* Knoten, die nicht in E vorkommen.

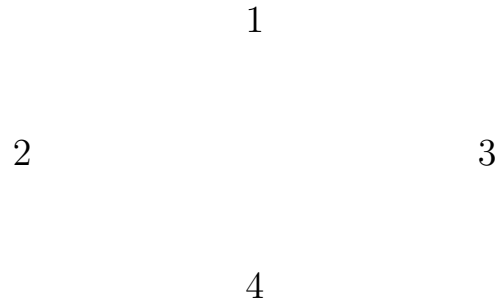


Abbildung 7 Beispielgraph G zur Demonstration der Transformation mit r in Abbildung 5.4.

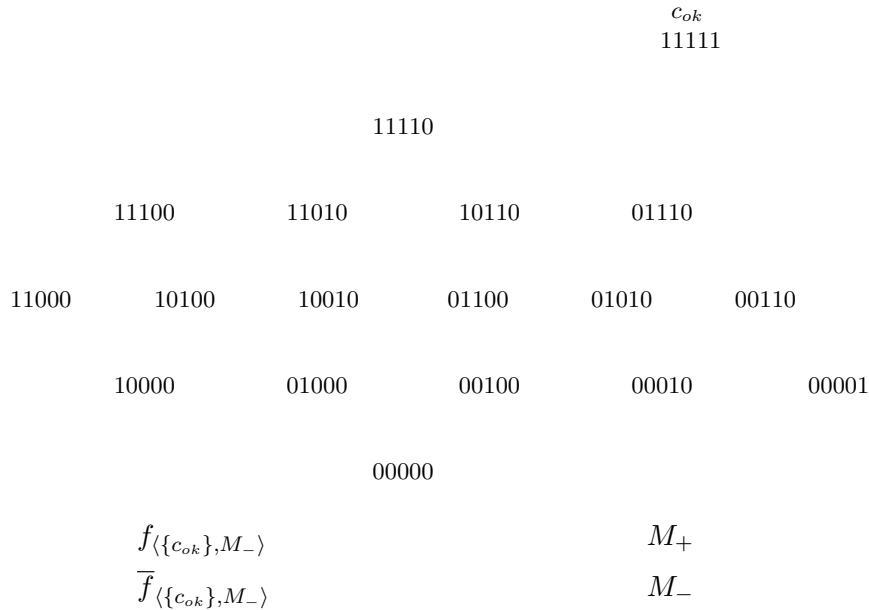


Abbildung 8 Die Elemente der aus dem Graphen G aus Abbildung 5.4 konstruierten Eingabe von MNCB, eingezeichnet in den entsprechenden Verband. M_-^{min} darf sich ausschließlich aus dunkel unterlegten Fälle zusammensetzen, da nur dann die Eigenschaft **M1** erfüllt ist. Um **M2** auch zu erfüllen, müssen die ausgewählten Fälle alle in M_- enthaltenen abdecken. Ein entsprechendes M_-^{min} wäre beispielsweise $\{11010, 00110\}$.

Die Abbildung der zu überdeckenden Elemente wird in Schritt 3 durch den Isomorphismus χ realisiert. Dabei haben wir c_{ok} am einen und die abzudeckenden Fälle aus M_- am anderen “Ende” des Verbandes positioniert (Abbildung 5.4)². Für je zwei Knoten i und j , die in G *nicht* direkt durch eine Kante verbunden sind (und damit auch in keiner Clique gemeinsam vorkommen können), fügen wir in Schritt 4 den Fall $\chi_n(\{i, j\})$ zu M_+ hinzu, der verhindert, dass die entsprechenden Fälle von einem Fall aus M_-^{min} gemeinsam abgedeckt werden. Damit erfüllt eine Menge M_- nur dann **M1**, wenn die darin enthaltenen Fälle mit χ_n^{-1} auf Cliquen in G abgebildet werden, und nur dann **M2**, wenn diese Cliquen sich zu G vereinigen und damit eine Überdeckung des Graphen darstellen. Dies wollen wir gleich formal beweisen.

Es fällt auf, dass die Dimension des Verbandes von uns eins größer gewählt wurde als die Menge der Knoten in G . Das ist notwendig, da es sonst unmöglich wäre, eine Cliquen-Überdeckung aus nur einer Clique zu konstruieren. Dies ist dann interessant, wenn G selbst eine Clique ist und die in Schritt 3 konstruierten Fälle aus M_- mit dem einzelnen Fall $1^{|V|}0$ abgedeckt werden können. Weil diese zusätzliche Dimension zu keinem Knoten korrespondiert, muss sie selbst nicht abgedeckt werden. Dass ein Negativfall in diesen künstlich erweiterten Bereich fällt, verhindern wir durch Hinzufügen des Falls $0^{|V|}1$ zu M_+ .

Wir zeigen jetzt, dass r Eingaben für MCP auch tatsächlich auf Eingaben für MNCB abbildet, und damit Voraussetzung (2) aus Definition 57 erfüllt.

Lemma 60 *Es gilt:*

$$G \in I_{\text{MCP}} \implies r(G) \in I_{\text{MNCB}}.$$

Beweis: Es ist zu zeigen, dass die mit r erzeugte Eingabe (n, c_{ok}, M_-, M_+) den formalen Voraussetzungen für MNCB genügt. Dabei ist nach Konstruktion klar, dass c_{ok} in M_+ liegt. Zu zeigen bleibt

$$M_+ \subseteq f_{\{c_{ok}, M_-\}}.$$

Wir erinnern uns, dass ein Fall c genau dann in $f_{(CB_+, CB_-)}$ enthalten ist, wenn gilt:

$$\exists c_{ok} \in CB_+ : \forall c_{ng} \in CB_- : d(c_{ok}, c_{ng}) \not\leq d(c_{ok}, c).$$

In unserem Fall enthält die positive Fallbasis lediglich einen Fall, nämlich $c_{ok} = 1^n$. Damit entfällt der Existenzquantor. Des Weiteren klassifiziert sich c_{ok} nach Bemerkung 16(a) selbst positiv. Wenn \bar{c} das bitweise Komplement von c bezeichnet, gilt außerdem mit $c_{ok} = 1^n$:

$$d(c_{ok}, c) = \bar{c}. \tag{4}$$

²Die Wahl von c_{ok} ist dabei offenbar willkürlich. Jeder Fall aus M_- muss jedoch zu c_{ok} genau $(n - 1)$ Bits Abstand haben. Der Übersichtlichkeit halber verwenden wir $c_{ok} = 1^n$, denn auf diese Weise definieren sich die Fälle aus M_- zu Einheitsvektoren, und der Isomorphismus χ_n bildet Knotenmengen auf Fälle anschaulich ab.

Schließlich bemerken wir noch:

$$\bar{c}_1 \preceq \bar{c}_2 \iff c_2 \preceq c_1. \quad (5)$$

Somit bleibt für jeden Fall c aus $\{\chi_n(\{i, j\}) \mid (i, j) \in \bar{E}\}$ zu zeigen, dass für kein c_{ng} aus M_- die Aussage $c \preceq c_{ng}$ gilt.

Betrachten wir nun die Fälle aus M_- , so korrespondiert jeder zu genau einem Knoten aus V , und enthält nach Definition von χ_n somit auch nur genau eine Eins. Jeder Fall c aus $\{\chi_n(\{i, j\}) \mid (i, j) \in \bar{E}\}$ hingegen wurde als Repräsentant einer fehlenden Kante konstruiert, und enthält, da $i \neq j$ vorausgesetzt wurde, genau zwei Einsen. Somit kann niemals $c \preceq c_{ng}$ gelten, was zu zeigen war. Somit erzeugt unsere Reduktion tatsächlich eine Untermenge der gültigen Eingaben für MNCB.

◇

Definition 61 Die Funktion t ordnet einem Graphen G und einer Menge $M_-^{min} \subseteq \{0, 1\}^n$ eine Kollektion ν von Teilmengen von $\{1, 2, \dots, n\}$ in folgender Weise zu:

$$\nu := \{\chi_n^{-1}(c) \mid c \in M_-^{min}\}.$$

Um Voraussetzung 3 aus der \mathcal{APX} -Reduzierbarkeits-Definition (Definition 57) zu erfüllen, müssen wir zeigen, dass t für jeden Graphen G und jedes M_-^{min} aus $S_{\text{MNCB}}(r(G))$ eine Ausgabe $t(G, M_-^{min})$ konstruiert, so dass die Folgerung

$$\pi_{\text{MNCB}}(r(G), M_-^{min}) \implies \pi_{\text{MCP}}(G, t(G, M_-^{min}))$$

gilt. Mit anderen Worten:

Lemma 62 Sei $G = (V, E)$ ein Graph und M_-^{min} eine geeignete Lösung zur Eingabe $r(G)$ in MNCB. Dann ist $t(G, M_-^{min})$ eine geeignete Lösung zu G in MCP.

Beweis: Sei $M_-^{min} = \{c_1, c_2, \dots, c_k\}$ eine geeignete negative Fallbasis zu $r(G) := (n, c_{ok}, M_-, M_+)$ und erfülle somit **M1** und **M2**. Um zu zeigen, dass

$$t(G, M_-^{min}) = \{V_1, V_2, \dots, V_k\} = \{\chi_n(c_1), \chi_n(c_2), \dots, \chi_n(c_k)\}$$

eine geeignete Lösung zu G in MCP darstellt, müssen wir nachweisen, dass es sich bei $\{V_1, V_2, \dots, V_k\}$ um eine Cliques-Überdeckung zu G handelt.

Dazu ist zu zeigen:

- (a) Jede Menge V_l bildet eine Clique in G : $\forall i, j \in V_l : i \neq j \implies (i, j) \in E$.
- (b) V_1, V_2, \dots, V_k bildet eine Überdeckung zu V : $\bigcup_{l=1}^k V_l = V$.

Zu (a):

Nach Konstruktionsschritt 4 gilt für zwei Knoten i und j aus V mit $i \neq j$ die Aussage $\chi_n(\{i, j\}) \in M_+ \iff (i, j) \in \overline{E}$ und damit insbesondere

$$\chi_n(\{i, j\}) \notin M_+ \implies (i, j) \in E. \quad (6)$$

Seien i und j nun zwei Knoten in einer der Mengen V_l . Mit Aussage (6) genügt es damit zu zeigen, dass der Fall $\chi_n(\{i, j\})$ nicht in M_+ liegt. Laut Konstruktion ist V_l aus einem Fall $c_l \in M_-^{min}$ entstanden, und damit muss $c_l[i] = c_l[j] = 1$ gelten. Wir nehmen an, es gilt

$$\chi_n(\{i, j\}) \in M_+.$$

M1 besagt $M_+ \subseteq f_{\langle \{c_{ok}\}, M_-^{min} \rangle}$ und es folgt

$$\chi_n(\{i, j\}) \in f_{\langle \{c_{ok}\}, M_-^{min} \rangle}.$$

Gemäß Bemerkung 16(b) kann die Menge der positiv klassifizierten Fälle durch das Entfernen von Fällen aus der negativen Fallbasis nur wachsen, es ist also

$$\chi_n(\{i, j\}) \in f_{\langle \{c_{ok}\}, \{c_l\} \rangle}.$$

Die Anwendung von Definition 15 ergibt $c_l \notin U(c_{ok}, \chi_n(\{i, j\}))$. Das bedeutet wiederum mit Definition 12, dass

$$d(c_{ok}, c_l) \not\leq d(c_{ok}, \chi_n(\{i, j\}))$$

ist und mit (4) folgt schließlich $\bar{c}_l \not\leq \overline{\chi_n(\{i, j\})}$, was nach (5) gleichbedeutend ist mit

$$\chi_n(\{i, j\}) \not\leq c_l.$$

Nun besteht $\chi_n(\{i, j\})$ ausschließlich aus Nullen, wenn man von den Stellen i und j absieht. Diese Stellen sind aber laut Voraussetzung in c_l ebenfalls 1. Folglich muss $\chi_n(\{i, j\}) \leq c_l$ gelten, was der letzten gefolgerten Aussage widerspricht. Offenbar war unsere Prämisse falsch, und es ist

$$\chi_n(\{i, j\}) \notin M_+.$$

Mit Aussage (6) erhalten wir $(i, j) \in E$, was zu zeigen war.

Jede konstruierte Menge V_l bildet also tatsächlich eine Clique in G .

Zu (b):

Nach Voraussetzung erfüllt das konstruierte M_-^{min} die Eigenschaft **M2**:

$$M_- \subseteq \bar{f}_{\langle \{c_{ok}\}, M_-^{min} \rangle}.$$

Damit gilt für jedes $c_i := \chi_n^{-1}(\{i\})$ aus M_- nach Definition 15, dass es ein $c'_i \in M_-^{min}$ gibt, für das

$$c'_i \in U(c_{ok}, c_i)$$

gilt. Da sowohl $c_i[i] = 1$ als auch $c_{ok}[i] = 1$ gilt, müssen auch alle Fälle im Unterverband zwischen diesen Vektoren an i -ter Stelle eine 1 besitzen, also auch c'_i . Einerseits hat die Reduktionsfunktion r zu jedem i aus $V = \{1, \dots, n-1\}$ einen Fall c_i in M_- eingefügt, zu dem ein solches c'_i in M_-^{min} existiert. Andererseits kann wegen $\chi(\{n\}) = 0^{|V|}1 \in M_+$ unter Beachtung von **M1** kein Fall aus M_-^{min} eine 1 an n -ter Stelle besitzen. Daher gilt:

$$\bigvee_{c'_i \in M_-^{min}} c'_i = 1^{|V|}0.$$

Nun wenden wir auf beiden Seiten χ_n^{-1} an:

$$\chi_n^{-1} \left(\bigvee_{c'_i \in M_-^{min}} c'_i \right) = \chi_n^{-1} (1^{|V|}0).$$

Mit $\chi_n^{-1}(1^{|V|}0) = V$ und Bemerkung 3, Teil (1) folgt

$$\bigcup_{c'_i \in M_-^{min}} \chi_n^{-1}(c'_i) = V,$$

und da jede Knotenmenge V_i aus der Anwendung von χ_n auf einen der k Fälle aus M_-^{min} konstruiert wurde, erhalten wir damit die gewünschte Aussage:

$$\bigcup_{i=1}^k V_i = V.$$

Mit (a) und (b) haben wir gezeigt, dass $\{V_1, V_2, \dots, V_k\}$ tatsächlich eine Cliques-Überdeckung zu G darstellt.

◇

Bei einer \mathcal{APX} -Reduktion müssen beide Reduktions-Funktionen effizient berechenbar sein. Dies ist hier gegeben:

Lemma 63 *Die Funktionen r und t sind in polynomieller Zeit berechenbar.*

Beweis: Betrachten wir zunächst die Funktion r , also die Konstruktion einer Eingabe (n, c_{ok}, M_+, M_-) für MNCB aus einem Graphen $G = (V, E)$. Von den vier Konstruktionsschritten aus Definition 59 sind die ersten beiden sicherlich als problemlos anzusehen. Die Funktion χ_n ist für jede beliebige, also maximal n -elementige Menge in der Zeit $O(n^2)$ berechenbar. Sie wird zur Konstruktion von M_- in 3. $|V| = (n - 1)$ -mal verwendet, somit benötigt dieser Schritt ebenfalls polynomielle Zeit in $|V|$. In Schritt 4 wird χ_n auf jede Kante aus \overline{E} angewandt, also höchstens $|V|^2$ mal. Ob eine bestimmte Kante in \overline{E} ist, kann direkt aus der Darstellung von E abgelesen werden. Damit ist r insgesamt in polynomieller Zeit berechenbar.

Um einen Funktionswert von t aus den Fällen einer Menge M_-^{min} zu bestimmen, wird lediglich die Umkehrfunktion von χ_n genau $|M_-^{min}|$ mal aufgerufen. Deren Auswertung benötigt wiederum $O(n^2)$ Schritte. Somit ist auch die Funktion r in polynomieller Zeit berechenbar.

◇

Um die \mathcal{APX} -Reduzierbarkeit von MCP auf MNCB zu zeigen, wollten wir nachweisen, dass die Funktionen t und r den Bedingungen (1) bis (4) aus Definition 57 genügen. Unser eigentliches Ziel ist es jedoch, die stärkeren Voraussetzungen für Lemma 58 zu erfüllen, um die bekannten Approximierbarkeits-Eigenschaften von MCP auf MNCB entsprechend übertragen zu können. Wir bemerken hier, dass Bedingung (b) aus Lemma 58 den Spezialfall von Voraussetzung (4) aus Definition 57 darstellt, bei dem q'_0 stets gleich q_0 zu wählen ist. Folglich genügt es zu zeigen, dass eben diese Bedingung (b) erfüllt ist. Dafür zeigen wir, dass die optimalen Lösungen zu G und $r(G)$ bezüglich des jeweiligen Maßes gleich groß sind:

Lemma 64 *Sei G ein Graph. Dann gilt:*

$$\text{OPT}_{\text{MCP}}(G) = \text{OPT}_{\text{MNCB}}(r(G)).$$

Beweis: Wir konnten in Lemma 62 bereits zeigen, dass zu jeder geeigneten Lösung in MNCB eine geeignete Lösung in MCP existiert (wie haben diese sogar mit t konstruiert), die dasselbe Maß besitzt. Können wir weiterhin nachweisen, dass es umgekehrt zu jeder geeigneten Lösung in MCP eine solche Lösung gleichen Maßes in MNCB gibt, so müssen auch die optimalen Lösungen in beiden Problemen zu den Eingaben G beziehungsweise $r(G)$ gleich groß sein.

Wir werden wieder konstruktiv vorgehen. Die Umkehrung zu t lässt sich direkt aus der Beschreibung von t ableiten:

$$t^{-1}(\{V_1, V_2, \dots, V_k\}) := \bigcup_{i=1}^k \chi_n(V_i).$$

Zuerst stellen wir fest, dass die Kardinalität aufgrund der Bijektivität von χ_n erhalten bleibt.

Wir zeigen: Wenn $\{V_1, V_2, \dots, V_k\}$ eine geeignete Lösung zu G ist, dann ist

$$M_-^{min} := \bigcup_{i=1}^k \chi_n(V_i)$$

eine geeignete Lösung zu $r(G)$.

Sei also $\nu := \{V_1, V_2, \dots, V_k\}$ eine Cliques-Überdeckung zu G .

Zu zeigen ist **M1**: $M_+ \subseteq f_{\langle \{c_{ok}\}, M_-^{min} \rangle}$.

Jeder Fall $\chi_n(\{i, j\})$ aus M_+ ist nach Definition 59 als Repräsentant einer fehlenden Kante zwischen den Knoten i und j konstruiert worden, es ist also

$$(i, j) \notin E.$$

Folglich kann es in G keine Clique geben, die sowohl i als auch j enthält. Da $\nu = \{V_1, V_2, \dots, V_k\}$ nach Voraussetzung eine Cliques-Überdeckung zu G darstellt, können i und j damit auch nicht gemeinsam in einer Knotenmenge V_l vorkommen. Nach Definition von t^{-1} bedeutet das, dass für jeden Fall c aus der konstruierten Menge $M_-^{min} = t^{-1}(\nu)$ gilt:

$$c[i] = 0 \vee c[j] = 0.$$

Folglich kann kein Fall c aus M_-^{min} den Fall $\chi_n(\{i, j\})$ aus M_+ bezüglich $c_{ok} = 1^n$ abdecken. Das heißt, für alle $c \in M_-^{min}$ ist

$$d(c_{ok}, c) \not\leq d(c_{ok}, \chi_n(\{i, j\})),$$

woraus mit den Definitionen 12 und 15 die gewünschte Aussage folgt:

$$\chi_n(\{i, j\}) \in f_{\langle \{c_{ok}\}, M_-^{min} \rangle}.$$

Zu zeigen ist **M2**: $M_- \subseteq \bar{f}_{\langle \{c_{ok}\}, M_-^{min} \rangle}$.

$\{V_1, V_2, \dots, V_k\}$ bildet nach Voraussetzung eine Überdeckung für V . Damit gibt es für jedes i aus V ein $j \in \{1, \dots, k\}$ so dass i in V_j ist. Folglich existiert auch Fall $c_j := \chi_n(V_j)$ in M_-^{min} , welcher an i -ter Stelle eine Eins besitzt. Damit ist offenbar für den Fall $\chi(\{i\})$ aus M_-

$$d(1^n, c_j) \leq d(1^n, \chi(\{i\})).$$

Mit anderen Worten, jeder Fall $\chi(\{i\})$ aus M_- wird von einem solchen c_j aus M_-^{min} bezüglich $c_{ok} = 1^n$ abgedeckt und damit gilt

$$\chi_n(\{i\}) \in \bar{f}_{\langle \{c_{ok}\}, M_-^{min} \rangle},$$

was zu beweisen war.

Wir haben damit gezeigt, dass t^{-1} eine geeignete Lösung zu $r(G)$ in MNCB auf eine geeignete Lösung zu G in MCP abbildet, wobei die Kardinalität der Lösung erhalten bleibt. In Lemma 62 hatten wir nachgewiesen, dass t dasselbe in die andere Richtung bewerkstelligt. Folglich müssen die jeweils optimalen Lösungen ebenfalls dieselbe Kardinalität besitzen.

◇

Lemma 65 *Der Approximationsfaktor der mit t konstruierten Cliques-Überdeckung ist gleich demjenigen der Fallmenge M_-^{min} , aus welcher diese konstruiert wurde. Für jedes G aus I_{MCP} und M_-^{min} aus $S_{MNCB}(r(G))$ gilt also:*

$$q_{MCP}(G, t(G, M_-^{min})) = q_{MNCB}(r(G), M_-^{min}).$$

Beweis: Sei G aus I_{MCP} und M_-^{min} aus $S_{MNCB}(r(G))$. Da wir mit t für jeden Fall genau eine Knotenmenge konstruieren, ist die Ordnung k der erzeugten Partition gleich der Kardinalität der Menge M_-^{min} aus der sie entstanden ist:

$$|t(M_-^{min})| = |M_-^{min}|.$$

Weiterhin sagt Lemma 64 aus, dass auch die optimalen Lösungen zu G und $r(G)$ dasselbe Maß besitzen:

$$\text{OPT}_{MCP}(G) = \text{OPT}_{MNCB}(r(G)).$$

Damit gilt insgesamt:

$$q_{MCP}(G, t(M_-^{min})) = \frac{|t(M_-^{min})|}{\text{OPT}_{MCP}(G)} = \frac{|M_-^{min}|}{\text{OPT}_{MNCB}(r(G))} = q_{MNCB}(r(G), M_-^{min}).$$

◇

Jetzt verfügen wir über die notwendigen Hilfsmittel, um die Resultate dieses Kapitels herzuleiten.

Satz 66 *Das Problem MINIMUM NEGATIVE CASEBASE_{dec} ist \mathcal{NP} -vollständig.*

Beweis: Wir zeigen $\text{MCP}_{dec} \leq^P \text{MNCB}_{dec}$.

Als Reduktionsfunktion bietet sich unter Beachtung unserer vorangehenden Überlegungen sofort die Funktion r' an mit

$$r'(G, k) := (r(G), k).$$

Offenbar erbt r' die polynomielle Zeitbeschränkung von r , die wir in Lemma 63 gezeigt haben. Des Weiteren sind die optimalen Lösungen zu G und $r(G)$ nach Lemma 64 gleich groß. Somit gilt

$$(G, k) \in \text{MCP}_{dec} \iff r'(G, k) \in \text{MNCB}_{dec}.$$

Damit ist die \mathcal{NP} -Härte von MNCB_{dec} nachgewiesen. Mit Satz 53 liegt das Problem außerdem in \mathcal{NP} und ist folglich \mathcal{NP} -vollständig.

◇

Nachdem wir die \mathcal{NP} -Härte nachgewiesen haben, wollen wir jetzt eine Aussage über die Nicht-Approximierbarkeit von MNCB herleiten. Ein Zwischenresultat liefert das folgende Lemma:

Lemma 67 *Es gilt:*

$$\text{MINIMUM CLIQUE PARTITION} \leq_{\mathcal{APX}} \text{MINIMUM NEGATIVE CASEBASE}.$$

Beweis: Mit Hilfe der Reduktionsfunktionen r und t lässt sich eine \mathcal{APX} -Reduktion von MCP auf MNCB durchführen. Sei G ein Graph. Wir haben folgende Aussagen bereits gezeigt:

- (A) Lemma 63 besagt: r und t sind in polynomieller Zeit berechenbar.
- (B) Lemma 60 besagt: r bildet Eingaben für MCP auf Eingaben für MNCB ab:

$$G \in I_{\text{MCP}} \implies r(G) \in I_{\text{MNCB}}.$$

- (C) Lemma 62 besagt: Falls M_-^{min} eine geeignete Lösung zu $r(G)$ in MNCB darstellt, so ist $t(G, M_-^{min})$ eine geeignete Lösung zu G .
- (D) Lemma 65 besagt: Der Approximationsfaktor der mit t konstruierten Cliques-Überdeckung ist gleich demjenigen der Fallmenge M_-^{min} , aus welcher diese konstruiert wurde. Für jedes G aus I_{MCP} und M_-^{min} aus $S_{\text{MNCB}}(r(G))$ gilt also:

$$q_{\text{MCP}}(G, t(G, M_-^{min})) = q_{\text{MNCB}}(r(G), M_-^{min}).$$

Die Aussagen (A) bis (C) entsprechen den Voraussetzungen (1) bis (3) aus Definition 57. Es bleibt noch (4) zu zeigen: Für jedes $q_0 \geq 1$ existiert ein $q'_0 \geq 1$, so dass für jedes $G \in I_{\text{MCP}}$ und jedes $M_-^{\text{min}} \in S_{\text{MNCB}}(r(G))$ mit $\pi_{\text{MNCB}}(r(G), M_-^{\text{min}}) = 1$ gilt:

$$q_{\text{MNCB}}(r(G), M_-^{\text{min}}) \leq q_0 \implies q_{\text{MCP}}(G, t(G, M_-^{\text{min}})) \leq q'_0.$$

Dies folgt aber sofort aus Aussage (D), indem man q'_0 gleich q_0 wählt. Damit ist MCP \mathcal{APX} -reduzierbar auf MNCB.

◇

Diese Eigenschaft verwenden wir jetzt, um das abschließende Nicht-Approximierbarkeits-Resultat für MNCB nachzuweisen.

Satz 68 *Unter der Voraussetzung $\mathcal{ZPP} \neq \mathcal{NP}$ existiert kein $\gamma > 0$, für das MINIMUM NEGATIVE CASEBASE $|M_-|^{(1-\gamma)}$ -approximierbar ist.*

Beweis: Wir werden die bekannten Approximierbarkeits-Eigenschaften von MCP unter Anwendung von Lemma 58 auf MNCB übertragen.

Mit Lemma 67 gilt $\text{MCP} \leq_{\mathcal{APX}} \text{MNCB}$. Lemma 65 wiederum besagt, dass für jeden Graphen G aus I_{MCP} und M_-^{min} aus $S_{\text{MNCB}}(r(G))$ gilt:

$$q_{\text{MCP}}(G, t(G, M_-^{\text{min}})) \leq q_{\text{MNCB}}(r(G), M_-^{\text{min}}).$$

Hiermit sind die Voraussetzungen für die Anwendung von Lemma 58 erfüllt. Mit Fakt 55 ist MCP für kein $\gamma > 0$ bis auf einen Faktor $|V|^{(1-\gamma)}$ -approximierbar, solange nicht $\mathcal{ZPP} = \mathcal{NP}$ gilt. Mit $|V| = |M_-|$ existiert nach Lemma 58 folglich unter derselben Voraussetzung ebenfalls kein $\gamma > 0$, so dass MNCB $|M_-|^{(1-\gamma)}$ -approximierbar ist.

◇

Schon aufgrund dieser Aussage kann MNCB_{dec} offenbar auch nicht in \mathcal{P} sein, denn sonst müsste ein exakter Lösungsalgorithmus existieren, und dieser würde als 1-Approximation die hergeleitete Aussage widerlegen. Dieser Schluss basiert jedoch auf der Voraussetzung $\mathcal{ZPP} \neq \mathcal{NP}$. Die \mathcal{NP} -Härte von MNCB_{dec} gilt jedoch davon unabhängig, wie wir in Satz 66 gezeigt haben.

5.5 Schlussfolgerungen

Wir wollen abschließend versuchen, die Ergebnisse aus diesem Abschnitt in ihrer Bedeutung einzuordnen.

Wir haben mit dem Resultat aus Satz 68 ein starkes Indiz gegeben, dass die PAC-Lernbarkeit polynomiell repräsentierbarer Funktionsklassen sich nicht über die Beschränkung für die Größe des Hypothesenraums nachweisen lässt. Dabei ist die Schwierigkeit, ähnlich wie beim Lernen von 3-Term-DNF-Formeln³, das Finden einer konsistenten Hypothese im Repräsentationsraum. Zwar konstruieren wir auf einfache Weise eine konsistente Hypothese, jedoch können wir nicht garantieren, dass ihre Fallbasengröße unabhängig von der zu lernenden Zielfunktion durch ein Polynom in n beschränkt ist.

Mit Hilfe der in Satz 66 konstruierten Reduktionsfunktion r' kann man immerhin analog zu der in [KV94] angewandten Methode zum Nachweis der “Unlernbarkeit” von 3-Term-DNF-Formeln zeigen, dass das effiziente PAC-Lernen der Menge

$$\mathcal{F} := \{f \mid \text{Es existiert ein } CB := \langle CB_+, CB_- \rangle \text{ mit } |CB_+| = 1 \text{ und } |CB_-| = k\}$$

\mathcal{NP} -hart ist, wenn man \mathcal{F} selbst als Hypothesenklasse verwendet. Für zukünftige Untersuchungen wäre interessant, ob sich dieses Resultat durch eine Anpassung der Reduktionsfunktion auf beliebige polynomiell repräsentierbare Klassen ausdehnen lässt.

Auf der anderen Seite wäre es denkbar, dass sich für bestimmte Teilklassen der von uns ins Visier genommenen Menge aller polynomiell repräsentierbaren Klassen die effiziente PAC-Lernbarkeit auf diese Weise zeigen lässt. Dies gilt beispielsweise für die Menge aller Monome; jedoch ist ein entsprechender Algorithmus nicht wirklich interessant, da sich Monome bekanntlich mit “herkömmlichen” Methoden⁴ recht einfach lernen lassen.

Weiterhin ist Occams Razor nur eine mögliche Beweismethode. Dabei wird unter anderem vorausgesetzt, dass stets eine konsistente Hypothese berechnet wird. Dies ist aber nicht zwingend für einen PAC-Algorithmus, der sowieso mit Fehlern in der resultierenden Hypothese kalkuliert. Denkbar wäre beispielsweise, das Rasiermesser erneut zu zücken, um durch den Konsistenz-Zwang entstandene “unangenehme” Ecken und Kanten einer Hypothese abzuschneiden, in der Hoffnung die Wahrscheinlichkeit von Objekten sei in diesen Bereichen sowieso gering.

Solche und andere Denkansätze, bei denen man sich mit Wahrscheinlichkeiten bestimmter Bereiche des Objektraums direkt auseinandersetzen muss, haben jedoch

³[KV94] Das Lernen von 3-Term-DNF-Formeln ist \mathcal{NP} -hart, wenn man als Hypothesenklasse ebenfalls die Menge aller 3-Term-DNF-Formeln verwendet.

⁴vergleiche Beispiel 5

mit der Komplexität der betrachteten Verbandsstruktur zu kämpfen. Insbesondere die Überlappung der von verschiedenen Fällen klassifizierten Negativbereiche macht eine Fehlerquantifizierung schwierig. Es wäre möglicherweise interessant, solchen Methoden ausführliche Untersuchungen zu widmen, diese würden den Rahmen einer solchen Arbeit jedoch sprengen.

Literatur

- [Aha91] D.W. Aha. Case-based learning algorithms. In *Proceedings of the DAR-PA Case-Based Reasoning Workshop*, pages 147–158. Morgan Kaufmann, Washington D.C., 1991.
- [BC94] D. P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1994.
- [FK98] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. System Sci.*, 57:187–199, 1998.
- [Ger67] H. Gericke. *Theorie der Verbände*. B-I Hochschultaschenbücher-Verlag, 1967.
- [Gil77] J. Gill. Computational complexity of probabilistic turing machines. *SIAM J. Comput.*, 6(4):675–695, Dec. 1977.
- [GJ79] M. Garey and D. Johnson. *Computers and Intractability — A Guide to the Theory of NP-completeness*. Freeman, 1979.
- [GL96] C. Globig and S. Lange. Case-based representability of classes of boolean functions. In *Proceedings of the 12th European Conference on Artificial Intelligence*, pages 117–121. John Wiley and Sons, 1996.
- [Hof93] A. Hoffmann. *Komplexität einer künstlichen Intelligenz*. PhD thesis, Technische Universität Berlin, 1993.
- [KV94] M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, 1994.
- [Mit97] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [Pap94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [Sat98] K. Satoh. Analysis of case-based representability of boolean functions by monotone theory. In *Proceedings of the 9th Conference on Algorithmic Learning Theory, LNAI 1501*, pages 179–190, 1998.
- [SN00] K. Satoh and R. Nakagawa. Discovering critical cases in case-based reasoning. In *Online Proceedings of the Sixth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, USA, 2000.
- [Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

